algorithms

MDPI

Article

Measuring the Performance of Ant Colony Optimization Algorithms for the Dynamic Traveling Salesman Problem

Michalis Mavrovouniotis [1,2,*], Maria N. Anastasiadou [1] and Diofantos Hadjimitsis [1,2]

[1] ERATOSTHENES Centre of Excellence, 3012 Limassol, Cyprus; maria.anastasiadou@eratosthenes.org.cy (M.A.); d.hadjimitsis@cut.ac.cy (D.H.)
[2] Department of Civil Engineering and Geomatics, Cyprus University of Technology, 3036 Limassol, Cyprus
* Correspondence: michalis.mavrovouniotis@eratosthenes.org.cy

Abstract: Ant colony optimization (ACO) has proven its adaptation capabilities on optimization problems with dynamic environments. In this work, the dynamic traveling salesman problem (DTSP) is used as the base problem to generate dynamic test cases. Two types of dynamic changes for the DTSP are considered: (1) node changes and (2) weight changes. In the experiments, ACO algorithms are systematically compared in different DTSP test cases. Statistical tests are performed using the arithmetic mean and standard deviation of ACO algorithms, which is the standard method of comparing ACO algorithms. To complement the comparisons, the quantiles of the distribution are also used to measure the peak-, average-, and bad-case performance of ACO algorithms. The experimental results demonstrate some advantages of using quantiles for evaluating the performance of ACO algorithms in some DTSP test cases.

Keywords: ant colony optimization; dynamic optimization; traveling salesman problem

Citation: Mavrovouniotis, M.; Anastasiadou, M.N.; Hadjimitsis, D. Measuring the Performance of Ant Colony Optimization Algorithms for the Dynamic Traveling Salesman Problem. *Algorithms* **2023**, *16*, 545. https://doi.org/10.3390/a16120545

Academic Editors: Nikola Ivković and Matej Črepinšek

Received: 24 October 2023
Revised: 23 November 2023
Accepted: 25 November 2023
Published: 28 November 2023

1. Introduction

Ant colony optimization (ACO) is a mainstream swarm intelligence algorithm inspired by the foraging behavior of real ant colonies [1–3]. In ACO, a colony of artificial ants constructs solutions guided by artificial pheromone trails and some heuristic information. ACO has been successfully applied in various complex optimization problems [4–6]. However, in most real-world applications, the optimization problem has a dynamic environment [7]. In particular, the environment of the optimization problem, including objective function, constraints, search space, input variables, and so on, may change over time.

Major challenges occur when addressing dynamic optimization problems because repeated optimization of the changing optimum is required. The optimal solution in which the algorithm has converged to it before the dynamic change may not be the optimal solution after the dynamic change. Consequently, the optimization process needs to be restarted in order to search for a new optimal solution. However, it may not be very efficient to restart the optimization process from scratch in complex problems (e.g., $\mathcal{NP}$-hard combinatorial problems), as they require huge computational efforts and time [8].

A more efficient way to address dynamic optimization problems is to use knowledge from previously optimized environments to speed up the re-optimization process [9]. With ACO algorithms, this knowledge can be transferred via the pheromone trails generated in the previous environments. However, the pheromone trails of the previous environment may contain information directing the search process to the previous optimal solution, which may represent a poor local optimum in the current environment [10]. Several strategies have been integrated to enhance the adaptation capabilities of ACO in dynamic optimization problems, such as the generation of new ants using immigrant schemes [11,12], population-based pheromone update strategies [13], multiple ant colonies [14], and memetic computing [15].

In this work, the traveling salesman problem (TSP) is used as the base optimization problem to generate dynamic test cases using the benchmark generator proposed in [10]. A conventional ACO [16] and a population-based ACO [13] that use pheromone evaporation and memory archive, respectively, to adapt to dynamic changes are applied to the generated dynamic TSP (DTSP) test cases.

The most common method to evaluate the performance of ACO in dynamic environments is used in the experiments, that is, to obtain the mean and standard deviation (for multiple independent executions) for the generated DTSPs. Statistical tests are performed to compare the algorithms. In [17], it is argued that, for stochastic algorithms (e.g., ACO algorithms), this way of comparison may not be very informative. This is because the distribution of executing ACO algorithms multiple times may be asymmetrical and, hence, outlier values may affect the mean value. Consequently, inadequate information regarding the average performance of the algorithms may be provided. In contrast, quantiles of the distribution may address this issue with the use of median values as an average performance measurement.

The rest of the paper is structured as follows. Section 2 provides a description of the TSP as well as a description of the generation of dynamic test cases. Section 3 describes the application of the ACO metaheuristic to the TSP. In addition, the pheromone update policies in dynamic environments for the two ACO algorithms used in the experiments are described. Section 4 presents the experimental setup and results. An analysis of the obtained results is provided. Section 5 concludes this work.

2. The Dynamic Traveling Salesman Problem

2.1. Problem Definition

The TSP is a well-known combinatorial optimization problem proven to be $\mathcal{NP}$-hard [18]. The problem can be described as follows: a salesperson must visit multiple cities once and only once, starting from one city and returning finally, to the same city [19]. Typically, a fully connected weighted graph $G = (N, A)$ is used to model the TSP, where $N = \{1, \ldots, n\}$ is the set of n cities and $A = \{(i, j) \in N : i \neq j\}$ is a set of arcs connecting these cities. Each arc (i, j) is associated with a non-negative value d_{ij}, which is the distance between cities i and j.

The TSP solution $s = [1, \ldots, n]$ is a permutation of city indices, and the objective function of the TSP is defined as follows:

$$\phi(s) = \sum_{i=1}^{n-1} d_{s(i)s(i+1)} + d_{s(n)s(1)}. \tag{1}$$

2.2. Generating Dynamic Environments

There are two main dynamic versions of the TSP in which: (a) the values associated with the arcs are changing [12,20–22]; and (b) the cities to be visited are changing [11,13,23]. The dynamic benchmark generator (Available at https://github.com/Mavrovouniotis/ACODTSP (accessed on 23 February 2023)) used in [10] is adopted to generate DTSP test cases. Each DTSP test case is characterized by the type of change (either the weights or the nodes change), the magnitude of change (either small, medium, or severe), and the frequency of change (either fast or slow).

For the DTSP with weight changes, a factor that increases or decreases the value of arcs connecting cities i and j is assigned every environmental period T, i.e., $d_{ij}(T)$, where $T = \lceil t/f \rceil$, f is the frequency of a dynamic change, and t is the algorithmic iteration count. The value of the arc (i, j) changes as follows:

$$d_{ij}(T+1) = \begin{cases} d_{ij}(0) + \mathcal{R}_{ij}, & \text{if arc}(i,j) \in A_{rnd}(T), \\ d_{ij}(T), & \text{otherwise}, \end{cases} \tag{2}$$

where $\mathcal{R}_{ij}$ is a random number drawn from a normal distribution with a zero mean and

standard deviation proportional to the initial value of the arc $d_{ij}(0)$, $A_{rnd}(T) \subset A$ is a set of randomly selected arcs with size $\lceil mn(n-1) \rceil$ in which their values are subject to change at environmental period index T, and $m \in (0,1)$ is the magnitude of change. Therefore, the quality value of solution s for the DTSP depends on the parameter t and, thus, is evaluated as $\phi(s,t)$. In other words, the quality of a solution s at time t may be different at time $t+1$.

For the DTSP with node changes, a newly generated set of cities, i.e., $N_{rnd}(T)$, is initialized, with n random cities in the range of the actual set of cities N. A dynamic change is generated for every f function evaluation, in which exactly $\lceil mn \rceil$ cities are randomly chosen from $N_{rnd}(T)$ to replace exactly $\lceil mn \rceil$ randomly chosen cities from N, where f and m define the frequency and magnitude of a dynamic change, respectively, as above.

