Risk-Aware Task Migration for Multiplex Unmanned Swarm Networks in Adversarial Environments*

Kai Di[1], **Tienyu Zuo**[2], **Pan Li**[1†], **Yuanshuang Jiang**[2], **Fulin Chen**[2] and **Yichuan Jiang**[2]
[1]Hangzhou School of Automation, Zhejiang Normal University
[2]School of Computer Science and Engineering, Southeast University
{dikai, tienyu.zuo, lipan, yuanshuangjiang, chenfulin, yjiang}@seu.edu.cn

Abstract

With the rapid development and deep integration of artificial intelligence and automation technologies, autonomous unmanned swarms dynamically organize into multiplex network structures based on diverse task requirements in adversarial environments. Frequent task variations lead to load imbalances among agents and between network layers, significantly increasing the risk of enemy detection and destruction. Existing approaches typically simplify multiplex networks into single-layer structures for task scheduling, failing to address these load imbalance issues. Moreover, the coupling between task dynamics and network multiplexity dramatically increases the complexity of designing task migration strategies, and it is proven $\mathcal{NP}$-hard to achieve such load balancing. To address these challenges, this paper proposes a risk-aware task migration method that achieves dynamic load balancing by matching task requirements with both intra-layer agent capabilities and inter-layer swarm capabilities. Simulation results demonstrate that our approach significantly outperforms benchmark algorithms in task completion cost, task completion proportion, and system robustness. In particular, the algorithm achieves solutions statistically indistinguishable from the optimal solutions computed by the CPLEX solver, while exhibiting significantly reduced computational overhead.

1 Introduction

With the rapid advancement of artificial intelligence and automation technologies, autonomous unmanned swarms face complex challenges in adversarial missions [Di *et al.*, 2022a]. These systems may simultaneously perform diverse tasks such as reconnaissance, strike operations, and communications, each requiring distinct coordination patterns among agents [Yan and Di, 2022].

To efficiently accomplish these adversarial missions, agents naturally emerge multiplex networks through self-organization, manifesting as distinct network layers: information-sharing networks for reconnaissance tasks [Zhang *et al.*, 2024], collaborative combat networks for strike operations [Di *et al.*, 2022b], and relay transmission networks for communication tasks [Mandi *et al.*, 2024].

To systematically illustrate the characteristics of multiplex network structures, Appendix A1 presents the emergent multiplex network architecture where agents dynamically participate across multiple coordination layers. Through this complex structure, agents establish heterogeneous coordination patterns: concurrent operation in the information-sharing layer enables intelligence exchange, while participation in the collaborative strike layer facilitates combat task execution. This flexibility in cross-layer coordination represents a significant advancement over traditional single-layer architectures, substantially enhancing system adaptability and operational efficiency.

The emergence of these multiplex network structures, while enhancing system adaptability, introduces fundamental challenges in task scheduling and resource allocation mechanisms. In adversarial environments characterized by high dynamicity, temporal variations in task requirements inevitably generate load imbalances across network layers, leading to both diminished system performance and elevated vulnerability to adversarial detection. Existing approaches primarily rely on single-layer network abstractions [Yan and Di, 2022; Di *et al.*, 2022a], fundamentally failing to address the intricate inter-layer dependencies inherent in multiplex networks. This limitation becomes critical when coupled with task dynamics, as the resulting exponential expansion of the solution space renders traditional scheduling methodologies ineffective for practical deployment [He and Lv, 2024; Shi *et al.*, 2023].

To address these challenges, we propose a novel risk-aware task migration method that uniquely integrates centralized and distributed strategies. Our approach introduces an innovative two-scale architecture: an inter-layer migration mechanism coupled with intra-layer optimization. The inter-layer phase employs a sophisticated agent selection mechanism based on comprehensive adaptability scoring, enabling intelligent cross-layer task redistribution. This is complemented by an efficient intra-layer phase that implements distributed

*The appendices to this paper can be accessed online at: https://github.com/Kay941016/IJCAI2025

†Corresponding author.

progressive migration strategies, achieving fine-grained load balancing through local optimizations. The synergistic combination of these mechanisms enables our method to simultaneously maintain global load equilibrium while maximizing local migration efficiency.

To rigorously evaluate the effectiveness of our proposed method, we conducted extensive experimental validations across multiple critical dimensions, including task quantity variations, agent population scaling, and network layer configurations. The empirical results demonstrate the significant superiority of the method over benchmark algorithms in various operational scenarios [Long *et al.*, 2021; Jiang *et al.*, 2020]. Notably, the algorithm maintains computational efficiency with execution times consistently below 80ms, even under highly dynamic conditions. Furthermore, while achieving solution quality comparable to that of the CPLEX optimization solver [Pierre *et al.*, 2020], our method exhibits significantly reduced computational complexity, thereby establishing its practical viability for real-world deployments.

2 Related Works

Research on Adversarial Environments. Recent studies in adversarial game theory and agent control systems have focused primarily on addressing dynamic risks through individual agent strategies [Chung *et al.*, 2019]. In adversarial environments, autonomous unmanned swarms inherently form multiplex network structures due to diverse task requirements. This structural complexity, coupled with dynamic risks from opponent interference, creates unprecedented challenges in system stability. Traditional approaches fail to consider how dynamic risks propagate through different network layers simultaneously, leaving a critical gap in addressing the challenges of multiplex networked swarms [Cavorsi *et al.*, 2024; Wang *et al.*, 2024].

