Reducing Efficiency Degradation Due to Scheduled Agent Suspensions by Task Handover in Multi-Agent Cooperative Patrol Problems

Sota Tsuiki, Keisuke Yoneda, and Toshiharu Sugawara
Department of Computer Science and Communications Engineering, Waseda University
Tokyo 1698555, Japan
{s.tsuiki,k.yoneda}@isl.cs.waseda.ac.jp, sugawara@waseda.jp

Abstract

This paper proposes a method to mitigate the significant performance degradation due to planned suspensions in the *multi-agent cooperative patrol problem*. In recent years, there has been an increased demand to utilize multiple intelligent agents that control robots. Furthermore, cooperation between multiple agents is required for performing tasks that are complex and/or cover large spaces. However, since robots are machines, they must be periodically inspected or replaced with new ones to prevent unintended breakdowns for continuous operation and to prolong the lifetime of agents as much as possible. However, such suspension of agents for inspection can cause a sudden deterioration in performance, which is not ignorable in some applications. Meanwhile, such suspensions are usually planned; thus, we can know in advance which agents will stop, and when, to anticipate a preparation period before the actual suspension time. Thus, we introduce a negotiation method in which the agents that are scheduled to be suspended hand over some responsible and important tasks to other agents to reduce the impact of a sudden performance degradation. The experimental results show that the proposed method considerably reduces the performance degradation, especially for security patrol applications.

Introduction

In recent years, the advancement of computational technology has brought us closer to applications that utilize networked intelligent agents to control robots and/or provide sophisticated services. Furthermore, the use of multiple cooperative agents is expected to increase, owing to the size of the environment and the complexity of the required tasks. One problem framework for these types of applications is the *multi-agent cooperative patrol problem* (MACPP), in which agents simultaneously patrol a specific environment to execute required tasks, such as cleaning or status checking for security, while collaborating with each other. However, because agents usually determine their actions independently, their actions often negatively interfere with each other, for example, for synchronization and collision avoidance, so that the overall efficiency may decrease. To prevent such a loss of efficiency, cooperation and coordination of agents is required. However, it is not easy to design/implement coordinated behaviors in advance because many factors must be anticipated, such as time and space constraints as well as processing capacity that cannot often be determined in advance. Therefore, an autonomous strategy learning method is required for efficient cooperative behavior by considering the characteristics of the environment/tasks and the learned behaviors of other agents.

Many studies on the MACPP have been performed. For example, Yoneda et al. (Yoneda, Kato, and Sugawara 2013) proposed a method *adaptive meta-target decision strategy* (AMTDS) in which each agent autonomously determines an appropriate patrol strategy using reinforcement learning, with learning the frequency of visitation requests for each location in the environment. Sugiyama et al. (Sugiyama, Sea, and Sugawara 2019) proposed the *AMTDS with enhanced divisional cooperation* (AMTDS/EDC) by adding light-weight negotiation for the division-of-labor by dividing the environment to improve efficiency. However, when the number of agents decreased for periodic suspension, for example, they reported that it caused a temporary but significant decrease in the total performance. For example, the security patrol, which is an important application of MACPP, has a concern of becoming a security hole even if it is short-term. However, if we consider agents as robot-like machines, a decrease in the number of agents is probable due to breakdowns as well as periodic inspections. Moreover, to ensure the longevity and continuous operation of robots, periodic inspections cannot be avoided. Thus, we have to tackle the problem of performance degradation due to periodic inspection and maintenance without imposing any additional load on human staff.

Therefore, we propose another negotiation method for reducing the overall performance degradation due to the suspension of some agents. This is accomplished by autonomously delegating parts of the tasks for which the agents are responsible to be performed cooperatively, if we know which agents will stop and when. In fact, periodic maintenance/inspection is necessary for machines, such as robots, for sustainable operations and to increase their lifetimes. Furthermore, the plan for stoppage associated with this purpose is known in advance. Thus, by integrating our method into the conventional method (Sugiyama, Sea, and Sugawara 2019), we are able to decrease the occurrences of

security holes (in security patrols) or the amount of trash that is left uncleaned (in cleaning tasks) through communications for negotiation using Wi-Fi at close distances. We experimentally evaluate the proposed method in comparison with the existing method, and show that it can considerably reduce the performance degradation due to the planned suspension in the MACPP.