3. Addressing the Dynamic Traveling Salesman Problem

3.1. Ant Colony Optimization Metaheuristic

ACO has been designed for graph problems and, thus, it can be applied directly to the TSP, which is modeled as a weighted graph [24–26]. All arcs are associated with a pheromone trail value τ_{ij} and some heuristic information value η_{ij}. Initially, the pheromone trails are assigned with equal values, i.e., $\tau_{ij} \leftarrow \tau_0, \forall \in A$, where τ_0 is the initial pheromone value. The heuristic information at environment T is defined as $\eta_{ij} = 1/d_{ij}(T)$. A colony of ω artificial ants is initially positioned in different randomly selected cities. Each ant k will select the next city to visit based on the existing pheromone trails τ_{ij} and the defined heuristic information η_{ij} of arc (i,j). Each ant will visit all cities once, in order to represent a feasible DTSP solution. Then, the pheromone trails associated with the ω solutions constructed by the ants will be reinforced according to their quality value.

3.1.1. Constructing Solutions

With a probability $(1-q_0)$, where $0 \leq q_0 \leq 1$ is a parameter of the decision rule, the k-th chooses the next city j from city i, from a probability distribution that is defined as follows:

$$p_{ij}^k = \begin{cases} \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in \mathcal{N}_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta}, & \text{if } j \in \mathcal{N}_i^k, \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where τ_{ij} and η_{ij} are the existing pheromone trail and heuristic information values, respectively, $\mathcal{N}_i^k$ is the set of cities that ant k has not visited yet, and α and β are the two parameters that determine the relative influence of the pheromone trail and heuristic information, respectively. With probability q_0, ant k chooses the next city, i.e., j, with the maximum probability as follows [27]:

$$j = \underset{j \in \mathcal{N}_i^k}{\operatorname{argmax}} \left[\tau_{ij}\right]^\alpha \left[\eta_{ij}\right]^\beta. \tag{4}$$

This selection process will continue until each ant has visited all cities once and only once, as shown in Algorithm 1. The constructed solutions will be evaluated based on Equation (3).

Algorithm 1 Construct Solutions (t)

```
for each ant k do
    s^k ← ∅
    r ← random{1, ..., n}
    s^k ← s^k ∪ r
    while s^k length not equal with n do
        j ← select next city ∈ N_i^k
        s^k ← s^k ∪ j
    end while
    C^k ← φ(s^k, t)
end for
```

3.1.2. Updating Pheromone Trails

The pheromone update procedure consists of two parts: (1) pheromone evaporation, in which all pheromone trails are reduced by a constant rate; and (2) pheromone deposit, in which ants increase the pheromone trails associated with their constructed solutions.

In this work, the pheromone policy of the $\mathcal{MAX} - \mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) algorithm is used, which is one of the best-performing ACO variants [28,29]. Firstly, in the pheromone update procedure of $\mathcal{MM}$AS, an evaporation is performed on all pheromone trails as follows:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij}, \forall\ (i,j) \in A, \tag{5}$$

where $0 < \rho \leq 1$ is the rate of evaporation.

Thereafter, the best ant is allowed to deposit pheromone as follows:

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau_{ij}^{best}, \forall\ (i,j) \in s^{best}, \tag{6}$$

where s^{best} is the solution generated by the best ant and $\Delta\tau_{ij}^{best} = 1/C^{best}$, where $C^{best} = \phi(s^{best}, t)$ is the quality value of DTSP solution s^{best}. The best ant to deposit pheromone may be either the best-so-far ant, in which case $\Delta\tau_{ij}^{best} = 1/C^{bs}$, where $C^{bs} = \phi(s^{bs}, t)$ is the solution quality of the best-so-far ant, or the iteration-best ant, in which case $\Delta\tau_{ij}^{best} = 1/C^{ib}$, where $C^{ib} = \phi(s^{ib}, t)$ is the solution quality of the best ant of the iteration. By default, the iteration-best ant is used to update the pheromone trails and, occasionally, the best-so-far ant. The pheromone trail values in $\mathcal{MM}$AS are kept to the interval $[\tau_{min}, \tau_{max}]$, where τ_{min} and τ_{max} are the minimum and maximum pheromone trails limits, respectively. The overall $\mathcal{MM}$AS pheromone update is presented in Algorithm 2, lines 1–20.

Algorithm 2 Pheromone Update (t, s^{ib}, s^{bs})

1: **if** $\mathcal{MM}$AS is selected **then**
2: $\tau_{max} \leftarrow 1/\rho C^{bs}$
3: $\tau_{min} \leftarrow \tau_{max}\left(1 - \sqrt[n]{0.05}\right) / \left((cand-1) \cdot \sqrt[n]{0.05}\right)$
4: **for** $\forall (i,j) \in A$ **do**
5: $\tau_{ij} \leftarrow (1-\rho)\tau_{ij}$
6: **end for**
7: $s^{best} \leftarrow s^{ib}\ ||\ s^{bs}$
8: $\Delta\tau_{ij}^{best} \leftarrow 1/C^{ib}\ ||\ 1/C^{bs}$
9: **for** each arc$(i,j) \in s^{best}$ **do**
10: $\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau_{ij}^{best}$
11: **end for**
12: **for** $\forall (i,j) \in A$ **do**
13: **if** $\tau_{ij} > \tau_{max}$ **then**
14: $\tau_{ij} \leftarrow \tau_{max}$
15: **end if**
16: **if** $\tau_{ij} < \tau_{min}$ **then**
17: $\tau_{ij} \leftarrow \tau_{min}$
18: **end if**
19: **end for**
20: **end if**
21: **if** P-ACO is selected **then**
22: $\Delta\tau \leftarrow (\tau_{max} - \tau_0)/K$
23: **if** $t == K$ **then**
24: $pop(t) \leftarrow pop(t) \setminus s^r$
25: **for** each arc$(i,j) \in s^r$ **do**
26: $\tau_{ij} \leftarrow \tau_{ij} - \Delta\tau$
27: **end for**
28: **end if**
29: $pop(t) \leftarrow pop(t) \cup s^{ib}$
30: **for** each arc$(i,j) \in s^{ib}$ **do**
31: $\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau$
32: **end for**
33: **end if**

3.2. Adapting in Dynamic Environments

ACO algorithms have proven effective in addressing dynamic optimization problems because they are very robust algorithms, according to Bonabeau et al. [30]. However, when a dynamic change occurs, the ACO's pheromone trails of the previous environment will become outdated because they will be associated with the previous optimum. A straightforward way to tackle the dynamic changes in the DTSP using a conventional ACO, such as the $\mathcal{MM}$AS, is to restart the optimization process whenever changes occur. In particular, the pheromone trail limit values τ_{max} and τ_{min} are reset back to their initial values, and all the pheromone trails are re-initialized to the initial τ_0 value, as in Equation (5), whenever a dynamic change is detected.

There are two concerns when restarting an ACO: (1) the dynamic changes are not always detectable [31], and (2) a portion of the previous pheromone trails that can help discover the new optimum faster is erased [32]. For the first concern, the changes in the DTSP can be detected by re-evaluating a constant solution (or detector) on every iteration (note that more than one detector can be used). If a change occurs in the quality of the detector, then a dynamic change is recorded. In this way, a change is always detected for the DTSP with node changes but not for the DTSP with weight changes. This is because the weights affected by the dynamic change may not belong to that particular detector, and hence, its quality will not be affected [10]. For the second concern, the pheromone trails of the previous environment will be most probably useful when the changing environments are correlated, e.g., when the dynamic changes are small to medium, the new environment is more likely to be similar to the previous environment. As a consequence, some of the previous pheromone trails will still be useful to the new environment.

