Exchange Strategies for Multi-Colony Ant Algorithms in Dynamic Environments

Michalis Mavrovouniotis[1,2], Changhe Li[3], Danial Yazdani[4], Diofantos Handjimitsis[1,2]
[1]*ERATOSTHENES Centre of Excellence,*
Limassol, Cyprus.
[2]*Department of Civil Engineering and Geomatics, Cyprus University of Technology,*
Limassol, Cyprus.
[3]*School of Automation, China University of Geosciences,*
Hubei Key Laboratory of Advanced Control and Intelligent Automation for Complex Systems,
Wuhan, China.
[4]*Faculty of Engineering & Information Technology, University of Technology Sydney,*
Sydney, Australia.
michalis.mavrovouniotis@eratosthenes.org.cy, changhe.lw@gmail.com, danial.yazdani@gmail.com, d.hadjimitsis@cut.ac.cy

***Abstract*—In dynamic optimization problems where optimal solutions change over time, traditional ant colony optimization (ACO) algorithms face limitations. This study explores the adaptation of multi-colony ACO algorithms, known for their enhanced search capabilities in stationary problems, to tackle optimization problems in dynamic environments. Various strategies for exchanging information between colonies, which is a critical factor influencing algorithm performance, are investigated. Using the dynamic traveling salesman problem as a foundation, we generate test cases to reflect real-world complexities. Our results on a set of problem instances reveal that the choice of communication strategy between colonies significantly impacts the adaptability and efficiency of multi-colony ACO algorithms in tracking moving optimum.**

***Index Terms*—Ant colony optimization, multi-colony ant algorithm, dynamic optimization problem, dynamic traveling salesman problem, tracking moving optimum**

I. Introduction

Ant colony optimization (ACO) algorithms have been designed to address challenging stationary optimization problems [1], [2]. Their aim is to locate the problem's global optimum precisely and efficiently. However, many real-world optimization problems have a dynamic environment in which the global optimum changes over time (or moves in the search space) [3], [4]. This fact imposes challenges to ACO algorithms because once they converge to an optimum, the generated pheromone trails will bias the searching process towards that optimum after the environmental change.

A straightforward way to help ACO algorithms escape from the previously converged optimum, is to re-initialize all the pheromone trails uniformly after each environmental change. In this way, the re-optimization process starts from scratch and it is unbiased. However, such strategy requires large computational efforts, especially in $\mathcal{NP}$-hard combinatorial optimization problems [5], [6]. Additionally, this strategy overlooks the potential benefits of utilizing historical knowledge to enhance the optimization process of post-environmental changes. This oversight is particularly significant in problems where successive environmental states bear a degree of similarity – a common characteristic in numerous real-world scenarios [7].

In order to leverage the historical knowledge for accelerating and improving the re-optimization process after environmental changes, ACO algorithms must be enhanced so that they can accept the knowledge transferred. Over the years, several strategies have been proposed, including increasing diversity after a change [8], maintaining diversity during the execution [9]–[11], memory-based schemes [12], memetic algorithms [13], [14], and multi-colony approaches [15].

Multi-colony approaches have shown promising performance in optimization problems with dynamic environments [15]. In particular, the searching capability of ACO is improved since more than one colony explores and/or exploits in parallel updating their own pheromones independently. However, parallel ACO implementations require more computational resources than single ACO implementations since more than one copy of the algorithm will be executed [16]. Nowadays, the increasing availability of parallel computation power encourages the adoption of parallel multi-colony approaches [17]–[19]. As a result, more areas of the search space are more likely to be covered. In the experiments, several communication strategies are investigated that differ on the way colonies exchange information. A communication strategy

This work was partially supported by the European Union's HORIZON Research and Innovation Programme under grant agreement No 101120657, project ENFIELD (European Lighthouse to Manifest Trustworthy and Green AI), the AI-OBSERVER project funded from the European Union's Horizon Europe Framework Programme HORIZON-WIDERA-2021-ACCESS-03 (Twinning) under the Grant Agreement No 101079468, and the 'EXCELSIOR': ERATOSTHENES: Excellence Research Centre for Earth Surveillance and Space-Based Monitoring of the Environment H2020 Widespread Teaming project (www.excelsior2020.eu). 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.

is principally defined by the following key characteristics:

- The total number of colonies
- The designated neighborhood for information exchange
- The specific intervals at which information is exchanged

The rest of the paper is organized as follows. Section II describes the generation of dynamic environments. The traveling salesman problem (TSP) is used as the base problem to generate dynamic test cases. Section III describes ACO for the dynamic TSP (DTSP). Section IV describe the multi-colony ACO framework. Section V gives the experimental results of multi-colony ACO algorithms investigating their exchange strategies. Finally, Section VI concludes this paper and outlines several future works.

II. Dynamic Environments

A. Base Problem

The TSP is used as the base problem to generate dynamic test cases. The objective of the TSP is to find the shortest Hamiltonian path for a salesman given a set of cities. In particular, a Hamiltonian path is the path of visiting all cities once, starting from one city and return to it. The problem is represented by a complete weighted graph $G = (N, A)$, where N is a set of n cities and A is a set of arcs fully connecting the cities. Each arc $(i, j) \in A$ is associated with a non-negative value $w_{ij} \in \mathbb{R}^+$, which represents the distance between cities i and j for the classical TSP.

