A Survey of Mathematical Structures for Lunar Networks

Alan Hylton and Robert Short and Jacob Cleveland
NASA Glenn Research Center

Olivia Freides and Zander Memon
American University

Robert Cardona and Robert Green and Justin Curry
University at Albany - State University of New York

Sriram Gopalakrishnan
Université de Bordeaux

Devavrat Vivek Dabke
Princeton University

Brittany Story and Michael Moy
Colorado State University

Brendan Mallery
Tufts University

***Abstract*— To sustain the current and increasing accessibility of space, a scalable communications infrastructure (i.e. the Solar System Internet, SSI) is necessary. The goal of this paper is to begin the discovery of the fundamental underlying mathematical structure of space networks to help the research community harness these structures for algorithm development and optimization. To ensure the applicability of the research, the approaches are considered through the lens of simulated scenarios inspired by the Artemis Back-to-the-Moon mission set for 2024.**

We note that any approach to an SSI must fit under the umbrella of Delay Tolerant Networking (DTN), due to celestial mobility, high link latencies, high variance in link latencies, disconnections, lack of end-to-end paths, and so on. These difficulties are exacerbated by the fact that the underlying structure of a space network is a time-evolving network and may experience multiple discontinuities in its topology.

In this paper we propose several novel approaches to a mathematical foundation for Delay Tolerant Networking Theory that fall outside the traditional scope of temporal network theory. These techniques include methods from Topological Data Analysis, Dynamic Graph Analysis, Applied Algebraic Geometry, Probability Theory, and Game Theory. Some of these methods include tools adapted to the study of dynamic metric spaces, such as zigzag persistent homology and their higher parameter analogs. We find that several of these methods target desired engineering outcomes such as discovery and automatic subnetting. While each approach is theoretical, they are also algorithmic in nature and offer immediate practical applications. The paper concludes with comparisons of the various methods along with suggestions for future work.

1. Introduction

NASA is adopting Delay Tolerant Networking (DTN) as a standardized approach to space networking in order to address several challenges. These include physical realities, such as mobility and propagation delay. They also include network difficulties, such as high time-variance and a potential lack of end-to-end connectivity. Also considered are process challenges, such as scheduling; NASA notes that manual scheduling approaches can take up to five days for a given transmission [1]. While the physics of satellites separated by interplanetary distances cannot be overcome, the resulting system can be studied and turned into a functioning disconnected network.

DTN glues together otherwise disparate network components in an overlay network. This allows the network to carry *bundles*, the primary data unit in a DTN. For example, this could include point-to-point radios, but also existing Transmission Control Protocol (TCP) networks – such as ground stations connected by the Internet. DTN accomplishes this by means of store, carry, and forward action – data are stored upon receipt, carried until a meaningful contact is established, and then forwarded. This notion of *meaningful* can be elusive, as traditional networking theory stops short of the topologically-complex (i.e., "many" connected components) and time-varying graphs inherent in space networks. This paper is part of a series of papers that explore the mathematical foundations of DTNs. This allows better algorithms (such as routing and policy) to be implemented to act on the DTN protocol, for example so that the DTN protocol might know which links are meaningful. See [2][3][4].

After explaining how and where intuition and test cases come from, this paper considers temporal graph theory, an alternative approach to the classical contact graph routing (CGR) pathfinding algorithm. This paper also considers methods to handle the difficulties of dynamic graphs, some applications of computational topology, algebraic modeling of DTNs, and applications of game theory to traffic flows. It is the authors' hope that these new tools and our observations about them enable next steps in DTN research to finally realize a Solar System Internet. Some project ideas are listed in the future work section.

2. Network Simulation Information

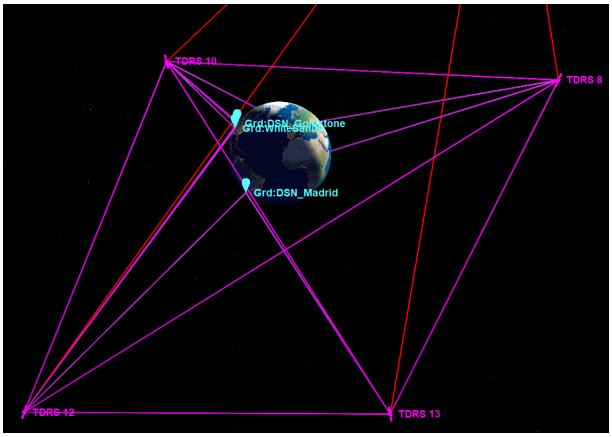


Figure 1. Sample Space Network

Visualization and modeling are crucial to our understanding of delay tolerant networks. Orbital analysis software serves

2022 IEEE Aerospace Conference (AERO) | 978-1-6654-3760-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/AERO53065.2022.9843305

as the foundation on which to apply the mathematical tools discussed here. Throughout this paper, these simulations function to visualize the complexities of a delay tolerant network though the lens of lunar networks; see the example network of Figure 1.

Creating autonomous routing algorithms requires a complex understanding of the assumptions and nuances of a delay tolerant network. Networks in space depend heavily on time, propogation delay, intermittent connections, and hardware limitations. As such, the heterogeneity of a space network only amplifies the need for accurate and accessible visualization techniques.

To accomplish orbital visualization and analysis, the Satellite Orbital Analysis Program (SOAP)[1] was used. SOAP allows users to create and simulate space scenarios including surface and orbital assets, among others, over a specified time frame.

We developed a lunar orbital simulation built from NASA missions to begin modeling a basis for lunar networking. The objects are split up into ground stations on Earth and the Moon, along with future and current satellites. The two Earth ground stations were placed at the Glenn Research Center's main campus in Cleveland, Ohio and the Kennedy Space Center in Merritt Island, Florida. There were four ground stations placed on the Moon: one centered on each pole and the other two centered on the far and near sides relative to Earth. This simulation contained seven satellites: the Lunar Reconnaissance Orbiter (LRO) which launched in 2009; the Gateway station, which is expected to be fully assembled in 2024; and the five Time History of Events and Macroscale Interactions during Substorms (THEMIS) satellites launched in 2007 [6].

Notably, the THEMIS satellites A-E started out orbiting Earth. Then at the start of the Acceleration, Reconnection, Turbulence and Electrodynamics of the Moon's Interaction with the Sun (ARTEMIS) mission, THEMIS B and THEMIS C were repurposed and redirected into lunar orbit in 2010 as ARTEMIS P1 and ARTEMIS P2 respectively [7].

In making the model, each ground station was set by latitude and longitude coordinates. Both NASA centers' coordinates were taken directly from an internet search, and the lunar ground stations' coordinates were chosen to lie directly in the middle of each pole latitudinally and the middle of the equator on both sides (near and far) longitudinally. Each Earth-orbiting THEMIS satellite (A, D, E) was entered into the model through a two-line element (TLE) from Celestrak [8].

The ARTEMIS P1 and P2 satellites were manually constructed to match data from the Planetary Data System (PDS) generated by the Jet Propulsion Laboratory. Both approximations were based on their respective definitive trajectory.bsp file with a configuration set at 2012/11/27 at 00:00:00. The LRO orbital parameters were pulled from the most recent data entry into NASA's catalog denoting the parameters corresponding to the first 50,000 orbits where parameter inputs were specified inside the spreadsheet [9].

Lastly, the Lunar Gateway Near-Rectilinear Halo Orbit was imported through the PDS system from NASA Jet Propulsion Laboratory as a .bsp file [10].

Figure 1 shows a visualization of the simulation from SOAP from a lunar viewpoint. The simulation was color coded into groups of satellites and ground stations. The ARTEMIS satellites are in yellow, THEMIS in orange, GATEWAY in purple, Earth ground stations in pink, lunar ground stations in blue, and the Lunar Reconnaissance Orbiter (LRO) in red.

3. TEMPORAL GRAPH THEORY

Graphs are one of the fundamental mathematical structures used to study computer networks. A finite graph $G = (V, E)$ consists of a finite vertex set V and a finite edge set E, consisting of pairs of vertices, i.e. $E \subseteq V \times V$. If we identify $(i, j) = (j, i)$, then we say a graph is a undirected or simply a "graph;" if the ordering of the vertices matters in a pair, then we say that we have a directed graph or digraph. This abstract structure allows us to model relationships between objects, such as connectivity. In the setting of computer networks, an edge might be a link between two devices.

In this setting, it is natural to study the importance of a device to the network overall. An example of something that measures this importance is *centrality* which ranks nodes based on their importance to connecting the graph. It is also natural to study how one optimizes in a network from an individual, greedy perspective versus from the perspective of the network overall.

Graphs can also be "decorated" by coloring their vertices, adding directions to the edges, or weights indicating the capacity or cost of a connection, for example. Once graphs are decorated, algorithms and analysis can be applied to solve several problems, such as single source shortest path, max flow, or minimum spanning tree, to name a few. There are several well-established tools for solving these problems.

One of the implicit assumptions in the construction of many terrestrial networks is that there exists a "backbone" of the system that will largely go unchanged as time passes. Another assumption is that communications can be passed near instantaneously. The way graphs are used to model these networks for engineering purposes often rely on these assumptions.

Both of these assumptions quickly fall apart in the context of space networks as the assets are subject to orbital mechanics and may lose line of sight or have a nontrivial one-way light time. Thus, when designing a networking system in this setting, one needs to consider more general tools. However, many of the most powerful theorems and algorithms that are useful for networking problems such as max-flow min-cut or Dijkstra's algorithm are built to work with a specific collection of common graph decorations such as directed edges and edge weights. This begs the question: how can one bridge the gap between our system assumptions and the tools which have been successful in past network engineering?

One approach taken by the developers of Contact Graph Routing (CGR) is to construct a graph that represents aspects of our time-evolving network in a way that fits the assumptions of a desired mathematical tool (in this case Dijkstra's algorithm). This approach can be fruitful but also limited as these representations are not the most natural way to think about time-varying systems. Thus, robust analysis of these networks is limited[2]. In light of this, we consider a different collection of graph decorations which more naturally model

[1] SOAP is developed by the Aerospace Corporation [5].

[2] The reader may refer to a mathematical analysis of CGR [11].

the time-varying nature of our system.

The first tool we consider is a time-varying digraph. We define a time-varying digraph as a triple $G = (V, E, T)$, where V is a finite vertex set, $E \subseteq V \times V$ is a set of directed edges, and $T = \{T_e \mid e \in E\}$, where each T_e is a set labeling the edge e. In a continuous time-varying digraph, each $T_e \in T$ is a set of intervals associated to the edge $e \in E$. For a discrete time-varying digraph $T_e \subseteq \mathbb{N}$. In both cases, T_e can be thought of as telling us when each connection is available: in the discrete case, it is available at a discreet time step $a_i \in T_e$, and in the continuous case it is available for each interval $I^i \in T_e$. Note that we allow loops, that is, edges that connect a vertex to itself. Discrete time-varying digraphs, sometimes referred to as time-evolving or temporal digraphs, have been a fruitful subject of study in computer science literature [12][13][14]. However, the authors have found very little literature concerning continuous time-varying digraphs as a subject of study.

