Vol.2019-ICS-194 No.8
2019/3/10

Strategies for Energy-Efficient Multi-Agent Continuous Patrolling Tasks

Lingying Wu[1,a)] Ayumi Sugiyama[1,b)] Toshiharu Sugawara[1,c)]

Abstract: This paper proposes an autonomous learning method of target decision strategies and coordinated behavior aiming at reducing energy cost on the premise of satisfying quality requirements in continuous patrolling tasks by multiple cooperative agents. There is always a trade-off between energy efficiency and level of perfection. In real-world applications, it is usually more desirable to minimize the cost of energy and carry out the tasks to the required level of quality instead of fulfilling tasks perfectly by ignoring energy efficiency. We extended our previous method of target decision strategies learning, by incorporating a number of behavioral strategies, with which agents individually estimate whether the requirement is reached and decide their action plan to reduce energy consumption. We experimentally show that agents with the proposed methods learn to decide the strategies based on energy consumption and performance measure for event occurrence, which is the amount of vacuumed dirt in the case of the cleaning task, and are able to reduce energy consumption while cooperatively maintain the given requirements of quality.

Keywords: Multi-agent systems, Learning, Continuous Cleaning, Energy Efficiency

1. Introduction

Thanks to modern advances in robotics and computational technologies, robot applications have gained popularity in real-world environments. From the 1990s, application mode of robots has been changing from cell to system, resulting in cooperative robotics becoming an important field in artificial intelligence [1]. The performance of single-robot systems are limited due to restrictions of speed, movement and battery capacity. In contrast, multi-robots could complete tasks by compensation and cooperation even if work environment changes or partial system failure occurred. Accordingly, coordination and cooperation by multiple independent robots have recently attracted considerable attentions.

In this study, we aimed at the continuous area cleaning domain by multiple agents, which are software that control robots to repeatedly visit and clean the given area with non-uniform frequencies of visit. To ensure continuous performance of the robots, periodical recharge is required. The most challenging issue of our research is that we assume shallow coordination between agents instead of sophisticated coordination, because agents may only have CPUs with limited performance, constrained amount of memory, and restricted battery capabilities. As might be expected, it is reasonable to assume that agents must decide their action and strategies based on limited information such as local viewpoints, personal knowledge and narrow communication bandwidth.

Recent applications lay emphasis on effectiveness, performance requirement and energy efficiency, but there is a trade-off between energy efficiency and level of perfection. In real-world applications, rather than fulfilling the tasks perfectly by ignoring energy cost, usually we place a higher value on reduction of energy cost. For instance, in the area cleaning task we focus on, it is not necessary to keep the cleanliness of environment extremely low all the time. Instead, we would prefer the agents cooperatively satisfying the given requirement of cleanliness with the least possible cost of energy.

In this paper, we extend our previous method *adaptive meta-target decision strategy* (AMTDS) [2] so that agents with the extended method select the appropriate target decision strategies by monitoring the local energy consumption as well as checking if the expected value of remaining dirt amount. Furthermore, we also propose methods for requirement estimation and self-contribution evaluation. With these methods, agents are capable of estimating whether the given requirement level is reached or not on their own, and understanding the importance of themselves with regard to the system. When agents judge that the requirement is satisfied, they would consider to take energy-saving action depending on their importance (or the contribution degree), which is decided by their recent and expected performance.

On top of that, we introduce two behavioral strategies for agents to take in substitution for moving on to the next target. Taking the *return* action, agents stop patrolling and head toward the charging base to charge. Taking the *wait* action, agents rest at the charging base and do not move around for a certain interval of time. We experimentally show that agents with these behavior could successfully save energy while keeping the cleanliness of the environment around the given requirement levels.

The paper is organized as follows: Section 2 discusses some

[1] Waseda University, Shinjuku, Tokyo 169-8555, Japan
[a)] lingying.wu@isl.cs.waseda.ac.jp
[b)] a.sugiyama@isl.cs.waseda.ac.jp
[c)] sugawara@waseda.jp

related work. Section 3 describes the models of environment and agents, including strategies for selecting targets and generating paths, and then, explains the definition of performance measures to clarify the main purpose of our work. Section 4 introduces our proposals called the *adaptive meta-target decision strategy for energy saving and cleanliness* (AMTDS/ESC), which is a variation of AMTDS proposed by Yoneda et al. [2], requirement estimation, self-contribution evaluation, and energy-saving behavior. Section 5 shows the experiments we conducted to evaluate our methods. The results indicated that our methods enabled agents to individually select target decision strategies and to reduce cost of energy while cooperatively maintain the requirement levels of quality. Finally, Section 6 summarizes the result and gives some work left to do in the future.

