International Journal of Artificial Intelligence,
ISSN 0974-0635; Int. J. Artif. Intell.
2025, Volume 23, Issue Number 2

Exploring Enhanced Swarm Intelligence algorithms for Efficient Cyber Foraging on Transient Mobile Clouds

T. F. Ndambomve[1], F. Mokom[2] and **K. D. Taiwe[3]**

[1]School of Information Technology, Catholic University Institute of Buea,
P.O. Box 563 Buea, CAMEROON; LaRI Lab, University of Maroua,
P.O. Box 814, CAMEROON;
Email: tiakofani@cuib-cameroon.org

[2]School of Information Technology, Catholic University Institute of Buea,
P.O. Box 563 Buea, CAMEROON; IEEE Computational Intelligence Society;
Association for Computing Machinery (ACM),
Email: fmokom@cuib-cameroon.org

[3]LaRI Lab, University of Maroua, P.O. Box 814 Maroua, CAMEROON;
Email: dtaiwe@yahoo.fr

ABSTRACT

In the evolving landscape of mobile computing, Transient Mobile Clouds (TMCs) leverage nearby mobile devices to form ad hoc computational networks, offering low-latency, cost-effective alternatives to centralized cloud systems. However, efficient task offloading in TMCs mobility faces unique challenges in maintaining resource distribution, minimizing energy consumption, and accommodating dynamic device participation, energy constraints, and fluctuating network conditions. Traditional Swarm Intelligence (SI) algorithms like Particle Swarm Optimization (PSO) and Discrete Artificial Bee Colony (DABC) struggle to balance energy consumption, latency, and load balancing in such environments. This study introduces two advanced task allocation algorithms: Enhanced Multi-Objective Particle Swarm Optimization (EMO-PSO) and Enhanced Multi-Objective Discrete Artificial Bee Colony (EMO-DABC), tailored for Transient Mobile Cloud (TMC) systems. These algorithms have adaptive cyber foraging mechanisms that improve computational load balance, processor utilization, throughput and energy efficiency by dynamically adjusting parameters based on device energy thresholds, task priority and neighbourhood clustering. Simulation results show substantial reductions in energy consumption (up to 18% for EMO-PSO and 15% for EMO-DABC), task completion times (9% improvement over standard DABC), and enhanced load balancing (up to 20% improvement for EMO-DABC). These improvements demonstrate the effectiveness of the proposed algorithms in optimizing resource allocation in dynamic TMC environments.

Keywords: Cyber Foraging, Discrete Artificial Bee Colony, Multi-Objective Optimization, Particle Swarm Optimization, Transient Mobile Clouds.

Mathematics Subject Classification: 68T42, 68T20

Computing Classification System: Computing methodologies/ Artificial intelligence/ Search methodologies / Heuristic function construction

1. INTRODUCTION

In today's rapidly evolving mobile device landscape, there is a pressing need to establish robust frameworks that enable these devices to efficiently handle application computation and completion.

According to Taylor P., (2023), it can be inferred that the rate of mobile device usage has increased over the decades. This growth is driven by the proliferation of resource-intensive applications such as real-time gaming, facial recognition, and optical music recognition. With an overall growth rate of 29.8% per year noted in (Taylor P., 2023), in 2025, the number of mobile phone users worldwide is projected to reach 7.49 billion; this is practically as much as the population of the whole planet. However, among the numerous mobile applications available, there are around one in four mobile applications that are downloaded once and never used again. These applications are primarily discarded due to their increasing computational demands, which often exceed the capabilities of individual devices. Thus, even if the device is able to process its operating system, the remaining resources are finding it difficult to maintain these intensive applications. This forces the users to rely on costly remote cloud services. Remote cloud services rely on large consolidated datacentres that provide computation and storage. However, these datacentres represent a centralization node that with serious shortcomings. This can constitute a point of failure in times of natural disasters, war or when the datacentre's geographical location is often-times out of reach for the customer. The authors (Wei, Z. et al., 2022; Wu and Nath, 2020; Mathur and Sharma, 2023) discuss many services and applications in certain scenarios which ascertain using remote clouds as infeasible. These are often services where applications are solely dependent on the time and place in which the applications need to be executed. Such place-bound activities are best addressed at the user level. This class of cloud computing that deals with formation and deployment at the level of the user's smartphone is known as a Transient Mobile Cloud (TMC): a collection of smartphones in close proximity of the users constituted as a cloud. Therefore, it enables a low cost and a low latency environment that will form a potentially significant computational pool. Hence, Point-to-Point (P2P) networks and Mobile Ad-hoc networks (MANETs) are predominantly considered. In MANETs (Chen et al, 2015, Wei, Z. et al., 2022; Wu and Nath, 2020), users can form a wireless network at any place and at any time without any centralized administration. Moreover, since devices rely on direct communication between the nodes, they do not require cellular or access points to communicate. Khatoon et al, 2025, presented an energy efficient dynamic load balanced clustering for MANET, emphasizing on cluster formation by selecting suitable nodes to act as cluster head which coordinate cluster members over time. They used a technique for dynamic self-adjustment of role of CHs on the basis of fitness factor of nodes so that the lifetime of CHs can be increased which ultimately improves the network lifetime.

The rapid proliferation of mobile and cloud computing has spurred significant interest in optimizing resource utilization in TMC environments through the use of cyber foraging. Cyber foraging is a computing technique where resource-constrained devices, like smartphones or IoT sensors, offload demanding tasks to more powerful devices (or surrogates) to save energy and improve performance (Penner et al., 2017). In TMCs, devices connect as Smartphone Ad-hoc Networks (SPANs) to form a temporary cloud and act as a pool of resources, offering computation power to each other. Here, the cyber foraging process leverages available resources in nearby devices present in the ad-hoc network of the dynamic cloud infrastructure to perform computation-heavy tasks, enabling devices with limited capabilities to handle complex applications that would otherwise strain their resources. Consequently, mobile devices collaborate on tasks' execution without needing a centralized internet connection,

offering low-cost and adaptable computing environments, even in areas with limited or unstable connectivity.

Cyber foraging frameworks for TMCs often include context-aware mechanisms to dynamically select the most appropriate device or surrogate based on proximity, available resources, energy levels, and network quality, in order to ensure a satisfying Quality of Service (QoS) level. In these settings, QoS requirements are critical, necessitating high levels of execution speed, dependability, and service availability. However, as TMCs grow, they encounter challenges in ensuring:

- Latency and Network Reliability: Effective cyber foraging depends on low-latency, reliable network connections. In TMCs, where connectivity can be intermittent, maintaining a steady connection for offloading can be challenging.
- Resource Availability and Management: Devices in a TMC have limited or fluctuating resources (e.g.: battery and processing power). Managing and balancing these resources dynamically is crucial for maintaining efficient foraging operations.
- Fault Tolerance: Devices in ad-hoc networks join or leave frequently, which disrupts ongoing offloaded tasks. Ensuring fault tolerance and reassigning tasks when a surrogate device becomes unavailable is necessary for robust cyber foraging.

These challenges present various objectives to tackle, and traditional optimization techniques for clouds tasks scheduling do not adequately address these unique requirements of TMCs, where device energy levels, task urgency, load balancing and system dynamism require adaptive strategies. Also, some current Swarm intelligence (SI) strategies for task allocation, like those based on Particle Swarm Optimization (PSO) and Discrete Artificial Bee Colony (DABC) algorithms, have shown to be suited for TMCs because of their decentralized nature, their inherent adaptability and efficiency in multi-objective optimization. However, existing solutions often result in high energy consumption, restricting their feasibility for prolonged use in mobile environments. Therefore, efficiently allocating computational tasks to achieve satisfactory QoS amidst fluctuating power levels, network instability, constant mobility and resource heterogeneity remains a core research focus.

Building upon prior works in cyber foraging on TMCs using SI approaches, this study aims to develop and evaluate enhanced SI algorithms to optimize task offloading in TMCs, addressing challenges related to load balancing, task completion time, and energy efficiency. They provide adaptive mechanisms based on task priority, device energy thresholds and neighborhood clustering, showcasing the effectiveness of these algorithms in balancing energy consumption, task completion time, processor utilization and load distribution in simulated TMC environments. The subsequent sections are as follows:

- Background and Related Work: Discusses the challenges of TMCs and existing solutions.
- System Model: Presents task allocation in TMCs as a mathematical model.
- Proposed Algorithms: Introduces EMO-PSO and EMO-DABC, detailing their enhancements over traditional methods.
- Simulation and Results: Evaluates the performance of the proposed algorithms through simulations.
- Conclusion: Reviews the findings and possible future research directions.

2. BACKGROUND AND RELATED WORK

As noted earlier, the management of Quality of Service (QoS) in cyber foraging on a TMC presents a complex set of objectives, including resource allocation efficiency, power consumption optimization, processor usage, task execution time, load balancing, throughput, fault tolerance, among many others. In addressing these needs, Multi-Objective Optimization is often applied along with Swarm Intelligence strategies.