Therefore, a more efficient way is to allow the pheromone evaporation defined in Equation (5) to remove the useless pheromone trails and utilize the useful pheromone trails of the previous environment. Specifically, the useless pheromone trails will reach the τ_{min} value due to the constant deduction of pheromone evaporation, whereas the useful pheromone trails will be reinforced up to the τ_{max} due to the pheromone deposit in Equation (6). In other words, ACO will be able to adapt to the newly generated environment.

Figure 1 demonstrates the performance of $\mathcal{MM}$AS with and without pheromone re-initialization and proves our claims above. Specifically, it can be easily observed that when the dynamic changes are small (i.e., $m = 0.1$), adapting to the changes maintains better offline performance (see Equation (9)) in most environmental changes (except when $T = 3$), whereas when the dynamic changes are severe (i.e., $m = 0.75$), re-initializing the pheromone trails maintains better offline performance in most environmental changes (except when $T = 7$ and $T = 9$). More details regarding the experimental setup of Figure 1 are given later in Section 4.

A more advanced way is to use the pheromone update policy of the population-based ACO (P-ACO) that has been specifically designed to address optimization problems with dynamic environments [13]. The overall P-ACO pheromone procedure is presented in Algorithm 2, lines 21–33. In particular, an archive, i.e., $pop(t)$, of iteration-best ants is maintained. For every iteration t, the iteration-best ant enters $pop(t)$, and a positive constant pheromone update is added to the trails associated with the arcs that belong to its solution as follows:

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau, \forall\ (i,j) \in s^{ib}, \tag{7}$$

where $\Delta\tau = (\tau_{max} - \tau_0)/K$ is the constant pheromone value added, s^{ib} is the solution of the iteration-best ant, K is the size of $pop(t)$, and τ_{max} and τ_0 are the maximum and initial pheromone trail values, respectively.

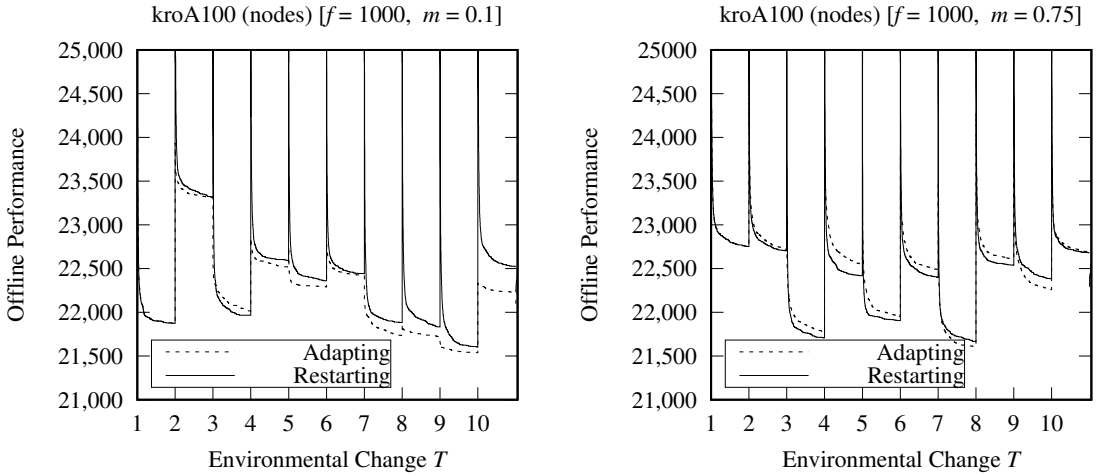

Figure 1. Offline performance results (averaged over 50 runs) of $\mathcal{MM}$AS without pheromone re-initialization (Adapting) and of $\mathcal{MM}$AS with pheromone re-initialization (Restarting) for each environmental change on `kroA100` with nodes with small dynamic changes (**left**) and severe dynamic changes (**right**).

Whenever a dynamic change occurs, the solutions represented by the current ants in the population list may become invalid (e.g., when existing nodes are removed and replaced with new nodes). Therefore, these solutions are repaired heuristically using the *KeepElite* principle [33]. Specifically, the affected nodes are removed from the solution, reconnecting the successor and predecessor nodes. The new nodes are inserted in the best possible position of the solution, causing, in this way, the least increase in the solution quality. At the same time, the corresponding pheromone trails associated with the arcs of the affected nodes are updated.

When $pop(t)$ is full, the current iteration-best ant needs to replace an existing ant, say r, in $pop(t)$, following a negative constant update to its corresponding pheromone trails, which is defined as follows:

$$\tau_{ij} \leftarrow \tau_{ij} - \Delta\tau, \forall\ (i,j) \in s^r, \tag{8}$$

where $\Delta\tau = (\tau_{max} - \tau_0)/K$ is the constant deducted pheromone value (the same with the added value) and s^r is the solution of the ant to be replaced, typically the oldest entry of the population list $pop(t)$. Note that pheromone evaporation is not used in the P-ACO algorithm. However, it is able to adapt to the changes because outdated pheromone trails are removed directly from the population list.

The overall application of $\mathcal{MM}$AS and P-ACO algorithms for the DTSP is described in Algorithm 3. It is worth mentioning that in both $\mathcal{MM}$AS and P-ACO, the solution of the best-so-far ant is repaired using *KeepElite*, and the heuristic information is updated, reflecting the newly generated distances as in Equation (2), whenever a dynamic change occurs (see Algorithm 3, lines 16–33).

Algorithm 3 ACO for DTSP

```
t ← 0
C^bs ← ∞
for ∀(i, j) ∈ A do
    τ_ij ← τ_0
    η_ij ← 1/d_ij(0)
end for
while termination condition not met do
    Construct Solutions(t)                % Algorithm 1
    s^ib ← find best ant at iteration t
    if φ(s^ib, t) is better than φ(s^bs, t) then
        s^bs ← s^ib
        C^bs ← C^ib
    end if
    Pheromone Update(t, s^ib, s^bs)       % Algorithm 2
    if dynamic change occurs then
        if MMAS | | P-ACO then
            Repair s^bs
            for ∀(i, j) ∈ A do
                η_ij ← 1/d_ij(T)
            end for
            if P-ACO then
                for ∀(i, j) ∈ A do
                    τ_ij ← τ_0
                end for
                for each ant k ∈ pop(t) do
                    Repair s^k
                    for each arc(i, j) ∈ s^k do
                        τ_ij ← τ_ij + Δτ
                    end for
                end for
            end if
        end if
    end if
    t ← t + 1
end while
```

4. Experimental Results

4.1. Experimental Setup

The dynamic benchmark generator, described in Section 2, is used to generate the DTSP test cases. The following three stationary TSP benchmark instances are obtained from TSPLIB (Available at http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/ (accessed on 3 April 2023)): `kroA100`, `rat575`, and `pr1002`. The magnitude of change m is set to slightly (i.e., $m = 0.1$), medium (i.e., $m = 0.25$ and $m = 0.5$), and severely (i.e., $m = 0.75$) changing environments. The frequency of change f is set to $10n$ iterations (i.e., $f = 1000$ for `kroA100`, $f = 5750$ for `rat575`, and $f = 10{,}020$ for `pr1002`) to allow sufficient time for the ACO algorithms to converge—and it is scalable for each problem instance. For each type of DTSP (i.e., weights or nodes changes), a total of four test cases are generated from each static benchmark instance. Ten environmental changes are allowed for each DTSP test case.