2. Related Work

Several studies dealing with continuous patrolling problems have been conducted. Ahmadi and Stone [3] defined the formulation of continuous area sweeping task, and introduced an initial approach that non-uniformly visits the environment to minimize the estimated cost. They [4] then extended the approach to multi-robot scenario, which conducts area partitioning by negotiation between agents. Moreira et al. [5] argued that multi-agent patrolling can be a good benchmark for multi-agent systems, and proposed a software simulator constructed strictly for the patrolling tasks. Santana et al. [6] solved the multi-agent patrolling problem using reinforcement learning by automatically adapting the strategies of agents to the environment.

As previous work, Yoneda et al. [7] proposed the autonomous reinforcement learning of the meta-strategy to decide the target decision strategies for coordination, called the AMTDS as mentioned in the previous section. With this method, agents investigate different strategies and individually identify the most effective ones in respect of quality. In other words, agents learn to select the strategies which result in vacuuming larger amount of dirt. Yoneda et al. [2] and Sugiyama et al. [8] improved the method by incorporating self-monitoring and environmental learning to avoid performance degradation due to over-selection and make the method more practical. Sugiyama et al. [9] further extended the method for prompting autonomous division of labor by introducing simple communication to negotiate for task allocations. However, energy cost was not taken into consideration in these studies, so that agents keep working and target on minimizing the amount of dirt in the environment even if it has been almost cleaned.

3. Model Description

In this paper, we focus on cleaning task by multiple autonomous agents and use a *continuous cooperative patrolling problem* (CCPP) model [9], in which agents move around and visit locations with required and different frequencies for given purposes. In our model, agents are required to minimize the energy cost which satisfying given requirements. In addition, to ensure continuous performance, it is necessary for agents to periodically return to their bases and charge.

3.1 Environment

The environment in which agents move and work is described by graph $G = (V, E)$, where $V = \{v_1, ...v_m\}$ is the set of nodes with x and y coordinates, and E is the set of edges. The length of each edge in E is assumed to be one so that any graph can be expressed by adding dummy nodes if necessary. We introduce a discrete time unit called *tick*. In one tick, any agent can move to one of the neighboring nodes and work on the node it visits.

Each node owns a value of probability of event occurrence denoted as P_v for node $v \in V$. In the case of the cleaning task, an event corresponds to the accumulation of dirt, so that P_v represents the probability that one piece of dirt is accumulated at v per tick. A high value of P_v means that v is a dirtier node at which dirt easily accumulates. When an agent visits a node, the accumulated dirt is cleaned. Therefore, the amount of dirt at v at time t can be expressed as $L_t(v)$, which is updated based on P_v every tick as

$$L_t(v) \leftarrow \begin{cases} 0 & \text{if an agent has visited } v \text{ at } t, \\ L_{t-1}(v) + 1 & \text{if a piece of dirt appears} \\ & \text{with probability } P_v \text{ at } t, \\ L_{t-1}(v) & \text{otherwise.} \end{cases} \tag{1}$$

3.2 Agent

Let $A = \{1, ...n\}$ be a set of agents, and $v^i(t) \in V$ be the position of agent $i \in A$ at time t. For simplification, we assume that agents know the structure of environment $G = (V, E)$ and that multiple agents staying at the same place is allowed. Although these assumptions often do not hold in real-world applications, a number of efficient algorithms for map creation [10] and collision avoidance [11] have been proposed, which can be used to simplify our model.

We assume an application environment where agents are equipped with indicators, such as infrared emission and reflecting devices, so that in addition to having a map of the environment, agents are capable of getting their own and other agent's positions. On the other hand, since agents only have a few resources including limited CPU power and battery capacity, sophisticated coordination should be avoided. Each agent decides action plans based on local view and shallow coordination, by which they can only exchange superficial data such as past and current locations, but do not acquire deep knowledge such as other's plans of actions and long-term targets.

In this paper, agents are given the probability of dirt accumulation $\{P_v | v \in V\}$. Of course, they do not know the actual amount of accumulated dirt, $L_t(v)$, but instead, they can estimate it by calculating the expected value, $EL_t(v)$. The expected value of $L_t(v)$ at any future time t is defined as

$$EL_t(v) = P_v \cdot (t - t^v_{visit}), \tag{2}$$

where t^v_{visit} is the most recent time when an agent (may not be i) visited and cleaned v. Agents can know t^v_{visit} since they have exchanged their locations with others following the above assumption.

