Bol. Soc. Paran. Mat.
©SPM – E-ISSN-2175-1188
SPM: www.spm.uem.br/bspm

(3s.) **v. 2026 (44) 3** : 1–15.
ISSN-0037-8712
doi:10.5269/bspm.79317

Enhanced Artificial Bee Colony Algorithm for the Probabilistic Traveling Salesman Problem with Deadlines

Fadoua El Asri, Chakir Tajani and Hanane Fakhouri

ABSTRACT: This study addresses the challenge of routing under uncertainty when timing constraints are critical. We propose ABC-DT, an enhanced Artificial Bee Colony algorithm developed for the #P-hard Probabilistic Traveling Salesman Problem with Deadlines (PTSPD). The Artificial Bee Colony (ABC) approach has shown strong results in many combinatorial contexts, largely due to its simple structure and its balanced mix of exploration and exploitation. Its limited number of control parameters and flexible neighborhood design make it well suited to discrete and stochastic problems such as the PTSPD, where it often surpasses conventional heuristics in both efficiency and reliability. However, standard ABC suffers from premature convergence and suboptimal handling of probabilistic structures. Thus, ABC-DT combines three innovations, which are: adaptive neighborhood operators weighted by probabilities and deadlines, dynamic parameter tuning, and hybrid scout diversification. Experiments on benchmarks (22–100+ nodes) show 7–23% lower expected costs, with reduced variability compared to ABC-standard. Validation studies confirm this (+10–15% costs without weighted operators). The algorithm scales well, handling 152-node instances, and is flexible for fixed or proportional penalties, making it viable for urgent logistics.

Key Words: Probabilistic traveling Salesman problem with deadlines, stochastic optimization, metaheuristics, artificial bee colony.

Contents

1. Introduction

In recent years, logistics systems have had to manage growing levels of uncertainty while still ensuring punctual deliveries. Whether in medical dispatching, online retail, or express transport, decision makers must plan routes that adapt to unpredictable customer availability and strict delivery times. This combination of randomness and time pressure leads to optimization problems that are both stochastic and combinatorial in nature. Classical deterministic formulations fail to capture this dual aspect, which motivated the definition of stochastic routing models such as the Probabilistic Traveling Salesman Problem with Deadlines (PTSPD).

2020 *Mathematics Subject Classification:* 90C27, 90C15, 90C59, 68T20.
Submitted October 02,2025. Published February 22, 2026

The PTSPD is a direct extension of the Probabilistic Traveling Salesman Problem (PTSP), first proposed in [1]. The original PTSP described uncertainty in customer attendance through individual visit probabilities. A few years later, [2] added delivery deadlines, which introduced a temporal dimension and made the model closer to real delivery operations. Although this version captures realistic behavior, it is computationally heavy. In fact, [3] showed that the PTSPD belongs to the #P-hard class, which makes exact solutions practically impossible for large instances.

Early work mainly relied on classical exact algorithms, including dynamic programming and branch-and-bound. These techniques can reach optimal solutions, but their runtime grows exponentially as the number of customers increases. To limit computation time, heuristic and approximation methods were developed [4]. They often perform well on small instances, but accuracy deteriorates as the uncertainty grows or the instance size increases. This observation motivated a shift toward population-based metaheuristics, which can navigate large, irregular search spaces more effectively.

Among these approaches, Ant Colony Optimization (ACO) has been applied in probabilistic routing [5,6]. Variants that incorporate Lévy-flight steps, as well as the Variable Neighborhood Search (VNS) framework, have also yielded competitive results [7,8]. On the computing side, progress in Monte Carlo sampling, distributed execution, and GPU-based acceleration has improved scalability and reduced evaluation time [9,10]. Taken together, these strands of work have established metaheuristics as a practical option for large stochastic routing instances.

Within this research context, the Artificial Bee Colony (ABC) algorithm has drawn particular interest for its simple design and solid search capability. Originally introduced in [11] for continuous optimization, it was later adapted to discrete and combinatorial domains [12,13]. Its applications now cover scheduling, routing, production planning, disassembly line balancing, and energy optimization [14,15,16,17,18,19,20]. A comprehensive review of these applications can be found in [21]. In stochastic routing problems, ABC has often shown performance comparable to — and sometimes better than — that of classical heuristics [15].

However, the standard ABC is not ideally suited to problems such as the PTSPD. Its neighborhood generation process is random and does not take into account the probabilistic or temporal nature of the task, often leading to premature convergence. Moreover, the fixed control parameters make the algorithm less flexible when probabilities or deadlines vary across instances. These factors tend to limit its robustness, especially when randomness is high or time constraints are tight.

In this work, we introduce an enhanced variant called ABC-DT (Artificial Bee Colony with Deadline Tuning). The proposed method addresses the main shortcomings of the standard ABC on the PTSPD. It steers the search using both probability cues and deadline urgency. Customers with high activation probability or tight due dates are sampled more frequently, which improves the balance between expected travel cost and penalties. Control parameters are adapted during the search so that the algorithm can shift between diversification and intensification as the search progresses. A hybrid scout routine combines elite-guided adjustments with random restarts to preserve diversity and reduce stagnation. Additional refinements—adaptive neighborhood sizing and a soft acceptance rule inspired by simulated annealing—further stabilize convergence and enhance robustness.

Extensive computational experiments were performed on PTSPD benchmark instances containing 22 to 152 nodes and under several probabilistic scenarios. The results show that ABC-DT consistently outperforms the standard ABC in terms of expected cost, convergence speed, and robustness. Furthermore, comparisons with ACO-Lévy [8] and VNS [7] confirm the adaptability and scalability of the proposed method for stochastic routing with time constraints.

The remainder of this paper is organized as follows. Section 2 describes the mathematical formulation of the PTSPD. Section 3 outlines the standard ABC and the proposed ABC-DT algorithms. Section 4 presents and discusses the experimental results, while Section 5 concludes the paper and outlines future research directions.

2. Probabilistic Traveling Salesman Problem with Deadlines (PTSPD)

The Probabilistic Traveling Salesman Problem with Deadlines (PTSPD) extends the probabilistic TSP by introducing a deadline for each customer. We consider a complete undirected graph $G = (V, E)$ with a depot 0 and customers $1, \ldots, n$. Each edge $(i, j) \in E$ has a nonnegative distance d_{ij}. Customer i

is independently present with probability $p_i \in [0,1]$ and must be served before a given deadline $l_i > 0$. If service at customer i starts after l_i, a penalty is incurred—either a fixed amount $\phi_i > 0$ or a proportional amount with rate $\lambda_i > 0$. An *a priori* tour is defined as $\sigma = (0, \sigma_1, \ldots, \sigma_n, 0)$, representing the planned visiting order before the realization of customer presence.

2.1. Expected Travel Distance

Since customer attendance is random, the executed tour skips absent customers but preserves the order defined by σ. To compute the expected travel distance, consider the contribution of each edge $(\sigma(i), \sigma(j))$ with $i < j$. This edge is traversed only when both customers $\sigma(i)$ and $\sigma(j)$ are present, and all intermediate customers $\sigma(k)$ for $i < k < j$ are absent. Assuming independence, the probability that all these intermediate customers are absent equals $\prod_{k=i+1}^{j-1}(1-p_{\sigma(k)})$. By linearity of expectation [2], the expected travel distance is:

$$\begin{aligned}\mathbb{E}[T(\sigma)] = &\sum_{i=1}^{n}\sum_{j=i+1}^{n} p_{\sigma(i)}\, p_{\sigma(j)}\, d_{\sigma(i),\sigma(j)} \prod_{k=i+1}^{j-1}\big(1-p_{\sigma(k)}\big) \\ &+ \sum_{j=1}^{n} p_{\sigma(j)}\, d_{0,\sigma(j)} \prod_{k=1}^{j-1}\big(1-p_{\sigma(k)}\big) \;+\; \sum_{i=1}^{n} p_{\sigma(i)}\, d_{\sigma(i),0} \prod_{k=i+1}^{n}\big(1-p_{\sigma(k)}\big).\end{aligned} \tag{2.1}$$

This expression weights each potential edge by the probability that its two customers are present while all customers between their positions on σ are absent.

