2022年度　卒業論文

マルチエージェント協調巡回問題における エネルギー消費抑制手法の提案

松本 航平

早稲田大学 基幹理工学部
情報理工学科

学籍番号　1W193102

提出日　2023/

指導教授　菅原 俊治

目 次

概 要

後で追記する

1　序論

後で追記する

近年，ロボット技術が発達し，巡回パトロールや清掃などといった，人間が日常的に繰り返す作業や，災害地や原子力発電所，宇宙などでの作業を複数の自律的ロボットで代替する動きが加速している．このような複数のロボットが協調して共通の作業を行う問題は，ロボットを自律的に動作するエージェントとしてモデル化し，マルチエージェント協調巡回問題 (*multi-agent cooperative patrolling problem*, MACPP) と呼ばれる．MACPP の研究では，複数のエージェントが協力・協調することで，与えられた環境において効率的かつ効果的に巡回を行うための方法・アルゴリズムを見出すことを目的としている．

巡回効率だけを追求した高度な行動や学習は，確かに巡回の効率は向上するものの，必要以上にエネルギーを消費する可能性があり，これは MACPP の極めて大きな課題となっている．特に，本研究で想定する自律型エージェントは独自のバッテリを持ち，頻繁な充電を強いられることになる．一方，アプリケーションによっては，巡回作業に対する品質要求があり，それを超えることは必ずしも期待されているわけではない．例えば，清掃問題では，ある程度環境がきれいになっていれば十分であり，過度の巡回作業はかえって単位エネルギー当たりの作業効率を低下させる．また，環境が複雑で大規模な場合，どれだけのエージェントが必要かを事前に判断することができないこともあり，エージェントが少ないと品質要求を満たせず，多すぎるとエネルギーの浪費に繋がる．

マルチエージェントシステムにおける協調の観点からエネルギー効率に注目した研究 [1, 2, 3] がある．例えば，[2] では，マルチロボットにおける探索問題において，移動ロボットの総運動エネルギーを節約するために，運動時間を短縮する分散型協調手法を提案している．しかし，これらの研究は，タスクのために効率的に移動・作業し，結果として総消費エネルギーを削減する方法を目指したものである．一方，本手法では，他のエージェントが要求されるタスクの質を満たすことができれば，あるエージェントが自らの判断でしばらく停止したり，システムから退出したりすることを実現しようとするものである．また，[4, 5] では，エージェントが要求を満たせば自律的に一時停止してエネルギーを節約し，再び充電して探索に戻るという手法を提案している，しかし，その方法は不十分であり，エージェントは依然として不必要に環境内を巡回していることが分かった．

そこで本研究では，[5] の手法を拡張し，その後の行動によって起こりうる貢献を自律的に予測し，環境の現状を推定することで要求品質の全体的な達成度を把握しながら，より効率的に要求品質の充足とエネルギー消費の削減を両立する手法を提案する．主な違いは，エージェントが休止・充電している間の巡回タスクの進捗は，他のエージェントの行動に依存し，エージェントごとに異なるため，個々のエージェントから見たエネルギー節約行動の自律学習を導入しているという点である．

さらに，この学習が進むとエージェントは，短い休止時間で巡回する Busy グループと，比較的長い時間休止してエネルギーを消費しない Energy save グループに分かれることを発見した．そのため，後者のグループのエージェントは，品質要求を満たしたまま，順次，活動を停止することができる，実験の結果，提案手法により，従来手法と比べて大幅にエネルギー消費量を削減することができることが分かった．また，Busy グループに属するエージェントの数は，環境条件によって変化することが分かった．その後，品質要求を満たしつつ，Energy save グループに属するエージェントを順次停止することで，活動しているエージェント数を削減することができた．

2　関連研究

後で，杉山さんの研究も追加する、領域分解についても追加 (SMASH 参考)

MACPP とその応用に関する研究は数多く行われている [6, 7, 8, 9, 10]．例えば， [9] では，*edge Markov evolution graphs* を用いた動的環境変化のモデルを提案した．また，[10] では，巡回問題をベイズ適応型遷移分離部分観測可能マルコフ決定過程として定式化し，モンテカルロ木探索法を拡張した分散型オンライン学習・計画アルゴリズムを提案している．さらに，[11] では，人口ニューラルネットワークを用いて，個人の idlness から共用の idlness を予測する方法を提案している．しかし，これらの研究では，バッテリの放電やエネルギー消費による定期的な充電を無視し，効率性のみを考慮したものである．[12] では，Q 学習によって決定される計画戦略のもと，複数のエージェントが定期的に充電しながら協調して巡回する AMTDS(*adaptive meta-target decision strategy*) という手法を提案している．さらに，ここから杉山さんの研究．しかし，この研究もエネルギー消費量の削減を考慮せず，巡回の効率化を目的とした学習のみを行うものである．