Discrete time-varying digraphs

Discrete time-varying digraphs have a discrete set decorating each edge which indicates which times that edge is available. We first consider a way of describing how a time-varying digraph changes over its discrete time steps.

Definition 3.1. Let $G = (V, E, T)$ be a discrete time-varying digraph with times given by T_e for each edge e. Let $\Lambda = \bigcup_{e \in E} T_e$ be ordered so that $\Lambda = \{\lambda_1, \ldots, \lambda_n\}$ and $\lambda_{i-1} < \lambda_i$. Define the *static expansion* to be the graph H with vertex set V_H and edge set E_H such that:

- If $u \in V$, then $u_i \in V_H$ for each $i \in \Lambda \cup \{\lambda_1 - 1\}$.
- If $e = (u, v) \in E$ and $\lambda_i \in T_e$, then $(u_{\lambda_{i-1}}, v_{\lambda_i}) \in E_H$.

An advantage of this representation is that it allows us to think of the vertices of this expanded digraph as images of our original vertices in time [12].

Dynamic Connectivity

Another way to expand a time-varying digraph into a larger structure is to construct a sequence of graphs. The idea is to represent how the network appears at any interval through discrete steps.

Definition 3.2. Let $G = (V, E, T)$ be a discrete time-varying digraph with times given by T_e for each edge e. Let $\Lambda = \bigcup_{e \in E} T_e$ be ordered so that $\Lambda = \{\lambda_1, \ldots, \lambda_n\}$ and $\lambda_{i-1} < \lambda_i$. Define the *digraph sequence* to be the sequence of graphs $\mathcal{G} = \{G_\lambda\}_{\lambda \in \Lambda} = \{(V, E_\lambda)\}_{\lambda \in \Lambda}$ where the edge sets are given by $E_\lambda = \{e \in E \mid \lambda \in T_e\}$.

Note that each digraph in the sequence has the same (identifiable) vertices, but the topology of the digraph is changing. We could augment this definition by including edge weights, node colorings, etc. However, we do not allow for a change in the number or identity of vertices. The length of this sequence could be finite or infinite depending on the context.

In a digraph $G = (V, E)$, we say a *path* P is a sequence of vertices, i.e. $P = (v_{i_0}, v_{i_1}, \ldots, v_{i_k})$, such that for $1 \leq j \leq k$, $(v_{i_{j-1}}, v_{i_j}) \in E$. In other words, a path is a sequence of vertices where each consecutive pair of vertices in the sequence forms an edge in the digraph. P has *length* k. For a sequence of digraphs, we can define an s-journey J^s as a sequence of vertices, i.e. $J^s = (v_{i_0}, v_{i_1}, \ldots, v_{i_k})$, such that for $1 \leq j \leq k$, $(v_{i_{j-1}}, v_{i_j}) \in E_{s+j-1}$.

Given a digraph G, let $\mathcal{P}$ be the set of all finite paths and $\mathcal{P}(i, j)$ be the set of all finite paths where the first vertex is v_i and the last vertex is v_j. We define the shortest path length $SP(i, j)$ as

$$SP(i, j) = \min_{P \in \mathcal{P}(i,j)} length(P).$$

Therefore, we can now define the diameter of G as

$$diameter(G) = \max_{i,j \in |V|} SP(i, j).$$

Notably, $SP(i, j)$ can be ∞, so the diameter can be, too - in particular, this happens when no path exists.

From here, we can define *s-diameter*, which defines a diameter given a particular timestep s. Given a digraph sequence $\mathcal{G}$, let $\mathcal{J}^s$ be the set of all finite s-journeys and $\mathcal{J}^s(i, j)$ be the set of all finite s-journeys where the first vertex is v_i and the last vertex is v_j. We define the shortest journey length $SJ^s(i, j)$ as

$$SJ^s(i, j) = \min_{J^s \in \mathcal{J}^s(i,j)} length(J^s)$$

Therefore, we can now define the s-diameter of $\mathcal{G}$ as

$$diameter^s(\mathcal{G}) = \max_{i,j \in |V|} SJ^s(i, j)$$

Notably, $SJ^s(i, j)$ can be ∞, so the s-diameter can be, too.

We can thus define the following notions of *connectivity*. A graph is *(strongly) connected* if it has finite diameter. In graph theory for directed graphs, it is common to use *strongly connected* if the graph has finite diameter, but we will simply say connected.

A digraph sequence is:

- δ-*disconnected* if the s-diameter is ∞ for all s;
- δ-*weakly connected* if the s-diameter is finite for some s;
- δ-*connected* if the s-diameter is finite for all s;
- and δ-*uniformly connected* if there exists a finite C such that the s-diameter is at most C for all s.

Furthermore, we say a digraph is *non-stranding* if for all vertices there exists an outbound edge. We say a digraph is *holding* if for all vertices there exists an edge to itself. We say a digraph sequence is *non-stranding* if all digraphs in the sequence are non-stranding. And we say a digraph sequence is *holding* if all digraphs in the sequence are holding. Finally, we say a digraph sequence is *fixed* if all digraphs in the sequence are the same, i.e. $\mathcal{G} = \{G_k\}$ and $G_k = G$ for all k. We say that G is the *base graph* of a fixed sequence.

First, note that we have harmonized the static and dynamic graph properties.

Proposition 3.3 (Harmonization of Fixed Sequences)**.** *A fixed digraph sequence $\mathcal{G}$ is connected if and only if the base graph G is connected. Moreover, such a connected graph sequence is δ-uniformly connected, and its uniform bound is the diameter d of the base graph. This bound is in fact tight, i.e. the s-diameter is d for all s.*

Proof. At a high-level, for a fixed digraph sequence, journeys and paths are the same. Thus, all of the definitions for shortest path and shortest journey coincide. We proceed more formally.

First, we show that all s-journeys for all s in the digraph sequence are also paths on the base graph. If a sequence of vertices $(v_0, \ldots, v_k)$ is an s_0-journey for arbitrary timestep s_0, then $1 \leq j \leq k$, $(v_{i_{j-1}}, v_{i_j}) \in E_{s_0+j-1}$ by definition of an s-journey. However, since this is a fixed digraph sequence, we can rewrite this as $(v_{i_{j-1}}, v_{i_j}) \in E_b$. This is exactly the definition of a path on base graph G. Symmetrically, note that every path on G is a valid s-journey for every timestep s.

Now, suppose that $\mathcal{G}$ is connected: this implies that for every timestep s, there exists an s-journey between every pair of vertices. Each s-journey is also a path on G, which means that there is a path from each pair of vertices in G. G is thus connected.

Suppose that G is connected. This implies that there is a path between every pair of vertices. For each timestep s, we know that a path is a valid s-journey. Therefore, there must be an s-journey between every pair of vertices for every s.

Finally, let us inspect the s-diameter. Suppose there is a timestep s_0 such that the s_0-diameter is strictly less than d, the diameter of G. Then, there exists an s_0-journey of length less than d between each pair of vertices. But, since each s_0-journey is a path, then there must exist a path between each pair of vertices of length less than d, which would imply that the diameter of G is less than d, which is a contradiction.

In the other direction, note that if the diameter of G is d, then there exists a path of length at most d between each pair of vertices. Since every path is an s-journey for every timestep s, then there exists an s-journey of length at most d from each pair of vertices for every s. Therefore, there is a uniform bound d on the s-diameter.

□

Next, we note that holding is an important property that links static and dynamic notions of connectivity.

Proposition 3.4 (Uniform Connectivity of Connected Holding Sequences)**.** *A holding digraph sequence $\mathcal{G}$ where each digraph in the sequence is connected is δ-uniformly connected. In particular, the number of vertices n is a uniform bound.*

Proof. Fix a digraph sequence $\mathcal{G}$. We select an arbitrary starting time t_0 and source vertex v_s We will show that there exists an t_0-journey between v_s and every other vertex in our finite vertex set V and each journey has length at most n where $n = |V|$.

To accomplish this, we will show three properties of something called "reachability sets" of the vertex: weak monotonicity, complementary inclusion, and then strong monotonicity. To start, note that for a finite journey of length i, we can write the journey as

$$J = (v_{t_0}, v_{t_1}, \ldots, v_{t_i})$$

We say this journey "reaches" or "ends at" vertex v_{t_i} and "starts at" vertex v_{t_0}. A *reachability set* is a set $U_i \subseteq V$ of vertices that can be reached in an t_0-journey of exactly length i such that the journey starts at v_s. We let $U_0 = \{v_s\}$.

We will first show that for a holding digraph sequence, the sequence of U_i satisfy weak monotonicity, in that

$$U_0 \subseteq U_1 \subseteq \cdots U_n \subseteq \cdots$$

Note that with a holding digraph sequence, U_1 is non-empty, in that it at least includes v_s. Therefore, suppose that vertex $v_u \in U_k$ for some k. Then, $v_u \in U_{k+1}$ since $(v_u, v_u) \in E_{k+s}$ by the holding property. In other words, we can extend a length k journey that reaches v_u to a length $k+1$ journey by appending v_u.

Next, we show that for any digraph sequence of connected digraphs, we have complementary inclusion. In other words, if $V - U_k$ is non-empty, then there exists some vertex $v_c \in U_{k+1}$ such that $v_c \in V - U_k$. To show this, we simply leverage the definition of connected static graphs: suppose $V - U_k$ is non-empty. For some vertex $v_{c-1} \in U_k$ (which in a holding digraph, must be non-empty), there must be an edge in E_k to some vertex $v_c \in V - U_k$; otherwise the graph would not be connected.

Finally, we combine these properties to get strong monotonicity: either $U_k \subset U_{k+1}$ or $U_k = V$. Moreover, if $U_k \subset U_{k+1}$, then $|U_k| + 1 \leq |U_{k+1}|$. In other words, the chain of reachability sets must increase in size by at least 1. Thus, we can conclude that $U_{n-1} = V$. □

Finally, we provide a simple result that shows that the dynamic diameter must change incrementally.

Proposition 3.5 (Non-Stranding Bound on s-diameter)**.** *A non-stranding digraph sequence $\mathcal{G}$ with finite s-diameter has finite t-diameter for all $t \leq s$. Moreover, if the s-diameter is some finite value c, then the $(s-1)$-diameter is at most $c+1$. Finally, note that if the s-diameter is infinite, then the t-diameter is infinite for all $t \geq s$.*

Proof. Proof left to an interested reader. □

4. DIRECTED MULTIGRAPHS

Contact Graph Routing (CGR) is an approach to routing that has been proposed for use in DTN, specifically for networks in space. In this section, we propose a model of time-evolving networks as an alternative to the one currently used in CGR. This model will allow for an improvement to the pathfinding algorithm; we will outline the ideas here and leave details to future work [15]. We begin by introducing the necessary terms from CGR.

