CHAPTER 2

Mathematics

2.1 Introduction: Graphs as Matrices

This chapter describes the mathematics in the GraphBLAS standard. The GraphBLAS define a narrow set of mathematical operations that have been found to be useful for implementing a wide range of graph operations. At the heart of the GraphBLAS are 2D mathematical objects called matrices. The matrices are usually sparse, which implies that the majority of the elements in the matrix are zero and are often not stored to make their implementation more efficient. Sparsity is independent of the GraphBLAS mathematics. All the mathematics defined in the GraphBLAS will work regardless of whether the underlying matrix is sparse or dense.

Graphs represent connections between vertices with edges. Matrices can represent a wide range of graphs using *adjacency* matrices or *incidence* matrices. Adjacency matrices are often easier to analyze while incidence matrices are often better for representing data. Fortunately, the two are easily connected by the fundamental mathematical operation of the GraphBLAS: matrix-matrix multiply. One of the great features of the GraphBLAS mathematics is that no matter what kind of graph or matrix is being use, the core operations remain the same. In other words, a very small number of matrix operations can be used manipulate a very wide range of graphs.

The mathematics of the GraphBLAS will be described using a "center outward" approach. Initially, the most important specific cases will be described that are at the center of GraphBLAS. The conditions on these cases will then be relaxed to arrive at more general definition. This approach has the advantage of being more easily understandable and describing the most important cases first.

2.1.1 Adjacency Matrix: Undirected Graphs, Directed Graphs, Weighted Graphs

Given an adjacency matrix $\mathbf{A}$, if $\mathbf{A}(v_1, v_2) = 1$, then there exists an edge going from vertex v_1 to vertex v_2. Likewise, if $\mathbf{A}(v_1, v_2) = 0$, then there is no edge between v_1 and v_2. Adjacency matrices

have direction, which means that $\mathbf{A}(v_1, v_2)$ is not the same as $\mathbf{A}(v_2, v_1)$. Adjacency matrices can also have edge weights. If $\mathbf{A}(v_1, v_2) = w_{ij}$, and $w_{ij} \neq 0$, then the edge going from v_1 to v_2 is said to have weight w_{ij}. Adjacency matrices provide a simple way to represent the connections between vertices in a graph between one set of vertices and another. Adjacency matrices are often square and both out-vertices (rows) and the in-vertices (columns) are the same set of vertices. Adjacency matrices can be rectangular in which case the out-vertices (rows) and the in-vertices (columns) are different sets of vertices. Such graphs are often called bipartite graphs. In summary, adjacency matrices can represent a wide range of graphs, which include any graph with any set of the following properties: directed, weighted, and/or bipartite.

2.1.2 Incidence Matrix: Multi-Graphs, Hyper-Graphs, Multipartite Graphs

An incidence, or edge matrix $\mathbf{E}$, uses the rows to represent every edge in the graph and the columns represent every vertex. There are a number of conventions for denoting an edge in an incidence matrix. One such convention is to set $\mathbf{E}(i, v_1) = -1$ and $\mathbf{E}(i, v_2) = 1$ to indicate that edge i is a connection from v_1 to v_2. Incidence matrices are useful because they can easily represent multi-graphs, hyper-graphs, and multi-partite graphs. These complex graphs are difficult to capture with an adjacency matrix. A multi-graph has multiple edges between the same vertices. If there was another edge, j, from v_1 to v_2, this can be captured in an incidence matrix by setting $\mathbf{E}(j, v_1) = -1$ and $\mathbf{E}(j, v_2) = 1$. In a hyper-graph, one edge can go between more than two vertices. For example, to denote edge i has a connection from v_1 to v_2 and v_3 can be accomplished by also setting $\mathbf{E}(i, v_3) = 1$. Furthermore, v_1, v_2, and v_3 can be drawn from different classes of vertices and so $\mathbf{E}$ can be used to represent multi-partite graphs. Thus, an incidence matrix can be used to represent a graph with any set of the following graph properties: directed, weighted, multi-partite, multi-edge, and/or hyper-edge.

2.2 Matrix Definition: Starting Vertices, Ending Vertices, Edge Weight Types

The canonical matrix of the GraphBLAS has N rows and M columns of real numbers. Such a matrix can be denoted as

$$\mathbf{A} : \mathbb{R}^{N \times M}$$

The canonical row and and column indexes of the matrix $\mathbf{A}$ are $i \in I = \{1, \ldots, N\}$ and $j \in J = \{1, \ldots, M\}$, so that any particular value $\mathbf{A}$ can be denoted as $\mathbf{A}(i, j)$. [Note: a specific

GraphBLAS *implementation* might use IEEE 64 bit double precision floating point numbers to represent real numbers, 64 bit unsigned integers to represent row and column indices, and the compressed sparse rows (CSR) format or the compressed sparse columns (CSC) format to store the non-zero values inside the matrix.]

