TRAVELLING SALESMAN PROBLEM

Course Details: **(CSL2020)-DATA STRUCTURE AND ALGORITHMS**

Suchetana Chakraborty

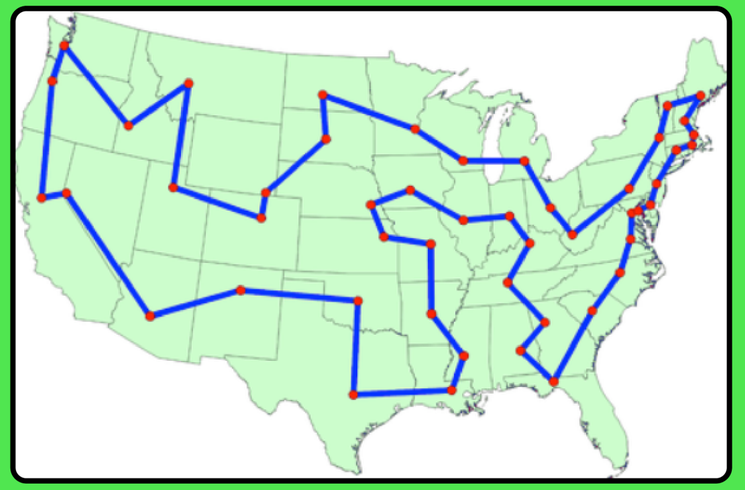
Mentor TA: **Dixit Dutt Bohra & Shubham Kumar**

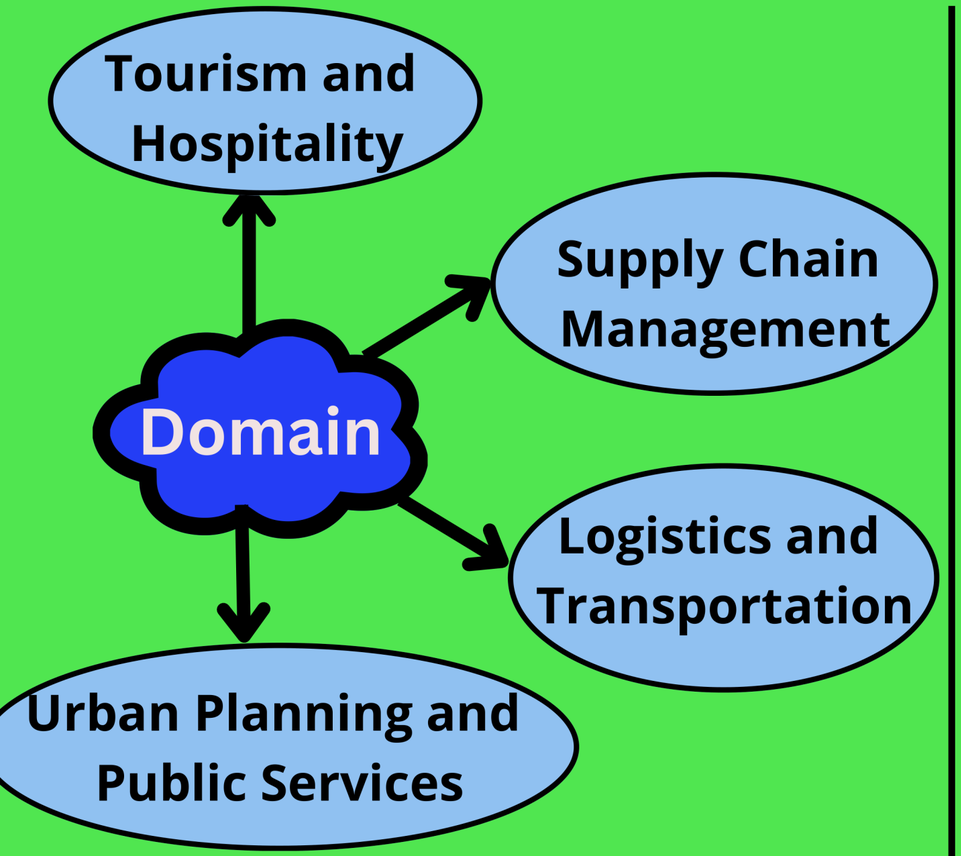
Team Members:

1. Pushkin Dugam (B22ME052)
2. Gopala Ram Jyani (B22ME075)
3. Deepu (B22CY001)
4. Shivangini Rathore (B22PH012)

Problem Statement

Given a set of cities and distances between every pair of cities, the problem is to find the shortest possible route that visits every city exactly once and returns to the starting point.