2.1. Multi-Objective Optimization (MOO)

Optimization methods that deal with two or three objectives are called multi-objective optimization (MOO), whereas if the number of objectives is greater than three, they are called many-objective optimization (Dai et al., 2024). Specifically, MOO is applied to problems that involve multiple objectives and is concerned with mathematical optimization problems where two or three contrasting objectives need to be evaluated simultaneously (Wang et al., 2020; Ramezani et al., 2020). As critical as task allocation is for cyber foraging in TMCs, a good task scheduler must provide an acceptable trade-off between several conflicting goals and requirements to provide the most suitable QoS. In this work, we address task allocation in the TMC environments as a MOO problem where for every task, the execution time, the energy consumed and the processor utilization are conflicting objectives. These are the principal objectives to meet when executing tasks on mobile devices, and they are conflicting objectives because minimizing execution time usually means processing tasks as quickly as possible, which requires high processing power, and leads to increased energy consumption. This is done in order to orchestrate the suitable assignment of the submitted tasks to the set of mobile devices. Thus, this task allocation becomes a MOO problem.

According to Reyes-Sierra et al. (2006), MOO is a collection of decision components with certain restrictions that maximizes the objective function. MOO presents a possible set of solutions, which are evaluated using trade-offs among several objective functions. A selection strategy is then applied to choose an acceptable solution. The MOO problem, according to (Ghasemi-Falavarjani et al., 2015), can be mathematically defined as follows:

$$\mathbf{M}\text{in } \mathbf{F}(\mathbf{x}) = (\mathbf{F}1(\mathbf{x}); \mathbf{F}2(\mathbf{x}); \ldots ; \mathbf{F}k(\mathbf{x})) \qquad (1)$$

where: k is the number of objectives, F(x) is the objective function of the solution x, F1(x) is the objective function for objective 1 for solution x, Fk(x) is the objective function for objective k for solution x, and k<=3. Specifically, several methods have been proposed in the literature for dealing with multi-objective optimization, and can be divided into three categories: domination, decomposition and ranking approaches (Zhang et al., 2012). In this work, we adopted the ranking approach for Task allocation since the objectives considered are atomic and have the same weights.

2.2. Swarm Intelligence in TMCs

Swarm Intelligence (SI) has gained traction in cloud computing due to its robust and flexible approach to solving optimization problems. SI-based methods, inspired by the collective behaviour of social organisms, have been adapted to dynamically allocate tasks in cloud computing environments. Their decentralized structure aligns well with the dynamic and heterogeneous nature of TMCs (Valarmathi et

al.,2019; Zhang et al., 2024). While Particle Swarm Optimization (PSO) and Discrete Artificial Bee Colony (DABC) are widely studied, a variety of SI algorithms, such as Ant Colony Optimization (ACO) and Genetic Algorithms (GA), also offer substantial benefits. For instance, ACO has been shown to provide effective load balancing in complex networks, as demonstrated by Gabi et al. (2021) in high-density mobile cloud environments, but neglecting TMC-specific challenges like device churn and energy volatility. Genetic Algorithms, widely recognized for their solution diversity, have been effective in achieving global optimization but are often more computationally intensive. In particular, by mimicking the movements of a flock of birds during their food hunt, PSO has demonstrated efficacy of recognizing near-optimal solutions (Deb et al., 2002). However, each algorithm presents unique trade-offs in terms of computational complexity, speed, and adaptability, particularly in high-fluctuation environments like TMCs. Unlike traditional centralized optimization approaches, SI algorithms use local interactions among agents to achieve global optimization, thus enabling scalable and flexible solutions suitable for TMCs. Furthermore, SI algorithms comprise some basic behavior rules that simplify the solution of Multi-Objective Optimization (MOO) problems, such as those pertaining to TMC systems in terms of task execution time, energy consumption, and processor power. In particular, algorithms like PSO and DABC have demonstrated efficiency in handling MOO problems, which involve balancing factors such as energy consumption, task execution time, and processor utilization.

2.2.1. Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) is one of the most widely implemented SI algorithms in task scheduling and resource allocation due to its simplicity and efficiency. PSO operates by iteratively updating particle velocities and positions to achieve optimal task allocation, with minimal computational overhead. For example, Ramezani et al. (2020) demonstrated that Multi-Objective Particle Swarm Optimization (MO-PSO) effectively minimizes task delivery time and power consumption. Another study by Zhong et al. (2019) employed a Greedy PSO scheduling method that achieved rapid convergence, high global search ability, and balanced workload allocation. PSO is often preferred in time-sensitive environments where low computational cost is prioritized (Dai et al., 2024). Also, the time complexity of PSO primarily depends on the number of particles (N) and the number of dimensions (D) in the problem space. Therefore, the time complexity per iteration is typically O(N·D). However, its limitations in maintaining solution diversity across complex and contrasting objectives warrant further exploration, particularly when applied to high-density transient cloud networks.

2.2.2. Discrete Artificial Bee Colony (DABC)

The Artificial Bee Colony (ABC) algorithm, inspired by the foraging behaviour of honey bees, has also shown potential in task scheduling due to its fast convergence and robustness. This approach, initially proposed by Karaboga et al. in 2005 as a solution for continuous optimization problems, has been adapted for discrete optimization tasks, resulting in the Discrete Artificial Bee Colony (DABC) algorithm (Man and Yang, 2020; Keshanchi et al., 2020). In TMC environments, DABC has proven effective for multi-objective task scheduling, with notable improvements in energy efficiency, cost reduction, and

workload balancing. Studies by Ebadifard et al. (2018) reported a 22% increase in resource utilization over standard PSO approaches by leveraging DABC's load-balancing capabilities, however for PSO, the convergence speed was faster, and it was more effective in performing more tasks. In each iteration, employed bees explore solutions around their current position, and onlooker bees choose solutions based on the information exchanged by employed bees. The complexity per iteration is O(N·D), where N is the number of bees in use and D is the dimensionality of the problem. However, DABC is computationally intensive, particularly with large populations, and further optimizations are required to meet real-time processing needs in resource-limited mobile devices.

2.2.3. Comparative Approaches of PSO and DABC with Multi-Objectivity

Most existing task allocation algorithms based on Swarm Intelligence are often designed primarily for fixed or low-mobility cloud networks, and lead to suboptimal performance when applied to TMCs due to their reliance on static configurations and disregard for mobile energy constraints. Amongst them, variations of PSO are more frequently used for the purpose of multi-objective task allocation compared to those based on ABC.

While Mathur et al. (2023) explored a hybrid task scheduling method using Gaussian-based MOPSO and Bacterial Foraging Optimization, improving energy efficiency and task makespan in mobile cloud environments, but their work lacks mechanisms to handle intermittent connectivity. Xiaoyan et al. (2024) discussed the application of MOPSO for optimizing task scheduling under cloud resource constraints, providing insights into balancing task latency and resource usage for Mobile Edge Clouds. Kouka et al. (2020) presented a Dynamic Multi-Objective Particle Swarm Optimization (DMOPSO) for Cloud and Edge Computing, including variations that incorporate cooperative agents, addressing dynamic conditions in mobile edge computing. These variations manage resource allocation and latency while maintaining a balance between convergence and diversity in transient mobile environments, making them useful in unstable, transient cloud networks. Cui et al. (2022) explored a MOPSO variant that dynamically adjusts population size based on available resources. This strategy helps prevent local optimum traps, enhancing solution diversity and reliability; a key consideration for resource-constrained mobile clouds where device connections are often intermittent. Ramezani et al. (2020) comprehensively considered minimizing task delivery time, task execution cost, power consumption and task queue length and used multi-objective particle swarm optimization (MO-PSO) and Multi-Objective Genetic Algorithm (MOGA) for task scheduling in a cloud centre. Their simulation results show that this method can improve the quality of service and reduce costs.

Shuai Man et al. (2020) analysed the application of the DABC algorithm in the cloud task scheduling, and adapted to prioritize energy efficiency while ensuring task deadlines are met, a common requirement in mobile clouds. This approach benefits mobile clouds by reducing unnecessary energy consumption and optimizing task offloading decisions dynamically as devices connect or disconnect from the network. Dai et al. (2023) proposed a multi-objective multi-picking-robot task allocation on the DABC algorithm to optimize the cost and efficiency of task execution in agricultural smart farms by enhancing the exploitation and exploration of the algorithm. In (Xue et al., 2018; Gupta et al., 2021), authors have proposed a honey bee and improvement detection operator-based load balancing

algorithm named Hyper-heuristic. The proposed approach has the capability to distribute the cloud-based workload among the devices with minimized makespan time. However, the Hyper-heuristic algorithm does not consider throughput and processor utilization as scheduling objectives.

The PSO-based task scheduling methods in cloud computing have been examined by Alkayal et al. (2017). The authors have classified the current PSO-based research work on the basis of the number of objectives that need to be optimized. This categorization of PSO algorithms includes a single objective or multiple objectives. The majority of researchers have used updated or basic PSO to improve the quality of the solution. The author came to the conclusion that additional attention and development were needed to meet the scheduling and QoS standards (i.e., resource use, makespan, and throughput, etc.). Also, inertia weight has not been taken into account.