A matrix of complex numbers is denoted

$$\mathbf{A} : \mathbb{C}^{N \times M}$$

A matrix of integers $\{\ldots, -1, 0, 1, \ldots\}$ is denoted

$$\mathbf{A} : \mathbb{Z}^{N \times M}$$

A matrix of natural numbers $\{1, 2, 3, \ldots\}$ is denoted

$$\mathbf{A} : \mathbb{N}^{N \times M}$$

Canonical row and column indices are natural numbers $I, J : \mathbb{N}$. In some GraphBLAS implementations these indices could be non-negative integers $I = \{0, \ldots, N-1\}$ and $J = \{0, \ldots, M-1\}$.

For the GraphBLAS a matrix is defined as the following 2D mapping

$$\mathbf{A} : I \times J \to \mathbb{S}$$

where the indices $I, J : \mathbb{Z}$ are finite sets of integers with with N and M elements respectively, and $\mathbb{S} \in \{\mathbb{R}, \mathbb{Z}, \mathbb{N}, \ldots\}$ is a set of scalars. Without loss of generality matrices can be denoted

$$\mathbf{A} : \mathbb{S}^{N \times M}$$

If the internal storage format of the matrix needs to be indicated, this can be done by

$$\mathbf{A} : \mathbb{S}^{N \times M}_{\mathrm{CSC}} \qquad \text{or} \qquad \mathbf{A} : \mathbb{S}^{N \times M}_{\mathrm{CSR}}$$

A *vector* is a matrix where either $N = 1$ or $M = 1$. A column vector is denoted

$$\mathbf{v} = \mathbb{S}^{N \times 1}$$

A row vector is denoted

$$\mathbf{v} = \mathbb{S}^{1 \times M}$$

A scalar is a single element of a set $s \in \mathbb{S}$ and has no matrix dimensions.

2.3 Scalar Operations: Combining and Scaling Graph Edge Weights

The GraphBLAS matrix operations are built on top of scalar operations. The primary scalar operations are standard arithmetic addition (e.g., $1 + 1 = 2$) and multiplication (e.g., $2 \times 2 = 4$). The GraphBLAS also allow these scalar operations of addition and multiplication to be defined by the implementation or the user. To prevent confusion with standard addition and multiplication, $\oplus$ will be used to denote scalar addition and $\otimes$ will be use to denote scalar multiplication. In this notation, standard arithmetic addition and arithmetic multiplication of real numbers $a, b, c \in \mathbb{R}$, where $\oplus \equiv +$ and $\oplus \equiv \times$ results in

$$c = a \oplus b \qquad \Rightarrow \qquad c = a + b$$

and

$$c = a \otimes b \qquad \Rightarrow \qquad c = a \times b$$

Allowing $\oplus$ and $\otimes$ to be implementation (or user) defined functions enables the GraphBLAS to succinctly implement a wide range of algorithms on scalars of all different types (not just real numbers).

2.4 Scalar Properties: Composable Graph Edge Weight Operations

Certain $\oplus$ and $\otimes$ combinations over certain sets of scalars are particular useful because they preserve desirable mathematical properties such as associativity

$$(a \oplus b) \oplus c = a \oplus (b \oplus c) \qquad\qquad (a \otimes b) \otimes c = a \otimes (b \otimes c)$$

and distributivity

$$a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$$

Associativity, and distributivity are *extremely* useful properties for building graph applications because they allow the builder to swap operations without changing the result. They also increase opportunities for exploiting parallelism by the runtime.

Example combinations of $\oplus$ and $\otimes$ that preserve scalar associativity and distributivity include (but are not limited to) standard arithmetic

$$\oplus \equiv + \qquad \otimes \equiv \times \qquad a, b, c \in \mathbb{R}$$

max-plus algebras

$$\oplus \equiv \max \qquad \otimes \equiv + \qquad a, b, c \in \{-\infty \cup \mathbb{R}\}$$

min-max algebras

$$\oplus \equiv \max \qquad \otimes \equiv \min \qquad a, b, c \in [0, \infty]$$

finite (Galois) fields such as GF(2)

$$\oplus \equiv \text{xor} \qquad \otimes \equiv \text{and} \qquad a, b, c \in [0, 1]$$

and power set algebras

$$\oplus \equiv \cup \qquad \otimes \equiv \cap \qquad a, b, c \subset \mathbb{Z}$$

These operations also preserve scalar commutativity. Other functions can also be defined for $\oplus$ and $\otimes$ that do not preserve the above properties. For example, it is often useful for $\oplus$ or $\otimes$ to pull in other data such as vertex labels of a graph, such as the select2nd operation used in breadth-first search.