Why its an important/relevant problem?

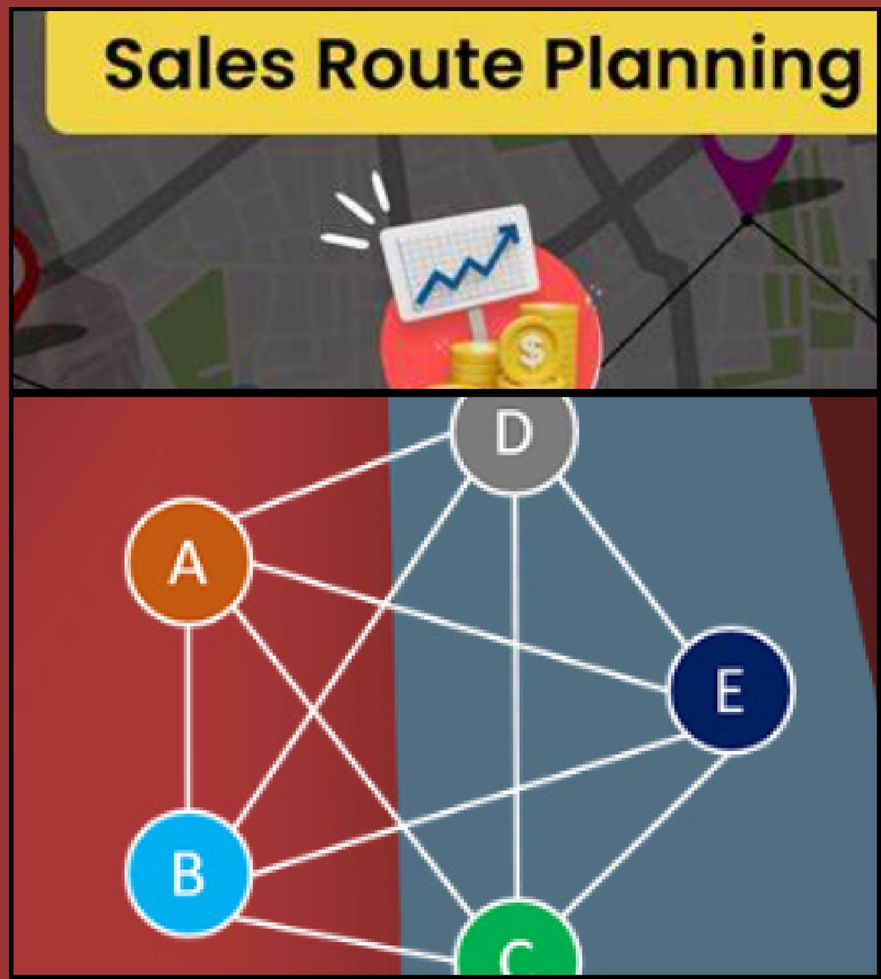
- By efficiently solving TSP, businesses can **optimize resource usage, minimize costs, and enhance operational efficiency** in areas such as logistics, transportation, and supply chain management.
- Improved route planning leads to **faster delivery times, reduced waiting periods, and heightened customer satisfaction**, vital for maintaining loyalty and competitive advantage.
- Moreover, TSP solutions indirectly contribute to **environmental sustainability** by minimizing travel distances and fuel consumption.
- Additionally, TSP serves as a benchmark problem in optimization, driving advancements in algorithm design and methodologies.

Why it’s a challenging problem?

As the number of cities increases, the possible **number of routes grows exponentially**, making it impractical to evaluate every potential solution. Additionally, finding the optimal solution requires examining all possible permutations, which becomes **computationally prohibitive for large instances**. Real-world constraints such as time windows, vehicle capacities, and varying travel costs further complicate the problem.

