UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA

CARLOS WESTERMANN

Hungarian Algorithm

Porto Alegre
May 2025

1 Introduction

This work presents an implementation of the Hungarian algorithm for solving the maximum weight perfect matching problem in weighted bipartite graphs, based on Schrijver's implementation, using Johnson's method and Dijsktra's algorithm that was implemented in previous work. The objective is to evaluate the practical performance of this implementation and validate the theoretical complexity bounds.

1.1 A note on the complexity analysis

The general complexity from Fact 1.1 in the notes is $O(n(m + n \log n))$, which applies to general bipartite graphs. However, for complete bipartite graphs (as used I have used in tesing), we have:

- **Complete Bipartite Graph**: $m = n^2$ (every vertex in S connects to every vertex in T)
- **Substitution**: $O(n(m + n \log n)) = O(n(n^2 + n \log n)) = O(n^3 + n^2 \log n)$
- **Asymptotic Dominance**: Since n^3 grows faster than $n^2 \log n$, we have $O(n^3 + n^2 \log n) = O(n^3)$, however, as our graphs are of moderate size, I decided to maintain the term in analysis as it may have a meaningful impact in practice

2 Implementation Details

- **Weight Matrix**: Stores the original bipartite graph weights
- **Matching Vector**: Maintains current matching M
- **Residual Graph**: Represents H_M with transformed edge weights

3 Testing Environment

All test cases were executed on:

- Chip: Apple M3 Pro
- Total Number of Cores: 12 (6 performance and 6 efficiency)

- Memory: 18 GB

4 Test Data Generation

For each graph size, 15 graphs in the following structure were generated:

1. **Graph Structure**: Complete bipartite graphs $G = (S \cup T, A)$ with $|S| = |T| = n$
2. **Graph Sizes**: $n \in \{100, 200, 300, 400, 500, 1000, 1500, 3000\}$
3. **Weight Distribution**: Edge weights uniformly distributed in $[0, n^2]$.

Each generated instance includes the optimal solution computed using scipy linear assignment solver for verification, to ensure my implementation produces the correct result.

5 Experimental Results and Analysis

5.1 Iteration Count

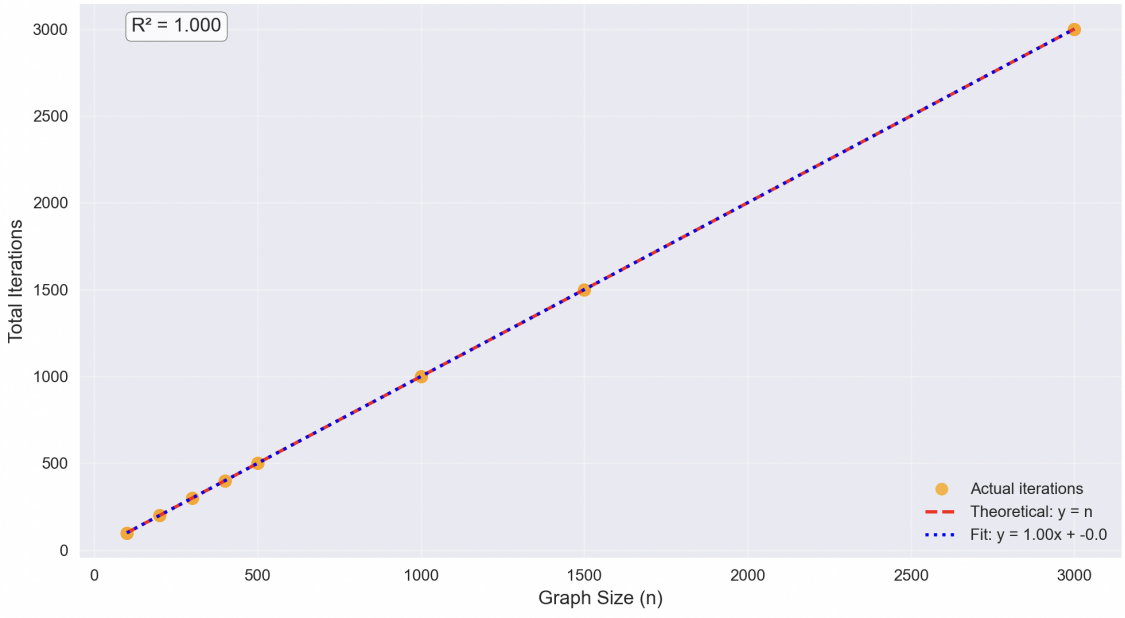

Figure 1 – Iterations vs Number of Vertices

Figure 1 shows that we can directly confirm the theoretical expectation that the algorithm should iterate n times n a correct implementation. The perfect linear relationship (R^2 = 1.000) demonstrates that the algorithm requires precisely n iterations for a com-