Background on Contact Graph Routing

The fundamental idea behind CGR is a particular type of graph called a *contact graph* that is used to describe a time-evolving network over a period of time. A version of Dijkstra's algorithm [16], [17] can be used to find paths through this graph, which correspond to routes through the network. CGR encompasses more than this technique for finding routes through a network; in fact, it establishes a broader strategy for managing lists of routes, selecting a route for given data, and queuing, working within the framework of DTN. However, we will limit our focus to the pathfinding portion of CGR.

Here we introduce the main objects of CGR, following [11]. Given objects A and B in our time-evolving network, we will record the intervals of time during which these objects can communicate and will call these intervals of time *contacts*. A contact from A to B will be written as $C_{A,B}^{t_0,t_1}$, where t_0 and t_1 are the start and end times of the contact. We will refer to A and B as the *source* and *destination* of the contact, written for an arbitrary contact C as $C.src$ and $C.dst$. Similarly, we will write $C.start$ and $C.end$ for the start and end times. In practice, additional information will need to be stored for each contact, including data rate and one-way light time, both averaged over the duration of the contact. To describe our network over a fixed period of time, we will assume we have a record of all contacts that occur: this set of contacts will be called a *contact plan*.

A *contact graph* is a graph that summarizes a contact plan, hence summarizing a time-evolving network. In a contact graph, the contacts of a given contact plan are the vertices — note that this is in sharp contrast with the usual graph-theoretic models of networks, in which the communicating objects are the vertices. An edge between two contacts then represents a period during which data can be stored in an object between being transmitted during the contacts. That is, we place an edge from $C_{A,B}^{t_0,t_1}$ to $C_{D,E}^{t_2,t_3}$ when $B = D$ and $t_3 > t_0$. This produces a graph that contains all the information of the contact plan, in which paths are sequences of contacts along which data could be sent. This allows us to route data through the time-evolving network represented by the contact graph.

In practice, a contact graph will be constructed over the course of a routing algorithm, and a slightly different one will be constructed based on the source S and destination D of the desired route. We will add in a "root contact" $C_{S,S}^{t_{start},\infty}$, available any time after t_{start}, the starting time of the route, and with edges from it to all contacts with source S and ending time after t_{start}. Similarly, we will add a "terminal contact" $C_{D,D}$, with edges from all contacts with destination D to this contact. These are artificial contacts that act as the starting and ending vertices of a routing algorithm; they are necessary because other choices of starting and ending vertices would assume we know the first and last contacts along which data should be sent. Furthermore, because of the way the graph is constructed during the routing algorithm, we may leave out any contacts that cannot be reached by a path from $C_{S,S}^{t_{start},\infty}$, possibly simplifying the graph.

CGR uses a version of Dijkstra's algorithm called the Contact Graph Dijkstra Search [11] to find paths in the contact graph from the root contact to the terminal contact, hence finding a route in a time-evolving network. Rather than optimize for path length, as in the classic version of Dijkstra's algorithm, the CGR algorithm instead optimizes for arrival time. That is, an arrival time is recorded for each contact (this is the earliest time the first byte of data can arrive to the contact's destination), and these arrival times are updated over the course of the algorithm. These updates take into account when each contact is available and their one-way light times. The basic principle of Dijkstra's algorithm applies, since arrival times cannot decrease along a path. This allows the algorithm to find the path with the earliest arrival time to the destination.

A Multigraph Model for CGR

The approach taken by CGR, as described above, is to model a time-evolving network as a static graph-theoretic structure, namely a contact graph. This provides a data structure for use in a routing algorithm. We propose an alternate model of a time-evolving network: a directed multigraph with labeled edges. A modified version of Dijkstra's algorithm on these multigraphs will generally perform better than the previously described version of Dijkstra's algorithm on contact graphs.

Mathematically, directed multigraphs can be described as follows.

Definition 4.1. A *directed multigraph* is a set V of vertices and a set E of edges, along with source and target functions $s, t\colon E \to V$. We will consider directed multigraphs without loops: that is, we will assume $s(e) \neq t(e)$ for each edge e.

The source and target functions specify the direction of each edge; we will imagine information being sent along an edge e from the source $s(e)$ to the target $t(e)$ and draw edges as arrows. Already, we can see that an edge in a multigraph closely resembles our description of a contact. We can model a time-evolving network with a directed multigraph by letting each contact with source A and destination B be an edge from A to B. The notation from CGR of $C.src$ and $C.dst$ replace the source and target functions s and t above. Thus, a directed multigraph model of time-evolving network consists of a set of vertices and a set of contacts that form the edges of the graph; a contact $C_{A,B}^{t_0,t_1}$ is an edge from A to B. The start and end times of a contact can be thought of as labels on an edge, as can the other information associated to each contact, including one-way light time and data rate [11]. These labeled, directed multigraphs provide a natural way to present the data of a contact plan, and like contact graphs, they describe a time-evolving network by a static, graph-theoretic structure.

An Alternate Pathfinding Algorithm for CGR

Just as the original Contact Graph Dijkstra Search can be succinctly described as "Dijkstra's on a contact graph", the alternate algorithm proposed here can be described as a modified version of Dijkstra's algorithm on the multigraphs described above. Dijkstra's algorithm can be adapted to run on a multigraph by simply exploring through each edge connected to the current vertex. In the simple case of minimizing path length in a weighted graph, the algorithm is not particularly interesting, since among the edges between a pair of vertices, the one with minimal weight will always be used. However, in our setting where edges are contacts, not all contacts are available at all times. Like in the Contact Graph Dijkstra Search, we will optimize for arrival time, which means that the arrival time to the current vertex is always known and can be used to determine which contacts exiting the current vertex are available.

The new algorithm is given in pseudocode below. For now, we will illustrate the algorithm, along with intuition for why it should perform better than the original Contact Graph Dijkstra Search, with a simple example. Figure 2 shows a contact graph and a multigraph representing the same time-evolving network, with three vertices A, B, and D. We will consider the problem of finding the path from A to D, starting at time 0, with earliest arrival time. The contact graph has been given the appropriate root and terminal vertices for this problem. We sketch the process of a Dijkstra search through both graphs, assuming each contact has a one-way light time

of 0 for simplicity.

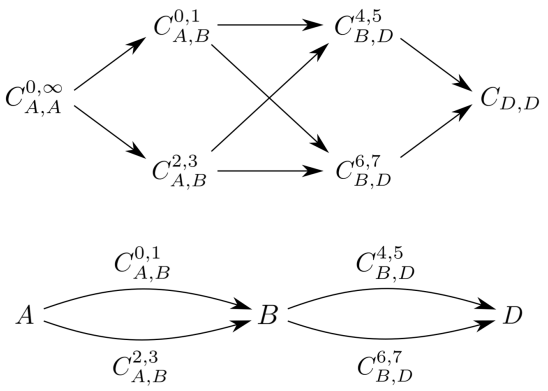


Figure 2. A contact graph (top) and the corresponding multigraph (bottom). The contact graph has root and terminal contacts in order to route from A to D starting at time 0.

For the contact graph, we begin with $C^{0,\infty}_{A,A}$ as the current contact with arrival time 0. We explore contacts $C^{0,1}_{A,B}$ and $C^{2,3}_{A,B}$, assigning them arrival times of 0 and 2 respectively, and mark $C^{0,\infty}_{A,A}$ as visited. Next, $C^{0,1}_{A,B}$ becomes the current contact as it has arrival time earlier than $C^{2,3}_{A,B}$. We explore $C^{4,5}_{B,D}$ and $C^{6,7}_{B,D}$, assigning them arrival times of 4 and 6, and mark $C^{0,1}_{A,B}$ as visited. At this point $C^{2,3}_{A,B}$ has the earliest arrival time of the unvisited contacts, so it becomes the current contact. We examine $C^{4,5}_{B,D}$ and $C^{6,7}_{B,D}$, finding that their arrival times will not be improved by reaching them through $C^{2,3}_{A,B}$, and mark $C^{2,3}_{A,B}$ as visited. Next, $C^{4,5}_{B,D}$ becomes the current contact, finds a route to $C_{D,D}$ with an arrival time of 4, and is marked as visited. Since the arrival time to $C_{D,D}$ is better than the arrival time of the other unvisited contact $C^{6,7}_{B,D}$, the algorithm terminates, and the sequence of contacts that produced the best path is $C^{0,1}_{A,B}, C^{4,5}_{B,D}$.

For the multigraph, we begin with A as the current vertex with arrival time 0. We explore through contacts $C^{0,1}_{A,B}$ and $C^{2,3}_{A,B}$, as both of these are available after time 0. We find that $C^{0,1}_{A,B}$ produces an earlier arrival time, assign B an arrival time of 0, and mark A as visited. Next B becomes the current vertex, and we explore through contacts $C^{4,5}_{B,D}$ and $C^{6,7}_{B,D}$, as both are available after time 0. We find that $C^{4,5}_{B,D}$ produces an earlier arrival time, assign D an arrival time of 4, and mark B as visited. At this point, the destination D has the earliest arrival time of any unvisited vertex (it is the only unvisited vertex), so the algorithm terminates. The result agrees with that for the contact graph: the sequence of contacts that produced the best path is $C^{0,1}_{A,B}, C^{4,5}_{B,D}$.

We can see that these algorithms take similar steps. The step in which $C^{0,\infty}_{A,A}$ was the current contact corresponds to the step in which A was the current vertex, as both explored contacts $C^{0,1}_{A,B}$ and $C^{2,3}_{A,B}$. Similarly, the step in which $C^{0,1}_{A,B}$ was the current contact corresponds to the step in which B was the current vertex. But the final step in the contact graph algorithm, in which $C^{2,3}_{A,B}$ was the current vertex, does not have an analog in the multigraph. In fact, reviewing the steps in the contact graph, we can see that letting $C^{2,3}_{A,B}$ be the current contact is a redundant step, since it can only reach contacts with source B. These contacts were all explored when $C^{0,1}_{A,B}$ was the current contact, and their arrival times can no longer be improved, because $C^{0,1}_{A,B}$ had earlier arrival time than $C^{2,3}_{A,B}$. The steps for the multigraph do not include any analog of this redundant step, because once B has been marked as visited, no contacts with destination B need to be viewed again. This behavior is not specific to this example; the ideas shown here generalize to a proof that a Dijkstra search through a contact graph takes at least as many iterations as a modified Dijkstra search through the corresponding multigraph. The proof will be given in [15].

Pseudocode for the algorithm is provided here. We closely follow the format of the original Contact Graph Dijkstra Search in [11] to allow for an easy comparison. A proof of correctness can be given that mimics the proof for the classic version of Dijkstra's algorithm, relying on the fact that arrival times cannot decrease as a path is extended. Future work may include an alternate version of Yen's algorithm, which is also used by CGR (see [11]).