The modified *offline performance* [34] is used as the performance metric, which is defined as follows:

$$\bar{P}_{offline} = \frac{1}{E} \sum_{t=1}^{E} \phi(s^{bs'}, t), \tag{9}$$

where E is the number of observations taken, and $\phi(s^{bs'}, t)$ is the solution quality value of the best-so-far solution since the last dynamic change.

The common ACO parameters of $\mathcal{MM}$AS and P-ACO are set to $\alpha = 1$, $\beta = 5$, and $\omega = 25$. The evaporation rate of $\mathcal{MM}$AS is investigated with values $\rho = \{0.1, 0.2, 0.5, 0.8\}$, and it is set to $\rho = 0.8$. Furthermore, the exploitation strength of the decision rule is set to $q_0 = 0.0$, and the initial pheromone trail value $\tau_0 = 1/\rho C^{nn}$, where $C^{nn} = \phi(s^{nn}, 0)$ is the solution quality generated by the nearest-neighbor heuristic. The maximum and minimum pheromone trail limits are set to $\tau_{max} = 1/\rho C^{best}$ and $\tau_{min} = \tau_{max}\left(1 - \sqrt[n]{0.05}\right) / \left((cand - 1) \cdot \sqrt[n]{0.05}\right)$, respectively, where *cand* is the number of different choices available to an ant at each step. The frequency with which the best-so-far ant deposits pheromone occasionally is set for every 25 iterations [29]. The population list size of the P-ACO is investigated with values $K = \{2, 3, 5, 10\}$, and it is set to $K = 3$. Furthermore, the initial pheromone trail value is set to $\tau_0 = 1/(n - 1)$, the exploitation strength of the decision rule is set to $q_0 = 0.5$, and the maximum pheromone trails limit is to $\tau_{max} = 1$ [35].

4.2. Statistical Comparisons Using Mean and Standard Deviation

Since ACO algorithms are non-deterministic stochastic metaheuristics, 50 independent runs were executed on the same set of random seed numbers to perform statistical comparisons. Tables 1 and 2 shows the mean and standard deviation of the $\bar{P}_{offline}$ results obtained by the $\mathcal{MM}$AS and P-ACO algorithms for the DTSP with node and weight changes, respectively. In addition, the statistical test comparisons are reported using the Wilcoxon rank-sum statistical test with a significance level of 0.05. Figures 2–4 show plots of $\bar{P}_{offline}$ against the environmental changes, for `kroA100`, `rat575`, and `pr1002`, respectively. From the comparisons, the following observations can be drawn.

Table 1. Mean and standard deviation results of ACO algorithms for the DTSP with node changes.

Algorithm $m \Rightarrow$	**0.1**	**0.25**	**0.5**	**0.75**
		`kroA100`		
$\mathcal{MM}$AS	22,223.86 ± 102.9	22,492.60 ± 108.1	22,537.36 ± 95.2	22,472.96 ± 70.7
P-ACO	**22,118.66 ± 49.0**	**22,350.54 ± 67.1**	**22,419.70 ± 54.7**	**22,320.62 ± 48.8**
		`rat575`		
$\mathcal{MM}$AS	**6573.22 ± 25.1**	6577.54 ± 66.1	**6399.26 ± 22.5**	6401.76 ± 15.3
P-ACO	6581.46 ± 25.9	**6443.86 ± 20.4**	6566.00 ± 43.8	**6392.60 ± 19.8**
		`pr1002`		
$\mathcal{MM}$AS	**305,793.32 ± 1719.4**	**315,315.60 ± 1633.72**	314,533.80 ± 1364.8	**315,447.28 ± 1419.8**
P-ACO	308,342.64 ± 733.1	315,763.52 ± 456.0	**313,705.60 ± 445.2**	315,500.44 ± 490.4

Bold values indicate statistical significance.

Table 2. Mean and standard deviation results of ACO algorithms for the DTSP with weight changes.

Algorithm $m \Rightarrow$	**0.1**	**0.25**	**0.5**	**0.75**
		`kroA100`		
$\mathcal{MM}$AS	20,543.36 ± 164.7	**20,600.40 ± 62.0**	20,222.00 ± 60.6	**19,869.66 ± 58.2**
P-ACO	**20,462.34 ± 36.4**	20,663.38 ± 64.7	**20,196.60 ± 44.7**	19,900.18 ± 37.9
		`rat575`		
$\mathcal{MM}$AS	**6621.04 ± 23.4**	**6465.78 ± 23.1**	**6404.72 ± 22.1**	**6399.72 ± 12.7**
P-ACO	6758.06 ± 70.5	6675.62 ± 25.3	6643.98 ± 15.0	6618.88 ± 13.2
		`pr1002`		
$\mathcal{MM}$AS	**266,132.52 ± 1625.4**	**264,653.36 ± 1902.2**	**268,062.60 ± 1104.2**	**268,531.12 ± 1255.6**
P-ACO	282,904.76 ± 592.3	276.911.88 ± 394.2	275,933.72 ± 382.7	275,841.04 ± 494.7

Bold values indicate statistical significance.

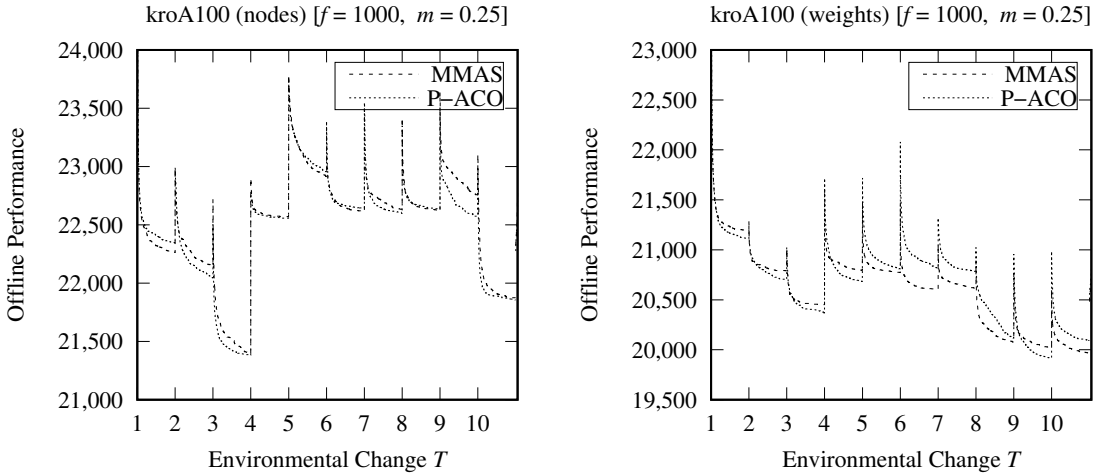

Figure 2. Offline performance results (averaged over 50 runs) of $\mathcal{MM}$AS and P-ACO for each environmental change on `kroA100` with node (**left**) and weight (**right**) changes.