Research on Task Migration. Task migration studies have explored various dynamic factors including task variations, capability changes, and benefit fluctuations through single-layer optimization approaches. While such methods prove effective in conventional scenarios, they become inadequate when dealing with multiplex network structures where task requirements and agent capabilities are coupled across multiple layers [Mukhopadhyay *et al.*, 2022]. Existing studies typically assume homogeneous agents or one-to-one task-agent relationships, failing to address scenarios where tasks require coordinated execution across different network layers, particularly where dynamic factors can cascade through multiple network layers [Lin *et al.*, 2024; Shi *et al.*, 2024].

Research on Multiplex Networks. While multiplex network studies have revealed insights into risk propagation and dynamic factor interactions, current approaches often overlook the challenges posed by adversarial environments where network layers must adapt to both task variations and opponent actions [Li *et al.*, 2021; Lyu *et al.*, 2023]. Existing studies acknowledge risk propagation across layers but lack systematic approaches to handle load imbalances that can lead to cascading failures across the network structure [Zhou and Kumar, 2023; Santilli *et al.*, 2022]. This gap highlights the need for a comprehensive approach that can effectively manage task migration while considering both the multiplex nature of networks and the dynamic characteristics of adversarial environments.

3 Problem Statement

3.1 Preliminaries

Multiplex Unmanned Swarm Networks. We propose a multiplex network model to capture the diverse coordination relationships in autonomous unmanned swarms consisting of a set of agents $\mathcal{A}$. The network structure is formalized as a three-dimensional adjacency matrix $\mathcal{M} = [M^{[1]}, M^{[2]}, ..., M^{[|\mathcal{M}|]}]$, where each layer $M^{[l]}$ represents a distinct physical or information link type. Within the l^{th} layer, participating agents $A^{[l]} = \{a_1^{[l]}, a_2^{[l]}, ..., a_N^{[l]}\}$ establish different coordination patterns simultaneously.

Adversarial Environments. In adversarial environments, agents face two major risks when executing tasks:

- **Agent-Level Risk:** The higher the task load on an individual agent, the more likely it is to be detected and destroyed by the enemy [Di *et al.*, 2022b].
- **Layer-Level Risk:** The higher the overall load on a network layer, the more susceptible all agents within that layer become to being implicated and rendered unable to function normally [Di *et al.*, 2022a].

Considering these two types of risks, this paper defines the probability of an agent a_i working normally as p_i:

$$p_i = \overbrace{\left(1 - \psi_A\left(\sum_{t_k \in T(a_i)} q_k\right)\right)}^{\text{Agent-Level Risk}} \times \underbrace{\prod_{A^{[l]} \in \mathcal{A}_i}\left(1 - \psi_B\left(\sum_{a_i \in A^{[l]}} \frac{\check{Q}_i}{|A^{[l]}|}\right)\right)}_{\text{Layer-Level Risk}} \tag{1}$$

where $\psi_A(\cdot)$ and $\psi_B(\cdot)$ are monotonically increasing convex functions bounded between 0 and 1, describing the impact of an agent task load and the network layer load on the agent probability of working normally, respectively. As the agent load increases, the probability of the agent being destroyed also increases, as described by $\psi_A(\cdot)$ [Che *et al.*, 2018]. Similarly, $\psi_B(\cdot)$ captures the increasing probability of agents within a layer being implicated and destroyed as the layer load increases [Jiang and Jiang, 2008]. $T(a_i)$ represents the set of tasks currently being executed by agent a_i, while $\mathcal{A}_i$ denotes the set of network layers to which agent a_i belongs. Finally, $\check{Q}_i$ represents the total load of all tasks undertaken by agent a_i, and q_k represents the load of task t_k.

3.2 Optimization Objective

The primary objectives of this paper are to optimize task migration through the multiplex network structure among agents in adversarial scenarios, thereby reducing task completion

cost and increasing task completion probability. To achieve these objectives, we introduce two decision variables:

- x_i^k: A binary variable indicating the execution status of task t_k by agent a_i. If $x_i^k = 1$, agent a_i is currently responsible for executing task t_k; otherwise, $x_i^k = 0$.
- y_{ij}^k: A binary variable indicating the migration status of task t_k from agent a_i to agent a_j. If $y_{ij}^k = 1$, agent a_i migrates task t_k to agent a_j; otherwise, $y_{ij}^k = 0$.

Using these decision variables, we define these two optimization objectives as follows:

Definition 1. ***Task Completion Cost:*** *The task completion cost comprises the task execution cost and the task migration cost. The task execution cost represents the cost incurred by agents to complete all the loads associated with a task, while the migration cost represents the cost required to transfer tasks between different agents. In this paper, we denote the task completion cost under migration strategy π as $\mathcal{C}(\pi)$, which is defined as follows:*

$$\mathcal{C}(\pi) = \overbrace{\sum_{t_k \in \mathcal{T}} \sum_{a_i \in \mathcal{A}} \frac{q_k \cdot x_i^k}{v_i}}^{\text{task execution cost}} + \underbrace{\sum_{t_k \in \mathcal{T}} \sum_{a_i, a_j \in \mathcal{A}} w_{i,j} \cdot y_{ij}^k \cdot q_k}_{\text{task migration cost}} \quad (2)$$

where v_i represents the execution capability of agent a_i, which determines the rate at which the agent executes tasks; $w_{i,j}$ represents the link condition between agents a_i and a_j, which affects the cost of task migration between agents.