Why a data-driven solution looks promising?

Firstly, by leveraging historical data on routes, traffic patterns, and delivery schedules, a data-driven approach can identify patterns and trends to **inform more efficient route planning**. Additionally, real-time data on factors like traffic congestion, weather conditions, and road closures enables dynamic adjustments to routes, optimizing them in response to changing circumstances. Overall, a data-driven solution offers the potential to improve the accuracy, efficiency, and adaptability of TSP solutions, making it a promising approach for addressing this challenging problem.



Current Status

Exact Algorithms

1] Naive Brute Force Approach:

- Enumerates all possible permutations of cities .
- Calculates the total cost for each permutation.
- Keeps track of the minimum cost permutation.

Time Complexity: O(n!)
Space Complexity: O(n)

2] Dynamic Programming:

solves complex problems by breaking them down into simpler subproblems, solving each subproblem just once, and storing their solutions
Time Complexity: O(2^n · n^2)
Space Complexity: O(2^n · n)

3] Branch and Bound:

partitions the search space into smaller regions (branches) and uses bounds to intelligently prune regions that do not contain an optimal solution.
Time Complexity: Exponential
(depends on the branching factor)
Space Complexity: O(n)

Heuristic Methods

1] Nearest Neighbor:

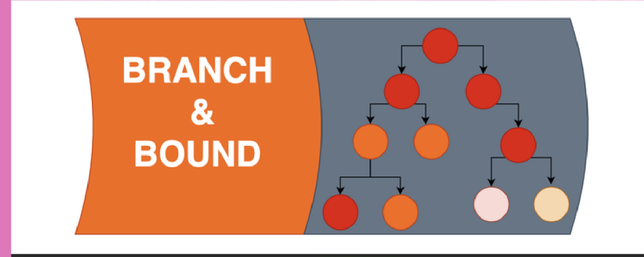
Greedy approach that selects the nearest unvisited city at each step.
Time Complexity: O(n^2)
Space Complexity: O(n)

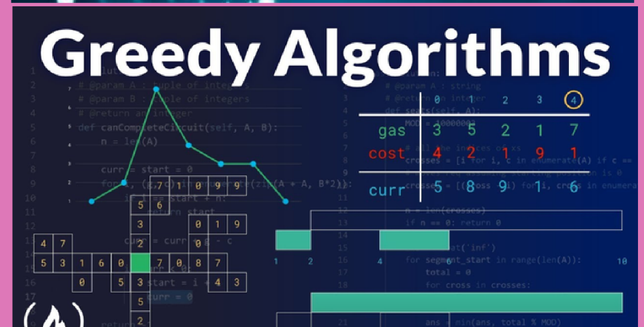
2] Christofides Algorithm:

Guarantees a solution within 3/2 times the optimal solution for symmetric TSP by combining minimum spanning tree and minimum-weight perfect matching.
Time Complexity: O(n3)
Space Complexity: O(n2)

3] Genetic Algorithms:

Evolves solutions using genetic operators (selection, crossover, mutation).





Limitations

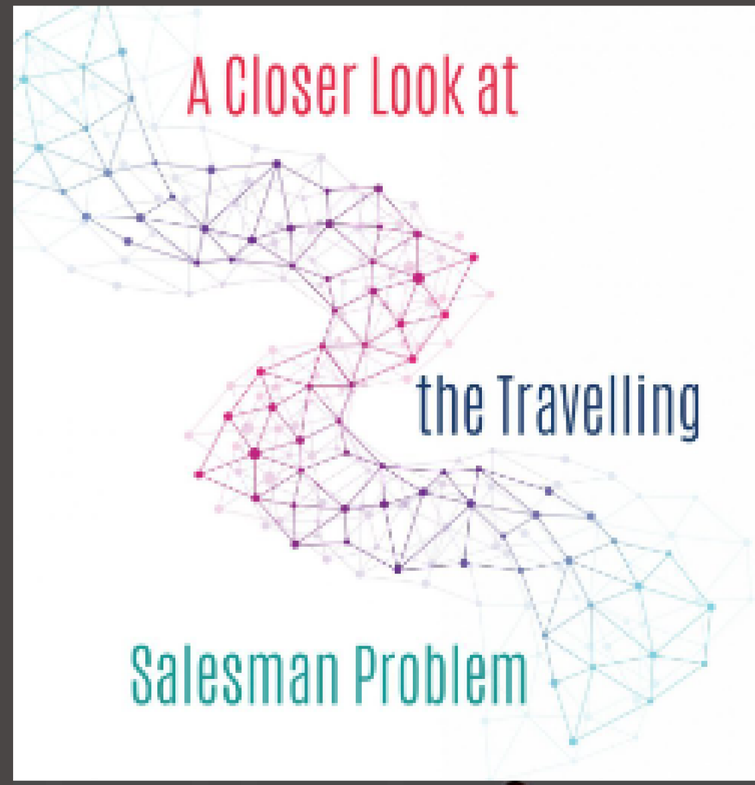
Despite decades of research, no algorithm has been discovered that can solve TSP optimally in polynomial time for all instances.