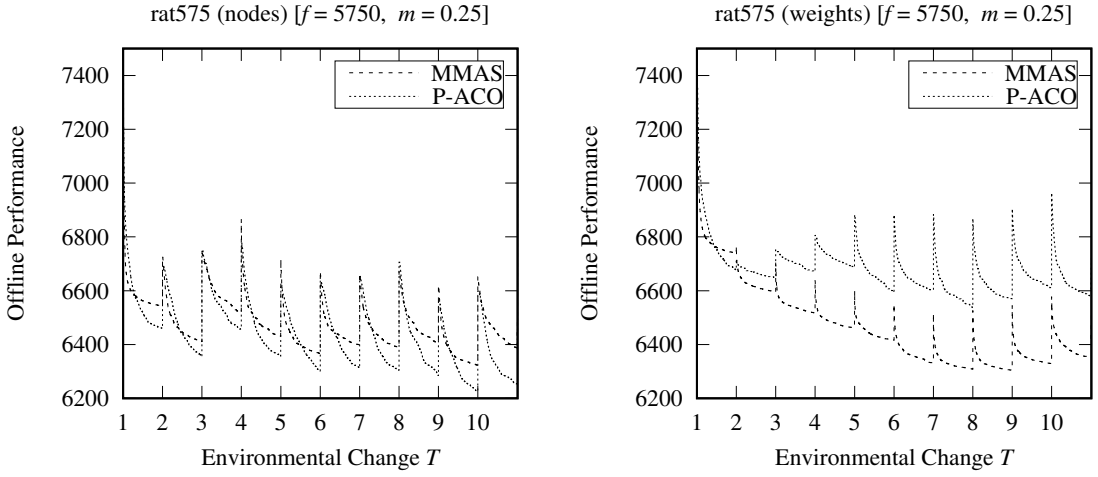

Figure 3. Offline performance results (averaged over 50 runs) of $\mathcal{MM}$AS and P-ACO for each environmental change on `rat575` with node (**left**) and weight (**right**) changes.

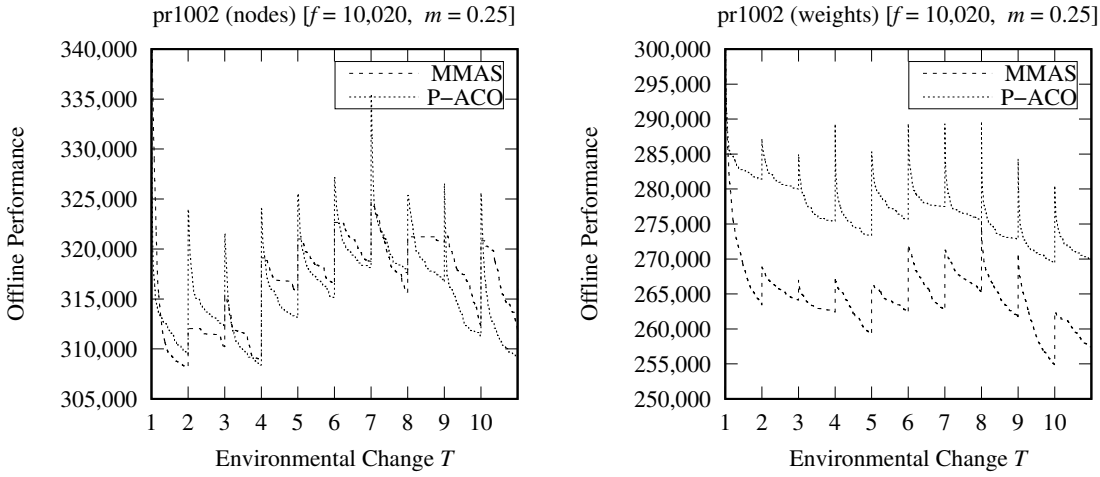

Figure 4. Offline performance results (averaged over 50 runs) of $\mathcal{MM}$AS and P-ACO for each environmental change on `pr1002` with node (**left**) and weight (**right**) changes.

First, P-ACO performs significantly better than $\mathcal{MM}$AS in most DTSPs with node changes. This is because the P-ACO has been designed to address DTSPs with node changes. In particular, when ants are currently stored in the population list when a dynamic change occurs, they will be repaired heuristically using change-related information (i.e., the inserted and removed nodes). Therefore, a constant amount of pheromones will be removed directly from the outdated trails associated with the arcs connecting the nodes to be removed. At the same time, the pheromone trails associated with the arcs connecting the nodes to be inserted will receive a constant amount of pheromone. This specific pheromone update will have a direct effect on the solution construction of the next iteration. On the other hand, the pheromone evaporation of $\mathcal{MM}$AS requires more iterations to express its effect (e.g., more than the frequency used in these experiments [10]). Also, the removal of outdated pheromone trails is not applied locally to the affected areas, as in the P-ACO, because the evaporation is applied globally on all arcs.

Second, $\mathcal{MM}$AS performs significantly better than P-ACO in most DTSPs with weight changes. This is because repairing the ants stored in the population list has no effect when a dynamic change of this type occurs. In particular, the feasibility of the stored solutions is not affected as in the case of the DTSP with node changes. Only the solution quality will change (which will be re-evaluated); however, because a constant pheromone update is performed, it has no effect on the P-ACO's behavior. However, since the pheromone trails in P-ACO are removed directly because of the constant pheromone removal, there is a higher risk of erasing knowledge from previous environments. On the other hand, the pheromone evaporation in $\mathcal{MM}$AS is able to better adapt to these types of dynamic changes because the decrease in the pheromone trails is performed gradually.

4.3. Quantile Comparisons Using Peak, Average, and Bad-Case Performances

Although the mean and standard deviation values presented in the previous section are useful in performing statistical comparisons between ACO algorithms, they are not very informative [17]. For example, they do not provide any indication of the expected solution quality value in case ACO is executed once, which is the case in real-world applications. On the other hand, the quantiles of the distribution values (from multiple independent runs) address this issue as follows: if the quantile Q_p of the solution quality value of an ACO algorithm is C^{bs}, then the probability of achieving a solution quality value better or equal to C^{bs} is greater than or equal to p. Tables 3 and 4 show the quantiles $Q_{0.10}$, $Q_{0.50}$, and $Q_{0.90}$ of $\mathcal{MM}$AS and P-ACO for the DTSP with node and weight changes, respectively. Quantile $Q_{0.10}$ indicates the peak performance, $Q_{0.50}$ indicates the average performance, and $Q_{0.90}$ indicates the bad-case performance of algorithms. In addition, Figures 5–7 show the box plots of $\mathcal{MM}$AS and P-ACO drawn between the first and third quartile of the distribution on `kroA100`, `rat575`, and `pr1002`, respectively, for the different DTSP test cases.

Table 3. Experimental results of ACO algorithms for the DTSP with node changes.

Algorithm	**kroA100**				**rat575**				**pr1002**			
$m \Rightarrow$	**0.1**	**0.25**	**0.5**	**0.75**	**0.1**	**0.25**	**0.5**	**0.75**	**0.1**	**0.25**	**0.5**	**0.75**
						$Q_{0.10}$						
$\mathcal{MM}$AS	22,127	22,390	22,440	22,385	6573	6459	**6375**	6381	**302,494**	**311,643**	**311,175**	**312,291**
P-ACO	**22,084**	**22,266**	**22,356**	**22,268**	**6546**	**6418**	6397	**6367**	307,513	315,147	313,007	314,951
						$Q_{0.50}$						
$\mathcal{MM}$AS	22,186	22,467	22,520	22,486	**6570**	6483	**6395**	6402	**306,249**	315,903	314,514	316,560
P-ACO	**22,102**	**22,341**	**22,416**	**22,319**	6583	**6445**	6432	**6390**	308,586	**315,880**	**313,964**	**315,455**
						$Q_{0.90}$						
$\mathcal{MM}$AS	22,400	22,670	22,676	22,556	6613	6518	**6431**	6420	309,279	318,927	317,478	318,266
P-ACO	**27,212**	**22,432**	**22,506**	**22,386**	**6610**	**6467**	6481	**6419**	**309,059**	**316,302**	**314,222**	**316,100**

Bold values indicate the best results of each comparison.

Table 4. Experimental results of ACO algorithms for the DTSP with weight changes.

