Introducing Tropical Geometric Approaches to Delay Tolerant Networking Optimization

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

Brendan Mallery
Tufts University

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

Devavrat Vivek Dabke
Princeton University

Olivia Freides
American University

***Abstract*— Delay Tolerant Networking (DTN) is the standard approach to the networking of space systems with the goal of supporting the Solar System Internet (SSI). Current space networks have a small scale and often depend on rigorously scheduled (pre-determined) contact opportunities; this manual approach inhibits scalability. The goal of this paper is to recast these scheduling problems in order to apply the optimization machinery of tropical geometry.**

Contact opportunities in space are dependent on such factors as orbital mechanics and asset availability, which induce time-varying connectivity; indeed, end-to-end connectivity might never occur. Routing optimization within this structure is classically difficult and typically depends on Dijkstra's algorithm as applied to contact graphs. Alternatively, we follow the successes of tropical geometry in train schedule optimization, job assignments, and even traditional networking, by extending this approach to this more general (i.e. disconnected) problem space.

These successes imply tropical geometry provides a useful framework in the context of DTNs, starting with applications to queuing theory and long-haul links. Recently, tropical geometry has been applied to parametric path optimization on graphs with variable edge weights. In this work, we extend these advances to account for the problem of routing in a space network, and find that tropical geometry is well-suited to the challenges offered by this new setting, including contact schedules featuring probabilities. Our approach leverages the combinatorial nature of the problem to give feasible shortest path trees in the presence of variable channel conditions and latency, evolving topologies, and uncertainty inherent in space routing.

We discuss our tropical approach to DTN for two Python implementations, a Verilog Tropical ALU implementation, tropical frameworks for other parametric graph problems, and solution stability. Lastly, a future work section is included to illuminate the path ahead.

1. Introduction

One major complication associated with space networking is that the links between assets are constantly changing. Both in terms of links literally coming up and down, but also in terms of channel characteristics. One such characteristic is latency, which we can think of as the time it takes for information to leave the transmitter and arrive at the receiver. In space, this latency mainly arises from light travel time, which can be significant (as many as tens of minutes between Earth and Mars).

In addition, the main form of transmissions being electromagnetic waves means that a space network will suffer due to the inverse-square law. This law basically dictates that the power received is inversely proportional to the square of the distance to the transmitter. From an information theoretic perspective, this severely limits the data rate one could hope to achieve based on the physics of light alone, let alone any complications with modulations or electronics. Note that both latency and the received power depend on distance, a quantity which is constantly changing for assets in space. Section 2 details an algorithm and two implementations that account for these limitations by casting them as a parametric shortest path problem.

Below, in Figure 1, we see a small example of a space network between the Earth and some satellites. Despite the relatively 'small' size - there are only six assets - the graph is highly dependent on orbital mechanics and hence time. This emphasizes the need for rigorous approaches to parametric graphs in space communications.

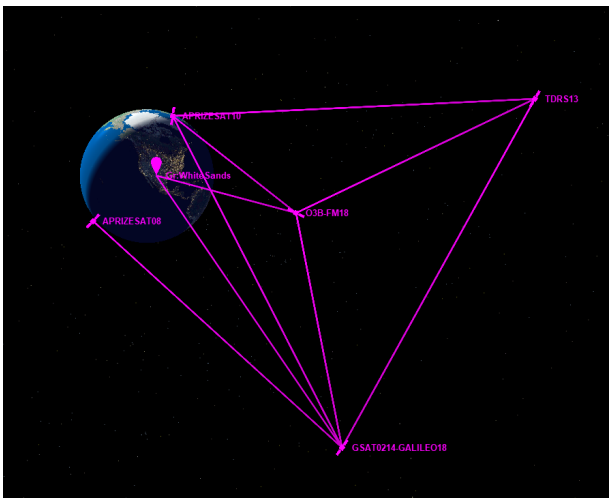

Figure 1: Example space network

Parametric graphs are modeled as graphs with variable weights, see Definition 3.1. These weights could correspond to distance (in light-seconds) or bit rate (in Mbps) for applications to DTN. Regardless of their interpretation, one aims to find optimal paths as a function of these weight parameters x_i, which themselves could depend on time. The complexity of this problem is reflected in a partition of parameter space, which depends on the topology of the network as well as given weights. Over each region in this partition, optimal routing is determined using a shortest path tree in the graph, but different regions may have the same or different trees. Generating these regions and trees is accomplished in the tropical setting using Joswig's algorithm [1], which is reviewed below. For a concrete example of a parametric graph, consider Figure 2. Depending on the value of x, different routing decisions will be used and a complete classification

2022 IEEE Aerospace Conference (AERO) | 978-1-6654-3760-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/AERO53065.2022.9843242

of these routing decisions, using tropical geometry, will be provided later in the paper; see Figure 3.

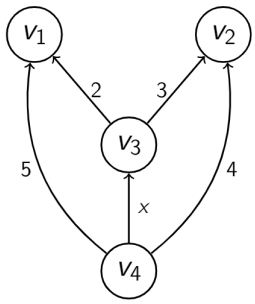

Figure 2: Example parametric graph.

Another application of these methods comes from a recent analysis of the current standard of DTN routing, contact graph routing[2]. Originally, it was thought that contact graphs were acyclic, but it has recently been shown that they can in fact feature routing loops [3]. Hence, one feature of our analysis is the ability to track how topological features - such as loops - both arise and disappear as the underlying parameters vary. This motivates a more general parametric approach.

To summarize our contributions, we initiate a more general parametric and tropical geometric approach to time-evolving graphs with variable edge characterizations. We provide a slight generalization of the tropically inspired method of Joswig [1] for solving the parametric shortest path problem and prove two novel implementations of this algorithm: one in Python and one using a Verilog 32-bit tropical arithmetic logic unit (ALU). Key to our approach is the extraction of regions of parameter space where certain shortest path trees are optimal. By understanding these, one could design in advance what the routing tables should like for portions of a Solar System Internet. We then move beyond the tropical setting of Joswig to provide a more general mathematical framework for other parametric graph problems and characterize stability of solutions to these in a novel way. We anticipate that these methods will be highly applicable to DTN as the characteristics of these networks include not only time varying capacities, but also other time varying summaries of note, such as centrality.

A Brief View of the Tropics

Before we proceed with our main line of analysis, we offer a quick summary of tropical geometry and its application to graph theory. Those interested in a deeper text are referred to [4]; there is also a technical memorandum published by the authors that takes a more leisurely pace [5].

Tropical geometry, in broad strokes, can be thought of as a piecewise-linear version of algebraic geometry, which studies solution sets (i.e. zero sets) of systems of polynomials. The polyhedral view of tropical geometry allows one to phrase things like optimization problems fairly easily, as we will see in Section 2 and beyond.

Whereas ordinary geometry occurs over the ring of real numbers $\mathbb{R}$, tropical geometry occurs over the min-plus[1] semiring $\mathbb{T} = \mathbb{R} \cup \{\infty\}$ where the operations of addition and multiplication are redefined as follows:

$$a \oplus b := \min\{a, b\}$$

$$a \otimes b := a + b$$

To illustrate how this works, we offer the following examples:

- $5 \oplus 7 = 5$,
- $5 \otimes 7 = 12$,
- $a^{\oplus b} = a$,
- $a^{\otimes b} = ba$,
- $5 \oplus \infty = 5$, and
- $5 \otimes \infty = \infty$.