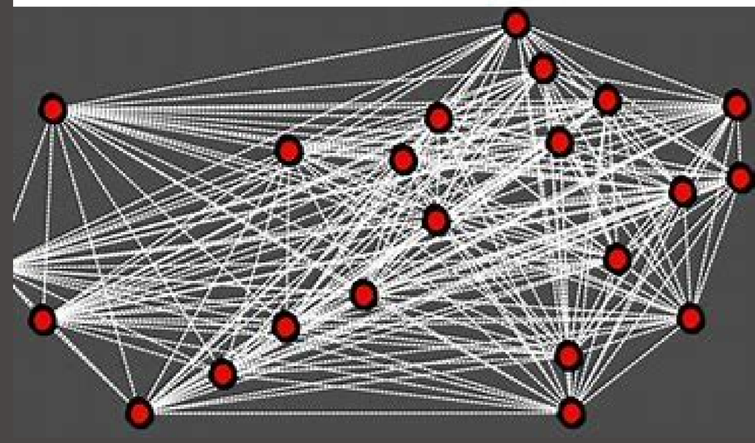
Exact Algorithms:

1. Exponential Time Complexity: Exact algorithms guarantee optimal solutions but have exponential time complexity, making them impractical for large instances.
2. High Memory Requirements: They require storing large amounts of data, leading to high memory usage, which can be limiting for large-scale instances.
3. Infeasibility for Large Instances: Due to their time complexity, exact algorithms may not be feasible for solving TSP instances with hundreds or thousands of cities within a reasonable time frame.

Heuristic Algorithms:

1. Suboptimal Solutions: Heuristic algorithms do not guarantee optimality, often producing suboptimal solutions, especially for complex TSP instances.
2. Sensitivity to Initialization: They can be sensitive to initial configurations or parameter settings, impacting solution quality and robustness.
3. Risk of Local Optima: Heuristic algorithms may get stuck in local optima, limiting their ability to find the globally optimal solution.
4. Scalability Challenges: While more scalable than exact algorithms, heuristic algorithms may still struggle with very large instances, affecting solution quality and efficiency.





Complexity of Real-world Scenarios: Real-world TSP instances often involve additional complexities such as time windows, multiple vehicles, varying travel costs, and dynamic traffic conditions. Addressing these complexities adds further challenges to TSP optimization and may require customized algorithms tailored to specific application domains.

Our Idea

Integrating the Christofides algorithm and MST construction into the genetic algorithm for solving the Traveling Salesman Problem (TSP) allows us to harness the unique strengths of each approach, resulting in a hybrid solution that is more powerful than the sum of its parts.

Approach

1. **Christofides Algorithm**: Known for its near-optimal solutions, the Christofides algorithm constructs an MST and converts it into a Hamiltonian tour, guaranteeing solutions no worse than 3/2 times the optimal.
2. **Genetic Algorithm (GA):** Utilizing evolution-inspired techniques, the GA efficiently explores the solution space, iteratively improving candidate solutions through selection, crossover, and mutation.
3. **Integration:** By integrating the Christofides algorithm and MST construction into the GA, we create a symbiotic relationship where the GA provides flexibility and adaptability, while the Christofides algorithm offers reliable solution refinement.

