GraphBLAS

Peter M. Kogge

GraphBLAS References

- GraphBLAS Forum: http://graphblas.org/
- Overview:
  - https://resources.sei.cmu.edu/asset_files/Presentation/2016_017_001_474272.pdf
  - https://people.eecs.berkeley.edu/~aydin/GABB17.pdf
- Math Background:
  - http://www.mit.edu/~kepner/GraphBLAS/GraphBLAS-Math-release.pdf
- Tutorials:
  - http://faculty.cse.tamu.edu/davis/suitesparse_files/Davis_GraphBLAS_Oct2017.pdf
- V1.2 C API Spec: https://people.eecs.berkeley.edu/~aydin/GraphBLAS_API_C.pdf
- Suitesparse implementation: http://faculty.cse.tamu.edu/davis/suitesparse.html

Linear Algebra Review

- Computations involving "linear equations" with representation as matrices and vectors
- Core function: matrix multiplication
  - A is NxM matrix, B is MxR matrix
  - If C = AxB (also written just AB or A+.*B)
    - C[i, k] = A[i,1]*B[1,k] + A[i,2]*B[2,k] + … A[i,N]*B[N,k]
- Key observation: + & * need not be traditional math operators
- GraphBLAS observation: many graph algorithms expressible as matrix operations with different operators

Changing the Operators: Semirings

Rules of linear algebra hold whenever "*" and "+" are functions that form a **semi-ring**:

- + is **commutative**: a + b = b + a
- Both are **associative**:
  - a + (b + c) = (a + b) + c
  - a * (b * c) = (a * b) * c
- * **distributes** over +:
  - (a + b)*c = a*c + b*c
  - a*(b + c) = a*b + a*c
- Both are **monoids**, i.e. both have identities:
  - a + 0 = a ("0" is additive identity)
  - a*1 = a ("1" is multiplicative identity)
- Additive identity is multiplicative **annihilator**
  - 0*a = a*0 = 0
- Neither + nor * need have inverses

Lets call:
+ the reduction operator
*** the combination operator**

Graphs as Matrices

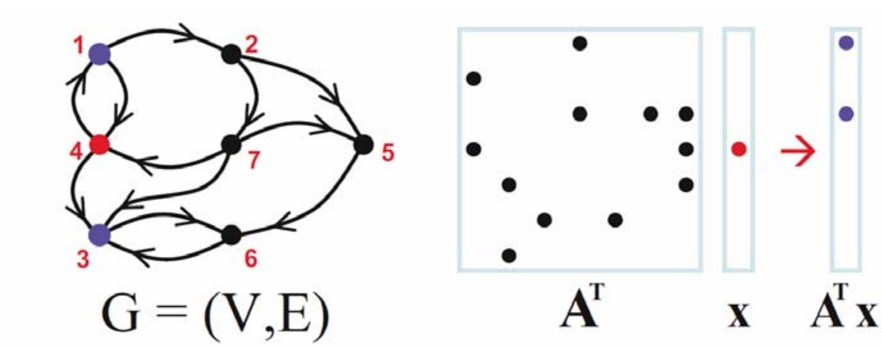

Figure 1.1. Matrix graph duality.
Adjacency matrix **A** is dual with the corresponding graph. In addition, vector matrix multiply is dual with breadth-first search.