2.5 Matrix Properties: Composable Operations on Entire Graphs

Associativity, distributivity, and commutativity are very powerful properties of the GraphBLAS and separate it from standard graph libraries because these properties allow the GraphBLAS to be composable (i.e., you can re-order operations and know that you will get the same answer). Composability is what allows the GraphBLAS to implement a wide range of graph algorithms with just a few functions.

Let $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{S}^{N \times M}$, be matrices with elements $a = \mathbf{A}(i, j)$, $b = \mathbf{B}(i, j)$, and $c = \mathbf{C}(i, j)$. Associativity, distributivity, and commutativity of scalar operations translates into similar properties on matrix operations in the following manner.

Additive Commutativity Allows graphs to be swapped and combined via matrix element-wise addition without changing the result

$$a \oplus b = b \oplus a \qquad \Rightarrow \qquad \mathbf{A} \oplus \mathbf{B} = \mathbf{B} \oplus \mathbf{A}$$

where matrix element-wise addition is given by $\mathbf{C}(i, j) = \mathbf{A}(i, j) \oplus \mathbf{B}(i, j)$

Multiplicative Commutativity Allows graphs to be swapped, intersected, and scaled via matrix element-wsie multiplication without changing the result

$$a \otimes b = b \otimes a \qquad \Rightarrow \qquad \mathbf{A} \otimes \mathbf{B} = \mathbf{B} \otimes \mathbf{A}$$

where matrix element-wise (Hadamard) multiplication is given by $\mathbf{C}(i,j) = \mathbf{A}(i,j) \otimes \mathbf{B}(i,j)$

Additive Associativity Allows graphs to be combined via matrix element-wise addition in any grouping without changing the result

$$(a \oplus b) \oplus c = a \oplus (b \oplus c) \qquad \Rightarrow \qquad (\mathbf{A} \oplus \mathbf{B}) \oplus \mathbf{C} = \mathbf{A} \oplus (\mathbf{B} \oplus \mathbf{C})$$

Multiplicative Associativity Allows graphs to be intersected and scaled via matrix element-wise multiplication in any grouping without changing the result

$$(a \otimes b) \otimes c = a \otimes (b \otimes c) \qquad \Rightarrow \qquad (\mathbf{A} \otimes \mathbf{B}) \otimes \mathbf{C} = \mathbf{A} \otimes (\mathbf{B} \otimes \mathbf{C})$$

Element-Wise Distributivity Allows graphs to be intersected and/or scaled and then combined or vice-verse without changing the result

$$a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c) \qquad \Rightarrow \qquad \mathbf{A} \otimes (\mathbf{B} \oplus \mathbf{C}) = (\mathbf{A} \otimes \mathbf{B}) \oplus (\mathbf{A} \otimes \mathbf{C})$$

Matrix Multiply Distributivity Allows graphs to be transformed via matrix multiply and then combined or vice-verse without changing the result

$$a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c) \qquad \Rightarrow \qquad \mathbf{A}(\mathbf{B} \oplus \mathbf{C}) = (\mathbf{AB}) \oplus (\mathbf{AC})$$

where matrix multiply $\mathbf{C} = \mathbf{AB}$ is given by

$$\mathbf{C}(i,j) = \bigoplus_{k=1}^{L} \mathbf{A}(i,k) \otimes \mathbf{B}(k,j)$$

for matrices $\mathbf{A} : \mathbb{S}^{N \times M}$, $\mathbf{B} : \mathbb{S}^{M \times L}$, and $\mathbf{C} : \mathbb{S}^{N \times L}$

Matrix Multiply Associativity is another implication of scalar distributivity and allows graphs to be transformed via matrix multiply in any grouping without changing the result

$$a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c) \qquad \Rightarrow \qquad (\mathbf{AB})\mathbf{C} = \mathbf{B}(\mathbf{AC})$$

Matrix Multiply Commutativity In most circumstances $\mathbf{AB} \neq \mathbf{BA}$, but we should write down the cases when it does, because it is really useful when it does commute.