A battery in agent i is denoted by $B^i = (B^i_{max}, B^i_{cons}, k^i_{charge})$, where $B^i_{max} > 0$ is the maximal capacity of the battery, $B^i_{cons} > 0$

is the amount of battery consumption per tick, and parameter $k^i_{charge} > 0$ indicates the speed of charge. Let $b^i(t)$ represents the remaining capacity of battery in i at time t. When i moves, $b^i(t)$ is updated by

$$b^i(t+1) \leftarrow b^i(t) - B^i_{cons} \tag{3}$$

every tick. When i charges its battery at the charging base, v^i_{base}, the required time for a full charge starting from t is proportional to the amount of consumed battery:

$$T^i_{charge}(t) = k^i_{charge} \cdot (B^i_{max} - b^i(t)). \tag{4}$$

We assume that agents consume B^i_{cons} every time they move, regardless the amount of vacuumed dirt. Accordingly, the amount of energy consumption by agent i from time $t-1$ to time t is defined as

$$E_t(i) = \begin{cases} 0 & \text{if } i \text{ is charging or stays at the same} \\ & \text{place at } t, \\ B^i_{cons} & \text{otherwise.} \end{cases} \tag{5}$$

The parameters B^i_{max}, B^i_{cons} and k^i_{charge} can be independent of i, but they are assumed to be the same in this paper for simplicity. According to the above assumption, periodical return to charging bases is required for agents to ensure continuous patrol, which means that they must return to v^i_{base} before $b^i(t)$ becomes zero.

3.3 Plan Creation in Agents

The plan creation of agents can be divided into three stages: *action selection*, *target decision* and *path generation*. First, agent i selects the action to take next, which is the main part of our proposed methods and will be explained in detail in Section 4. When selected to move, the agent then decides the target node $v^i_{tar} \in V$ and generated the appropriate path from current node to $v^i_{tar} \in V$. There are lots of algorithms to determine targets and paths. We use several simple strategies since our main purpose was not about to propose planning algorithms.

3.3.1 Target Decision Strategies

Agent i decide the next target node $v^i_{tar} \in V$ based on (1) which node is expected to accumulate the largest amount of dirt and (2) which node is unlikely to be visited by other agents in short time. Our propose extend the AMTDS [2], by which each agent learns the appropriate strategy based on Q-learning from the following four strategies.

Random Selection (R):

Agent i randomly selects v^i_{tar} from V.

Probabilistic Greedy Selection (PGS):

For positive integer N_g and time t, let $V^t_g \subset V$ be the set of N_g nodes with the highest values of $EL_t(v)$. Agent i randomly selects v^i_{tar} from V^t_g. Note that randomness is introduced to avoid concentration of targets by multiple agents.

Repulsive Selection (RS):

Agent i selects the node which has the longest summative distance from all agents. Let V^i_{rep} be the set of $N_{rep} > 0$ nodes which i randomly selected from V, and $d(v_i, v_j)$ be the length of minimum path between v_i and $v_j \in V$. Then v^i_{tar} is decided as

$$v^i_{tar} = \underset{v \in V^i_{rep}}{\arg\min} \sum_{i \in A} d(v^i(t), v). \tag{6}$$

Balanced Neighbor-Preferential Selection (BNPS):

BNPS is an advanced version of PGS. The basic idea is that if agent i estimates there are dirty nodes in the neighborhood using learned threshold, i selects v^i_{tar} from those nodes. Otherwise, i selects v^i_{tar} using PGS. Since explanation of BNPS is beyond the scope of this paper, please refer to [7] for details.

3.3.2 Path Generation Strategy

Before agent i generates the path to v^i_{tar}, it will check the remaining capacity of its battery in advance to see if v^i_{tar} is reachable. Otherwise, i changes v^i_{tar} to its charging base v^i_{base}, and generates a path to return and charge.

We consider the *gradual path generation* (GPG) method as path generation strategy. In general, agents move along the shortest path, but if there are dirtier nodes near the path, they drop by and clean them. Since the explanation of GPG is beyond the scope of this paper, please refer to [7] for more details. We chose this method as the previous research [2] has shown that GPG always outperformed the simple shortest path strategy.

3.4 Performance Measures

Our purpose is to minimize the overall energy consumption on the premise of maintaining quality requirements, which is the total dirt amount in the environment. Therefore, we evaluate our proposed methods in two aspects: *cumulative existence duration of dirt* and *total energy consumption*.