A particular TSP solution $\pi = \{1, \ldots, n\}$ is a permutation of city indices and the objective function is calculated as follows:

$$f(\pi) = w_{\pi(n)\pi(1)} + \sum_{i=1}^{n-1} w_{\pi(i)\pi(i+1)}. \tag{1}$$

B. Generating Dynamic Environments

Every TSP problem instance consists of a weight matrix that contains all the weights associated with the arcs of the corresponding graph G. In order to generate dynamic test cases the dynamic generator for combinatorial optimization problems is adopted [20]. In particular, the weight matrix of the TSP is subject to changes as follows:

$$\mathbf{W}(T) = \{w_{ij}(T)\}_{n \cdot n}, \tag{2}$$

where $\mathbf{W}(\cdot)$ is the weight matrix and T is the environmental period index which is synchronized with the algorithmic iterations during the optimization process. Therefore, the environmental period index is defined as $T = \lceil t/f \rceil$, where f is the frequency of change and t is the iteration counter of the optimization algorithm. As a consequence the objective function in Eq. (1) becomes $f(\pi, t)$.

The key concept to generate a dynamic test case is to replace nodes from the current working node set $N_{in}(T)$, where $N_{in}(0) = N$, with newly introduced nodes drawn from another set $N_{out}(T)$. The latter set $N_{out}(T)$ is initially generated with n new random nodes in the range of the N set. Real-world applications that encompass the aforementioned types of dynamic changes can be found in many fields, including transportation and logistics. For example, changes in the visiting locations (i.e., node changes) of a fleet of vehicles during their operational time.

A dynamic change of this type occurs as follows [20], [21]. Every f evaluations exactly $\lceil mn \rceil$ nodes are randomly selected from $N_{out}(T)$ to replace exactly $\lceil mn \rceil$ randomly selected nodes from $N_{in}(T)$, where m ($m \in (0, 1]$) defines the magnitude of change. The higher the value of m, the more nodes will be replaced and consequently the more weights in Eq. (2) will be affected.

III. Ant Colony Optimization for DTSP

In this work, the $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) [22] variant is utilized which is one of the most-studied ACO variants. The $\mathcal{MM}$AS has been successfully applied in several combinatorial optimization problems [23], and has also proven its adaptation capabilities in dynamic optimization problems [20].

In $\mathcal{MM}$AS, all the arcs (i, j) of the problem are associated with a pheromone trail value τ_{ij} which is uniformly initialized as follows: $\tau_{ij} \leftarrow \tau_0, \forall (i, j) \in A$, where τ_0 is the initial pheromone trail value. In addition, all the arcs (i, j) are associated with a heuristic information value η_{ij}, which in the case of the DTSP is set to $\eta_{ij} = 1/w_{ij}(T), \forall (i, j) \in A$.

A. Solution Construction

A colony of artificial ants start their tour in a randomly selected city. Then, each ant k chooses the next city biased by the existing pheromone trail and heuristic information values associated with arc (i, j). The tour is completed until all cities are visited.

The probability distribution with which ant k visits city j from city i 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, respectively, the existing pheromone trail and the priori available knowledge associated with arc (i, j). Parameters α and β determine the relative influence of τ_{ij} and η_{ij}, respectively, while $\mathcal{N}_i^k$ is the set of unvisited cities of size $cand$ for the k–th ant adjacent to city i.

B. Pheromone Update

The pheromone update policy of the $\mathcal{MM}$AS variant [22] first reduces the pheromone trails on all arcs due to evaporation, and then reinforces the pheromone trails on the arcs of the solution constructed by the best ant. The pheromone evaporation is applied as follows:

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij}, \forall (i, j) \in A, \tag{4}$$

where ρ ($\rho \in (0, 1]$) is the evaporation rate.

To reinforce the pheromone trails, the best ant is used to deposits pheromone on the arcs of its visited cities as follows:

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau_{ij}^{best}, \forall (i, j) \in \pi^{best}, \tag{5}$$

where $\Delta\tau_{ij}^{best}$ is the amount of pheromone that the best ant deposits which is inversely proportional to the quality of its solution, i.e., $1/C^{best}$ where $C^{best} = f(\pi^{best}, t)$. The "best" ant that is allowed to deposit pheromone may be either the best-so-far ant (i.e., the ant representing the best solution from all iterations so far), $\pi^{best} = \pi^{bs}$, or the iteration-best ant, in which case $\pi^{best} = \pi^{ib}$. These two ants are allowed to deposit pheromone accordingly in an alternate way [22]. More precisely, the best-so-far ant is allowed to deposit pheromone every g^{bs} iterations whereas in the rest of the iterations it is the iteration-best ant that deposits pheromone.

Furthermore, lower and upper limits of the pheromone trail values are explicitly imposed as follows:

$$\tau_{ij} \leftarrow \begin{cases} \tau_{max}, & \text{if } \tau_{ij} > \tau_{max}, \\ \tau_{min}, & \text{if } \tau_{ij} < \tau_{min}, \\ \tau_{ij}, & \text{otherwise}, \end{cases} \quad \forall (i,j) \in A, \tag{6}$$

where τ_{min} and τ_{max} are, respectively, the minimum and maximum pheromone trails.