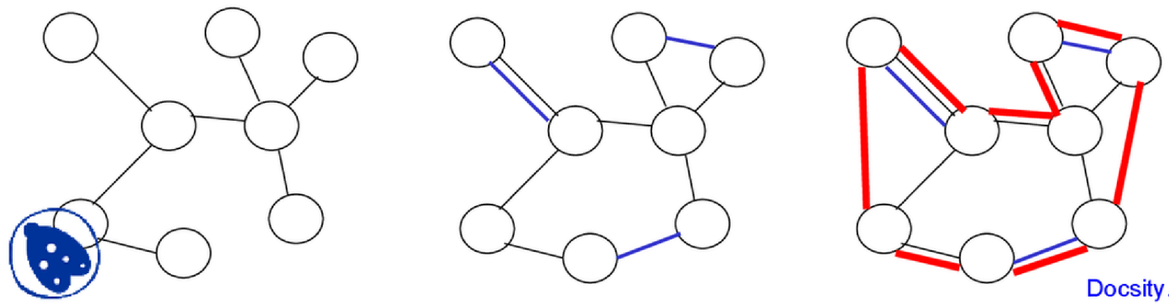
Christofides' algorithm for TSP

Given an instance for TSP problem,

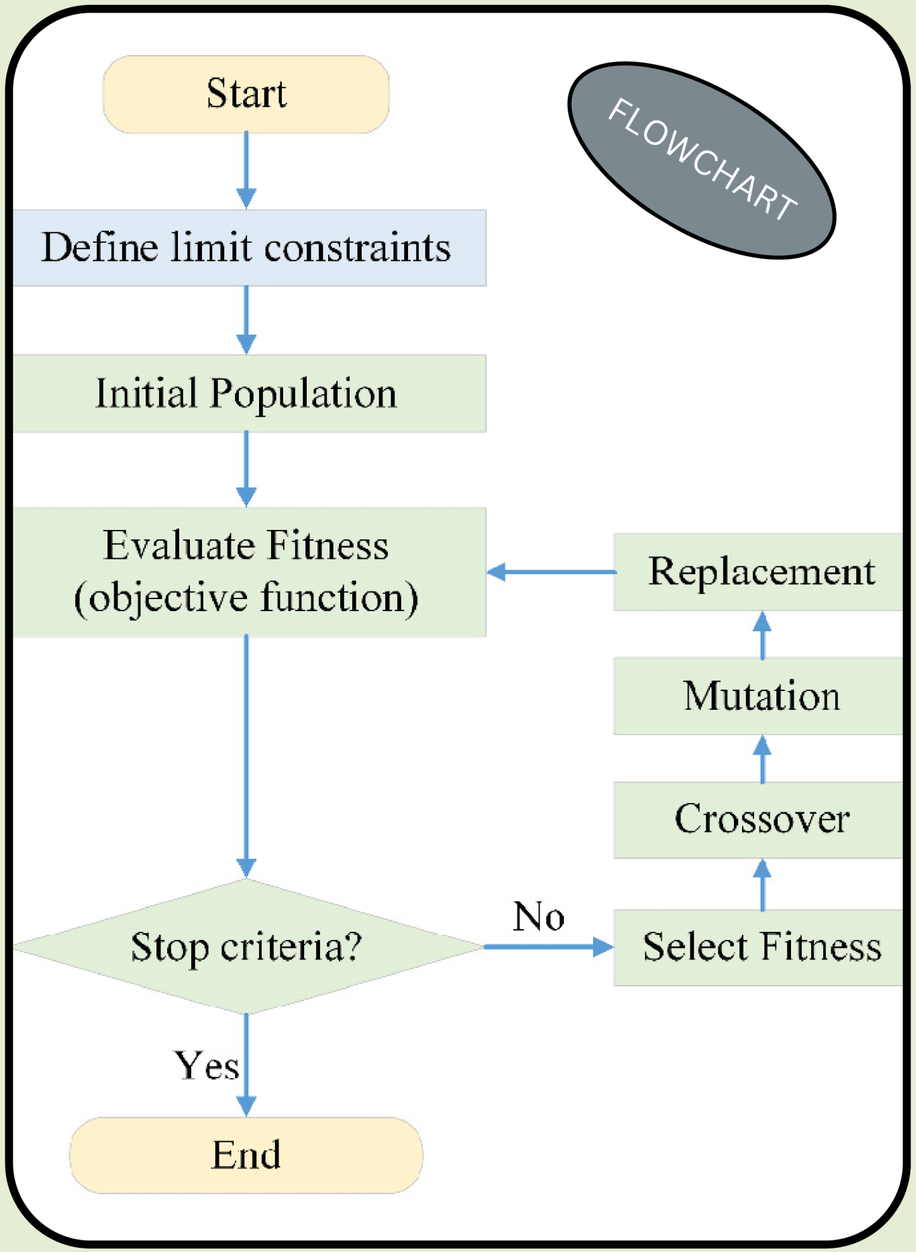
1. Find a minimum spanning tree **T** for that instance.
2. Find a min-cost perfect matching **M** for odd-degree nodes of **T**.
3. **T** ∪ **M** has an Euler tour; traverse the arcs of the Euler tour, by taking shortcuts when necessary, to get a **TSP tour**.

