Graph Clustering with GraphBLAS

Cameron Quilici

Texas A&M University

TEXAS A&M UNIVERSITY
Department of Computer Science & Engineering

Background

- A **graph** is a pair $G = (V, E)$ with V a set of **vertices** and $E \subseteq \{\{x, y\} \ : \ x, y \in V, x \neq y\}$ a set of **edges**.
- A **directed graph** is a graph whose edges have orientation and can be expressed as $G = (V, E)$ with $E \subseteq \{(x, y) \ : \ (x, y) \subseteq V \times V\}$.
- Every finite graph may be expressed as an **adjacency matrix** $\mathbf{A} \in \mathbb{R}^{n \times n}$ where

$$\mathbf{A}(i, j) = \begin{cases} 1 & \text{if } v_i v_j \in E \\ 0 & \text{otherwise.} \end{cases}$$

- A **clustering** $\mathcal{C}$ of a graph G with n vertices is a collection of k disjoint subgraphs such that $1 \leq k \leq n$.

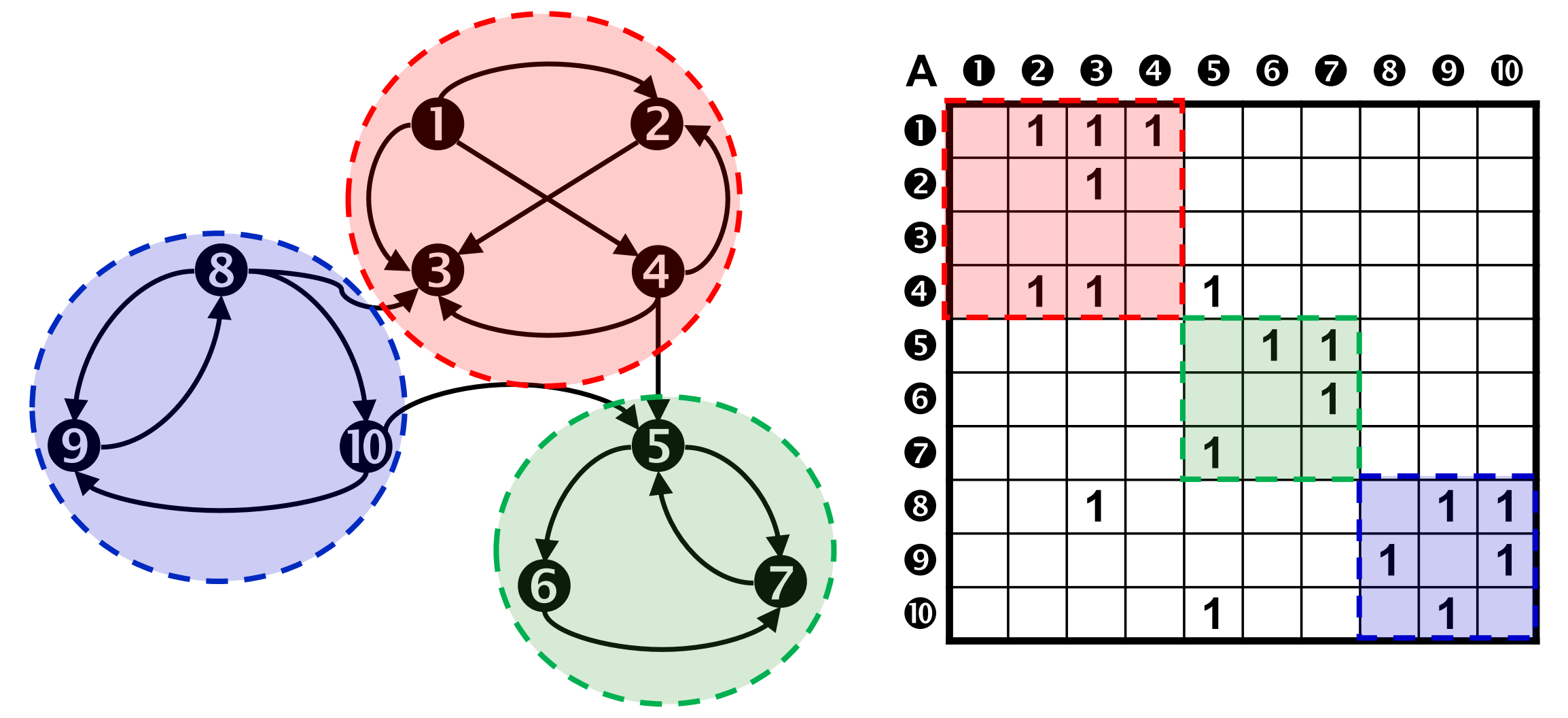


Figure 1. The directed graph (left) G_1 along with its adjacency matrix representation (right). A possible clustering $\mathcal{C}_1 = \{C_1, C_2, C_3\}$ of G_1 is shown.

- Matrix and vector multiplication of adjacency matrices can translate to graph operations.