Definition 2. ***Task Completion Probability:*** *In an adversarial environment, the overall task completion probability is influenced by various factors, including the agent skills, the task requirements and the probability of agent working normally. The completion probability can then be given by:*

$$\mathcal{P}(\pi) = \sum_{t_k \in \mathcal{T}} \sum_{a_i \in \mathcal{A}} \psi_p \left(\frac{p_i \cdot x_i^k \cdot |S_i \cap R_k|}{|R_k|} \right) / |\mathcal{T}| \quad (3)$$

where $p_i \in [0, 1]$ is the probability of agent a_i working normally (Eq. (1)), S_i and R_k denote the skill sets of agent a_i and task t_k, respectively, and $\psi_p : [0, 1] \to [0, 1]$ is a monotonically increasing convex function mapping the ratio of possessed to required skills to a probability value, capturing the impact of various factors on task completion. $\mathcal{T}$ represents the set of all tasks.

3.3 Mathematical Model

Based on the above definitions, we can formulate the mathematical model of the Risk-Aware Multiplex Networked Task Migration (RA-MNTM) problem. In an adversarial setting, considering a given set of agents $\mathcal{A}$ and a multiplex network structure $\mathcal{M}$, our objective is to determine an optimal task migration strategy π^* that maximizes the utility function, which comprehensively considers the task completion cost $\mathcal{C}(\pi)$ and the task completion probability $\mathcal{P}(\pi)$. Mathematically, the RA-MNTM problem can be formulated as the following optimization problem:

$$\max \quad -\alpha_1 \cdot \mathcal{C}(\pi) + \alpha_2 \cdot \mathcal{P}(\pi) \quad \text{(RA-MNTM)}$$

$$\text{s.t.} \quad \text{Eq.(1)} \sim \text{Eq.(3)} \quad (4)$$

$$\sum_{i \in \mathcal{A}} x_i^k = 1, \quad \forall k \in \mathcal{T} \quad (5)$$

$$x_i^k = Z_i^k \cdot \left(1 - \sum_{j \in \mathcal{A}} y_{ij}^k\right) + \left(1 - Z_i^k\right) \cdot \sum_{j \in \mathcal{A}} y_{ji}^k, \quad \forall i \in \mathcal{A}, \forall k \in \mathcal{T} \quad (6)$$

$$\sum_{j \in \mathcal{A}} y_{ij}^k \leq Z_i^k, \quad \forall i \in \mathcal{A}, \forall k \in \mathcal{T} \quad (7)$$

The constraints can be classified into two categories:

- **Optimization Objective Constraint:** Constraint (4) represents the optimization objective, where α_1 and α_2 are non-negative weights that regulate the relative importance of the task completion probability and the task completion cost in the overall utility function, satisfying the condition $\alpha_1 + \alpha_2 = 1$.
- **Task Migration Constraints:** Constraints (5)-(7) capture the task migration relationships, where the binary variable Z_i^k indicates whether task t_k is initially assigned to agent a_i before the migration process. These constraints guarantee that if agent a_i is executing task t_k after migration, then either the task was already assigned to the agent before migration (*i.e.*, not migrated) or it was transferred from another agent during the migration phase.

3.4 Complexity Analysis

In this subsection, we discuss the complexity of the RA-MNTM problem proposed above. For this problem, finding a task migration strategy with the maximum utility is $\mathcal{NP}$-hard, which can be reduced to a well-known $\mathcal{NP}$-complete problem, *i.e.*, the Subset Sum Problem [Caprara *et al.*, 2000]. Due to space limitations, the detailed proof can be found in Appendix A2.

4 Two-Scale Adversarial Task Migration

The proposed Two-Scale Adversarial Task Migration (TSATM) algorithm addresses task migration in multiplex networks through a hierarchical approach that operates at two distinct scales: inter-layer and intra-layer. The key innovation of TSATM lies in leveraging this two-scale structure to achieve both global load balancing and local optimization.

4.1 Inter-layer Task Migration

Preliminaries. This algorithm identifies key agents across layers in a multiplex network system to optimize information exchange and task migration between layers. The selection process evaluates three main criteria: operational relia-

bility (probability of normal functioning), intra-layer closeness centrality, and agent connectivity across multiple layers. A fitness score metric is proposed to quantify these characteristics:

$$fs_i = \frac{\alpha_1 \cdot (\mathcal{D}_i + |\mathcal{A}_i|)}{\alpha_2 \cdot (1 - p_i)} \tag{8}$$

where $\mathcal{D}_i$ represents the closeness centrality of agent a_i, calculated as the reciprocal of the sum of distances from agent a_i to all other nodes in that network layer; $|\mathcal{A}_i|$ denotes the number of agents connected to a_i; and α_1, α_2 are weight parameters for the optimization objectives.

After selecting key agents, these agents gather network layer information and identify tasks that should be migrated to other layers to balance the workload across different network layers. During inter-layer task migration, only layers with low workloads and strong task reception capabilities are suitable for task immigration. Therefore, the target network layer must satisfy these conditions: low migration cost, high task requirement satisfaction, and low current workload. Considering these three factors, we propose a fitness comparison function $\xi^N\left(t_k, A^{[m]}, A^{[n]}\right)$ to evaluate the capability of a network layer to receive task t_k, defined as:

$$\begin{aligned}\xi^N\left(t_k, A^{[m]}, A^{[n]}\right) = \theta\left(w_{o,n} - w_{o,m}\right) \times \\ \theta\left(\frac{1-\psi_B\left(\left(Q^{[n]}+q_k\right)/\left|A^{[n]}\right|\right)}{1-\psi_B\left(Q^{[n]}/\left|A^{[n]}\right|\right)} \cdot N\left(R_k, A^{[n]}\right)\right. \\ \left. -\frac{1-\psi_B\left(\left(Q^{[m]}+q_k\right)/\left|A^{[m]}\right|\right)}{1-\psi_B\left(Q^{[m]}/\left|A^{[m]}\right|\right)} \cdot N\left(R_k, A^{[m]}\right)\right)\end{aligned} \tag{9}$$

where $w_{o,n}$ represents the migration cost from the key agent a_o in the source layer to the key agent a_n in the target layer $M^{[n]}$; $\psi_B(\cdot)$ denotes the destruction risk influenced by network layer workload; $N(R_k, A^{[n]}) = \sum_{a_i \in A^{[n]}} \frac{\theta(|S_i \cap R_k|)}{|A^{[n]}|}$ indicates the degree to which $A^{[n]}$ satisfies task requirements; $\theta(\cdot)$ is an indicator function that equals 1 when its argument is greater than or equal to 0, and 0 otherwise.

Algorithm Overview. Algorithm 1 presents the pseudocode for the Inter-layer Task Migration Algorithm. It takes key agents from each layer as input and identifies overloaded layers for migration, where $Q^{[i]}$ denotes the task load of layer $M^{[i]}$. Tasks are prioritized by size in a max-heap (lines 4-7) and processed greedily from the heap root. For each task, the algorithm selects the most suitable target layer based on the fitness comparison function $\xi^N\left(t_k, A^{[m]}, A^{[n]}\right)$ to balance workload across layers (lines 8-14). The migration process routes tasks through key agents: first to the key agent of the source layer, then to the key agent of the target layer, with migration costs calculated accordingly (lines 11-14). The algorithm terminates when workload balance is achieved across network layers.

Theoretical Analysis. We prove the rationality of the algorithm design, with the optimality of the network layer fitness comparison function shown in Theorem 1. We then analyze the time complexity of the algorithm, with the proof in Appendix A3.

Algorithm 1: Inter-layer Task Migration Algorithm

Input: Set of key agents $A^\star$, network layer $\hat{\mathcal{M}}$ with dynamic tasks
Output: Inter-layer migration cost $\mathcal{C}_{inter}$, updated task execution status $\vec{X}$ after migration

1 **Initialize:** $\mathcal{C}_{inter} = 0$, $\bar{Q} = \sum_{i \in \mathcal{M}} Q^{[i]}/|\mathcal{M}|$;
2 **for** $M^{[i]} \in \hat{\mathcal{M}}$ **do**
3 MaxHeap $\mathcal{H}.clear()$;
4 **if** $Q^{[i]} > \bar{Q} + \sigma_N$ **then**
5 **for** $a_j \in A^{[i]}$ **do**
6 **for** $t_k \in T(a_i)$ **do**
7 MaxHeap $\mathcal{H}.push\left(\langle\langle a_j, t_k\rangle, q_k\rangle\right)$;
8 **while** $Q^{[i]} > \bar{Q} + \sigma_N$ **do**
9 $a_{out}, t_{out} \leftarrow \mathcal{H}.top()$;
10 Select $A^{[m]}$ for all $\xi^N(t_{out}, A^{[m]}, A^{[n]}) > 0$;
11 Migrate t_{out} to key agent a_m from $A^{[m]}$;
12 $\mathcal{C}_{inter} \leftarrow \mathcal{C}_{inter} + w_{out,m}$;
13 $Q^{[i]} \leftarrow Q^{[i]} - q_{out}, Q^{[m]} \leftarrow Q^{[m]} + q_{out}$;
14 **return** $\mathcal{C}_{intra}, \vec{X}$

Theorem 1. *In an adversarial scenario, for network layers $A^{[m]}$ and $A^{[n]}$, if $\xi^N\left(t_k, A^{[m]}, A^{[n]}\right) > 0$, then transferring the task to the key agent within $A^{[m]}$ results in a better objective function value.*

Proof. To describe the impact of task transfer on the objective function, we divide it into the impact on task cost $\mathcal{C}(\pi)$, denoted as $\mathcal{C}(t_k, A^{[m]})$, and the impact on task completion probability $\mathcal{P}(\pi)$, denoted as $\mathcal{P}(t_k, A^{[m]})$.

Impact on task completion cost $\mathcal{C}(\pi)$: During inter-layer task transfer, the task is ultimately transferred to the key agent in another layer. The key agent may further transfer the task to other agents without considering the subsequent execution cost. The inter-layer transfer phase mainly considers the task transfer cost between key agents, which is $w_{o,m} \cdot q_k$. When the cost of transferring the task to the key agent in the destination layer is smaller, the impact on task completion cost is smaller. This holds if and only if $w_{o,m} \leq w_{o,n}$, which is satisfied when $\xi^N\left(t_k, A^{[m]}, A^{[n]}\right) > 0$, resulting in $\mathcal{C}(t_k, A^{[m]}) \leq \mathcal{C}(t_k, A^{[n]})$.