To see why $a^{\oplus b} = a$, note that $a^{\oplus b} = \min\{a, \ldots, a\} = a$, as this is a "added" to itself b times. To understand $a^{\otimes b}$, observe that for $b \in \mathbb{Z}_{\geq 0}$ we may view exponentiation as repeated application of the $\otimes$ operation, b times. To extend this definition to all integers, we note that $a^{\otimes -1} = -a$ since this is the additive (tropical multiplication) inverse of a. For $b \in \mathbb{Q}$, we can observe that $(a^{\otimes \frac{1}{n}})^n = a$ so $n(a^{\otimes \frac{1}{n}}) = a$, so $a^{\otimes \frac{1}{n}} = \frac{1}{n}a$. Lastly, now that we can have rationals in our exponent, we can perform a process known as *completion* to get to our extended real values, but this is beyond the scope of this paper. For those interested, see [6],[7],[8],[9], and [10].

Generalizing the rules of tropical arithmetic to tropical matrices casts new light on classical lessons from graph theory. Recall that associated to a (di)graph is the adjacency matrix A, where $A_{i,j} = 1$ if there is an edge from vertex i to vertex j, and $A_{i,j} = 0$ otherwise. It is well known that taking the n^{th} power of the adjacency matrix enumerates walks from vertex i to vertex j of length n.

Analogously, one can perform a similar process for a weighted (di)graph in the tropical setting, where the entries in the matrix are now the weights of the arcs between the edges. Taking tropical powers of this weight matrix now calculates the total weight of the shortest walk from vertex i to vertex j. See [5][11] for a detailed explanation on this process, as well as a proper definition of tropical matrix operations. These observations have yielded fruitful insights in the study of train schedules, robotics, and many other applications; see [12].

2. JOSWIG ALGORITHM GENERALIZATION

For this section, unless explicitly stated, we are working with weighted, parametric digraphs, i.e. directed graphs where some arcs have constant values associated with them (weights), and some arcs have variables associated with them (parameters). Further note that in [1], they work with parametric digraphs satisfying an additional condition which they call 'separability' which means that each parameter only appears as part of one arc's parameter expression. Figure 2 satisfies this separability property. For our work, we allow our graphs to not have this quality so that we can eventually substitute functions of time for our parameters. This is a minor distinction, but it does limit the claims we can make about our overall solution spaces at the end of our work. Lastly, we assume that all parameter values and edge weights are restricted to be non-negative, and that our graphs have

[1]One can define the max-plus semiring by instead including $-\infty$ and defining the $\oplus$ operation as $\max$ instead. We will use the min-plus semiring because in our optimization setting we wish to minimize path length.

a single sink (or source). These are typical assumptions of Dijkstra's algorithm, which is employed as a step in the Joswig algorithm.

Definitions

In the sections that follow, we assume the reader has some familiarity with topics covered in a typical data structures course. For the reader who wishes to gain this background knowledge, we refer to [13] and [14].

Let T be a spanning tree in a graph G. Define $P_T(v)$ as the cost of the unique path from the source s to the vertex v in T. Furthermore, define $d(vw)$ as the cost of the arc between v and w, with $d(vw) = \infty$ if there is no arc. For a general path $s = v_1, v_2, \ldots, v_{n-1}, v_n = v$, the cost is given by summation, i.e.

$$P(v) = \sum_{i=1}^{n-1} d(v_i v_{i+1}).$$

A *parametric arc* is an edge whose weight is given by a variable. We are interested in graphs with (possibly) multiple parametric arcs and their corresponding parametric shortest path trees. For emphasis, we recall that we restrict ourselves to non-negative edge weights for both constant and parametric arcs.

Now that we've defined some notation, we can describe how one approaches parametric graphs like the one in Figure 2. See Definition 3.1 for a formal definition of a parametric graph.

Algorithm

The algorithm given in [1] can be summarized as follows:

1. Take a weakly connected digraph G with (variable) edge parameters $x_1, \ldots, x_n$, initialized to any non-negative values $\alpha_1, \ldots, \alpha_n$, fixing a source vertex s. Denote this initialized graph $G(\alpha_1, \ldots, \alpha_n)$.
2. Perform Dijkstra's algorithm on the initialized graph $G(\alpha_1, \ldots, \alpha_n)$ to generate a shortest path tree T rooted at s.
3. Now considering G with original (un-initialized) edge parameters, iterate through each edge vw not in T as follows: consider $P_T(v)$ and $d(vw) + P_T(w)$. If they are incomparable, add $P_T(v) \leq d(vw) + P_T(w)$ to the system of inequalities associated with T. Then generate a new tree T', obtained by removing the path in T to w and replacing it with the path to w through v via vw. If T' is not already in the list, and is also a shortest path tree, append T' to the list.
4. Repeat step 3 for each tree until no new trees are generated.

The result is a family of shortest path trees, along with systems of inequalities associated to each tree that defines the region of parameter space where that tree is the shortest path tree. A visual representation of this output is shown in Figure 3[2]. This process of taking a parametric graph, iterating through its paths, and generating trees along with inequalities is the essence of the Joswig algorithm. This is in fact tropical because the systems of inequalities can be viewed as (systems of) tropical polynomials equations, an observation which is explored in [5]. Each shortest path tree T in the solution family is encoded as a set of monomials which minimize a tropical polynomial for a certain subset of parameter space, which is defined by the inequalities associated to T. Some trees will never be optimal for any value of the parameters, which is reflected by the associated set of inequalities giving an unfeasible solution set (i.e. bounding an empty region).

[2]Although we include dashed lines to indicate edges from our original graph, technically the shortest path tree only consists of the solid edges.

Implementations

The algorithm above was implemented[3] previously by Ewgenij Gawrilow in Polymake as an optimized extension of Polymake 4.1. In an effort to make the implementation more accessible and applicable to engineering problems, we developed two object oriented implementations written in Python with minimal dependencies. This not only makes it easier to read and understand, but also makes things like field programmable gate array (FPGA) implementations much more feasible. Moreover, the implementation generalizes the algorithm: while previous impementations required edge functions to be linear, the algorithm implemented here was modified to apply to arbitrary functions, such as $sin(t)$.

To begin, we should first acknowledge the efficacy of the naive solution: the naive solution here would be to just sample points in the parameter space of the edge weights and keep track of which sample points correspond to which shortest path trees. However, without any good way to pick samples, this severely limits the accuracy to which one can approximate the boundaries between different regions. Moreover, this approach doesn't leverage the fact that the regions in parameter space corresponding to a shortest path tree are all convex: If the points x and y both yield the same tree T, then any point on the line segment between x and y must also correspond to T. This observation gives us a good idea of how to start approximating solutions.