plete bipartite graph of size n, meaning each iteration increases matching size by one, guaranteeing each matching found is extreme.

5.2 Dijkstra's algorithm

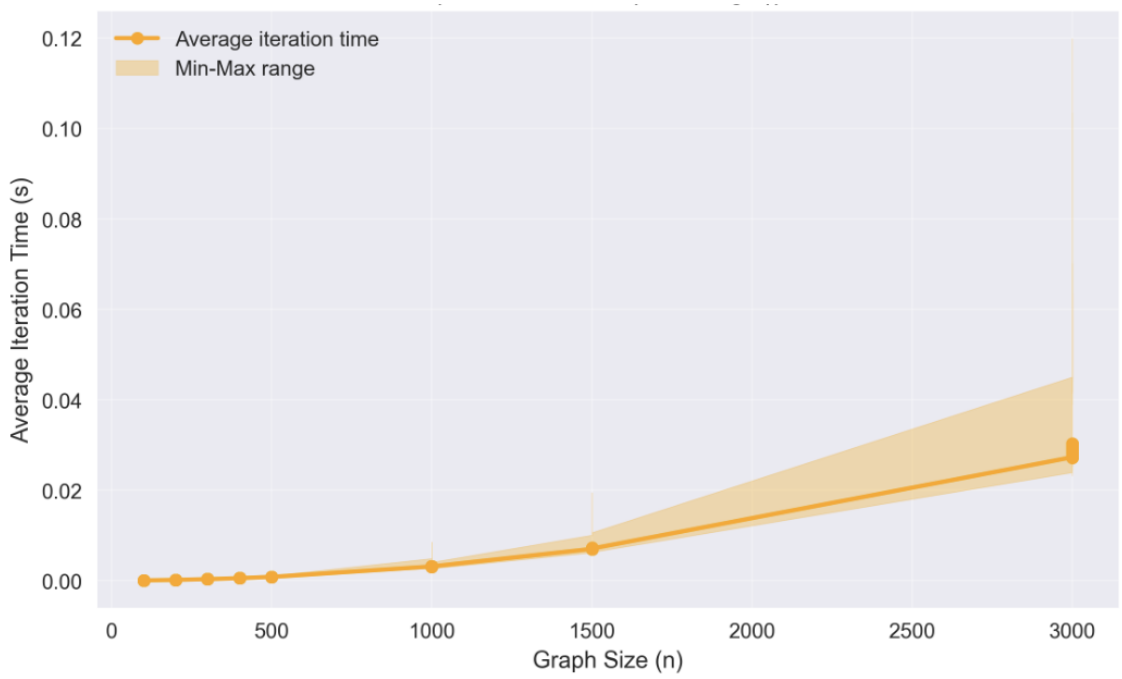

Figure 2 – Average time inside each iteration vs Number of vertices

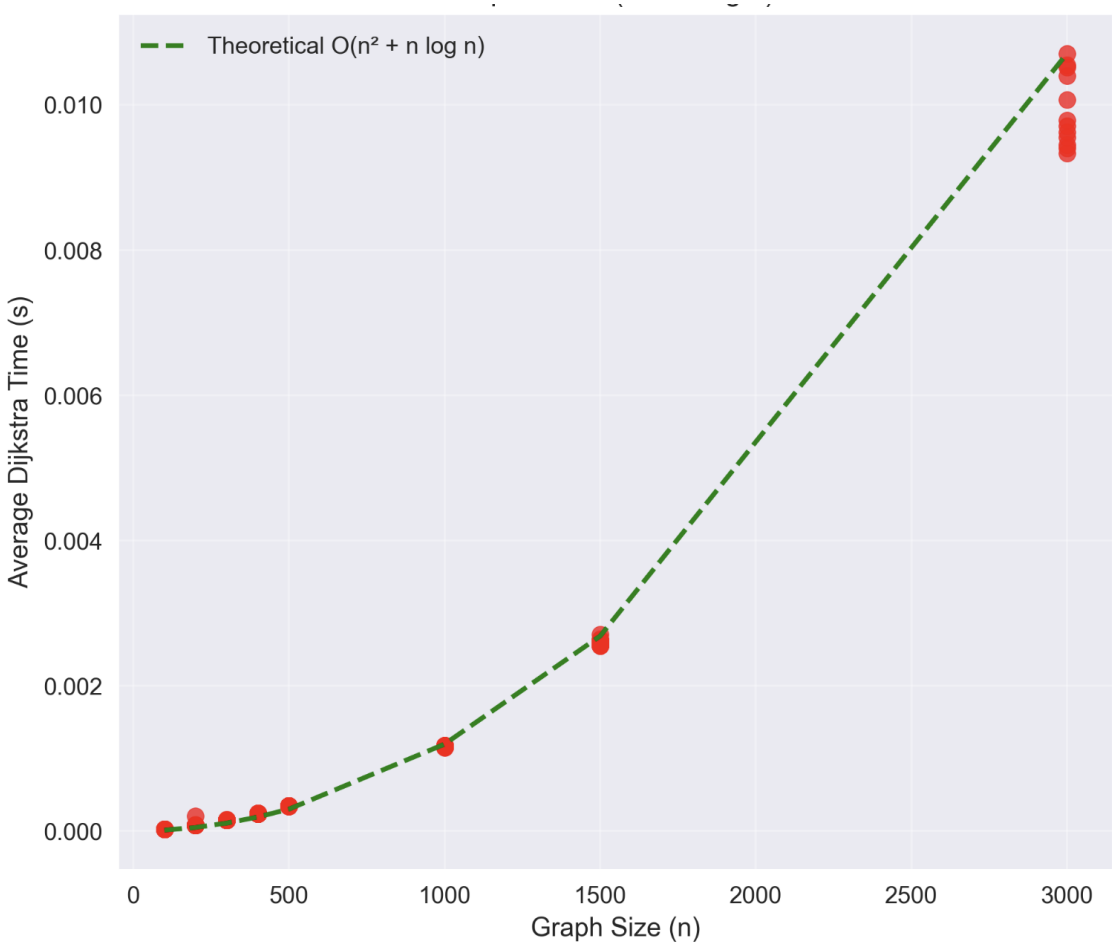

Figure 3 – Average time for Dijkstra's algorithm vs Number of Vertices

From Figures 2 and 3, we can observe that although Dijkstra's method respects the theoretical upper bound, it is not processing the graph in a very efficient form, which should not be the average case as seen previously in Lab 1. This may be happening because the implementation in this case is not stopping early when a target is found, finding all possible reachable vertices. This does not cause the algorithm to be wrong, but it is not fully optimized. Because this is the core contributor to the complexity of the algorithm, we will see that we reach a runtime very close to the upper bound in practice.

