CSE331 - Assignment #2

Doyeol Oh (20211187)
UNIST
South Korea
ohdoyoel@unist.ac.kr

1 INTRODUCTION

The Traveling Salesman Problem (TSP) is a fundamental NP-hard problem in combinatorial optimization. This assignment investigates both exact and approximate algorithms for solving the TSP, focusing on their design, scalability, and performance.

The following established methods were implemented:

- MST-based 2-approximation
- Christofides algorithm with Edmonds' Blossom algorithm
- Christofides algorithm with Gabow's Blossom algorithm
- Held-Karp dynamic programming

We also propose **Hylos**, a scalable hierarchical algorithm that partitions large instances using dynamic k-means clustering and solves subproblems with a hybrid of Held-Karp and Christofides approaches, integrating local and global optimization.

All algorithms are evaluated on standard benchmark datasets, comparing runtime, theoretical complexity, and approximation performance.

Full code is available at:

https://github.com/ohdoyoel/unist_cse331_assignment_2

2 PROBLEM STATEMENT

The *Traveling Salesman Problem (TSP)* is a classical NP-hard problem in combinatorial optimization. Given a set of n cities $V = \{v_1, v_2, \ldots, v_n\}$ and a symmetric, non-negative distance function $d : V \times V \to \mathbb{R}_{\geq 0}$ satisfying the triangle inequality, the goal is to find the shortest tour that:

- visits each city exactly once, and
- returns to the starting city.

Formally, the TSP seeks a minimum-cost Hamiltonian cycle in a complete undirected graph $G = (V, E)$ with edge weights $d(v_i, v_j)$. A tour is represented as an ordered sequence:

$$\pi = (v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_n}),$$

with cost:

$$\min_{\pi \in \mathcal{P}(V)} \sum_{i=1}^{n} d(v_{\pi_i}, v_{\pi_{i+1}}), \quad \text{where } v_{\pi_{n+1}} = v_{\pi_1}.$$

Despite its computational difficulty, the TSP has broad practical relevance in logistics, robotics, circuit design, and bioinformatics, making it a key benchmark in optimization research.

3 EXISTING ALGORITHMS

3.1 MST-based 2-Approximation Algorithm

This classical heuristic solves the metric TSP by leveraging the structure of a Minimum Spanning Tree (MST). [**?**] It guarantees a tour cost no more than twice the optimal by exploiting the triangle inequality.

3.1.1 ***Procedure.***

(1) Construct an MST T of the input graph $G = (V, E)$ using Prim's or Kruskal's algorithm.
(2) Perform a depth-first preorder traversal of T to obtain a city sequence $\pi = (v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_n})$.
(3) Build a tour by visiting cities in π, skipping duplicates, and returning to the starting city.

The MST provides a lower bound on the optimal tour cost. The preorder traversal visits each edge at most twice, and shortcutting repeated nodes using the triangle inequality ensures the total cost remains within $2 \cdot OPT$.

3.1.2 ***Approximation Ratio.***

$$\text{TourCost} \leq 2 \cdot \text{MSTCost} \leq 2 \cdot OPT \quad \Rightarrow \quad \frac{\text{TourCost}}{OPT} \leq 2$$

3.1.3 ***Time Complexity.***

- MST construction: $O(E \log V)) = O(n^2 \log n)$
- Preorder traversal: $O(V) = O(n)$
- Tour construction with shortcutting: $O(V) = O(n)$

Overall time complexity:

$$O(n^2 \log n)$$

3.2 Christofides with 2 Blossom Algorithms

Christofides' algorithm [**?**] is a classical 1.5-approximation for the metric TSP, improving upon the MST-based method by adding a minimum-weight perfect matching on the odd-degree vertices of the MST.

3.2.1 ***Procedure.***

Given a complete graph $G = (V, E)$ with symmetric, non-negative weights satisfying the triangle inequality:

(1) Build an MST T of G.
(2) Identify the set $O \subset V$ of odd-degree vertices in T.
(3) Compute a minimum-weight perfect matching M on O using either Edmonds' or Gabow's Blossom algorithm.
(4) Form a multigraph $H = T \cup M$, which is Eulerian.
(5) Find an Eulerian tour on H and shortcut repeated vertices to obtain a Hamiltonian cycle.