The cumulative existence duration of dirt at certain intervals of time is defined as

$$D_{t_s,t_e} = \frac{\sum_{v \in V} \sum_{t=t_s+1}^{t_e} L_t(v)}{t_e - t_s}, \tag{7}$$

and the total energy consumption of all agents at certain intervals of time is defined as

$$C_{t_s,t_e} = \frac{\sum_{i \in A} \sum_{t=t_s+1}^{t_e} E_t(i)}{t_e - t_s}, \tag{8}$$

where positive integers t_s and t_e are the start and end times of the interval with $t_s < t_e$.

Although smaller values of D_{t_s,t_e} and C_{t_s,t_e} are better, there is a trade-off between them. In our case, rather than keeping the environment extremely clean, it is more desirable to clean the environment to the required extent using less energy.

Given a requirement level of dirt cumulative existence duration $D_{req} > 0$, instead of minimizing D_{t_s,t_e}, agents aim at minimizing C_{t_s,t_e} and making D_{t_s,t_e} small enough to satisfy the condition $D_{t_s,t_e} \le D_{req}$.

4. Proposed Method

Our proposed methods cover the first and second stages of agent plan creation. First, we describe the methods for estimating requirement and evaluating self-contribution, with which agents decide the next action by taking into account the status of environment and themselves. Next, we propose two behavioral strategies taken by agents as a substitute for moving toward the next target with the intention of decreasing energy cost. Finally, we present a

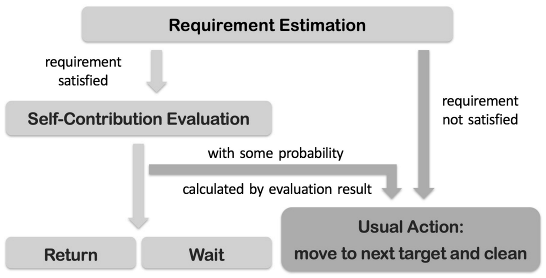
Requirement Estimation
requirement satisfied
requirement not satisfied
Self-Contribution Evaluation
with some probability
calculated by evaluation result
Return
Wait
Usual Action: move to next target and clean

Fig. 1 Plan Creation of Agents.

variation of the previous method AMTDS [2]. **Fig. 1** presents an overview on plan creation of agents with the proposed methods.

4.1 Requirement Estimation

At the first stage of plan creation, agents decide the action to take. In order to do so, it is necessary for agents to know the current status of the environment. They estimate the total dirt amount of the environment, then decide whether the given requirement is satisfied or not. For agent i at time t, i randomly selects $N_{range} > 0$ nodes from the set of nodes it has visited and form $V_{rand}(v^i(t))$. Each agent estimates on their own by calculating the average value of expected accumulated dirt in $V_{rand}(v^i(t))$:

$$EV_t^i = \frac{\sum_{v \in V_{rand}(v^i(t))} EL_t(v)}{N_{range}} \tag{9}$$

When EV_t^i is smaller than the value of given requirement $D_{req}^{e,|A|} > 0$ for environment e, i considers the requirement is reached. It then decides the next action to take based on the result of *self-contribution evaluation*, which will be explained in the following sections. Otherwise, i simply selects the next target and generates path to the destination.

4.2 Self-Contribution Evaluation

For agents to select their next action, they evaluate their self-contribution with the aim of understanding how important they will be for the system. They consider themselves as important by taking into account (1) their recent performance and (2) whether they find the dirty regions and are possible to vacuum large amount of dirt in the future.

Agent i evaluates its local performance $Imp^i(t)$ by comparing the average values of vacuumed dirt in the past during long and short terms, and the average value of expected vacuum (accumulated) dirt in the future. The average values in the past U_p^i (including U_s^i for short term and U_l^i for long term) and future U_f^i are calculated respectively by

$$U_p^i = u_{t_0,t_c}^i = \frac{\sum_{t_0<t\le t_c} L_t(v^i(t))}{t_c - t_0},\ t_0 = \begin{cases} t_c - T_s & \text{short term,} \\ t_c - T_l & \text{long term.} \end{cases} \tag{10}$$

$$U_f^i = u_{t_f,t_c}^i = \frac{\sum_{t_c<t\le t_f} EL_t(v^i(t))}{t_f - t_c}, \tag{11}$$

where t_c is the current time, T_s and T_l ($T_s < T_l$) are fixed integers, t_f is the future time when i arrives at the next target, $L_t(v^i(t))$ is the amount of vacuumed dirt by i at time t, and $EL_t(v^i(t))$ is the expected value of dirt amount at future time t. Note that in order to calculate U_f^i, the next target node is decided in advance. Last of all, the value of importance is obtained by

$$Imp^i(t) = \begin{cases} \frac{U_s^i + U_f^i}{U_l^i} & \text{if } U_s^i + U_f^i \le U_l^i \\ 0 & \text{if } U_l^i = 0, \\ 1 & \text{otherwise.} \end{cases} \tag{12}$$

The above equations indicate that the agent is considered important for the system when it has cleaned more in recent times and find the dirty regions so is expected to vacuum large amount of dirt in the future. In contrast, if the agent gave poor performance in the past and is not expected to vacuum large amount of dirt, it will identify itself as not important. As a result, the less important an agent is, the higher probability it will take one of the energy-saving actions. Note that the self-contribution evaluation is conducted only after agents estimate requirement and suppose the requirement level is reached.