Algorithm 1 Contact Conditional Dijkstra's Algorithm

Data: Contact plan CP, root vertex v_r, destination vertex v_d, $initial\_time$
Result: Route R from v_r to v_d with best delivery time BDT

construct vertex set V from all sources and destinations of CP
for all $v \in V$, set $v.arr\_time = \infty$, $v.visited = False$, $v.pred = \{\}$
$v_r.arr\_time = initial\_time$
$v_{curr} = v_r$
while true do
    $\mathrm{CCRP(CP, V, v_{curr})}$
    $v_{next} = \mathrm{VSP(V)}$
    if $v_{next} \neq \{\}$ **then**
        $v_{curr} = v_{next}$
    else
        break
    end if
end while
route reconstruction using predecessors to find R
$BDT = v_d.arr\_time$

5. SUMMARY GRAPHS

The time-varying nature of space networks makes it challenging to use traditional graph theory and graph analysis to model delay tolerant networks. Static graphs are simple methods of conceptualizing networks, but often do not fully represent the heterogeneity of delay tolerant networks. Static graphs lack the nuance to convey edges that come in and out, extensive traversal time, and lack of end-to-end connectivity that are characteristic of space networks. On the other hand, while temporal graphs better encapsulate space networks, they are far more complicated to work with and make graph analysis more challenging.

In this section, we propose a novel concept of graph summarization. In essence, we convert a complex, dense data struc-

Algorithm 2 Conditional Contact Review Procedure (CCRP)

Data: CP, V, v_{curr}
Result: Revised V
for contact $C \in CP$ such that $C.src = v_{curr}$ **do**
  if $C.end \leq v_{curr}.arr\_time$ **then**
    skip C
  end if
  if $C.dst.visited$ **then**
    skip C
  end if
  $arr\_time = \max(C.start, v_{curr}.arr\_time) + C.owlt + owlt_{mgn}$
  if $arr\_time < C.dst.arr\_time$ **then**
    $C.dst.arr\_time = arr\_time$
    $C.dst.pred = C$
  end if
end for
$v_{curr}.visited = True$

Algorithm 3 Vertex Selection Procedure (VSP)

Data: V
Result: v_{next}
$v_{next} = \{\}$
$t_{earliest\_arrival} = \infty$
for vertex $v \in V$ **do**
  if $v.visited$ **then**
    skip v
  end if
  if $v.arr\_time \geq v_d.arr\_time$ **then**
    skip v
  end if
  if $v.arr\_time < t_{earliest\_arrival}$ **then**
    $t_{earliest\_arrival} = v.arr\_time$
    $v_{next} = v$
  end if
end for

ture (that of a digraph sequence) into a compressed representation of a single weighted digraph. Summarization captures essential characteristics of a digraph sequence, without the overwhelming storage cost of maintaining the full sequence. In addition, summarization can extract salient information that may not be so obvious from the full sequence.

We will loosely use the term "summary graph" to refer to a weighted digraph that arises from a "summarization" of a graph sequence. As from before, we assume the definition of a digraph and that of a discrete-time digraph sequence.

Edge Weighting and Completeness—We can also define the concept of an "edge weight" that assigns a *weight* to each edge. More precisely, we write a digraph $G = (V, E, w)$, where

$$w : E \to \mathbb{R}_{\geq 0} \cup \{+\infty\}.$$

This weight could represent a variety of things for a static (di)graph. Introductory examples of edge-weighting techniques include, but are not limited to:

1. The amount of time required to traverse the edge
2. The capacity of the edge
3. The cost of traversing the edge
4. The percentage of time in which the edge is disconnected in a given time interval
5. The percentage of time in which the edge is connected in a given time interval

Notably, we have not placed any conditions so far on the weight function, other than that it be a non-negative real number or positive infinity. By convention, though, depending on what the edge weight represents, we may "extend" the domain of the weight function to be defined over all pairings $V \times V$ and assign any pair $(i, j) \notin E$ the weight 0 or $+\infty$ depending on context. In this sense, we will consider each (di)graph in a sequence to be a complete (di)graph to make certain definitions easier to wrangle.

Finally, we must harmonize the definition of the weight function with a digraph sequence, as it is not immediately obvious how this definition would extend to a graph sequence. We define it as follows

$$\mathcal{G} = (G_t = (V, E_t)_{t \in \mathbb{T}}, w)$$

where

$$w : \mathbb{V} \times \mathbb{V} \times \mathbb{T} \to \mathbb{R}_{\geq 0}.$$

In other words, the weight function maps from a pair of vertices and the time-indexing set to a real-number. In our notation, we will write $w_e : \mathbb{T} \to \mathbb{R}_{\geq 0}$ to refer to the weight function of a particular pair of vertices $e = (i, j)$.

We finally impose one additional condition: that w_e be integrable with respect to the standard measure of the time-indexing set.

Example: Traversal-Time DiGraph Journey Summarization with Traversal Time—We already defined a few simple methods of attributing edge weights above. In this section we walk through a more complex example of defining edge weights.

Here, we assume that the weight function on the digraph sequence defines the "traversal time" of a particular edge, i.e. how many units of time are required to traverse the edge. For now, we only consider continuous-time sequences, but our definitions could be easily adapted to the discrete-time case with a bit of care. Here, by convention, traversal time is $+\infty$ if there is no edge; therefore, we can assume our digraph sequence is complete.

We define the *velocity* as $\nu_{(i,j)}(t) = \dfrac{1}{w_{(i,j)}(t)}$. We denote the velocity to be $+\infty$ if $w = 0$ and $\nu = 0$ if $w = +\infty$. Therefore, we can define the notion of s-traversal time of an edge ($tt^s((i, j))$) as follows:

Define the s-traversal completion set

$$tcs^s((i, j)) = \left\{ x : \int_s^x \nu(t)dt \geq 1 \right\}$$

with the convention that $\displaystyle\int_y^y (+\infty)dt = +\infty$. If $tcs^s((i, j))$ is empty, then $tt^s((i, j)) = +\infty$; otherwise,

$$tt^s((i, j)) = \inf tcs^s((i, j)) - s.$$

What does this definition accomplish? It captures the amount of time (in some time unit) it takes to traverse one whole edge given that the traversal velocity is given by ν. We define it through the $\inf$ of a set to handle some analytic issues with $+\infty$. As a helper, we define the s-traversal end time of an edge $tet^s((i, j)) = \inf tcs^s((i, j))$, which is defined to be $+\infty$ if $tcs^s((i, j))$ is empty.

We will, therefore, define the s-traversal time of a path (using all of the previous definitions of a path) with the following recursion:

$$\begin{aligned} tt^s(P)_1 &= tt^s(e_1) \\ tet^s(P)_1 &= tet^s(e_1) \\ tt^s(P)_2 &= tt^{tet^s(P)_1}(e_2) \\ tet^s(P)_2 &= tet^s(P)_1 + tt^s(P)_2 \\ &\vdots \\ tt^s(P)_k &= tt^{tet^s(P)_{k-1}}(e_k) \\ tet^s(P)_k &= tet^s(P)_{k-1} + tt^s(P)_k \end{aligned}$$

with the convention that if tt or tet reaches $+\infty$ at any step, then so do the rest. We can write $tt^s(P) = tt^s(P)_k$.

Thus, we can define the notion of a shortest path $SP^t(i,j)$ as the set of shortest paths minimum cost with respect to s-traversal time and that have s-traversal time. This set could be empty. We can, therefore, use the same definitions as before of ω, i.e. shortest path participation.

As such, we have developed a model for defining summarization functions along edge weights, where edge weights can represent higher-level changes about the network structure and connectivity. In instances where the complexity of time-varying graphs proves too computationally heavy for graph analysis, summary graphs could serve as a bridge from static graphs to temporal graphs.

6. Zigzag Persistent Homology and Applied Topology

As discussed in earlier sections, graphs have long been used as a tool to model communication networks. One weakness all graphical models share is computational intensity. As the number of communication devices in the network increases, the computational complexity drastically increases. Even worse, the number of space communication devices increases by the day [18]. As such, we need tools to reduce the computational complexity of graph-dependent network algorithms.

One proposal to reduce computational complexity involves subnetworking. Subnetworking is where certain nodes are grouped together based on different factors, such as continuous connectivity or far removal from another network system. But for any subnetwork grouping, one can ask whether this network is "close enough" to this other system. Zigzag persistence is a tool from algebraic topology that captures the characteristics of a space as it evolves over time. In the context of space networking, zigzag persistence tells us how the number of connected components changes with respect to time. Further, there exists a measure called the bottleneck distance which provides a way to measure distance between two zigzag persistence diagrams. The goal is to show that a "good" subnetworking decision would correspond to a small bottleneck distance.

Introduction to Zigzag Persistent Homology

In this section we explore zigzag persistent homology, something which allows us to track network connectivity changes. Zigzag persistent homology is based off of standard persistent homology, a tool from algebraic topology that is used to study features of topological spaces. The main differences between zigzag and standard persistence stems from the inclusion maps; in zigzag persistence, the inclusion maps are allowed to go either direction whereas in standard persistence, they are restricted to a single direction. This difference allows us to consider changes in connectivity as time elapses. First, we define all of terms needed to apply zigzag persistence to a space network. Then, we look at two different examples of space networks and discuss how zigzag persistence can provide information about potential subnetwork groupings.

Let X_i be a topological space. We define a *zigzag sequence of topological spaces* as

$$X_1 \leftrightarrow X_2 \leftrightarrow \ldots \leftrightarrow X_n$$

where $\leftrightarrow$ is an inclusion map that either maps $X_i \rightarrow X_{i+1}$ or $X_i \leftarrow X_{i+1}$. Recall, the homology functor (with coefficients in a field) is a functor that maps topological spaces to vector spaces. Hence, the *zigzag module* of a zigzag sequence of topological spaces is

$$H_p(X_1) \leftrightarrow H_p(X_2) \leftrightarrow \ldots \leftrightarrow H_p(X_n).$$

We can use zigzag persistence (with $\mathbb{Z}/2\mathbb{Z}$ coefficients) to understand the structure and features of a topological space. For example, $H_1(X_i)$ describes how many 1-dimensional holes are in X_i. In the example below, we see that a 1-dimensional hole is born at $t = 2$ and dies at $t = 3$. We will use this tool as a way to understand the structure of space networks and use it to determine feasible subnetworking techniques.

Let G_t be a graph with vertices V_t and edges E_t at time t. We define a *time-varying graph* G to be a sequence of graphs, $G_1, G_2, \ldots, G_n$ that changes over time t. For this paper, we will first assume that our vertices do not change. Thus for any time step t, we have a graph $G_t(V, E_t)$. The most natural way to model a communication network is to treat satellites, ground stations, and other members of the network as the vertices and treat contacts between members of the network as edges. Thus, for any G_t, this forms a simplicial complex. From this simplicial complex, for a given time interval $[1, n]$, let N be the zigzag sequence

$$G_1(V, E_1) \leftrightarrow G_2(V, E_2) \leftrightarrow \ldots \leftrightarrow G_n(V, E_n).$$

Note, each arrow represents the inclusion map between two graphs. This arrow will change direction based on the direction of the inclusion. Then, we can apply the homology functor to our zigzag sequence to get

$$H_p(G_1(V, E_1)) \leftrightarrow H_p(G_2(V, E_2)) \leftrightarrow \ldots \leftrightarrow H_p(G_n(V, E_n)).$$

Therefore, we can calculate the p-th homology for our zigzag sequence of graphs. Note, for $p \geq 2$, the p-th homology of a graph will be zero, so we are specifically interested in $p = 0, 1$.