3.2.2 ***Blossom Algorithms.***

Edmonds' Algorithm. Handles general graphs by contracting odd-length cycles ("blossoms") during the search for augmenting paths. Dual variables guide the matching toward minimal total weight.

Gabow's Improvement. Optimizes Edmonds' method using efficient data structures, priority queues, and forest-based blossom management. [1] It avoids redundant operations and systematically updates dual variables.

3.2.3 *Approximation Ratio*.

$$\text{TourCost} \leq \text{MSTCost} + \text{MatchingCost} \leq OPT + \frac{1}{2}OPT = \frac{3}{2}OPT$$

$$\Rightarrow \quad \frac{\text{TourCost}}{OPT} \leq 1.5$$

3.2.4 *Time Complexity*.

- MST construction: $O(E \log V) = O(n^2 \log n)$
- Finding odd-degree vertices: $O(V) = O(n)$
- Minimum-weight perfect matching:
  - Edmonds' algorithm: $O(V^4) = O(n^4)$
  - Gabow's implementation: $O(V^3) = O(n^3)$
- Euler tour and shortcutting: $O(E) = O(n^2)$

Overall time complexity: Dominated by the perfect matching step,

$O(n^3)$ when using Gabow's algorithm,
or $O(n^4)$ with Edmonds' original version.

3.3 Held-Karp Algorithm

The Held-Karp algorithm [2] provides an exact solution to the TSP using dynamic programming. It computes the minimum cost of visiting all cities by solving overlapping subproblems over subsets of cities, offering exponential but significantly better performance than brute-force enumeration.

3.3.1 *Procedure*.

Let $V = \{v_1, v_2, \ldots, v_n\}$ be the set of cities, with v_1 as the starting point. Define $C[S][j]$ as the minimum cost to reach city v_j from v_1 while visiting every city in subset $S \subseteq V$ exactly once and ending at v_j.

(1) **Initialization:** $C[\{v_1, v_j\}][j] = d(v_1, v_j)$ for all $j \neq 1$.
(2) **Recurrence:** For all subsets $S \subseteq V$ with $v_1 \in S$ and $|S| > 2$, and all $j \in S$, $j \neq 1$:

$$C[S][j] = \min_{k \in S \setminus \{j\}} \left(C[S \setminus \{j\}][k] + d(v_k, v_j) \right)$$

(3) **Final step:**

$$\min_{j \neq 1} \left(C[V][j] + d(v_j, v_1) \right)$$

3.3.2 *Approximation Ratio*.

As an exact algorithm, Held-Karp always returns the optimal solution:

$$\frac{\text{TourCost}}{OPT} = 1$$

3.3.3 *Time Complexity*.

- Time: $O(n \cdot 2^n)$ entries in the DP table $\times$ $O(n)$ time to compute = $O(n^2 \cdot 2^n)$

Overall time complexity :

$$O(n^2 \cdot 2^n).$$

Although exponential, the algorithm is significantly more efficient than brute-force enumeration $O(n!)$, and remains practical for instances with $n \leq 22$.

4 PROPOSED ALGORITHM: Hylos

4.1 Motivation

Hylos stands for *Hierarchically Localized Optimization Strategy*—a scalable heuristic designed to solve large-scale instances of the Traveling Salesman Problem (TSP). The core idea is simple: apply the exact Held-Karp algorithm to small subproblems and use Christofides' approximation for larger ones. Since Held-Karp becomes infeasible beyond $n \leq 22$, we adopt a divide-and-conquer approach to scale effectively.

