Autonomous Energy-Saving Behaviors with Fulfilling Requirements for Multi-Agent Cooperative Patrolling Problem

Kohei Matsumoto, Keisuke Yoneda and Toshiharu Sugawara[a]
Department of Computer Science and Communications Engineering, Waseda University, Tokyo 1698555, Japan

Keywords: Multi-Agent Cooperative Patrolling, Coordination, Energy Saving, Self-Assessment.

Abstract: In this study, we propose a method to autonomously reduce energy consumption in the multi-agent cooperative patrol problem (MACPP) while fulfilling quality requirements. While it is crucial for the system to perform patrolling tasks as feasibly as possible, performing tasks beyond the required quality may consume unnecessary energy. First, we propose a method to reduce energy consumption by having agents individually estimate whether a given quality requirement is met through learning and consider energy-saving behaviors when diligent patrolling behavior is determined to be unnecessary. Second, we propose a method to deactivate redundant agents based on the values of parameters learned by each agent. Comparison experiments with the existing methods show that the proposed method can effectively reduce energy consumption while fulfilling the requirements. We also demonstrate that the proposed method can deactivate some agents for further energy savings.

1 INTRODUCTION

Recently, robot technology has advanced and has accelerated the use of multiple autonomous robots as a replacement for tasks that humans repeat daily, such as patrolling and cleaning, as well as for tasks in hazardous locations, such as disaster-stricken areas, nuclear power plants, and outer space. One such problem in which multiple robots cooperate to perform a common task is formulated as the *multi-agent cooperative patrolling problem* (MACPP), where robots are modeled as agents capable of operating autonomously. Studies of MACPP aim to find methods/algorithms for efficient and effective patrolling in the given environments through the cooperation and coordination of multiple agents.

Sophisticated actions and learning that aim to pursue only efficiency may consume more energy than necessary despite improving patrol efficiency, which is a crucial issue for MACPP. The autonomous agents envisioned in this study, in particular, have their own batteries and will be forced to be frequently recharged. Meanwhile, some applications have quality requirements for the patrolling tasks and are not necessarily expected to exceed them. For example, in a cleaning application, it is sufficient if the environment is clean to some extent, and excessive patrolling will in fact reduce the effectiveness of the work per unit of energy. Furthermore, if the environment is complex and large, it may not be possible to determine in advance how many agents are needed; fewer agents cannot fulfill the quality requirements, whereas too many agents will result in a waste of energy.

A few studies focused on energy efficiency in the context of collaborations in multi-agent systems (Kim et al., 2016; Benkrid et al., 2019; Notomista et al., 2022). For example, (Benkrid et al., 2019) proposed a decentralized coordination method that shortens the motion time to conserve the total motion energy of the mobile robots in a multi-robot exploration problem. However, these studies aimed at the methods to efficiently move and work for their tasks, consequently reducing the total energy consumption. Meanwhile, our method attempts to realize that some agents will stop for a while or exit the system by their own decision if other agents can satisfy the required quality of tasks. (Wu and Sugawara, 2019; Wu et al., 2019) also proposed the method for MACPP, with which agents autonomously pause for energy saving if they can fulfill the requirement and return to recharge for future exploration. However, we found that their method

[a] https://orcid.org/0000-0002-9271-4507

was insufficient, and agents still moved around the environment unnecessarily.

Therefore, we propose a method to both fulfill the quality requirement and reduce energy consumption more effectively by extending their method (Wu et al., 2019), while autonomously foreseeing the possible contributions by their subsequent actions and understanding the overall achievement of the required quality by estimating the current state of the environment. The main difference is that because the progress of the patrolling task while an agent pauses/recharges relies on the behaviors of other agents and is not the same for each agent, we introduce independent learning for the energy-saving behavior from the viewpoint of individual agents.

We further found that as this learning progresses, the agents split into two groups: the group of busy agents that move with short pausing time and the group of energy-saving agents that pause for relatively long periods of time to not consume energy. Therefore, the agents in the latter group can stop their operations sequentially while still satisfying the quality requirements. Our experiments show that our method can significantly reduce energy consumption compared with the previous study. In addition, we found that the number of busy agent groups varies with environmental conditions. Subsequently, we successfully reduced the number of operating agents by sequentially deactivating agents in the energy-saving group while fulfilling the quality requirements.

2 RELATED WORK

There have been many studies on MACPP and its applications (Hattori and Sugawara, 2021; Tevyashov et al., 2022; Wiandt and Simon, 2018; Othmani-Guibourg et al., 2017; Zhou et al., 2019). For example, (Othmani-Guibourg et al., 2017) proposed a model of dynamically changing environments based on the *edge Markov evolution graphs*. (Zhou et al., 2019) formulated the patrol problem as a Bayesian adaptive transition-separating partially observable Markov decision process and proposed a distributed online learning and planning algorithm that extends the Monte Carlo tree search method. In (Othmani-Guibourg et al., 2018), the authors proposed a method for predicting shared idleness from individual idleness using artificial neural networks. However, these studies only consider the efficiency by ignoring the periodical pauses due to battery discharge and energy consumption. (Yoneda et al., 2013) proposed a method called *adaptive meta-target decision strategy* (AMTDS), in which multiple agents cooperatively patrol the environment with periodic battery charge under a planning strategy determined by Q-learning. However, this study also aimed at only learning to improve patrolling efficiency without considering the reduction of energy consumption.