Algorithm	**kroA100**				**rat575**				**pr1002**			
$m \Rightarrow$	**0.1**	**0.25**	**0.5**	**0.75**	**0.1**	**0.25**	**0.5**	**0.75**	**0.1**	**0.25**	**0.5**	**0.75**
						$Q_{0.10}$						
$\mathcal{MM}$AS	**20,415**	**20,531**	**20,147**	**19,804**	**6594**	**6493**	**6380**	**6383**	**260,009**	**258,675**	**266,618**	**267,049**
P-ACO	20,424	20,590	**20,147**	19,854	6658	6643	6624	6603	28,2195	276,460	275,443	275,157
						$Q_{0.50}$						
$\mathcal{MM}$AS	20,483	**20,590**	20,223	**19,873**	**6618**	**6465**	**6402**	**6399**	**267,186**	**265,953**	**268,035**	**268,321**
P-ACO	**20,453**	20,654	**20,196**	19,895	6767	6680	6644	6620	282,958	276,900	275,998	275,900
						$Q_{0.90}$						
$\mathcal{MM}$AS	20,881	**20,689**	20,308	**19,944**	**6658**	**6499**	**6439**	**6417**	**270,615**	**269,053**	**269,799**	**270,295**
P-ACO	**20,519**	20,753	**20,245**	19,956	6833	6706	6662	6634	283,574	277,404	276,372	276,496

Bold values indicate the best results of each comparison.

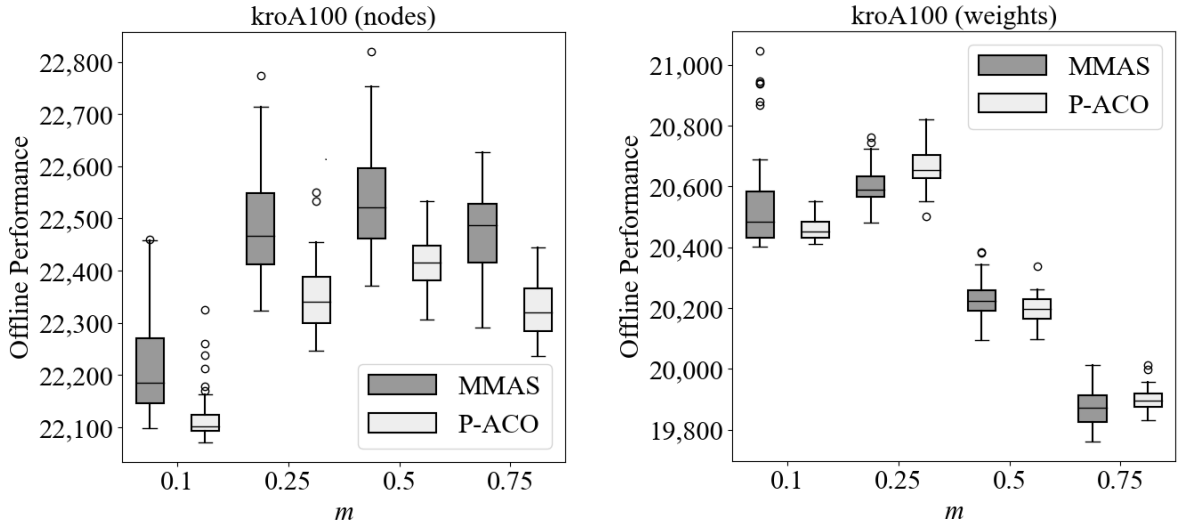

Figure 5. Box plots of $\mathcal{MM}$AS and P-ACO drawn between the first ($Q_{0.25}$) and third quartile ($Q_{0.75}$) of the distribution on `kroA100` with node (**left**) and weight (**right**) changes.

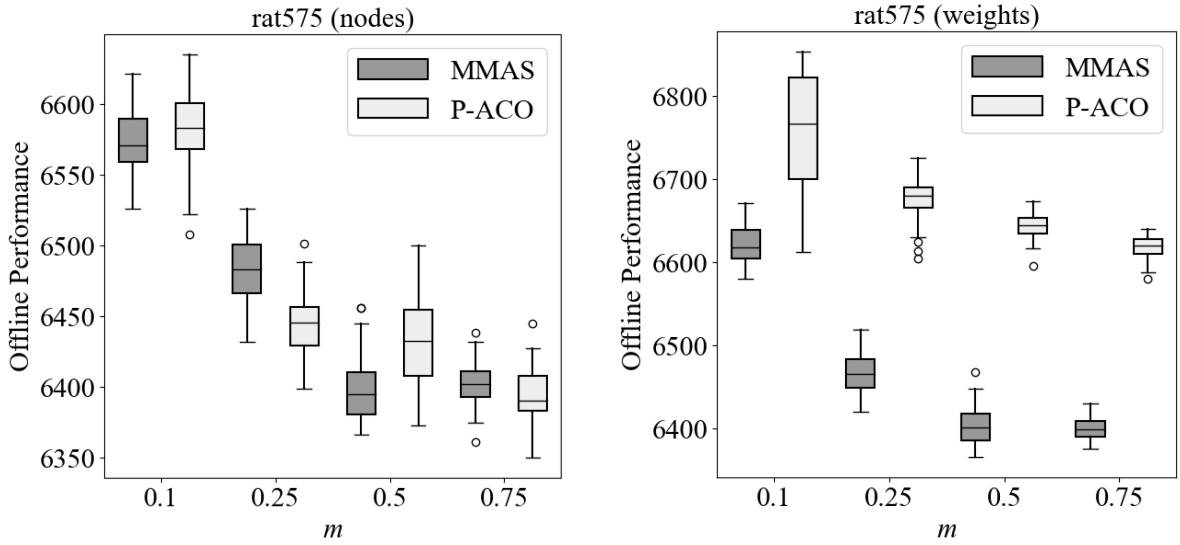

Figure 6. Box plots of $\mathcal{MM}$AS and P-ACO drawn between the first ($Q_{0.25}$) and third quartile ($Q_{0.75}$) of the distribution on `rat575` with node (**left**) and weight (**right**) changes.

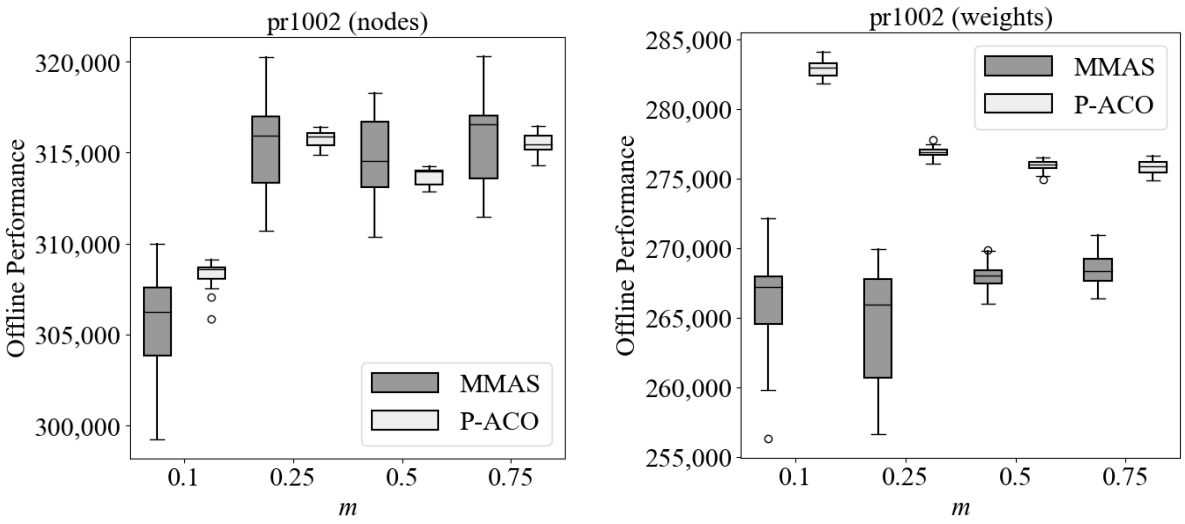

Figure 7. Box plots of $\mathcal{MM}$AS and P-ACO drawn between the first ($Q_{0.25}$) and third quartile ($Q_{0.75}$) of the distribution on `pr1002` with node (**left**) and weight (**right**) changes.