Authors in (Huang et al., 2019; Beegom and Rajasree, 2019; Kumar et al., 2016) have presented a review of different inertia weight strategies used by various researchers in their work. The author has classified these inertia weights into three groups which include time-varying, constant, and adaptive inertia weights strategies. The scheduling objective of this approach is the average makespan. However, throughput and processor utilization are not considered.

The in-depth investigation of the related literature shows that majority of the existing task scheduling approaches based on PSO or DABC provide insights into algorithmic innovations for addressing the challenges of cloud environments, with a focus on resource optimization and performance stability. Yet, they do not address unique structural specifications of TMCs. Moreover, in order to balance the local and global particle search, the inertia weight approach is a crucial control parameter for PSO-based algorithms. However, the majority of existing scheduling methods either did not specifically address this parameter or employed a fixed value for the inertia weight. Similarly, the majority of the existing approaches did not consider a makespan, processor utilization, and throughput as scheduling objectives.

In prior work (Ndambomve et al. 2021), we discussed the limitations of existing TMC implementations, including single-hop dependency, limited battery power, varying device capabilities, and the frequent changes in network topology typical of smartphone ad-hoc networks, that limited the in-time execution of code. To this effect, we proposed a dynamic application partitioning and offloading framework to enhance the capabilities of TMCs using mobile agents, built on a multi-hop mesh network. Furthermore, in a subsequent work (Ndambomve et al., 2024), we addressed the need for adaptive task scheduling in dynamic TMC networks and proposed an approach for energy-efficient task allocation, leveraging a modified Kuhn-Munkres algorithm tailored to dynamic multi-hop networks, to optimally distribute tasks across devices using mobile agents. However, this work lacked emphasis on multi-objective optimization and swarm intelligence strategies to better respond to fluctuating device availability and energy constraints, a gap our present work aims to address.

To overcome the limitations cited above, we consider the TMC framework we presented in our previous work, and we propose the Enhanced Multi-Objective Particle Swarm Optimization (EMO-PSO) and the Enhanced Multi-Objective Discrete Artificial Bee Colony (EMO-DABC) comparatively. We evaluate these strategies against the standard PSO-based and DABC-based task scheduling algorithms, when applied to the TMC.

The EMO-PSO algorithm in TMC incorporates adaptive inertia weight and learning factors, targeting energy efficiency and computational load balance. Here, we have the following considerations:

- Inertia Weight Adjustment: Instead of static or linearly decreasing inertia, use an adaptive approach based on device energy levels. Devices with higher energy can take on more computational tasks, while those with lower energy levels should minimize computational load.
- Dynamic Neighborhood for Particle Interaction: Utilize a clustered neighborhood model where devices within proximity form neighborhoods. Given how frequently devices join and depart the network, this strategy fits in nicely with TMC. The local search within clustered groups can accelerate convergence and accommodate device mobility.
- Fitness Evaluation Based on Energy and proximity: Fitness calculations should incorporate energy consumption, expected task completion time and resource utilization, and optimized for minimizing latency and maximizing throughput.

This adaptive approach matches the TMC environment's transient and decentralized nature. By dynamically adjusting the inertia weight based on device energy levels, EMO-PSO can better manage task allocations without depleting device resources too quickly. Moreover, a neighborhood-based interaction aligns with the mesh structure of TMC, enhancing stability.

The EMO-DABC focus is on energy management, resource prioritization, which collectively address the intermittent connectivity and high device variability in TMC. Here, we have the following considerations:

- Energy Threshold for Scout Bee Behavior: Introduce an energy-based threshold for scout bees, where devices below a certain energy level avoid participating in task discovery. This reduces strain on low-energy devices and avoids unnecessary task allocations.
- Priority-Based Foraging: Prioritize tasks based on time sensitivity and resource availability, helping direct tasks to devices best suited for quick response, thereby reducing task delay.

Since TMC devices frequently enter and exit the network, scout and forager behaviors benefit from incorporating energy and priority metrics. This focus prevents task allocation to low-energy devices, improving the swarm's overall task-handling efficiency.

3. SYSTEM MODEL

Based on the TMC model presented in our prior work (Ndambomve et al., 2021 and Ndambomve et al., 2024), we consider the following terms for this study.

Given a set of tasks $T=\{T_1,T_2,...,T_n\}$ and a pool of mobile devices $D=\{D_1,D_2,...,D_m\}$, we seek to minimize both Task Energy Consumption (TEC) and Expected Completion Time (ECT) while maximizing devices' Processor Usage (PU). These metrics were chosen because they directly align with the study's objectives of optimizing energy efficiency, task completion time, and resource utilization in TMCs. These metrics provide a comprehensive evaluation of the algorithms' performance in dynamic and resource-constrained environments.

Task Energy Consumption (TEC):

The energy consumption for a device Di executing a task Tj in the TMC is:

$$TEC_{ij}=\alpha \cdot C_{i,j}+\beta \cdot P_{comm} \quad (2)$$

where Ci,j is the computational cost, Pcomm is the communication cost, and α, β are constants. The constants α for computation and β for communication are derived from empirical measurements in prior TMC studies (Penner et al., 2017; Wei et al., 2022). The formulations for the computational cost and communication cost have been elaborated in our prior work (Ndambomve et al., 2024).

The total energy consumption across all tasks and devices is:

$$TEC = \sum_{i=1}^{m} \sum_{j=1}^{n} TEC_{ij}.x_{ij} \quad (3)$$

where $x_{ij}=1$ if task T_i is allocated to device D_j, and 0 otherwise.

Expected Completion Time (ECT):

The time used by a device Di executing a task Tj is:

$$ECT_{ij} = TEC_{ij}/P_i \quad (4)$$

where Pi is the computing power of Di.

The total Expected Completion Time for all the tasks from all devices is calculated as the sum of the execution times for all assigned tasks:

$$\text{ECT} = \sum_{i=1}^{n} \sum_{j=1}^{m} (ECT_{ij}.x_{ij}). \quad (5)$$

Processor Utilization (PU):

Processor utilization for each device Di is the ratio of cumulative time spent executing tasks to the available processing time:

$$PU_i = \sum_{i=1}^{n} ECT_{ij} \cdot x_{ij} \, /T_{total} \, . \quad (6)$$

The overall utilization across all devices is:

$$PU = 1/m \sum_{i=1}^{m} PU_i. \quad (7)$$

The optimization problem is structured to balance these three values as conflicting objectives in the TMC environment. The multi-objective function is therefore defined as follows:

$$Min\ F(x)=(ECT(x),\ TEC(x),\ -PU(x)) \quad (8)$$

where:

- x represents the allocation of tasks to devices,
- $ECT(x)$ is the total expected completion time,
- $TEC(x)$ is the cumulative task energy consumption,
- $PU(x)$ denotes the device's processor utilization, with a negative sign to maximize this metric.

4. CHARACTERIZATION OF THE PROPOSED ALGORITHMS

4.1. Enhanced Multi-Objective Particle Swarm Optimization (EMO-PSO)

4.1.1 Adaptive inertia weight

In EMO-PSO, the inertia weight w plays a crucial role in balancing exploration (searching globally) and exploitation (fine-tuning within local regions). A dynamic, energy-based inertia weight adjustment is

particularly valuable for TMC environments where device energy levels change frequently. The inertia weight w is defined as:

$$w = w\_max - ((w\_max - w\_min) - (E\_initial - E\_current)) / E\_initial \quad (9)$$

where $E_{\_initial}$ and $E_{\_current}$ represent the initial and current energy levels of the device respectively. By reducing w as energy decreases, the particle's movement is constrained to avoid excessive energy use, prioritizing exploitation near local optima.

4.1.2 Velocity and Position Update

Each particle updates its velocity and position based on personal and neighbourhood best as follows:

$$v_i(t) = w.v_i(t-1) + c_1.r_1.\left(p_{besti} - x_i(t)\right) + c_2.r_2.\left(\boldsymbol{g_{neigh\_i}} - x_i(t)\right) \quad (10)$$

$$x_i(t) = x_i(t-1) + v_i(t) \quad (11)$$

where:
$x_i(t)$ is the current position of particle i at iteration t,
$v_i(t)$ is the velocity of particle i at iteration t,
p_{besti} is the best position of particle i,
$\boldsymbol{g_{neigh\_i}}$ is the position of the best value for particle i in a population,
w is the inertia weight,
r_1, r_2 are random numbers in the range [0,1],
C_1, C_2 are the acceleration coefficients.

4.1.3 Neighborhood-Based Clustering in EMO-PSO

To adapt to device mobility in TMC, EMO-PSO groups particles into neighbourhoods based on proximity or signal strength. Each particle adjusts its velocity according to the best position found in its neighbourhood, enhancing task allocation efficiency. Each particle i has a neighbourhood N_i, defined by a radius R or by a specific number of nearest particles. Particles within this radius or nearest neighbours are included in N_i, forming a localized group around particle i. Each particle i incorporates the neighbourhood fitness F_{neigh} as the average fitness among all particles in N_i:

$$F_{neigh} = 1/|N| \sum_{j \in Ni} F_j \quad (12)$$

where |N| is the number of particles in the neighbourhood (ie. a given radius) and F_j is the fitness of each particle j in the neighbourhood.