2.2. Expected Penalty

In addition to travel distance, the PTSPD includes penalties for late service. Let $A_{\sigma(i)}$ denote the (random) start-of-service time at customer $\sigma(i)$, and let $\mathbb{E}[A_{\sigma(i)}]$ be its expected value. The penalty term depends on the probability that service starts after the deadline, $P\big(A_{\sigma(i)} > l_{\sigma(i)}\big)$. Computing these probabilities exactly is #P-hard even for Euclidean instances [3]. Therefore, the literature uses accurate and efficient approximations.

A standard $O(n^2)$ recursion for $\mathbb{E}[A_{\sigma(i)}]$ along the tour is [4]:

$$\mathbb{E}[A_{\sigma(i)}] = \sum_{h=1}^{i-1} p_{\sigma(h)}\,\big(\mathbb{E}[A_{\sigma(h)}] + d_{\sigma(h),\sigma(i)}\big) \prod_{k=h+1}^{i-1}\big(1-p_{\sigma(k)}\big), \qquad \mathbb{E}[A_{\sigma(1)}] = d_{0,\sigma(1)}. \tag{2.2}$$

Using these expected times, two penalty approximations are commonly employed:

Fixed penalties [4]:

$$\mathbb{E}[P_\sigma] \;\approx\; \sum_{i=1}^{n} p_{\sigma(i)}\,\phi_{\sigma(i)}\,\mathbf{1}\big\{\mathbb{E}[A_{\sigma(i)}] > l_{\sigma(i)}\big\}, \tag{2.3}$$

where $\mathbf{1}\{\cdot\}$ equals 1 if the condition holds, and 0 otherwise.

Proportional penalties [3]:

$$\mathbb{E}[P_\sigma] \;\approx\; \sum_{i=1}^{n} p_{\sigma(i)}\,\lambda_{\sigma(i)}\,\max\big\{0,\, \mathbb{E}[A_{\sigma(i)}] - l_{\sigma(i)}\big\}. \tag{2.4}$$

2.3. Objective Function

The total expected cost of an *a priori* tour σ combines the expected travel distance and the expected penalties:

$$\mathbb{E}[C(\sigma)] \;=\; \mathbb{E}[T(\sigma)] \;+\; \mathbb{E}[P_\sigma]. \tag{2.5}$$

The PTSPD goal is to find an a priori tour $\sigma = (0, \sigma_1, \ldots, \sigma_n, 0)$ that minimizes this expected cost:

$$\min_{\sigma \in S_n} \; \mathbb{E}[C(\sigma)],$$

where S_n denotes the set of permutations of $\{1, \ldots, n\}$.

3. Artificial Bee Colony Algorithms for PTSPD

The mathematical model of the PTSPD, presented in Section 2, highlights the strong coupling between stochastic and combinatorial aspects, which makes exact optimization impractical for realistic problem sizes. To address this complexity, we employ the Artificial Bee Colony metaheuristic and compare two variants: the baseline ABC-standard and an enhanced implementation, ABC-DT (Deadline Tuning).

3.1. ABC-Standard

The standard ABC algorithm [22] builds on the original formulation proposed in [11] and adapts it to discrete routing settings such as the PTSPD. In this context, each candidate solution—often referred to as a *tour*—represents an ordered list of nodes that defines a complete route. Throughout the optimization process, different types of artificial bees cooperate to improve these tours: Employed bees explore modifications around their current solutions, while onlooker bees tend to follow routes with higher fitness. Occasionally, scout bees abandon unpromising tours and introduce new random ones to refresh the population.

This cooperation between exploration and exploitation enables the algorithm to move effectively between local search and global diversification, which accounts for its strong performance in various combinatorial problems [23].

Because of its simplicity and limited number of control parameters, the standard ABC is often used as a baseline for evaluating more adaptive or problem-specific extensions on PTSPD instances.

The algorithm generates N_{tours} candidate solutions in the initialization step. Each tour $\sigma_i = (0, \sigma_i(1), \ldots, \sigma_i(n), 0)$ is a permutation of the n clients that starts and ends at the depot, where σ_i denotes the i-th tour and $\sigma(i)$ is the customer located at position i. The cost $\mathbb{E}[C(\sigma_i)]$ of each tour is then estimated using the approximations given in Section 2.

During the employed bee phase, every current solution σ_i is modified to generate a neighbor σ_i'. This is done by randomly swapping positions within the tour, a simple yet effective way to explore new candidate routes:

$$\sigma_i' = (\sigma_i(1), \ldots, \sigma_i(a-1), \sigma_i(b), \sigma_i(a+1), \ldots, \sigma_i(b-1), \sigma_i(a), \sigma_i(b+1), \ldots, \sigma_i(n)), \tag{3.1}$$

where indices $a, b \in \{1, \ldots, n\}$ are sampled uniformly, and the exchange count k adheres to Uniform($k_{\min} = n/2, k_{\max} = n$), with bounds proportional to n. Substitution ensues if $\mathbb{E}[C(\sigma_i')] < \mathbb{E}[C(\sigma_i)]$.

The onlooker bee phase focuses on evaluating promising solutions. Tours are selected with a probability proportional to their fitness:

$$g(\sigma_i) = \frac{1}{\mathbb{E}[C(\sigma_i)]}, \quad p_i = \frac{g(\sigma_i)}{\sum_{j=1}^{N_{\text{tours}}} g(\sigma_j)}, \tag{3.2}$$

A fraction $p_c \in (0, 1)$ of the tours is then chosen and perturbed according to Equation (3.1), while retaining the best candidates.

Finally, the scout bee phase replaces solutions that have not improved for C_s cycles with new, randomly generated tours, ensuring that the search does not stagnate. This balance between random exploration and fitness-based exploitation allows the algorithm to cover the solution space effectively. Parallel evaluation is used when the number of tours is large, and caching avoids redundant cost computations.
The full procedure is summarized in Algorithm 1 and its diagram is illustrated in Figure 1.

Algorithm 1 (ABC-Standard for the PTSPD)
Input: *population size N_{tours}, $p_c \in (0, 1)$, stagnation limit C_s, maximum iterations $I_{\max}$.*
Output: *$\sigma^\star$ minimizing $f(\sigma) = \mathbb{E}[C(\sigma)]$.*

1. ***Initialization:*** *generate N_{tours} random tours $\{\sigma_i\}$, set counters $s_i \leftarrow 0$, evaluate $\mathbb{E}[C(\sigma)]$, and set $\sigma^\star \leftarrow \arg\min_i \mathbb{E}[C(\sigma)]$.*

2. *For $t = 1, \ldots, I_{\max}$ do*

(a) ***Employed phase:*** *for each i, build σ_i' by applying k independent random swaps of positions in σ_i; accept if $\mathbb{E}[C(\sigma_i')] < \mathbb{E}[C(\sigma)]$, else $s_i \leftarrow s_i + 1$.*

(b) ***Onlookers phase:*** *Select $\lfloor p_c\, N_{tours} \rfloor$ tours according to Eq. 3.2 and apply a k-swap; accept improvements only.*

(c) ***Scouts phase:*** *for any i with $s_i \geq C_s$, reinitialize σ_i at random, set $s_i \leftarrow 0$, and evaluate $\mathbb{E}[C(\sigma_i)]$.*

3. ***Return*** $\sigma^\star$ *and* $\mathbb{E}[C(\sigma^\star)]$.