Zigzag persistence of a simple network

A basic network example is as follows. Consider a network with 3 nodes, $V = \{a, b, c\}$. Suppose connection ab is born at time $t = 0$ and dies at $t = 3$, connection bc is born at time $t = 1$ and dies at $t = 3$, and connection ac is born at time $t = 2$ and never dies. Figure 3 is a pictorial representation of these graphs.

Hence, our zigzag sequence is given in Figure 3.

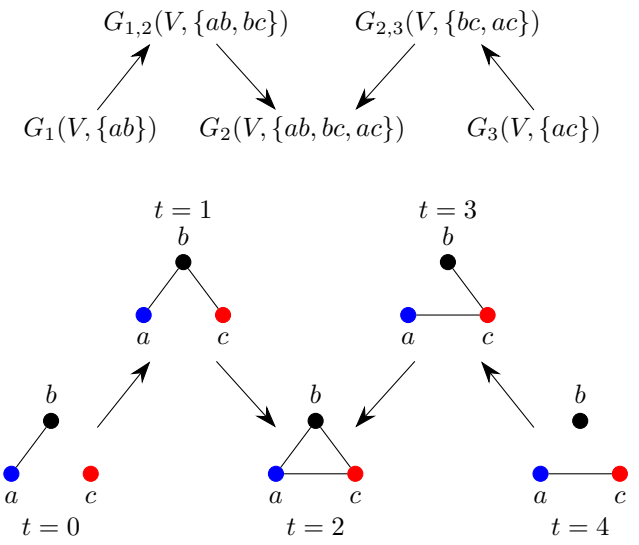


Figure 3. The zigzag sequence for the triangle example.

To find the zigzag persistence barcodes, we first calculate the persistence barcode at $t = 0$. We have two components born at $t = 0$, vertex c and the vertices and edge needed for edge ab which corresponds to the birth of two 0-dimensional bars. At time $t = 1$, the 0-dimensional bar corresponding to c dies. Then at $t = 2$, a one-dimensional hole is born which corresponds to the birth of a one-dimensional bar. Time $t = 3$ shows the death of the 1-dimensional bar and finally at $t = 4$, a 0-dimensional bar is born that corresponds to node b. In summary our barcode consists of 4 bars, 3 in dimension 0 and 1 in dimension 1 with birth and death times, $(0, \infty)$ $(0, 1)$, and $(4, \infty)$ for the 0-dimensional bars and $(2, 3)$ for the 1-dimensional bar.

As an extension of this simple case, consider the network formed by the Lunar Reconnaissance Orbiter (LRO), the Mars Reconnaissance Orbiter (MRO), and the deep space network ground station located at Canberra, Australia. Using SOAP, the modeling program introduced in Section 2, we generate the connection times for this network over one day. In this example, we have 20 connections that take place at different times throughout the day.

Note, calculating the zigzag persistence by hand for even this space network would be tedious. Instead, we use the software package Dionysus [19], which calculates the 0-dimensional and 1-dimensional homology for network. See Table 4 for the small example birth and death times of each 0 and 1 dimensional homology bar. Observing this data, we can conclude that there are no immediately obvious subnetwork groupings, because there are no 1-dimensional homology bars that last for a long period of time. Intuitively, this follows from the size of our network.

Calculating the zigzag persistence summarizes the connectivity information of the network. The 0-dimensional homology tells us how often we have a communication device that is disconnected from the network. Further, the 1-dimensional homology captures when all communication devices are connected. This information can be used to group satellites by either continuous connectivity or by location. This example is too small to observe the behavior we want to use to split our network into subnetworks. Hence, we consider a larger example.

(Birth, Death)	(Birth,Death) (con't)
0-dim	**0-dim**
(194.984, 3980.21)	(54650.5, 57151.6)
(7001.92, 10870.5)	(56556.1, 59102.1)
(0, 17688.8)	(61457.3, 63958.7)
(13808.9, 17760.7)	(61457.4, 65992.4)
(22043.8, 24552.3)	(68264.2, 70765.7)
(20615.8, 24650.9)	(68264.3, 72882.6)
(28947.8, 31441.6)	(75071.1, 77572.8)
(27422.7, 31541.2)	(75071.3, 79772.9)
(35851.1, 38331.2)	(81878.2, inf)
(34229.7, 38431.4)	(81878.1, inf)
(41036.6, 45170.6)	(0 ,inf)
(42753.4, 45219.3)	**1-dim**
(47843.5, 50344.6)	(45321.7, 47843.4)
(49654.9, 52104.2)	(52211.9, 54650.3)

Figure 4. Zigzag persistence for the simple network example.

Example of a larger network

To test this concept, we chose a selection of satellites and ground stations that intuitively lend themselves to subnetworking. Consider the network with the MRO, LRO, TDRS 8, TDRS 10, TDRS 12, TDRS 13, Canberra, Madrid, Goldstone, White Sands, and Guam, see Figure 5. We use SOAP to model the simulated network; see Figure 5 for the network connectivity for a single time step.

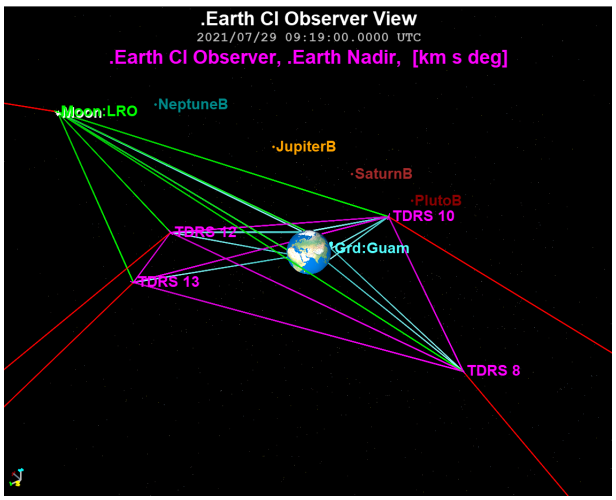


Figure 5. A picture of the larger example network from the SOAP simulation. Note, connections are denoted by a line between two devices.

Intuitively, it might make sense to group the four TDRS satellites together, as they are always connected. Alternatively, we could group each TDRS with the ground station it is closest to. There might be additional groupings not listed, but zigzag persistence might provide insight as to what those groupings are. Over the course of 1 day, the zigzag persistence diagram has 110 bars which captures the connectivity of the network for that day. Of note, there are eight 1-dimensional holes that live for the entire day. Hence, this tells us that some subsection of satellites are always connected. We can use this

information to construct a new network where those satellites are represented by a single node. Further work will be done to test the different subnetworking options by recalculating the zigzag persistence of the new network model and computing the bottleneck distance between the two networks. Please see Section 9 for more detail.

7. Graph Varieties and Applied Algebraic Geometry

In [3], the authors describe a way to realize Dijkstra's algorithm for pathfinding in a graph G as the ability of local sections of a suitable sheaf $\mathcal{F}$ to lift to global sections. In [20], a general procedure is described to construct from a graph G a quasiprojective scheme $\chi(G)$ that in some sense parametrizes the various embeddings of G in $\mathbb{P}^2$.

The various embeddings of a graph G in $\mathbb{P}^2$ are represented as points of $\chi(G)$. $\chi(G)$ is a closed subscheme of $\mathbb{P}^2$ endowed with the Zariski topology. As points are not open in the Zariski topology on $\chi(G)$ in general, naïevly attempting to place a sheaf $\mathcal{F}$ on $\chi(G)$ that models the pathfinding sheaf described in [3] fails. In essence, the Zariski topology is too coarse to encode useful topological information. To correct this failing with a view towards constructing a sheaf on $\chi(G)$ that encodes the same data as that of the pathfinding sheaf on G itself, we introduce the notion of the discrete site on $\chi(G)$.

Graph Varieties

In this section, we closely follow [20], summarizing the parts which are relevant and necessary for our application.

Fix, once and for all, an algebraically closed field k. Note that k can be replaced by an arbitrary commutative ring without substantial change to the following summarized theory. Throughout this section, let $G = (V, E)$ be a graph, with $\#V = n$ and $\#E = r$. We define

$$\mathrm{Gr}(G) = \prod_{v \in V(G)} \mathbb{P}^2 \times \prod_{e \in E(G)} \widehat{\mathbb{P}^2}$$

The Zariski topology on $(\mathbb{P}^2)^n \times (\mathbb{P}^2)^r$ induces a topology on $\mathrm{Gr}(G)$. For a point $P \in \mathrm{Gr}(G)$ and a vertex $v \in V(G)$ (resp. an edge $e \in E(G)$), we write $P(v)$ (resp. $P(e)$) for the image of P under the projection onto the coordinate corresponding to the vertex v (resp. edge e). $P \in \mathrm{Gr}(G)$ is called a *picture* of G if, for all $v \in V$, $e \in E$,

$$v \in e \implies P(v) \in P(e) \qquad (*)$$

As the condition $(*)$ can be described in terms of Plücker coordinates on $\chi(G)$, it is Zariski closed in $\mathrm{Gr}(G)$. In general, the picture space $\chi(G)$ need not be Zariski connected. We do, however, have the following useful and easily verified fact:

Proposition 7.1. *If $G_1, \ldots, G_s$ are the connected components of G, then*

$$\chi(G) \cong \chi(G_1) \times \cdots \times \chi(G_s)$$

By virtue of (7.1), when attempting to compute explicit equations that cut out the picture space of $\chi(G)$, we may assume without loss of generality that G is connected, carrying out the necessary computations on connected components and taking products at the end if not.

Let U_0 be the standard affine open subset of $\mathbb{P}^2$ defined by

$$\begin{aligned} U_0 &:= \{[a_0 : a_1 : a_2] : a_0 \in k^*, a_1, a_2 \in k\} \\ &= \{[1 : a_1 : a_2] : a_1, a_2 \in k\} \end{aligned}$$

The *affine picture space* $\widetilde{\chi}(G)$ of G is $\widetilde{\chi}(G) := \chi(G) \cap U_0$.

The Equations Defining $\widetilde{\chi}(G)$

Once again we closely follow the general theory described in [20], summarizing the portions important to our application.