Related Work

There are many studies on the MACPP to achieve cooperative and coordinated behaviors of multiple agents. In general, there are two main approaches to the coordination of agents: the first approach is to divide the environment into the responsible areas explicitly and assign one of them to each agent (Ahmadi and Stone 2006; Elor and Bruckstein 2009; Kato and Sugawara 2013) For example, Elor and Bruckstein (Elor and Bruckstein 2009) proposed an area partitioning method to balance the sizes of subareas allocated to individual agents based on the model of balloon pressure. Kato and Sugawara (Kato and Sugawara 2013) extended this method to apply more complicated environments with obstacles, slopes, and different visiting requirements. The other approach is to let each agent autonomously select an appropriate patrol strategy according to the environment and other agents' behaviors without partitioning the area explicitly (Kalra, Ferguson, and Stentz 2005; Elmaliach, Agmon, and Kaminka 2007; Sampaio, Ramalho, and Tedesco 2010; Yoneda, Kato, and Sugawara 2013; Sugiyama, Sea, and Sugawara 2019). However, as mentioned previously, these studies do not consider the performance degradation due to suspension of a few agents. We take the latter approach because it seems to be less affected by the periodic suspensions, although it still causes unignorable degradation.

Some studies have investigated the planned suspensions using a multi-agent system framework. Panteleev et al. and Ghita et al. tried to make plans for the periodic maintenance/inspection and repair of tech equipment in a multi-agent simulated environment (Panteleev et al. 2014; Ghita, Agnès, and Xavier 2018). Unlike our study, these studies focused on the generation of a work plan for staff for periodic inspections so as to reduce staff workload. There are also many studies on planned suspensions in different fields. Gavranis et al. proposed an exact solution algorithm for a wide variety of flight and maintenance planning (FMP) problems (Gavranis and Kozanidis 2015). Seif et al. extended this method to be applied to operations and maintenance planning problems that are more general than the FMP problem (Seif and Andrew 2018). Chen et al. (Chen and Tang 2019) proposed a management workflow design that integrates building information modeling and digital programming for the operation and maintenance of a building. However, to the best of our knowledge, there are no studies that discuss the autonomous planned suspension of multiple intelligent agents.

Model

The model description is almost identical to that in Sugiyama et al. (Sugiyama, Sea, and Sugawara 2019) except that we focus on the security patrol as well as the cleaning task. We describe the model of our problem briefly; the details are presented in (Sugiyama, Sea, and Sugawara 2019).

Environment and Agents

We denote the set of agents $A = \{1, \ldots, n\}$. The environment in which agents patrol is denoted by the graph $G = (V, E)$, which can be embedded in a two-dimensional space, where $V = \{v_1, \cdots, v_n\}$ is the set of nodes that correspond to locations, and E is a set of edges whose elements $e_{k,l} \in E$ represent the paths between nodes v_k and v_l. By adding dummy nodes if necessary, we can assume the length of all edges to be 1. Let $d(v_l, v_k)$ be the length of the shortest path and $m(v_l, v_k)$ be the Euclidean distance between $\forall(v_l, v_k) \in V \times V$. The discrete time whose unit is *step*, is introduced. In one step, any agent can move to a neighboring node, and reconnoiter that node. Expressing the location of agent $i \in A$ at time t as $v_t^i \in V$, we define the distances between agents i and j at t by $d(i, j) = d(v_t^i, v_t^j)$ and $m(i, j) = m(v_t^i, v_t^j)$. The Euclidean distance is used to determine if communication is possible.

Every node has the occurrence probability $p(v)$ ($0 \leq p(v) \leq 1$) of events that at least one agent $i \in A$ has to visit v, where events mean dirt accumulation to vacuum, or occurrences of events to be observed. The number of unobserved events $L_t(v)$ in v at time t is set to 0 if i visits and explores v at t; otherwise, it is incremented by one at every step with probability $p(v)$. Hence, agents have to more frequently visit the nodes with higher probability of occurrence; thus, such nodes, for example, require a high level of security.