Example:

Stage 1 | Stage 2 | Stage 3



GENETIC ALGORITHM



In the following implementation, cities are taken as genes, string generated using these characters is called a chromosome, while a fitness score which is equal to the path length of all the cities mentioned, is used to target a population.Fitness Score is defined as the length of the path described by the gene. Lesser the path length fitter is the gene. The fittest of all the genes in the gene pool survive the population test and move to the next iteration. The number of iterations depends upon the value of a cooling variable. The value of the cooling variable keeps on decreasing with each iteration and reaches a threshold after a certain number of iterations.

Benefits:

1. **High-Quality Solutions:** The Christofides algorithm ensures near-optimal solutions, while the GA efficiently refines and explores the solution space.
2. **Adaptability:** The hybrid approach adapts to different problem instances and solution characteristics, offering flexibility in tackling various TSP scenarios.
3. **Efficiency:** Leveraging the strengths of both methods, the hybrid optimization approach efficiently solves the TSP, producing high-quality solutions in a timely manner.

Give Github repo link

https://github.com/DSA-IITJ-2024/ideathon-code-submission-gopaljyani2005.git

Highlight results

we have obtained a path in the range of 1 to 1.5 time of the optimal path by christofides algorithm and then optimized it by 100 generations

Analyze the cost

Evolution: O(g * m * n^2)

Initialization: O(n^3)

Christofides Algorithm:

Time Complexity: O(n^3 log n)

Space Complexity: O(n^2)

MST construction can be performed in O(n^2 log n) time using Kruskal's or Prim's algorithm.

Benefits:

Genetic Algorithm (GA):

Time Complexity: O(g * n * m), where g is the number of generations, n is the population size, and m is the complexity of fitness evaluation.

Space Complexity: O(n * m), where n is the population size and m is the size of the solution representation.

Conclusion

By integrating the Christofides algorithm and MST construction into the Genetic Algorithm framework, our hybrid approach offers a compelling solution strategy for efficiently tackling the TSP. It achieves a delicate balance between solution quality and computational efficiency, making it suitable for real-world applications with large-scale TSP instances.

Throughout this project, we have gained valuable insights and learnings that contribute to our understanding of optimization algorithms and problem-solving techniques. Key findings include:

- The importance of leveraging diverse optimization methods: By integrating the Christofides algorithm, Minimum Spanning Tree (MST) construction, and Genetic Algorithm (GA), we have witnessed the benefits of combining different approaches to address complex optimization problems like the Traveling Salesman Problem (TSP).
- The significance of hybridization: Our exploration of hybrid optimization approaches has highlighted the potential for synergistic interactions between algorithms, leading to enhanced solution quality, efficiency, and robustness.

The scope of future extension

Future extensions of the proposed hybrid optimization approach for the Traveling Salesman Problem (TSP) present exciting opportunities for further innovation and improvement. Here are some potential areas of expansion:

1] Hybridization with Other Metaheuristic Techniques:

- Explore combinations with other metaheuristic techniques and Investigate how the integration of multiple metaheuristic techniques can further enhance solution quality and convergence speed.