4.3 Energy-Saving Action

We proposed two behavioral strategies aiming at saving energy, *Return* and *Wait*, that agents take in substitution for usual action, which is moving to the next target. An agent will have a chance to take these action only when it supposes that the given requirement is satisfied based on estimation. Note that only one of the two energy-saving action is applied in one experiment. The probability of taking the energy-saving action *act* is calculated based on the result of self-contribution evaluation by

$$P_{act}^i(t) = 1 - Imp^i(t). \tag{13}$$

4.3.1 Return Action

Every time after agent i continuously moves for $T_{check} > 0$ ticks, it conducts requirement estimation to decide whether to take the *Return* action. In addition, i also checks the remaining capacity of battery $b^i(t)$, so that the action will only be taken when $b^i(t)$ is lower than k_{return} of B_{max}^i.

Taking the *Return* action, agent i immediately and directly goes back to the charging base v_{base}^i and starts to charge. On its way back, the agent keeps cleaning and does not conduct requirement estimation or self-contribution evaluation.

4.3.2 Wait Action

Agent i conducts requirement estimation at the charging base every time after its battery is fully charged. According to the result, i decides whether to take the *Wait* action. If yes, the agent simply stays at v_{base}^i for $T_{wait} > 0$ ticks without cleaning. There is no limitation to the number of taking *Wait* action continuously, meaning that it is possible for an agent to take *Wait* action again and again based on the estimation results.

4.4 Autonomous Strategy Selection

Since the main purpose of this paper is to reduce overall energy cost, we extend AMTDS and call the new methods *AMTDS for energy saving and cleanliness* (AMTDS/ESC). With the new method, agents choose appropriate target decision strategies for coordinated cleaning tasks from (1) the amount of dirt vacuumed up in the past and (2) the amount of energy consumption in the past by using reinforcement learning. A larger value of vacuumed dirt amount and a smaller value of energy cost are preferred.

Suppose that agent i selects the next target v^i_{tar} with strategy s, where s is one of the target decision strategies described in previous sections. After i moves to v^i_{tar} along the path generated by GPG, it calculates the reward of s by

$$u^i_{t_0,t_0+d_{travel}} = \frac{\sum_{t_0<t\leq d_{travel}} L_t(i) / \sum_{t_0<t\leq d_{travel}} E_t(i)}{d_{travel}}, \qquad (14)$$

where d_{travel} is the length of travel from time t_0 when i started until the time it arrived at its target. Finally, the Q-value of s is updated as

$$Q(s) \leftarrow (1-\alpha)\cdot Q(s) + \alpha \cdot u^i_{t_0,t_0+d_{travel}}. \qquad (15)$$

As noted above, agents with AMTDS/ESC learn to select the strategy which minimizes the energy cost and maximizes the amount vacuumed dirt per tick at the same time. In addition, the ε-greedy method is used during learning.

5. Experiments and Discussion

We evaluated the proposed methods in a simulation environment similar to our previous work [2]. By comparing the performance measures to previous methods, we introduced self-contribution evaluation and energy-saving strategies to the process of plan creation, and show that our methods enable agents to cooperatively reduce energy consumption while keeping the given requirements satisfied.

5.1 Experimental Setting

Four environments with different characteristics are prepared for the experiments in order to observe the behavior of agents. Each environment is represented by a two-dimensional grid graph, where G is defined as a 51×51 grid. Node v is expressed by (x, y), where $-50 \leq x, y \leq 50$. The charging bases for all agents are at the same place so that $v_{base} = v^i_{base} = (0, 0)$, and multiple agents can charge simultaneously.