Meanwhile, (Kim et al., 2016; Benkrid et al., 2019; Notomista et al., 2022; Kim et al., 2016; Wu et al., 2019; Latif et al., 2021) in part focused on the energy saving. (Kim et al., 2016) introduced several subrobots besides the firefighting robots and attempted to extend the total operating time in firefighting tasks. (Latif et al., 2021) proposed an energy-conscious distributed task allocation algorithm to solve continuous tasks, such as foraging, for cooperative swarm robots to achieve highly effective missions. However, unlike ours, these studies also consider the extension of operating time. By contrast, (Wu et al., 2019) extended AMTDS to save energy subject to the quality requirements of patrolling. However, their method is insufficient for energy saving, and agents' activities still contain unnecessary actions. Therefore, we further reduced energy consumption while fulfilling quality requirements by introducing learning from individual viewpoints. Moreover, we also proposed a method to deactivate several agents for patrolling in order to reduce energy consumption.

3 BACKGROUND AND PROBLEM FORMULATION

3.1 Environment

Because our method is an extension of the energy saving method for cleaning applications, *adaptive meta-target decision strategy for energy saving and cleanliness* (AMTDS/ESC) proposed by Wu *et al.* (Wu et al., 2019), we follow their problem formulation and the models of the environment and agents' activities for MACPP. The environment in which the agents move around for patrolling is represented by the graph $G = (V, E)$, which can be embedded in a two-dimensional Euclidean space, where $V = \{v_1, \ldots, v_n\}$ is the set of nodes corresponding to the locations that agents visit, and E is the set of edges $e_{i,j}$ connecting nodes v_i and v_j corresponding to the paths along which the agents move.

We denote the set of n agents by $A = \{1, \ldots, n\}$ and introduce discrete time whose unit is *step*. For simplicity, all edge lengths are assumed to be 1 by adding dummy nodes if necessary. Hence, an agent can move to one of the neighboring nodes with no obstacle in a step. Let $d(v_i, v_j)$ be the shortest distance (number of edges) between v_i and v_j.

An event occurs at node $v \in V$ with its *event occurrence probability* (EOP), $0 \leq p(v) \leq 1$, and accumulates in v. Thus, the number of events accumulated in v at time t, $L_t(v)$, is updated by

$$L_t(v) = \begin{cases} L_{t-1}(v) + 1 & \text{(with EPO } p(v)\text{)} \\ L_{t-1}(v) & \text{(otherwise)} \end{cases}$$

However, the event in v is processed and $L_t(v) = 0$ when an agent visits v at time t. The interpretation of events differs depending on the applications; for example, in a cleaning application, $p(v)$ indicates the tendency to be dirty at location v, and $L_t(v)$ expresses the degree of accumulated dirt. In an application for security surveillance patrol, $p(v)$ indicates the degree of required security for critical locations specified by the application owner, and $L_t(v)$ can be interpreted as the alert level. We assume that $p(v)$ for all nodes is specified in advance.

3.2 Agent Model

Agent $i \in A$ has a battery whose capacity is described by $B_{max} > 0$ and can operate continuously by repeatedly recharging it at the charging base, B_i; that is, after leaving B_i with a full battery, i patrols the environment and returns to B_i before it runs out. We assume that the current battery capacity $b_i(t)$ at t is decremented, i.e., $b_i(t) \leftarrow b_i(t-1) - 1$ when i moves to a neighboring node. $b_i \leq 0$ signifies the battery runout. By introducing the charging speed constant $k_{charge} > 0$, it takes $(B_{max} - b_i) \cdot k_{charge}$ steps to make its battery full (so $b_i = B_{max}$).

Agent i knows $p(v)$ of all nodes v but cannot know the current number of the accumulated events $L_t(v)$ if i is not on v. Therefore, the agents calculate the expected value $E^i(L_t(v))$ from $p(v)$ at time t. For this calculation, we assumed that agents can know their own and other agents' locations. This can be achieved easily by current technology; for example, using sensors such as infrared rays or GPS, by direct communication between agents, or by cloud robotics, i.e., by sharing information via a cloud. However, agents do not share and infer their internal information and decisions, such as strategies to set the destinations and the planned routes to those destinations. We did not consider collisions between agents because this study focuses on learning energy-saving actions while meeting the required quality, and we believe that it is easy to generate detour routes to avoid collisions in the grid-like environment used in our experiments. In addition, several algorithms that generate collision-free routes were proposed (e.g., (Yamauchi et al., 2022; Ma et al., 2017)), and we can use one of these algorithms for collision avoidance.

3.3 Target Decision Strategy

We describe the agents' behaviors used in AMTDS/ESC (Wu et al., 2019). When an agent explores the environment further after reaching its previous goal or after its battery is full, it determines the next target node $v^i_{tar} \in V$ and then moves along the shortest path towards it.[1] Agent i selects the strategy that i learned as the best for the target decision by Q-learning from the following four strategies.

Random Selection (R): Agent i randomly selects $v^i_{tar} \in V$ from the environment.

Probabilistic Greedy Selection (PGS): Agent i randomly selects v^i_{tar} from the top N_g nodes of $E^i(L_t(v))$.

Prioritizing Unvisited Interval (PI): Agent i randomly selects $v^i_{tar} \in V$ from the top N_i nodes with large intervals between the most recently visited time and the current time.

Balanced Neighbor-Preferential Selection (BNPS): Agent i examines $E^i(L_t(v))$ in the neighbors and prioritizes v^i_{tar} if it is higher than the threshold; otherwise it uses the PGS.

For more details, please refer to the original (Yoneda et al., 2013; Wu et al., 2019).