$\widetilde{\chi}(G)$ is defined, as a subvariety of

$$\mathbb{A}^{2n+2r} = \mathrm{Spec}\, A_G$$

where

$$A_G = k[\{x_v, y_v, m_e : v \in V, e \in E\}]$$

by

$$V(I_e)_{e \in E, e=[v,w]} \qquad (1)$$

where

$$I_e = (y_v - m_e(x_v - x_w))$$

In essence, $\widetilde{\chi}(G)$ parametrizes the various possible embeddings of G in $\mathbb{P}^2$.

Moreover, if $P = (v_1, \ldots, v_s, v_1)$ is a polygon in G, and $e_i = \{v_i, v_i + 1\}$ for $i = 1, \ldots, s$, then the following equation, which represents traversal of P along each edge

$$L(P) = \sum_{i=1}^{s} m_{e_i}(x_{v_i} - x_{v_{i+1}}) = 0 \qquad (2)$$

must be satisfied. We can therefore take

$$R_G = k[\{m_e, x_v : v \in V, e \in E\}],$$

and write $\widetilde{\chi}(G) = \mathbb{A}^1 \times X$ where

$$X = \mathrm{Spec}\, R_G/(L(P))_P.$$

Computing $L(P)$

In this section, we give a brief description of the algorithm described in [20] to compute $L(P)$. We also provide an explicit example of such a computation, as well as code written in Macaulay2 that, given a directed graph, produces the polynomials $L(P)$.

In order to compute the $L(P)$, we simply need to determine all of the polygons in G.

For each edge $e \in E(G)$, choose an orientation of e arbitrarily. For $e = [v, w] \in E$ a (now oriented) edge, define

$$x_e = x_w - x_v$$

Let $C = \mathbb{Z}[E(G)]/\sim$, where $[v, w] \sim -[w, v]$. Let

$$Z = \left\{ \sum_e c_e e \in C : \sum_e c_e x_e = 0 \right\}$$

be the submodule of C generated by the cycles associated to polygons $P = (v_1, \ldots, v_s, v_1)$:

$$z(P) = \sum_{i=1}^{s} [v_i, v_{i+1}].$$

Let T be any spanning tree of G, and set $S = E \setminus T$. For any edge $e = [v, w] \in S$, the set $T \cup \{e\}$ contains a unique polygon $P_T(e) = (v_1 = v, \ldots, v_s = w, v)$ which in turn gives rise to a cycle

$$z_T(e) = \left(\sum_{f \in T} c_{e,f}^T f \right) - e$$

where the coefficients $c_{e,f}^T \in \{0, 1, -1\}$. The coefficients $c_{e,f}^T$ can be thought of as describing how to traverse the edge $f \in T$ as we trace out the polygon $P_T(e)$.

We can define an injective map of $\mathbb{Z}$-modules

$$[v, w] \mapsto m_{v,w}(x_v - x_w) : C \to R'_G$$

that sends Z to the polynomials $L(P)$. It follows that upon fixing any spanning tree T of G, we obtain an ideal $(L(P_T(e)) : e \in S)$ that defines $\widetilde{\chi}(G)$.

Explicitly, if $e = [v, w] \in S$, then

$$L(P_T(e)) = \sum_{f \in T} c_{e,f}^T (m_e - m_f) x_f.$$

The short calculation required to obtain this equality can be found in [20].

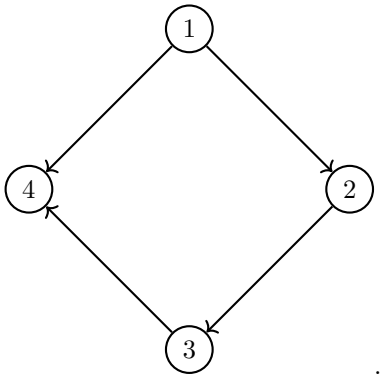
The ideal $(L(P_T)(e) : e \in S)$ that cuts out $\widetilde{\chi}(G)$ can then be conveniently written as a matrix equation:

$$M_T X_T = 0$$

where

$$M_T = [c_{e,f}^T (m_e - m_f)]_{e \in S, f \in T}, \qquad {}^t(X_T) = [x_f]_{f \in T}$$

Example 7.2. Let $G = C_4$ be the cycle graph on four vertices. We fix the following (arbitrary) orientation of the edges $E(G)$:



.

Let T be the spanning tree defined by $T := \{[1, 2], [2, 3], [3, 4]\}$. Then $S = E(G) \setminus T = \{[1, 4]\}$. Taking $T \cup \{[1, 4]\}$ gives the unique polygon

$$P_T([1, 4]) = (1, 2, 3, 4, 1) = [1, 2] + [2, 3] + [3, 4] - [1, 4].$$

It follows that the matrix $C_T = (c_{e,f}^T)_{e \in S, f \in T}$ is

$$\begin{array}{c} \\ [1,4] \end{array} \begin{array}{c} \begin{array}{ccc} [1,2] & [2,3] & [3,4] \end{array} \\ \begin{pmatrix} 1 & 1 & -1 \end{pmatrix} \end{array}.$$

We can read off the matrix M_T directly from this:

$$M_T = \begin{pmatrix} m_{14} - m_{12} & m_{14} - m_{23} & m_{34} - m_{14} \end{pmatrix}.$$

Finally,

$$\begin{aligned} M_T X_T = &(m_{14} - m_{12})(x_2 - x_1) \\ &+ (m_{14} - m_{23})(x_3 - x_2) \\ &+ (m_{34} - m_{14})(x_4 - x_3) \end{aligned}$$

so we see that

$$\widetilde{\chi}(G) = \operatorname{Spec} \frac{k[m_{12}, m_{23}, m_{34}, m_{14}, x_1, x_2, x_3, x_4]}{(M_T X_T)}.$$

In this example, the orientation we chose was naturally motivated by our numbering of the vertices of G. That is, given an undirected edge $e = [v, w] = [w, v]$ between two vertices in $V(G)$, we chose to direct this edge as $[v, w]$ if $v < w$ and $[w, v]$ otherwise. In applications to space networking, there need not be such a canonical choice of orientation of edges. As choosing a different orientation of an edge corresponds to multiplication by a unit in the ideal $(M_T X_T)$ defining $\widetilde{\chi}(G)$ however, we can simply choose orientations of edges at random without affecting the ideal that defines the picture space of our network.

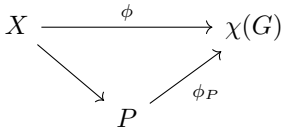
Code to generate the affine scheme $\widetilde{\chi}(G)$, with $k = \mathbb{C}$ is available for download[3]. It relies on [21] and [22].

Sheaves on $\mathbf{Sch}_{(disc)}$

The general theory of sites and Grothendieck topologies is concisely summarized in [23].

Let $\mathcal{S} = \mathbf{Sch}$ be the category of all schemes. That is, the category whose objects are schemes X and whose morphisms are morphisms of schemes. We endow this category with the discrete Grothendieck topology. Under this topology, any sieve[4] on any object is covering. By abuse of nation, we will call the resulting site $\mathcal{S}$.

For any picture (i.e. point) $P \in \chi(G)$, we can consider P to be a closed subscheme of $\chi(G)$. The collection of all pictures in $\chi(G)$ generates a sieve on the object $\chi(G)$. Indeed, $P \in \mathrm{Ob}(\mathcal{S}/\chi(\mathcal{G}))$ since each picture P has a natural inclusion morphism $P \hookrightarrow \chi(G)$, and we can take $\mathcal{U}$ to be the sieve on $\chi(G)$ generated by the collection of morphisms $\{\phi_P : P \hookrightarrow \chi(G)\}$. Explicitly, $\mathcal{U}$ is the collection of morphisms $\phi : X \to \chi(G)$ in $\mathcal{S}$ that admit a factorization:



for some $\phi_P : P \hookrightarrow \chi(G)$.

[3]It can be found at `https://github.com/SriramGDev/graph-varieties`

[4]The reader is referred to [24] for details.

By definition, a presheaf $\mathcal{F}$ on $\mathcal{S}$ is a contravariant functor $\mathcal{F} : \mathcal{S} \to \mathbf{Set}$. Again by definition, the $\mathcal{U}$-local section data is the data of, for each morphism $X \to \chi(G)$ in $\mathcal{U}$, a section $s_X \in \mathcal{F}(\mathcal{X})$ such that for all morphisms $Y \to X$ in $\mathcal{S}$, $s_X|_Y = s_Y$. In particular, this allows us to associate to each inclusion $P \hookrightarrow \chi(G)$ a set. More generally, we can work with C-valued presheaves and sheaves for some arbitrary category C as opposed to $\mathbf{Set}$ in order to associate to each picture $P \in \chi(G)$ an object $\mathcal{F}(\mathcal{P}) \in \mathcal{C}$.

8. GAME THEORETIC NETWORKING

While game theory has found success in elucidating terrestrial network interactions, little research has been done to implement game-theoretical approaches on delay tolerant networks. As space communications becomes increasingly complex, the need for autonomous or semi-autonomous decision-making becomes apparent.

Game theory provides a foundation for how decision-making could function in a non-cooperative game with incomplete information. Under a non-cooperative game, individual actors in a space network would be able to make decisions that maximize their own payoffs or minimize their individual transmission cost without losing optimal transmission across the entire network. In [25] the authors introduce the notion of network games with incomplete information, wherein players have an understanding of the size of the network, but not of the individual connections made between players. This is especially relevant to delay tolerant networks, in which vertices are countable, but connections are not consistently present.

It is worth noting that while cooperative games provide the strongest foundation for network optimization, creating a cooperative game across an entire delay tolerant network lacks feasibility. Because space networks are particularly heterogeneous and variable, creating a global cooperative game proves too challenging. On a time scale of a given transmission lifespan, the network might not be able to share information back and forth to cooperate end-to-end, rendering any algorithm that relies on complete information useless. There is a potential for cooperative games within the sub-networks described in Section 9. This would create a grouping of smaller, cooperative games to form a larger space network. However, instances of transmission from one sub-network to another would require another layer of decision-making structure on top of the given sub-networks. Thus, there is potential for a non-cooperative game on cooperative sub-networks.

We can define a game as a graph $G = (V, E)$ where players are represented by vertices in a directed graph and connections are represented as time-stamped edges between the vertices. In game theory, the *Nash equilibrium* is a set of strategies for each player such that no player unilaterally would deviate from their initial strategy, even after witnessing other players' choices.

If we are able to frame a space network as a game, then we can utilize the concept of a Nash Equilibrium to measure how individual player's path optimization translates into an equilibrium for the network. A Nash Equilibrium occurs in a non-cooperative game with incomplete information as described above. We define a Nash Equilibrium as the point in a game wherein no individual player would rationally make a unilateral move away from equilibrium. Thus, the network reaches a stable equilibrium without consistent communication between players.