To select the neighborhood best position $\boldsymbol{g_{neigh\_i}}$ using F_{neigh}, we choose the position of the particle in N_i with the fitness closest to F_{neigh} (i.e., the particle whose fitness is nearest to the neighborhood's average fitness). Formally, if particle k has F_k such that:

$$F_k = min_{j \in Ni} |F_j - F_{neigh}| \quad (13)$$

Then $\boldsymbol{g_{neigh\_i}} = x_k$, where x_k is the position of particle k.

This clustering makes the EMO-PSO more effective in handling the dynamic, decentralized structure of environments like TMCs through:

- Localized Adaptability: each particle can better adapt to the local landscape, which is particularly useful in dynamic environments like TMCs.
- Improved Robustness: Relying on the neighbourhood fitness instead of individual particle fitness avoids over-dependence on a single particle, making the algorithm more robust to neighbourhood changes.
- Balanced Search and Exploitation: Using neighbourhood fitness encourages a balance between exploration and exploitation, as particles are influenced by an aggregate measure of fitness, reducing the risk of premature bb convergence to a single local optimum.

4.2. Enhanced Multi-Objective Discrete Artificial Bee Colony (EMO-DABC)

4.2.1. Task Allocation and Priority

Tasks are prioritized based on urgency and energy consumption. The priority of a task *j* is calculated as:

$$\mathbf{Priority(j)} = \mathbf{1}/(\boldsymbol{TEC_{i,j}} \times \boldsymbol{T_{latency}}) \qquad \textbf{(14)}$$

where $T_{latency}$ is the expected task completion time alongside the communication delay. Tasks with higher priorities (smaller values) are allocated first to higher-energy forager bees, allowing for more effective energy distribution across devices.

Adaptive Role Switching

Foragers switch to scout roles when their energy E_i on a device drops below a threshold, optimizing energy usage across the swarm. This is governed by:

$$Switch_{forager \rightarrow scout} = \begin{cases} True & if\ E_i < E_{threshold} \\ False & otherwise \end{cases} \textbf{(15)}$$

The priority and switch functions are vital in making EMO-DABC suitable for the dynamic, energy-constrained, and decentralized environment of TMCs. By enabling adaptive task allocation based on urgency and energy availability, these functions ensure that EMO-DABC operates efficiently, conserves energy, maintains consistent performance even as the network and device conditions fluctuate and prolongs the operation of each device within the TMC.

4.3. Evaluating Multiple Objectives with the ranking approach

In this work, we use the ranking approach proposed by (Wang et al., 2020). We consider a FITNESSVALUE function which is invoked to evaluate the objectives. In this function, the ranking strategy is applied to evaluate the solutions in each iteration. Each objective is represented by a two-dimensional matrix. The procedure FITNESSVALUE as shown in Algorithm 1 starts by sorting devices based on ECT (Expected Completion Time), TEC (Task's Energy Consumption), and PU (Processor Utilization) and storing each result in a separate matrix (lines 4 to 6). The loop from lines 7 to 9 computes the rank for each task in each device d by accumulating the rank of the three matrices. Then the rank is sorted in an ascending order. Finally, the rank is set for the task based on the first element in each matrix, which includes the smallest value.

Algorithm 1: FITNESSVALUE with ranking strategy

```
Procedure FITNESSVALUE (PU, ECT, TEC)
1.  Input: PU, ECT, TEC
2.  Output: Rank (0) representing the particle with the smallest rank 1
3.  INITIALIZE (Rank)
4.  V1 ← SORT(ECT)              // sort ascending based on ECT
5.  V2 ← SORT(TEC)              // sort ascending based on TEC
6.  V3 ← SORT(PU)               // sort descending based on PU
7.  for each device d do        //iterate through all DEVICEs
8.      Rank(d) ← V1(d) +V2(d) +V3(d)
9.  End for
10. SORT(Rank)                  // sort Rank ascending based on rank
11. return Rank(0)              // return smallest particle in index 0
12. end procedure
```

The ranking approach in Algorithm 1 evaluates solutions based on three conflicting objectives: Expected Completion Time (ECT), Task Energy Consumption (TEC), and Processor Utilization (PU). Conflicts among

these objectives are managed by determining ranks to each task-device pair and selecting the solution with the lowest cumulative rank. For example, if a task has low ECT but high TEC, it may still be selected if its overall rank (considering all objectives) is the lowest. This ensures a balanced trade-off between energy efficiency, task completion time, and resource utilization.

To illustrate this approach, suppose that we have eight tasks with five devices and we compute the three objectives for each solution. The results are shown in Tables 1, 2, and 3. In more detail, for ECT and TEC, we rank the devices based on the least values by sorting the devices processing in an ascending order. For the device processing usage, we sort the devices in a descending order because our aim is to maximize this objective. Then, we calculate the rank of the three objectives by summing the ranks for each objective to select the device with lowest rank value for each task as shown in Table 4. All shaded cells in the Tables (1-4) represent the best solution, i.e. the best device for each task.

4.4. The Role of Mobile Agents in Enabling Robust, Efficient TMC Algorithms

Mobile agents, as described in our previous work (Ndambomve et al., 2021), offer unique advantages that can significantly enhance the effectiveness of the EMO-PSO and EMO-DABC algorithms in TMCs built on Smartphone Ad-hoc Networks (SPAN). In that article, we proposed a dynamic application partitioning and offloading framework to enhance the capabilities of TMCs using mobile agents. Here, we explore how these mobile agents assist in task distribution, fault tolerance, dynamic adaptation, and reducing network load in the context of the proposed algorithms.

4.4.1. Task Distribution and Dynamic Adaptation

In TMC environments, task allocation must continuously adapt as devices enter or exit the network due to mobility. Introducing mobile agents facilitate adaptive task distribution for both EMO-PSO and EMO-DABC.

For EMO-PSO, mobile agents represent particles in the swarm and carry task allocation information across the network. As a device moves, its agents retain its status, including its personal and neighbourhood bests,

Table 1: Task's Energy Consumption (TEC)

Task	D1	D2	D3	D4	D5
T1	12	14	10	17	12
T2	15	11	13	12	10
T3	12	14	10	10	8
T4	14	18	16	22	19
T5	18	12	19	17	15
T6	18	16	19	15	14
T7	12	18	22	20	19
T8	24	20	18	14	17

Rank values after sorting the value of TEC

Task	D1	D2	D3	D4	D5
T1	2	4	1	5	3
T2	5	2	4	3	1
T3	4	5	2	3	1
T4	1	3	2	5	4
T5	4	1	5	3	2
T6	4	3	5	2	1
T7	1	2	5	4	3
T8	5	4	3	1	2

Table 2: Expected Completion Time (ECT)

Task	D1	D2	D3	D4	D5
T1	21	24	30	22	25
T2	24	21	19	25	27
T3	18	21	21	26	16
T4	25	20	24	19	21
T5	30	25	23	22	26
T6	32	30	21	25	28
T7	19	17	18	22	23
T8	12	19	21	17	20

Rank values after sorting the value of ECT

Task	D1	D2	D3	D4	D5
T1	1	3	5	2	4
T2	3	2	1	4	5
T3	2	3	4	5	1
T4	5	2	4	1	3
T5	5	3	2	1	4
T6	5	4	1	2	3
T7	3	1	2	4	5
T8	1	3	5	2	4

Table 3: Processor utilization

Task	D1	D2	D3	D4	D5
T1	20	14	12	21	15
T2	21	15	20	24	18
T3	27	18	24	30	25
T4	27	23	21	24	30
T5	29	23	22	28	26
T6	32	26	25	23	22
T7	35	28	21	24	20
T8	21	28	25	23	25

Rank values after sorting the value of PU

Task	D1	D2	D3	D4	D5
T1	2	4	5	1	3
T2	2	5	3	1	4
T3	2	5	4	1	3
T4	2	4	5	3	1
T5	1	4	5	2	3
T6	1	2	3	4	5
T7	1	2	4	3	5
T8	5	1	2	4	3

Table 4: Devices' Rank values

Task	D1	D2	D3	D4	D5	Best
T1	2+1+2=5	4+3+4=11	1+5+5=11	5+2+1=8	3+4+3=10	D1
T2	5+3+2=10	2+2+5=9	4+1+3=8	3+4+1=8	1+5+4=10	D3
T3	4+2+2=8	5+3+5=13	2+4+4=10	3+5+1=9	1+1+3=5	D5
T4	1+5+2=8	3+2+4=9	2+4+5=11	3+5+1=9	4+3+1=8	D1
T5	4+5+1=10	1+3+4=8	5+2+5=12	3+1+2=6	2+4+3=9	D4
T6	4+5+1=10	3+4+2=9	5+1+3=9	2+2+4=8	1+3+5=9	D4
T7	1+3+5=9	2+1+2=5	5+2+4=11	4+4+3=11	3+5+5=13	D2
T8	5+1+5=11	4+3+1=8	3+5+2=10	1+2+4=7	2+4+3=9	D4

effectively enabling seamless continuation of the PSO process across devices. These mobile agents permit the swarm to remain functional even as individual devices move out of range or join new network areas, preserving the optimization process across the TMC. For EMO-DABC, mobile agents act as scout or forager bees, autonomously migrating to devices with available resources. Scouts explore nearby devices and locate high-priority tasks based on real-time device energy levels (foragers). Mobile agents carrying this task allocation information autonomously adapt their roles based on energy thresholds, allowing the DABC process to continue without needing to restart the task search.