Some environments have regions where dirt easily accumulated, which are considered important, and agents would like to focus on visiting them. The coordinates and shapes of these regions are outlined in **Fig. 2**. In Env.(a), dirt is accumulated uniformly and we define $P_v = 0$ or 5×10^{-6} for any node v, and the values were randomly chosen for each node at the beginning of all experiments. Dirt easily accumulated near the wall in Env. (b), while Env. (d) contains several independent block regions which are easy to get dirty compared to other places. Env. (c) is the most complicated environment, which has both of the above characteristics. Accordingly, P_v for $v \in V$ in Envs. (b)-(d) is defined as

$$P_v = \begin{cases} 10^{-3} & \text{if } v \text{ is in the red regions,} \\ 10^{-4} & \text{if } v \text{ is in the gray regions,} \\ 10^{-6} & \text{otherwise.} \end{cases} \qquad (16)$$

We deployed 20 agents in the environments, and all of them are assumed to be homogeneous: they use the same strategy for path generation and select one of the five target decision strategies (R, PGS, RS, BNPS, and AMTDS/ESC). With strategies other than

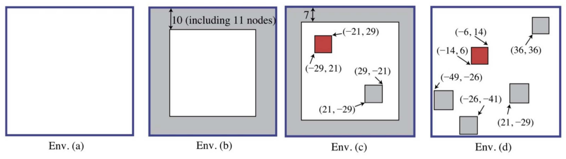

Fig. 2 Experimental environments [2]

Table 1 Parameters for target decision strategies

Methods	Parameters	Values
PGS	N_g	5
RS	N^i_{rep}	100
BNPS	α	0.1
	d_{th}	15
AMTDS/ESC	α	0.1
	ϵ	0.05

AMTDS/ESC, agents always use a single target decision strategy. In contrast, agents with AMTDS/ESC independently select one from R, PGS, RS, or BNPS based on local learning. Note that in all experiments, the probability of dirt accumulation $\{P_v | v \in V\}$ is assumed to be given.

About batteries in agents, we set $B_{max} = 2700$, $B_{cons} = 3$, and $k_{charge} = 1$ in all the experiments for every agent. Consequently, agents could continuously operate up to 900 ticks and require 2700 ticks for a full charge when the battery is running out of power, which makes the maximum cycle of operation and charge 3600 ticks. Therefore, when all the agents constantly work to make full use of their batteries, the theoretical value of total energy consumption per tick C_{t_s,t_e} will be around 15. In the following experiments, we compared the resulting C_{t_s,t_e} to this value to evaluate our proposed methods.

The main purpose of our research is to reduce the cost of energy. In order to do so, agents estimate the status of the environment and importance of themselves. According to the results, agents take one of the *Return* and *Wait* action instead of heading toward the next target under certain circumstances. The decision of action selection is individually made by each agent with their local viewpoints.

We compared the values of D_{t_s,t_e} and C_{t_s,t_e} every 100 ticks. The parameter values used in target decision strategies are listed in **Table 1**. These values were determined by taking into account the size of experimental environments and the number of agents, but are not optimal. The experimental results below were the averages of several independent trails with different random seeds, where the length of each trial was 500,000 ticks.

5.2 Energy-Saving Strategies

We compared the performance results of three different agent behavioral regimes. In the first experiment, agents only take usual action and do not care about energy efficiency. In the second experiment, there are chances for agents to take *Return* action instead of usual behavior so that they stop patrolling and go back to charging base and charge. In the third experiment, with some probability agents take *Wait* action after their batteries are fully charged so that rather than leaving for patrol, they rest at the charging base for a while.

Table 2 Parameters for energy-saving strategies

Methods	Parameters	Values
Requirement Estimation	N_{range}	100
	$D^{a,20}_{req}$	45
	$D^{b,20}_{req}$	600
	$D^{c,20}_{req}$	400
	$D^{d,20}_{req}$	110
Self-Contribution Evaluation	T_s	20
	T_l	50
Return Action	T_{check}	100
	k_{return}	$\frac{1}{3}$
Wait Action	T_{wait}	20

The parameter values used for requirement estimation, self-contribution evaluation and energy-saving action are listed in **Table 2**. The quality requirements are determined under the principle of picking a value lower than that when agents only take usual action. The performance measures for each environment are shown in **Fig. 3**, where the dotted lines colored in red represent the given requirements of cumulative existence duration of dirt. We set the values of quality requirement $D^{e,20}_{req}$ for each environment e as the baseline of cumulative existence duration of dirt, and 15, the theoretical value of energy consumption per tick, as the baseline of total energy cost.

As shown in Fig. 3, introduction of energy-saving strategies has increased the cumulative existence duration of dirt in all circumstances. Our proposed method of meta-strategy AMTDS/ESC always outperformed other regimes with single target decision strategy, and agents with the method were able to cooperatively satisfy the given requirements. However, there is still room for improvement since we expected the requirements to be strictly meet. Especially for *Wait* action in Envs. (b), (c) and (d), D_{t_s,t_e} were 8% lower than the baseline, meaning that agents could actually rest more save more energy.