It is worth noting that Nash Equilibrium does not always correspond to the socially optimal outcome. A socially optimal outcome sums the costs (or payoffs) of the entire network and can be derived using communication between players and cooperation towards a network-wide optimal outcome. However, there are instances where a Nash Equilibrium in an incomplete, non-cooperative game is identical to the socially optimal network distribution. In [26] the authors demonstrate that a Nash Equilibrium designed to achieve 100% connection across the network while minimizing the number of edges produces a social optimum, that is the Nash Equilibrium is the socially optimal outcome. If we can show that a Nash Equilibrium corresponds to a socially optimal, network-wide outcome, then we can suggest that communication between nodes in a network is not always necessary, therefore significantly reducing communication burdens. Even if there is a substantial gap between a Nash Equilibrium and a socially optimal equilibrium, we can still use the Nash Equilibrium as a metric for the benefits produced from communication between nodes.

Additionally it is possible that a heterogeneous space network need not utilize exclusively Nash Equilibrium or a more traditional cooperative socially optimal outcome. In theory, categorizing types of transmission could be useful in deciding instances where a Nash Equilibrium would be a sufficient foundation for decision making or where complete information is needed for end-to-end transmission.

Game Theoretical Solutions for Congestion

There is also potential for a game-theoretical approach to congestion-based transmission, where traversal time across a given edge is a function of the number of bundles transmitted across the edge. In instances where traversal time is not constant, modeling techniques that allow for adaptability with incomplete information are especially important.

The graph below in Figure 6 is an example game with 6 nodes. There is one source node (A) and one sink node (F). We assume that there are two paths bundles can take to get from source to sink: A-B-C-F and A-D-E-F. We take the edge weights to be traversal time.

Note that both paths have a congestion function, where the traversal time is a function of the number of bundles being sent along that edge. In this example we assume that we must transmit 100 bundles from A to F, and that no data can be temporarily stored at the vertices along the way.

In this example we can think of each player as a bundle, and we can assume that each bundle wants to take the path that will minimize their individual traversal time.

To find a Nash Equilibrium we need to determine the number of bundles that should traverse the upper and lower paths respectively such that no bundle would decrease their traversal time by switching to the other bundle.

Because both upper and lower paths have equivalent traversal times for their non-congestion based edges, 10 seconds and 20 seconds respectively, we can just focus our attention on the edges where traversal time is a congestion function,

$$x/50 = y/40,$$

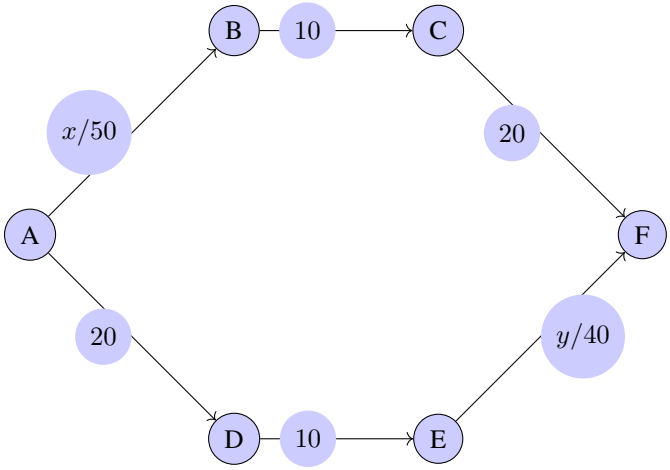


Figure 6. Example game

and a Nash Equilibrium occurs when

$$x = 56, y = 44.$$

Thus the traversal time for an individual bundle along the upper path is

$$(56/50) + 10 + 20(1.12 + 10 + 20) = 31.12,$$

and the traversal time for an individual bundle along the lower path is

$$20 + 10 + (44/40)20 + 10 + (1.1) = 31.1.$$

The reason that this solution is a Nash Equilibrium is because any bundle that switches paths would incur a longer traversal time. For example, if a bundle from the upper path switches to the lower path, the congestion function is now:

$$20 + 10 + (45/40) = 30 + 1.125 = 31.125,$$

and thus the bundle would only be increasing the traversal time upon switching paths.

9. Conclusion and Future Work

In this paper, we have presented several topics that we hope can contribute to a firmer mathematical foundation for Delay Tolerant Networking. Here we summarize the different approaches and how they relate to foundational networking aspects. Then, we dive deeper into how this research may continue with a future works portion for each topic.

Putting it all together

Much like the layers of the OSI model represent different layers of abstraction, our mathematical models represent different layers of abstraction in network understanding. At its core, a network is built on the connections present, and we present several different methods for modeling connectivity in a time-evolving network. Each comes from graph theory, but each presents its own approach with corresponding pros and cons.

- Temporal graph theory provides precise connection information for time-evolving networks with a variety of representations. Different representations can be better or worse as networks grow in size.
- Directed multigraphs are a fruitful data structure for routing algorithms, as demonstrated by the updated pathfinding algorithm for CGR presented in Section 4.
- Summary graphs provide useful statistics for comparing connection availability, and may provide bridges from temporal to more traditional graph theoretic results using centrality measures. However, summary graphs represent a potential for information loss, such as ordering of intervals, that could be extremely relevant for different applications.

Once connectivity is established, being able to analyze connectivity structures and discover strong subnetworking options is our next layer up. While small networks might be better understood directly, part of the goal in introducing algebraic objects is to provide support in returns to scale. Invariants from applied algebraic geometry and topology can be brought to bear on networks of any size.

- Homology algebraizes connectivity information from a graph enabling computer legibility and returns to scale.
- Zigzag persistent homology provides a means of tracking continuous chunks of connectivity over time, including joining and separating connections. This provides a strong foundation for automated subnetworking decisions.
- Graph varieties provide a different algebraic interpretation of connectivity information which enables more direct relations to more powerful data structures, such as sheaves.

Once structures are established and analyzed, optimization is the next layer up. Game theory provides an interesting framework for constructing and comparing network optimization approaches that may be helpful in addressing congestion in delay tolerant networks.

Temporal Graph Theory

Temporal graph theory is a still young field with plenty of room for growth. While we introduce some of our work in formalizing temporal graphs in Section 3, there is plenty more to say and prove about temporal graphs and graph sequences in forthcoming papers. This includes studying various properties of temporal graphs and determining extensions of network properties to temporal networks. One direction of exploration in temporal graph theory that could be particularly fruitful for space networks is the study of periodic temporal networks. The fact that orbits are often naturally periodic could be leveraged in the study of periodic temporal graphs and leveraged for the benefit of network analysis.

Another key thing to note is a shift in information available in a temporal graph. In the cases we are considering in this paper, the graphs are pre-determined and known, so global statistics can be computed. However, in practice we may only know the current configuration of a network without knowing its future configurations – perhaps relying on probability to provide possible futures. A temporal graph construction that views temporal graphs as a dynamical system rather than a fixed object, would be extremely valuable to the field. This would also be remarkably applicable to future space networks.

Another consideration is how local perspectives influence the system. Note that from the vantage point of any given node at any given time, its imagining of the network will differ from every other node. As such, constructing network views locally will have great influence on consistency in routing across different components. It is imperative that tools, such as sheaves, are utilized to synthesize and validate consistency of data across temporal networks. This may be useful in the probabilistic or deterministic settings.

Directed Multigraphs

In Section 4, we proposed a method for modeling time-evolving networks using directed multigraphs, and provided an alternate approach to pathfinding in Contact Graph Routing based on these multigraphs. Upcoming work in [15] will provide more of the details of this alternate algorithm, including a proof that it requires less than or equal to the number of iterations taken by the previous Contact Graph Dijkstra Search. It will also include experiments on simple networks simulated in SOAP that compare the two algorithms.

Future work in this area will likely continue to reformulate CGR in the language of directed multigraphs. As mentioned above, one opportunity for this change in perspective may be in the implementation of Yen's algorithm, which uses Dijkstra's algorithm repeatedly. While the existing version of CGR uses a version of Yen's algorithm for contact graphs, it is likely this could also be put in the framework of the multigraphs considered here. We hope that reformulating CGR in the language of multigraphs will not only increase the speed of computations but also lead to a more transparent approach to routing and serve as a foundation for future algorithms.

Centrality Measures and Summary Graphs

Summary graphs represent an interesting way to collect summary statistics for time-varying graphs. Determining which statistics are worth representing in this structure is a significant future project. Since summary graphs represent a kind of lossy compression of time-varying graph structures, knowing which statistics are preserving valuable information from the time-varying graphs is valuable outright. Moreover, if we want to feed this information into a machine learning algorithm for optimization, summary graphs seem primed for supplying concise yet relevant information.

Another interesting application of summary graphs comes in the form of network centrality measures. Typical network centrality measures are better suited to static graphs rather than dynamic graphs. So, applying network centrality measures to the summary graph can yield interesting ways to detect central nodes in a dynamic graph. Also, different summary statistics will likely correspond to different rankings for the same centrality measures. Such a comparison would be of great interest to us.

There are also adaptations of network centrality measures for time-varying networks already. It would be interesting to compare the results of these dynamic centrality measures with the same measures applied to different summary graphs. Certainly, this could detect some of the information loss from compressing to the summary graph.

Zigzag Persistent Homology and Applied Topology

Our goal is to construct a subnetwork and compare its zigzag persistence diagram to the zigzag persistence diagram of the original network. One way we can do this is by applying the bottleneck distance to the two diagrams. The bottleneck distance can be described in the following way. Suppose we have the zigzag persistence diagrams of two space networks. The two networks are "close" if the birth and death times of a bar in one diagram are close to the birth and death times of a bar in the other diagram. If there is a nice correspondence for every bar in the diagram, then the two diagrams are considered close per the bottleneck distance. For a more precise understanding of the bottleneck distance, please see the paper on zigzag persistence by Edelsbrunner and Harer [27].

Another way we hope to apply zigzag persistence to space networking is to apply it within the context of CGR. After constructing the subnetworks, one can construct a contact plan with this new structure. This reduces the number of vertices which in turn reduces the computation time. Again, we can use the bottleneck distance to compare the two networks and to see if this small change does not drastically change the underlying structure of the network. We hope to use this as a way to justify subnetwork construction and show the feasibility of the groupings.

The final extension we would like to explore is looking at the clique complex of the network as opposed to the simplicial complex associated to the network. A collection of vertices with all pairwise connections is a simplex in a clique complex; if we have three points connected, we fill in the triangular face, if we have four points connected to each other, we fill in the tetrahedron. This provides more information about the level of connectivity and the interaction between multiple devices.

Graph Varieties and Applied Algebraic Geometry

The theory of graph varieties allows us to model a static network as an algebraic variety. Space networks, however, are not static. Thus, it is necessary to formulate a theory of graph varieties that can be applied to temporal networks. A naïve first approach to such a theory is to view a temporal network as a sequence of graphs and simply compute the corresponding sequence of graph varieties. This approach does not take into account the relationships between graphs, so it is not optimal. It would be more beneficial to build a single graph variety that encodes the data of a temporal graph or interpret the picture variety of a temporal graph as a suitable subvariety of the picture variety of a sufficiently general graph. Subsequently, an algebro-geometric analogue of the Dijkstra sheaf would, in theory, give insight into the problem of finding shortest paths in a temporal network.