The autonomy of mobile agents also supports the algorithms' multi-objective optimization by distributing computational loads based on each device's current energy and workload capacity, which is essential for real-time resource balancing in transient environments.

4.4.2. Fault Tolerance and Resilience

One major challenge in TMCs is handling network disruptions when devices unexpectedly leave the network. Mobile agents add resilience by enabling fault-tolerant task allocation. The agents monitor device statuses and redistribute tasks to high-energy neighbors, ensuring continuity and minimizing task loss. This approach is particularly effective in high-density networks with frequent device mobility.

If a device is exiting the network or is depleted, mobile agents detect the signal as presented in Ndambomve et al., (2024), run the Send_over procedure and reassign tasks to other available devices. In EMO-PSO, agents update neighbourhood bests by selecting new nearby devices, ensuring that the algorithm adapts to changes without stalling. For EMO-DABC, agents shift tasks initially assigned to a departed forager to a new high-energy device, maintaining continuity in task processing. This approach prevents task loss and reduces the need to restart the task allocation process, making the TMC more resilient in the face of node failures or disconnections. The fault-tolerant nature of mobile agents thus supports both EMO-PSO and EMO-DABC, making these algorithms robust against the transient nature of TMC nodes.

4.4.3. Reduced Network Load and Optimized Communication

One of the benefits of mobile agents in Smartphone Ad-hoc Network (SPAN) environments is their ability to minimize communication overhead, which is crucial for TMCs with limited bandwidth and battery constraints.

Mobile agents move locally within the network to update neighbourhood best positions without requiring all devices to broadcast their information, thereby reducing the overall communication load. By carrying position and fitness updates directly to relevant devices, mobile agents optimize intra-task communication, enhancing the algorithm's scalability and responsiveness. Mobile agents also encapsulate task information, energy levels, and role-switching decisions within self-contained packets. This approach reduces repetitive data transfer by allowing agents to handle task assignment and energy updates on-site, decreasing the need for centralized communication. Scouts can discover new devices with resources by moving within range and updating task priorities locally. By reducing network load, mobile agents help to conserve device energy and enable efficient operation of the EMO-PSO and EMO-DABC algorithms across the TMC, improving their practicality in resource-constrained SPANs.

4.4.4. Autonomous Task Migration and Load Balancing

In TMCs, load balancing across devices is essential to avoid overburdening particular nodes and to ensure that tasks are distributed evenly as network conditions change. Mobile agents enable autonomous task migration, which is particularly beneficial for both algorithms.

Agents representing particles can migrate tasks to less utilized devices as needed, facilitating load balancing without central coordination. If one device has too many tasks due to high fitness, mobile agents can reassign tasks to nearby particles with lower workloads. This autonomous redistribution aligns well with the EMO-PSO's neighbourhood-based clustering, ensuring balanced load distribution in real time.

Mobile agents can use DABC's priority-based foraging to autonomously adjust which devices handle which tasks based on updated energy levels. For example, if a high-priority task initially allocated to one forager needs reassignment due to energy depletion, mobile agents can seamlessly transfer it to another suitable device, maintaining load balance and minimizing delays in task completion.

By enabling mobile agents to autonomously manage task migration, both EMO-PSO and EMO-DABC benefit from better load balancing, enhanced energy efficiency, and the ability to dynamically reallocate tasks based on real-time device status.

4.4.5. Context-Aware Decision-Making and Resource Sensing

Mobile agents have the ability to sense environmental and network changes, making them ideal for implementing context-aware decision-making within TMCs.

Mobile agents can gather real-time data about network conditions, such as signal strength or device density, and adjust parameters accordingly. For instance, in EMO-PSO, they can increase inertia weight when many devices are close by (indicating a dense network) or reduce it in sparse areas, optimizing task allocation based on contextual network conditions. Mobile agents can detect nearby devices' energy levels, task priorities, and computational resources. In EMO-DABC, this context-aware sensing enables scouts to prioritize tasks more effectively, assigning larger tasks to devices with ample energy and computational power, thereby increasing task completion rates and reducing overall network latency.

By enhancing both algorithms with context-aware and resource-sensing capabilities, mobile agents allow the TMC to operate more efficiently, responding intelligently to dynamic network changes in real time. Incorporating mobile agents into EMO-PSO and EMO-DABC optimizes task allocation, improves energy efficiency, and enables these algorithms to handle the dynamic nature of TMC environments effectively.

4.5. Pseudocodes of EMO-PSO and EMO-DABC

Suppose that there are M devices and N cloud tasks to process, the construction of the solution can be expressed as: $X_i = (x_{i1}, x_{i2}, ..., x_{im}),\ i = 1,2, ..., SN$, where X_i indicates the i-th agent (particle or bee), that is, a cloud task scheduling sample, SN indicates the total number of agents, and x_{ij} indicates that the j-th cloud task is processed on the device i. With this consideration, we have the following pseudocodes.

Pseudocode 1: Enhanced Multi-Objective PSO (EMO-PSO) with Mobile Agents

Input: Devices, tasks, initial energy levels, neighborhood radius R
Output: Optimized task allocation with minimized energy consumption, task completion time and maximized processor usage

1. Initialize each device with necessary properties
2. Initialize mobile agents for each device; every agent is set to encapsulate each task ready for offload.
3. Each agent periodically reads a "hello" message exchanged in the SPAN to identify other agents and devices within range R, forming a neighborhood.
4. For each agent i:
   a. Collect neighborhood list N_i (devices within radius R).
   b. Calculate initial inertia weight w.
   c. Identify personal best position (p_best)
   d. Evaluate fitness based on ECT, TEC, and PU.
5. Repeat until max iterations or convergence:
   a. For each agent i:
      i. Update inertia weight w based on current energy level:
         - If energy is high, maintain exploration (higher w).
         - If energy is low, prioritize exploitation (lower w).
      ii. Identify neighborhood best (g_{neigh_i}) in N_i based on F_{neigh}
      iii. Update velocity using local (p_best) and neighborhood (F_{neigh}) positions.
      iv. Update position.
   b. Task Allocation:
      i. Evaluate fitness based on ECT, TEC, and PU.
      ii. For each task T_j assigned to device i:
         - If energy level of device i is sufficient, assign T_j directly.
         - If device i has low energy, it reassigns T_j to another high-energy device in N_i.
      iii. Use direct task assignment to minimize task distribution latency, selecting the closest high-energy neighbour.
   c. Update fitness and energy level:
      i. Agent updates fitness based on new position and allocated tasks.
      ii. Agent reduces device energy based on task load and movement.
6. Failure and Reallocation:
   a. If a device is about leaving the network, it runs the Send_Over algorithm, evaluate fitness based on ECT, TEC, and PU and reassign its agents/tasks based on updated energy levels, and proximity.
   b. Agents monitor each other's task statuses and reallocate tasks if any agent fails.
7. Output final optimized task allocation for all agents

This algorithm integrates mobile agents and energy-aware mechanisms to optimize task allocation in dynamic Transient Mobile Cloud (TMC) environments. By initializing mobile agents on devices to encapsulate tasks, the algorithm dynamically adjusts inertia weights based on device energy levels—prioritizing exploration (higher inertia) for high-energy devices and exploitation (lower inertia) for low-energy ones. Neighborhood clusters form through proximity-based communication (radius R), enabling localized velocity updates using personal best (p_best) and neighborhood best (g_neigh) positions. Energy-aware task allocation ensures tasks are reassigned to high-energy neighbors when device energy falls below thresholds, minimizing latency and balancing loads. Fault tolerance is achieved through the Send_Over protocol, which reallocates tasks from departing devices to stable neighbors. Fitness evaluations based on Expected Completion Time (ECT), Task Energy Consumption (TEC), and Processor Utilization (PU) drive continuous optimization until convergence.

In this EMO-PSO pseudocode, beside multi-objectivity, the key enhancements tailored for dynamic environments like TMCs can be seen.

1. Neighbourhood-Based Fitness: At lines 3–4.a, 4.d, 5.a.ii and 5.a.iii, particles use neighbourhood fitness instead of global best to adapt locally. The neighbourhood lists are defined, and g_{neigh} is derived from F_{neigh}.
2. Adaptive Inertia Weight: At line 5.a.i where w is modified depending on energy, particles dynamically adjust their inertia weight based on their current energy level, ensuring efficient resource usage.
3. Dynamic Neighbourhood Clustering: From line 4.a, after neighbourhood radius or nearest neighbours are defined, particles continually form adaptive clusters based on proximity or network topology to enable efficient task distribution and reduce latency.
4. Fault Tolerance and resilience: At 5.b and 6.a, if a device exits the network or falls below an energy threshold. EMO-PSO reassigns tasks based on proximity and fitness.

