Simplex Network Graphs

By Joseph Juma, May 6th 2023

Graphs of Nodes and Edges

A common mathematical structure is a “Graph”, which is comprised of vertices (a point), sometimes referred to as “nodes”, and edges connecting these vertices. Interestingly both Nodes and Edges are a the simplex for 0 dimensional and 1 dimensional spaces respectively. So common graphs can be considered to be a mathematical set of zero-dimensional (”0D”) and one-dimensional (”1D”) simplexes arranged in some arbitrary yet meaningful manner. However, what happens if we begin to add in higher dimensional simplexes?

For example, the two-dimensional (”2D”) simplex, a triangle introduces new questions as a potential graph primitive. First however we need to identify a difference in behavior between the existing simplexes - Edges and Nodes. In graphs Nodes and edges link one another together, but a single edge can only have at most two nodes it links to. On the other hand a given node can be connected to any number of edges except in the case where the type of Graph limits this behavior. How do we understand this behavior?

Well, let’s consider that a higher-dimensional simplex always has a number of lower-dimensional simplexes. A triangle is comprised of 3 vertices, and 3 edges. An edge is comprised of 2 vertices. A tetrahedron (the “3D” simplex) is comprised of 6 edges, 4 vertices and 4 triangles.

This means that to link a higher-dimensional simplex to other higher dimensional simplexes they must have an overlapping constituent element - a shared lower dimensional simplex. Given that graphs are often the 0D and 1D simplex, we often only see this with vertices. However let’s consider a 3D graph comprised of Triangles. It would be possible for an arbitrary number of triangles to share the same edge, as shown in image 1.1.

Image 1.1 - Shared Edge Mesh

This illustrates that given a graph comprised of simplexes of various dimensionality up to some positive integer ordinal dimension n, any n-dimensional simplex (”n-simplex”) may be connected to other n-simplexes by a shared simplex of lower (m) dimensionality (an ”m-simplex”). Further there is no arbitrary limit on the number of n-simplexes which share an m-simplex. As there are multiple m-dimensionalities (equal to n-1), we may define the set of m-dimensionalities for a given n value, denoted M. This is defined as shown Equation 1.1.

Equation 1.1

$$M = \{0, 1, ..., (n-1)\}|n$$

Therefore we can define the a simplex graph $S = \{n, E\}$ where n is the highest dimensional simplex found in the graph, and E is the set of simplex elements in the graph, being of any dimensionality in the range, $[0, n]$.

Mathematical Definition

A “simplex”(s) is a set with dimensionality (denoted either n or $dim(s)$) comprised of elements which are all lower-dimensionality simplexes.

$$s = \{S_i : \{s_{(i,0)}, s_{(i,1)}, ..., s_{(i,z_i-1)}\} | i = \{0, 1, ..., (n-1)\}\}$$

$$dim(s) = n = \#s$$

Given the value, z_i is the number of i dimensional simplexes comprising a higher dimensional simplex. The following are the z_i values for 1 to 3 dimensional simplexes (edges, triangles and tetrahedrons).

$$\{z_0 = 2\} | n = 1$$
$$\{z_0 = 3, z_1 = 3\} | n = 2$$
$$\{z_0 = 4, z_1 = 6, z_2 = 4\} | n = 3$$

It can be noted that the dimensionality of a simplex plus 1 ($n + 1$) is equal to the z_0 value for that simplex. The z_i for a given simplex of n-dimensionality and i-value is governed by the function $Z(n, i)$ with the definition,

$$Z(n, i) = \frac{\prod_{i=0}^{n} k + i}{!(i+1)}$$

This formula is a generalized representation of the triangle and tetrahedral expansions. [1,2]

Implications in Functional Graph Construction

Now that we have established the above defined relationship for simplex graphs in n-dimensions, there are a number of fascinating features arising from such graphs. For example: edge-weighted graphs can be generalized in a few manners:

1. An, “n-simplex weighted graph”, where each distinct simplex which is both not a 0-simplex and not also an element of a higher-dimensional simplex has an associated weight.
2. A “strict n-simplex weighted graph”, where only n-simplexes have an associated weight.
3. An “edge-weighted graph”, where all 1-simplexes (edges), and only edges have an associated weight. This form of graph behaves identical to an arbitrarily constructed 1D graph with homologous connectivity to any high-dimensional simplex graphs and so it is a special case.
4. A “[1,n] domain simplex weighted graph”, where all simplexes, including those which are members of other simplexes that are also not 0-simplexes (vertices) are assigned a weight.