3.4 Metrics for Task Quality from Subjective Viewpoint

We introduce two evaluation measures: the *time left without checking events* D_{t_s,t_e}, and the *total energy consumption* C_{t_s,t_e}, as follows:

$$D_{t_s,t_e} = \sum_{v \in V} \sum_{t=t_s+1}^{t_e} L_t(v), \text{ and} \tag{1}$$

$$C_{t_s,t_e} = \sum_{i \in A} \sum_{t=t_s+1}^{t_e} \mathcal{E}_t(i), \tag{2}$$

where $[t_s, t_e]$ $(t_s < t_e)$ denotes a time interval, and $\mathcal{E}_t(i)$ represents the energy consumed by agent i at t; therefore, $\mathcal{E}_t(i) = 1$ if i moved to a neighboring node and $\mathcal{E}_t(i) = 0$ otherwise. For example, D_{t_s,t_e} represents the cumulative time of dust left without being vacuumed in a cleaning application and the cumulative time and

[1] More precisely, it will drop in at any node near the shortest path that has a large value of $E^i(L_t(v))$, using the subgoal determination algorithm (Yoneda et al., 2013).

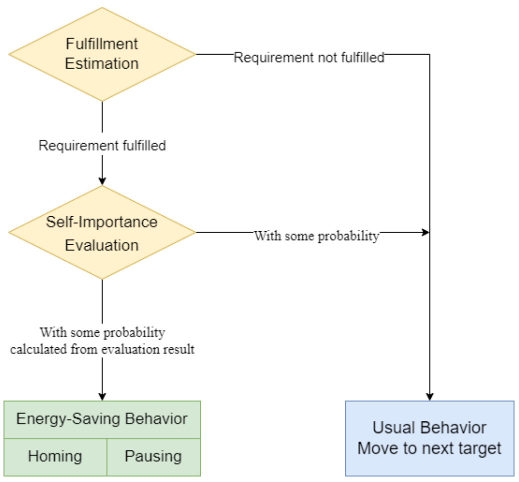


Figure 1: Flow for energy-saving behavior.

the number of secure locations left unchecked in security patrols. Therefore, agents cooperatively maintain D_{t_s,t_e} less than the required value $D_{req} \geq 0$ per step,

$$D_{t_s,t_e} \leq D_{req} \times (t_e - t_s) \tag{3}$$

Our aim is to keep C_{t_s,t_e} as low as possible while satisfying the quality requirement Formula (3). Note that $D(s)$ and $C(s)$ are used instead of D_{t_s,t_e} and C_{t_s,t_e}, by omitting the subscripts unless confused.

4 PROPOSED METHOD

4.1 Estimating Quality Requirement Fulfillment

We explain the proposed method by describing how it differs from the energy-saving behavior in the previous study (Wu et al., 2019). Figure 1 shows the flow of the behaviors involved in the energy-saving behavior of agents. Agents need to understand the current state of the environment and compare it with the required task quality to reduce energy consumption. However, any agent cannot get the actual state and must estimate it. In the *fulfilling estimation*, agents regularly check if the required quality is accomplished while learning how they can predict the future environmental state from individual viewpoints, although in the previous method (Wu et al., 2019), all agents estimate it in a uniformed viewpoint.

For this, agent i computes the expected value $E^i(L_{t_c+T}(v))$ for $\forall v \in V$ at a future time $t_c + T$ using

$$E^i(L_{t_c+T}(v)) = p(v) \times \{(t_c + T) - t^v_{vis}\} \tag{4}$$

where t_c denotes the current time, $T > 0$ represents the parameter to specify how far into the future i estimates, and t^v_{vis} indicates the most recent time when an agent visited v. Then, i should check if

$$\sum_{v \in V} E^i(L_{t_c+T}(v)) \leq D_{req} \tag{5}$$

is satisfied. Note that agent i estimates the future state here to foresee the impact of its energy-saving behavior, while i stops patrolling for some time.

However, this estimation by Formula (4) is an ideal case based on the uniformed workload and ignores the efforts of other agents. In particular, other agents maintain the required quality D_{req} even if the agent stops for energy saving or charging. Such efforts of other agents are obviously different depending on the individual viewpoints because agents visit different locations with different characteristics. Therefore, agents have to learn the possible efforts of other agents during the energy-saving behavior.

We introduce the learning parameter K^i for $\forall i \in A$ to adjust such differences. Then, agent i evaluates the environment using the following Formula (6), instead of Formula (5).

$$\sum_{v \in V} E^i(L_{t_c+T}(v)) \div K^i \leq D_{req} \tag{6}$$

Moreover, K^i is individually updated by

$$\begin{cases} K^i \leftarrow (1-\alpha)K^i + \alpha \dfrac{D_{req}}{E^i(D_t)} K^i & (\text{if } E^i(D_t) \leq D_{req}) \\ K^i \leftarrow K^i - \left(\dfrac{E^i(D_t)}{D_{req}} - 1 \right) & (\text{if } E^i(D_t) > D_{req}) \end{cases} \tag{7}$$

where the estimated number of events at t is defined as $E^i(D_t) = \sum_{v \in V} E^i(L_t(v))$, and $\alpha > 0$ is the learning rate. When K^i is updated is explained in Section 4.3.2.

4.2 Self-Assessment for Energy-Saving Behaviors

Agent i regularly checks Formula (6), and if it is satisfied, i evaluates its contribution toward fulfilling the quality requirement and foresees the impact of its energy-saving behavior on the future state of the environment (Fig. 1). For this purpose, we introduce the *self-assessment value* of agent i at t, ${S_{ass}}^i(t)$, which represents the degree of its recent contribution to determine if i should continue to patrol for the cooperative task or stop patrolling by adopting an energy-saving behavior.