Agent $i \in A$ has a finite battery capacity and must return to its own charging base before it runs out. For $\forall v \in V$, i has an *importance value* $p^i(v)$, which is also the predicted event occurrence probability at v estimated from its own event explorations. Therefore, $p^i(v)$ differs among agents, even for the same node v, because it is estimated on the basis of individual observations. Agent i determines the next target node v_{tar}^i using the strategy s chosen by the AMTDS and then generates the shortest or reasonable path to v_{tar}^i. The AMTDS is the learning method for choosing the target decision strategy s among four simple strategies by Q-learning to decide the next goal to move. The reward for Q-learning is the number of observed events per step for each strategy. After i reaches v_{tar}^i, i determines the next target node again and repeats these actions until its battery power becomes low. Thus, each agent periodically returns to its charging base to charge for constant patrolling. Please see (Yoneda, Kato, and Sugawara 2013; Sugiyama, Sea, and Sugawara 2019) for the details of the AMTDS, the four simple strategies and the control for periodical returns.

Agents i and j can communicate with each other when their distance is less than or equal to d_{co}, i.e., $m(i, j) \leq d_{co}$. To suppress excessive communications, i retains $T_{lst}^{i,j}$, which is the last time when it communicated with j, and i does not communicate with j until $T_{lst}^{i,j} + B$, where positive integer B is the *minimum communication interval*.

When agent i visits node v at t and finds events to be observed, it updates the importance value $p^i(v)$ as

$$p^i(v) = (1-\alpha)p^i(v) + \alpha\frac{1}{t - t^v_{vis}}, \quad (1)$$

where α ($0 < \alpha \leq 1$) is the learning rate, and t^v_{vis} is the time when any agent visited node v most recently because of the assumption that each agent can know other agents' positions. Then, i individually has the *responsible nodes* V^i_R ($\subset V$), which is the set of nodes whose importance value $p^i(v)$ is up to the upper N^i_R-th value; therefore, it consists of nodes to which i decides to visit more frequently. Note that the responsible nodes is updated when agents return to their charging base. Initially $N^i_R = |V|$, so that $V^i_R = V$ for $\forall i \in A$. Next, N^i_R is updated dynamically depending on the contribution of each agent. How to determine N^i_R is explained below.

Performance Measure

If we consider our MACPP application with patrols for security surveillance in a large area, agents need to cooperate with each other to minimize the number of events that are left unobserved. Therefore, we introduce the *maximal number of unobserved events*, which is calculated by

$$U_{t_s,t_e}(s) = \max_{v \in V, t_s \leq t \leq t_e} L_t(v)$$

during the interval from t_s to t_e ($t_s < t_e$). If $U(s)$ is large, some nodes are left unobserved, so security weakness may exist. On the other hand, if we consider cleaning task applications, we have to reduce the amount of time trash exists that is left uncleaned; this can be defined as

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

which is called the *cumulative unobserved duration of events*. Thus, a large $D(s)$ indicates that the environment is left dirty. Therefore, the smaller values of $U(s)$ and $D(s)$ indicate more efficient patrol.

Proposed Method

We attempted to solve the problem by integrating the proposed negotiation for task delegations to prevent/reduce a sudden deterioration in efficiency with the AMTDS/EDC (Sugiyama, Sea, and Sugawara 2019). Our proposed method is an extension of the negotiation method in AMTDS/EDC, but the main difference is that their method is to balance the workload to improve the efficiency, whereas in our method, the tasks of the agents that will be stopped are delegated to other agents, and thus, intentionally unbalanced. We refer to the proposed method as the *AMTDS with task handover for scheduled suspension* (AMTDS/TH).

Scheduled Suspension of Agents

For a subset of agents A_S ($\subset A$), a scheduled suspension of A_S for inspection or maintenance is specified by a tuple of A_S and three positive integers (A_S, T_{sp}, T_{rs}, G_{rsv}), where T_{sp} is the scheduled suspension time, T_{rs} is the scheduled resume time, and G_{rsv} is the preparation period before suspension. After the tuple for the scheduled suspension is announced to all agents, they enter the preparation period at $T_{sp} - G_{rsv}$, i.e., they start preparations to reduce a sudden deterioration by the suspension. Next, the agents calculate the *maximum number of chances to negotiate* $N_{pn}(t)$ until the scheduled suspension at t

$$N_{pn}(t) = \frac{T_{sp} - t}{B}$$

to estimate the remaining negotiation chances with other agents.