IV. Multi-Colony Ant Algorithms

Multi-colony ACO algorithms were successfully applied in single objective optimization problems [24], as well as in multi-objective optimization problems [25], and in stationary optimization problems [26], as well as in dynamic optimization problems [15], which is the main focus of this work.

The idea of multi-colony ACO algorithms is to enable μ ($\mu > 1$) number of colonies to tackle an optimization problem simultaneously as described in Algorithm 1. The advantage of multi-colony ACO algorithms against conventional (single colony) ACO algorithms is that they enhance the exploration capabilities of ACO. This is due to the fact that each colony performs trail updates independently, and thus, multiple areas of the search space can be covered at the same time. As a result, premature convergence is addressed by enhancing the diversity generated by the colonies, and makes multi-colony ACO an ideal solution to track the moving optimum in dynamic optimization problems.

To enable communication between the colonies when optimizing a particular problem, a migration policy must be defined according to the following five aspects: 1) how many colonies used; 2) which colonies will communicate with each other; 3) how often the colonies will communicate; 4) for how long the colonies will be communicating; and 5) what kind of information the colonies will exchange. For the last two concerns only one option is considered in this work, that is, the best-so-far ant of the source colony is allowed to broadcast its information to destination colonies for a single iteration and the information is received by the destination colony only if the quality of the best-so-far ant of the source colony is better. For the remaining three aspects the options presented in Table I are investigated.

The simplest migration policy of multi-colony ACO algorithms is the parallel independent runs (`PIR`) in which the colonies do not communicate with each other [27]. Migration policies in which communication is enabled are defined by three different types of topologies used in the island model of parallel evolutionary algorithms [28]: `RING`, `CUBE`, and complete (`CMPL`), forming different neighborhoods for each colony. In the ring topology shown in Figs. 1(a), 2(a), and 3(a) each colony will always communicate with one colony, i.e., the colony next to it. In the complete topology shown in Figs. 1(b), 2(b), and 3(b) each colony will communicate with the remaining $(\mu - 1)$ colonies. The size of the neighborhood for a colony in the cube topology depends on the μ parameter. For example, each colonies in a d-dimensional cube topology

Algorithm 1 Multi-Colony ACO Framework

```
for each colony do
    initialize pheromone trails
    initialize heuristic information
end for
while termination condition is satisfied do
    for each colony do
        construct solutions
        update colony best ant
        update pheromone trails
    end for
    migration based on topology and schedule
    update best solution from all colonies
end while
```

TABLE I: Configuration of exchange strategies for multi-colony ACO algorithms.

μ	Topology	Schedule
4	`RING`	*short*
8	`CUBE`	*long*
16	`COMPL`	*automatic*

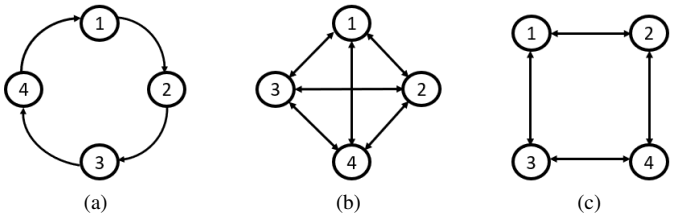


Fig. 1: Communication strategies of four colonies with (a) `RING`; (b) `COMPL`; and (c) `CUBE` topologies.

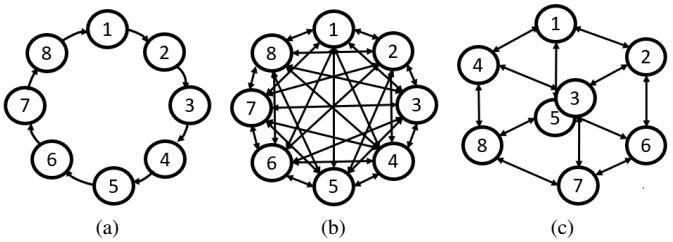
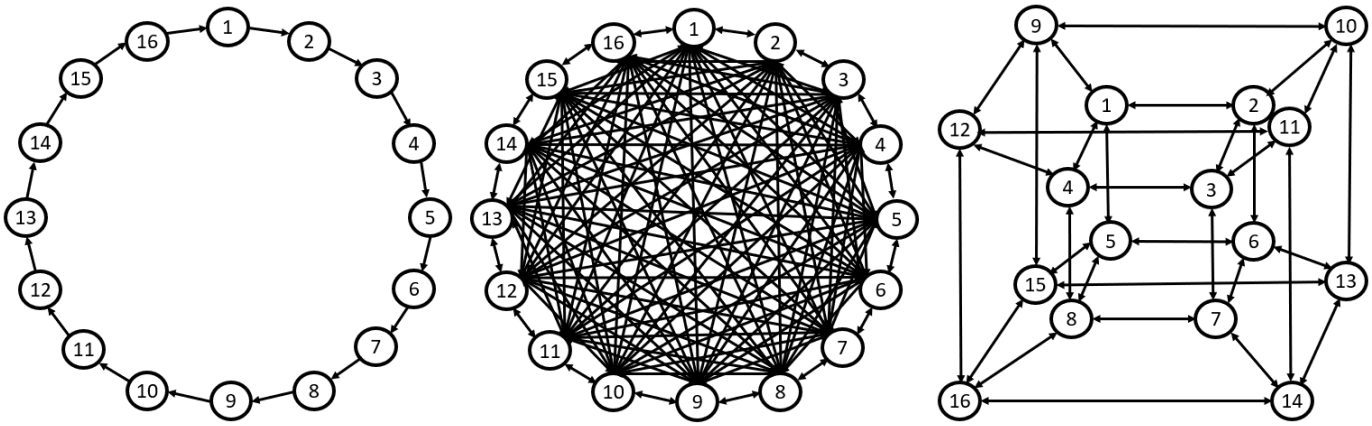


Fig. 2: Communication strategies of eight colonies with (a) `RING`; (b) `COMPL`; and (c) `CUBE` topologies.