Another issue that the use of quantiles may address is when the distribution of executing ACO algorithms multiple times is asymmetrical (i.e., having several outliers such as the distribution of $\mathcal{MM}$AS in Figure 5 for `kroA100` with weight changes, and P-ACO in Figure 6 for `rat575` with weight changes), affecting the mean comparisons [17]. Therefore, the quantile $Q_{0.50}$, that defines the median of the distribution, may be a more adequate choice to measure the average performance of ACO algorithms than the mean values shown in Section 4.2. This effect is demonstrated when comparing the mean results of ACO algorithms in Table 1 (and Table 2) with the corresponding median results in Table 3 (and Table 4). Specifically, it can be observed that the comparisons of the DTSP with weight changes are consistent, whereas the comparisons of the DTSP with node changes have several inconsistencies as follows. P-ACO has better mean values than $\mathcal{MM}$AS in most DTSP cases of `pr1002`, while $\mathcal{MM}$AS has better median values than P-ACO on the same DTSP cases. The difference regarding the distribution of P-ACO and $\mathcal{MM}$AS can be observed in Figure 7, in which P-ACO has a more symmetrical and concentrated distribution than $\mathcal{MM}$AS.

From the quantile comparisons in Tables 3 and 4, it can also be observed that: first, the peak performances of both ACO algorithms are better, as expected, than the average performance or bad-case performance, and second, the average performance (i.e., $Q_{0.50}$) comparisons in Table 4 for the DTSP with weight changes are consistent with the peak (i.e., $Q_{0.10}$) and bad-case (i.e., $Q_{0.90}$) performance comparisons since $\mathcal{MM}$AS outperforms P-ACO in all cases. In contrast, when comparing the average performance results with the peak performance results for the DTSP with node changes in Table 3, there are some inconsistencies. For example, $\mathcal{MM}$AS has a better peak performance than P-ACO on the `pr1002` test cases, whereas P-ACO has a better average performance than $\mathcal{MM}$AS on the same test cases.

In summary, $\mathcal{MM}$AS may be a better choice for DTSP with weight changes when the independent execution is performed in parallel. This is because it has a better peak performance than P-ACO, and hence, it is expected to obtain a solution at least as good as $Q_{0.10}$. In contrast, P-ACO may be a better choice when a single execution is performed because there are more chances in obtaining a solution as good as $Q_{0.50}$.

5. Conclusions

In this work, two ACO variations are applied to optimization problems in a dynamic environment. The TSP is used as the base problem to systematically generate dynamic test cases. Two types of dynamic changes are considered: (1) when nodes change and (2) when

weights change. From the experiments, the follow concluding remarks can be drawn. First, P-ACO performs significantly better than $\mathcal{MM}$AS in most DTSPs with node changes. Second, $\mathcal{MM}$AS is significantly better than P-ACO in DTSP with weight changes. Third, P-ACO has better average performance than $\mathcal{MM}$AS in most DTSP with node changes. Fourth, $\mathcal{MM}$AS has a better peak performance in most DTSPs with weight changes.

In general, complex problems arising in practical applications, such as training a neural network, require huge computational power. As a consequence, the computation power for optimizing complex problems as well as performing several independent executions (due to the stochastic nature of swarm and evolutionary computation algorithms) in parallel may not be available. Therefore, optimization algorithms with better average performances may be the best choice rather than optimization algorithms with better peak performance.

For future work, it will be interesting to apply and evaluate ACO algorithms in more practical applications, such as the agile earth observation satellites' scheduling problem [36,37]. In fact, this particular problem can be solved as a constrained TSP [38] and consists of several unexpected environmental changes that may affect the scheduled tasks [39,40]. Hence, the adaptation capabilities and evaluation of the ACO demonstrated in this work may be suitable to address the aforementioned problem.

Author Contributions: Conceptualization, M.M.; methodology, M.M.; software, M.M.; validation, M.M. and M.N.A.; formal analysis, M.M. and M.N.A.; investigation, M.M.; resources, M.M.; data curation, M.M.; writing—original draft preparation, M.M.; writing—review and editing, M.M., M.N.A. and D.H.; visualization, M.M.; supervision, M.M.; funding acquisition, D.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially funded by the 'EXCELSIOR' project (European Union's Horizon 2020 research and innovation programme), grant number "857510"; and the 'AI-OBSERVER' project (European Union's Horizon Europe framework programme), grant number "101079468".

Data Availability Statement: All data used in this research are provided in this article.

Acknowledgments: The authors acknowledge the "EXCELSIOR": ERATOSTHENES: Excellence Research Centre for Earth Surveillance and Space-Based Monitoring of the Environment H2020 Widespread Teaming project (www.excelsior2020.eu, accessed on 26 November 2023). The "EXCELSIOR" project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 857510, from the Government of the Republic of Cyprus through the Directorate General for the European Programmes, Coordination and Development and the Cyprus University of Technology. The authors acknowledge the "AI-OBSERVER" project funded by the European Union's Horizon Europe Framework Programme HORIZON-WIDERA-2021-ACCESS-03 (Twinning) under the Grant Agreement No. 101079468.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Dorigo, M.; Gambardella, L.M. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Trans. Evol. Comput.* **1997**, *1*, 53–66. [CrossRef]
2. Dorigo, M.; Caro, G.D.; Gambardella, L.M. Ant Algorithms for Discrete Optimization. *Artif. Life* **1999**, *5*, 137–172. [CrossRef] [PubMed]
3. Colorni, A.; Dorigo, M.; Maniezzo, V. Distributed optimization by ant colonies. In Proceedings of the European Conference on Artificial Life, Paris, France, 13 December 1991; pp. 134–142.
4. Mavrovouniotis, M.; Ellinas, G.; Polycarpou, M. Electric Vehicle Charging Scheduling Using Ant Colony System. In Proceedings of the 2019 IEEE Congress on Evolutionary Computation (CEC), Wellington, New Zealand, 10–13 June 2019; pp. 2581–2588.
5. Mavrovouniotis, M.; Yang, S. Dynamic vehicle routing: A memetic ant colony optimization approach. In *Automated Scheduling and Planning*; Studies in Computational Intelligence; Uyar, A., Ozcan, E., Urquhart, N., Eds.; Springer: Berlin/Heidelberg, Germany, 2013; pp. 283–301, Volume 505.
6. Mavrovouniotis, M.; Li, C.; Ellinas, G.; Polycarpou, M. Parallel Ant Colony optimization for the Electric Vehicle Routing Problem. In Proceedings of the 2019 IEEE Symposium Series on Computational Intelligence (SSCI), Xiamen, China, 6–9 December 2019; pp. 1660–1667.
7. Jin, Y.; Branke, J. Evolutionary optimization in uncertain environments–a survey. *IEEE Trans. Evol. Comput.* **2005**, *9*, 303–317. [CrossRef]