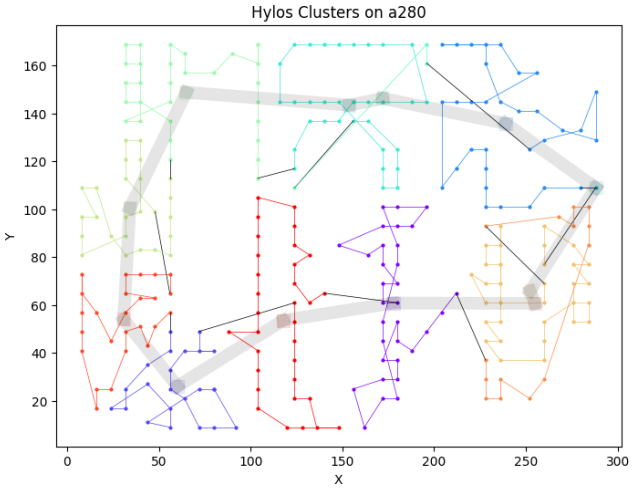
First, dynamic k-means is used to group cities into spatially compact clusters, ensuring each remains small enough for efficient TSP solving.

Next, a TSP is solved over the cluster centroids to determine the inter-cluster visiting order. If the number of clusters is small, Held-Karp is used; otherwise, Christofides is applied.

Then, each cluster is independently solved using Held-Karp for small clusters and Christofides for larger ones.

Finally, each cluster selects entry and exit points from its local tour to minimize the inter-cluster transition cost. The local tours are reordered accordingly.

This hierarchical structure allows Hylos to balance optimality and scalability by tailoring solvers to the scale of each subproblem.



4.2 Procedure

Hylos operates in four main stages:

(1) **Clustering:** Input cities are grouped using k-means. We define:

$$\text{MAX\_SIZE} = \max(22,\ 2\sqrt{n}), \quad k = \left\lceil \frac{n}{\text{MAX\_SIZE}} \right\rceil$$

(2) **Inter-cluster TSP:** A TSP is solved on the centroids:
  - Held-Karp if $k \leq 22$
  - Christofides otherwise
(3) **Intra-cluster TSP:** Each cluster is solved individually:
  - Held-Karp if size ≤ 22
  - Christofides otherwise
(4) **Entry/Exit Alignment:** For each pair of adjacent clusters, entry and exit points are selected to minimize the transition cost. Tours are adjusted accordingly.

Dataset	Metrics	MST 2-Approximation	Christofides (Edmonds)	Christofides (Gabow)	Held-Karp	Hylos
ulysses16	Time	1.0782e-05	4.3229e-05	3.0349e-05	0.0412702	0.043268 ± 0.002866
	Cost	84.1331	79.4897	75.003	73.9876	73.9876
	Approximation Ratio	1.13527	1.07261	1.01207	1	1
ulysses22	Time	1.2908e-05	5.106e-05	5.3138e-05	5.05119	4.348854 ± 0.311893
	Cost	86.7668	85.0368	78.4333	75.3097	75.3097
	Approximation Ratio	1.14672	1.12386	1.03658	1	1
a280	Time	0.000443099	0.000927949	0.000989282	-	1.398360 ± 1.836770
	Cost	3575.45	3109.33	3022.64	-	3751.25 ± 110.63
	Approximation Ratio	1.38221	1.20201	1.1685	-	1.450170 ± 0.042768
xql662	Time	0.00158809	0.00437905	0.00593478	-	1.830262 ± 2.292609
	Cost	3648.01	3001.22	2929.48	-	3647.75 ± 126.82
	Approximation Ratio	1.42748	1.17439	1.14632	-	1.427377 ± 0.049626
kz9976	Time	0.462964	1.24306	0.629829	-	2.897765 ± 2.361282
	Cost	1.45792e+06	1.26889e+06	1.52529e+06	-	1.417675e+06 ± 11124.89
	Approximation Ratio	1.37296	1.19495	1.43641	-	1.335067 ± 0.010476
mona-lisa100k	Time	71.6649	69.6809	66.7726	-	8.0578 ± 2.4575
	Cost	8.3282e+06	7.20499e+06	7.20499e+06	-	7.08212e+06 ± 16478.30

Table 1: Comparison of TSP Algorithms on Various Datasets

4.3 Approximation Ratio

Although Hylos lacks a fixed approximation bound, each component has a known guarantee:

- **Intra-cluster:** Held-Karp (1.0), Christofides (1.5)
- **Inter-cluster:** Christofides (1.5), with distortion from centroid-based connection