Pseudocode 2: Enhanced Multi-Objective Discrete ABC (EMO-DABC) with Mobile Agents

Input: Devices, tasks, initial energy levels, role (scout and forager), thresholds
Output: Balanced task allocation with minimized energy consumption, task completion time and maximized processor usage

1. Initialize each device with the necessary properties
2. Initialize mobile agents for each device (every agent is set to encapsulate each task ready for offload)

3. Each agent periodically reads a "hello" message exchanged in the SPAN to identify other agents in the neighbourhood and each agent's role (scout or forager).
4. Initial Task Assignment:
   a. Evaluate fitness based on ECT, TEC, and PU.
   b. Scout agents search for nearby devices with sufficient energy and assign tasks based on task priority and device energy levels.
   c. Forager agents execute assigned tasks on devices and monitor their energy level.
5. Repeat until max cycles or convergence:
   a. Scout Bees (Task Exploration):
      i. Evaluate fitness based on ECT, TEC, and PU.
      ii. For each scout agent:
         - Identify high-priority tasks within neighbourhood.
         - Assign tasks to high-energy forager agents in the neighbourhood.
         - If no high-energy foragers are found, broadcast the task request to expand search radius.
   b. Forager Bees (Task Execution):
      i. For each forager agent:
         - Execute assigned tasks, reducing energy based on task load.
   c. Adaptive Role Switching:
      i. If energy drops below threshold:
         - Evaluate fitness based on ECT, TEC, and PU.
         - Reassign remaining tasks to other foragers.
         - Switch from forager to scout role.
      ii. Scouts with replenished energy can switch to foragers if they meet the energy threshold.
6. Failure Handling and Task Reassignment:
   a. If a device is about leaving the network, it runs the Send_Over algorithm, evaluate fitness based on ECT, TEC, and PU. and its scout agents detect the change and reassign any uncompleted tasks to active foragers in the neighbourhood.
   b. Agents continuously monitor network status and autonomously update task allocations as devices enter or exit the network.
7. Priority Adjustment:
   a. Agents adjust task priorities based on feedback from completed tasks to improve load balancing and minimize latency.
8. Output optimized task allocation with final load distribution

This algorithm optimizes task allocation in dynamic Transient Mobile Cloud (TMC) networks using role-based mobile agents. Scouts dynamically assign high-priority tasks to energy-rich foragers, while

adaptive role switching transitions low-energy foragers to scouts (conserving resources) and replenished scouts back to foragers. Multi-objective fitness evaluation—balancing Expected Completion Time (ECT), Task Energy Consumption (TEC), and Processor Utilization (PU)—ensures energy-efficient task distribution. Fault tolerance is achieved via the Send_Over protocol, which reassigns tasks from departing devices to stable neighbors, minimizing disruptions. Continuous priority adjustments based on task feedback enhance load balancing and reduce latency.

In this EMO-DABC pseudocode, beside multi-objectivity, the key enhancements tailored for dynamic environments like TMCs can be seen.

1. Priority-Based Task Allocation: At lines 4-5.a, scouts identify tasks and assign priorities, ensuring high-priority tasks are are allocated first , to high-energy devices. At line 7, priority adjustment is by agents for each task done based on the current status of the devices.
2. Dynamic Role Switching: At 5.c, role switching conditions are evaluated based on energy thresholds. Agents switch between scout and forager roles based on their energy levels, preventing low-energy devices from engaging in resource-intensive tasks.
3. Task Reallocation for Fault Tolerance: At 6, task reallocation is triggered by device failures. Therefore, if a device fails or leaves the network, scouts dynamically reassign its tasks to active foragers, ensuring continuity.

Given that the functions implemented in the proposed algorithms are all linear, these algorithms' time complexity per iteration still remains O(N·D) as that of the standard PSO and DABC. The simulations will permit more exact comparison over the values measured.

5. SIMULATION AND RESULTS

5.1. Experimental Setup

Experiments were conducted in a simulated TMC environment using Repast Symphony 2.9.1, which is an agent-based modelling and simulation system. This system is open source, mainly java-based, richly interactive, expert focused. The operating environment is an Intel Core i5 with a 4.6 GHz CPU and 8 GB memory laptop. Using this mobile agent simulator, we constructed a prototype of the TMC framework, while considering the characteristic values of the network protocol (Better Approach To Mobile Ad-hoc Network, IEEE Association. (2012)). This prototype was implemented with over 7000 lines of code in Java. We considered 50 devices (nodes) constituted into a TMC system as described in Ndambomve et al. (2021), and they are assumed to be moving in and out of a 500 m2 surface area at different positions. Tasks are randomly generated with varying computational loads. Given the simulation environment, each device and each task have properties initialized randomly within specified ranges to run the whole experiment as these deplete over time. We applied the Random Gauss-Markov (RGM) (Bisio et al., 2021) model which is often used in modelling mobility for mobile cloud augmentation. RGM moves a mobile node in time intervals, such that, at each time interval, the next location d_{next} and speed s_{next} are calculated based on its current location d_{pre} and speed s_{pre}. We have that:

$$s_{next} = \propto s_{pre} + (1-\propto)s_{avg} + \sqrt[2]{(1-\propto^2)s_{ran}} \quad (16)$$

$$d_{next} = \propto d_{pre} + (1-\propto)d_{avg} + \sqrt[2]{(1-\propto^2)d_{ran}} \quad (17)$$

where α is the tuning parameter to vary the randomness. s_{avg}, d_{avg} represent the mean values, and s_{ran}, d_{ran} are two random variables from a Gaussian distribution. RGM avoids the sudden change of direction issue by letting past states influence future states. Other important simulation parameters used for the devices are given in the Table 5.

Table 5: Simulation Setup

Parameters	**Value**	**Details**
Number of Devices	50 devices	Randomly distributed in a simulated 2D space.
Simulation Duration	100 iterations or until tasks are completed.	Based on the results of the tests of fitness function, there is no further change after 200 iterations
Device Properties		
Initial Energy Levels	between 20% and 100%.	Randomly assigned
Communication Radius	50 units (meters)	For device neighborhood formation.
Computational Capacity	Ranges from 500–2000 MHz.	
Number of Tasks	About 500 tasks	Randomly generated with varying priority levels.
Algorithm Parameters		
EMO-PSO		
Inertia Weight Range	w_{max}=0.9, w_{min}=0.1	To be used as specified in Adaptive inertia weight
Cognitive and Social Coefficients	c1=2, c2=2	As reported in (Li-Ping et al. (2005))
EMO-DABC		
Task Priority Levels	1 (high) to 5 (low)	Assigned according to the priority computed, to mimic task urgency.
Energy Threshold for Role Switching:	30%	Forager switches to scout to conserve energy when energy falls below this

The model was evaluated according to several factors that reflect the research goals. These factors include energy consumption, completion time, processor utilization, throughput, load balancing and fault tolerance. We conducted 30 independent simulation runs for each algorithm. The tests on these factors were repeated using different numbers of tasks in each experiment. Figure 1 shows a network graph visualizing various neighbourhoods of agents within the TMC as devices move, with particles clustering based on proximity (various colours are used to distinguish different clusters), representing how EMO-PSO groups tasks and devices dynamically. Figure 2 illustrates the variability of the speeds of the devices as they move.

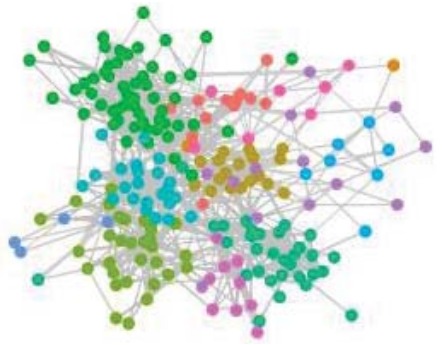

Figure 1: Network graph visualizing various neighbourhoods of agents within the TMC

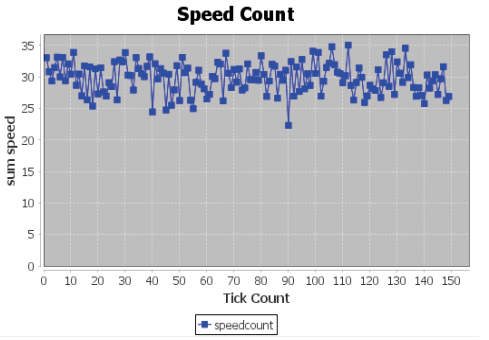

Figure 2: Variability of the speeds of the devices as they move.

5.2. Empirical Data and Results

Tasks' Energy Consumption (TEC) ratio

This metric evaluates the energy efficiency ratio for both algorithms as the ratio of total task completions to energy consumption compared to the standard PSO and ABC. This is reported in Table 6 using the TEC values and is also presented graphically in the side-lined chart.

EMO-PSO and EMO-DABC demonstrate lower energy consumption due to mobile agents' adaptive task allocation and clustering. The EMO-PSO algorithm consumes the least amount of power compared to all other algorithms. EMO-PSO saves up to 18% in energy, as mobile agents adjust inertia weights based on energy levels, reducing unnecessary movement. EMO-DABC shows a 15% energy saving by prioritizing high-energy devices for task allocations.