To calculate ${S_{ass}}^i(t_c)$, we define three parameters for $\forall i \in A$, as follows:

$$U_s^i(t_c) = \frac{\sum_{t_c - T_s < t \le t_c} L_t(v^i(t))}{T_s} \tag{8}$$

$$U_l^i(t_c) = \frac{\sum_{t_c - T_l < t \le t_c} L_t(v^i(t))}{T_l} \tag{9}$$

$$U_f^i(t_c) = \frac{\sum_{t_c < t \le t_c + T_f} E^i(L_t(v^i(t)))}{T_f}, \tag{10}$$

where T_s and T_l (where $0 < T_s < T_l$) denote the past short- and long-term evaluation periods for the past, T_f (> 0) indicates the evaluation period for the future, and $v^i(t)$ represents the node where i was or will be at t. Intuitively, the *short-term past contribution* $U_s^i(t_c)$ and the *long-term past contribution* U_l^i represent the numbers of events processed by i in the past, whereas the *estimated contribution* $U_f^i(t_c)$ estimates the number of events that will be processed by i in the future until $t_c + T_f$. Note that the expected value $E^i(L_t(v))$ is used instead of $L_t(v)$ only in Formula (10).

Now, we define the self-assessment value ${S_{ass}}^i(t)$ ($0 \le {S_{ass}}^i(t) \le 1$) using

$${S_{ass}}^i(t) = \begin{cases} 0 & (\text{if } U_l^i = 0) \\ \dfrac{U_s^i + U_f^i}{U_l^i} & (\text{else if } U_s^i + U_f^i \le U_l^i) \\ 1 & (\text{otherwise}) \end{cases} \tag{11}$$

Then, i chooses an energy-saving behavior at the probability of $P^i(t)$ computed using equation (12).

$$P^i(t) = 1 - {S_{ass}}^i(t) \tag{12}$$

Therefore, a low self-assessment value facilitates the agent's energy-saving behavior.

4.3 Energy Saving Behaviors

Agents take one of two energy-saving behaviors: *Homing* or *Pausing*, to eliminate movements that they deem unnecessary.

Homing: Agent stops patrolling and returns to the charging base regardless of the remaining battery level.

Pausing: After charging is complete, the agent waits for S_{pause} steps without starting to move, where S_{pause} is a positive integer.

We will explain when the agents adopt these behaviors.

4.3.1 Homing Behavior

Agent i checks the requirement fulfillment (Formula (6)) every T_{homing} steps after its battery level becomes low, i.e., $b^i(t) < k_{homing} \cdot B_{max}$, where positive integer T_{homing} is the parameter that avoids the frequent return to charge by homing, and k_{homing} ($0 < k_{homing} < 1$) is the parameter that decides low battery level. If Formula (6) does not hold, i continues to patrol; otherwise, with probability $P_i(t)$, i follows a homing behavior in which i changes the current target to its charging base B_i. Note that agents patrolling far from B_i may return to base before this check.

Algorithm 1: PLength: To decide pausing time-length.

```
Require: $S_{pause} > 0$, $\gamma_p = 0$, $T_{\gamma_p} > 0$
 1: while $\gamma_p \le T_{\gamma_p}$ do
 2:     $T \leftarrow (\gamma_p + 1) \cdot S_{pause}$ // $T$ is used in Formula (6)
 3:     if Formula (6) holds then
 4:         $\gamma_p \leftarrow \gamma_p + 1$
 5:     else
 6:         break
 7:     end if
 8: end while
 9: // Agent takes a pausing behavior whose pausing
10: // time is $\gamma_p \cdot S_{pause}$. If $\gamma_p = 0$, agent immediately
11: // leaves for patrolling.
12: return $\gamma_p \cdot S_{pause}$.
```

4.3.2 Variable-Length Pausing Behavior

In the previous method (Wu et al., 2019), when the battery is fully charged at time t, agent i checks the condition of the requirement (Formula (5)), and if the condition is fulfilled, i takes the pausing behavior with a constant pausing time S_{pause} steps.

Meanwhile, in our proposed method, agents take variable pausing time depending on the condition of the estimated states. i takes the following *variable-length pausing behavior* with probability $P_i(t)$ it completes charging at t. However, as an exception, i always takes the variable-length pausing behavior only when it returns to the charging base B_i by homing behavior.

First, in the variable-length pausing behavior, i determines the pausing time as follows (see PLenght in Algorithm 1). Initially, agent i set $\gamma_p = 0$ and $T = S_{pause}$. When the battery is fully charged at time t, agent i checks the requirement fulfillment (Formula (6)), and if it does not hold, i immediately leaves the charging base for patrolling, i.e., the pausing time is zero. If it holds, i increments γ_p by 1 and sets $T \leftarrow (\gamma_p + 1) \cdot S_{pause}$. Then, i checks the fulfillment Formula (6) again; i repeats this process until i can decide the length of the pausing time or $\gamma_p = T_{\gamma_p}$, where T_{γ_p} is the maximal value of this iteration. Subsequently, i takes the pausing behavior whose length is $\gamma_p \cdot S_{pause}$. Agents always leave their charging bases after the variable-length pausing behavior.

Unlike the previous method, agent i always takes the variable-length pausing behavior after a full charge if i returns to the charging base by a homing behavior, as mentioned previously. This exception can be explained by the fact that only homing behavior has a marginal impact on energy saving because a homing behavior shortens the current patrolling time and reduces the time required for a full charge. Thus, we believe that it is natural to always call a variable-length pausing behavior to decrease energy consumption. Certainly, it is possible that the pausing time is zero by PLength, even in such a case. Hereafter, a variable-length pausing behavior is simply referred to as a pausing behavior.

Parameter K^i for $\forall i \in A$ is updated by Formula (7) after i completes a pausing behavior even if the return value of PLength is zero, because PLength carefully estimates the current state to obtain the maximum duration that each agent can wait up to while fulfilling its quality requirements.

