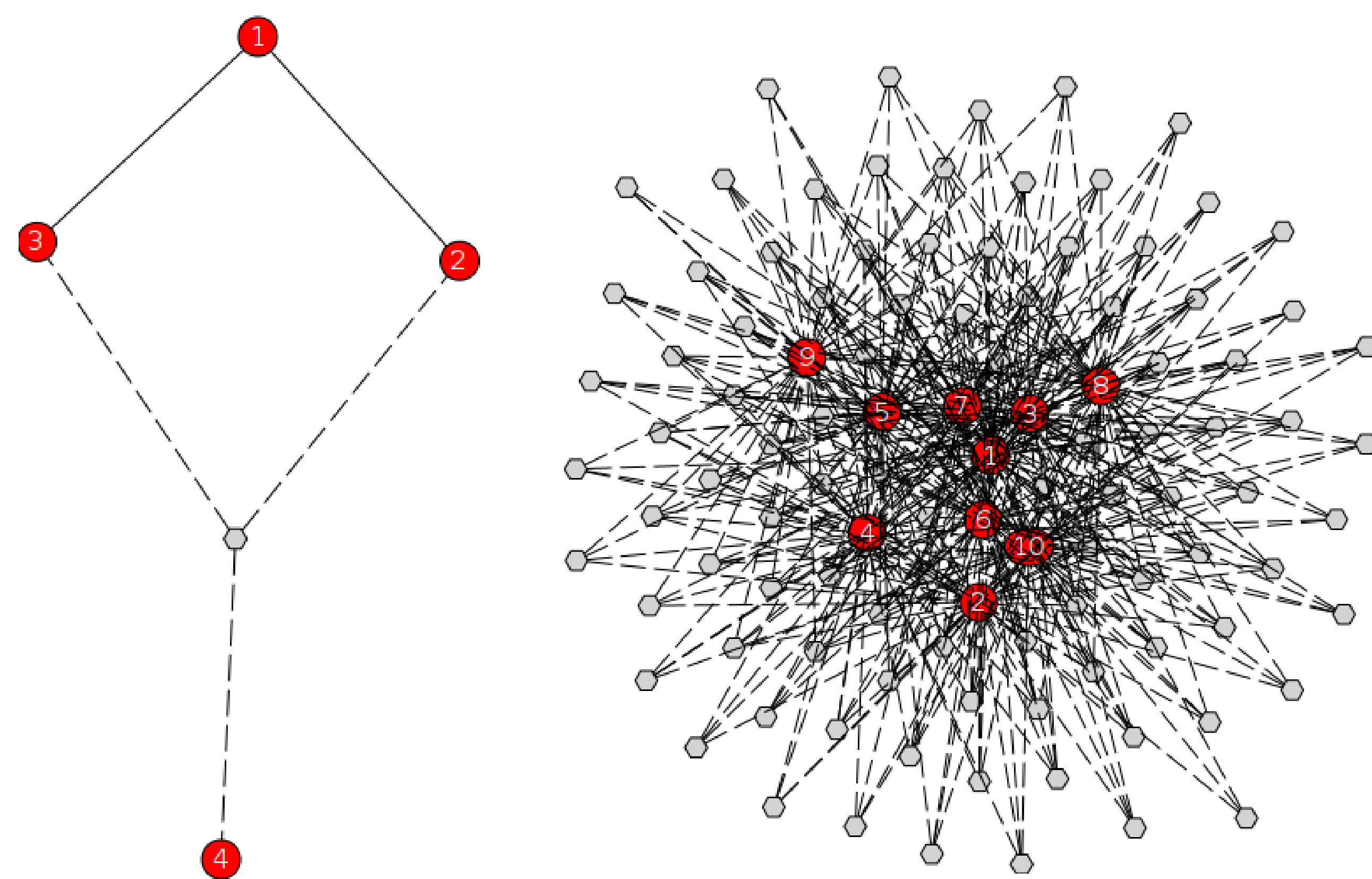


## What Is a Hypergraph?

A **hypergraph**  $H = (V, \mathcal{E})$  consists of

- a finite set  $V$  of *vertices*, and
- a collection  $\mathcal{E} \subseteq 2^V$  of nonempty subsets of  $V$  called **hyperedges**.