  - For instance, in the graph above, $\mathbf{A}^k$ has the property that $\mathbf{A}(i, j) = x$ means there exist x paths of length k from vertex i to vertex j.
- Not all graph operations can be realized with traditional matrix multiplication. Instead, use arbitrary **semirings**.
- $\langle D, \oplus, \otimes, 0 \rangle$ is a GraphBLAS **semiring** if: (1) $\langle D, \oplus, 0 \rangle$ is a commutative monoid and (2) $\otimes$ is a closed binary operator.

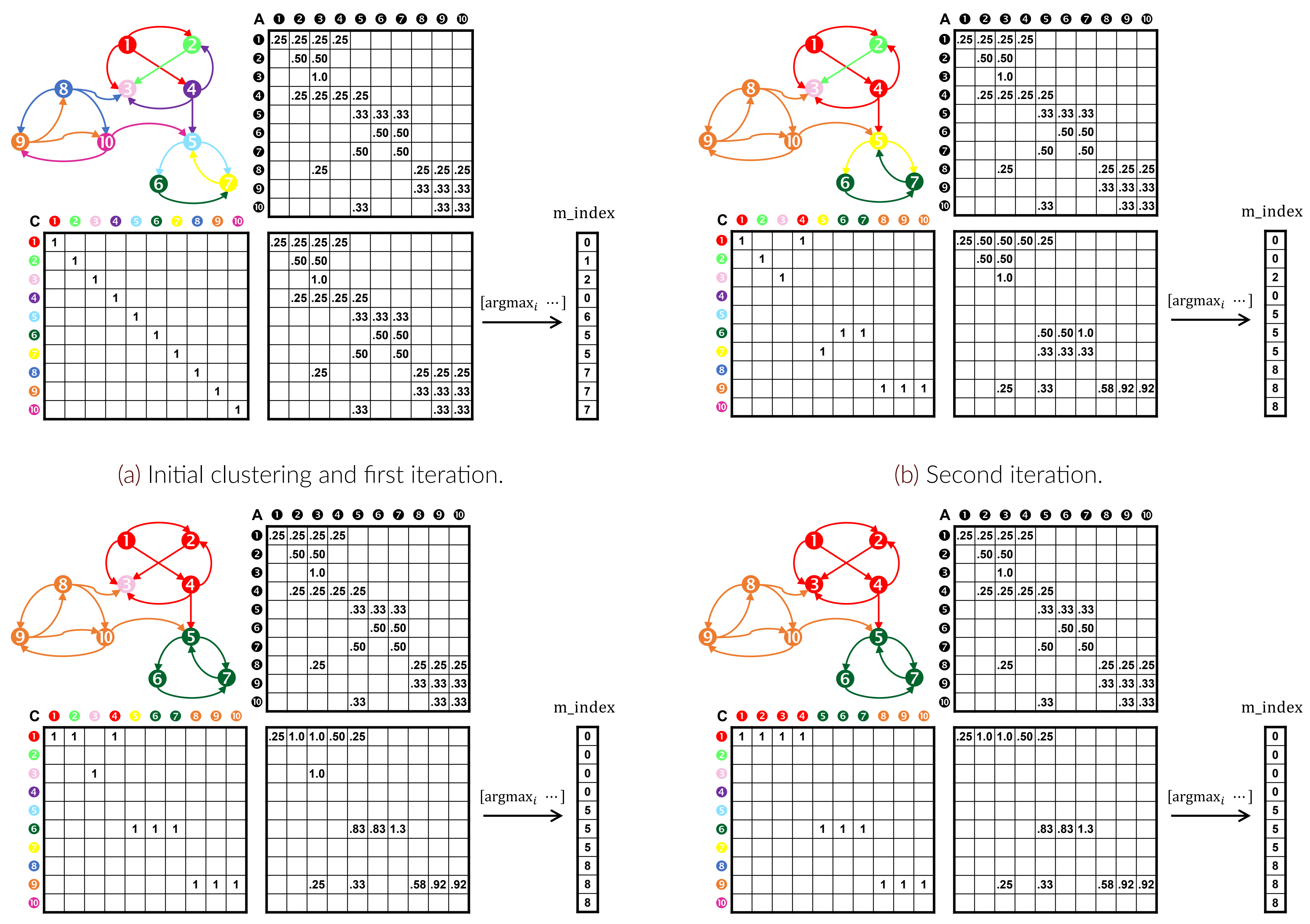
$$\mathbf{C} = \mathbf{AB} \iff \mathbf{C}(i, j) = \sum_{k=1}^{n} \mathbf{A}(i, k) \cdot \mathbf{B}(k, j) \quad \text{(Traditional)}$$