Our first implementation is a boundary approximation method based on binary search, and utilizes convexity of these regions in an essential way. The purpose of approximating is to verify our other implementation, but also eventually the Polymake implementation. The problem with this method is that it is inefficient in both memory and time, since it requires sampling a large number of points. The advantage, however, when compared to the naive solution, is that this approximation can be tuned to arbitrary precision, limited by floating point accuracy, among other things. So, while it may not be practical for deployment in a space network, it will at least be accurate enough to determine the validity of our more efficient methods.

This method operates recursively. In one dimension, i.e. only one edge weight is variable, the shortest path tree is found and recorded for when the parameter is set to the minimum value (e.g. 0). Then another tree is found for setting the parameter to the sum of all constant weights (so ignoring parameters) in the graph[4]. If they're the same tree, then we only have one shortest path tree feasible and are finished. If they're not, then one recursively binary searches between these left and right bounds to find the boundary between the two regions, slowly squeezing these bounds in. In the case that a sample lands on a region with a shortest path tree not yet sampled, we will

[3]Link to implementation: `https://polymake.org/extensions/polytropes`

[4]In one dimension, this is guaranteed to lie in the region extending off to infinity since any path without the parametric arc, assuming such a path exists, will be better than any path including the parametric arc. That is, for any value of the parameter above this sum, you will still get the same shortest path tree.

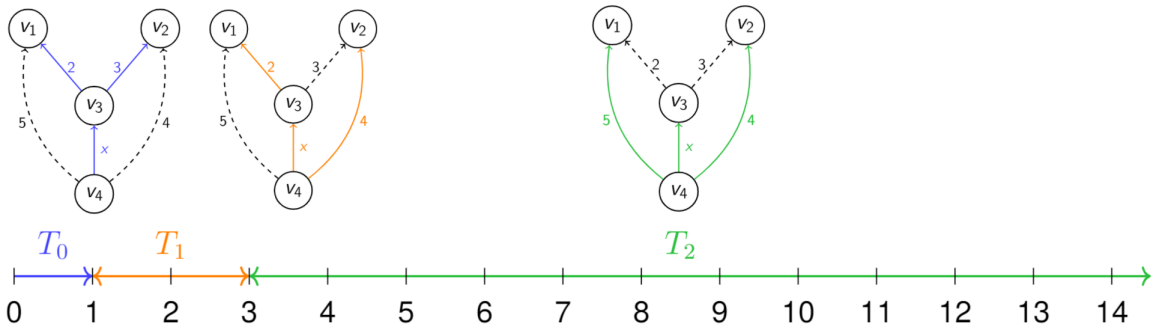

Figure 3: Analytical solution of cell decomposition and associated shortest path trees corresponding to Figure 2. Results from applying the Joswig algorithm described below. Each interval shows the values of parameter x for which each tree $(T0, T1, T2)$ optimal. Note that there is a fourth tree omitted, basically $T0$ but with the path to v_1 going along the leftmost edge with weight 5. It is omitted because there are no values of x that make it optimal.

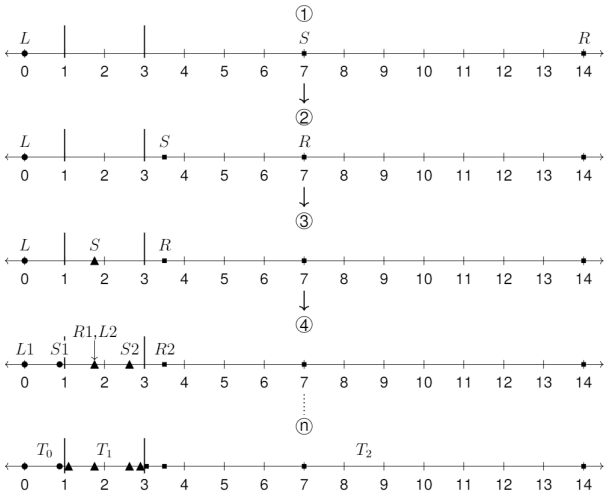

Figure 4: Example sequence of steps in the binary search.

now split the problem into two recursive binary searches, one between the left bound and the sample point, and the other between the sample point and the right bound. This process is repeated until the recursion depth reaches the limit set by the programmer. This recursion depth is what will ultimately determine the accuracy of the approximation, and depths of 5 or 10 seem to suffice for most applications.

Here's an example corresponding to Figure 4. The output of this example from our implementation is given in Figure 5.

1. Initialize [5] the search with $L = 0$, $R = 14 = 2+3+4+5$, and $S = (L + R)/2 = 7$.
2. Check containment of L, S, and R from step 1. Generate sample $S = 3.5$ since $L = 0$ corresponds to $T0$ and $R = 7$ corresponds to $T2$.
3. Repeat above to generate sample $S = 1.75$.
4. Checking sample $S = 1.75$ reveals containment in the region for $T1$, unique from $T0$ and $T2$ regions. Generate two new samples for two new searches, one between step 3's L and S, and another between step 3's S and R.

[5] In one dimension, we can ignore all parameter values above the sum of all constant weights (i.e. not parameters). There could probably be tighter bounds, and in some trivial cases, such as no constant weights, this is actually too tight as it is.

⋮

n. After repeated iterations, end up with a reasonable approximation of boundaries at 1 and 3.

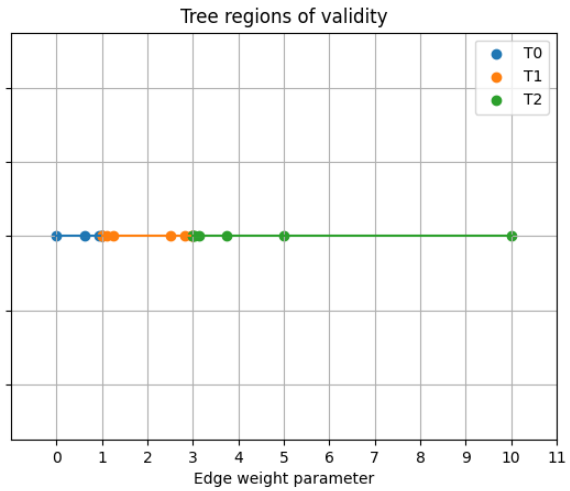

Figure 5: Example output of binary (first) implementation applied to Figure 2. Note that this agrees with the solution given in Figure 3.

For two dimensions (i.e. two parametric edges), we essentially repeat the one dimensional process outlined above on a series of lines going through our region of interest. Since we are working in two dimensions, we no longer have the guarantee that regions are constant beyond the total weight of the non-parametrized edges. We provide an example in Figure 6 with parameters x and y. For this case, it is easiest to just pick arbitrarily large values, say M, assuming one of the boundaries isn't $y = x$, and search along all lines between $(0, 0)$, $(0, M)$, $(M, 0)$, and (M, M). Then, picking a number of sample points n, e.g. $n = 5$, one draws sets of 5 lines, each set of lines drawn between a vertex and points along one of the opposite edges. An example of this is shown in Figure 7. The intuition here is that we are trying to avoid sampling a line perfectly over a boundary, although this may occur in some cases. Generically, our sample lines will go through several regions to help 'detect' as many boundaries as possible, and this is one approach. The result of this two dimensional binary search is shown in Figure 8.