8. Yang, S.; Jiang, Y.; Nguyen, T. Metaheuristics for dynamic combinatorial optimization problems. *IMA J. Manag. Math.* **2013**, *24*, 451–480. [CrossRef]
9. Nguyen, T.; Yang, S.; Branke, J. Evolutionary Dynamic Optimization: A survey of the state of the Art. *Swarm Evol. Comput.* **2012**, *6*, 1–24. [CrossRef]
10. Mavrovouniotis, M.; Yang, S.; Van, M.; Li, C.; Polycarpou, M. Ant Colony Optimization Algorithms for Dynamic Optimization: A Case Study of the Dynamic Travelling Salesperson Problem [Research Frontier]. *IEEE Comput. Intell. Mag.* **2020**, *15*, 52–63. [CrossRef]
11. Mavrovouniotis, M.; Yang, S. Interactive and non-interactive hybrid immigrants schemes for ant algorithms in dynamic environments. In Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC), Beijing, China, 6–11 July 2014; pp. 1542–1549.
12. Mavrovouniotis, M.; Yang, S. An immigrants scheme based on environmental information for ant colony optimization for the dynamic travelling salesman problem. In Proceedings of the Artificial Evolution, Angers, France, 24–26 October 2012; Volume 7401, pp. 1–12.
13. Guntsch, M.; Middendorf, M. Applying Population Based ACO to Dynamic Optimization Problems. In Proceedings of the Ant Algorithms, Brussels, Belgium, 12–14 September 2002; Volume 2463, pp. 111–122.
14. Melo, L.; Pereira, F.; Costa, E. Multi-caste Ant Colony Algorithm for the Dynamic Traveling Salesperson Problem. In Proceedings of the Adaptive and Natural Computing Algorithms, Lausanne, Switzerland, 4–6 April 2013; Volume 7824, pp. 179–188.
15. Mavrovouniotis, M.; Müller, F.M.; Yang, S. An ant colony optimization based memetic algorithm for the dynamic travelling salesman problem. In Proceedings of the 2015 Genetic and Evolutionary Computation Conference (GECCO15), Madrid, Spain, 11–15 July 2015; pp. 49–56.
16. Stützle, T.; Hoos, H. Improvements on the Ant-System: Introducing the MAX-MIN Ant System. In Proceedings of the Artificial Neural Nets and Genetic Algorithms, Norwich, UK, 1997; pp. 245–249.
17. Ivković, N.; Kudelić, R.; Črepinšek, M. Probability and Certainty in the Performance of Evolutionary and Swarm Optimization Algorithms. *Mathematics* **2022**, *10*, 4364. [CrossRef]
18. Garey, M.; Johnson, D. *Computer and Intractability: A Guide to the Theory of NP-Completeness*; Freeman: San Francisco, CA, USA, 1979.
19. Flood, M.M. The Traveling-Salesman Problem. *Oper. Res.* **1956**, *4*, 61–75. [CrossRef]
20. Eyckelhof, C.; Snoek, M. Ant Systems for a Dynamic TSP. In Proceedings of the Ant Algorithms, Brussels, Belgium, 12–14 September 2002; Volume 2463, pp. 88–99.
21. Liu, J. Rank-Based Ant Colony Optimization Applied to Dynamic Traveling Salesman Problems. *Eng. Optim.* **2005**, *37*, 831–847. [CrossRef]
22. Mavrovouniotis, M.; Van, M.; Yang, S. Pheromone modification strategy for the dynamic travelling salesman problem with weight changes. In Proceedings of the 2017 IEEE Symposium Series on Computational Intelligence (SSCI), Honolulu, HI, USA, 27 November–1 December 2017; pp. 1577–1584.
23. de Oliveira, S.M.; Bezerra, L.C.; Stützle, T.; Dorigo, M.; Wanner, E.F.; de Souza, S.R. A computational study on ant colony optimization for the traveling salesman problem with dynamic demands. *Comput. Oper. Res.* **2021**, *135*, 105359. [CrossRef]
24. Dorigo, M. Ant colony optimization. *Scholarpedia* **2007**, *2*, 1461. [CrossRef]
25. Dorigo, M.; Maniezzo, V.; Colorni, A. Ant system: Optimization by a colony of cooperating agents. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **1996**, *26*, 29–41. [CrossRef] [PubMed]
26. Dorigo, M.; Birattari, M.; Stützle, T. Ant colony optimization. *IEEE Comput. Intell. Mag.* **2006**, *1*, 28–39. [CrossRef]
27. Gambardella, L.M.; Dorigo, M. Ant-Q: A reinforcement learning approach to the traveling salesman problem. In *Machine Learning Proceedings 1995*; Prieditis, A., Russell, S., Eds.; Morgan Kaufmann: San Francisco, CA, USA, 1995; pp. 252–260.
28. Stützle, T.; Hoos, H. MAX–MIN Ant System and local search for the traveling salesman problem. In Proceedings of the 1997 IEEE International Conference on Evolutionary Computation, Indianapolis, IN, USA, 13–16 April 1997; pp. 309–314.
29. Stützle, T.; Hoos, H.H. MAX–MIN Ant System. *Future Gener. Comput. Syst.* **2000**, *16*, 889–914. [CrossRef]
30. Bonabeau, E.; Dorigo, M.; Theraulaz, G. *Swarm Intelligence: From Natural to Artificial Systems*; Oxford University Press: New York, NY, USA, 1999.
31. Richter, H. Detecting change in dynamic fitness landscapes. In Proceedings of the 2009 IEEE Congress on Evolutionary Computation, Trondheim, Norway, 18–21 May 2009; pp. 1613–1620.
32. Angus, D.; Hendtlass, T. Ant Colony Optimisation Applied to a Dynamically Changing Problem. In Proceedings of the Developments in Applied Artificial Intelligence, Cairns, Australia, 17–20 June 2002; Volume 2358, pp. 618–627.
33. Guntsch, M.; Middendorf, M.; Schmeck, H. An Ant Colony Optimization Approach to Dynamic TSP. In Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation, GECCO'01, San Francisco, CA, USA, 7–11 July 2001; pp. 860–867.
34. Branke, J.; Schmeck, H. Designing Evolutionary Algorithms for Dynamic Optimization Problems. In *Advances in Evolutionary Computing*; Ghosh, A., Tsutsui, S., Eds.; Natural Computing Series; Springer: Berlin/Heidelberg, Germany, 2003; pp. 239–262.
35. Oliveira, S.; Hussin, M.S.; Roli, A.; Dorigo, M.; Stützle, T. Analysis of the population-based ant colony optimization algorithm for the TSP and the QAP. In Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), Donostia, Spain, 5–8 June 2017; pp. 1734–1741.
36. Chen, X.; Reinelt, G.; Dai, G.; Wang, M. Priority-based and conflict-avoidance heuristics for multi-satellite scheduling. *Appl. Soft Comput.* **2018**, *69*, 177–191. [CrossRef]

37. Wang, X.; Wu, G.; Xing, L.; Pedrycz, W. Agile Earth Observation Satellite Scheduling Over 20 Years: Formulations, Methods, and Future Directions. *IEEE Syst. J.* **2021**, *15*, 3881–3892. [CrossRef]
38. Du, B.; Li, S.; She, Y.; Li, W.; Liao, H.; Wang, H. Area targets observation mission planning of agile satellite considering the drift angle constraint. *J. Astron. Telesc. Instruments Syst.* **2018**, *4*, 047002.
39. Povéda, G.; Regnier-Coudert, O.; Teichteil-Königsbuch, F.; Dupont, G.; Arnold, A.; Guerra, J.; Picard, M. Evolutionary Approaches to Dynamic Earth Observation Satellites Mission Planning under Uncertainty. In Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '19, Prague, Czech Republic, 13–17 July 2019; pp. 1302–1310.
40. He, L.; Liu, X.L.; Chen, Y.W.; Xing, L.N.; Liu, K. Hierarchical scheduling for real-time agile satellite task scheduling in a dynamic environment. *Adv. Space Res.* **2019**, *63*, 897–912. [CrossRef]