(a) (b) (c)

Fig. 3: Communication strategies of sixteen colonies with (a) RING; (b) COMPL; and (c) CUBE topologies.

will have neighborhood size d as shown in Figs. 1(c), 2(c), and 3(c).

Finally, different types of migration schedules to exchange information are investigated in this work as shown in Table I. In the *short* and *long* schedules the colonies will communicated every $f/100$ and $f/10$ iterations, respectively [16], [26]. The fixed value of the migration frequency depends on the size of the problem, that is, $f = 10n$, for scalability purposes. In the *automatic* schedule colonies will communicate whenever a new best-so-far solution is discovered [15].

V. Experimental Results

A. Experimental Setup

The dynamic benchmark generator for combinatorial optimization problems described in Section II is used to generate DTSPs. The following three static benchmark instances are obtained from TSPLIB[1]: kroA200, rd400 and u1060. The magnitude of change m was set to 0.1, 0.25, 0.5 and 0.75 indicating slightly, to medium, to severely changing environments. The frequency of change f was set to $10n$ iterations to allow sufficient time for ACO algorithms to converge. A total of 4 DTSPs are generated from each static instance and 10 environmental changes were allowed for each DTSP test case.

Each ACO algorithm executes 30 independent runs on the same set of random seed numbers. The modified *offline performance* [29] was used to evaluate the performance of the algorithms, which is defined as follows:

$$\bar{P}_{offline} = \frac{1}{E}\sum_{t=1}^{E} C^*, \tag{7}$$

where E is the number of observations taken, and $C^* = f(\pi^{bs}, t)$ is the value of the best-so-far solution since the last dynamic change.

The ACO algorithms are configured following the guidelines in [22]. The size of colonies of multi-colony ACO algorithms is set to 25 ants. The number of homogeneous colonies investigated in this work is set as follows: $\mu \in \{1, 2, 4, 8, 16\}$. The evaporation rate is set to $\rho = 0.8$, the initial pheromone trail value is set to $\tau_0 = 1/\rho C^{nn}$, the upper and lower pheromone trails 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 $C^{nn} = f(\pi^{nn}, 0)$ is the solution quality generated by the nearest-neighbor heuristic initially and $cand$ is the number of different choices available to an ant at each step. The two parameters of the decision rule are set to $\alpha = 1$ and $\beta = 5$. The frequency with which the best-so-far ant is allowed to deposit pheromone is set to $g^{bs} = 25$ iterations.

B. Effect of Multiple Colonies

In this set of experiments a conventional ACO (when $\mu = 1$) is compared with multi-colony ACO (when $\mu > 1$). To investigate the effect of multiple parallel colonies no communication strategy is used and hence PIR is utilized. Figs. 4, 5, and 6 present the offline performance results of PIR with different number of colonies on kroA200, rd400, and u1060 problem instances, respectively. From the experimental results it can be observed that as the number of colonies increases the solution quality of PIR improves for all dynamic cases of m. The results are expected because the colonies are more likely to search different areas of the search space promoting in this way exploration. Therefore, when a dynamic change occurs there are more than one solution to utilize in order to transfer knowledge from the previous environment.

The experiments have been extended with three more μ values: 32, 64, and 128 colonies to investigate the scalability of multi-colony ACO. From Figs. 4, 5, and 6, it can be observed that the solution quality is further improved when increasing the number of colonies. This is expected because the computational resources allocated for the optimization process are more. However, the improvement in the offline performance from 16 colonies to 128 colonies may not worth the computational efforts.

[1]Available from http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

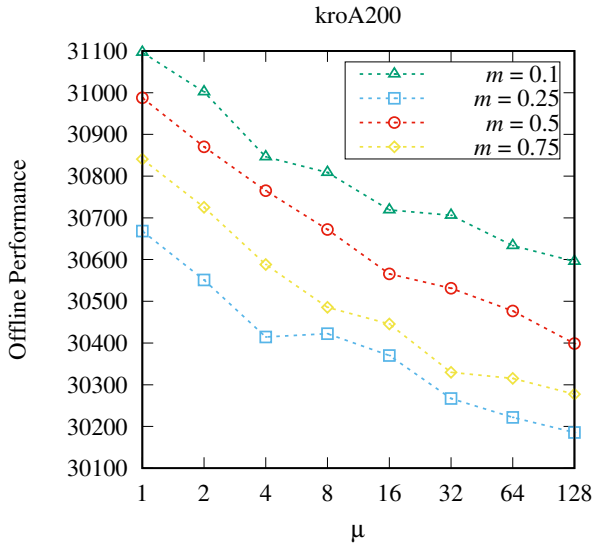


Fig. 4: Offline performance results of PIR with different number of colonies for kroA200 problem instances.