$$\mathbf{C} = \mathbf{A} \oplus . \otimes \mathbf{B} \iff \mathbf{C}(i, j) = \bigoplus_{k=1}^{n} \mathbf{A}(i, k) \otimes \mathbf{B}(k, j). \quad \text{(Arbitrary)}$$

Problem Statement

- The **GraphBLAS standard** formalizes the notion of **graph algorithms as linear algebraic operations** by providing a set of well-defined matrix and vector operations based on semirings [1]. In other words, the standard aims to provide a consistent set of "building blocks" which can be used to create graph algorithms in the language of linear algebra.
- **SuiteSparse:GraphBLAS** is the first complete implementation of the GraphBLAS C standard.
- We seek to implement the following graph clustering algorithms and cluster quality functions using SuiteSparse:GraphBLAS:
  - Peer Pressure Clustering (PPC)
  - Markov Cluster Algorithm (MCL)
  - Quality metrics: Performance, Coverage, and Modularity (Q)

Peer Pressure Implementation



(a) Initial clustering and first iteration.

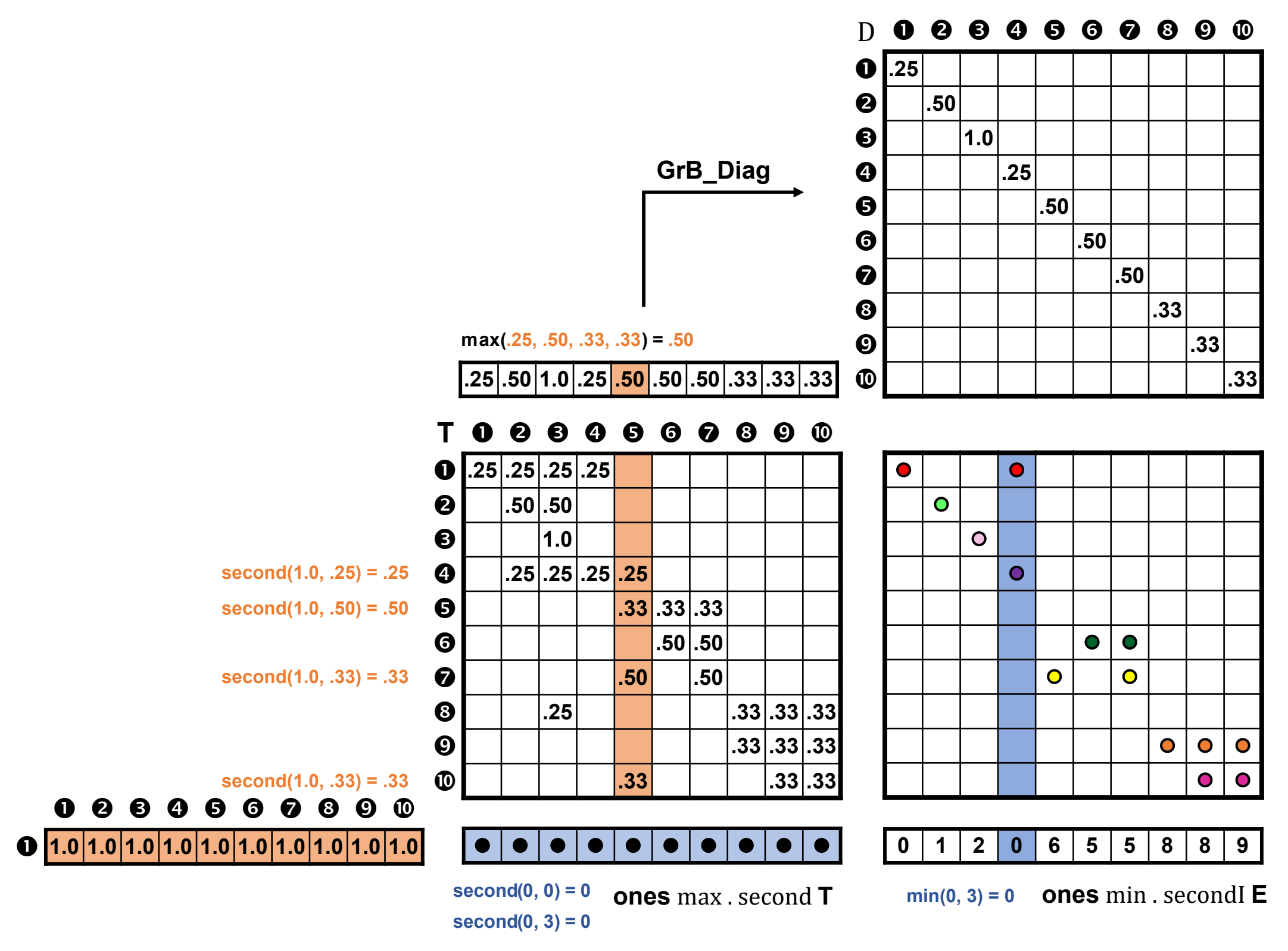
(b) Second iteration.

(c) Third iteration.

(d) Fourth iteration and final clustering.

Figure 2. Example of the peer pressure clustering algorithm on the working example [2]. Though not shown, each vertex has a self-edge.

```
C = I
while (True)
    T = C (plus,second) A
    m = ones (max,second) T
    D = diag (m)
    E = T (any,eq) D
    m_index = ones (min,secondI) E
    C_new = I(:, m_index)
    if C ≈ C_new then return C
```