Impact on task completion probability $\mathcal{P}(\pi)$: Inter-layer task transfer affects the expected task completion probability in two aspects: 1) The probability of agents working normally, which is related to the layer load after task transfer and is given by $1 - \psi_B\left(\left(Q^{[m]} + q_k\right)/\left|A^{[m]}\right|\right)$. The impact rate on this probability is $\frac{1-\psi_B\left(\left(Q^{[m]}+q_k\right)/\left|A^{[m]}\right|\right)}{1-\psi_B\left(Q^{[m]}/\left|A^{[m]}\right|\right)}$; 2) The capability provided by the network layer, described by the capability ratio $N(R_k, A^{[m]})$. When the destination layer better meets the task requirements and has a lower current load, it

Algorithm 2: Intra-layer Task Migration Algorithm

Input: Network layer $M^{[l]}$, set of agents $\hat{A} \in A^{[l]}$ in the network layer experiencing task dynamic factors
Output: Intra-layer task migration cost $\mathcal{C}_{intra}$, task execution status $\vec{X}$ of network layer $A^{[l]}$

1 **Initialize:** $\mathcal{C}_{intra} = 0$, $\bar{Q} = \sum_{a_i \in A^{[l]}} \check{Q}_i$;
2 **for** $a_i \in \hat{A}$ **do**
3 **for** $t_k \in T(a_i)$ **do**
4 MaxHeap $\mathcal{H}.push\left(\left\langle \left\langle a_i, t_k \right\rangle, q_k \right\rangle\right)$;
5 **while** $!\mathcal{H}.empty()$ **do**
6 $a_{out}, t_{out} \leftarrow \mathcal{H}.pop()$;
7 **while** $\exists \xi^A(t_{out}, a_{in}, a_{out}) > 0$ **do**
8 **if** $Q_{out} > \bar{Q} + \delta$ **then**
9 Select a_{in} for all $\xi^A(t_{out}, a_{in}, a_m) > 0$;
10 Migrate t_{out} from a_{out} to a_{in};
11 $\mathcal{C}_{intra} \leftarrow \mathcal{C}_{intra} + w_{out,in}$;
12 $a_{out} \leftarrow a_{in}$;
13 **return** $\mathcal{C}_{intra}$, $\vec{X}$

is more likely to accept the task, balancing the load between layers. After the inter-layer task transfer, the task completion probability loss in the network layer is:

$$\psi_p\left(\frac{1-\psi_B\left(\left(Q^{[m]}+q_k\right)/\left|A^{[m]}\right|\right)}{1-\psi_B\left(Q^{[m]}/\left|A^{[m]}\right|\right)} \cdot N\left(R_k, A^{[n]}\right)\right) \tag{10}$$

Combining $\mathcal{C}(t_k, A^{[m]})$ and $\mathcal{P}(t_k, A^{[m]})$ in the form of an optimization objective function, we can conclude that when $\xi^N\left(t_k, A^{[m]}, A^{[n]}\right) > 0$, transferring task t_k to $A^{[m]}$ results in a better optimized objective value. □

4.2 Intra-layer Task Migration

Preliminaries. The Intra-layer Task Migration Algorithm considers three factors influencing the ability of an agent to receive tasks: agent load, task execution capability, and migration cost. This paper proposes the agent task acceptance capability comparison function:

$$\begin{aligned}\xi^A(t_k, a_m, a_n) =& \theta\left(1/v_n + w_{o,n} - 1/v_m - w_{o,m}\right) \times \\ & \theta\left(\frac{1-\psi_A\left(\check{Q}_m + q_k\right)}{1-\psi_A\left(\check{Q}_m\right)} \cdot |S_m \cap R_k| \right. \\ & \left. - \frac{1-\psi_A\left(\check{Q}_n + q_k\right)}{1-\psi_A\left(\check{Q}_n\right)} \cdot |S_n \cap R_k|\right)\end{aligned} \tag{11}$$

where v_n is the capability of agent a_n to execute task t_k, $w_{o,n}$ is the migration cost, $\psi_A(\cdot)$ is the destruction probability based on load, and S_n is the set of capabilities of a_n.

Algorithm Overview. Algorithm 2 presents the pseudocode for the Intra-layer Task Migration Algorithm. The algorithm begins by adding tasks from agents experiencing task dynamics to a max heap (lines 2-4). It then greedily selects the highest-load task to migrate, choosing the agent with the strongest receiving capability near the source agent as the migration target (line 8). The task is migrated to this target agent, and the algorithm checks if the receiving agent needs to further migrate tasks. If so, the migration process repeats, cascading tasks to suitable agents (lines 7-12). By iteratively selecting tasks from the max heap for intra-layer migration, load balancing is achieved among the agents.

Theoretical Analysis. We prove the rationality of the algorithm design, with the optimality of the agent task acceptance capability comparison function shown in Theorem 2, and analyze its time complexity, with the proof in Appendix A4.

Theorem 2. *In an adversarial scenario, for agents a_m and a_n, if $\xi^A(t_k, a_m, a_n) > 0$, then migrating task t_k to agent a_m will not result in a worse optimized objective function value compared to migrating it to agent a_n.*

Proof. Let $\mathcal{C}(t_k, a_m)$ denote the impact on task completion cost when migrating task t_k to agent a_m, and let $\mathcal{P}(t_k, a_m)$ represent the impact on task completion probability when migrating task t_k to agent a_m. The specific calculation process is as follows:

Impact on task completion cost $\mathcal{C}(\pi)$: In the intra-layer task migration phase, the task completion cost consists of the task execution cost and the task migration cost. The impact on task completion cost can be divided into the sum of the impacts on task execution cost and task migration cost, which is $q_k/v_m + w_{o,m} \cdot q_k$. In this case, $\mathcal{C}(t_k, a_m) \leq \mathcal{C}(t_k, a_n)$ holds if and only if $q_k/v_m + w_{o,m} \cdot q_k \leq q_k/v_n + w_{o,n} \cdot q_k$. When $\xi^A(t_k, a_m, a_n) \geq 0$, we have $1/v_m + w_{o,m} \leq 1/v_n + w_{o,n}$, and thus $\mathcal{C}(t_k, a_m) \leq \mathcal{C}(t_k, a_n)$.