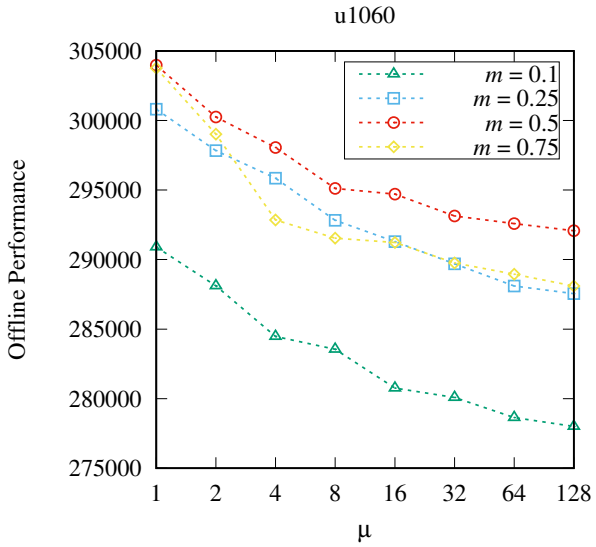


Fig. 6: Offline performance results of PIR with different number of colonies for u1060 problem instance.

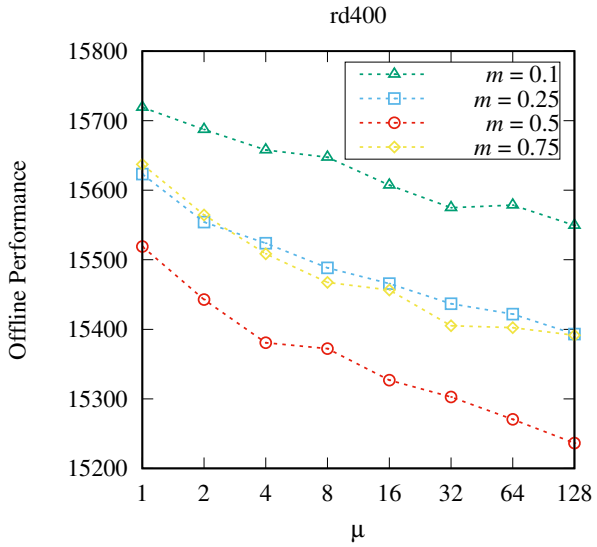


Fig. 5: Offline performance results of PIR with different number of colonies for rd400 problem instance.

C. Effect of Communication

In this set of experiments, twelve distinct exchange strategies are investigated, encompassing four different topologies each with three different μ sizes. The migration schedule is set to automatic (i.e., whenever a new best-so-far ant is discovered to allow the colonies to exchange information). Table II presents the experimental results of the multi-colony ACO algorithms, and statistical comparisons are performed for each colony size μ. Kruskal–Wallis statistical tests are performed, followed by posthoc pairwise comparisons using Wilcoxon rank-sum statistical tests with p-values adjusted by Bonferonni correction.

From Table II it can be observed that PIR is significantly worse than the other three exchange strategies in almost all test cases. When the colonies are exchanging information, in this case the best-so-far ant, the searching is directed towards promising areas of the search space promoting in this way exploitation. As a consequence, a good balance between exploration (promoted by multiple colonies as discussed above) and exploitation is maintained enhancing the searching capabilities of ACO when information is exchanged. Also, in the u1060 cases it can be observed that only when the communication of 16 colonies (i.e., CMPL) exists, the offline performance is better than PIR for all test cases. This shows that as the problem size increases more communicating colonies are required.

When comparing only the three multi-colony ACO algorithms that exchange information it can be observed that: in most dynamic test cases CUBE and CMPL perform better than RING. This can be explained by the size of the neighborhood of the RING topology which is only one. Therefore, the capacity of information exchanged in the RING topology is limited.

D. Effect of Migration Schedule

In the previous set experiments, the migration schedule of the investigated multi-colony ACO algorithms had a varying frequency because it was set to *automatic*. In this set of experiments, fixed frequencies (i.e., *short* and *long*) for the migration schedule are compared with the *automatic* frequency to investigate the effect on the generated diversity which is defined as follows:

$$\bar{T}_{diversity} = \frac{1}{\omega(\omega-1)} \sum_{p=1}^{\omega} \sum_{q \neq p}^{\omega} \left(1 - \frac{CA(p,q)}{n}\right), \quad (8)$$

TABLE II: Experimental results regarding the offline performance of multi-colony ACO algorithms with different exchange strategies. Bold values indicate the best results.