Negotiations during Preparation Period

In normal time (i.e., no suspension is scheduled), agent $\forall i \in A$ communicates with other agents to exchange the importance values of the nodes selected from V^i_R to balance the workload and enhance divisional cooperation for better efficiency, as described in the conventional method (Sugiyama, Sea, and Sugawara 2019). Meanwhile, during the preparation period, we introduce the negotiation for *unidirectional task delegation* between agents $i \in A_S$ and $j \in A \setminus A_S$, in which only i transfers the importance values in V^i_R to j; this negotiation message may lead to one-way indirect delegation to hand over some tasks to j, because j is likely to visit the received nodes more frequently due to the increased importance values, although i will visit them fewer times. Note that even after j receives the importance value of v, j will re-evaluate the received importance values by itself, so that it is possible that v is excluded from V^j_R. Conversely, i may revisit v; if it finds that j has not visited v, then i will increase its importance values again, because the task handover for v did not work well. Also note that agents in A_S do not communicate each other during the preparation period, whereas agents in $A \setminus A_S$ mutually negotiate the same as in normal times.

Unidirectional task delegation between $i \in A_S$ and $j \in A \setminus A_S$ proceeds as follows: Agent i selects the nodes in V^i_R that have the top e_g (> 0) values of $p^i(v)$. Then, i transfers a certain ratio of $p^i(v)$ of any selected node to j. This results in the update of importance values in i and j as

$$p^j(v) \leftarrow p^j(v) + p^i(v) \times \delta_c$$
$$p^i(v) \leftarrow p^i(v) \times (1 - \delta_c)$$

for each v of the selected nodes. Parameter $\delta_c (0 < \delta_c < 1)$ specifies the ratio of importance values to transfer. The positive integer e_g is calculated by

$$e_g = \min\left(N^i_R, N^i_{dmax}, \left\lfloor \frac{N^i_R}{N_{pn}(t)} \times \gamma_c \right\rfloor\right),$$

where $N^i_{dmax} (0 < N^i_{dmax} < N^i_R)$ is a threshold to prevent unanticipated changes, and γ_c is the ratio of transfer. After passing the importance values, N^i_R and N^j_R are modified by

$$N^i_R \leftarrow N^i_R - e_g$$
$$N^j_R \leftarrow \min(|V|, N^j_R + e_g)$$

After the updates of $p^i(v)$ and $p^j(v)$, their responsible nodes V^i_R and V^j_R will be recalculated at their charging base.

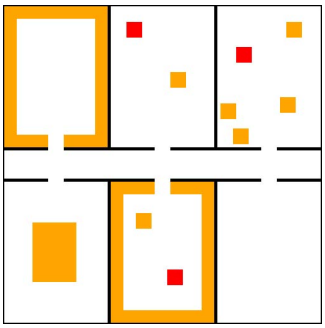


Figure 1: Experimental environment.

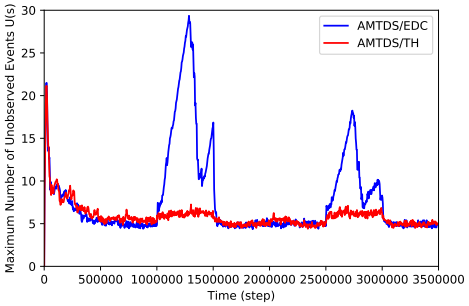


Figure 2: Transition of $U(s)$ over time.

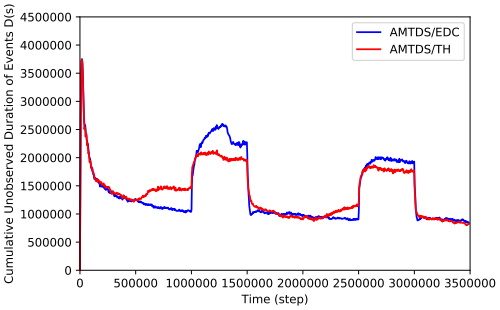


Figure 3: Transition of $D(s)$ over time.

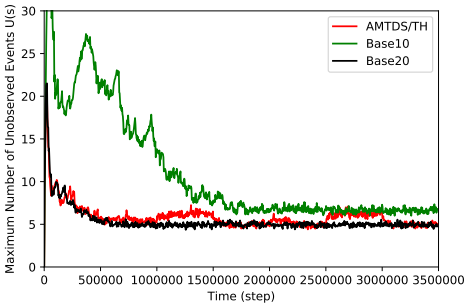


Figure 4: Performance comparison ($U(s)$).