Impact on task completion probability $\mathcal{P}(\pi)$: In the intra-layer task migration phase, two factors affect the task completion probability: 1) The probability of agent a_m working normally: The probability of agent a_m working normally is determined by both the network layer load and the agent load. During intra-layer task migration, tasks are only migrated within a single network layer; therefore, only the impact of the agent load on the normal working probability is considered. After task t_k is migrated to agent a_m, the probability of the agent working normally is $1 - \psi_A\left(\check{Q}_m + q_k\right)$, and the impact ratio of task migration on the probability of the agent being destroyed is $\frac{1-\psi_A\left(\check{Q}_m+q_k\right)}{1-\psi_A\left(\check{Q}_m\right)}$; 2) The capability provided by agent a_m: The capability set of agent a_m is S_m. Therefore, after intra-layer task migration, the expected task completion rate loss on agent a_m is as follows:

$$\psi_p\left(\frac{1-\psi_A\left(\check{Q}_m + q_k\right)}{1-\psi_A\left(\check{Q}_m\right)} \cdot \frac{|S_m \cap R_k|}{|R_k|}\right) \tag{12}$$

Combining $\mathcal{C}(t_k, a_m)$ and $\mathcal{P}(t_k, a_m)$, and composing them in the form of an optimization objective, we can conclude that when $\xi^A(t_k, a_m, a_n) > 0$, migrating task t_k to agent a_m yields a better objective function value. □

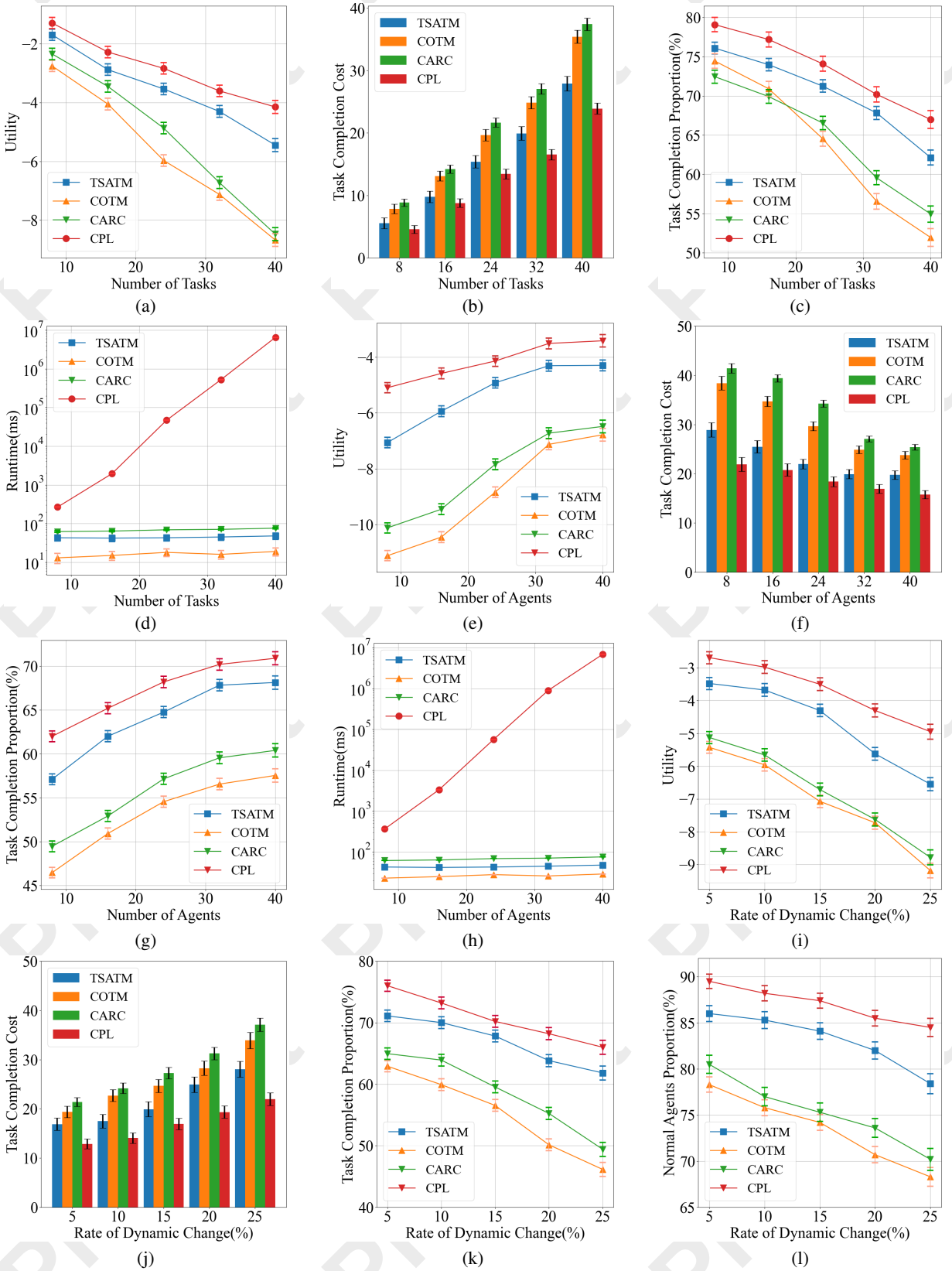

Figure 1: Performance analysis under varying parameters: Performance analysis under varying task numbers: (a)-(d); Performance analysis under varying agent numbers: (e)-(h); Performance analysis under varying task load change rates: (i)-(l).