一方，[1, 2, 3, 1, 5, 13] では一部，省エネルギーに着目している．[1] では，消防ロボット以外にも複数のサブロボットを導入し，消火作業における総作業時間の延長を図った．また，[13] では，協調型群ロボットに対して，採餌などの連続タスクを解決するためのエネルギー配慮型分散タスク配分アルゴリズムを提案し，高効率なミッションの実現を実現した．しかし，これらの研究では，本研究とは異なり，動作時間の延長も考慮されている．これに対して，[5] では，AMTDS を巡回の品質要求に合わせて拡張し，省エネルギー化を図った．しかし，この手法によるエネルギー削減は不十分であり，エージェントの活動にはまだ不要な行動が含まれている．そこで，本研究では，個々の視点からの学習を導入することで，品質要求を満たしながら，さらにエネルギー消費を削減した．さらに，複数のエージェントを停止する手法を提案し，消費エネルギーの削減を図った．

3　モデルの定義

本研究は，[5] で提案された清掃問題におけるエネルギー節約手法である，*adaptive meta-target decision strategy for energy saving and cleanliness* (AMTDS/ESC) の拡張である．また，従来手法との比較を行うため，本研究で用いる問題の定式化と環境，エージェントの活動モデルは [5] で用いられているものを踏襲する．

3.1　環境

エージェントの巡回環境を，2次元ユークリッド空間に埋め込み可能なグラフ$G = (V, E)$で表す．ここで，$V = \{v_1, \ldots, v_n\}$はノード集合を表し，各ノード乗にエージェントやイベント，障害物が存在する．また，Eはエージェントが移動する経路に対応するノードv_iとv_j間のエッジ$e_{i,j}$である．

さらに，ステップを単位とする離散時間を導入する．簡単のため，必要に応じてダミーノードを追加することで，全てのエッジの長さは1に保たれる．したがって，エージェントは1ステップで障害物のない隣接ノードに移動することができる．ここで，v_iとv_jの最短距離(エッジの数)を$d(v_i, v_j)$とする．

全てのノード$v \in V$上でイベントが発生し，そのイベント発生確率を$p(v)$ $(0 \leq p(v) \leq 1)$とする．毎時刻tにおいて，ノードvに蓄積されたイベント数$L_t(v)$は以下の式で更新される．

$$L_t(v) = \begin{cases} L_{t-1}(v) + 1 & (\text{確率 } p(v) \text{ のイベント発生時}) \\ L_{t-1}(v) & (\text{その他}) \end{cases}$$

時刻tにエージェントがノードvを訪れた時にv上のイベントは処理され，$L_t(v) = 0$となる．イベントの解釈は，アプリケーションによって異なり，例えば，掃除のアプリケーションでは，$p(v)$は場所vの汚れやすさを，$L_t(v)$は汚れの蓄積度合いを表す．また，防犯監視パトロールのアプリケーションでは，$p(v)$はアプリケーションの所有者が指定した，重要な場所に対する必要な防犯度合いを示し，$L_t(v)$は警戒レベルと解釈することができる．本研究では，全ノードの$p(v)$はあらかじめ指定されていると仮定する．

3.2　エージェント

n個のエージェントの集合を$A = \{1, \ldots, n\}$と表す．エージェント$i \in A$はバッテリを持ち，充電基地v^i_{base}で充電を繰り返すことで連続動作が可能である．つまり，バッテリが満タンの状態で$v^i_{base}i$を出発したiは，環境を巡回し，再びv^i_{base}に戻ってくるという動作をする．エージェントiのバッテリ性能を$(B^i_{max}, B^i_{drain}, k^i_{charge})$で表す．ここで，$B^i_{max}$はエージェントのバッテリ容量，$B^i_{drain}$は1ステップで消費するバッテリ消費量，$k^i_{charge}$はバッテリ残量を1増加させるために必要なステップ数である．時刻tにおけるエージェントiのバッテリ残量を$b_i(0 \leq b_i \leq B_{max})$とすると，$i$が1ステップで隣接するノードに移動したとき，$b_i(t)$は以下の式に従って更新される．