Not explored in this paper is the theory of cellules, introduced in [20]. These provide a natural way to interpret subnetworks of a static network modeled as a graph as points of picture varieties associated to these static networks. Solving shortest path problems in subnetworks could then be interpreted as solving an analogous problem on the points of the picture variety corresponding to these cellules.

Game Theoretic Networking

As space networks become increasingly complex, requiring global communication for decision-making is likely impossible given propagation delays that outlast windows of opportunity for real-time feedback. It is clear that new modeling techniques for routing decisions are essential for further space exploration. Game theory provides a well-developed foundation for understanding how decision-making might function in a network with incomplete information.

It is, however, likely that there are subnetworks that can share enough data to be cooperative although between two such subnetworks cooperation is infeasible. Such mixed approaches have been used to study Wi-Fi congestion in apartment buildings and could be generalized and reapplied to large-scale DTNs [28]. It should be investigated if the output could be used for real-time decision making for routing decisions, for example for load balancing.

10. Acknowledgements

The authors wish to acknowledge Greg Henselman-Petrusek, who graciously led us through his short-course on matroid theory[5].

References

[1] NASA, "Delay/disruption tolerant networking," Sept. 2020. [Online]. Available: https://www.nasa.gov/directorates/heo/scan/engineering/technology/disruption_tolerant_networking

[2] A. Hylton, R. Short, R. Green, and M. Toksoz-Exley, "A mathematical analysis of an example delay tolerant network using the theory of sheaves," in *2020 IEEE Aerospace Conference*, 2020, pp. 1–11.

[3] M. Moy, R. Cardona, R. Green, J. Cleveland, A. Hylton, and R. Short, "Path optimization sheaves," 2020. [Online]. Available: https://arxiv.org/abs/2012.05974

[4] R. Short, A. Hylton, R. Cardona, R. Green, G. Bainbridge, M. Moy, and J. Cleveland, "Towards sheaf theoretic analyses for delay tolerant networking," in *2021 IEEE Aerospace Conference (50100)*, 2021, pp. 1–9.

[5] D. Stodden and G. Galasso, "Space system visualization and analysis using the satellite orbit analysis program (soap)," in *1995 IEEE Aerospace Applications Conference. Proceedings*, vol. 1, 1995, pp. 369–387 vol.1.

[6] K. Mars, "Gateway," https://www.nasa.gov/gateway, Dec 2019. [Online]. Available: https://www.nasa.gov/gateway

[7] M. Hatfield, "Time history of events and macroscale interactions during substorms mission: Tracking the space storms responsible for triggering auroras," https://www.nasa.gov/mission_pages/themis/mission/index.html, May 2020. [Online]. Available: https://www.nasa.gov/mission_pages/themis/mission/index.html

[8] "Search of satcat," https://celestrak.com/satcat/search-results.php. [Online]. Available: https://celestrak.com/satcat/search-results.php

[9] "Pgda - 50,000 lro orbits," https://pgda.gsfc.nasa.gov/products/77. [Online]. Available: https://pgda.gsfc.nasa.gov/products/77

[10] "Index of /pub/naif/misc/more_projects/dsg," https://naif.jpl.nasa.gov/pub/naif/misc/MORE_PROJECTS/DSG/. [Online]. Available: https://naif.jpl.nasa.gov/pub/naif/misc/MORE_PROJECTS/DSG/

[11] J. A. Fraire, O. De Jonckère, and S. C. Burleigh, "Routing in the space internet: A contact graph routing tutorial," *Journal of Network and Computer Applications*, vol. 174, January 2021.

[5]Available here: `http://gregoryhenselman.org/teaching.html`

[12] O. Michail, "An introduction to temporal graphs: An algorithmic perspective," *CoRR*, vol. abs/1503.00278, 2015. [Online]. Available: http://arxiv.org/abs/1503.00278

[13] G. B. Mertzios, O. Michail, and P. G. Spirakis, "Temporal network optimization subject to connectivity constraints," *CoRR*, vol. abs/1502.04382, 2015. [Online]. Available: http://arxiv.org/abs/1502.04382

[14] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro, "Time-varying graphs and dynamic networks," *CoRR*, vol. abs/1012.0009, 2010. [Online]. Available: http://arxiv.org/abs/1012.0009

[15] M. Moy, A. Hylton, and R. Short, "An alternate pathfinding algorithm for contact graph routing," 2022, in preparation.

[16] E. W. Dijkstra *et al.*, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[17] J. Bang-Jensen and G. Z. Gutin, *Digraphs: Theory, Algorithms and Applications*. Springer-Verlag London, 2009.

[18] N. Mohanta, "How many satellites are orbiting the earth in 2021?" May 2021. [Online]. Available: https://www.geospatialworld.net/blogs/how-many-satellites-are-orbiting-the-earth-in-2021/

[19] D. Morozov, "Dionysus 2," https://mrzv.org/software/dionysus2/, 2017.

[20] J. L. Martin, "Geometry of graph varieties," *Transactions of the American Mathematical Society*, vol. 355, pp. 4151–4169, 2003.

[21] J. Burkart, D. C. II, C. Jansen, A. Taylor, and A. O'Keefe, "Graphs: graphs and directed graphs (digraphs). Version 0.3.2," A *Macaulay2* package available at https://github.com/Macaulay2/M2/tree/master/M2/Macaulay2/packages.

[22] C. Hillar, R. Krone, and A. Leykin, "EquivariantGB: Equivariant Groebner bases and related algorithms. Version 0.2," A *Macaulay2* package available at "".

[23] P. Deligne, "Sga 4 1/2 â cohomologie étale," *Lecture Notes in Mathematics*, vol. 569, 1977.

[24] R. Goldblatt, *Topoi: The Categorial Analysis of Logic*, ser. Dover Books on Mathematics. Dover Publications, 2006.

[25] M. O. Jackson and Y. Zenou, "Games on networks," *Handbook of Game Theory*, vol. 4, 2014.

[26] A. Gulyàs, J. J. Bìrò, A. Kőrösi, G. Rètvàri, and D. Krioukov, "Navigable networks as nash equilibria of navigation games," *Nature Communications*, 2015.

[27] H. Edelsbrunner and J. Harer, "Persistent homology—a survey," *Discrete & Computational Geometry - DCG*, vol. 453, 01 2008.

[28] M. Van Heesch, P. L. J. Wissink, R. Ranji, M. Nobakht, and F. D. Hartog, "Combining cooperative with non-cooperative game theory to model wi-fi congestion in apartment blocks," *IEEE Access*, vol. 8, pp. 64 603–64 616, 2020.

Alan Hylton should probably be designing tube audio circuits, but instead directs Delay Tolerant Networking (DTN) research and development at the NASA Glenn Research Center, where he is humbled to work with his powerful and multidisciplinary team. His formal education is in mathematics from Cleveland State University and Lehigh University, and he considers it his mission to advocate for students. Where possible, he creates venues for mathematicians to work on applied problems, who add an essential diversity to the group.

Robert Short earned his PhD in mathematics from Lehigh University in 2018. He worked as a Visiting Assistant Professor of Mathematics at John Carroll University until he joined the Secure Networks, System Integration and Test Branch at NASA Glenn Research Center in 2020. His research interests lie in the intersection of abstract mathematics and real world applications. Currently, his focus is on the foundations of networking theory and how to efficiently route data through a network using local information.

Jacob Cleveland is a Senior studying Mathematics and Computer Engineering at the University of Nebraska at Omaha. They joined the Secure Networks, System Integration and Test Branch as a Pathways Intern at NASA Glenn Research Center in 2020. Since joining, they have performed several research projects applying pure mathematics to engineering problems such as networking in space, star tracking, and artificial neural networks.

Olivia Freides is pursuing a masters in Data Science at American University with a concentration in Environmental Science. She earned her bachelors of science in Statistics in 2021 from American University. She was an applications of pure mathematics intern at NASA Glenn research center in 2021, and is a graduate researcher in the mathematics and statistics department at American University. Olivia's research interests span from applied mathematics and topology to environmental science and remote sensing.

Zander Memon is a senior mathematics and economics major at American University. He worked as an applications of pure mathematics intern at NASA Glenn Research Center in 2021, and has been a research assistant in both the mathematics and economics departments at American University since 2019. Zander's research interests include applied mathematics, algebraic topology, game theory, and topological data analysis. After graduation he plans to pursue graduate school in mathematics

Robert Cardona is a PhD student in applied topology. He studied computer engineering and mathematics before going on to work as a software developer. He then obtained a masters at Freie Universität Berlin and continued on to study applied topology at Albany.

Robert Green is a 2nd year mathematics PhD student at the University at Albany. He is studying topics including applications of Topological Data Analysis (TDA) and Graph Theory to space networking problems under Justin Curry. He previously completed his undergraduate and masters degrees in mathematics from American University where he worked with Michael Robinson on topics such as TDA and Signal Processing.

Justin Curry is an Assistant Professor of Mathematics and Statistics at the University at Albany, SUNY. Before arriving at Albany in 2017, he was a Visiting Assistant Professor at Duke University. Professor Curry earned his PhD in mathematics from the University of Pennsylvania in 2014, under the direction of Robert Ghrist. His research interests include the use of category theory in applied mathematics, with particular emphasis on applied sheaf theory, and inverse problems in topological data analysis (TDA).

Sriram Gopalakrishnan obtained his undergraduate degree in mathematics from the University of Utah and is currently finishing his MS in mathematics at Leiden University and Université de Bordeaux through the ALGANT program. His research interests are primarily computational algebraic geometry and number theory. His current research centers on problems concerning theoretical computer science and algebraic geometry.

Devavrat Vivek Dabke is a 4th year PhD Candidate at Princeton University in Applied and Computational Mathematics under the supervision of Bernard Chazelle. His primary research is in natural algorithms, which intersects graph theory, probability, and machine learning. He is most interested in the theory and applications of dynamic networks and relishes the opportunity to convert everyday problems into graphs. He enjoys figure skating, long walks, transportation, and lakes.

Brittany Story is a PhD student in mathematics at Colorado State University. She is set to graduate in Spring 2022 and will be working as a postdoc under Vasileios Maroulas after graduation. Her research interests include topological data analysis and using mathematics to study networks.

Michael Moy is pursuing a PhD in mathematics at Colorado State University, having completed his master's there in 2021. His research is focused on applied topology. During the summers of 2020 and 2021, he worked as an intern at NASA through the SCaN Internship Project. His research areas at NASA have included machine learning and mathematical approaches to networking.

Brendan Mallery is a PhD student studying mathematics at Tufts University. Previously he received a Masters in Mathematics from the University at Albany, SUNY in 2020, and a Bachelors in Mathematics and Chemistry from Bowdoin College in 2018. His research interests include geometric group theory, optimal transport and applied sheaf theory.