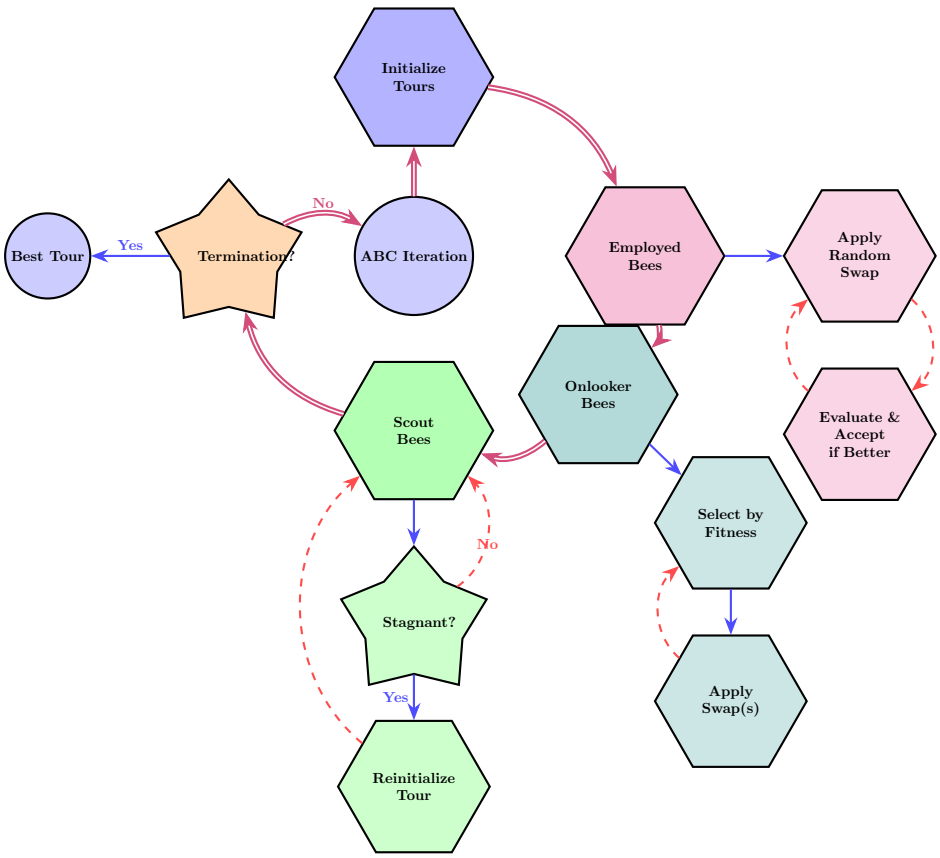


Figure 1: Flowchart of the ABC-standard algorithm.

3.2. ABC-DT

The cost structure of the PTSPD is not trivial. It depends jointly on the activation probabilities of the customers p_i, the travel distances d_{ij}, and the expected arrival times $\mathbb{E}[A_{\sigma(i)}]$. Because of this interplay, simple random exchanges between nodes are often not enough: they tend to produce tours that accumulate unnecessary penalties or waste time when probabilities are heterogeneous or deadlines are tight. As a result, the algorithm may converge slowly, and its performance can fluctuate considerably. This is precisely the motivation for introducing ABC-DT, whose modifications are designed with the particularities of the PTSPD in mind.

The ABC-DT version extends the Artificial Bee Colony algorithm so that it better reflects the probabilistic and time-limited nature of the PTSPD. It combines adaptive neighborhood exploration, a dynamic adjustment of parameters, and an improved scout stage that helps keep population diversity and avoids early convergence.

Distance-Weighted (DW) Operator: This operator gives more importance to nodes that are likely to be visited and have tighter deadlines. For a current node i, the probability of selection is defined as:

$$w_i^{\mathrm{DW}} = \frac{p_{\sigma(i)}}{l_{\sigma(i)} + \epsilon},$$

where $p_{\sigma(i)}$ and $l_{\sigma(i)}$ represent the activation probability and the deadline of node $\sigma(i)$. Among the k_{neigh} nearest neighbors of $\sigma(i)$, the next node j is selected using:

$$w_j^{\mathrm{DW}} = \frac{p_{\sigma(j)}}{d_{\sigma(i),\sigma(j)} + \epsilon}.$$

This approach corresponds to a distance-based Restricted Candidate List (RCL) that reduces unnecessary evaluations and keeps the moves spatially coherent.

Shortest-Path (SP) Operator: When time limits dominate, ABC-DT switches to the SP operator. In this case, the weights are computed as:

$$w_i^{\mathrm{SP}} = \frac{1}{l_{\sigma(i)} + \epsilon}, \qquad w_j^{\mathrm{SP}} = \frac{1}{d_{\sigma(i),\sigma(j)} + \epsilon}.$$

This operator focuses on urgent and nearby nodes, reducing the expected tardiness $\mathbb{E}[T(\sigma)] + \mathbb{E}[P_\sigma]$. The choice between DW and SP depends on the probability dispersion $\{p_i\}$ and deadlines $\{l_i\}$ tightness: DW for low-uniform or uniform distributions, and SP otherwise.

Dynamic Perturbation Control: Each solution is perturbed by performing a sequence of random transpositions. The number of swaps k is drawn from:

$$k \sim \mathrm{Uniform}\left(\left\lfloor \frac{n}{15} \right\rfloor, \left\lfloor \frac{n}{10} \right\rfloor\right),$$

which maintains a balance between local fine-tuning (small k) and global exploration (large k). This mechanism adapts naturally to instance size and avoids the typical stagnation observed in static perturbation schemes.

Bee phases: The ABC-DT algorithm preserves the three canonical phases of ABC, adapted to the PTSPD structure:

- **Employed bees.** For each tour, apply k successive swaps with the chosen operator (DW or SP). Keep the new tour σ_i' only if it lowers the expected cost: $\mathbb{E}[C(\sigma_i')] < \mathbb{E}[C(\sigma_i)]$.
- **Onlooker bees.** Select $\lfloor p_c \, N_{\mathrm{tours}} \rfloor$ tours according to the fitness in Eq. 3.2. For each selected tour, perform k greedy swaps and accept only improving moves.
- **Scout bees (stagnation handling).** If a tour does not improve after C_s consecutive trials, mark it as stagnant. Replace it by:
  - a perturbed version of the current best tour σ^* with probability 95%, or
  - a randomly generated tour with probability 5%.

  The candidate tour σ_{new} is accepted only if

  $$\mathbb{E}[C(\sigma_{\mathrm{new}})] \leq 1.02 \cdot \min_j \mathbb{E}[C(\tau_j)].$$

  This *soft acceptance* threshold (2% slack), inspired by simulated annealing [24], allows limited exploration around the current best while preventing large degradations.