A key observation is that agents with energy-saving strategies successfully reduced the total energy cost about 20%-50% for *Return* action, and 10%-35% for *Wait* action. Within the four environments, agents in Env. (b) gave the best performance in respect of overall energy consumption. We considered the possible reason to be that the dirty regions in Envs. (c) and (d) cause significant difference between dirt amount in different regions. As a result, it is harder for agents to accurately estimate the total dirt amount in the environment and make correct action selection.

On the other hand, we found that *Return* action always outperformed *Wait* action with lower values of energy consumption. Since we set $T_{check} = 100$, requirement estimation was conducted every 100 ticks. During a operation cycle of an agent, requirement estimation could be conducted 90 times at a maximum by an agent with *Return* action, while estimation was only conducted after charging by an agent with *Wait* action. As a result, agents with *Return* action have a higher chance of performing energy-saving behavior. As future work, we plan to combine these two action and let the agents learn the parameters.

6. Conclusion and Future Work

Aiming at reducing cost of energy, we extended the learning method of target decision strategies in continuous cleaning tasks by multiple agents with shallow communication. On top of that, we proposed methods for agents to independently estimate whether the quality of environment reaches the given requirement levels, and to evaluate the importance of themselves with regard to the system. Based on the results, agents choose to remain usual behavior or to select other action such as returning to the charging base or resting for a while with an objective of saving energy.

The experimental results demonstrated that our proposed methods enable agents to reduce cost of energy while cooperatively maintain the given requirements of dirt amount. Within the five target decision strategies, AMTDS/ESC was able to give the best performance in respect of both dirt amount and energy consumption. For energy-saving action, *Return* outperformed *Wait*, whereas these strategies can be combined. As previously mentioned, we plan to integrate the two strategies and let agents learn to choose appropriate values of parameter such as length of wait time and check interval. In addition, we set the locations of charging base for all agents at the middle of the map in this paper. Putting the charging bases into different locations or setting multiple charging bases might make it easier for agents to perform energy-saving behavior.

On our research agenda, we would also like to focus on enabling agents to autonomously and individually evaluate their importance with regard to the system or their recent contribution and performance. With this functionality, the continuous system can eliminate old robots and introduce new ones without effecting the overall performance of the system.

References

[1] Z. Chen, L. Lin and G. Yan: An Approach to Scientific Cooperative Robotics through MAS (multi-agent system), *Robot*, Vol.23, No.4, pp. 368-373 (2001).

[2] K. Yoneda, A. Sugiyama, C. Kato and T. Sugawara: Learning and relearning of target decision strategies in continuous coordinated cleaning tasks with shallow coordination, *Web Intelligence*, Vol. 13, No. 4. IOS Press, pp. 279–294 (2015).

[3] M. Ahmadi and P. Stone: Continuous area sweeping: A task definition and initial approach, *Proc. of the 12th International Conference on Advanced Robotics*, pp. 316–323 (2005).

[4] M. Ahmadi and P. Stone: A multi-robot system for continuous area sweeping tasks, *Proc. of the 2006 IEEE International Conference on Robotics and Automation*, pp. 1724– 1729 (2006).

[5] D. Moreira, G. Ramalho and P. Tedesco: SimPatrol - Towards the Establishment of Multi-agent Patrolling as a Benchmark for Multi-agent Systems, *Proceedings of the International Conference on Agents and Artificial Intelligence*, pp. 570-575 (2009).

[6] H. Santana, G. Ramalho, V. Corruble and B. Ratitch: Multi-Agent Patrolling with Reinforcement Learning, *Proc. of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, Vol. 3, pp. 1122–1129 (2004).

[7] K. Yoneda, A. Sugiyama, C. Kato and T. Sugawara: Autonomous Learning of Target Decision Strategies without Communications for Continuous Coordinated Cleaning Tasks, *2013 IEEE/WIC/ACM International Conferences on Web Intelligence (WI) and Intelligent Agent Technology (IAT)*, pp. 216-223 (2013).

[8] A. Sugiyama and T. Sugawara: Meta-strategy for cooperative tasks with learning of environments in multi-agent continuous tasks, *Proc. of the 30th Annual ACM Symp. on Applied Computing*, pp. 494–500 (2015).

[9] A. Sugiyama, V. Sea, T. Sugawara: Effective Task Allocation by Enhancing Divisional Cooperation in Multi-Agent Continuous Patrolling Tasks, *2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 33–40 (2016).