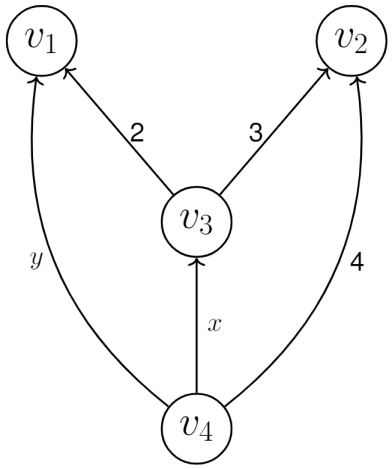

Figure 6: Example graph with two parameters. Note separability.

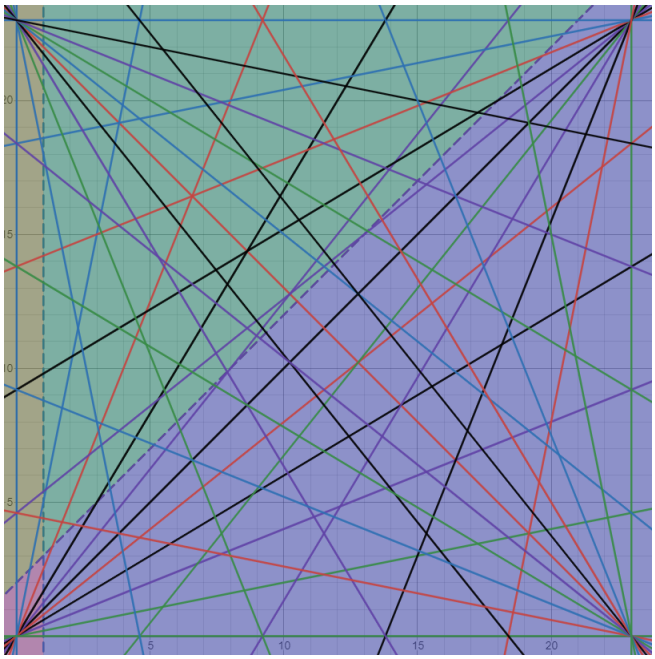

Figure 7: Lines for two parameter binary search, $n = 5$ sample lines, corresponding to Figure 6

Our second implementation[6] is a modified version of the algorithm described above and in [1]. It works by generating a list of all paths possible in the graph, along with the total cumulative weight along each path, as well as whether or not there are variable edges along that path. For each path not on the tree being considered, it solves for the boundary between two regions. One region corresponds to the tree with the original path. The other other region corresponds to the tree with the new path. For example, consider Figure 3 which shows the tree T_0. Fix v_4 and v_2 and consider the two paths between them, one going through v_3 with weight $x + 3$, and one directly between v_4 and v_2 with weight 4. We know the boundary between the region for T_0 and the region for T_1 is exactly when $x + 3 = 4$, i.e. $x = 1$, since we are fixing the other edges, namely the path going from

[6]Link to implementation: https://github.com/jacleveland/joswig

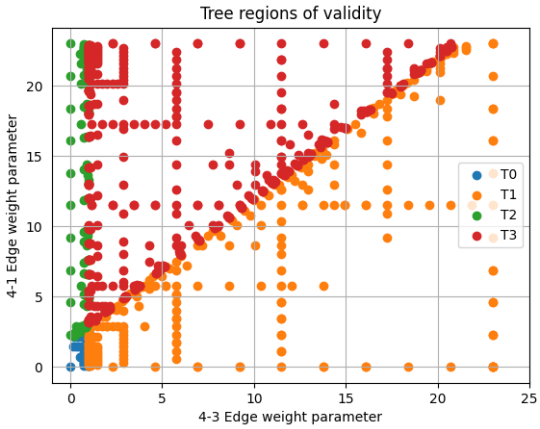

Figure 8: Example output of binary (first) implementation applied to Figure 6.

v_4 to v_1 through v_3. Lastly, we would find the boundary between T_1 and T_2 by noting that the boundary between them is exactly when $x + 2 = 5$, i.e. $x = 3$. For our example, in Figure 3, since there is only one parameter, our boundaries are fixed values. The output of this implementation applied to the graph in Figure 2 is shown in Figure 9, which shows that our boundaries between regions are at $x = 1$ and $x = 3$. In two parameters, the boundaries between cells are lines. The dashed lines between the regions in Figure 7 are such boundaries.

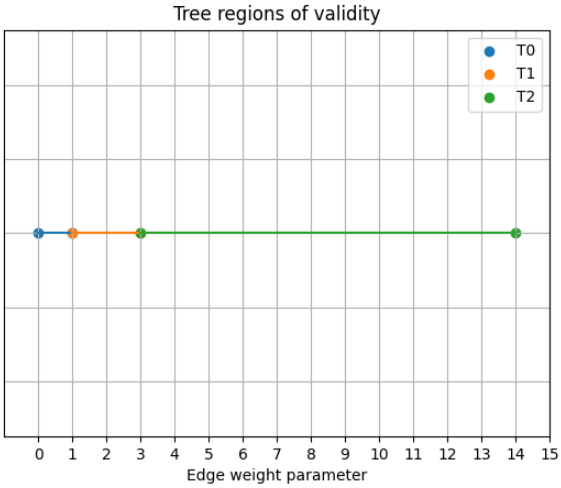

Figure 9: Example output of our Joswig (second) implementation applied to Figure 2. Note that this agrees with the solution given in Figure 3.

More generally, for a graph with d-dimensional parameter space, these boundaries[7] are $d - 1$ dimensional affine subspaces of $\mathbb{R}^d$. Assuming separability, i.e. all of the variable edge weights are represented by unique x_i, we know that there will only be coefficients of ± 1 or 0 in front of each parameter, along with a constant representing the shifting

[7]Note that points on the boundary represent parameter values associated with multiple trees that are equally optimal. One may convince themselves of this fact by observing that in Figure 3, when $x = 1$, both $T0$ and $T1$ are optimal because $x + 3 = 4$ when $x = 1$.

of the affine subspace.[8] In essence, we are solving for this constant, and the sign of the coefficient of each variable, which will depend on which path each variable appears on, if at all[9].

Verilog Implementation

In addition to working on Python implementations, great strides were also made in creating our Verilog implementation[10] of a 32-bit tropical arithmetic logic unit (ALU). This tropical ALU is capable of executing instructions for the $\oplus$ and $\otimes$ as well as the logical AND and OR operations. The 32nd bit indicates the ∞ of the tropical semiring, i.e. 32'b1xxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx. The arithmetic operations here represent the usual operations as defined in §1. Note that for $\otimes$, overflowing results in finite values for simplicity, as demonstrated by the waveform in Figure 10. An alternative is to have $\otimes$ overflow to ∞ since any value larger than $2^{31} - 1$ could be considered to be ∞. As a compromise, future revisions could include an overflow mode that selects between overflowing normally and overflowing to infinity.