One Step of BFS

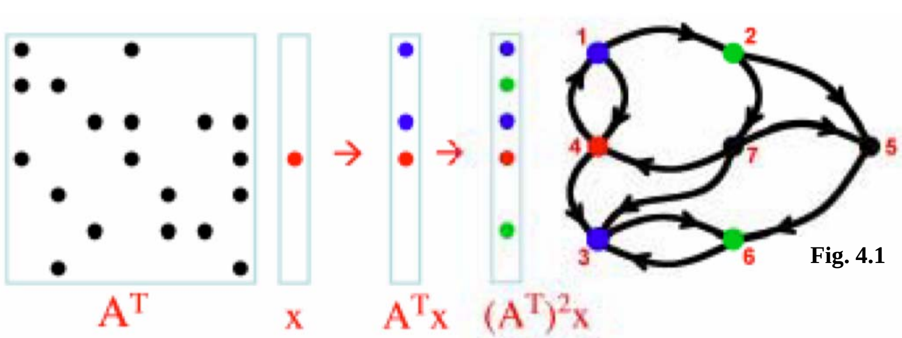

Fig. 4.1

- Matrix Domain: booleans
- + = OR
- * = AND

Minimum Paths

- Assume G has weighted edges (positive only)
- A is adjacency matrix but with ∞ for no edge
- $C_k[u,v]$ = min distance from u to v in *exactly* k steps
  - $C_1[u,v] = A$
- Now assume mat mult ◇ + = min, * = +
- $(A^{\diamond}A)[u,v] = \min_{w=1,N}(A[u,w]+A[w,v]) = A^{\diamond 2}$

  min distance from u to v thru w
- Thus $C_2 = A^{\diamond 2}$; $C_3 = A^{\diamond 3}$; …
- $\min_{i=0,\infty} C_i\ [u,v]$ = min distance from u to v

Useful Semi Rings

+: Reduction Operation			+: Reduction Operation			Sample Usage
Function	Domain	Identity	Function	Domain	Identity	
Normal Add	Ints, floats	0	Normal Multiply	Ints, floats	1	Linear Algebra
OR	Boolean	0	AND	Boolean	0	BFS
min	Ints, floats	∞	Normal Add	Ints, floats	0	Minimum paths

GraphBLAS

- C package to implement linear algebra
  - with different operators
  - and matrices that may be sparse
- Computations occur in "opaque space" separate from main
- API has several subsets of functionality
  - Define data types to use as matrix elements
  - Define new monoids and semi-rings
  - Transfer sparse data between main & opaque space
  - Perform linear matrix operations in opaque space

Table 2.1: GraphBLAS opaque objects and their types.

GrB_Object types	Description
GrB_Type	User-defined scalar type.
GrB_UnaryOp	Unary operator, built-in or associated with a single-argument C function.
GrB_BinaryOp	Binary operator, built-in or associated with a two-argument C function.
GrB_Monoid	Monoid algebraic structure.
GrB_Semiring	A GraphBLAS semiring algebraic structure.
GrB_Matrix	Two-dimensional collection of elements; typically sparse.
GrB_Vector	One-dimensional collection of elements.
GrB_Descriptor	Descriptor object, used to modify behavior of methods.

Table 2.2: Predefined GrB_Type values, the corresponding C type (for scalar parameters), and domains for GraphBLAS.

GrB_Type values	C type	domain
GrB_BOOL	bool	{false, true}
GrB_INT8	int8_t	$\mathbb{Z} \cap [-2^7, 2^7)$
GrB_UINT8	uint8_t	$\mathbb{Z} \cap [0, 2^8)$
GrB_INT16	int16_t	$\mathbb{Z} \cap [-2^{15}, 2^{15})$
GrB_UINT16	uint16_t	$\mathbb{Z} \cap [0, 2^{16})$
GrB_INT32	int32_t	$\mathbb{Z} \cap [-2^{31}, 2^{31})$
GrB_UINT32	uint32_t	$\mathbb{Z} \cap [0, 2^{32})$
GrB_INT64	int64_t	$\mathbb{Z} \cap [-2^{63}, 2^{63})$
GrB_UINT64	uint64_t	$\mathbb{Z} \cap [0, 2^{64})$
GrB_FP32	float	IEEE 754 binary32
GrB_FP64	double	IEEE 754 binary64

Predefined Monoids

(b) Predefined Operators.

