Report OF

A combined ant colony optimization with Levy flight mechanism for the probabilistic traveling salesman problem with deadlines

Author

Amir Hossein Kamranpour

University

KNTU

Course

Advanced Algorithm

January 24, 2025

Abstract

This report delves into the Probabilistic Traveling Salesman Problem with Deadlines (PTSPD), a challenging combinatorial optimization problem, and proposes an innovative solution by integrating the Levy Flight Mechanism into the Ant Colony Optimization (ACO) algorithm. The combination aims to overcome PTSPD's inherent complexities by leveraging the strengths of both approaches.

Due to the complexity of the original Levy Flight Mechanism, a simplified version with a new formula is developed to enhance its practicality and adaptability within the ACO framework. The modified Levy Flight Mechanism probabilistically adjusts the next-step selection of ants, allowing them to explore less-visited areas of the graph. This strategic adjustment balances exploitation and exploration, significantly reducing the likelihood of being trapped in local optima.

The results obtained from this enhanced method are analyzed and compared against precise implementations using Python. Through these evaluations, the report concludes with insights into the effectiveness and efficiency of the proposed hybrid algorithm for solving PTSPD.

1 Probabilistic Traveling Salesman Problem with Deadlines (PTSPD)

The Probabilistic Traveling Salesman Problem with Deadlines (PTSPD) is a variant of the classic Traveling Salesman Problem (TSP), where the challenge is to visit a set of customers who may not all be available at the time of the visit. This problem introduces elements of uncertainty regarding the availability of customers and incorporates time constraints or deadlines that must be met. PTSPD is a complex combinatorial optimization problem that is NP-Hard, making it computationally challenging to solve for large instances.

1.1 Problem Definition

In PTSPD, a salesman must visit each customer exactly once, but unlike the classical TSP, each customer has a probability p_i of being available for service at any given time. Moreover, customers have deadlines by which they must be visited. The objective of the PTSPD is to minimize the expected total travel time while ensuring that the deadlines are met and that the availability of customers is respected. The problem is typically formulated as follows:

- $N = \{1, 2, ..., n\}$: A set of customers, where each customer i has a probability p_i of being available.
- d_{ij}: The Euclidean distance between customer i and customer j, which can be calculated using the distance formula.
- l_i: The deadline by which customer i must be visited.
- λ_i: The penalty incurred for missing the deadline for customer i.

1.2 Objective Function

The goal is to find a route that starts and ends at the depot, visits each customer exactly once, and minimizes the expected travel time. The expected travel time and penalty are both important components of the objective function. Mathematically, the objective function is typically represented as:

$$E(travel\ time\ for\ \xi) + E(penalties\ for\ \xi)$$

where E denotes the expected value. The expected travel time is a weighted sum of the distances between customers, taking into account the probability of each customer being visited. The expected penalties are computed based on the likelihood of missing deadlines.

1.3 Constraints

The PTSPD involves both probabilistic and time-based constraints. These constraints are what make PTSPD more complex than the classic TSP:

- **Availability Constraints:** Each customer has a probability p_i of needing service. Customers with a lower probability may be skipped during the tour.
- **Deadline Constraints:** Each customer must be visited within a specified time window defined by their deadline. If the deadline is missed, a penalty λ_i is incurred.

These constraints make PTSPD a stochastic optimization problem where the uncertainty in customer availability must be accounted for, adding complexity to the problem formulation.

1.4 Applications

The PTSPD has numerous real-world applications, especially in fields where uncertainty and deadlines are crucial factors. Some of the most notable applications include:

- **Logistics and Delivery:** Planning routes for delivery vehicles, where the time of arrival and customer availability are uncertain.
- **Maintenance and Service:** Scheduling visits for maintenance personnel or service providers, where some customers may not need service, and deadlines must be met to avoid penalties.
- **Healthcare Services:** Optimizing routes for emergency or routine healthcare visits, where availability of patients and time constraints are key factors.