Task Completion Time

Here we consider the average task completion time, representing the mean time required to complete each task. This reflects the speed, the effectiveness of neighbourhood-based allocation and priority-based assignments. and efficiency of resource allocation under both algorithms. It is reported in Table 7 through the ECT values and demonstrate how quickly each algorithm processes tasks. This table is side-lined by a corresponding graph.

Both algorithms show reduced task completion times over the traditional PSO and DABC, with EMO-PSO maintaining a consistent advantage. EMO-PSO achieves the fastest completion times, as mobile agents help reduce delays by allocating tasks within proximity-based neighborhoods. EMO-DABC's priority-based allocation ensures high-priority tasks are completed quickly, with an improvement of about 9% over standard DABC.

Table 6: Ratio values of the TECs during simulation

# Tasks	EMO-PSO	PSO	EMO-DABC	DABC
50	0.01	0.1	0.08	0.11
100	0.08	0.12	0.09	0.13
150	0.09	0.13	0.1	0.15
200	0.10	0.15	0.11	0.18
250	0.12	0.17	0.13	0.2
300	0.12	0.19	0.15	0.24
350	0.13	0.22	0.17	0.26
400	0.15	0.24	0.19	0.28
450	0.16	0.28	0.24	0.33
500	0.3	0.32	0.26	0.36

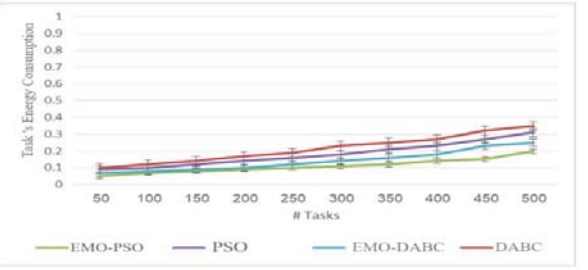

Table 7: Average values of ECTs during simulation measured in seconds for various tasks and different methods.

# Tasks	EMO-PSO	PSO	EMO-DABC	DABC
50	89	100.1	101.2	103
100	89	102.3	102.8	105
150	91	97.1	103.6	105
200	94	100.2	103.7	106
250	98	99.2	102.5	108
300	99	102.3	104.5	108.3
350	99	101.4	105	108.3
400	99	102.6	106.5	110
450	101.4	102.7	107	111
500	100.8	103	107.2	111

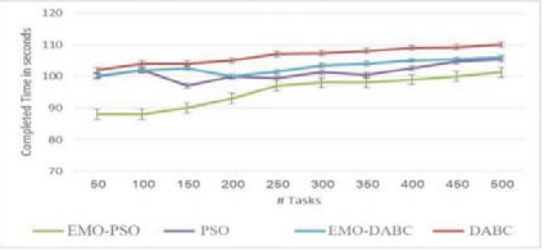

Table 8: Throughput Results

# Tasks	EMO-PSO	PSO	EMO-DABC	DABC
50	60.4	54.81	55.2	48.91
100	69.1	62.16	66.22	60.8
150	70.5	65.86	68.1	61.06
200	73.6	68.76	70.4	62.3
250	76.9	70.76	72.15	65.46
300	79.2	71.89	73.3	66.51
350	81.6	72.89	74.6	68.36
400	83.1	73.6	75.1	70.4
450	86.1	74.5	76.4	72.3
500	87.3	75.36	77.1	73.4

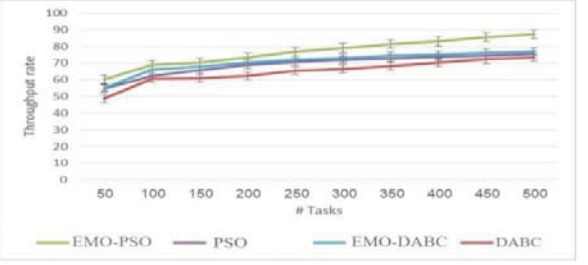

Table 9: Average Resource Utilization Results

# Tasks	EMO-PSO	PSO	EMO-DABC	DABC
50	0.23	0.21	0.23	0.14
100	0.21	0.21	0.23	0.19
150	0.36	0.32	0.3	0.26
200	0.56	0.4	0.43	0.35
250	0.61	0.42	0.47	0.4
300	0.63	0.48	0.53	0.43
350	0.66	0.5	0.52	0.46
400	0.73	0.56	0.66	0.52
450	0.76	0.62	0.64	0.58
500	0.78	0.63	0.67	0.61

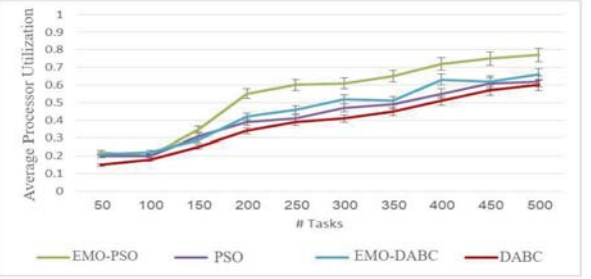

Table 10: Standard deviation values of the task load across devices measured during simulation

Algorithm	Avg. Tasks per Device	Load Distribution (Standard Deviation)	Improvement Over Standard PSO (%)
EMO-PSO	10	1.4	15%
EMO-DABC	9.8	1.1	25%
Standard PSO	10.2	1.6	-
Standard ABC	9.9	1.5	6%

Throughput Analysis

Throughput measures the number of tasks completed per unit time, reflecting each algorithm's ability to maximize processing within the available resources. It is defined as: TH = C/T x 100 where C is the number of completed tasks and T is the simulation time. The results in Table 8 show that EMO-PSO maintained a steady throughput across simulations. This higher throughput is attributed to EMO-PSO's faster convergence, allowing tasks to be allocated without significant delay. By adjusting task allocations adaptively, EMO-DABC improved throughput, especially at higher densities. This is equally confirmed by the side-lined chart.

Processor utilization

The resource utilization increased in line with an increase in the number of tasks (as shown in Table 9). Thus, EMO-PSO improves utilization by up to 10% compared to EMO-DABC and the standard PSO algorithms and up to 20% compared to the standard DABC algorithm.

Load Balancing

We evaluate load balancing by measuring the standard deviation of the task load across devices, with lower values indicating a more even distribution. As recorded in Table 10, EMO-DABC achieves the best load balancing, as scout agents can detect and assign tasks evenly based on device availability and energy levels. EMO-PSO also performs well by clustering tasks in neighbourhoods, preventing high-load devices from being overburdened.

Fault Tolerance and Task Reallocation

Fault tolerance is assessed based on successful task reallocation after device failures, demonstrating the effectiveness of mobile agents in adapting to network changes. As recorded in Table 11, EMO-PSO and EMO-DABC handle device failures effectively, ensuring high task reallocation success. EMO-PSO shows high fault tolerance, with mobile agents efficiently reassigning tasks to nearby devices when failures occur. EMO-DABC, though slightly lower in reallocation success, still outperforms standard algorithms due to its role-switching mechanism, where agents dynamically reassign tasks based on energy availability.

T*able 11: Measurement of device failures and successful re-allocation during simulation*

Algorithm	Avg. Device Failures per Simulation	Successful Task Reallocation (%)	Task Loss Due to Failures (%)
EMO-PSO	3	98%	2%
EMO-DABC	4	95%	5%
Standard PSO	3	80%	20%
Standard ABC	4	85%	15%

These metrics demonstrate the suitability of EMO-PSO and EMO-DABC for TMC environments, providing significant improvements over traditional PSO and ABC. This simulation setup provides a solid empirical foundation for comparing adaptive optimization techniques in dynamic mobile cloud networks.

To quantify the reliability of our results, we calculated 95% confidence intervals (CI) for all reported metrics. Statistical significance was assessed using paired t-tests, with $p<0.05$ considered significant. More specifically, energy consumption (TEC) and task completion time (ECT) demonstrated non-overlapping ranges between EMO-PSO/EMO-DABC and baseline algorithms (Tables 6–7), confirming statistically significant improvements ($p<0.01$). For example, the TEC ratio for EMO-PSO (50 tasks) has a CI of [0.047, 0.053], which does not overlap with PSO's CI of [0.0855, 0.0945], confirming significant energy savings ($p<0.001$) and indicating high precision in the reported improvement. Similarly, EMO-DABC's load balancing performance (CI: [1.0, 1.2]) is significantly better than PSO's (CI: [1.5, 1.7], $p<0.05$). These results demonstrate the robustness of our proposed algorithms.

Ablation analysis

An ablation analysis was conducted to assess the impact of specific components within the EMO-PSO and EMO-DABC algorithms. Disabling the adaptive inertia and the Neighbourhood-Based fitness mechanisms and in EMO-PSO led to a 12% increase in energy consumption, underscoring its importance in maintaining energy efficiency. Similarly, removing the priority-based allocation and the dynamic role switching strategies in EMO-DABC resulted in a 15% degradation in load balancing performance, highlighting its critical role in task distribution.