These four cases do not comprise the only varieties of simplex weighting, but instead are specific cases which can be found from a parameterized logic set. The three parameterized axioms that form this logical set are:

- The range of dimensionalities whose associated simplexes are assigned a weight. This is denoted by $[x, y]$ where, $0 < x \leq n$ and, $x \leq y \leq n$.
- Whether simplexes which are elements of another simplex may be assigned a weight. This is denoted by b, which has a boolean value equal to either 0 (elements are not assigned a weight) or 1 (elements are assigned a weight).
- If only one dimensionality is given associated weights, denoted by $x = y$, then the 2nd axiom is automatically nullified. As such the b parameter, only needs specification when $x! = y$.

Given these three axioms it’s possible to parameterize the possible simplex-weighting logics on a graph via “Simplex Weighting Notation”, show in Equation 2.1.

Equation 2.1 - Simplex Weighting Notation

$$< x, y, b > |(|x - y| > 0)$$
$$< [x, y], b > |(|x - y| > 0)$$
$$< x > |(|x - y| = 0)$$

The logical notation which maps to each of the possible notations can then be written as shown in Equation 2.2

Equation 2.2 - Simplex Weighting Notation and Logical Notation

$$|x - y| > 0 \rightarrow < x, y, b >$$
$$|x - y| = 0 \rightarrow < x >$$

This notation can then be used to parameterize the different logic cases, such as the previous 4 varieties of simplex weighting. For example,

1. An, “n-simplex weighted graph” assigns every non-zero simplex ($x \geq 1$) up to an n-simplex ($y = n$) a weight value unless it is an element of another simplex ($b = 0$). This is represented by the notation, $< 1, n, 0 >$.
2. A, “strict n-simplex weighted graph” assigns only n-dimensional simplexes ($x = y, y = n$) an associated weight. Since $x = y$, there is no question as to sub-element assignment. Therefore this may be represented by the notation, $< n >$.
3. A, “edge-weighted graph” assigns only 1-dimensional simplexes (edges) an associated weight. This means ($x = y, x = 1$) and because $x = y$, there is no need to consider sub-element assignment. This yields the notation, $< 1 >$.
4. A, “[1,n] domain simplex weighted graph” associates a weight with every non-zero simplex ($x = 1, x! = y$) , even if they are sub-elements of another simplex ($b = 1$). The notation for this is, $< 1, n, 1 >$.

However, what if one wishes to know how many species of simplex-weighted graphs there are for a given dimensionality (n)? Or to define permutatively exactly what these species were? First let’s denote the case where $|x - y| = 0$, as $\Lambda_0(n)$ and the case of $|x - y| > 0$, as $\Lambda_1(n)$. The respective species counting functions (”species cardinality functions”) will likewise be denoted, $\Gamma_0(n)$ and $\Gamma_1(n)$. Given this, the set defining notation of possible species in both cases are defined as shown in Equation 2.3.

Equation 2.3 - Species Set Definitions

$$\Lambda_0(n) = \{< x >: x \in [1, n]\}$$
$$\Lambda_1(n) = \{< x, y, b >: x \in [1, (y - 1)], y \in [(x + 1), n]\}$$

The values of $\Lambda_0(n)$ are easily understood as just being the set of rules where there is one rule-set per dimensional ordinal between 1 and n. On the other hand, the $\Lambda_1(n)$ value is more complex as it is the combination of all possible values of (x,y) where $1 \leq x < y$, $x < y \leq n$ and each of the possible b values (1 and 0). An example of the $\Lambda_1(3)$ species is shown in Equation 2.4.

Equation 2.4 - 3D Simplex Weighting Species

$$\Lambda_1(3) = \{< 1, 2, 0 >, < 1, 2, 1 >, < 1, 3, 0 >, < 1, 3, 1 >,$$
$$< 2, 3, 0 >, < 2, 3, 1 >\}$$

If one wishes to calculate the size of these sets, the calculation for the $\Lambda_0(n)$ case is simply the size of the set which is equal to the dimension parameter ($\#\Lambda_0(n) = \Gamma_0(n) = n$). The cardinality for the $\Lambda_1(n)$ case is somewhat more complex though, as it is the non-overlapping combinations of possible (x,y) values multiplied by the number of possible b values (2). This results in the formula shown in Equation 2.5, which is also the edge-to-vertex formula for a polytope due to the algebraic and geometric relationship between those phenomenon. Further, both of the species cardinality functions together are shown in Equation 2.6.