2] Dynamic Adaptation and Parameter Tuning:

- Develop adaptive strategies to dynamically adjust algorithm parameters (e.g., mutation rate, population size) during runtime based on problem characteristics or solution progress.
- Implement mechanisms for automatic parameter tuning using machine learning or adaptive algorithms to optimize performance across diverse problem instances.

3] Parallel and Distributed Computing:

- Investigate efficient parallelization strategies for genetic algorithms and Christofides algorithm to exploit the computational resources effectively.

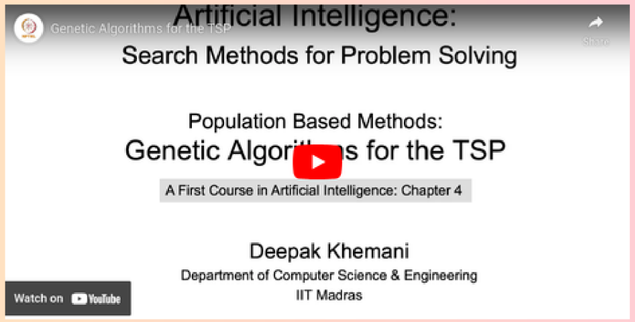
4] Integration with Problem-Specific Heuristics:

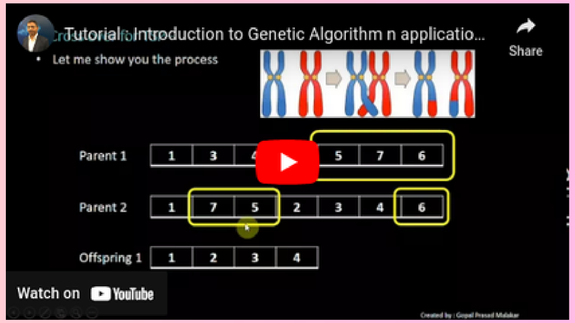
- Incorporate problem-specific heuristics or domain knowledge into the optimization process to tailor the algorithm's behavior to specific problem instances or application domains.
- Investigate how integrating domain-specific knowledge can lead to further improvements in solution quality and convergence speed for TSP instances with unique characteristics.

5] Real-World Applications and Benchmarking:

- Apply the hybrid optimization approach to real-world TSP instances arising in logistics, transportation, and network routing domains.
- Conduct comprehensive benchmarking studies to evaluate the performance of the proposed approach against state-of-the-art algorithms and assess its practical applicability and scalability.

Acknowledgement / References





https://www.bing.com/ck/a?!&&p=2cf222a4d76b4f8fJmltdHM9MTcxNDQzNTIwMCZpZ3VpZD0xZTY2NWRmOS1kMWE5LTY0MWItMDFmZS00ZDhlZDAxYjY1ZGUmaW5zaWQ9NTIyNA&ptn=3&ver=2&hsh=3&fclid=1e665df9-d1a9-641b-01fe-4d8ed01b65de&psq=christofides+algorithm+with+Integer+Linear+Programming++github&u=a1aHR0cHM6Ly9naXRodWIuY29tL0Fya2F5OTIvVFNQLUNocmlzdG9maWRlcy1BbGdvcml0aG0&ntb=1

https://www.bing.com/ck/a?!&&p=2320aa8a6ed24b48JmltdHM9MTcxNDQzNTIwMCZpZ3VpZD0xZTY2NWRmOS1kMWE5LTY0MWItMDFmZS00ZDhlZDAxYjY1ZGUmaW5zaWQ9NTQwNQ&ptn=3&ver=2&hsh=3&fclid=1e665df9-d1a9-641b-01fe-4d8ed01b65de&psq=geeks+for+geeks++algorithm+for+TSP+&u=a1aHR0cHM6Ly93d3cuZ2Vla3Nmb3JnZWVrcy5vcmcvdHJhdmVsaW5nLXNhbGVzbWFuLXByb2JsZW0tdXNpbmctZ2VuZXRpYy1hbGdvcml0aG0v&ntb=1

All four members have contributed equally to each component of the project, ensuring a collaborative effort and balanced workload distribution.