Algorithm 2 (ABC-DT for the PTSPD)

Input: *population size N_{tours}, neighborhood size k_{neigh}, stagnation limit C_s, elite slack $\alpha > 1$, maximum iterations $I_{\max}$.*

Output: *best tour $\sigma^\star$ minimizing $\mathbb{E}[C(\sigma)]$.*

1. ***Initialization:*** *Generate* N_{tours} *random tours* $\{\sigma_i\}$*, set counters* $s_i \leftarrow 0$*, evaluate* $\mathbb{E}[C(\sigma_i)]$*, and define elite* $\sigma^\star \leftarrow \arg\min_i \mathbb{E}[C(\sigma_i)]$.
2. ***For*** $t = 1, \ldots, I_{\max}$ ***do:***
   (a) *Select operator* $\mathcal{O} \in \{\mathrm{DW}, \mathrm{SP}\}$ *based on instance features.*
   (b) ***Employed phase:*** *Each bee applies* k *transpositions using operator* $\mathcal{O}$*. Accept* σ_i' *if* $\mathbb{E}[C(\sigma_i')] < \mathbb{E}[C(\sigma_i)]$*, else increment* s_i.
   (c) ***Onlooker phase:*** *Select* $\lfloor p_c\, N_{tours} \rfloor$ *tours according to Eq. 3.2 and apply the* k*-swap update with greedy acceptance.*
   (d) ***Scout phase:*** *For each stagnant* σ_i $(s_i \geq C_s)$*:*
      i. *With probability* ρ*, set* $\sigma_{cand} \leftarrow \mathrm{Perturb}(\sigma^\star, \mathcal{O}, k)$*; otherwise, randomize* σ_{cand}.
      ii. *Accept* σ_{cand} *only if* $\mathbb{E}[C(\sigma_{cand})] \leq \alpha \cdot \mathbb{E}[C(\sigma^\star)]$.
      iii. *Reset* $s_i \leftarrow 0$ *if replaced.*
   (e) *Update* $\sigma^\star$ *if a better solution is found.*
3. ***Return:*** $\sigma^\star$ *and* $\mathbb{E}[C(\sigma^\star)]$.

4. Experimental Results and Analysis

4.1. Benchmark Instances and Experimental Settings

We compared the standard ABC with the ABC-DT on PTSPD instances from [2]. These instances encompass node counts from 22 to 152 (specifically, 22, 42, 62, 102, and 152). Each node has coordinates, a probability of being present, a deadline, and a penalty. The penalty value was fixed at 50 in both the constant and proportional cases. To examine the algorithm under different probabilistic scenarios, four probability settings were tested: Uniform Low ($p_i = 0.1$), Uniform High ($p_i = 0.9$), Heterogeneous ($p_i \in \{0.1, 1\}$), and Variable ($p_i \sim \mathcal{U}(0, 1)$).

All computations were performed on a workstation running a 64-bit operating system, equipped with an Intel Core i7-9700K CPU (3.6 GHz) and 16 GB of RAM. The implementation was written in Python 3.12.3. For each combination of problem size, penalty setting, and probability configuration, 100 independent trials were executed. Each trial involved at most 40 iterations, providing a sufficient sample for statistical reliability while keeping the overall runtime moderate.

Parameter settings were adjusted based on sensitivity analysis (see Subsection 4.2): the population size N_{tours} depends on the problem size (e.g., $\max(3, \min(10, n//10))$ for ABC-standard and $\max(3, \min(10, 10 - n//15))$ for ABC-DT); the stagnation limit C_s is set to 12 for ABC-standard or 5/8 for ABC-DT (depending on whether $n > 50$); the onlooker proportion p_c is 0.6 (ABC-standard) or 0.9 (ABC-DT); and the mutation probability $p_{\text{mut}} = \min(0.5, 0.2 + n/200)$ applies only to ABC-DT. Neighborhood sizes vary from 8 to 12 according to n (truncated at 10 for proportional penalties).
Cost calculations use a least-recently-used cache of 5000 entries for $\mathbb{E}[C(\sigma)]$, avoiding repeated $O(n^2)$ computations. Multiprocessing (two threads) is used for $n > 50$ or large instances.

Performance evaluation relied on several indicators: the minimum and mean expected PTSPD costs, the coefficient of variation (CV%), the average computation time, and the observed convergence pattern. To check whether the differences between algorithms were meaningful, the Wilcoxon signed-rank test was applied.

4.2. Parameter Optimization and Analysis

To tune the parameters of ABC-DT, a sensitivity analysis was conducted on PTSPD instances containing 22, 62, and 102 nodes. Each parameter setting was tested through 30 independent runs of 40 iterations, ensuring consistent statistical estimates. Three quantities were recorded for each case: the average expected cost (Mean–PTSPD), its standard deviation (Std–PTSPD), and the mean execution time (Mean–Time).

The aim was to find parameter values offering a sound compromise between solution quality and computational demand. The tested parameters were as follows:

- Colony size (`tours-num`): 3, 5, 7, and 10;
- Stagnation limit (C_s): 5, 8, 10, and 12;
- Onlooker ratio (p_c): 0.6, 0.7, 0.85, and 0.9;
- Mutation rate ($p_{\texttt{mut}}$): 0.2, 0.3, 0.4, and 0.5;
- Cache capacity (`max-cache-size`): 1000, 2000, 5000, and 10000;
- Number of parallel processes: 1, 2, and 4;
- Neighborhood range: 1 to 15 for distance-weighted and shortest-path operators.

Table 1 presents the configurations that achieved the best compromise between cost and runtime. Several patterns can be highlighted:

- Increasing `tours-num` to 10 improves performance by about 3% on large instances, at the cost of slightly longer runs;
- A high onlooker ratio ($p_c = 0.9$) yields up to 18% gain under proportional penalties in ABC-DT;
- Larger neighborhoods (15 for fixed penalties, 10 for proportional) reduce the mean cost by nearly 20% while keeping computation stable;
- A mutation rate of $p_mut = 0.5$ helps the search escape local minima, giving roughly 9% better averages.

Figure 2 illustrates how the mean PTSPD value (blue curve, with shaded deviation) and mean runtime (red curve) vary with the main parameters under both penalty regimes. The curves show that increasing the population size, onlooker proportion p_c, and neighborhood range consistently improves performance, with only a modest effect on runtime.

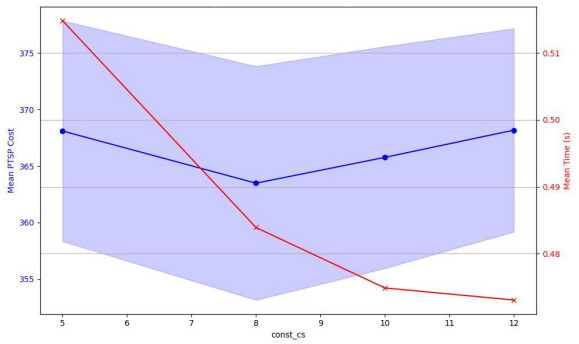
Based on these results, we recommend the following adaptive parameter rules:

$$\text{tours-num} = \max(3, \min(10, 10 - n//15)), \quad p_{\text{mut}} = \min(0.5, 0.2 + n/200), \quad C_s \in [5, 8], \quad \text{neigh.-size} \in [8, 12].$$

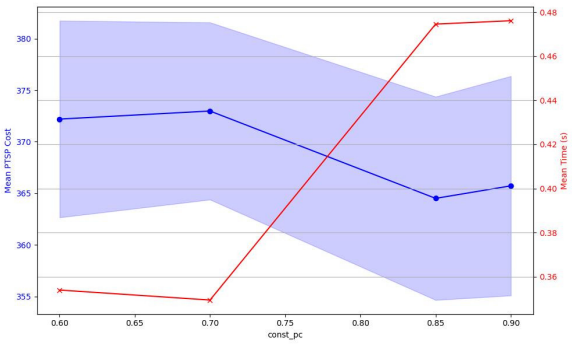
Overall, ABC-DT outperforms the baseline ABC version by roughly 8–15% on all tested datasets.