Given the computational complexity of PTSPD, finding an optimal solution requires advanced optimization techniques, particularly for large instances or real-time decision-making scenarios.

2 Model Formulation

The Probabilistic Traveling Salesman Problem with Deadlines (PTSPD) is defined as a problem where a salesman must find an optimal tour starting and ending at a depot, visiting a subset of customers, with each customer having a probability of needing service and a specific deadline by which they must be visited.

2.1 Mathematical Formulation

Let the set of customers be denoted by $N = \{1, 2, ..., n\}$, where 0 represents the depot. The distances between customers are given by the Euclidean distance d_{ij}, where $d_{ij} = d_{ji}$, for all $i, j \in N \cup \{0\}$. Each customer i has an associated probability p_i of needing service, a deadline l_i, and a penalty λ_i for missing the deadline.

The objective is to minimize the expected travel time while considering the penalties for missing deadlines. The expected travel time and penalty for a given tour ξ is calculated as:

$$E(travel\ time\ for\ \xi) + E(penalties\ for\ \xi)$$

where:

$$E(travel\ time\ for\ \xi) = \sum_{j=1}^{n} p_j d_{0j} \prod_{k=1}^{j-1}(1-p_k) + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} p_i p_j d_{ij} \prod_{k=i+1}^{j-1}(1-p_k) + \sum_{i=1}^{n} p_i d_{id_0} \prod_{k=i+1}^{n}(1-p_k)$$

This equation represents three terms:

- The travel time from the depot to the first customer,
- The travel time from one customer to another,
- The travel time from the last customer back to the depot.

Each of these travel times is weighted by the probability of visiting each customer.

The penalties for missing the deadlines are calculated as:

$$E(penalties\ for\ \xi) = \sum_{i=2}^{n} p_i \lambda_i V_i$$

where V_i is an indicator variable defined as:

$$V_i = \begin{cases} 1 & \text{if the expected arrival time at customer } i \text{ is later than its deadline,} \\ 0 & \text{otherwise.} \end{cases}$$

Thus, the objective is to minimize the sum of the expected travel time and expected penalties.

2.2 Solution Representation

The solution is represented by a permutation $\xi : \{0, 1, ..., n\} \to N \cup \{0\}$ where $\xi(0) = 0$ and $\xi(i) \neq 0$ for all $i > 0$. The permutation defines the order in which customers are visited.

2.3 Recourse Model

The solution is obtained in two phases:

- Phase 1: A preliminary solution is generated, determining the sequence in which customers will be visited.
- Phase 2: A stochastic variable is used to simulate the random availability of customers, and a recourse action is taken based on the initial solution.

The recourse model allows adjustments to the tour based on the realizations of customer availability, ensuring that the deadlines are met and minimizing penalties.

2.4 Objective Function

The objective function is to minimize the total cost, which includes both the expected travel time and the penalties for missed deadlines. The final optimization problem is:

$$\text{Minimize } E(travel\ time\ for\ \xi) + E(penalties\ for\ \xi)$$

subject to the constraints:

- The tour must start and end at the depot,
- Each customer must be visited exactly once,
- The deadlines must be respected.

3 Ant Colony Optimization (ACO)

In this section, we will explore Ant Colony Optimization (ACO), a metaheuristic algorithm inspired by the natural foraging behavior of ants. In nature, ants communicate and find optimal paths through the deposition of pheromones, which act as a form of indirect communication. When ants follow a pheromone trail, they are more likely to continue down paths with stronger pheromone intensity, which eventually leads them to the shortest or most efficient routes. ACO mimics this natural process to solve combinatorial optimization problems, such as the Probabilistic Traveling Salesman Problem with Deadlines (PTSPD).