[10] D.Hahnel, W.Burgard, D.Fox and S.Thrun: An efficient Fast SLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements, *Proc. of IEEE/RSJ International Con-*

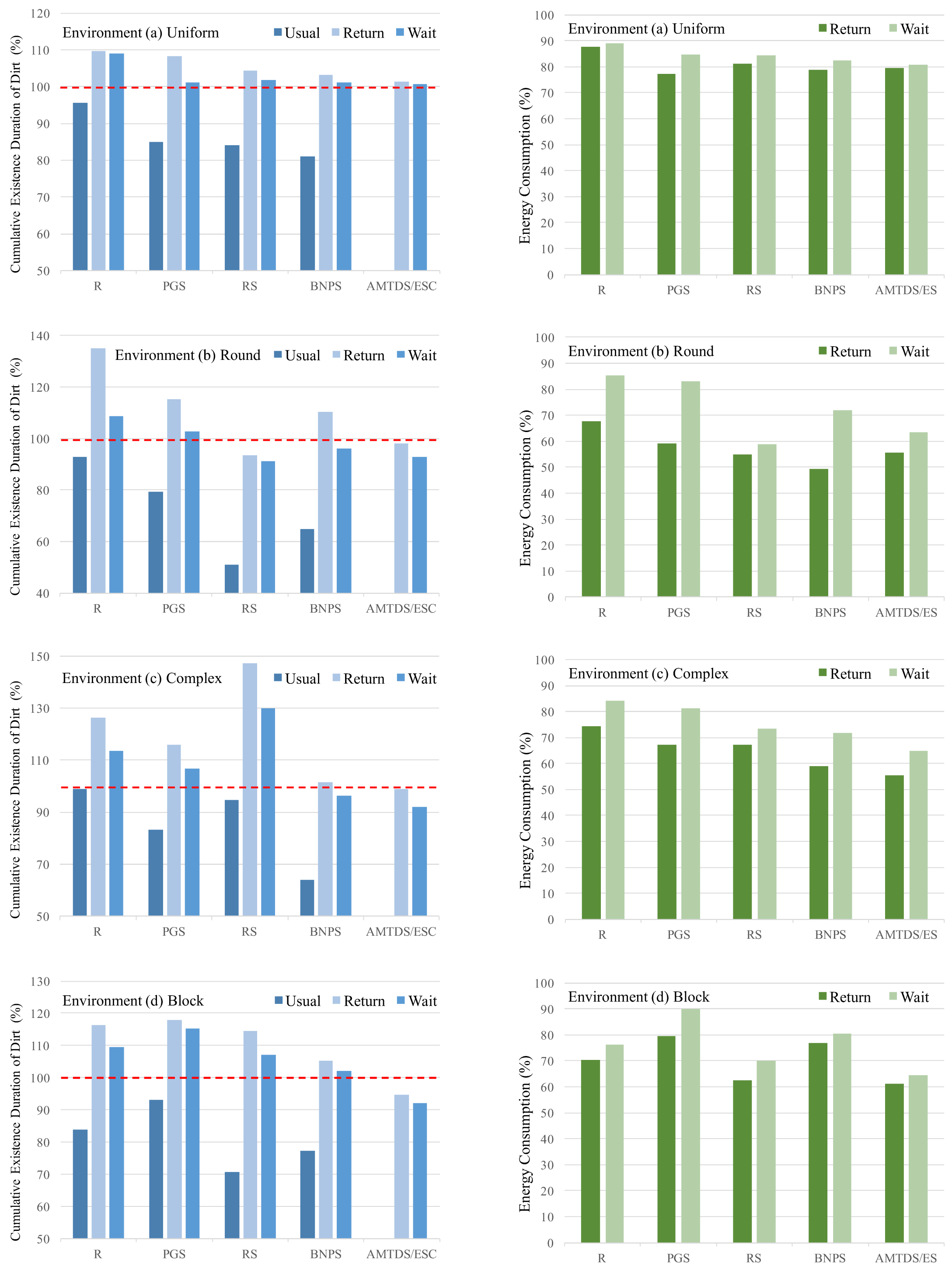

Fig. 3 Results of performance measures.

ference on Intelligent Robots and Systems (IROS 2003), Vol. 1, pp. 206–211 (2003).

[11] C. Cai, C. Yang, Q. Zhu and Y. Liang: Collision Avoidance in Multi-Robot Systems, *Proc. of the 2007 IEEE International Conference on Mechatronics and Automation*, pp. 2795-2800 (2007).

[12] C. Beatriz, E. Toledo and N.R. Jennings: Learning when and how to coordinate, *Web Intelligence and Agent Systems* 1(3), pp. 203–218 (2003).