2.6 0-Element: No Graph Edge

Sparse matrices play an important role in GraphBLAS. Many implementations of sparse matrices reduce storage by not storing the 0 valued elements in the matrix. In adjacency matrices and incidence matrices the 0 element is equivalent to no edge between the vertices represented by the corresponding row and column. In most cases, the 0 element is standard arithmetic 0. The GraphBLAS also allows the 0 element to be defined by the implementation or user. This can be particularly helpful when combined with user defined $\oplus$ and $\otimes$ operations. Specifically, if the 0 element has certain properties with respect scaler $\oplus$ and $\otimes$ then sparsity of matrix operations can be managed efficiently. These properties are the additive identity

$$a \oplus 0 = a$$

and the multiplicative annihilator

$$a \otimes 0 = 0$$

Note: the above behavior of $\oplus$ and $\otimes$ with respect to 0 is a requirement for the GraphBLAS.

Example combinations of $\oplus$ and $\otimes$ that exhibit the additive identity and multiplicative annihilator are:

Standard arithmetic over the real numbers $a \in \mathbb{R}$, $\oplus \equiv +$, $\otimes \equiv \times$, $0 \equiv 0 \Rightarrow$
additive identity: $a \oplus 0 = a + 0 = a$
multiplicative annihilator: $a \otimes 0 = a \times 0 = 0$

Max-plus algebras over $a \in \{-\infty \cup \mathbb{R}\}$, $\oplus \equiv \max$, $\otimes \equiv +$, $0 \equiv -\infty \Rightarrow$
additive identity: $a \oplus 0 = \max(a, -\infty) = a$
multiplicative annihilator: $a \otimes 0 = a + -\infty = -\infty$

Min-max algebras over $a \in [0, \infty]$, $\oplus \equiv \min$, $\otimes \equiv \max$, $0 \equiv \infty \Rightarrow$
additive identity: $a \oplus 0 = \min(a, \infty) = a$
multiplicative annihilator: $a \otimes 0 = \max(a, \infty) = \infty$

The Galois field GF(2) over $a \in [0, 1]$, $\oplus \equiv \text{xor}$, $\otimes \equiv \text{and}$, $0 \equiv 0 \Rightarrow$
additive identity: $a \oplus 0 = \text{xor}(a, 0) = a$
multiplicative annihilator: $a \otimes 0 = \text{and}(a, 0) = 0$

Power set algebras over sets of integers $a \subset \mathbb{Z}$, $\oplus \equiv \cup$, $\otimes \equiv \cap$, $0 \equiv \emptyset \Rightarrow$
additive identity: $a \oplus 0 = a \cup \emptyset = a$
multiplicative annihilator: $a \otimes 0 = a \cap \emptyset = \emptyset$

2.7 Matrix Graph Operations Overview

The core of the GraphBLAS is the ability to perform a wide range of graph operations on diverse types of graphs with a small number of matrix operations:

Sparse Construct a **Sparse** matrix from row, column, and value triples. Implements graph construction from a set of starting vertices, ending vertices, and edge weights.

Find **Find** the row, column, and value triples corresponding to the non-zero elements in a sparse matrix.

Transpose Flips or **Transpose**s the rows and the columns of a sparse matrix. Implements reversing the direction of the graph. Can be implemented with **Find** and **Sparse**.

SpGEMM **Sp**arse **GE**energized **M**atrix **M**ultiply. Implements single-source breadth first search, multi-source breadth first search, weighted breadth first search.

SpRef **Sp**arse matrix index **Ref**erence. Implements sub-graph selection. Can be implemented with **Sparse** and **SpGEMM**.

SpAsgn **Sp**arse matrix index **Assign**. Implements sub-graph assignment. Can be implemented with **Sparse** and **SpGEMM**.

SpEWiseX **Sp**arse matrix **E**lement**W**ise binary operation. Implements graph union and intersection along with edge weight scale and combine.

Apply **Apply** unary operation to a sparse matrix. Implements graph edge weight modification. Can be implemented via **SpEWiseX**.

Reduce **Reduce** sparse matrix. Implements vertex degree calculations. Cam be implemented via **SpGEMM**.

The above set of functions has been shown to be useful for implementing a wide range of graph algorithms. These functions strike a balance between providing enough functions to be useful to an application builders and while being few enough that they can be implemented effectively. Furthermore, from an implementation perspective, there are only four functions that are truly fundamental: **Sparse**, **Find**, **SpGEMM**, and **SpEWwiseX**. The other GraphBLAS functions can be implemented from these functions four functions.