Let α_i be the distortion between centroid pair (c_i, c_{i+1}) and their actual entry/exit points (e_i, e_{i+1}):

$$\alpha_i = \frac{d(e_i, e_{i+1})}{d(c_i, c_{i+1})}, \quad \alpha = \max_i \alpha_i$$

Then the total cost satisfies:

$$\text{ApproxRatio}_{\text{Hylos}} \leq r + \alpha \cdot \rho$$

where $r \in [1.0, 1.5]$ is the intra-cluster bound, and $\rho \in [1.0, 1.5]$ is the inter-cluster bound. Since α is not empirically bounded, we conservatively express:

$$\text{ApproxRatio}_{\text{Hylos}} \leq 1.5 + 1.5\alpha$$

This highlights that total tour quality depends on how closely centroid connections reflect true entry/exit transitions. While this appears weak theoretically, empirical performance is strong.

4.4 Time Complexity

Let n be the total number of cities, $k = O(\sqrt{n})$ the number of clusters, and $m = O(\sqrt{n})$ the average cluster size. The clustering requires $O(kni)$, where k is the number of clusters, n is the number of cities, and i is the number of iterations until convergence.

Best Case (Held-Karp used throughout):

$$O(kni + k^2 2^k + km^2 2^m + n) = O(n^{1.5} i + n2^{\sqrt{n}})$$

Average Case (Mixed usage):

$$O(kni + k^3 + rm^2 2^m + (k - r)m^3 + n)$$

Worst Case (All Christofides):

$$O(kni + k^3 + km^3 + n) = O(n^{1.5} i + n^{1.5})$$

In all cases, Hylos avoids the exponential blow-up of $O(2^n n^2)$ from solving the full problem with Held-Karp.

5 EXPERIMENTS

5.1 Datasets

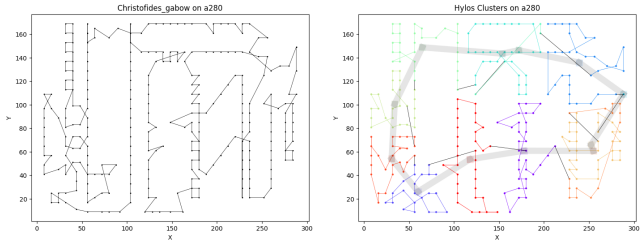
We evaluate all algorithms on a range of benchmark TSP instances of varying size and complexity: `ulysses16`, `ulysses22`, `a280`, `xql662`, `kz9976`, and `mona-lisa100k`. These datasets represent small to large-scale problems and are well-suited for assessing both the quality and scalability of each algorithm.

5.2 Runtime Analysis

Table 1 presents empirical runtimes across all algorithms and datasets. These results are consistent with the theoretical time complexities of each method:

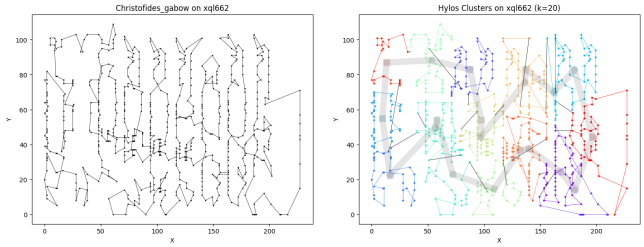
- **MST 2-Approximation** ($O(n^2)$) is the fastest in all cases, requiring under 2 ms even for the largest dataset (`mona-lisa100k`).
- **Christofides (Edmonds/Gabow)** scale polynomially ($O(n^3)$), showing higher overhead on large instances: 0.6–1.2s on `kz9976` and over a minute on `mona-lisa100k`.
- **Held-Karp** ($O(n^2 \cdot 2^n)$) is tractable only for small datasets such as `ulysses16` and `ulysses22`, and fails to terminate on larger inputs.
- **Hylos** adapts dynamically to input scale. Although slightly slower on small datasets due to clustering overhead, it clearly outperforms other methods on large-scale inputs. On `mona-lisa100k`, it completes in just 8.06 seconds, over 8× faster than the fastest Christofides variant.