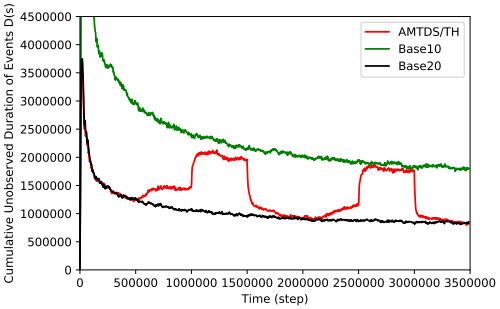


Figure 5: Performance comparison ($D(s)$).

Experimental Evaluation

Experimental Setting

The objective of our experiments is to show that the proposed method AMTDS/TH is able to suppress a sudden performance degradation when multiple agents enter a scheduled suspension compared with the performance of the conventional method, AMTDS/EDC. We conducted two experiments to evaluate our method in two patrol problems: the cleaning task (for the comparison with that of the conventional method); and security patrol by multiple robots, which requires tighter control. To verify if our method is effective for multiple scheduled suspensions, two scheduled suspensions were conducted in this experiment.

The experimental environment and setting, which are identical to the one used in (Sugiyama, Sea, and Sugawara 2019) for comparison, is a 101×101 grid, as shown in Fig. 1, where the black lines are walls. Colored regions have higher probabilities of event occurrences, which is specified by

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

for $\forall v \in V$. The number of agents is 20 as an example, the charging base for all agents is set at the center of the environment, and multiple agents can exist, only at the center location for charge.

The length of one experiment is 3,500,000 steps. We set two scheduled suspensions, which are specified by (A_S, T_{sp}, T_{rs}, G_{rsv})= (A_1, 1000000, 1500000, 500000) and (A_2, 2000000, 2500000, 500000). A_1 and $A_2 = A \backslash A_1$ are the set of ten agents randomly selected from A in each run. Other parameters used in the various experiments are shown in Table 1. The graphs shown below are the average values of 20 experimental runs, unless otherwise mentioned.

Comparison of Degradation in Efficiency

We plotted the value of $U(s)$ (we omit the subscripts for simplicity) for security patrol in Fig. 2, and $D(s)$ for the cleaning task in Fig. 3. These figures indicates that, in both experiments, agents in the AMTDS/TH could reduce the sudden degradation of performance due to the scheduled suspensions more than that of the agents in AMTDS/EDC. These results show that the agents scheduled to be suspended were

Table 1: Experimental parameters.

Description	Parameter	Value
Number of agents	$\|A\|$	20
Communication Range	d_{co}	5
Minimum communication interval	B	10800
Data collection interval	$t_e - t_s$	3600
Threshold to prevent unanticipated changes	N^i_{dmax}	500
Learning rate for importance value	α	0.1

able to effectively hand over some parts of their tasks to other agents, and thus the efficiency degradation in $U(s)$ and $D(s)$ after two suspensions became quite small.

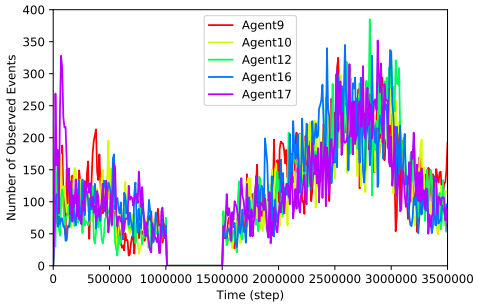
In particular, we can see from Fig. 2 that the proposed method significantly reduces the sharp degradation of performance (i.e., increase in $U(s)$) by approximately 77% during the first scheduled suspension, and 65% during the second. Moreover, while half of the agents did not operate, $U(s)$ was kept almost as low as that before they stopped. On the other hand, the value of $D(s)$ gradually increased during the preparation period by the gradual handover of tasks. However, the sudden degradation in efficiency (the increase in $D(s)$) has been tempered, especially during the first scheduled suspension.