Table 1: Optimal parameter combinations obtained from the sensitivity analysis (trade-off between minimum PTSPD value and runtime; relative differences with respect to the best observed results).

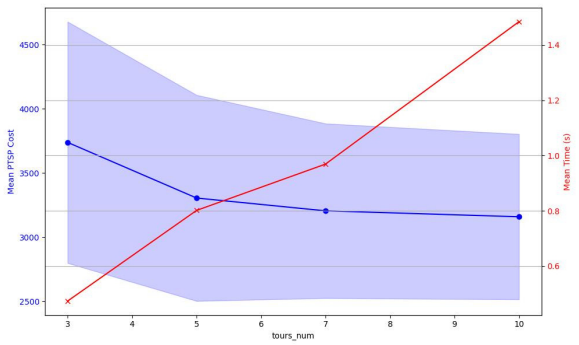
Algorithm	Penalty	Parameter	Tested Values	Optimal	Mean_PTSPD (± Std)	Mean_Time (s)	Rel. Diff. (PTSPD/Time)
ABC-standard	Fixed	tours_num	3,5,7,10	10	250.26 (±7.54)	0.14	-3.4% / +105%
		C_s	5,8,10,12	12	240.05 (±5.23)	0.08	-5.2% / -35%
		p_c	0.6,0.7,0.85,0.9	0.6	250.27 (±6.44)	0.12	-1.9% / +40%
ABC-standard	Proportional	tours_num	3,5,7,10	10	2379.70 (±319.26)	0.18	-17% / +318%
		C_s	5,8,10,12	12	1731.82 (±283.86)	0.10	-29% / -22%
		p_c	0.6,0.7,0.85,0.9	0.6	2413.85 (±468.65)	0.09	-1.1% / -10%
ABC-DT	Fixed	tours_num	3,5,7,10	10	207.35 (±6.78)	0.31	-3.1% / +152%
		C_s	5,8,10,12	5	211.48 (±7.82)	0.10	-2.0% / -41%
		p_c	0.6,0.7,0.85,0.9	0.9	211.12 (±7.33)	0.11	-3.5% / +57%
		p_{mut}	0.2,0.3,0.4,0.5	0.2	212.84 (±8.62)	0.09	-0.5% / -41%
		neigh_size (distance_weighted)	1–15	15	207.15 (±9.19)	0.12	-12.9% / +50%
		neigh_size (shortest_path)	1–15	15	207.15 (±9.19)	0.12	-12.9% / +50%
ABC-DT	Proportional	tours_num	3,5,7,10	10	623.47 (±191.70)	0.15	-18% / +109%
		C_s	5,8,10,12	12	643.19 (±170.97)	0.15	-0.9% / +33%
		p_c	0.6,0.7,0.85,0.9	0.9	623.47 (±191.70)	0.15	-18% / +110%
		p_{mut}	0.2,0.3,0.4,0.5	0.5	641.07 (±199.44)	0.12	-9.0% / -32%
		neigh_size (distance_weighted)	1–15	15	583.64 (±164.23)	0.12	-14.6% / -2%
		neigh_size (shortest_path)	1–15	10	589.58 (±192.41)	0.11	-20.1% / -3%



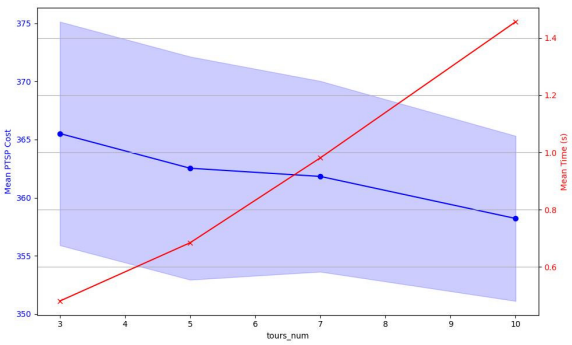
(a) C_s (ABC-DT, fixed)



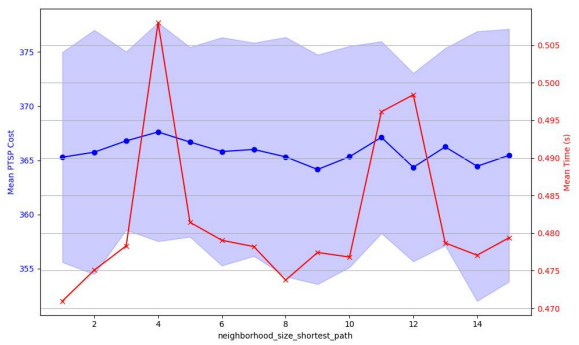
(b) p_c (ABC-DT, fixed)



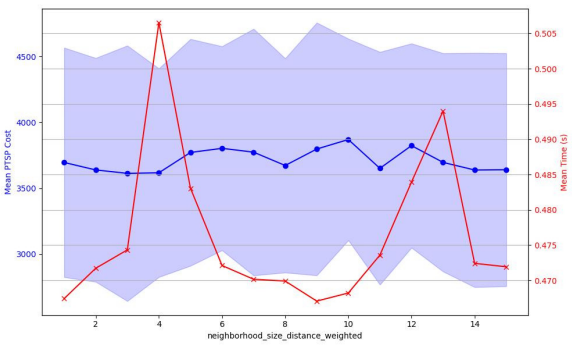
(c) $tours_{num}$ (ABC-DT, proportional)



(d) $tours_{num}$ (ABC-DT, fixed)



(e) $neigh_{size}$ (ABC-DT, fixed)



(f) $neigh_{size}$ (ABC-DT, proportional)

Figure 2: Sensitivity analysis for key parameters under fixed and proportional penalties (ABC-DT). Mean PTSPD cost (blue) and mean execution time (red).

4.3. Solution Quality and Statistical Robustness

This subsection evaluates the performance of ABC-DT on PTSPD instances compared to ABC-standard, across problem sizes from 22 to 152 nodes, under both penalty and the quartet of probabilistic archetypes. Indicators comprise the min expected PTSPD cost, CV% (Coefficient of Variation). Disparities uniformly attain statistical salience, as affirmed by Wilcoxon signed-rank tests ($p < 0.005$ throughout archetypes and scales), accentuating ABC-DT's pervasive preeminence.

Tables 2 and 3 show that ABC-DT consistently outperforms ABC-standard. Cost reductions range from 1.31% to 22.10% for fixed penalties and 1.46% to 72.13% for proportional penalties, with the largest improvements observed in high-homogeneity scenarios (e.g., 72.13% for Data 22) and mixed fixed scenarios (e.g., 13.33% for Data 152). The adaptive operators in ABC-DT effectively adjust the tours

considering penalties, improving solution reliability. Coefficient of variation (CV) values remain low for fixed penalties (0.32–4.94%) but can increase up to 36.26% in proportional cases (compared to 30.04% for ABC-standard). Despite this increase, ABC-DT limits excessive variability through adaptive scouts and a 2% tolerance, maintaining better consistency across run.