3. PARAMETER SPACE DECOMPOSITIONS FOR WEIGHTED GRAPHS

In this section, we discuss several natural mathematical extensions of the framework introduced in [1]. First, we discuss what happens when one changes the problem under consideration from the single source shortest path problem to other problems of interest for a weighted graph. Second, we consider how the parameter space perspective allows us to characterize some of these problems, even when they don't yield tropical formulations. One upshot of this section is the ability to define a principled notion of stability of solutions to these parametric problems. The results of this section are outside the purview of traditional tropical geometry, though the perspective taken is heavily inspired by the tropical geometric approach to parametric routing.

Tropical Graph Problems

To formally describe more general parametric problems on graphs, we introduce some terminology:

Definition 3.1. Let G be a graph with n nodes V and k edges E. Let $E' \subset E$ be a subset of edges of size $m \leq k$, and suppose that we assign to each $e \in E'$ an affine function in the real variable x_e, and all other edges $s \in E \setminus E'$ a constant weight. We call weighted graphs as defined above *parametric graphs*. We refer to $\mathbb{R}^m$, where $m = |E'|$, as the *parameter space* of G, and identify each $x \in \mathbb{R}^m$ with a fixed choice of parameter values on G.

For a fixed parametric graph G with $m \leq k$ parametric edges, every point x in parameter space $\mathbb{R}^m$ corresponds to a different choice of weights for G.

[8]Even if one didn't have separability for the parametric graph, one could force separability by a relabeling process where x on two different arcs would be replaced by y and z, and the process would be performed on this more general graph.

[9]It is important to note, however, that some of these inequalities will be superfluous, if for example you have two parallel arcs (i.e. same source and sink) with the same exact variables, but the constants of one path add up to a smaller weight than the other. This issue can be solved by pruning the graph to begin before performing the computations, or alternatively, applying separability and treating the variables separately.

[10]Verilog implementation available here: `https://github.com/jacleveland/tropicalu`.

Definition 3.2. A parametric problem P on a weighted graph G with m parametric edges is described by the following:

- A set of discrete objects called the solution set J;
- An assignment $h : \mathbb{R}^m \to \mathcal{P}(\mathcal{J})$, where $\mathcal{P}(\mathcal{J})$ is the power set of J.

Example 3.3. The parametric all-pairs shortest path problem on a weighted graph $G = (V, E)$ has as its solution the set of all paths U where:

- For each $x, y \in V$, U contains a unique path p_{xy} from x to y;
- If p_{xy} has a subpath from u to w, then p_{uw} is given by this subpath.

The assignment $h : \mathbb{R}^m \to J$ maps each weight vector to the set(s)[11] of lowest weight paths in J.

Example 3.4. The parameterized max-flow problem on a weighted graph G with source v_{src} and sink v_{sk} has as its solution set all graph cuts separating v_{src} from v_{sk}, i.e. the set of all minimal sets of edges $Q = \{e_1, e_2, \ldots, e_l\}$ such that when Q is removed from G, v_{src} and v_{sk} are in different connected components of G. The assignment $h : \mathbb{R}^m \to J$ maps each weight vector to the minimal weight cut(s) in J.

Definition 3.5. Let G be a parameterized graph with m parameterized edges. Let P be a problem on G with a discrete solution set J. Let $h : \mathbb{R}^m \to J$ be the assignment function for P. For each $\alpha \in J$, let $U_\alpha := h^{-1}(\alpha)$ be the region in $\mathbb{R}^m$ assigned to α. We say that $\mathcal{U}_J := \{U_\alpha \mid \alpha \in J\}$ be the decomposition of $\mathbb{R}^m$ induced by J.

Following through the definitions, one quickly sees that the decomposition of $\mathbb{R}^m$ induced by the all-pairs shortest path problem on a graph G with m parameterized edges is in fact the same tropical hypersurface introduced in [1]. We observe that many natural parametric problems on graphs may also be phrased in terms of tropical equations, and hence will decompose parameter space into cells given by a tropical hypersurface. We will term such problems as *tropical graph problems*. We now give a (certainly non-exhaustive) list of tropical graph problems:

- Single source shortest path
- All-pairs shortest path
- Minimum spanning tree
- Traveling salesman
- Max-flow

Knowing that a problem is a tropical graph problem immediately provides a lot of information about the parameter space decomposition. This is due to the regularity of tropical hypersurfaces: for instance, we are guaranteed that for any solution $\alpha \in J$ the region assigned to α is a convex, m-dimensional set with piecewise linear boundary, i.e. convex polytopes. Furthermore, we know almost all parameter vectors have a unique solution.[12]

[11]Note that two paths p_{xy} and s_{xy} can be equivalent in terms of having optimal path length and allowing equivalently optimal paths to branch off of them as well. Recall that in Section 2, any point on the boundary between two cells had at least two equivalent shortest path trees. Hence some weight vectors will map to multiple equally optimal solutions.

[12]Note that parameter vectors on the boundary between two cells will correspond to two or more different solutions in J.

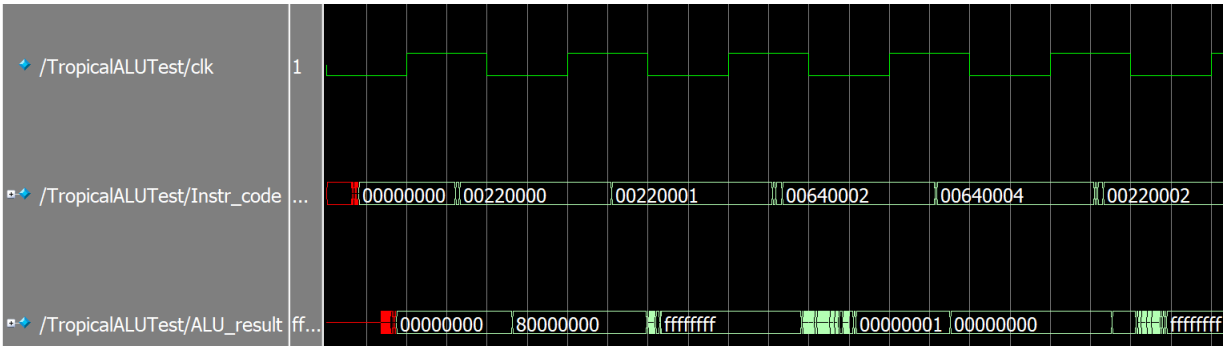

Figure 10: Waveform associated with the Verilog Tropical ALU implementation.

```
R0 = 0x00000000; //register file
R1 = 0x80000000; //initialization
R2 = 0xFFFFFFFF;
R3 = 0x7FFFFFFF;
R4 = 0x00000001;

R0 = R1 & R2;
R0 = R1 | R2;
R0 = min(R3,R4); //oplus
R0 = R3 + R4;    //otimes
R0 = min(R1,R2);
```

Figure 11: High level representation of program being demonstrated in the waveform of Figure 10.

This structure makes such decompositions of parameter space much easier to determine with algorithms, using for instance the algorithm described in the previous sections, or with substantial modification.