$$b_i(t+1) \leftarrow b_i(t) - B^i_{drain} \tag{1}$$

$b^i(t)$が0になるとそのエージェントは移動できなくなってしまうので，自身のバッテリ残量が0になる前に戻らなければならない．そこで，以下の式に示すように，エージェントiはノードvから充電基地v^i_{base}までの移動に必要な最小バッテリ量であるポテンシャル$\mathcal{P}^i(v)$を計算する．

$$\mathcal{P}^i(v) = d(v, v^i_{base}) \times B^i_{drain} \tag{2}$$

エージェントiは目標ノードv^i_{tar}を後の章で説明する目標決定戦略によって決定した際，実際に移動する前に，現在のバッテリ残量でv^i_{tar}に到達し，その後充電基地に戻ることが

できるかを，以下の式を用いて判定する．

$$b^i(t) \leq \mathcal{P}^i(v) + d(v_t^i, v_{tar}^i) \times B_{drain}^i \tag{3}$$

この条件を満たさない場合，以下のように目標ノード v_{tar}^i を更新し，充電基地に戻る．

$$v_{tar}^i \leftarrow v_{base}^i \tag{4}$$

エージェントは v_{base}^i に到着後，バッテリ残量が最大になるまで充電し，充電完了後は再び環境を巡回する．ここで，満充電 (つまり，$b_i = B_{max}$) になるまでに，$(B_{max} - b_i) \times k_{charge}$ ステップかかる．

エージェント i は，すべてのノード v に対し，v のイベント発生確率の予測値を表す重要度 $p^i(v)(0 \leq p^i(v) \leq 1)$ を持つ．$p^i(v)$ は各エージェントが独立して保持しており，その値はエージェントごとに異なる．i が v 上にいない場合，現在の蓄積イベント数 $L_t(v)$ は知ることができない．そこで，エージェントは時刻 t の $p(v)$ から期待値 $E^i(L_t(v))$ を以下の式に従って計算する．

$$E^i(L_t(v)) = p^i(v) \times (t - t_{vis}^v) \tag{5}$$

この計算のために，エージェントは自分と他のエージェントの位置を知ることができると仮定する．これは，現在の技術で容易に実現可能であるためである．例えば，赤外線や GPS などのセンサーを用いたり，エージェント間で直接通信したり，クラウドロボティクス，すなわち，クラウドを介して情報を共有したりすることで実現できる．しかし，エージェントは目的地を設定するための戦略や，目的地までの計画経路など，エージェント内部の情報や判断を共有・推論することはできない．

また，エージェント i と j はエージェント間の情報交換や交渉を用いる際に，互いに通信可能である．しかし，エージェントの過度の通信によるコスト増加や干渉を防ぐために，通信可能範囲 $d_{co}(> 0)$ が存在する．i と j が以下の式を満たすとき，互いに通信可能である．

$$m(v^i, v^j) < d_{co} \tag{6}$$

加えて，同様の目的から，時間面の制約である最低通信間隔 $B(> 0)$ が存在する．i は j と最後に通信した時刻 $T_{lst}^{i,j}$ を保持しており，$T_{lst}^{i,j} + B$ まで通信を行うことはできない．

さらに，本研究では，要求品質を満たしつつエネルギー節約行動を学習することに重点を置いており，実験で用いたグリッド状の環境では，衝突を回避する迂回経路の生成が容易であると考えられるため，エージェント間の衝突は考慮しないこととする．[14, 15] のように，衝突が発生しない経路を生成するアルゴリズムはいくつか提案されており，これらのアルゴリズムの 1 つを衝突回避に用いることができる．

3.3　評価指標

評価指標は，扱う MACPP の種類によって異なるが，本研究では評価指標として，以下の式で定められるイベント残存時間の総和 D_{t_s,t_e} と，エージェントの総エネルギー消費