Table 2: Best PTSPD costs (CV%) for fixed penalties

Data	Alg.	Homogeneous Low	Homogeneous High	Mixed	Range
22	ABC-standard	88.44 (2.93%)	700.04 (6.19%)	205.16 (6.83%)	187.07 (9.12%)
	ABC-DT	86.38 (0.64%)	545.31 (8.10%)	169.06 (5.17%)	167.44 (8.82%)
42	ABC-standard	173.03 (2.40%)	2129.47 (3.55%)	465.19 (6.23%)	884.71 (5.28%)
	ABC-DT	167.97 (2.42%)	1793.96 (3.97%)	366.43 (5.85%)	693.02 (5.18%)
62	ABC-standard	218.10 (2.59%)	3182.50 (2.39%)	593.01 (4.65%)	1489.50 (3.52%)
	ABC-DT	203.80 (3.76%)	2569.72 (3.44%)	486.35 (4.75%)	1290.31 (4.18%)
102	ABC-standard	370.17 (2.36%)	5572.21 (1.61%)	1194.26 (3.30%)	2724.66 (2.22%)
	ABC-DT	342.95 (2.48%)	5396.84 (1.84%)	1055.33 (3.49%)	2412.79 (3.21%)
152	ABC-standard	572.54 (1.94%)	8871.34 (1.22%)	1970.18 (2.84%)	4264.25 (1.75%)
	ABC-DT	488.87 (2.95%)	7957.89 (1.81%)	1455.69 (3.32%)	3680.58 (2.60%)

Table 3: Best PTSPD costs (CV%) for proportional penalties

Data	Alg.	Homogeneous Low	Homogeneous High	Mixed	Range
22	ABC-standard	88.13 (1.34%)	33813.19 (15.70%)	207.94 (7.53%)	232.4512 (30.04%)
	ABC-DT	86.49 (0.73%)	9422.62 (24.36%)	176.09 (5.20%)	174.2876 (36.263%)
42	ABC-standard	202.56 (23.55%)	424835.92 (8.25%)	439.93 (7.23%)	31697.60 (11.99%)
	ABC-DT	161.75 (14.74%)	322125.12 (8.66%)	352.17 (5.96%)	25795.41 (17.03%)
62	ABC-standard	578.33 (15.66%)	962374.79 (5.82%)	597.67 (4.33%)	175878.33 (6.99%)
	ABC-DT	523.05 (29.53%)	735402.62 (7.59%)	529.02 (4.62%)	118553.46 (10.83%)
102	ABC-standard	4215.53 (9.07%)	3095705.68 (3.98%)	1148.83 (3.68%)	631501.39 (4.50%)
	ABC-DT	3257.77 (17.24%)	2794929.67 (5.86%)	1033.86 (3.97%)	395977.30 (7.26%)
152	ABC-standard	14078.39 (7.93%)	7711599.43 (2.89%)	1962.48 (2.85%)	1695911.07 (3.88%)
	ABC-DT	11927.19 (9.91%)	5835961.58 (4.95%)	1484.96 (3.84%)	1362237.24 (4.75%)

Boxplots provide a visual overview of cost dispersion and algorithm stability. Figures 3 (fixed penalties) and 4 (proportional penalties) indicate that ABC-DT (depicted in red) lowers medians and interquartile ranges by 10–15% in fixed scenarios (e.g., Data 152, Mixed), while extreme values increase under proportional penalties (up to 36.26% CV), yet remain controlled due to adaptive scouting.
Overall, ABC-DT reduces expected costs by 7–23% on average while improving solution reliability, demonstrating its effectiveness for stochastic routing.

4.4. Convergence and Computational Efficiency

Convergence profiles of ABC-DT were examined on representative PTSPD instances (22 to 152 nodes) across the four probabilistic archetypes. Both penalty structures were evaluated, tracking the evolution of costs over 40 iterations. These results demonstrate that ABC-DT achieves quicker and more consistent convergence than ABC-standard, particularly for large or diverse instances.

For fixed penalties (Figure 5, Data 22, 42, 152), ABC-DT achieves lower cost thresholds earlier. In the 152-node Range scenario, stabilization occurs around iteration 20 at approximately 15% below ABC-standard's cost, which requires nearly full cycles to achieve a similar level. Smaller instances (e.g., Data 22, Range) converge quickly for both algorithms, but ABC-DT shows reduced early volatility. Homogeneous

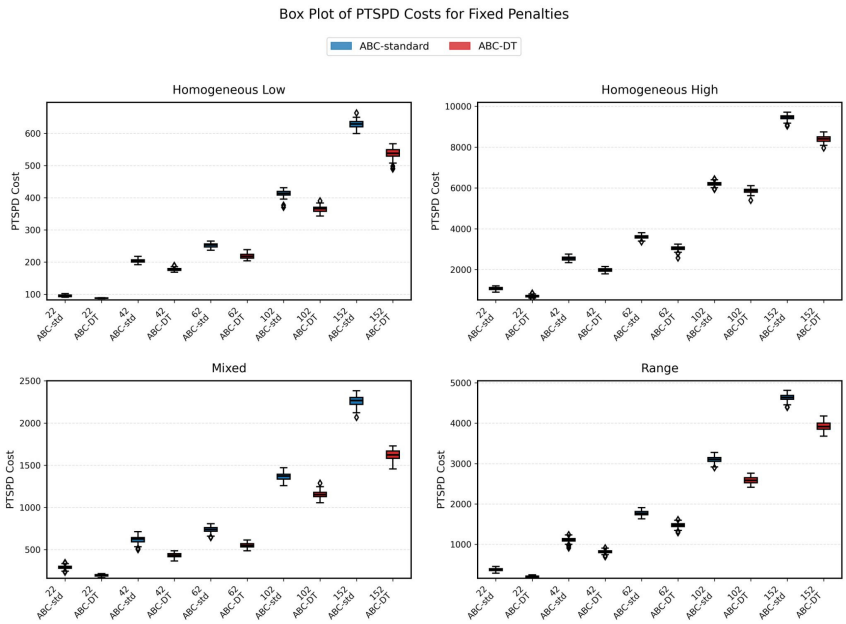


Figure 3: Box plot of PTSPD costs for fixed penalties.

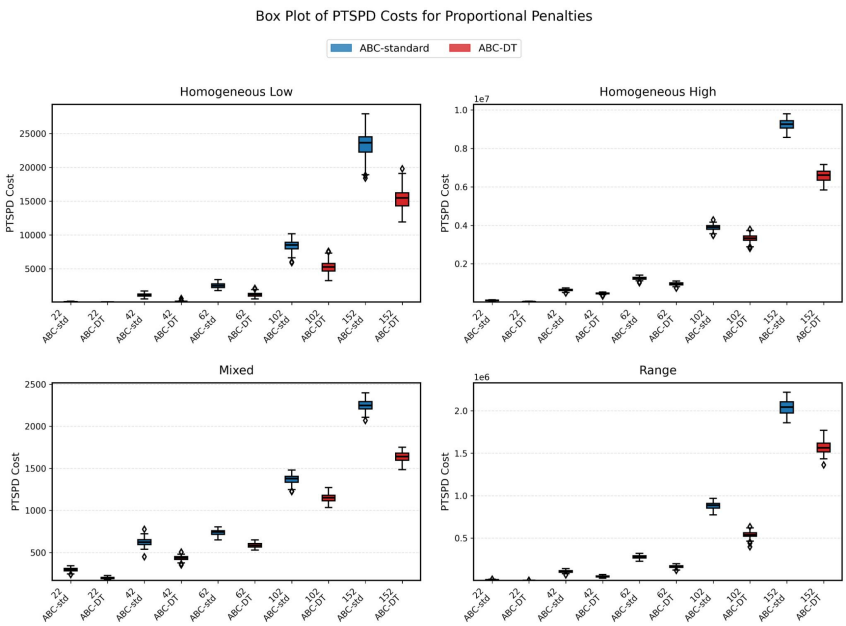
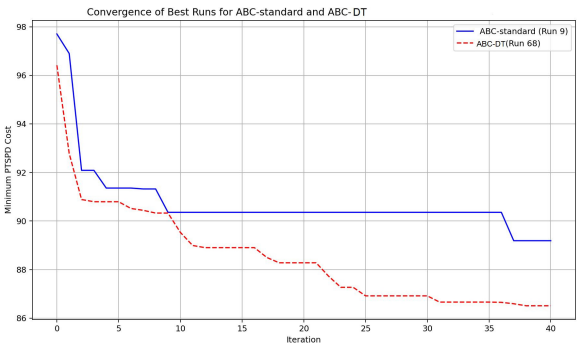