4.4 Deactivation of Unnecessary Agents

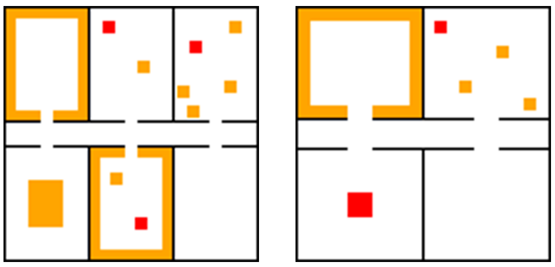
From the perspective of energy saving, if the number of agents needed for patrolling is sufficient, even excessive, it is effective not only to temporarily stop agents but also to deactivate the patrolling of some agents. Furthermore, these ceased agents can be used elsewhere or as backups in case of agent failure. However, we cannot know in advance which agents should be deactivated and determine how many agents are needed to meet the required quality in the environment, which is complex and whose distribution of POE is not uniform.

As we explain the data in detail in Section 5.3, we found large variations in the value of K^i among agents and that they were divided into two main clusters. In particular, because the agents belonging to the cluster with relatively larger values of K^i took pausing behaviors for a relatively longer duration to reduce energy consumption, we propose to deactivate agent $i_{de} \in A$ that has the largest value of K^i, i.e.,

$$i_{de} = \arg\max_{i \in A} K^i \tag{13}$$

Subsequently, if some agents still have a large value of K^i, the same deactivation process is repeated until all agents have relatively smaller K^i while the requirement is fulfilled.

Therefore, in every $D_{int} > 0$ steps, the system counts x_{pause}, which is the number of times the function PLength has been called by all agents, and if x_{pause} and $K_{i_{de}}$ are large, i.e.,


(a) Environment 1. (b) Environment 2.
Figure 2: Experimental environments.

$$x_{pause} \geq N_{deact} \text{ and } \max_{i \in A} K^i \geq K_{deact}, \tag{14}$$

then, i_{de} is deactivated. Herein, $K_{deact} > 0$ denotes the threshold value for deactivation, and N_{deact} indicates the threshold to decide if several pausing behaviors are attempted during the recent D_{int} from the system's viewpoint.

5 EXPERIMENTS AND DISCUSSION

5.1 Experimental Setting

We conducted experiments to compare the proposed method AMTDS/ER with the conventional method AMTDS(Yoneda et al., 2013) to verify the effectiveness of the proposed method, in which agents do not perform energy-saving behaviors, and that of AMTDS/ESC(Wu et al., 2019), in which the agents take homing and pausing behaviors but do not learn the parameter that decides when agents should take energy-saving behaviors. We demonstrate that the proposed method is more effective than other methods in that the agents can reduce energy consumption while fulfilling the quality requirement by investigating evaluation metrics $D(s)$ and $C(s)$. We analyzed the distribution of the values of K^i for $\forall i \in A$, and agents were divided into two groups in accordance with these values. Finally, we examined if our deactivation method could reduce the number of patrolling agents while fulfilling the quality requirement. We list the parameter values used in our experiments in Table 1. Note that the value of T in Formula (6) is defined in Algorithm 1.

We prepared two environments that have a two-dimensional grid structure with a size of 101×101, as shown in Fig. 2. The environment $G_1 = (V_1, E_1)$ for the first experiment (Exp. 1) is the same as the environment used for the experiment in (Wu et al., 2019),

Table 1: List of parameter description.

Description	Symbol	Value
Maximal battery capacity	B_{max}	900
Charging speed constant	k_{charge}	3
Past short-term evaluation period	T_s	20
Past long-term evaluation period	T_l	50
Future evaluation period for self-assessment	T_f	10
Learning rate for parameter K^i	α	0.1
Check point time for a homing behavior	T_{homing}	100
Check point for remaining battery capacity for a homing behavior	k_{homing}	$1/3$
Length of a pausing behavior	S_{pause}	100
Interval to check deactivation	D_{int}	$250,000$
Threshold for too many system-wide pause operations	T_{γ_p}	$1,000$
Parameter for deciding additional deactivation	N_{deact}	100
Threshold for deactivation	K_{deact}	1.0

for comparison (Fig. 2a). This environment has six separate rooms and a corridor connecting these rooms in the center. The black lines represent walls (obstacles that agents cannot pass through). Each node $v \in V_1$ is represented by a coordinate of integers (x_v, y_v), where $-50 \le x_v, y_v \le 50$. The EOP $p(v)$ for node $\forall v \in V_1$ was set according to colors in Fig. 2, as follows.

$$p(v) = \begin{cases} 10^{-3} & \text{(if } v \text{ is in a red region)} \\ 10^{-4} & \text{(if } v \text{ is in an orange region)} \\ 10^{-6} & \text{(if } v \text{ is in a white region)} \end{cases} \quad (15)$$

Therefore, the deeper the color, the higher the EOP.

In the second experiment (Exp. 2), the environment has four rooms, but the size of each room was slightly wider (Fig. 2b). The EOPs are also specified by Formula (15) using color as in Exp. 1. The second environment looks simpler, and the total number of events occurring there is smaller. However, this necessitates more energy-saving behaviors by agents.

The number of agents $|A|$ is 20, and the charging bases for all agents are placed at the center $(0,0)$. Agents leave the charging bases with full batteries, patrol the environment, return to the charging bases before the batteries are dead, and repeat this cycle of activities. We set the battery capacity to $B_{max} = 900$, and the charging speed constant $k_{charge} = 3$. Thus, it takes 2700 steps for the battery to fully charge from 0. Therefore, the maximum activity cycle time is 3600 steps; hence, we set the data collection interval $t_e - t_s$ for calculating $D(s)$ and $C(s)$ to 3600. We also set the quality requirement value to $D_{req} = 600$, and the initial value of the parameter K^i is 1.0. The values of $D(s)$ and $C(s)$ shown below were the average values of 50 independent trials.