5 Experiments

In this section, we compare our TSATM algorithm against Cost-Oriented Task Migration (COTM) algorithm proposed in [Long *et al.*, 2021], Context-Aware Robot Capability Migration (CARC) algorithm [Jiang *et al.*, 2020], and the optimal solution (CPL) obtained using the CPLEX solver [Pierre *et al.*, 2020].

5.1 Performance Analysis Under Varying Task Numbers

The experimental framework consists of 6 network layers, with 32 agents randomly distributed across these layers. The number of tasks ranges from 8 to 40, with an initial-to-dynamic task ratio of 1:1 and task arrival intervals within the range of [0,3]. The dynamic load variation rate is set at 15% per time interval.

Fig. 1(a)&1(b) illustrate that the proposed TSATM algorithm exhibits superior performance in terms of both objective function value and task completion cost, particularly in scenarios with high task loads, when compared to the COTM and CARC algorithms. This superior performance can be attributed to the two-scale migration strategy of the TSATM, which uses key agents for the global exchange of information, thus facilitating an efficient load balancing across the network layers.

Moreover, as depicted in Fig. 1(c), TSATM sustains higher expected task completion proportions under increasing task loads, closely approximating the optimal solution obtained by CPLEX. While COTM and CARC algorithms treat multiplex networks as single-layer structures, which often leads to sub-optimal local optima, the inter-layer migration mechanism of TSATM achieves a more balanced load distribution. It is noteworthy that, as illustrated in Fig. 1(d), all algorithms, with the exception of CPLEX, maintain execution times below 100ms.

5.2 Performance Analysis Under Varying Agent Numbers

We evaluate system performance by varying the agent population from 8 to 40, with agents randomly distributed across multiple network layers. Each agent can participate in different network layers simultaneously, maintaining consistent parameters with previous experimental settings.

As illustrated in Fig. 1(e)&1(f), the objective function value demonstrates positive correlation with agent numbers before reaching a plateau, with TSATM consistently outperforming benchmark algorithms while approaching the optimal solution provided by CPLEX. This enhancement is attributed to increased agent availability for task migration and reduced average agent load. The task completion costs exhibit a declining trend as the agent population grows, with TSATM achieving superior cost efficiency through its hierarchical migration strategy utilizing key agents.

Analysis of task completion proportions (Fig. 1(g)) reveals improved performance with increasing agent numbers, though marginal benefits diminish beyond certain thresholds. Significantly, while CPLEX exhibits exponential growth in computational complexity, TSATM maintains execution times below 80ms (Fig. 1(h)), demonstrating robust scalability through its heuristic approach. This balance between efficiency and performance validates the practical applicability of TSATM in large-scale deployments.

5.3 Performance Analysis Under Varying Task Load Change Rates

This empirical investigation examines the impact of dynamic task load variation rates (5%-25%) on system performance metrics through comparative analysis of four algorithms: TSATM, COTM, CARC, and CPL.

The experimental results in Fig. 1(i) demonstrate that while the objective function values exhibit a downward trend with increasing variation rates, TSATM maintains superior stability and achieves optimal performance approximating that of the optimal solution (CPL). Fig. 1(j) reveals that despite universal increases in task completion costs, TSATM demonstrates minimal cost escalation among all evaluated algorithms.

Further analysis in Fig. 1(k) indicates that expected task completion proportions decrease proportionally with higher variation rates, where TSATM exhibits enhanced resilience and significantly lower degradation compared to COTM and CARC. The agent operational reliability assessment in Fig. 1(l) shows consistent performance degradation across all algorithms, while TSATM maintains operational efficiency above 75%.

5.4 Performance Analysis Under Varying Objective Function Weight Parameters

The experimental results demonstrate that the proposed TSATM algorithm outperforms the COTM and CARC algorithms in terms of task completion cost and expected task completion proportion, as detailed in Appendix A5.

6 Conclusions

This paper presents a novel risk-aware task migration method for multiplex network systems in adversarial environments. Our approach introduces a two-scale architecture that uniquely combines centralized inter-layer migration with distributed intra-layer optimization. Through an innovative agent selection mechanism and distributed progressive migration strategy, our method effectively addresses the complex challenges across multiple network layers. Extensive experimental validation demonstrates the superiority of the method in terms of task completion cost, task completion proportion and system robustness.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Nos. 62476121, 62303111, 62076060, 61932007), the Guangxi Science and Technology Major Program (No. AA24206003), the Key Research and Development Program of Guangxi (No. AB2410317), the Key Research and Development Program of Jiangsu Province of China (No. BE2022157), and the Open Funds of KuiYuan Laboratory (No. KY202432).

References

[Caprara *et al.*, 2000] Alberto Caprara, Hans Kellerer, and Ulrich Pferschy. The multiple subset sum problem. *SIAM Journal on Optimization*, 11(2):308–319, 2000.

[Cavorsi *et al.*, 2024] Matthew Cavorsi, Lorenzo Sabattini, and Stephanie Gil. Multirobot adversarial resilience using control barrier functions. *IEEE Transactions on Robotics*, 40:797–815, 2024.