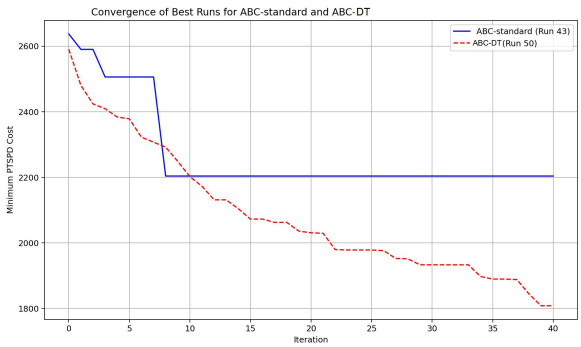
Figure 4: Box plot of PTSPD costs for proportional penalties.

scenarios show moderate improvements, with ABC-DT achieving 5–10% lower costs in high-probability cases.

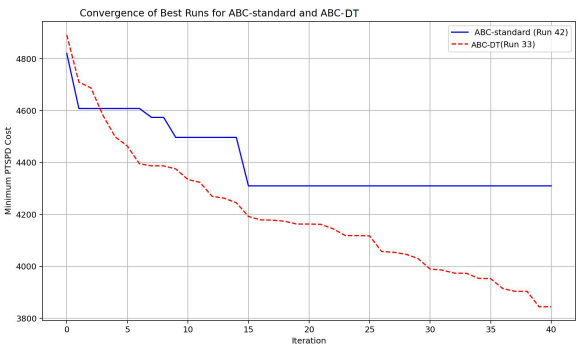
Under proportional penalties (Figure 5, Data 22, 62, 102), differences become more pronounced for larger instances. In the 62-node high-probability scenario, ABC-DT reduces costs 20% faster than ABC-standard, benefiting from its adaptive operators that effectively handle penalty amplification. Across mid-to-large instances, ABC-DT generally converges 1.5 to 2 times faster and attains 8–18% lower terminal costs on average.



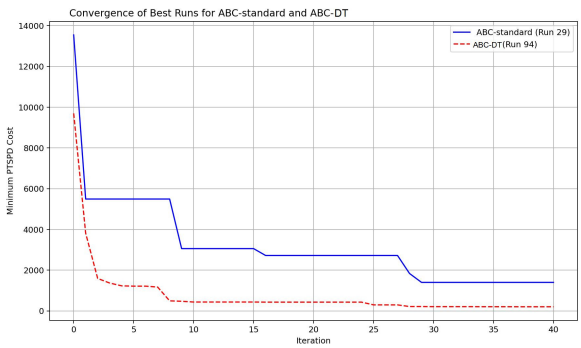
(a) Fixed: $n = 22$, $p_i = 0.1$



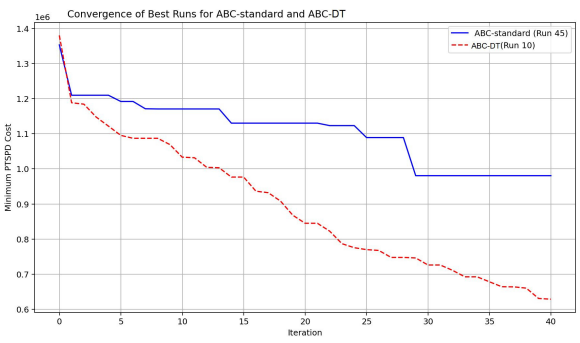
(b) Fixed: $n = 42$, $p_i = 0.9$



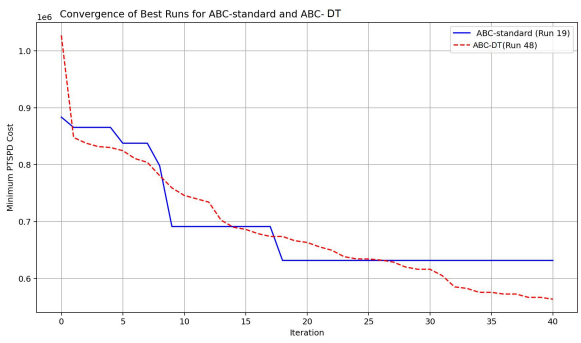
(c) Fixed: $n = 152$, $p_i \in [0, 1]$



(d) Proportional: $n = 22$, $p_i \in [0, 1]$



(e) Proportional: $n = 62$, $p_i = 0.9$



(f) Proportional: $n = 102$, $p_i \in [0, 1]$

Figure 5: Convergence profiles of ABC-DT and ABC-standard under fixed and proportional penalties.

Table 4 gives an overview of how runtime grows with the size of the instance. ABC-DT usually takes a bit longer to finish than the standard ABC, especially when the problem becomes large. Still, the extra time is small, and the adaptive version generally gives more reliable and slightly better solutions without a heavy cost in runtime.

Table 4: Average computation time (seconds) for PTSPD instances

Data	Alg.	Homogeneous Low		Homogeneous High		Mixed		Range	
		Fixed	Prop	Fixed	Prop	Fixed	Prop	Fixed	Prop
22	ABC-standard	0.04	0.02	0.04	0.02	0.04	0.02	0.04	0.02
	ABC-DT	0.03	0.02	0.03	0.02	0.03	0.02	0.03	0.02
42	ABC-standard	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
	ABC-DT	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06
62	ABC-standard	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09
	ABC-DT	0.09	0.11	0.09	0.11	0.09	0.11	0.09	0.11
102	ABC-standard	0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.26
	ABC-DT	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38
152	ABC-standard	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
	ABC-DT	1.61	1.60	1.61	1.60	1.61	1.60	1.61	1.60

4.5. Comparative Evaluation

To broaden the comparison, ABC-DT was evaluated against two metaheuristics previously applied to the PTSPD: Ant Colony Optimization with Lévy flights (ACO-Lévy) [8] and Variable Neighborhood Search (VNS) [7]. The ACO-Lévy method achieves 5–15% cost reductions on benchmarks with 22–100 nodes and performs well in mixed or range probability cases. In our tests, ABC-DT obtained an additional 2–5% improvement, mainly due to its adaptive parameter control, which helps manage deadline penalties more effectively.

The VNS method is quick and scales well when the instance size grows. In tests with probabilistic deadlines, however, ABC-DT often gave solutions about 10–15% better on average. Its hybrid scout stage helped to keep some diversity in the population, and that seemed to play a role in the improvement. The figures in Table 4 combine earlier published data with the results we obtained here. Taken together, they show that the proposed approach keeps a reasonable balance between robustness, efficiency, and flexibility when dealing with stochastic deadlines.