ACOs		DTSP test case			
kroA200, $m \Rightarrow$		0.1	0.25	0.5	0.75
$\mu = 4$,	PIR	30846.32	30414.32	30765.08	30587.98
	RING	30902.39	30443.88	30646.43	30526.60
	CUBE	**30809.07**	30430.57	**30612.58**	30483.73
	CMPL	30863.33	**30388.57**	30644.64	**30428.42**
$\mu = 8$,	PIR	30808.83	30422.27	30671.99	30485.64
	RING	30776.29	30360.00	30571.83	30366.42
	CUBE	**30746**.84	**30288.52**	**30517.00**	**30294.61**
	CMPL	30821.79	30323.71	30538.05	30309.10
$\mu = 16$,	PIR	30718.92	30369.94	30565.86	30445.57
	RING	30693.25	30313.86	30517.29	30289.45
	CUBE	**30688.44**	30296.96	30460.38	**30261.09**
	CMPL	30728.31	**30284.39**	**30407.73**	30274.51
rd400, $m \Rightarrow$		0.1	0.25	0.5	0.75
$\mu = 4$,	PIR	15657.98	15523.75	15380.61	15508.52
	RING	15607.28	**15437.68**	15323.90	15408.93
	CUBE	15620.78	15451.91	**15293.46**	15400.73
	CMPL	**15582.23**	15473.12	15311.94	**15386.24**
$\mu = 8$,	PIR	15647.45	15488.31	15372.17	15467.57
	RING	15581.76	15418.75	15270.87	15385.63
	CUBE	15577.39	15398.65	**15237.10**	15354.23
	CMPL	**15537.28**	**15380.31**	15250.05	**15331.34**
$\mu = 16$,	PIR	15607.40	15465.78	15326.98	15456.58
	RING	15555.59	15405.67	15250.97	15337.96
	CUBE	**15510.29**	15348.02	15222.75	**15283.65**
	CMPL	15519.06	**15340.94**	**15200.47**	15291.62
u1060, $m \Rightarrow$		0.1	0.25	0.5	0.75
$\mu = 4$,	PIR	**284471.60**	295851.51	298059.59	**292848.03**
	RING	284776.60	**294925.61**	**296909.30**	294525.59
	CUBE	285128.91	296382.86	298195.53	294788.96
	CMPL	284602.56	296707.43	297352.74	295199.26
$\mu = 8$,	PIR	283556.08	**292820.87**	**295116.58**	291538.04
	RING	283280.01	294747.28	295497.37	291035.86
	CUBE	**281753.50**	294093.74	295180.04	290949.11
	CMPL	282412.06	294200.76	295516.63	**290901.82**
$\mu = 16$,	PIR	280752.68	291285.87	294700.43	291232.63
	RING	281183.61	293614.26	294456.36	291130.21
	CUBE	280515.02	293514.45	293423.69	291203.54
	CMPL	**279384.70**	**291103.37**	**293302.67**	**290035.24**

where $CA(p, q)$ is the total number of common arcs between the TSP solutions constructed by the p-th and q-th ants and ω is the total number of ants in all colonies (i.e., $\omega = 25\mu$).

The total diversity together with the offline performance against the algorithmic iterations are plotted in Fig. 7(a) and 7(b), respectively, for the CUBE with 8 colonies. Note that for the remaining topologies the observations are similar. From Fig. 7(a), it can be observed that the migration schedule with *short* fixed frequency maintains lower total diversity than the migration schedule with *long* fixed frequency. This effect is expected because with *long* fixed frequency the colonies are given enough time to explore different areas. In contrast, with *short* fixed frequency the colonies are more likely to exploit a similar area in the search space since the best-so-far ant is communicated in early stages of the optimization process.

Furthermore, the *automatic* frequency maintains a diversity level that falls in between of the two fixed frequencies because of its varying nature. However, from Fig. 7(b) it can be observed that the offline performance of the *automatic* frequency is better than both fixed frequencies in most environmental changes. This observation is contradictory with the general strategy when addressing dynamic optimization problems, that is, to increase or maintain the diversity during the optimization process. However, the observations here show that maintaining higher diversity does not always mean better offline performance in dynamic environments.

VI. Conclusions

In this work, we investigate multi-colony ACO algorithms with different exchange strategies in dynamic environments. The exchange strategies are characterized by the number of colonies, their topology, and the migration schedule. The TSP is used as the base problem to generate dynamic test cases in the experiments. The impact that the designed exchange strategies have on the performance of multi-colony ACO algorithms is investigated on a series of DTSPs that are systematically generated. From the experimental results, the following concluding remarks can be drawn. First, multiple parallel colonies enhance the adaptation capabilities of conventional single colony ACO algorithms in dynamic environments. Second, the communication between colonies is essential in multi-colony ACO. Third, exchange strategies in which the colonies communicate with larger number of neighbor colonies have often better performance.

For future work, practical applications with a dynamic environment will be investigated. An interesting application is the agile earth observation satellites scheduling problem [30], [31] which can also be transformed and solved as a TSP [32]. This problem also has several unexpected environmental changes [33], [34], and hence the adaptation capabilities of multi-colony ACO algorithms demonstrated in this work have the potential to be beneficial.