5.3 Scalability

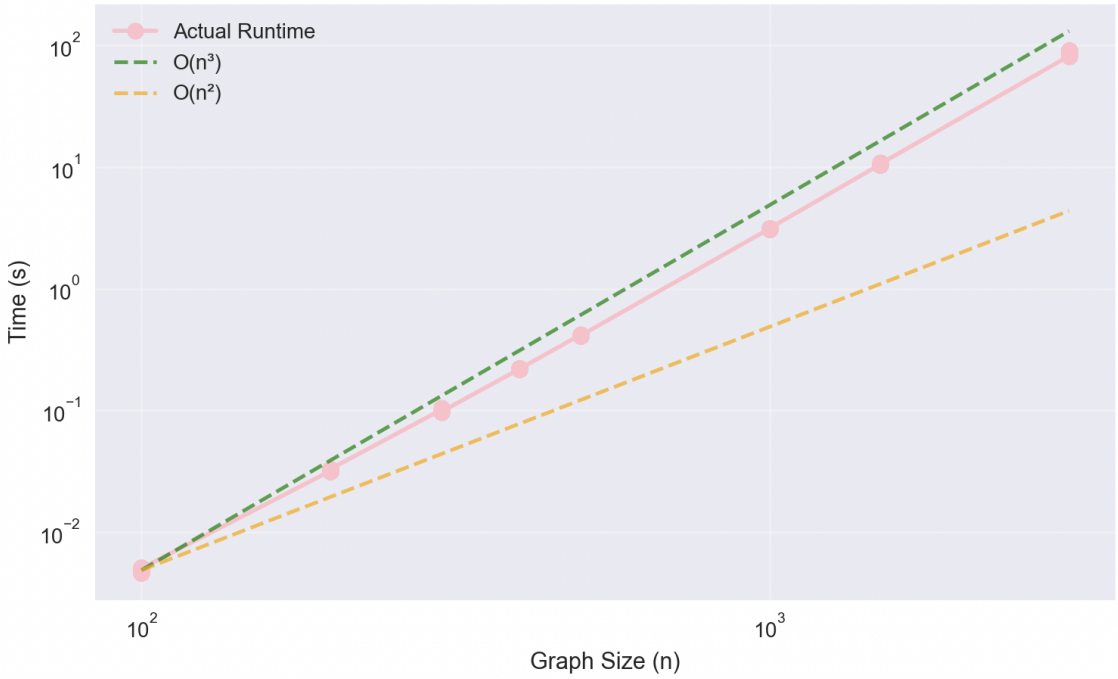

Figure 4 – Runtime vs Number of Vertices (logarithmic scale)

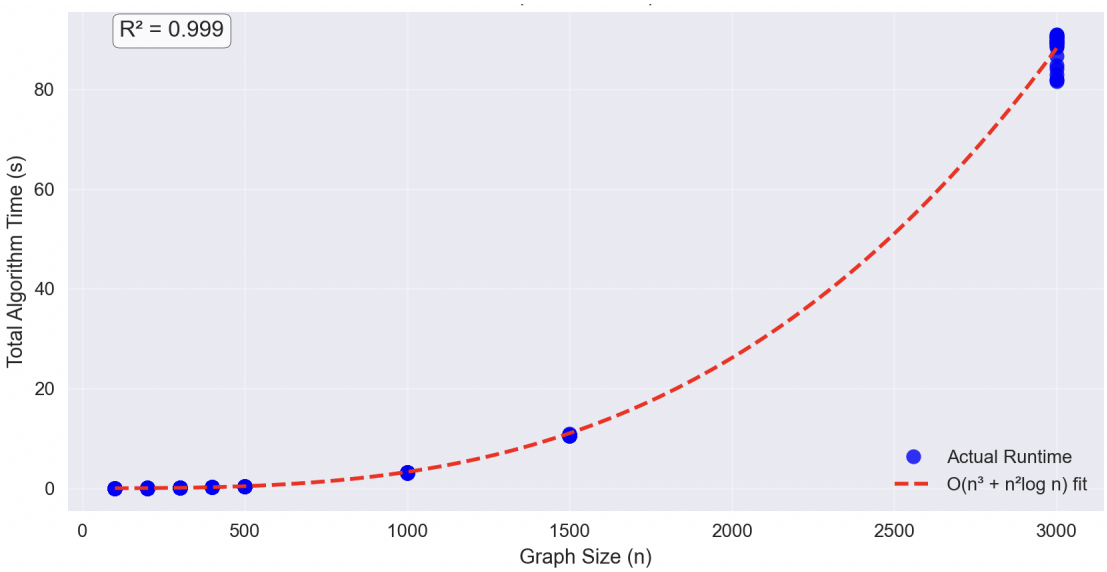

Figure 5 – Runtime vs Number of Vertices (linear scale) with a comparison to the curve $O(n^3 + n^2 \log n)$

The runtime analysis provides validation of the expected complexity of the algorithm. The fit ($R^2 = 0.997$) to the curve $O(n^3 + n^2 \log n)$ confirms that our implementation achieves the theoretical limits.

6 Conclusion

The implementation successfully validates the Hungarian algorithm theory when implemented using Johnson's method. We can see that the implementation follows the theoretical limits and respects the proposed execution of the algorithm.