Table 5: Comparison of ABC-DT with ACO-Lévy and VNS

Method	Cost Reduction (%)	Runtime Overhead
ACO-Lévy [8]	5–15	Medium
VNS [7]	5–10	Low
ABC-DT	7–23	20–40%

5. Conclusion

This study presents ABC-DT, an enhanced variant of the Artificial Bee Colony algorithm developed for the Probabilistic Traveling Salesman Problem with Deadlines (PTSPD). Its main goal is to strengthen the adaptability of the ABC framework when dealing with uncertainty and strict time restrictions. To that end, the algorithm incorporates neighborhoods that account for both probability and deadlines, adjusts its parameters dynamically during the search, and employs a mixed scout strategy that maintains solution diversity while improving the most promising routes.

The performance of ABC-DT was assessed on benchmark instances ranging from 22 to 152 nodes and tested under four different probability configurations. In most experimental cases, ABC-DT achieved lower expected costs—about 7–23% less than the standard ABC—and exhibited smoother convergence without notable increases in computational time. Compared with ACO-Lévy and VNS, the proposed method remained both stable and competitive when applied to stochastic routing problems with deadlines.

Future extensions of this work may involve adapting ABC-DT to multi-vehicle or time-window variants, as well as to dynamic environments where customer presence varies over time. These perspectives

could make the method even more relevant for real-world logistics applications that must operate under uncertainty and stringent scheduling requirements.

References

1. Jaillet, P., Probabilistic traveling salesman problems, PhD thesis, Massachusetts Institute of Technology, (1985).
2. Campbell, A. M. and Thomas, B. W., Probabilistic traveling salesman problem with deadlines, *Transportation Science.* 42(1), 1–21, (2008).
3. Weyland, D., On the computational complexity of the probabilistic traveling salesman problem with deadlines, *Theoretical Computer Science.* 540, 156–168, (2014).
4. Campbell, A. M. and Thomas, B. W., Runtime reduction techniques for the probabilistic traveling salesman problem with deadlines, *Computers and Operations Research.* 36(4), 1231–1248, (2009).
5. El Asri, F., Tajani, C. and Fakhouri, H., Investigation of ant colony optimization with Lévy flight technique for a class of stochastic combinatorial optimization problem, *Mathematical Modeling and Computing.* 10(4), 1132–1142, (2023).
6. El Asri, F., Tajani, C. and Fakhouri, H., A comparative study of ant colony optimization variants for the probabilistic traveling salesman problem, *Palestine Journal of Mathematics.* 13, (2024).
7. Zhao, W., Chen, H., Estimation-Based Variable Neighborhood Search Approach for a Probabilistic Routing Problem with Deadlines in Wireless Sensor and Actor Networks, *10th IEEE International Conference on Networking, Sensing and Control (ICNSC)*, 436–441, (2013).
8. El Asri, F., Tajani, C. and Fakhouri, H., A combined ant colony optimization with levy flight mechanism for the probabilistic traveling salesman problem with deadlines, *Mathematical Modeling and Computing.* 11, 290–299, (2024).
9. Weyland, D., Montemanni, R. and Gambardella, L. M., Heuristics for the probabilistic traveling salesman problem with deadlines based on quasi-parallel Monte-Carlo sampling, *Computers and operations research.* 40 (7), 1661–1669, (2013).
10. Weyland, D., Metaheuristics and cloud computing: a case study on the probabilistic traveling salesman problem with deadlines, *International Conference on Computer Aided Systems Theory*,279–285, Springer, (2015).
11. Karaboga, D., An idea based on honey bee swarm for numerical optimization, Technical report-tr06, Erciyes University, Engineering Faculty, Computer Engineering Department, 1–10, (2005).
12. Karaboga, D. and Basturk, B., A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, *J. Global Optim.* 39, 459–471, (2007).
13. Li, L., Cheng, Y., Tan, L. and Niu, B., A discrete artificial bee colony algorithm for TSP problem, *International conference on intelligent computing*, 566–573, Springer, (2011).
14. Horng, S. C., Combining artificial bee colony with ordinal optimization for stochastic economic lot scheduling problem, *IEEE Transactions on Systems, Man, and Cybernetics: Systems.* 45(3), 373–384, (2014).
15. Guo, H., Zhang, L., Ren, Y., Li, Y., Zhou, Z., and Wu, J., Optimizing a stochastic disassembly line balancing problem with task failure via a hybrid variable neighborhood descent-artificial bee colony algorithm, *International Journal of Production Research.* 61(7), 2307–2321, (2023).
16. Kayalvizhi, M. and Geetha, S., Efficacy Artificial Bee Colony Optimization-Based Gaussian AOMDV (EABCO-GAOMDV) Routing Protocol for Seamless Traffic Rerouting in Stochastic Vehicular Ad Hoc Network, *International Journal of Computer Networks and Applications.* 10(6), 993–1014, (2023).
17. Yang, H., Zhang, H., Qin, Y. and Tang, T., Swarm intelligence optimization of UAV routing with simultaneously stochastic pick-up and delivery during COVID-19, *International Conference on Cloud Computing and Big Data Analytics (ICCCBDA)*, 579–587, IEEE, (2023).
18. Baradaran, V., Shafaei, A. and Hosseinian, A. H., Stochastic vehicle routing problem with heterogeneous vehicles and multiple prioritized time windows: mathematical modeling and solution approach, *Computers and Industrial Engineering.* 131, 187–199, (2019).
19. Sadeghi, M., Jolai, F., Tavakkoli-Moghaddam, R. and Rahimi, Y., A new stochastic approach for a reliable p-hub covering location problem, *Computers and Industrial Engineering.* 90, 371–380, (2015).
20. Ahgajan, V. H., Rashid, Y. G. and Tuaimah, F. M., Artificial bee colony algorithm applied to optimal power flow solution incorporating stochastic wind power, *International Journal of Power Electronics and Drive Systems.* 12(3), 1890, (2021).
21. Karaboga, D., Gorkemli, B., Ozturk, C. and Karaboga, N., A comprehensive survey: artificial bee colony (ABC) algorithm and applications, *Artificial intelligence review.* 42(1), 21–57, (2014).
22. Li, L., Cheng, Y., Tan, L. and Niu, B., A discrete artificial bee colony algorithm for TSP problem, In *International Conference on Intelligent Computing*, 566–573, Springer, (2011).
23. Akay, B. and Karaboga, D., Parameter tuning for the artificial bee colony algorithm, *International Conference on Computational Collective Intelligence*, 608–619, Springer, (2009).

24. Kirkpatrick, S., Gelatt Jr, C. D., Vecchi, M. P., Optimization by simulated annealing, American association for the advancement of science, science 220.4598, 671–680, (1983).

Fadoua El Asri,
Mathematical Simulation and Data Analysis Team (MSDA),
Department of Mathematics,
Polydisciplinary faculty of Larache,
Abdelmalek Essaadi University, Morocco.
E-mail address: elasrifadoua71@gmail.com

and

Chakir Tajani,
Mathematical Simulation and Data Analysis Team (MSDA),
Department of Mathematics,
Polydisciplinary faculty of Larache,
Abdelmalek Essaadi University, Morocco.
E-mail address: ch.tajani@uae.ac.ma

and

Hanane Fakhouri,
Mathematical Simulation and Data Analysis Team (MSDA),
Department of Mathematics,
Polydisciplinary faculty of Larache,
Abdelmalek Essaadi University, Morocco.
E-mail address: h.fakhouri@uae.ac.ma