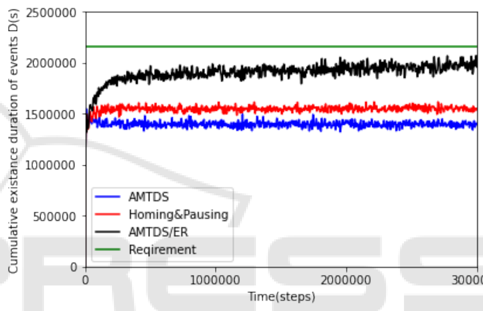


Figure 3: $C(S)$ over time in Exp. 1.

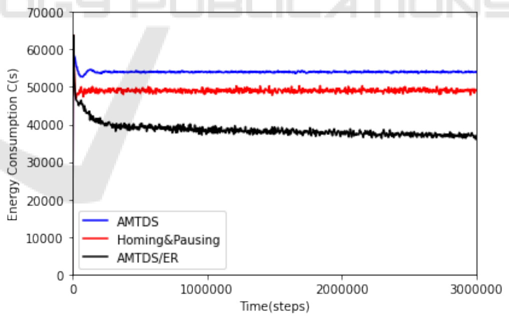


Figure 4: $D(S)$ over time in Exp. 1.

5.2 Performance Evaluation

Figures 4 and 4 present a plot of the transition of time left without checking events $D(s)$, and total energy consumption $C(s)$ every 3600 steps, respectively. Note that as $D_{req} = 600$, agents are required to keep $D(s) \le 2160000$ $(= 3600 D_{req})$ in Fig. 4. Figure 4 shows that all methods fulfilled the quality requirement. In particular, the conventional methods, AMTDS and AMTDS/ESC, maintained the $D(s)$ to considerably smaller values than the required value.

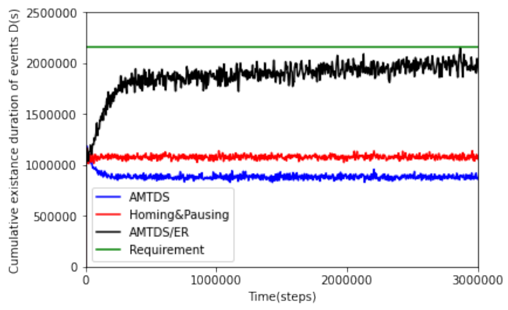


Figure 5: $C(S)$ over time in Exp. 2.

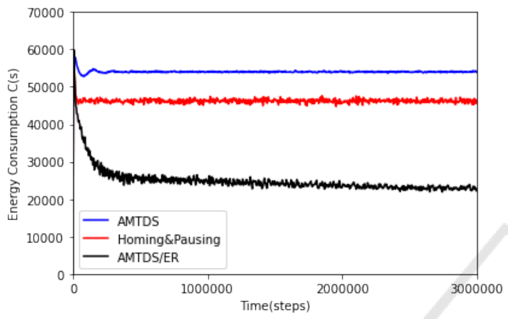


Figure 6: $D(S)$ over time in Exp. 2.

However, this suggested excessive energy consumption by patrolling beyond the requirements, and this situation is illustrated in Fig. 4. This figure indicates that the proposed method AMTDS/ER could reduce total energy consumption $C(s)$ by approximately 30.9% compared to AMTDS, although AMTDS/ER increased the value of $D(s)$ approximately by 40.3%, where these data are the average values between 2,000,000 and 3,000,000. AMTDS/ESC could also reduce energy consumption, but the reduction was limited; AMTDS/ER could reduce $C(s)$ approximately by 24.0% compared to AMTDS/ESC and increase $D(s)$ approximately by 26.2%. The increase in effective reduction can be attributed to the introduction of learning parameters K^i because it enabled each agent to predict the effect of other agents' contribution during its stops for energy-saving behaviors to some extent.

Similar results can be observed in Exp. 2, as shown in Figs. 6 and 6, which present plots of $D(s)$ and $C(s)$ over time in Environment 2. In this environment, the sum of the EOP values is much lower than that in Environment 1, and we should confirm if agents could take more energy-saving behaviors to reduce more energy than that in Exp. 1. Agents could achieve such expected behaviors; Fig. 6 shows that agents gradually increased the energy-saving behaviors by fulfilling the requirement while they decreased the consumed energy, as shown in Fig. 6. Thus, agents

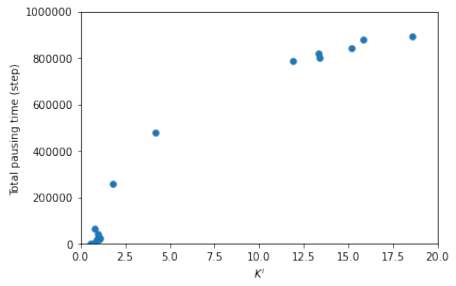


Figure 7: Parameter K^i and total pausing time (Exp. 1).

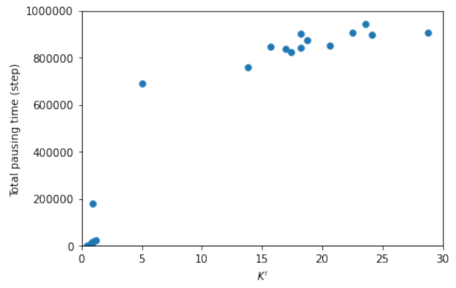


Figure 8: Parameter K^i and total pausing time (Exp. 2).

with AMTDS/ER consumed energy that was almost half of that consumed by agents with AMTDS/ESC.

5.3 Analysis of Behavior

We analyzed the characteristics of agents' energy-saving behaviors, particularly the sum of pausing times, which is partly affected by the value of the learning parameter K^i in each agent. We calculated the sum of pausing times from 2,000,000 to 3,000,000 steps for $\forall i \in A$ and plotted the relationship between this sum and K^i at 3,000,000 step in Figs. 7 (Exp. 1) and 8 (Exp. 2). Note that these scattered graphs were plotted using data obtained in one experimental trial randomly selected from Exp. 1 and Exp. 2, with no special intent.

