Our **linear algebraic approach** to graph algorithms provides a flexible framework that enables high performance code generation for **faster network analysis**.

### Introduction

Graph algorithms can be expressed as sequences of linear, algebra-like operations through the use of the adjacency matrix. Adjacency matrices are used to represent graphs instead of vertices and edges.

```latex
\begin{array}{ccccccc}
A & B & C & D \\
\hline
1 & 2 & & & & \\
& & 3 & & & \\
& & & 4 & & \\
& & & & 5 & 6 \\
\end{array}
```

Our work extends the use of linear algebra beyond simple graph traversal.

### Writing Graph Algorithms in LA

Many graph primitives can be cast in terms of LA operations.

- Neighbors of a vertex $v$—vector matrix products.
- Filtering—Hadamard products.
- Semi-rings—to represent Matrix Multiplication like operation (Single-Source Shortest Path uses the min-plus semiring).

This mapping gets rid of the need for “experts” to formulate graph algorithms in LA.

- Formally derived algorithms

### Families of Graph Algorithms

Multiple graph algorithms can be enumerated for the same specification.

Our LA approach uses triangle counting:

$$\Delta = 1/6 \Gamma(A^3)$$

This approach allows us to analyze graph algorithms for individual performance characteristics. The algorithm that best-fit the situation is chosen.

### Busting Myths about Linear Algebraic Approach

Unifying Edge and Vertex Centric Algorithms

$$A_0 \quad A_1 \quad A_2 \quad A_3$$

Both classes of algorithms can be expressed using a single linear algebraic framework.

### Depth-First Search in Linear Algebra

First look at expressing depth-first traversal of graphs in LA.

### References