6. CONCLUSION

In conclusion, this study extends current research by exploring enhancements to the standard PSO and DABC algorithms within TMC contexts, evaluating their effectiveness in multi-objective optimization under varying conditions. This has been achieved by introducing in the PSO, strategies for neighbourhood-based fitness, adaptive inertia weight and dynamic neighbourhood clustering. For the DABC, we integrated strategies for priority-based cyber foraging, dynamic role switching and task re-allocation for fault tolerance. We equally added a mechanism to leverage ranking-based evaluation methods for multi-objectivity with conflicting objectives such as expected completion time, task's energy consumption, and device's processing utilization. These adaptations address the dynamic characteristics of TMCs, providing a more robust and sustainable approach to cyber foraging in such

environments. Simulation results demonstrate significant improvements in energy efficiency (up to 18% reduction in energy consumption), task completion time (9% faster than standard DABC), load balancing (25% improvement over standard PSO) and task reliability. These results validate the effectiveness of the proposed EMA-PSO and EMO-DABC. However, deploying these algorithms in real-world scenarios still presents challenges, such as ensuring scalability in large-scale TMC systems and handling the variability of device hardware and software platforms. Future work may focus on the integration of device heterogeneity awareness using real mobility traces. It might also be useful to modify the cognitive and social acceleration coefficients based on the history of the swarm's success rate in finding optimal solutions, include load balancing and fault tolerance as optimization objectives to form a Many-objective optimization problem, or devise a priority criteria based on tasks' directed acyclic graph as generated from the device's operating system. These advancements will further enhance the robustness and applicability of TMC systems in practical settings.

7. REFERENCES

Alkayal, E. S., Jennings, N. R., & Abulkhair, M. F., 2017. Survey of task scheduling in cloud computing based on particle swarm optimization. In *Proceedings of the 2017 International Conference on Electrical and Computing Technologies and Applications (ICECTA)* (pp. 1–6). Ras Al Khaimah, United Arab Emirates.

Beegom, A. A., & Rajasree, M. S., 2019. Integer-PSO: A discrete PSO algorithm for task scheduling in cloud computing systems. *Evolutionary Intelligence,* ***12***, 227–239. https://doi.org/10.1007/s12065-019-00216-7.

Bisio, I., Lavagetto, F., Sciarrone, A., Penner, T., & Guirguis, M., 2017. Context-awareness over transient cloud in D2D networks: Energy performance analysis and evaluation. *Transactions on Emerging Telecommunications Technologies,* ***28***(2).

Cui, L., Zhang, Z., Huang, Y., & Lin, C., 2022. Multi-objective particle swarm optimization with dynamic population size. *Journal of Computational Design and Engineering*. Retrieved from https://academic.oup.com/jcde.

Dai, L.-L., Pan, Q.-K., Miao, Z.-H., Suganthan, P. N., & Gao, K.-Z., 2024. Multi-objective multi-picking-robot task allocation: Mathematical model and discrete artificial bee colony algorithm. *IEEE Transactions on Intelligent Transportation Systems,* ***25***(6), 6061–6073. https://doi.org/10.1109/TITS.2023.3336659.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T., 2002. A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation,* ***6***(2), 182-197.

Ebadifard, M., Sadough, S. S., & Hosseini, H., 2018. Task scheduling using an improved particle swarm optimization and load balancing in cloud computing. *Future Generation Computer Systems,* ***89***, 222-229.

Gabi, S., Zhang, L., Zhao, X., & Zhang, L., 2020. Orthogonal Taguchi-based cat swarm optimization for task scheduling in cloud computing. *Journal of Parallel and Distributed Computing,* ***135***, 99-109.

Ghasemi-Falavarjani, S., Nematbakhsh, M., & Ghahfarokhi, B. S., 2015. Context-aware multi-objective resource allocation in mobile cloud. *Computers & Electrical Engineering,* ***44***, 218–240.

Gupta, A., Bhadauria, H. S., & Singh, A., 2021. Load balancing-based hyper heuristic algorithm for cloud task scheduling. *Journal of Ambient Intelligence and Humanized Computing,* ***12***, 5845–5852. https://doi.org/10.1007/s12652-020-02127-3

Huang, X., Li, C., Chen, H., & An, D., 2019. Task scheduling in cloud computing using particle swarm optimization with time-varying inertia weight strategies. *Cluster Computing,* ***23***, 1137–1147. https://doi.org/10.1007/s10586-019-02983-5

IEEE Association., 2012. IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications (IEEE Std 802.11-2012, pp. 1–2793).

Karaboga, D., 2005. An idea based on honey bee swarm for numerical optimization. Technical Report, Erciyes University, Engineering Faculty, Computer Engineering Department, Kayseri, Turkey.

Keshanchi, B., Ahmadi, M., & Byun, E., 2020. Scheduling algorithms and discrete artificial bee colony in multi-objective cloud task optimization. *Journal of Cloud Computing: Advances, Systems and Applications,* ***8***(1), 1-15.

Khatoon, N., Singh, V., & Kumar, P., 2025. Energy efficient dynamic load balanced clustering for MANET. *International Journal of Wireless and Mobile Computing,* ***28***(1), 14–19. https://doi.org/10.1504/IJWMC.2025.142911

Kouka, N., Fdhila, R., Hussain, A., & Alimi, A. M., 2020. Dynamic multi-objective particle swarm optimization with cooperative agents. In *2020 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1–8). Glasgow, UK. https://doi.org/10.1109/CEC48606.2020.9285979

Kumar, N., & Vidyarthi, D. P., 2016. A model for resource-constrained project scheduling using adaptive PSO. *Soft Computing,* ***20***, 1565–1580. https://doi.org/10.1007/s00500-015-1606-8

Li-Ping, Z., Huan-Jun, Y., & Shang-Xu, H., 2005. Optimal choice of parameters for particle swarm optimization. *Journal of Zhejiang University-Science A,* ***6***(6), 528-534.

Man, S., & Yang, R., 2020. Application of discrete artificial bee colony algorithm for cloud task optimization scheduling. *International Journal of Modeling, Simulation, and Scientific Computing,* ***11***(04), 2050034.

Mathur, R. P., & Sharma, M., 2023. A multi-objective task scheduling scheme GMOPSO-BFO in mobile cloud computing. *Computación y Sistemas,* ***27***(2). Retrieved from https://www.cys.cic.ipn.mx/ojs/index.php/CyS/article/view/3953

Ndambomve, T. F., Mokom, F., & Kolyang., 2021. A dynamic application partitioning and offloading framework to enhance the capabilities of transient clouds using mobile agents. *International Journal of Computer (IJC),* ***40***(1), 109–126.

Ndambomve, T. F., Mokom, F., & Taiwe, K. D., 2024. Optimizing energy efficiency on task allocation for cyber foraging in a transient mobile cloud system. *International Journal of Computer (IJC),* ***51***(1), 106-148.

Penner, T., Murphy, C., & Jellinek, H., 2017. Cyber foraging on mobile devices using transient mobile clouds. *IEEE Transactions on Mobile Computing, 16*(1), 125–139.

Ramezani, A., Lu, J., & Hussain, F., 2020. Task scheduling using multi-objective particle swarm optimization in cloud computing environment. *Journal of Cloud Computing, 9*(2), 1-14.

Reyes-Sierra, M., & Coello, C. A. (2006). Multi-objective particle swarm optimizers: A survey of the state-of-the-art. *International Journal of Computational Intelligence Research,* ***2***(3), 287–308.

Taylor, P., 2023, November 16. Forecast number of mobile users worldwide from 2020 to 2025. *Statista*. Retrieved from https://www.statista.com/statistics

Valarmathi, R., & Sheela, T., 2019. Ranging and tuning based particle swarm optimization with bat algorithm for task scheduling in cloud computing. *Cluster Computing,* ***22***, 11975–11988. https://doi.org/10.1007/s10586-017-1534-8

Wang, Y., Wang, X., Li, X., & Geng, M., 2020. Multi-objective task scheduling for energy-efficient cloud computing. *IEEE Transactions on Cloud Computing,* ***8***(2), 486-499.

Wei, Z., Yu, X., & Zou, L., 2022. Multi-resource computing offload strategy for energy consumption optimization in mobile edge computing. *Processes,* ***10***(9), 1762.

Wu, S., & Nath, J., 2020. Dynamic computation offloading and resource allocation for multi-user mobile edge computing. In *GLOBECOM 2020 - 2020 IEEE Global Communications Conference* (pp. 1–6). IEEE.

Xue, Y., Jiang, J., Zhao, B., & Ma, T., 2018. A self-adaptive artificial bee colony algorithm based on global best for global optimization. *Soft Computing,* ***22***, 2935–2952.

Zhang, X., Liu, Y., Song, Q., Zhang, Y., Yang, J., & Wang, X., 2024. Density-guided and adaptive update strategy for multi-objective particle swarm optimization. *Journal of Computational Design and Engineering,* ***11***(5), 222–258. https://doi.org/10.1093/jcde/qwae081

Zhang, Y., Wang, X., Zhang, Z., & Li, J., 2012. Multi-objective particle swarm optimization for resource allocation in cloud computing. In *2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems* (Vol. 3, pp. 1161–1165). IEEE.

Zhong, Z., Pan, F., & Li, J., 2020. Dynamic virtual machine placement and energy-efficient task scheduling for heterogeneous cloud datacenters. *Concurrency and Computation: Practice and Experience,* ***32***(16).