No further axioms are imposed: each hyperedge may join any number of vertices. Hypergraphs generalize ordinary graphs (where each edge has exactly two endpoints) by allowing edges of arbitrary size.



Examples of a simple hypergraph (left) and a complex hypergraph (right).

## Hypergraph Uses and Applications

Hypergraphs generalize the concept of a graph by allowing each hyperedge to connect any number of vertices, making them exceptionally versatile for modeling complex relationships. As a result, they appear in:

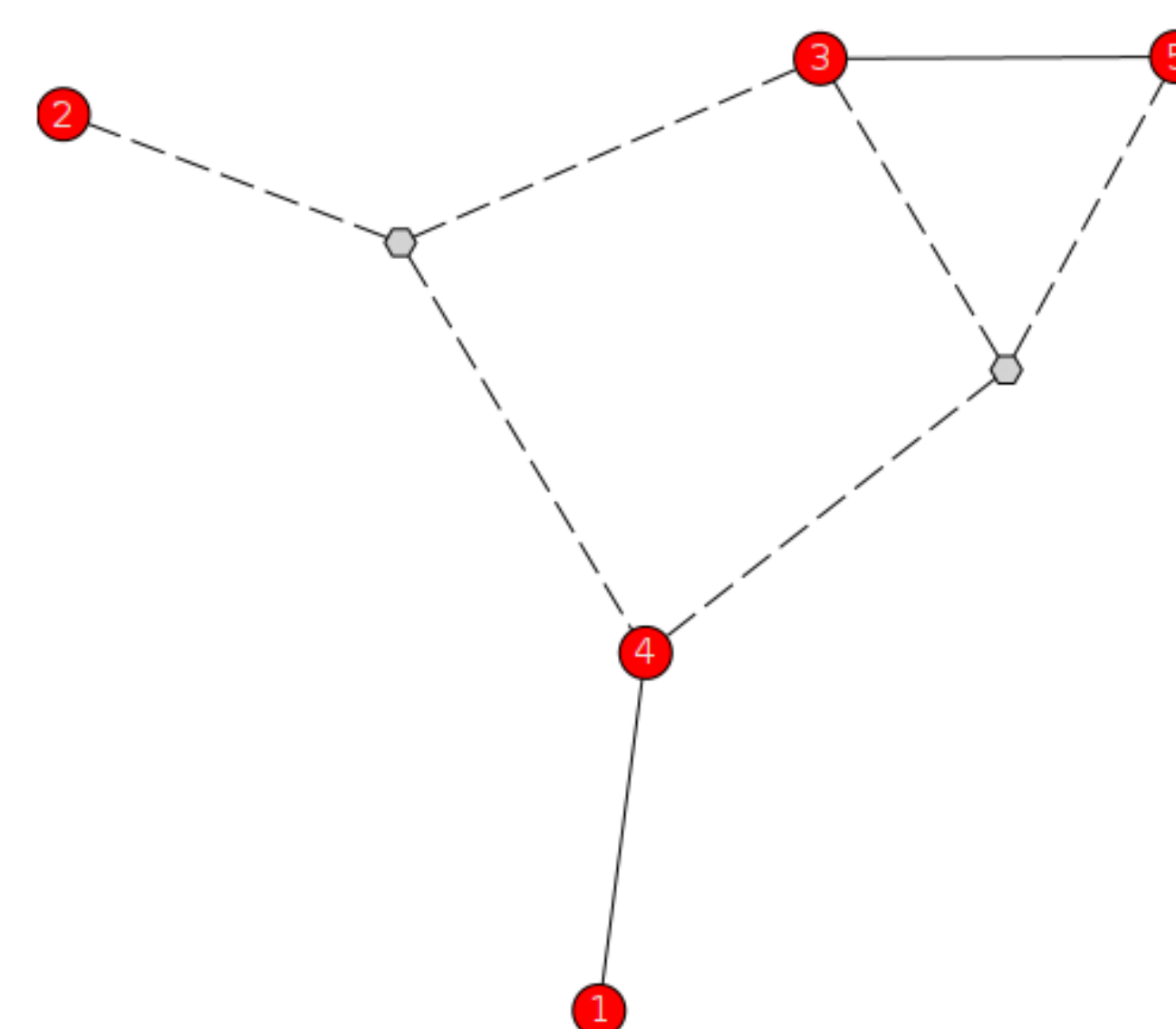
- **Combinatorial Optimization:** formulations of covering, hitting-set, and partition problems.
- **Database Theory:** representation of database schemas and query dependencies.
- **Machine Learning & Data Mining:** modeling higher-order feature interactions and clustering with set-based similarities.