ACO operates by simulating a colony of artificial ants that work together to find an optimal solution. These ants traverse a graph representing the problem, with each node representing a customer or location to be visited, and the edges representing possible routes between these nodes. The fundamental mechanism of ACO is based on the concept of pheromone trails. As ants move from one node to another, they deposit pheromones on the edges they travel. The intensity of these pheromone trails is proportional to the quality of the solution (i.e., the shorter or better the route, the more pheromone is deposited). This allows subsequent ants to be influenced by the pheromone trails, guiding them towards more promising paths.

One of the key features of ACO is the probabilistic decision-making process. At each step of their journey, ants probabilistically select the next node to visit, based on a combination of the pheromone intensity and a heuristic function (such as the inverse of the distance between nodes). This means that the ants are not guaranteed to choose the shortest route initially; rather, they explore various paths, and over time, the best paths are reinforced through pheromone deposition, while less optimal paths evaporate pheromones and are less likely to be selected.

The algorithm proceeds iteratively, with each iteration consisting of multiple ants traversing the graph. After all ants have completed their tours, the pheromone values are updated based on the quality of the solutions they have found. This update step

strengthens the pheromone trails on better paths, encouraging future ants to explore these paths, while reducing the pheromone levels on worse paths. The pheromone evaporation rate is a critical parameter in ACO, as it ensures that less optimal solutions do not dominate the search space over time.

ACO's main strength lies in its ability to perform parallel searches. Multiple ants explore the solution space simultaneously, allowing the algorithm to quickly search through different potential solutions and avoid getting stuck in local optima. However, despite these advantages, ACO has some limitations. For example, it may still converge prematurely to suboptimal solutions if the pheromone levels become too concentrated on specific paths early on in the search process. To address this issue, hybrid approaches such as combining ACO with the Levy Flight Mechanism are used to enhance exploration and prevent premature convergence.

In summary, ACO is a powerful and adaptive metaheuristic that has shown significant success in solving complex optimization problems like PTSPD. By using pheromone trails to guide the search process and updating these trails iteratively, ACO can efficiently explore a large solution space and find high-quality solutions to problems that would otherwise be computationally intractable.

4 ACO for PTSPD

In this section, we delve into the application of Ant Colony Optimization (ACO) for solving the Probabilistic Traveling Salesman Problem with Deadlines (PTSPD). The PTSPD is a stochastic variant of the classic Traveling Salesman Problem (TSP), where customer availability is probabilistic, and each customer has a strict deadline for being visited. This additional layer of complexity requires modifications to the traditional ACO algorithm to efficiently solve the problem while considering both customer availability and deadline constraints.

4.1 Problem Setup for ACO

The PTSPD is formulated as a graph where each node represents a customer or the depot, and the edges represent the possible routes between them. The objective of the problem is to find the optimal route that minimizes the expected travel time while ensuring that the customers are visited within their deadlines and the probabilistic constraints are respected.

In ACO for PTSPD, the ants are designed to explore various routes, with their decisions being influenced by pheromone trails and heuristic information. The problem's probabilistic nature is captured by the availability of customers, denoted by the probability p_i for each customer i. The deadlines, represented by l_i, are incorporated into the algorithm's penalty function, which is minimized during the optimization process.

4.2 The ACO Process for PTSPD

ACO for PTSPD follows the standard ACO procedure, but it is adapted to handle the specific challenges posed by PTSPD. The basic steps of the ACO algorithm are as follows:

4.2.1 Initialization:

Initially, a population of ants is placed at the depot. Each ant starts with a random path selection based on the pheromone levels and heuristic values. The pheromone levels

represent the desirability of a particular route, and the heuristic values reflect the inverse of the distance between customers.

4.2.2 Path Construction:

In each iteration, the ants probabilistically choose the next customer to visit based on the transition probability. This probability is determined by both the pheromone intensity and the heuristic value. The transition probability is given by:

$$p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_k} [\tau_{il}(t)]^\alpha [\eta_{il}]^\beta} & \text{if } j \in N_k \\ 0 & \text{else.} \end{cases}$$

where: - $\tau_{ij}(t)$ is the pheromone intensity on edge (i, j) at time t, - η_{ij} is the heuristic value, such as the inverse of the distance between nodes i and j, - α and β are parameters that control the relative importance of pheromone intensity and heuristic information, respectively, - N_k is the set of unvisited nodes by ant k.