Operator type	GraphBLAS identifier	Domains	Description	
GrB_UnaryOp	GrB_IDENTITY_T	$T \to T$	$f(x) = x,$	identity
GrB_UnaryOp	GrB_AINV_T	$T \to T$	$f(x) = -x,$	additive inverse
GrB_UnaryOp	GrB_MINV_T	$T \to T$	$f(x) = \frac{1}{x},$	multiplicative inverse
GrB_UnaryOp	GrB_LNOT	bool $\to$ bool	$f(x) = \neg x,$	logical inverse
GrB_BinaryOp	GrB_LOR	bool $\times$ bool $\to$ bool	$f(x, y) = x \vee y,$	logical OR
GrB_BinaryOp	GrB_LAND	bool $\times$ bool $\to$ bool	$f(x, y) = x \wedge y,$	logical AND
GrB_BinaryOp	GrB_LXOR	bool $\times$ bool $\to$ bool	$f(x, y) = x \oplus y,$	logical XOR
GrB_BinaryOp	GrB_EQ_T	$T \times T \to$ bool	$f(x, y) = (x == y)$	equal
GrB_BinaryOp	GrB_NE_T	$T \times T \to$ bool	$f(x, y) = (x \neq y)$	not equal
GrB_BinaryOp	GrB_GT_T	$T \times T \to$ bool	$f(x, y) = (x > y)$	greater than
GrB_BinaryOp	GrB_LT_T	$T \times T \to$ bool	$f(x, y) = (x < y)$	less than
GrB_BinaryOp	GrB_GE_T	$T \times T \to$ bool	$f(x, y) = (x \geq y)$	greater than or equal
GrB_BinaryOp	GrB_LE_T	$T \times T \to$ bool	$f(x, y) = (x \leq y)$	less than or equal
GrB_BinaryOp	GrB_FIRST_T	$T \times T \to T$	$f(x, y) = x,$	first argument
GrB_BinaryOp	GrB_SECOND_T	$T \times T \to T$	$f(x, y) = y,$	second argument
GrB_BinaryOp	GrB_MIN_T	$T \times T \to T$	$f(x, y) = (x < y)\ ?\ x : y,$	minimum
GrB_BinaryOp	GrB_MAX_T	$T \times T \to T$	$f(x, y) = (x > y)\ ?\ x : y,$	maximum
GrB_BinaryOp	GrB_PLUS_T	$T \times T \to T$	$f(x, y) = x + y,$	addition
GrB_BinaryOp	GrB_MINUS_T	$T \times T \to T$	$f(x, y) = x - y,$	subtraction
GrB_BinaryOp	GrB_TIMES_T	$T \times T \to T$	$f(x, y) = xy,$	multiplication
GrB_BinaryOp	GrB_DIV_T	$T \times T \to T$	$f(x, y) = \frac{x}{y},$	division

Suffix	C type
BOOL	bool
INT8	int8_t
UINT8	uint8_t
INT16	int16_t
UINT16	uint16_t
INT32	int32_t
UINT32	uint32_t
INT64	int64_t
UINT64	uint64_t
FP32	float
FP64	double

T may be any of suffixes on right table

Defining New Operations

- **GrB Type_new**: creates new type for values from a type known to the application program
- **GrB UnaryOp_new**: defines a new unary
  - Includes function pointer to a C function
- **GrB BinaryOp_new**: defines a new binary operator
  - Includes function pointer to a C function
- **GrB Monoid_new**: specifies a previously defined binary operator to be a monoid
  GrB Semiring_new: specifies a pair of operators as a new semiring.

Data Structures

- GraphBLAS matrices assumed to be "sparse"
- Unspecified elements are **structural zeros**
  - Additive identity/multiplicative annihilator from semiring
- Objects created as vectors or matrices
  - Have a "size" but when created all structural zeros
- **Context element**: object created in opaque space
- **Index array**: contiguous list of 64b uints
  - Used as indices into a vector/matrix
- **Mask**: contiguous list of bools
  - If x a mask, x[i]=0 => ith element is structural zero

Creating New Objects

- All created in opaque space
  - With user-invisible internal representation
- **GrB_xxx_new**: creates space for new object (vector or matrix) of some, but does not assign values
  - xxx is vector or matrix
- **GrB_xxx_dup**: duplicates some object.
- **GrB_free** frees all storage associated with an object
- Functions available to return properties of an object

Caller to Context Memory Transfers

- Move data from caller's space to opaque space
- **GrB_xxx_clear**: removes all elements from an object in a context element.
- **GrB_xxx_setElement**: takes (index, value) pair (for a vector) or a double index and value (for a matrix) from application space, and changes entry in the context vector
- **GrB_xxx_build**: takes one or two vectors of indices and a vector of values from application space, and updates corresponding entries in a context object.