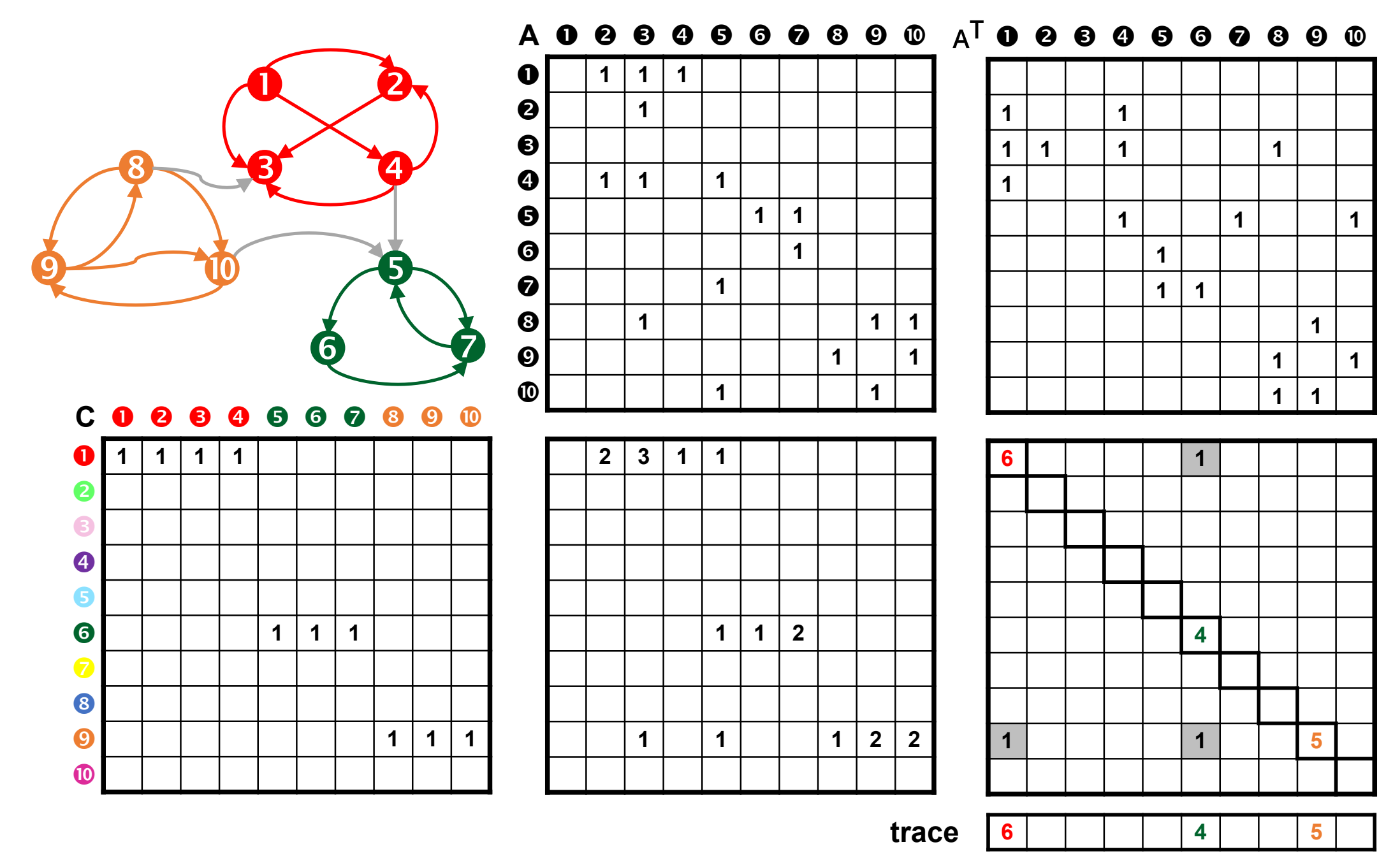
Markov Cluster Implementation

```
while (True)
    w = 1 ./ sum(A(:j)) // Normalize
      ↪  columns
    // Compute MSE of subsequent transfer
      ↪  matrices
    T = T^e              // Expansion step
    T = T .^ r           // Inflation step
    T = T(i, j) >= thr  // Pruning step
    // Terminate when MSE falls below some
      ↪  small value
```

- Based on the idea of random walks in a network structure [2].
- Native linear algebraic formulation, so transfers directly into GraphBLAS.
- Two phases: **expansion** (random walks) and **inflation** (heightens contrast between strong and weak connections)
- Prune small values to keep $\mathbf{T}$ sparse.
- Less interesting algorithm since $\mathbf{T}$ quickly becomes dense.