Both of these graphs suggest that the agents can be divided into roughly two clusters; one with relatively large values of K^i (e.g., $K^i \geq 10$) and a large sum of pausing times, and another one with small values of K^i (e.g., $K^i \leq 3$) and a relatively small sum of pausing times. We called the former cluster the *energy-saving group* while the latter is called the *busy group*. The findings of this analysis are as follows. First, because the length between 2,000,000 and 3,000,000 is 1,000,000, agents in the energy-saving group that paused more than 800,000 times rarely patrolled the environment. The non-pausing time was less than 200,000, but the charging speed constant $k_{charge} = 3$. Thus, their actual patrolling time was less than 50,000.

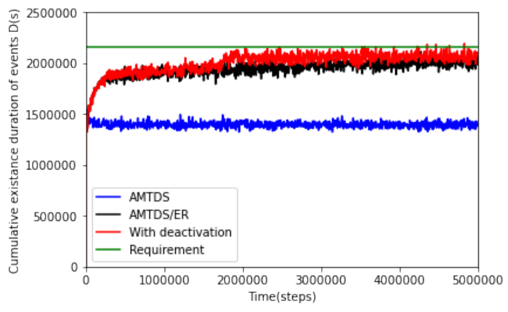


Figure 9: $C(S)$ over time in Exp. 3.

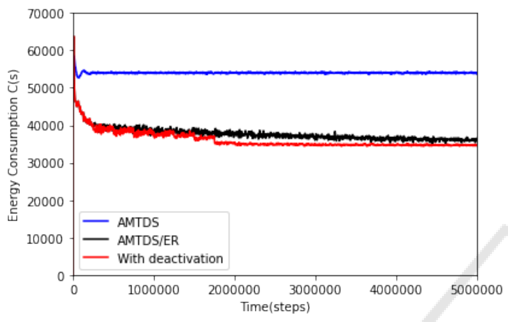


Figure 10: $D(S)$ over time in Exp. 3.

Meanwhile, most agents in the busy group rarely took the pausing behaviors. Note that homing behavior is only returning to the charging base regardless of the remaining capacity, shortening its time for charging. Therefore, the homing behaviors did not directly contribute to the energy savings. The agents in the busy group performed most of the required patrol to fulfill the requirement, whereas the remaining agents in the energy-saving group increased their pausing time, and this type of differentiation occurred through individual learning.

5.4 Evaluation of Deactivation

We have already described the deactivation method in Section 4.4 based on the analysis in the previous section. We conducted the third experiment (Exp. 3) under the same setting as Exp. 1 to evaluate the proposed deactivation method, i.e., to determine whether the number of agents patrolling can be reduced to reduce the energy consumption while fulfilling the quality requirement. As listed in Table 1, we checked the possibility of deactivation every $D_{int} = 250,000$.

Figures 10 and 10 show the transition of $D(s)$ and $C(s)$, respectively, until $5,000,000$ steps, where label “With deactivation” indicates the AMTDS/ER with the proposed deactivation method. In this particular experiment, the number of deactivated agents ranged between six and eight, mostly seven. Note again that

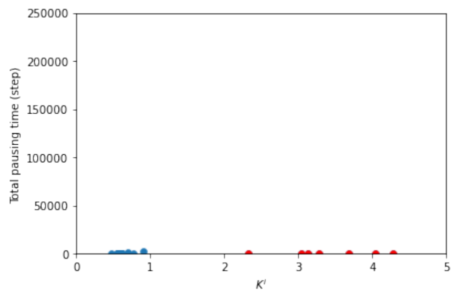
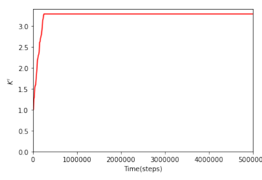


Figure 11: Parameter K^i and total pausing time (Exp. 3).


(a) $D = 250,000$ steps.

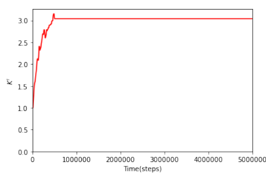
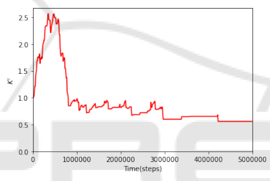

(b) $D = 500,000$ steps.

Figure 12: Value of K^i in agents not deactivated.


(a) Example 1.

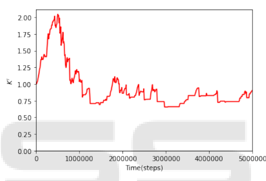

(b) Example 2.

Figure 13: Value of K^i in the agents deactivated at D steps.

smaller values of $D(s)$ and $C(s)$ are better.

Figure 10 indicates that the proposed AMTDS/ER with the deactivation method could raise $D(s)$ more while fulfilling the quality requirement compared to AMTDS/ER without deactivation, although their difference was small. This is because AMTDS/ER already maintained the value of D(s) near the required quality level. The consumed energy was also slightly reduced, as shown in Fig. 10. However, the main difference between with and without deactivation scenarios is that agents that were deactivated could completely transfer their patrolling tasks to other agents. Furthermore, we would like to emphasize that even though the performance differences in $D(s)$ and $C(s)$ are not large, the advantages of deactivation are that we can (1) use the deactivated agents in other environments, (2) leave those agents on standby and use them in case of failure, or (3) implement periodic inspections and alternating operations to extend the life of the system.