2.8 Sparse: Edge List to Graph

The GraphBLAS may use a variety of internal formats for representing sparse matrices. This data can often be imported as triples of vectors **i**, **j**, and **v** corresponding to the non-zero elements in the

sparse matrix. Constructing an NxM sparse matrix from triples can be denoted

$$\mathbf{A} = \mathbb{S}^{N \times M}(\mathbf{i}, \mathbf{j}, \mathbf{v}, \oplus)$$

where $\mathbf{i} : I^L$, $\mathbf{j} : J^L$, $\mathbf{i}, \mathbf{v} : \mathbb{S}^L$, are all L element vectors. The optional $\oplus$ function defines how multiple entries with the same row and column are handled. If $\oplus$ is undefined then the default is to combine the values using standard arithmetic addition $+$. Other variants include replacing any or all of the vector inputs with single element vectors. For example

$$\mathbf{A} = \mathbb{S}^{N \times M}(\mathbf{i}, \mathbf{j}, 1)$$

would use the value of 1 for input values. Likewise, a row vector can be constructed using

$$\mathbf{A} = \mathbb{S}^{N \times M}(1, \mathbf{j}, \mathbf{v})$$

and a column vector can be constructed using

$$\mathbf{A} = \mathbb{S}^{N \times M}(\mathbf{i}, 1, \mathbf{v})$$

The value type of the sparse matrix can be further specified via

$$\mathbf{A} : \mathbb{R}^{N \times M}(\mathbf{i}, \mathbf{j}, \mathbf{v})$$

2.9 Find: Graph to Vertex List

It is expected the GraphBLAS will need to send results to a other software components. Triples are a common interchange format. The GraphBLAS **find** command performs this operation by extracting the non-zero triples from a sparse matrix and can be denoted as

$$(\mathbf{i}, \mathbf{j}, \mathbf{v}) = \mathbf{A}$$

2.10 Transpose: Swap Start and End Vertices

Swapping the rows and columns of a sparse matrix is a common tool for changing the direction of vertices in a graph. The transpose is denoted as

$$\mathbf{B}(j, i) = \mathbf{A}^{\mathsf{T}}(i, j)$$

where $\mathbf{A} : \mathbb{S}^{N \times M}$ and $\mathbf{B} : \mathbb{S}^{M \times N}$

Transpose can be implemented using a combination of sparse and find as follows

$$(\mathbf{i}, \mathbf{j}, \mathbf{v}) = \mathbf{A}$$

$$\mathbf{B} = \mathbb{S}^{N \times M}(\mathbf{j}, \mathbf{i}, \mathbf{v})$$

2.11 SpGEMM: Weighted, Multi-Source, Breadth-First-Search

Matrix multiply is the most important operation in the GraphBLAS and can be used to implement a wide range of graph algorithms. In its most common form, SpGEMM performs a matrix multiply using standard arithmetic addition and multiplication

$$\mathbf{C} = \mathbf{AB}$$

or more explicitly

$$\mathbf{C}(i, j) = \sum_{k=1}^{L} \mathbf{A}(i, k)\mathbf{B}(k, j)$$

where $\mathbf{A} : \mathbb{R}^{N \times M}$, $\mathbf{B} : \mathbb{R}^{M \times L}$, and $\mathbf{C} : \mathbb{R}^{N \times L}$. **SpGEMM** has many important variants that include accumulating results, transposing inputs or outputs, and user defined addition and multiplication. These variants can be used alone or in combination. When these variants are combined with the wide range graphs that can be represented with sparse matrices, this results in many thousands of distinct graph operations that can be succinctly captured by multiplying two sparse matrices. As will be describe subsequently, all of these variants can be represented by the following mathematical statement

$$\mathbf{C}^{\mathsf{T}} \oplus= \mathbf{A}^{\mathsf{T}} \oplus.\otimes \mathbf{B}^{\mathsf{T}}$$

where $\mathbf{A} : \mathbb{S}^{N \times M}$, $\mathbf{B} : \mathbb{S}^{M \times L}$, and $\mathbf{C} : \mathbb{S}^{N \times L}$ and the symbols in blue represent optional SpGEMM operations that can be specified by the user.

2.11.1 Accumulation: Summing up Edge Weights

SpGEMM can be used to multiply and accumulate values into a matrix. One example is when the result of multiply $\mathbf{A}$ and $\mathbf{B}$ is added to the existing values in $\mathbf{C}$ (instead of replacing $\mathbf{C}$. This can be

written

$$\mathbf{C} \mathrel{+}= \mathbf{AB}$$

or more explicitly

$$\mathbf{C}(i,j) = \mathbf{C}(i,j) + \sum_{k=1}^{L} \mathbf{A}(i,k)\mathbf{B}(k,j)$$

2.11.2 Transposing Inputs or Outputs: Swapping Start and End Vertices

Another variant is to specify that the matrix multiply should be performed over the transpose of $\mathbf{A}$, $\mathbf{B}$, or $\mathbf{C}$.