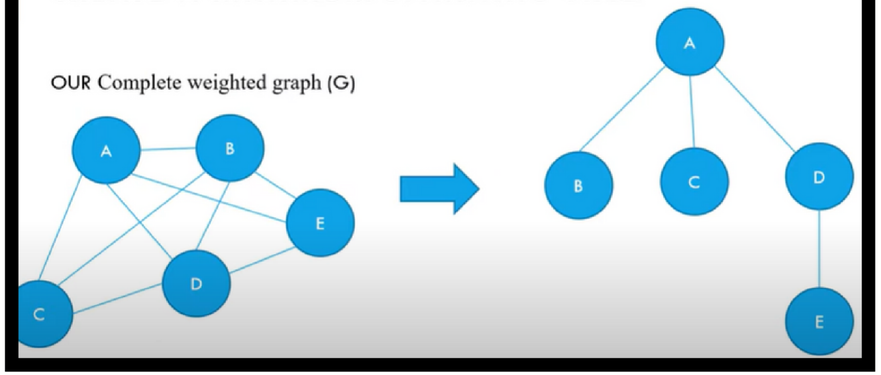
Rough

Kruskals OR Prims

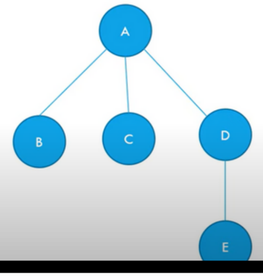
Bipartite matching Algorithm

CREATE A MINIMUM SPANNING TREE

OUR Complete weighted graph (G)

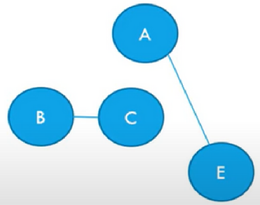


MAKE A SET OF VERTICES THAT HAVE AN ODD DEGREE

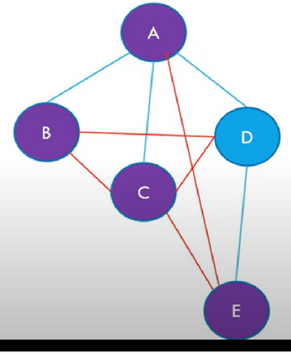


{B,C,A,E}

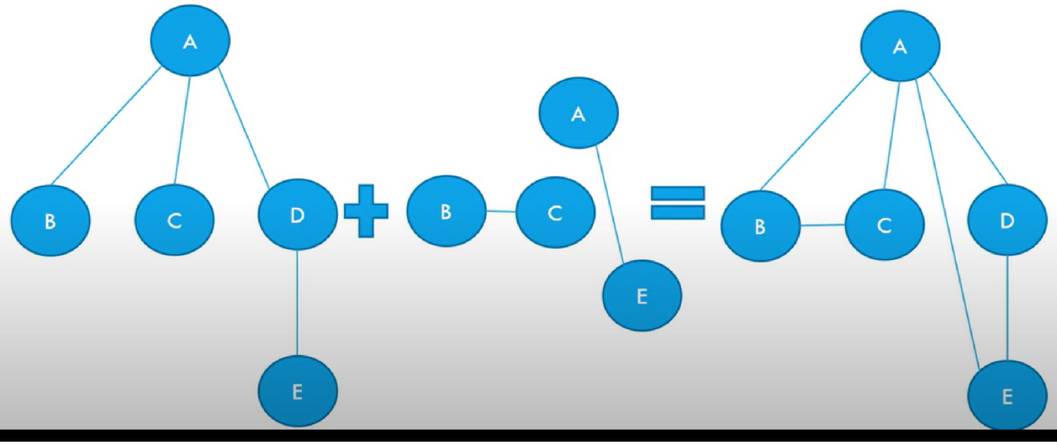
FIND A MINIMUM-COST, PERFECT MATCHING



{B,C,A,E}



COMBINE PERFECT MATCH TO MINIMUM SPANNING TREE



SHORTCUT BY REMOVING ALL REPEATED VERTICES

{A,B,C,A,E,D,A}

{A,B,C,E,D,A}