We are also guaranteed that if we take natural "combinations" of such problems, the resulting parameter space decomposition is once again tropical. In particular, consider the natural definition of a product of problems, i.e. a problem with solution set given by a Cartesian product of solution sets for other problems. As natural examples of problems that can be phrased as products of simpler problems, note that shortest path spanning tree may be written as a product of instances of shortest path with fixed endpoints, and that all pairs shortest path may in turn be written as a product of instances of shortest path spanning tree. The following lemma is trivial from a tropical geometry perspective, but is worth mentioning in our new language:

Lemma 3.6. *Suppose that P_I, P_J are problems with assignments given by tropical polynomials $h_I, h_J : \mathbb{R}^m \to \mathbb{T}$. Then the assignment for $P_I \times P_J$ is given by the tropical polynomial $h_I \bigotimes h_J$.* □

Though one may always define such a product regardless of whether the problems are tropical, it is comforting to know that tropical problems are closed under this operation.

More General Problems

From the above discussion, one might be inclined to think that all natural parameterized graph optimization problems are tropical. This is not the case, and a prominent example of problems that do not have this property arise from various centrality measures.

Example 3.7. A parameterized centrality problem on a weighted graph G has as its solution set all orderings of vertices, where we allow for orderings where two or more vertices to "tie". The assignment $h : \mathbb{R}^m \to J$ maps a parameter vector w to the ordering on vertices induced by a chosen centrality measure computed on G with weights given by w.

Commonly used centrality measures include eigenvector, Katz and betweenness centrality, see [15]. For a given centrality problem with assignment h, we may consider the decomposition induced by orderings on the vertex set. However, we find that the cells in this decomposition are in general no longer convex with piecewise linear boundaries.

Example 3.8. Consider parameterized eigenvector centrality on a parameterized graph G. Suppose that G has m parameterized edges, and consider the parameterized adjacency matrix $A_G(w)$, which is a function of $w \in \mathbb{R}^m$. Then we see that both the maximum eigenvalue and the corresponding maximal eigenvector

$$A_G(w)v = \lambda_{max}(w)v$$

depends on w. Notice that by definition of the adjacency matrix, the ith entry of the eigenvector v_i may be identified with the ith vertex of G. The eigenvector centrality of a graph is defined to be the ordering imposed on the vertices by the "score function" given by the entries of v, which in the parametric regime is a function of our parameter vector.

We see that solution regions are defined by the (in)equalities of the form

$$\sum_j a_{ij}(w)v_i = \lambda_{max}(w)v_i$$

$$v_i(w) \geq v_k(w), i \neq k.$$

These (in)equalities almost look like the defining (in)equalities of a semi-algebraic set, and indeed if λ_{max} were constant this would be the case. However, λ_{max} depends on the parametric characteristic polynomial of the matrix $A_G(w)$ and hence is (generically) not a polynomial.

Example 3.9. Despite the difficulties presented by the general case, for certain graph architectures one is able to compute the corresponding eigenvector centrality decomposition without too much difficulty. For instance, consider the star graph with n edges, i.e. the graph with $n+1$ vertices such that there is an edge from the first vertex to all other vertices, and no other vertices are connected by edges. The

parameter space of the weight space is $\mathbb{R}^n$. The principal eigenvector is given by $(1, \frac{w_1}{||w||_2}, \frac{w_2}{||w||_2}, \ldots, \frac{w_n}{||w||_2})$ where $||w||_2 = \sqrt{w_1^2 + w_2^2 \ldots + w_n^2}$. Thus we see that every possible permutation $\sigma \in S_{n+1}$ that fixes 1 is a possible solution for this problem, as the ranking is ordered by the ordering on the weights, apart from vertex 1 which is always central. The set of these are in bijection with the symmetric group S_n, and hence we define the set of regions

$$U_\sigma = \{(w_1, w_2, \ldots, w_n) \in \mathbb{R}^n \mid w_{\sigma(i)} \geq w_{\sigma(j)}, \sigma \in S_n\},$$

which define the regions of parameter space for which σ is the ordering induced by the eigenvector.

The situation for Katz centrality is similarly complicated. It is interesting to note that Katz centrality depends on a parameter such that when one takes an appropriate limit, eigenvector centrality is recovered, and hence we expect their respective parameter space decompositions to converge as well.

In contrast, the betweenness centrality decomposition yields a description that, while not tropical, can be constructed using convex polytopal regions:

Theorem 3.10. *Let β^* be the assignment induced by betweenness centrality. Then for all $\alpha \in J$ the region $h^{-1}(\alpha)$ is a union of convex polytopal regions, glued along (potentially empty) faces. In other words, $h^{-1}(\alpha)$ is a polytope complex.*

Proof: We may record the betweenness of N nodes as a vector in $\beta(w) \in \mathbb{R}^N$, which we write this way to emphasize the $w \in \mathbb{R}^m$ dependence. From this, we obtain a solution (for parameter w) to the betweenness centrality problem by considering the ordering of entries of $\beta(w)$, which we have denoted $\beta^*(w)$.

Consider the betweenness of a vertex $v \in V$, which we denote $\beta(w)_v$. Recall that this is given by the formula

$$\sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}.$$

The key observation is that both σ_{st} and $\sigma_{st}(v)$ are determined solutions to the all-pairs-shortest path problem, as both quantities are given by the size of subsets of the set of $\frac{n(n-1)}{2}$ optimal paths for a given weight vector. Hence, we see if $\beta(w)_v \geq \beta(w)_q$ for $v, q \in V$, then this inequality holds for all vectors w' in the same region as w in the all-pairs shortest path problem. Thus, $\beta^*(w) : \mathbb{R}^m \to J$ defines a face-wise constant function on the decomposition induced on $\mathbb{R}^m$ by the all-pairs shortest path problem on G. There may be multiple cells of the all-pairs shortest path problem that induce the same ordering α, hence $(\beta^*)^{-1}(\alpha)$ may be a union of polytopes, which are necessarily glued along (possibly empty) faces as they are chosen from a decomposition where all regions have this property. □

The authors are currently conducting a deeper investigation of the geometry of decompositions induced by various centrality problems.

Graph Symmetries and Parameter Space Decompositions

A challenge with constructing or even sampling parameter space decompositions is that for any graph of reasonable size one is quickly confronted with a high dimensional problem. However, if the graph topology in question has nontrivial automorphisms, we may be able to leverage the additional symmetry to simplify the decomposition, provided that the particular problem in question respects these symmetries. For simplicity, we will assume that all edges of G are parametric.

Definition 3.11. Let $G = (V, E)$ be a graph. A graph automorphism ψ is a bijection from G to itself such that if v and w are adjacent in G then $\psi(v)$ and $\psi(w)$ are also adjacent. The automorphisms of a graph form a group, which we denote $Aut(G)$.