## Transversals of a Hypergraph

A **transversal** (or *hitting set*) of a hypergraph is a collection of vertices that “hits” every hyperedge—meaning it contains at least one vertex from each edge.

In particular, our algorithm is focused on finding all of the **minimal transversals**, which are the smallest sets that still hit every hyperedge.

These minimal transversals are valuable as they represent the smallest way to cover all of the groups in your data or network. In practical terms, finding a minimal transversal helps you place the fewest sensors to monitor every channel, choose the smallest team to oversee all tasks, or allocate minimal resources to cover every requirement.



For example, in the hypergraph above, the minimal transversals are: (1,2,5), (1,3), (3,4), and (4,5).

## Generalized Nodes

Generalized nodes group together all vertices that belong to exactly the same set of hyperedges. By replacing each group  $X$  with a single auxiliary vertex  $v_X$ , one obtains a smaller “generalized hypergraph” whose minimal transversals can be computed far more efficiently. Once the transversals of the compressed hypergraph are found, each  $v_X$  is expanded back into the original vertices in  $X$ . This strategy:

- Collapses similar vertices to avoid redundant branching.
- Cuts down the number of intermediate partial transversals stored.

## Benchmarking and Results

By utilizing generalized nodes, we significantly reduce the number of intermediate transversals generated, reducing the number of operations required to generate all minimal transversals. The following table shows the results of our algorithm on select hypergraphs:

| Dataset      | Transversal (s) | GenNode Transversal (s) |
|--------------|-----------------|-------------------------|
| Fan10        | 0.007           | 0.006                   |
| L5           | 0.017           | 0.007                   |
| L6           | 1.884           | 0.153                   |
| L7           | 6.523           | 1.355                   |
| Rand_16_128  | 0.823           | 0.195                   |
| Random_32_16 | 109.068         | 19.943                  |
| dm30         | 1.217           | 0.243                   |
| K14_7        | 2.293           | 1.147                   |

While this method shows consistent speedups, during our experimentation we discovered a variety of other factors that can greatly impact the performance of our algorithm, such as the sorting of the hyperedges. Colex sorting, k-means clustering, and other methods can be used to group similar hyperedges together, which can lead to further speedups. We hope to continue to explore these methods in the future.

## Acknowledgements

We thank Dimitris J. Kavvadias and Elias C. Stavropoulos for their work on hypergraph transversal algorithms and the concept of utilizing generalied nodes. We are also grateful for the provisions of the ORCCA lab. This work was supported by funding from the USRI program.

## References

- [1] D. J. Kavvadias and E. C. Stavropoulos.  
An efficient algorithm for the transversal hypergraph generation.  
*Journal of Graph Algorithms and Applications*, 9(2):239–264, 2005.
- [2] Maplesoft.  
*Maple (2025)*.