Transposing the input matrix $\mathbf{A}$ implies

$$\mathbf{C} = \mathbf{A}^{\mathsf{T}}\mathbf{B}$$

or more explicitly

$$\mathbf{C}(i,j) = \sum_{k=1}^{L} \mathbf{A}(k,i)\mathbf{B}(k,j)$$

where $\mathbf{A} : \mathbb{R}^{M \times N}$.

Transposing the input matrix $\mathbf{B}$ implies

$$\mathbf{C} = \mathbf{A}\mathbf{B}^{\mathsf{T}}$$

or more explicitly

$$\mathbf{C}(i,j) = \sum_{k=1}^{L} \mathbf{A}(i,k)\mathbf{B}(j,k)$$

where $\mathbf{B} : \mathbb{R}^{L \times M}$.

Transposing the output matrix $\mathbf{C}$ implies

$$\mathbf{C}^{\mathsf{T}} = \mathbf{AB}$$

or more explicitly

$$\mathbf{C}(j,i) = \sum_{k=1}^{L} \mathbf{A}(i,k)\mathbf{B}(k,j)$$

where $\mathbf{C} : \mathbb{R}^{L \times N}$.

Other combinations include transposing both inputs $\mathbf{A}$ and $\mathbf{B}$

$$\mathbf{C} = \mathbf{A}^\mathsf{T}\mathbf{B}^\mathsf{T} \qquad \Rightarrow \qquad \mathbf{C}(i,j) = \sum_{k=1}^{L} \mathbf{A}(k,i)\mathbf{B}(j,k)$$

where $\mathbf{A} : \mathbb{R}^{M \times N}$ and $\mathbf{B} : \mathbb{R}^{L \times M}$; transposing both input $\mathbf{A}$ and output $\mathbf{C}$

$$\mathbf{C}^\mathsf{T} = \mathbf{A}^\mathsf{T}\mathbf{B} \qquad \Rightarrow \qquad \mathbf{C}(j,k) = \sum_{k=1}^{L} \mathbf{A}(k,i)\mathbf{B}(k,j)$$

where $\mathbf{A} : \mathbb{R}^{M \times N}$ and $\mathbf{C} : \mathbb{R}^{L \times N}$; and transposing both input $\mathbf{B}$ and output $\mathbf{C}$

$$\mathbf{C}^\mathsf{T} = \mathbf{A}\mathbf{B}^\mathsf{T} \qquad \Rightarrow \qquad \mathbf{C}(j,k) = \sum_{k=1}^{L} \mathbf{A}(i,k)\mathbf{B}(j,k)$$

where $\mathbf{B} : \mathbb{R}^{L \times M}$ and $\mathbf{C} : \mathbb{R}^{L \times N}$.

Normally, the transpose operation distributes over matrix multiplication $(\mathbf{AB})^\mathsf{T} = \mathbf{A}^\mathsf{T}\mathbf{B}^\mathsf{T}$ and so transposing both inputs $\mathbf{A}$ and $\mathbf{B}$ and the output $\mathbf{C}$ is rarely used. Nevertheless, for completeness, this operation is defined as

$$\mathbf{C}^\mathsf{T} = \mathbf{A}^\mathsf{T}\mathbf{B}^\mathsf{T} \qquad \Rightarrow \qquad \mathbf{C}(j,i) = \sum_{k=1}^{L} \mathbf{A}(k,i)\mathbf{B}(j,k)$$

where $\mathbf{A} : \mathbb{R}^{M \times N}$, $\mathbf{B} : \mathbb{R}^{L \times M}$, and $\mathbf{C} : \mathbb{R}^{L \times N}$.

2.11.3 Addition and Multiplication: Combining and Scaling Edges

Standard matrix multiplication on real numbers first performs scalar arithmetic multiplication on the elements and then performs scalar arithmetic addition on the results. The GraphBLAS allows the scalar operations of addition $\oplus$ and multiplication $\otimes$ to be replaced with user defined functions. This can be formally denoted as

$$\mathbf{C} = \mathbf{A} \oplus . \otimes \mathbf{B}$$

or more explicitly

$$\mathbf{C}(i,j) = \bigoplus_{k=1}^{L} \mathbf{A}(i,k) \otimes \mathbf{B}(k,j)$$