Lemma 3.12. *Let G be a graph with m edges, all of which are parameteric. Then for all such G (outside of 3 cases), $\psi \in Aut(G)$ induces an automorphism on parameter space $\mathbb{R}^m$, where the automorphism is given by permuting the coordinates of $\mathbb{R}^m$.*

Proof: For all graphs (excluding 3 cases that are outlined in Corollary 3.3 of [16]), the vertex automorphism group of a graph is isomorphic to the edge automorphism group of the graph (i.e. automorphisms of its line graph). Edge automorphisms are bijections from the edge set to itself satisfying certain conditions, and hence they induce bijections on the edge variables. □

Proposition 3.13. *Let P be a problem in the following set of parametric graph problems: End-to-end shortest path, shortest path tree, traveling salesman tour, all pairs shortest path, betweenness centrality, Katz centrality, eigenvector centrality. Let $h : \mathbb{R}^m \to J$ be the assignment for P. Then there is an action of $Aut(G)$ on J that commutes with the action of $Aut(G)$ on $\mathbb{R}^m$, i.e. such that $\psi(h^{-1}(x)) = h^{-1}(\psi(x))$ for all $\psi \in Aut(G)$.*

Proof: We will prove the results for all-pairs shortest path and for eigenvector centrality and comment that the other cases follow similarly.

Suppose that x is a solution to the end-to-end shortest path problem between vertex i and vertex j with respect to parameter vector $w = (w_1, w_2, \ldots, w_m)$. Then x is a path $p_{ij} = v_{i1}v_{i2}\ldots v_{il}$. $\psi \in Aut(G)$ acts on paths in G, by mapping p_{ij} to $\psi(p_{ij})$, where

$$\begin{aligned}\psi(p_{ij}) =& \psi(v_{i1})\psi(v_{i2})\ldots\psi(v_{il}) \\ =& v_{\psi(i1)}v_{\psi(i2)}\ldots v_{\psi(il)},\end{aligned}$$

where we have written the action in a way to emphasize the fact that graph automorphisms can be viewed as certain relabelings of the vertex set. We denote the induced action on labels as ψ. It is clear that p_{ij} is optimal for w iff ψp_{ij} is optimal for $(w_{\psi(1)}, w_{\psi(2)}, \ldots, w_{\psi(n)})$, which shows that $h^{-1}(\psi(x)) = \psi(h^{-1}(x))$ and proves the result for end-to-end shortest path. Now let x be a solution to all-pairs shortest path. Hence x is a union paths p_{ij} for each $i, j \in V$, and by defining the action of ψ to be the diagonal action on the union of these paths, we see that the same result holds for all-pairs shortest path.

For eigenvector centrality, note that the action of $\psi \in Aut(G)$ on G permutes the columns and rows of the adjacency matrix A of G. Hence we see if $Ax = \lambda x$ then $\psi(Ax') = \lambda x'$,

where x' is the vector with $x'_i = x_{\psi(i)}$. Thus we may define an induced action ψ on eigenvectors of A, and see that this commutes with the action of ψ on G. □

Corollary 3.14. *Let $v \in V$, and let $h(w) = \sigma_w$ denote the solution to any of the parameterized centrality problems in the above proposition, i.e. σ_w is the induced ordering from V to $[n]$. Then if $\sigma_w(v)$ is constant as a function of w, then v is fixed by all automorphisms of G.*

An example demonstrating the above corollary is the central vertex in the n-star graph. A similar result holds for solutions to the various routing problems in the Proposition 3.13, and more sophisticated statements relating the action of $Aut(G)$ to the symmetries of parameter space exist. In theory, incorporating these symmetries into algorithms such as the one described in [1] should result in faster computations, as it will only be necessary to compute a subspace of the total decomposition, and then use the action of $Aut(G)$ on parameter space to reconstruct the rest.

4. STABILITY OF SOLUTION SETS

Recall that $h : \mathbb{R}^m \to J$. Considering the geometry of $h^{-1}(x)$[13] gives rise to new perspectives for parametric problems on graphs. In some sense, the size and shape of the set $h^{-1}(x)$ corresponding to a solution $x \in J$ should reflect how "stable" to perturbation this optimal solution is. Consider for instance a convex solution set $h^{-1}(x) = U$ with large volume and "small" boundary measure. Then one can reasonably assume that small perturbations of most vectors in U will stay inside U. In contrast, if U has small volume, or has large volume but with a comparably large boundary (e.g. if U looks like a tree) then we lose such guarantees.

To make these ideas precise, we will need to appeal to the language of *measures*. Informally, a measure is a function that assigns a mass to a set. One of their primary applications is in probability theory, where a (normalized) measure on a space of possible events provides a way of assigning a probability to a subset of events occurring. With this in mind, consider the following definitions:

Definition 4.1. Let μ be a measure on a measurable space, and let $f : X \to Y$ be a measurable map. The pushforward measure $f_*(\mu)$ is defined by setting $f_*\mu(A) = \mu(f^{-1}(A))$ for all measurable $A \subset Y$.

Definition 4.2. Let μ be a compactly supported measure on $\mathbb{R}^m$, let G be a graph with m parameterized edges. Let $h : \mathbb{R}^m \to J$ be the assignment for a problem P with solution set J. Then we say $h_*\mu$ is the measured induced by h relative μ.

Before proceeding, we explain two natural ways one may obtain measures induced on parameter space in a networking setting. For concreteness, suppose our system of interest is modeled by n agents arranged on a graph with fixed topology, but possibly changing edge weights, which may represent for instance distance or available channel capacity.

Example 4.3. Suppose that our edge weights are subject to uncertainty, due to environmental factors or to account for the possibility transmission errors. This can be modeled by assigning a probability measure μ_i to the edge i, and letting $\mu = \prod_{i=1}^{m} \mu_i$. For instance, in the standard white noise model for communication channels, each μ_i is a one-dimensional Gaussian distribution, and μ is thus an m-dimensional Gaussian. μ induces a measure on each cell $h^{-1}(x)$ in our decomposition, corresponding to how often x is the optimal solution given the presence of uncertainty modeled by μ.

Example 4.4. Even in a completely deterministic system, time-evolving edge weights induce a measure on parameter space. For instance, suppose that our edge weights evolve periodically in time. We may view then view the evolution of this system as a closed loop $C : \mathbb{R} \to \mathbb{R}^m$ with period T in $\mathbb{R}^m$. This defines a measure μ_C on our solution set J, such that for any $x \in J$ we have

$$\mu_C(h^{-1}(x)) = \int_{h^{-1}(x)} C(t)dt,$$

i.e. the measure induced by the amount of time our system spends in the region $h^{-1}(x)$.

We now offer two different definitions of stability for a solution set $h^{-1}(x)$, relative to an induced measure $h_*\mu$.

Definition 4.5. Let $h : \mathbb{R}^m \to J$ be a parameterized problem on G, and let μ be a compactly supported measure on $\mathbb{R}^m$. Then for $x, y \in J$, we say that x is more stable than y if $h_*\mu(x) \geq h_*\mu(y)$. We call $x^* = argmax_{x \in J}\{h_*(\mu)(x)\}$ the maximally stable solution.

Definition 4.6. Let $h : \mathbb{R}^m \to J$ be a parameterized problem on G, and let μ be a compactly supported measure on $\mathbb{R}^m$. For each $x \in J$ and $\epsilon > 0$, let $W_\epsilon(x)$ be the subset of $h^{-1}(x)$ of all y such that $B_\epsilon(y) \subset h^{-1}(x)$. Then for $x, y \in J$, we say that x is more ϵ-stable than y if $\mu(W_\epsilon(x)) \geq \mu(W_\epsilon(y))$. $x^* = argmax_{x \in J}\{W_\epsilon(x)\}$ is the maximally ϵ-stable solution.