[Che *et al.*, 2018] Liang Che, Xuan Liu, and Zuyi Li. Mitigating false data attacks induced overloads using a corrective dispatch scheme. *IEEE Transactions on Smart Grid*, 10(3):3081–3091, 2018.

[Chung *et al.*, 2019] Jen Jen Chung, Andrew J Smith, Ryan Skeele, and Geoffrey A Hollinger. Risk-aware graph search with dynamic edge cost discovery. *The International Journal of Robotics Research*, 38(2-3):182–195, 2019.

[Di *et al.*, 2022a] Kai Di, Yifeng Zhou, Jiuchuan Jiang, Fuhan Yan, Shaofu Yang, and Yichuan Jiang. Risk-aware collection strategies for multirobot foraging in hazardous environments. *ACM Transactions on Autonomous and Adaptive Systems*, 16(3-4):1–38, 2022.

[Di *et al.*, 2022b] Kai Di, Yifeng Zhou, Fuhan Yan, Jiuchuan Jiang, Shaofu Yang, and Yichuan Jiang. A foraging strategy with risk response for individual robots in adversarial environments. *ACM Transactions on Intelligent Systems and Technology*, 13(5):1–29, 2022.

[He and Lv, 2024] Xiangkun He and Chen Lv. Robotic control in adversarial and sparse reward environments: A robust goal-conditioned reinforcement learning approach. *IEEE Transactions on Artificial Intelligence*, 5(1):244–253, 2024.

[Jiang and Jiang, 2008] Yichuan Jiang and Jiuchuan Jiang. Contextual resource negotiation-based task allocation and load balancing in complex software systems. *IEEE Transactions on Parallel and Distributed Systems*, 20(5):641–653, 2008.

[Jiang *et al.*, 2020] Jiuchuan Jiang, Bo An, Yichuan Jiang, and Donghui Lin. Context-aware reliable crowdsourcing in social networks. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(2):617–632, 2020.

[Li *et al.*, 2021] Cong Li, Yuan Zhang, and Xiang Li. Epidemic threshold in temporal multiplex networks with individual layer preference. *IEEE Transactions on Network Science and Engineering*, 8(1):814–824, 2021.

[Lin *et al.*, 2024] Peng Lin, Yan Liu, Zhizhong Zhang, F. Richard Yu, and Victor C. M. Leung. Cost-aware task offloading and migration for wireless virtual reality using interactive a3c approach. *IEEE Transactions on Vehicular Technology*, 73(7):10850–10855, 2024.

[Long *et al.*, 2021] Saiqin Long, Weifan Long, Zhetao Li, Kenli Li, Yuanqing Xia, and Zhuo Tang. A game-based approach for cost-aware task assignment with qos constraint in collaborative edge and cloud environments. *IEEE Transactions on Parallel and Distributed Systems*, 32(7):1629–1640, 2021.

[Lyu *et al.*, 2023] Chao Lyu, Yuhui Shi, Lijun Sun, and Chin-Teng Lin. Community detection in multiplex networks based on evolutionary multitask optimization and evolutionary clustering ensemble. *IEEE Transactions on Evolutionary Computation*, 27(3):728–742, 2023.

[Mandi *et al.*, 2024] Zhao Mandi, Shreeya Jain, and Shuran Song. Roco: Dialectic multi-robot collaboration with large language models. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 286–299. IEEE, 2024.

[Mukhopadhyay *et al.*, 2022] Atri Mukhopadhyay, George Iosifidis, and Marco Ruffini. Migration-aware network services with edge computing. *IEEE Transactions on Network and Service Management*, 19(2):1458–1471, 2022.

[Pierre *et al.*, 2020] Bonami Pierre, Lodi Andrea, and Zarpellon Giulia. A classifier to decide on the linearization of mixed-integer quadratic problems in cplex. *Operations Research*, 70(6):3303–3320, 2020.

[Santilli *et al.*, 2022] Matteo Santilli, Mauro Franceschelli, and Andrea Gasparri. Dynamic resilient containment control in multirobot systems. *IEEE Transactions on Robotics*, 38(1):57–70, 2022.

[Shi *et al.*, 2023] Guangyao Shi, Lifeng Zhou, and Pratap Tokekar. Robust multiple-path orienteering problem: Securing against adversarial attacks. *IEEE Transactions on Robotics*, 39(3):2060–2077, 2023.

[Shi *et al.*, 2024] You Shi, Changyan Yi, Ran Wang, Qiang Wu, Bing Chen, and Jun Cai. Service migration or task rerouting: A two-timescale online resource optimization for mec. *IEEE Transactions on Wireless Communications*, 23(2):1503–1519, 2024.

[Wang *et al.*, 2024] Zitong Wang, Yushan Li, Xiaoming Duan, and Jianping He. Topology-preserving motion coordination for multi-robot systems in adversarial environments. *IEEE Journal of Selected Topics in Signal Processing*, 18(3):473–486, 2024.

[Yan and Di, 2022] Fuhan Yan and Kai Di. Multi-robot task allocation in the environment with functional tasks. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 4710–4716, 2022.

[Zhang *et al.*, 2024] Boquan Zhang, Xiang Lin, Yifan Zhu, Jing Tian, and Zhi Zhu. Enhancing multi-uav reconnaissance and search through double critic ddpg with belief probability maps. *IEEE Transactions on Intelligent Vehicles*, 9(2):3827–3842, 2024.

[Zhou and Kumar, 2023] Lifeng Zhou and Vijay Kumar. Robust multi-robot active target tracking against sensing and communication attacks. *IEEE Transactions on Robotics*, 39(3):1768–1780, 2023.