5.5 Behaviors of Active and Deactivated Agents

We analyzed the behaviors of active and deactivated agents by investigating the transition of K^i. We plotted, in Fig. 11, the relationship between K^i and the total pausing time of i in one independent trial that was randomly selected from Exp. 3, where K^i is the value at 5,000,000 steps, and the total pausing time is the sum of the pause times during the last 1,000,000 steps of the experiment. Therefore, K^i of a deactivated agent i was the value when it was deactivated, and the total pausing time of i must be zero. Seven red plots correspond to deactivated agents. This figure indicates that deactivated agents had relatively high values of K^i and active agents continued patrolling with a very short pausing time. This implies that thirteen agents were sufficient to maintain the required quality set in this experiment.

Figures 13 and 13 present plots of the transition of K^i for agents that were deactivated at 250,000 and 500,000 steps (Fig. 13) and active agents (Fig. 13). We can see from Fig. 13 that the K^i values of these agents rapidly increased and were then selected for deactivation because the sum of its pausing time was the largest at that time. Subsequently, their K^i did not change. Meanwhile, active agents temporally increased their values of K^i, but after that, the K^i decreased and kept the relatively lower values because some agents were deactivated. Thus, the active agents must patrol more in their stead to fulfill the requirement.

Our experimental results show that the proposed method can reduce energy consumption more than the previous method (Wu et al., 2019) by introducing the individual learning parameter K^i. We further reduced the number of operating agents based on the value of K^i while fulfilling the quality requirements.

6 CONCLUSIONS

Although conventional studies for MACPP usually consider maximizing efficiency, it is often desirable to reduce the energy used while fulfilling the quality requirement. Therefore, we proposed a method that can save energy while meeting the requirement in MACPP. The basic idea of our method is that each agent individually foresees the state of the environment and assesses its own behavior from individual viewpoints. Then, it returns to the charging base or pauses for a while to save energy on determining that it is not necessary to start patrolling immediately. Moreover, we found that the proposed individual learning clusters the agents into two groups, and by using these findings, we could successfully deactivate redundant agents to reduce the number of working agents required for the current requirement.

This study assumed that the event probability (EOP) $p(v)$ in the environment is known, but this is not always possible. Therefore, our future work is to propose a method to reduce energy consumption while agents learn the EOP in the environment.

ACKNOWLEDGEMENTS

This work was supported by JSPS KAKENHI Grant Number 20H04245.

REFERENCES

Benkrid, A., Benallegue, A., and Achour, N. (2019). Multi-robot coordination for energy-efficient exploration. *Journal of Control, Automation and Electrical Systems*, 30(6):911–920.

Hattori, K. and Sugawara, T. (2021). Effective Area Partitioning in a Multi-Agent Patrolling Domain for Better Efficiency. In *Proceedings of the 13th International Conference on Agents and Artificial Intelligence - Volume 1: ICAART,*, pages 281–288. INSTICC, SciTePress.

Kim, J., Dietz, J. E., and Matson, E. T. (2016). Modeling of a multi-robot energy saving system to increase operating time of a firefighting robot. In *2016 IEEE Symp. on Technologies for Homeland Security (HST)*, pages 1–6.

Latif, E., Gui, Y., Munir, A., and Parasuraman, R. (2021). Energy-aware multi-robot task allocation in persistent tasks.

Ma, H., Li, J., Kumar, T. S., and Koenig, S. (2017). Lifelong multi-agent path finding for online pickup and delivery tasks. In *Proc. of the 16th Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '17, pages 837–845, Richland, SC. IFAAMAS.

Notomista, G., Mayya, S., Emam, Y., Kroninger, C., Bohannon, A., Hutchinson, S., and Egerstedt, M. (2022). A resilient and energy-aware task allocation framework for heterogeneous multirobot systems. *IEEE Transactions on Robotics*, 38(1):159–179.

Othmani-Guibourg, M., El Fallah-Seghrouchni, A., Farges, J.-L., and Potop-Butucaru, M. (2017). Multi-agent patrolling in dynamic environments. In *2017 IEEE International Conference on Agents (ICA)*, pages 72–77. IEEE.

Othmani-Guibourg, M., Fallah-Seghrouchni, A. E., and Farges, J.-L. (2018). Decentralized multi-agent patrolling strategies using global idleness estimation. In *International Conference on Principles and Practice of Multi-Agent Systems*, pages 603–611. Springer.

Tevyashov, G. K., Mamchenko, M. V., Migachev, A. N., Galin, R. R., Kulagin, K. A., Trefilov, P. M., Onisimov, R. O., and Goloburdin, N. V. (2022). Algorithm for multi-drone path planning and coverage of agricultural fields. In *Agriculture Digitalization and Organic Production*, pages 299–310. Springer.

Wiandt, B. and Simon, V. (2018). Autonomous graph partitioning for multi-agent patrolling problems. In *2018 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 261–268. IEEE.

Wu, L. and Sugawara, T. (2019). Strategies for Energy-Aware Multi-agent Continuous Cooperative Patrolling Problems Subject to Requirements. In *PRIMA 2019: Principles and Practice of Multi-Agent Systems*, pages 585–593, Cham. Springer International Publishing.

Wu, L., Sugiyama, A., and Sugawara, T. (2019). Energy-efficient strategies for multi-agent continuous cooperative patrolling problems. *Procedia Computer Science*, 159:465–474.

Yamauchi, T., Miyashita, Y., and Sugawara, T. (2022). Standby-Based Deadlock Avoidance Method for Multi-Agent Pickup and Delivery Tasks. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pages 1427–1435, Richland, SC. IFAAMAS.

Yoneda, K., Kato, C., and Sugawara, T. (2013). Autonomous learning of target decision strategies without communications for continuous coordinated cleaning tasks. In *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, volume 2, pages 216–223.

Zhou, X., Wang, W., Wang, T., Lei, Y., and Zhong, F. (2019). Bayesian reinforcement learning for multi-robot decentralized patrolling in uncertain environments. *IEEE Transactions on Vehicular Technology*, 68(12):11691–11703.