Stability describes the optimality of a solution up to uncertainty captured by the probability distribution μ. ϵ-stability also captures this, but additionally accounts for the possibility of a small perturbation of the inputs. This more closely mimics the stability described in the beginning of this section, in that sets with relatively small boundary measure are necessarily more stable. Notice that as $\epsilon \to 0$, ϵ-stability recovers stability.

Even for tropical graph problems, determining stability and ϵ-stability is computationally nontrivial. Exact solutions to this problem in full generality is at least as difficult as computing the volume of an n-dimensional polytope, which is known to be computationally challenging [17]. However, there are known to be more efficient approximation schemes (e.g. [18]), and it would be interesting to see how such schemes perform on computing stability or ϵ-stability from systems arising from data.

5. CONCLUSION AND FUTURE WORK

Tropical geometry offers a new potential avenue for approaching delay tolerant networking. The main potential comes from the fact that tropical geometry is a natural setting for studying optimization problems, many of which can be expressed tropically [19]. As demonstrated above, some normally intractable problems become feasible in the tropical

[13] Called the *fiber* of h at x.

setting. Moreover, tropical approaches have been verified to work for scheduling trains [12] which are similar to delay tolerant networks in many ways. One similarity is that trains may have to wait in certain stations, similar to how some bundles have to be buffered at certain nodes. Another similarity is that a train may travel along a path without end to end connectivity, similar to the nature of delay tolerant networks.

The algorithms and framework detailed in this paper extend the utility from trains and traditional linear optimization towards the temporal setting of DTNs, and can be used to analyze current and future networks. For example, given a network metric, one may ask for the optimal way to add another communication node.

Because tropical geometry lends itself to computation, including by FPGA, local algorithms that use a tropical-geometric approach to decision-making are feasible, offering a direct path to implementation.

We have also shown how the perspective of studying a parametric problem on a graph through the geometry of its parameter space bears fruit even outside the strict tropical framework. To our knowledge this perspective is underdeveloped in the temporal graph literature.

We end with suggestions for future directions:

Future Work

- Solving temporal networking problems could be made possible through the application of parametric graph optimization. One way to demonstrate this would be to simulate a space network using orbital analysis software, and then attempt to make routing decisions based on the state of the network over time.
- Determine stability of an optimal solution for mild perturbations of the associated graph in parameter space. How much error are we allowed to have in a given network to still meet a threshold quality of service?
- Study the case when edge weights are unpredictable, and instead when edge stability follows a probability distribution with respect to time. What modifications are needed to our approach for this case?
- Study what factors (e.g. bit rate, latency, network demand) are appropriate for analysis as parameters in the context of our approach.
- Finish off Verilog Tropical ALU by adding signed arithmetic, logical and arithmetic shifts, pipeline, flow control (branches, jumps), branch prediction, co-processors, and cache. Implement companion Python matrix multiplication programs for driving an FPGA with several instances of a Tropical ALU (CPU) onboard.
- Investigate the behavior along boundaries of cells in the single source shortest path case. Define a (co)sheaf over the cells and their boundaries that maps trees over these cells.
- Extend the Joswig implementation to higher parameter dimensions, arbitrary graphs, and the ability to solve for trees in time intervals when the parameters are given by arbitrary continuous functions in parameter space.

REFERENCES

[1] M. Joswig and B. Schröter, "Parametric shortest-path algorithms via tropical geometry," Nov 2020. [Online]. Available: https://arxiv.org/pdf/1904.01082.pdf

[2] J. A. Fraire, P. Madoery, S. Burleigh, M. Feldmann, J. Finochietto, A. Charif, N. Zergainoh, and R. Velazco, "Assessing Contact Graph Routing Performance and Reliability in Distributed Satellite Constellations," Jul. 2017. [Online]. Available: https://www.hindawi.com/journals/jcnc/2017/2830542/

[3] R. Short, R. Green, M. Moy, and B. Story, "What type of graph is a contact graph?" NASA, Glenn Research Center, Cleveland OH 44135, USA, Technical Memorandum Upcoming, 2021.

[4] D. Maclagan and B. Sturmfels, *Introduction to Tropical Geometry*, ser. Graduate Studies in Mathematics. American Mathematical Society, Providence, RI, 2015, vol. 161.

[5] J. Cleveland, B. Mallery, A. Hylton, and R. Short, "Planting a flag in the tropics the essential tropical geometric background for networking applications," Nov 2021. [Online]. Available: https://ntrs.nasa.gov/citations/20210022137

[6] W. Rudin, *Principles of Mathematical Analysis*. McGraw-Hill, Inc., 1953.

[7] E. G. H. Landau, *Foundations of Analysis*. Chelsea Publishing Company, 1951.

[8] H. A. Thurston, *The Number System*. Blackie & Son, Ltd., 1956.

[9] K. Knopp, *Theory and Application of Infinite Series*. Blackie & Son, Ltd., 1928.

[10] E. Hewitt and K. Stromberg, *Real and Abstract Analysis*. Pringer Publishing Co., Inc., 1965.

[11] M. Muldoon, "Tropical arithmetic and shortest paths," 2020. [Online]. Available: https://personalpages.manchester.ac.uk/staff/mark.muldoon/Teaching/DiscreteMaths/LectureNotes/TropicalShortestPaths.pdf

[12] T. van den Boom, "Model predictive scheduling of semi-cyclic discrete-even t systems using switching max-plus linear models," June 2018.

[13] M. A. Weiss, *Data Structures and Algorithm Analysis in Java*, 3rd ed. Pearson, 2011.

[14] G. G. Jørgen Bang-Jensen, *Digraphs: Theory, Algorithms and Applications*. Springer, 2002.

[15] F. Bloch, M. O. Jackson, and P. Tebaldi, "Centrality measures in networks," *Arxiv.org*, 2021. [Online]. Available: arxiv.org/abs/1608.05845

[16] L. Rodriguez, "Automorphism groups of simple graphs," 2014. [Online]. Available: https://www.whitman.edu/Documents/Academics/Mathematics/2014/rodriglr.pdf

[17] I. Bárány and Z. Füredi, "Computing the volume is difficult," *Discrete & Computational Geometry*, vol. 2, 1987. [Online]. Available: https://link.springer.com/article/10.1007/BF02187886

[18] M. Dyer, A. Frieze, and R. Kannan, "A random polynomial time algorithm for approximating the volume of convex bodies," *Journal of the ACM*, 1988. [Online]. Available: https://www.math.cmu.edu/~af1p/Texfiles/oldvolume.pdf

[19] M. Joswig, "Optimization and tropical geometry: Exercises and problems 1," April 2019. [Online]. Available: https://page.math.tu-berlin.de/~joswig/teaching/VL+PR-Optimization+and+Tropical+Geometry-SS19/problems1.pdf

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.

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.

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.

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).

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.

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.