4.2.3 Pheromone Update:

After the ants complete their tours, the pheromone levels are updated based on the quality of the solutions found. The pheromone update rule is as follows:

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^k$$

where: - ρ is the pheromone evaporation rate, - $\Delta\tau_{ij}^k$ is the amount of pheromone deposited by ant k on edge (i, j), proportional to the solution quality.

4.2.4 Iteration and Convergence:

The algorithm continues iterating, with each iteration refining the pheromone levels and guiding the ants toward the optimal path. The process repeats until a convergence criterion is met, such as a fixed number of iterations or the convergence of the pheromone levels.

4.3 Modifications for PTSPD

In PTSPD, the main modification to the standard ACO is the integration of the probabilistic availability of customers and the deadlines. These modifications influence the path construction and pheromone update steps.

4.3.1 Availability of Customers:

For each customer i, there is a probability p_i that they will need to be visited. This probability is incorporated into the path selection process. The transition probability is adjusted to account for the availability of each customer. The modified probability allows ants to explore customers with higher availability more frequently, while customers with low availability are less likely to be visited.

4.3.2 Deadlines and Penalties:

The deadline for each customer i is represented by l_i, and if an ant arrives at a customer after their deadline, a penalty λ_i is incurred. The penalty term is added to the objective function, and the ants aim to minimize both the travel time and the penalties for missed deadlines. The penalties are included in the objective function as follows:

$$E(penalties\ for\ \xi) = \sum_{i=2}^{n} p_i \lambda_i V_i$$

where V_i is an indicator variable that takes the value 1 if the expected arrival time at customer i exceeds the deadline, and 0 otherwise.

5 ACO with Levy's Flight

The Levy Flight Mechanism within ACO, as described in section 4.1, refers to a method inspired by random walks where the ants use long-distance jumps, governed by the Levy distribution, to explore the solution space. This mechanism is incorporated into ACO to address the issue of ants getting stuck in local optima. In the traditional ACO algorithm, ants select paths probabilistically based on pheromone intensity and heuristic information. However, without any mechanism to escape local optima, the algorithm can converge prematurely to suboptimal solutions.

The Levy distribution, which has a heavy-tailed behavior, allows ants to occasionally take large steps away from their current position, thus improving the algorithm's ability to explore a wider range of solutions. The formula for generating the step size S is given by:

$$S = \frac{u}{|v|^{\beta}}$$

where u and v are Gaussian-distributed random variables, and β controls the distribution's tail. The distribution parameters are calculated using the following formulas:

$$\sigma_u^2 = \frac{\Gamma(1+\beta)\sin(\frac{\pi\beta}{2})}{\Gamma(\frac{1+\beta}{2})\beta^{\frac{\beta-1}{2}}}$$

and

$$\sigma_v^2 = 1$$

where $\Gamma(z)$ is the Gamma function.

The process involves calculating the step length using Mantegna's formula. However, since this can be computationally intensive for ACO, a simplified approach is introduced, where the step length is controlled by predefined parameters such as $P_{threshold}$ and A (the altering ratio for Levy flight).

The adjusted transition probability after Levy flight is given by:

$$S_{new} = \frac{1}{A} \cdot \frac{1 - P_{threshold}}{1 - P_{Levy}}$$

This formula alters the selection probability based on the Levy flight, ensuring that longer jumps are possible when P_{Levy} exceeds the threshold.

These mechanisms in Levy Flight-ACO (LFACO) provide a more robust way of exploring the search space, helping avoid premature convergence to local optima, and ensuring more diverse solutions are generated, which is essential for solving complex combinatorial problems like PTSPD.