For example, on `kz9976`, Hylos completes in 2.89s on average, compared to 1.24s (Edmonds) and 0.63s (Gabow). However, the



(a) Christofides (b) Hylos

Figure 1: Comparison on `a280`



(a) Christofides (b) Hylos

Figure 2: Comparison on `xql662`

advantage is most evident on `mona-lisa100k`, where Christofides variants exceed 66 seconds, while Hylos remains under 10.

5.3 Approximation Quality

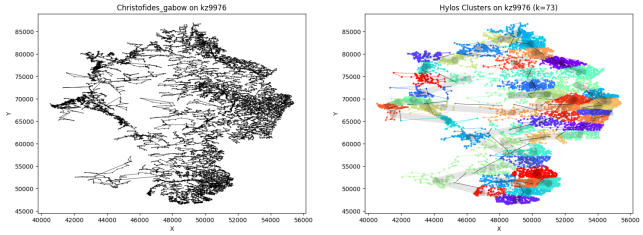
All algorithms are compared against ground-truth costs provided by TSPLIB and TTD. The results confirm a classic trade-off between runtime and solution quality:

- **MST 2-Approximation** yields the worst ratios—up to 1.45 on `a280` and 1.43 on `xql662`—matching its known theoretical bounds.
- **Christofides (Gabow)** offers the best heuristic performance, achieving approximation ratios of 1.17 on `xql662` and 1.14 on `kz9976`.
- **Held-Karp** serves as the benchmark, producing optimal solutions with approximation ratios 1.0 on small datasets.
- **Hylos** strikes a practical balance. It achieves competitive approximation ratios such as 1.45 (`a280`), 1.43 (`xql662`), and 1.33 (`kz9976`), while remaining efficient. On `mona-lisa100k`, Hylos produces the lowest cost among heuristics, outperforming Christofides by approximately 2%.

Overall, Hylos demonstrates strong scalability and quality, validating its use for real-world TSP scenarios where exact solutions are impractical.

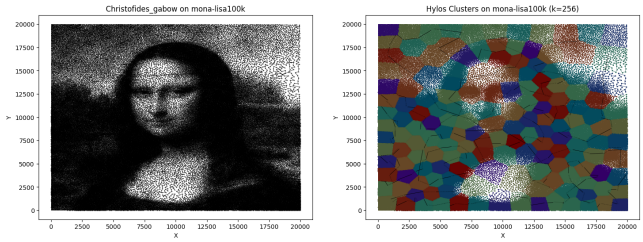
5.4 Visualization

Visualizations reveal that Hylos maintains well-structured tours through spatial clustering, whereas Christofides often yields entangled paths. This locality enforcement provides both computational gains and clearer route organization.



(a) Christofides (b) Hylos

Figure 3: Comparison on `kz9976`



(a) Christofides (b) Hylos

Figure 4: Comparison on `mona-lisa100k`

6 CONCLUSION

This study explored multiple approaches to the Traveling Salesman Problem (TSP), including classical approximation methods, an exact dynamic programming solution, and Hylos—a novel hierarchical algorithm designed for large-scale instances. Each algorithm was evaluated across benchmark datasets in terms of runtime and approximation quality.

The MST 2-Approximation and Christofides (Edmonds/Gabow) algorithms demonstrated the expected trade-offs between speed and accuracy. While Held-Karp offered optimal solutions for small instances, it was computationally infeasible for larger ones. Hylos, by contrast, achieved a strong balance between scalability and solution quality by dynamically combining Held-Karp and Christofides based on subproblem size.

Across various problem scales, Hylos consistently delivered near-optimal results for small inputs and maintained high efficiency for large ones. Its hybrid, modular design proved robust and adaptable.

In summary, Hylos offers a practical and scalable framework for solving the TSP. Its hierarchical structure lays the groundwork for future improvements such as adaptive clustering, parallel processing, and machine learning–based instance analysis to further enhance performance in real-world applications.

References

[1] Harold N Gabow. 1976. An efficient implementation of Edmonds' algorithm for maximum matching on graphs. *Journal of the ACM (JACM)* 23, 2 (1976), 221–234. doi:10.1145/321941.321942

[2] Michael Held and Richard M Karp. 1962. A dynamic programming approach to sequencing problems. *J. Soc. Indust. Appl. Math.* 10, 1 (1962), 196–210. doi:10.1137/0110015