References

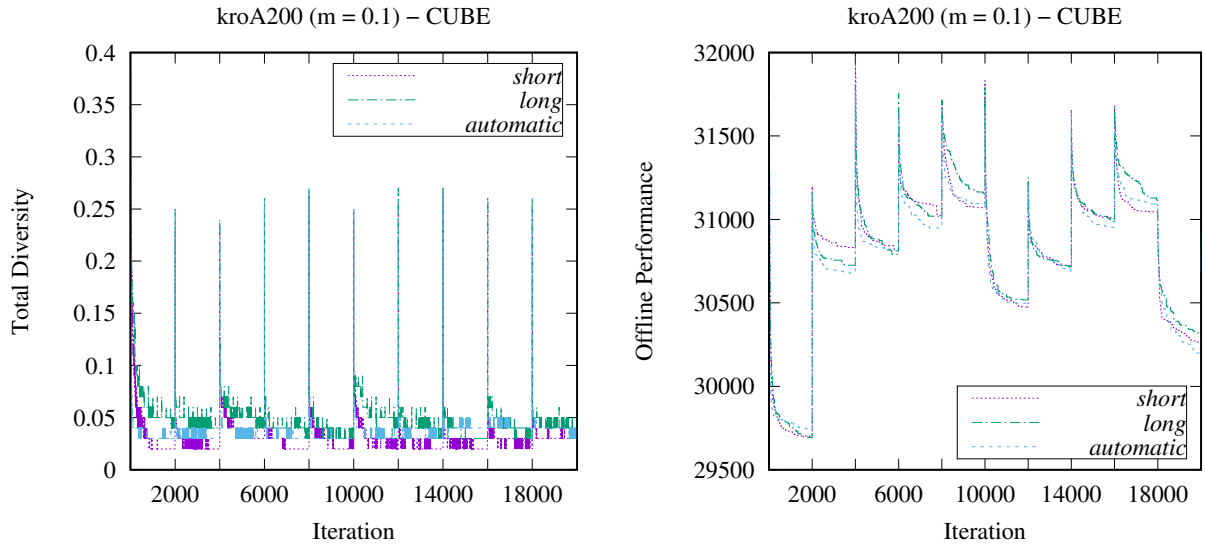
[1] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 26, no. 1, pp. 29–41, Feb 1996.

[2] M. Mavrovouniotis, G. Ellinas, and M. Polycarpou, "Electric vehicle charging scheduling using ant colony system," in *2019 IEEE Congress on Evolutionary Computation (CEC)*, June 2019, pp. 2581–2588.

[3] ——, "A survey of evolutionary continuous dynamic optimization over two decades—Part A," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 4, pp. 609–629, 2021.

[4] D. Yazdani, R. Cheng, D. Yazdani, J. Branke, Y. Jin, and X. Yao, "A survey of evolutionary continuous dynamic optimization over two decades—Part B,"*IEEE Transactions on Evolutionary Computation*, vol. 25, no. 4, pp. 630–650, 2021.

[5] M. Garey and D. Johnson, *Computer and intractability: A guide to the theory of $\mathcal{NP}$-completeness*. San Francisco: Freeman, 1979.



(a) (b)

Fig. 7: Offline performance (a) and total diversity (b) results on `kroA200` with $m = 0.1$ for the `CUBE` topology with 8 colonies.

[6] M. Mavrovouniotis, C. Li, and S. Yang, "A survey of swarm intelligence for dynamic optimization: Algorithms and applications," *Swarm and Evolutionary Computation*, vol. 33, pp. 1–17, April 2017.

[7] D. Angus and T. Hendtlass, "Ant colony optimisation applied to a dynamically changing problem," in *Developments in Applied Artificial Intelligence*, ser. Lecture Notes in Computer Science, T. Hendtlass and M. Ali, Eds., vol. 2358. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 618–627.

[8] M. Guntsch and M. Middendorf, "Pheromone modification strategies for ant algorithms applied to dynamic TSP," in *Applications of Evolutionary Computing*, ser. Lecture Notes in Computer Science, E. J. W. Boers, Ed., vol. 2037. Springer Berlin Heidelberg, 2001, pp. 213–222.

[9] M. Mavrovouniotis and S. Yang, "Interactive and non-interactive hybrid immigrants schemes for ant algorithms in dynamic environments," in *2014 IEEE Congress on Evolutionary Computation (CEC)*, July 2014, pp. 1542–1549.

[10] ——, "An immigrants scheme based on environmental information for ant colony optimization for the dynamic travelling salesman problem," in *Artificial Evolution*, ser. Lecture Notes in Computer Science, J.-K. Hao, P. Legrand, P.and Collet, N. Monmarché, E. Lutton, and M. Schoenauer, Eds., vol. 7401. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 1–12.

[11] ——, "Ant colony optimization with immigrants schemes for the dynamic vehicle routing problem," in *Applications of Evolutionary Computation*, ser. Lecture Notes in Computer Science, C. Di Chio, A. Agapitos, S. Cagnoni, C. Cotta, F. de Vega, G. Di Caro, R. Drechsler, A. Ekárt, A. Esparcia-Alcázar, M. Farooq, W. Langdon, J. Merelo-Guervós, M. Preuss, H. Richter, S. Silva, A. Simões, G. Squillero, E. Tarantino, A. Tettamanzi, J. Togelius, N. Urquhart, A. Uyar, and G. Yannakakis, Eds. Springer Berlin Heidelberg, 2012, vol. 7248, pp. 519–528.

[12] M. Guntsch and M. Middendorf, "Applying population based ACO to dynamic optimization problems," in *Ant Algorithms*, ser. Lecture Notes in Computer Science, M. Dorigo, G. Di Caro, and M. Sampels, Eds., vol. 2463. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 111–122.

[13] M. Mavrovouniotis and S. Yang, "Dynamic vehicle routing: A memetic ant colony optimization approach," in *Automated Scheduling and Planning*, ser. Studies in Computational Intelligence, A. Uyar, E. Ozcan, and N. Urquhart, Eds. Springer Berlin Heidelberg, 2013, vol. 505, pp. 283–301.