6 Results of Paper

In this section, a computational experiment is described to evaluate the performance of the newly developed Levy ACO (LFACO) algorithm in comparison with the traditional ACO algorithm. The study focuses on two sets of instances: the Range and Mixed datasets, both of which represent different scenarios for customer availability.

In the Range dataset, each customer has uniformly distributed probabilities.

On the other hand, the Mixed dataset includes customers with either a probability of 0.1 or 1, representing businesses of different sizes served by the same delivery person.

Both datasets feature deadlines for customers, and a penalty of 5 is applied for missing these deadlines. The deadline for each customer depends on the time window: if the window has a non-zero opening time, that time is considered the deadline; otherwise, the closing time of the window is used as the deadline.

The experiment was conducted using Python on a machine with an Intel Core i5-7200U processor, 8 GB of RAM, and a 64-bit operating system. The computational time and the quality of the obtained solutions were measured.

The experimental setup used the following parameters for both algorithms:

- $tau_0 = 0.01$
- $m = 7$
- alpha=1alpha = 1
- beta=2beta = 2
- rho=0.5rho = 0.5
- Pthreshold=0.8P$_{threshold} = 0.8A = 1A = 1$

- Q=1Q = 1

These values were chosen based on previous studies and empirical data.

6.1 Comparison of ACO and LFACO based on Expected Values

Data set Probability	LFACO Range	ACO Range	LFACO Mixed	ACO Mixed
5	5.29	5.27	2.751	2.753
22	275.69	304.87	94.97	102.91
42	252.63	387.43	222.90	247.90
62	430.23	455.35	323.10	323.81
102	673.71	677.78	629.06	685.36

Table 1: Comparison of LFACO and ACO for Range and Mixed datasets

6.2 Comparison of ACO and LFACO based on CPU Time

Data set Probability	LFACO Range	ACO Range	LFACO Mixed	ACO Mixed
5	0	0	0	0
22	1	1	1	1
42	4	6	3	6
62	21	31	20	23
102	95	206	94	273

Table 2: Comparison of LFACO and ACO for Range and Mixed datasets (Second Table)

7 Results of Amir Hossein Kamranpour

Data set	LFACO (Range)	ACO (Range)	LFACO (Mixed)
5	38.31	39.39	19.16
22	177.29	167.75	175.12
42	396.12	411.96	342.85
62	642.68	660.99	646.04
102	1013.01	1000.57	978.02

Table 3: Comparison of LFACO and ACO for Range and Mixed datasets

Parameters: $\tau_0 = 0.01, m = 7, \alpha = 1, \beta = 2, \rho = 0.5, P_{\text{threshold}} = 0.8, A = 1,$ and $Q = 1$.

Range vs Mixed Data: The "Range" dataset assigns uniformly distributed probabilities to customers, while the "Mixed" dataset uses random probabilities of 0.1 or 1.

8 Conclusion

The results obtained in my research show that the Ant Colony Optimization (ACO) algorithm with the Levy Flight (LFACO) mechanism generally performs better than the traditional ACO. However, contrary to the findings of the original paper, my findings indicate that this performance improvement does not necessarily increase with the size and complexity of the dataset (e.g., number of nodes or variables). In other words, the improvement of LFACO over ACO is not consistently proportional to the dataset's growth.

The analysis of the results also reveals that in some cases, the LFACO algorithm may not perform better and might even underperform compared to the ACO algorithm. These cases are typically influenced by the nature of the data and the parameters set for each algorithm. Nevertheless, in most experiments, LFACO demonstrated better overall performance.

The performance difference between the two algorithms is not significantly large and is

often limited to marginal improvements. This suggests that while Levy Flight enhances the search capabilities of the algorithm and helps avoid local optima, its standalone impact is limited and may not yield substantial improvements in certain problems.

Compared to the original paper's conclusion, which emphasizes the significant impact of Levy Flight, especially for larger and more complex problems, my findings provide a more moderated perspective. While Levy Flight remains a useful tool for search enhancement, its effect can be more limited depending on the specific conditions and datasets.