where $\mathbb{S}^{N \times M}$, $\mathbf{B} : \mathbb{S}^{M \times L}$, and $\mathbf{C} : \mathbb{S}^{N \times L}$. In this notation, standard matrix multiply can be written

$$\mathbf{C} = \mathbf{A} +.\times \mathbf{B}$$

where $\mathbb{S} \rightarrow \mathbb{R}$. Other matrix multiplications of interest include max-plus algebras

$$\mathbf{C} = \mathbf{A} \max.+ \mathbf{B}$$

or more explicitly

$$\mathbf{C}(i,j) = \max_k \{\mathbf{A}(i,k) + \mathbf{B}(k,j)\}$$

where $\mathbb{S} \rightarrow \{-\infty \cup \mathbb{R}\}$; min-max algebras

$$\mathbf{C} = \mathbf{A} \min.\max \mathbf{B}$$

or more explicitly

$$\mathbf{C}(i,j) = \min_k \{\max(\mathbf{A}(i,k), \mathbf{B}(k,j))\}$$

where $\mathbb{S} \rightarrow [0, \infty)$; the Galois field of order 2

$$\mathbf{C} = \mathbf{A} \text{ xor.and } \mathbf{B}$$

or more explicitly

$$\mathbf{C}(i,j) = \text{xor}_k \{\text{and}(\mathbf{A}(\text{i},\text{k}), \mathbf{B}(\text{k},\text{j}))\}$$

where $\mathbb{S} \rightarrow [0, 1]$; and power set algebras

$$\mathbf{C} = \mathbf{A} \cup.\cap \mathbf{B}$$

or more explicitly

$$\mathbf{C}(i,j) = \bigcup_k \mathbf{A}(i,k) \cap \mathbf{B}(k,j)$$

where $\mathbb{S} \rightarrow \{\mathbb{Z}\}$.

Accumulation also works with user defined addition and can be denoted

$$\mathbf{C} \ \oplus = \ \mathbf{A} \oplus.\otimes \mathbf{B}$$

or more explicitly

$$\mathbf{C}(i,j) = \mathbf{C}(i,j) \oplus \bigoplus_{k=1}^{L} \mathbf{A}(i,k) \otimes \mathbf{B}(k,j)$$

2.12 SpRef: Selecting Sub-Graphs

Selecting sub-graphs is a very common graph operation. The GraphBLAS performs this operation with the **SpRef** function by selecting starting vertices (row) and ending vertices (columns) from a matrix $\mathbf{A} : \mathbb{S}^{N \times M}$

$$\mathbf{B} = \mathbf{A}(\mathbf{i}, \mathbf{j})$$

or more explicitly

$$\mathbf{B}(i, j) = \mathbf{A}(\mathbf{i}(i), \mathbf{j}(j))$$

where $i \in \{1, ..., N_B\}$, $j \in \{1, ..., M_B\}$, $\mathbf{i} : I^{N_B}$, and $\mathbf{j} : J^{M_B}$ select specific sets of rows and columns in a specific order. The resulting matrix $\mathbf{B} : \mathbb{S}^{N_B \times M_B}$ can be larger or smaller than the input matrix $\mathbf{A}$. **SpRef** can also be used to replicate and/or permute rows and columns in a matrix.

SpRef can be implemented using sparse matrix multiply as

$$\mathbf{B} = \mathbf{P}(\mathbf{i})\ \mathbf{A}\ \mathbf{P}^{\mathsf{T}}(\mathbf{j})$$

where $\mathbf{P}(\mathbf{i})$ and $\mathbf{P}(\mathbf{j})$ are permutation matrices given by

$$\mathbf{P}(\mathbf{i}) = \mathbb{S}^{N_B \times N}(\{1, ..., N_B\}, \mathbf{i}, 1)$$

$$\mathbf{P}(\mathbf{j}) = \mathbb{S}^{M_B \times M}(\{1, ..., M_B\}, \mathbf{j}, 1)$$

2.13 SpAsgn: Modifying Sub-Graphs

Modifying sub-graphs is a very common graph operation. The GraphBLAS performs this operation with the **SpAsgn** function by selecting starting vertices (row) and ending vertices (columns) from a matrix $\mathbf{A} : \mathbb{S}^{N \times M}$ and assigning new values to them from another sparse matrix

$$\mathbf{A}(\mathbf{i}, \mathbf{j})\ \oplus\!\!=\ \mathbf{B}$$

or more explicitly

$$\mathbf{A}(\mathbf{i}(i), \mathbf{j}(j))\ \oplus\!\!=\ \mathbf{B}(i, j)$$

where $i \in \{1, ..., N_B\}$, $j \in \{1, ..., M_B\}$, $\mathbf{i} : I^{N_B}$ and $\mathbf{j} : J^{M_B}$ select specific sets of rows and columns in a specific order and $\oplus$ optionally allows $\mathbf{B}$ to added to the existing values of $\mathbf{A}$.