Equation 2.5 - Cardinality of Simplex Weighting Species

$$\#\Lambda_1(n) = \frac{n(n - 1)}{2} * 2 = n(n - 1)$$

Equation 2.6 - Species Cardinality Functions

$$
\#\Lambda_0(n) = \Gamma_0(n) = n \\
\#\Lambda_1(n) = \Gamma_1(n) = n(n-1)
$$

These formula further allow the characterization of the simplex graphs via the combined functions in Equation 2.7. Please note the sets computed via, $\Lambda(n)$ are combined into a single super-set, and likewise the two scalars calculated for $\Gamma(n)$ are added together.

Equation 2.7 - Combined Simplex Graph Weighting Formula

$$
\Lambda(n) = \begin{cases} \{< x >: x \in [1, n]\} \\ \{< x, y, b >: x \in [1, (y-1)], y \in [(x+1), n], b \in (0, 1)\} \end{cases} \\
\Gamma(n, \delta) = \begin{cases} n \\ n(n-1) \end{cases}
$$

Example Usage

In this section a series of examples will be shown to illustrate the usage of the above formula. First, let’s consider the demonstration of these formula for 3-dimensional ($n = 3$) space. We can use n to calculate the species cardinality as shown in Equation 3.1.

Equation 3.1 - Species Cardinality in n = 3

$$
\Gamma(n) = n + n(n-1) \\
\Gamma(3) = 3 + 3(3-1) = 3 + 6 = 9
$$

and to define the set of possible species in Equation 3.2.

Equation 3.2 - Species Set in n = 3

$$
\Lambda(3) = \{\{< 1 >, < 2 >, < 3 >\}, \{< 1, 2, 0 >, < 1, 3, 0 >, < 2, 3, 0 >, < 1, 2, 1 >, < 1, 3, 1 >, < 2, 3, 1 >\}\} \\
\Lambda(3) = \{< 1 >, < 2 >, < 3 >, < 1, 2, 0 >, < 1, 2, 1 >, < 1, 3, 0 >, < 1, 3, 1 >, < 2, 3, 0 >< 2, 3, 1 >\}
$$

From this we can easily see that the calculation of set-cardinality was accurate for the species set by counting the combined number of elements in the set is equal to 9, the same value calculated by the cardinality function. A second and third example can respectively be done with the $n = 2$ and $n = 4$ simplex graph spaces.

Equation 3.3 shows the calculation of species and cardinality for $n = 4$.

Equation 3.3 - Species and Cardinality for n = 4

$$
\Gamma(4) = 4 + 4(4-1) = 4 + 12 = 16 \\
\Lambda(4) = \{< 1 >, < 2 >, < 3 >, < 4 >, < 1, 2, 0 >, < 1, 2, 1 >, < 1, 3, 0 >, < 1, 3, 1 >, < 1, 4, 0 >, < 1, 4, 1 >, < 2,
$$

And Equation 3.4 illustrates this same process for $n = 2$.

Equation 3.4 - Species and Cardinality for n = 2

$$
\Gamma(2) = 2 + 2(2-1) = 2 + 2 = 4 \\
\Lambda(2) = \{< 1 >, < 2 >, < 1, 2, 0 >, < 1, 2, 1 >\}
$$

Given these examples one should now be able to employ this formula to calculate the species for a given n-dimensional simplex graph.

Practical Implications

Given the abstract nature of the above formula, one might ask for a practical application of these techniques. The foremost example would be in the practical development of software which supports n-simplex structures. Consider you wish to develop a library that provides graphing for up-to 4-simplex systems. You would need to be able to consider the logic of each species before being able to devise a shared software interface that encapsulates each of their logics.

In this circumstance, given the above formula we know this means that you would need to define 16 different class behaviors. To aid the developer in doing this, the mechanical methodology of the $\Lambda(n)$ function may be employed to generate the notations which define the rules for the species. From there a programmer may consider and aptly architect such a solution - allowing them to even spot approaches across n-values to allow for the generalization of their library to multiple dimensional simplex graphs. Such an approach may prove invaluable in creating highly versatile software, or finding an approach which can construct species via smaller logical components.

Citations

1. Triangle Expansions, https://en.wikipedia.org/wiki/Triangular_number
2. Tetrahedral Expansions, https://en.wikipedia.org/wiki/Tetrahedral_number
3. Simplex, https://en.wikipedia.org/wiki/Simplex