Cluster Quality Metrics

- In order to say what makes a particular clustering "good," **quality functions** are needed.
  - Mainly based on the idea that reasonable clusters will have more **intra-cluster** edges than **inter-cluster** edges.
- Including, but not limited to, Coverage (Cov), Performance (Perf) [3], and Modularity (Q) [4].
- $$\mathrm{Cov}(\mathcal{C}) = \frac{|E_{intra}|}{|E|} \qquad \mathrm{Perf}(\mathcal{C}) = \frac{|E_{intra}| + |N_{inter}|}{n(n-1)/2}$$



- $$Q = \sum_{c=1}^{n_c} \left[\frac{L_c}{|E|} - \left(\frac{d_c^+ \cdot d_c^-}{2 \cdot |E|} \right) \right]$$

Results

	com-Youtube				com-LiveJournal				com-DBLP			
n	1,134,890				3,997,962				317,080			
$nvals$	2,987,624				34,681,189				1,049,866			
	PPC1	PPC2	MCL	CDLP	PPC1	PPC2	MCL	CDLP	PPC1	PPC2	MCL	CDLP
Time (s)	6.084	2.324	18.16	22.47	39.48	50.15	54.28	79.04	2.653	0.7592	1.596	6.006
Cov	0.7838	0.1046	0.3241	0.6941	0.7844	0.1649	0.1761	0.9562	0.6251	0.3622	0.5952	0.6438
Perf	0.9134	0.9999	0.9997	0.8203	0.9084	0.9999	0.9999	0.4022	0.9996	0.9999	0.9999	0.9970
Mod	0.6294	0.1045	0.3238	0.4857	0.6688	0.1648	0.1761	0.4677	0.6240	0.3620	0.5951	0.6393
Avg. Size	26.74	1.355	4.893	19.69	34.87	2.119	3.922	111.4	8.963	2.151	8.328	14.02

Table 1. Benchmarking results for undirected graphs. PPC2 normalizes vertex weights via out-degree while PPC1 does not.

	wiki-Topcats						email-Eu-core					
n	1,791,489						1,005					
$nvals$	28,511,807						25,571					
	PPC1	PPC2	PPC3	PPC4	MCL	CDLP	PPC1	PPC2	PPC3	PPC4	MCL	CDLP
Time (s)	15.204	15.90	14.73	29.29	20.93	37.37	0.0102	0.0153	0.0118	0.0182	0.0185	0.0648
Cov	0.7908	0.0779	0.9378	0.2744	0.1639	0.9387	0.9971	0.2899	0.9609	0.3235	0.2545	1.000
Perf	0.6454	0.9999	0.3195	0.9934	0.9985	0.3008	0.1419	0.9722	0.2636	0.9666	0.9524	0.0621
Mod	0.2212	0.0775	0.1260	0.1652	0.1630	0.1357	0.0000	0.2422	0.0792	0.2698	0.2126	0.0000
Avg. Size	37.44	1.795	569.4	2.223	10.20	755.9	23.92	2.512	43.69	3.073	4.975	50.25

Table 2. Benchmarking results for directed graphs.

References

[1] B. Brock, A. Buluç, R. Kimmerer, J. Kitchen, M. Kumar, T. Mattson, S. McMillan, J. Moreira, M. Pelletier, and E. Welch, "The graphblas c api specification: Version 2.1.0," 2023.

[2] S. Dongen, "Graph clustering by flow simulation," *PhD thesis, Center for Math and Computer Science (CWI)*, 05 2000.

[3] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3, pp. 75–174, 2010.

[4] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Phys. Rev. E*, vol. 69, p. 026113, Feb 2004.

[5] E. Robinson, *6. Complex Graph Algorithms*, pp. 59–84.

quilicam@tamu.edu