The difference in the properties of the sudden performance degradation between $U(s)$ and $D(s)$ is due to how the agents cover the entire work. In the case of the security-patrol type problem ($U(s)$), agents are only required to individually visit nodes, especially focusing on the important nodes with high event occurrence probability. Therefore, the decreased number of agents did not cause a significant efficiency loss if the agents had learned enough during the preparation period. However, agents using the conventional method could not cope with the sudden halving of their numbers and were left unable to visit some important nodes. In the cleaning-type problem, on the other hand, the proposed learning of agents during the preparation period prevented decreasing efficiency, but only to some degree, because the agents had to cover all nodes; thus, the time left for the unobserved events accumulated. However, we were able to mitigate the sudden sharp degradation considerably, compared with the conventional method.

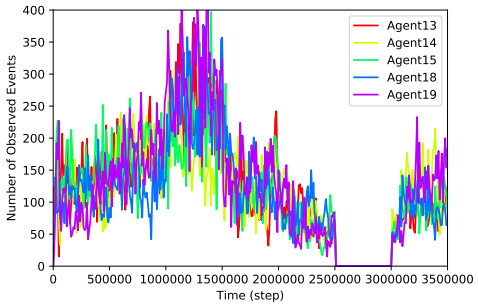
Evaluation of Performance

We need to investigate whether the proposed method works sufficiently to mitigate the performance degradation caused by the reduction in the number of agents. For this purpose, we examined the efficiency when the number of agents is fixed at 10 or 20 without scheduled suspension, and compared these results with those of our proposed method. The results are plotted in Fig. 4 (for $U(s)$) and Fig. 5 (for $D(s)$). Note that the labels "Base10" and "Base20" in these figures express the cases when the the number of agents $|A|$ is 10 and 20, respectively.

If we look at the curves labeled "Base10" and "AMTDS/TH" in these figures, we can see that agents using



(a) Agents in A_1



(b) Agents in A_2

Figure 6: Number of events observed in agents.

the proposed method (AMTDS/TH) exhibit the same or better performance during the scheduled suspensions than when ten agents work continuously without stopping. In particular, it is interesting to note that agents using AMTDS/TH can work much more efficiently during the first scheduled suspension. Another finding from our experiments is that working with more agents in an earlier stage may lead to faster convergences of learning. Therefore, for example, it may be effective to have twenty agents working initially, and then to have ten agents working after applying the proposed method. Of course, the performance of twenty agents using the AMTDS/TH is almost equal to that when twenty agents work continuously (i.e., Base20).

Lastly, we plotted the number of events observed by individual agents in A_1 and A_2 in Fig. 6 to investigate how agent workloads vary over time when we investigate the value of $U(s)$. Note that this figure was generated using data from a single randomly selected experimental run and we only plotted the data of five agents that are also selected randomly from A_1 and A_2. Figure 6a indicates that the number of events observed by agents in A_1 gradually decreased in the preparation period of the first scheduled suspension (between 500,000 and 1,000,000 steps). After returning to the work from the first suspension, they quickly increased the number of observed events. Next, they work to nearly the maximum to observe the events during the second suspension and, finally, the number of observed events decreased to the average level, which is approximately half of the maximum number of observed events. Figure 6b also shows a

trend that is almost the opposite to that of Fig. 6a. This analysis suggests that agents hand over their tasks by transferring a certain ratio of importance values of their responsible nodes to other agents with the proposed negotiation

Discussion

First, it should be noted that our proposed method contributed considerably to mitigate the sudden degradation of performance due to the temporal stop (suspension) of some agents. Because agents are autonomous learners, they can gradually compensate for the work of agents that have stopped; however, this process requires some time. Such degradation can be fatal in applications such as security patrols, for example. In contrast, the proposed method pays particular attention to security patrol applications and prevents a temporary but sudden degradation of efficiency.

In our experiment, we had a rather extreme setup, where half of the agents would suspend. The reason for this extreme setting is to confirm the effectiveness of the proposed method. In a real-world application, a small number of agents would take turns to suspend to perform periodic inspections. Even with such a small number of suspensions, however, sudden degradation is not ignorable in some MACPP applications, such as security patrol and patrol for constant data collection from environmental sensors.

Planned suspensions for periodic maintenance of machines and equipment, such as robots, usually need to be scheduled by human experts, carefully considering the impact on labor overload and functional degradation, so as not to disrupt operations. However, this is a heavy burden on the human experts (Panteleev et al. 2014). Therefore, we believe that autonomous maintenance transition by agents is important for sustainable system operation in the future.