[14] M. Mavrovouniotis, F. M. Müller, and S. Yang, "An ant colony optimization based memetic algorithm for the dynamic travelling salesman problem," in *Proceedings of the 2015 Genetic and Evolutionary Computation Conference (GECCO15)*, 2015, pp. 49–56.

[15] M. Mavrovouniotis, S. Yang, and X. Yao, "Multi-colony ant algorithms for the dynamic travelling salesman problem," in *2014 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments (CIDUE)*, Dec 2014, pp. 9–16.

[16] C. Twomey, T. Stützle, M. Dorigo, M. Manfrin, and M. Birattari, "An analysis of communication policies for homogeneous multi-colony ACO algorithms," *Information Sciences*, vol. 180, no. 12, pp. 2390–2404, 2010.

[17] M. Pedemonte, S. Nesmachnow, and H. Cancela, "A survey on parallel ant colony optimization," *Applied Soft Computing*, vol. 11, no. 8, pp. 5181–5197, 2011.

[18] P. González, R. R. Osorio, X. C. Pardo, J. R. Banga, and R. Doallo, "An efficient ant colony optimization framework for hpc environments," *Applied Soft Computing*, vol. 114, p. 108058, 2022.

[19] B. A. M. Menezes, H. Kuchen, H. A. Amorim Neto, and F. B. de Lima Neto, "Parallelization strategies for GPU-based ant colony optimization solving the traveling salesman problem," in *2019 IEEE Congress on Evolutionary Computation (CEC)*, 2019, pp. 3094–3101.

[20] M. Mavrovouniotis, S. Yang, M. Van, C. Li, and M. Polycarpou, "Ant colony optimization algorithms for dynamic optimization: A case study of the dynamic travelling salesperson problem [research frontier]," *IEEE Computational Intelligence Magazine*, vol. 15, no. 1, pp. 52–63, 2020.

[21] M. Guntsch, M. Middendorf, and H. Schmeck, "An ant colony optimization approach to dynamic TSP," in *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO'01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 860–867.

[22] T. Stützle and H. H. Hoos, "$\mathcal{MAX}$–$\mathcal{MIN}$ ant system," *Future Generation Computer Systems*, vol. 16, no. 8, pp. 889–914, June 2000.

[23] M. Dorigo, G. D. Caro, and L. M. Gambardella, "Ant algorithms for discrete optimization," *Artificial Life*, vol. 5, no. 2, pp. 137–172, 1999.

[24] M. Mavrovouniotis, C. Li, G. Ellinas, and M. Polycarpou, "Parallel ant colony optimization for the electric vehicle routing problem," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2019, pp. 1660–1667.

[25] J.-Y. Li, X.-Y. Deng, Z.-H. Zhan, L. Yu, K. C. Tan, K.-K. Lai, and J. Zhang, "A multipopulation multiobjective ant colony system considering travel and prevention costs for vehicle routing in covid-

19-like epidemics," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 25 062–25 076, 2022.
[26] M. Middendorf, F. Reischle, and H. Schmeck, "Multi colony ant algorithms," *Journal of Heuristics*, vol. 8, no. 3, pp. 305–320, May 2002.
[27] M. Manfrin, M. Birattari, T. Stützle, and M. Dorigo, "Parallel ant colony optimization for the traveling salesman problem," in *Ant Colony Optimization and Swarm Intelligence*, M. Dorigo, L. M. Gambardella, M. Birattari, A. Martinoli, R. Poli, and T. Stützle, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 224–234.
[28] E. Alba and M. Tomassini, "Parallelism and evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 5, pp. 443–462, 2002.
[29] J. Branke and H. Schmeck, "Designing evolutionary algorithms for dynamic optimization problems," in *Advances in Evolutionary Computing*, ser. Natural Computing Series, A. Ghosh and S. Tsutsui, Eds. Springer Berlin Heidelberg, 2003, pp. 239–262.
[30] X. Chen, G. Reinelt, G. Dai, and M. Wang, "Priority-based and conflict-avoidance heuristics for multi-satellite scheduling," *Applied Soft Computing*, vol. 69, pp. 177–191, 2018.
[31] X. Wang, G. Wu, L. Xing, and W. Pedrycz, "Agile earth observation satellite scheduling over 20 years: Formulations, methods, and future directions," *IEEE Systems Journal*, vol. 15, no. 3, pp. 3881–3892, 2021.
[32] B. Du, S. Li, Y. She, W. Li, H. Liao, and H. Wang, "Area targets observation mission planning of agile satellite considering the drift angle constraint," *Journal of Astronomical Telescopes, Instruments, and Systems*, vol. 4, no. 4, p. 047002, 2018.
[33] G. Povéda, O. Regnier-Coudert, F. Teichteil-Königsbuch, G. Dupont, A. Arnold, J. Guerra, and M. Picard, "Evolutionary approaches to dynamic earth observation satellites mission planning under uncertainty," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1302–1310.
[34] L. He, X.-L. Liu, Y.-W. Chen, L.-N. Xing, and K. Liu, "Hierarchical scheduling for real-time agile satellite task scheduling in a dynamic environment," *Advances in Space Research*, vol. 63, no. 2, pp. 897–912, 2019.