Context to Caller Memory Transfers

- Move data from opaque space to caller's space
- **GrB_xxx_extractElement**: returns value associated with a specified index in an object in context space.
- **GrB_xxx_extractTuples**: stores in caller's space two (or three for a matrix) equal-length vectors whose contexts are all the indices from the context object that are not structural zeros, and all the corresponding values.

GraphBLAS Operations

- Performed on opaque objects, with values returned to opaque space
- May have options:
  - Accumulation function: equivalent to C "+=+
  - Mask: specify which elements of target are allowed to be modified
  - Descriptor: optionally modify the execution of the called operation

Method	Target	Arg 1	Arg 2	Arg 3	Description	Page
mxm	C: matrix	A: matrix	B: matrix		$C[i,j] = A[i,*] \ \oplus . \otimes \ B[*,j]$	69
vxm	W: vector	U: vector	A: matrix		$W[j] = U \oplus . \otimes \ A[*,j]$	73
mxv	W: vector	A: matrix	U: vector		$W[i] = A[i,*] \oplus . \otimes \ U$	77
eWiseMult	W: vector	U: vector	V: vector		$W[i] = U[i] \ \otimes \ V[i]$	82
eWiseMult	C: matrix	A: matrix	B: matrix		$C[i,j] = A[i,j] \ \otimes \ B[i,j]$	86
eWiseAdd	W: vector	U: vector	V: vector		$W[i] = U[i] \ \oplus \ V[i]$	91
eWiseAdd	C: matrix	A: matrix	B: matrix		$C[i,j] = A[i,j] \ \oplus \ B[i,j]$95	
extract	W: vector	U: vector	I: Index		$W[i] = (U[I])[i]$	100
extract	C: matrix	A: matrix	I_R: index	I_C: index	$C[i,j] = (A[I_R, I_C])[i,j]$	104
extract	W: vector	A: matrix	I_R: index	J: uint	$W[i] = (A[I_R, J])[i]$	110
assign	W:vector	U: vector	I: index		$(W[I])[i] = \ U[i]$	114
assign	C: matrix	A: matrix	I_R: index	I_C: index	$(C[I_R, I_C])[i,j] = A[i,j]$	119
assign	C: matrix	U: vector	I_R: index	J: int	$(C[I_R, J])[i] = U[i]$	124
assign	C: matrix	U: vector	I: int	I_C: index	$(C[I, I_C])[i] = U[i]$	129
assign	W: vector	v: value	I: index		$W[I] = v$	133
assign	C: matrix	v: value	I_R: index	I_C: index	$C[I_R, I_C] = v$	138
apply	W: vector	f: function	U: vector		$W[i] = f(U[i])$	143
apply	C: matrix	f: function	A: matrix		$W[i,j] = f(A[i,j])$	147
reduce	W: vector	f: function	A: matrix		$W[i] = \ f/A[i,*]$	147
reduce	v: variable	f: function	U: vector		$v = \ f/U$	151
reduce	v: variable	f: function	A: matrix		$v = \ f/f/A[i,]$	155
transpose	C: matrix	A: matrix			$C = \ A^T$	160

Page numbers are relative to "The GraphBLAS C API Spec," Version 1.0.0, 05/29/2017
"Vectors" and "matrices" are in the GraphBLAS context.
An "index" is an array of 64b uints in caller's memory.
A "value" or "variable" is a scalar in the caller's memory.
"M" is an optional write index set used as a mask on the target.
"$\oplus$" is the additive operator from the specified semiring.
"$\otimes$" is the multiplicative operator from the specified semiring.
"$\oplus.\otimes$" is an inner product using the operators from the semiring.
"$\odot$" is an optional accumulating operator.
An operator by itself is applied element-by-element
The expression "f/" refers to the summation across all values in the operand using function f.

Execution Model

- Steps in executing GraphBLAS call
  - **Initialization**: check arguments for validity and consistency, and access from caller's space any values needed during the rest of the call.
  - **Computation**: all computation required by call on the opaque objects carried out, and values saved into opaque space
  - **Materialization**: When called for, data transferred back to the caller's space
- Two modes of execution:
  - **Blocking**: each GraphBLAS call must complete in opaque space before control returned to caller
  - **Non-blocking**: caller may resume before operation completed in opaque space
    - Only Initialization guaranteed to complete before return