Conclusion

We first discussed the sudden performance degradation occurring in the conventional method due to the scheduled suspensions for periodic inspection or replacement of agents. We also pointed out that such degradation is not ignorable in applications, such as security surveillance, because it may result in temporary security weaknesses. For this type of problem, we proposed a negotiation procedure for delegating some important tasks to other agents to reduce the performance degradation due to a scheduled suspension in the MACPP. Furthermore, after the suspended agents return to operate, the performance rapidly recovers to the same level as before the suspension.

In the future, we will focus on the security patrol problem, and would like to experiment in more realistic problem settings. In the real security patrol problem, for example, agents do not learn the security level of each location by patrolling, but the levels are usually given in advance. Therefore, we would like to develop a method for mitigating the performance degradation during scheduled suspensions without learning of importance in the environment.

Acknowledgement: This work was partly supported by JSPS KAKENHI Grant Numbers 17KT0044 and 20H04245.

References

Ahmadi, M., and Stone, P. 2006. A multi-robot system for continuous area sweeping tasks. In *Proc. 2006 IEEE Int. Conf. on Robotics and Automation, 2006. ICRA 2006.*, 1724–1729. IEEE.

Chen, C., and Tang, L. 2019. Bim-based integrated management workflow design for schedule and cost planning of building fabric maintenance. *Automation in construction* 107:102944.

Elmaliach, Y.; Agmon, N.; and Kaminka, G. A. 2007. Multi-robot area patrol under frequency constraints. In *Proc. 2007 IEEE Int. Conf. on Robotics and Automation*, 385–390.

Elor, Y., and Bruckstein, A. M. 2009. Multi-a(ge)nt graph patrolling and partitioning. In *2009 IEEE/WIC/ACM Int. J. Conf. on Web Intelligence and Intelligent Agent Technology*, Vol. 2, 52–57.

Gavranis, A., and Kozanidis, G. 2015. An exact solution algorithm for maximizing the fleet availability of a unit of aircraft subject to flight and maintenance requirements. *European Journal of Operational Research* 242(2):631–643.

Ghita, B.; Agnès, L.; and Xavier, D. 2018. Scheduling of production and maintenance activities using multi-agent systems. In *2018 IEEE 23rd Int. Conf. on Emerging Technologies and Factory Automation (ETFA)*, Vol. 1, 508–515. IEEE.

Kalra, N.; Ferguson, D.; and Stentz, A. 2005. Hoplites: A market-based framework for planned tight coordination in multirobot teams. In *Proc. of the 2005 IEEE Int. Conf. on Robotics and Automation*, 1170–1177.

Kato, C., and Sugawara, T. 2013. Decentralized area partitioning for a cooperative cleaning task. In *PRIMA 2013: Principles and Practice of Multi-Agent Systems*, 470–477. Berlin, Heidelberg: Springer.

Panteleev, V.; Kizim, A.; Kamaev, V.; and Shabalina, O. 2014. Developing a model of multi-agent system of a process of a tech inspection and equipment repair. In *Joint Conf. on Knowledge-Based Software Engineering*, 457–465. Springer.

Sampaio, P. A.; Ramalho, G.; and Tedesco, P. 2010. The gravitational strategy for the timed patrolling. In *2010 22nd IEEE Int. Conf. on Tools with Artificial Intelligence*, Vol. 1, 113–120.

Seif, J., and Andrew, J. Y. 2018. An extensive operations and maintenance planning problem with an efficient solution method. *Computers & Operations Research* 95:151–162.

Sugiyama, A.; Sea, V.; and Sugawara, T. 2019. Emergence of divisional cooperation with negotiation and re-learning and evaluation of flexibility in continuous cooperative patrol problem. *Knowledge and Information Systems* 60(3):1587–1609.

Yoneda, K.; Kato, C.; and Sugawara, T. 2013. Autonomous learning of target decision strategies without communications for continuous coordinated cleaning tasks. In *Proc. of the 2013 IEEE/WIC/ACM Int. J. Conf. on Web Intelligence and Intelligent Agent Technologies – Vol. 02*, 216–223. IEEE Computer Society.