The additive form of **SpAsgn** can be implemented using sparse matrix multiply as

$$\mathbf{A}\ \oplus\!\!=\ \mathbf{P}^{\mathsf{T}}(\mathbf{i})\ \mathbf{B}\ \mathbf{P}(\mathbf{j})$$

where $\mathbf{P}(\mathbf{i})$ and $\mathbf{P}(\mathbf{j})$ are permutation matrices given by

$$\mathbf{P}(\mathbf{i}) = \mathbb{S}^{N_B \times N}(\{1, ..., N_B\}, \mathbf{i}, 1)$$

$$\mathbf{P}(\mathbf{j}) = \mathbb{S}^{M_B \times M}(\{1, ..., M_B\}, \mathbf{j}, 1)$$

2.14 SpEWiseX: Combining Graphs, Intersecting Graphs, Scaling Graphs

Combining graphs along with adding their edge weights can be accomplished by adding together their sparse matrix representations

$$\mathbf{C} \oplus= \mathbf{A}$$

or more explicitly

$$\mathbf{C}(i,j) = \mathbf{C}(i,j) \oplus \mathbf{A}(i,j)$$

where $\mathbf{A}, \mathbf{C} : \mathbb{S}^{N \times M}$, $i \in \{1, ..., N\}$, and $j \in \{1, ..., M\}$.

Intersecting graphs along with scaling their edge weights can be accomplished by element-wise multiplication of their sparse matrix representations

$$\mathbf{C} = \mathbf{A} \otimes \mathbf{B}$$

or more explicitly

$$\mathbf{C}(i,j) = \mathbf{A}(i,j) \otimes \mathbf{B}(i,j)$$

where $\mathbf{A}, \mathbf{B}, \mathbf{C} : \mathbb{S}^{N \times M}$.

SpEWiseX combines both of these operations into a single function

$$\mathbf{C} \oplus= \mathbf{A} \otimes \mathbf{B}$$

or more explicitly

$$\mathbf{C}(i,j) = \mathbf{C}(i,j) \oplus \mathbf{A}(i,j) \otimes \mathbf{B}(i,j)$$

where $\oplus$, $\otimes$, and $\mathbf{B}$ are optional arguments.

2.15 Apply: Modify Edge Weights

Modifying edge weights can be done by via the element-wise by unary function $f()$ to the values of a sparse matrix

$$\mathbf{C} \ \oplus= \ f(\mathbf{A})$$

or more explicitly

$$\mathbf{C}(i,j) = \mathbf{C}(i,j) \oplus f(\mathbf{A}(i,j))$$

where $\mathbf{A}, \mathbf{C} : \mathbb{S}^{N \times M}$, and $f(0) = 0$.

Apply can be implemented via **SpEWiseX** via

$$\mathbf{C} \ \oplus= \ \mathbf{A} \otimes \mathbf{A}$$

where $\otimes \equiv f()$ and $f(a,a) = f(a)$.

2.16 Reduce: Compute Vertex Degrees

It is often desired to combined all the weights of all the vertices that come out of the same starting vertices. This aggregation can be represented as a matrix product as

$$\mathbf{c} \ \oplus=\mathbf{A} \ \oplus.\otimes \ \mathbf{1}$$

or more explicitly

$$\mathbf{c}(i,1) = \mathbf{c}(i,1) \ \oplus \ \bigoplus_j \mathbf{A}(i,j)$$

where $\mathbf{c} : \mathbb{S}^{N \times 1}$ and $\mathbf{A} : \mathbb{S}^{N \times M}$, and $\mathbf{1} : \mathbb{S}^{N \times 1}$ is a column vector of all ones.

Likewise, combining all the weights of all the vertices that go into the same ending vertices can be represented as matrix product as

$$\mathbf{c} \ \oplus=\mathbf{1} \ \oplus.\otimes \ \mathbf{A}$$

or more explicitly

$$\mathbf{c}(1,j) = \mathbf{c}(1,j) \ \oplus \ \bigoplus_i \mathbf{A}(i,j)$$

where $\mathbf{c} : \mathbb{S}^{1 \times N}$ and $\mathbf{A} : \mathbb{S}^{N \times M}$, and $\mathbf{1} : \mathbb{S}^{1 \times N}$ is a row vector of all ones.