SPARSE Project Report

Andrea Pelosi

In this document we present *SPARSE*, a Julia package developed as part of the course IN480 - Parallel and Distributed Computing. In *SPARSE*, we port and benchmark a graph algorithm built upon GraphBLAS, a mathematical abstraction to represent and operate with graphs using just a narrow set of linear algebra primitives.

SPARSE source code can be found at *sparse.jl*.

Preliminaries

Why GraphBLAS?

Contemporary computer architectures are good at processing linear and hierarchical data structures, such as lists, stack or trees. They do not behave equally well with graphs. Graph algorithms are indeed challenging to program: issues such as poor locality, frequent cache misses and difficulty of parallelization demolish graph algorithms performance, and unfortunately optimizations reduce portability. In order to address these problems, a group of graph algorithms researchers came up with GraphBLAS.

GraphBLAS aims to set standard building blocks for graph representation and computation in the language of linear algebra. Graph algorithms were already expressed in terms of operations on *sparse* matrices (matrices in which the majority of the elements are zero) but GraphBLAS takes a step further trying to standardize the approach. Amongst the many improvements GraphBLAS could lead, we point out efficiency, portability, compactness, composability and flexibility.

To appreciate these advantages, we first introduce the mathematics upon which GraphBLAS is built. Notice though that the following introduction is far from comprehensive: we describe only what is relevant to fully understand *SPARSE*, along with some additional examples.

Theoretical foundations

[add references]

Matrices as graph displaying tools

A graph G is a pair $G = (V, E)$ where V and E are sets whose elements are called vertices and edges respectively. Using matrices, it is possible to display graphs in different ways. One of the most common matrix representations for graphs is the *adiacency matrix*.

Adiacency matrix Given a graph $G = (V, E)$ where $|V| = n$, its *adiacency matrix* A is a matrix with n rows and n columns where $A(i, j) = 1$ if there is an edge going from the vertex i to the vertex j and $A(i, j) = 0$ otherwise. If G is a *weighted graph* (namely a graph in which every edge has weight), $A(i, j) = w$ if there is an edge going from i to j that weights w.

$A(i, j)$ denotes the element in the ith row and the jth column.

The canonical GraphBLAS matrix has m rows and n columns and is defined by the following mapping:

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

where $I, J \subseteq \mathbb{Z}$ sets of indices with m and n elements respectively and $\mathbb{S}$ a set of scalars, with $\mathbb{S} \in \{\mathbb{Z}, \mathbb{R}, \mathbb{C}, \dots\}$.

Thus, without loss of generality, we can denote a GraphBLAS matrix as:

$$\mathbf{A} : \mathbb{S}^{m \times n}$$

A GraphBLAS column vector is a GraphBLAS matrix where $n = 1$; a GraphBLAS row vector is a GraphBLAS matrix where $m = 1$. For the sake of brevity we shall refer to matrices and vectors instead of GraphBLAS matrices and GraphBLAS vectors. Notice that, even though vectors and matrices are usually sparse, the correctness of the GraphBLAS model is independent from such sparsity.

GraphBLAS fundamental building block

The most important GraphBLAS operation is matrix-matrix multiplication, since it allows to build a large variety of graph algorithms. Traditional matrix-matrix multiplication requires the use of arithmetic sum and product. Using different kinds of operators and different domains while working with matrices, allows to further extend the variety of graph operations that could be written. This extended matrix-matrix multiplication is performed on an algebraic *semiring*:

Semiring Given a set D and two binary operations $\oplus$ and $\otimes$ (called addition and multiplication respectively), a *semiring* is a triple $(D, \oplus, \otimes)$ where:

- $(D, \oplus)$ is a commutative monoid and 0 is the additive identity element;
- $(D, \otimes)$ is a monoid and 1 is the multiplicative identity element;
- Multiplication distributes over addition;
- The element 0 is a multiplicative annihilator.

One of the main advantages GraphBLAS provides is that $\oplus$ and $\otimes$ could be user-defined. Here there are some semirings examples widely used with graph algorithms (from [add reference]):

Semiring	$\oplus$	$\otimes$	domain	**0**	**1**
Standard arithmetic	$+$	$\times$	$\mathbb{R}$	0	1
max-plus algebras	max	$+$	$\{-\infty \bigcup \mathbb{R}\}$	$-\infty$	0
Galois fields	xor	and	$\{0, 1\}$	0	1
Power set algebras	$\bigcup$	$\bigcap$	$\mathscr{P}(\mathbb{Z})$	$\emptyset$	U

Most common operations overview

GraphBLAS objects

SuiteSparse:GraphBLAS and SuiteSparse:GraphBLAS.jl

Algorithm description