量 C_{t_s,t_e} を用いる．

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

$$C_{t_s,t_e} = \sum_{i \in A} \sum_{t=t_s+1}^{t_e} \mathcal{E}_t(i), \tag{8}$$

ここで，$[t_s, t_e]$ $(t_s < t_e)$ は時間間隔を表し，$\mathcal{E}_t(i)$ は t におけるエージェント i の消費エネルギーを表す．したがって，i が隣接ノードに移動したとき $\mathcal{E}_t(i) = 1$，それ以外は $\mathcal{E}_t(i) = 0$ となる．例えば，D_{t_s,t_e} は清掃問題において，掃除機をかけずに放置した埃の累積時間や，セキュリティパトロールにおいて，チェックせずに放置したセキュリティ場所の累積時間や数を表し，MACPP ではこれを減らすことが目的となる．

一般に，D_{t_s,t_e} と C_{t_s,t_e} の値はどちらも小さい方が良いとされるが，両者はトレードオフの関係である．つまり，D_{t_s,t_e} と C_{t_s,t_e} の値をどちらも最小にすることは困難である．したがって，[5] と同様に 1step におけるイベント量の要求値 D_{req} を設定した．エージェントは以下の式を満たせるように協調を行う．

$$D_{t_s,t_e} \leq D_{req} \times (t_e - t_s) \tag{9}$$

本研究では，品質要求 (式 (9)) を満たしつつ，C_{t_s,t_e} をできるだけ小さくすることが目的である．なお，本研究では単純化のため，これ以降，D_{t_s,t_e}, C_{t_s,t_e} を $D(s), C(s)$ と表す．

4 準備

この章では，提案手法の基になった AMTDS，AMTDS/LD，AMTDS/EDC，AMTDS/ESC について説明する．これらの手法では，エージェントは目標ノード v^i_{tar} を決定する目標決定戦略と，それまでの経路を生成する経路生成戦略に従い，環境内を巡回する．v^i_{tar} に到着した後，再び目標戦略に従って新しい目標ノードを決定するといったサイクルを各エージェントが繰り返し，継続的な環境巡回を行う．

4.1 Adaptive meta target decision strategy (AMTDS)

この節では，AMTDS/LD，AMTDS/ESC などの手法のベースとなった Adaptive meta target decision strategy (AMTDS) [12] について説明する．AMTDS は単純な複数の目標決定戦略の中から，強化学習アルゴリズムである Q 学習によって，各エージェントが自身にとって最適な戦略を選択するメタ戦略学習である．また，この手法では環境内のすべてのノード v におけるイベント発生確率 $p(v)$ は既知であるという仮定を導入しており，$p^i(v) = p(v)$ とする．

エージェント i は AMTDS によって目標決定戦略 $s \in S$ を選択し，s に従って目標ノード v^i_{tar} を決定する．その後，経路生成戦略に従って v^i_{tar} に移動する．ここで，S はエージェントが選択可能な目標決定戦略の集合である．目標決定戦略については 4.1.1，経路生成戦略については 4.1.2 で詳細を説明する．v^i_{tar} に到着後，v^i_{tar} の決定時刻 t_0 から d_{travel}

ステップ後に v^i_{tar} に到着するまでの 1 ステップあたりのイベント処理量を以下の式で計算する．

$$u_{t_0,t_0+d_{travel}} = \frac{\sum_{t_0+1\leq t\leq t_0+d_{travel}} L_t(v^i_t)}{d_{travel}} \tag{10}$$

さらに，これを報酬として，選択した戦略 s の Q 値 $Q^i(s)$ を以下の式に従って更新する．

$$Q^i(s) \leftarrow (1-\alpha)Q^i(s) + \alpha \times u_{t_0,t_0+d_{travel}} \tag{11}$$

ここで，$\alpha(0<\alpha\leq 1)$ は学習率である．$Q^i(s)$ の更新後，i は次に選択する目標決定戦略 s_{next} を ε-Greedy 法によって決定する．ε-Greedy 法では，s_{next} を確率 ε でランダムに選択し，確率 $1-\varepsilon$ で以下の式に従って選択する．

$$s_{next} \leftarrow \arg\max_s Q^i(s) \tag{12}$$

4.1.1 目標決定戦略

[12] では，各エージェントに以下の 4 つの基本的な目標決定戦略を S として与えている．それぞれの戦略は単独でも使用可能な独立したものとなっている．単独での使用を想定し，競合回避のためにランダム性を取り入れたものも存在する．

Random selection (R)
環境全体のノード集合 V からランダムに v^i_{tar} を選ぶ．

Probabilistic greedy selection (PGS)
環境全体のノード集合 V 内のノード v におけるイベント発生量の推定値 $E^i(L_t(v))$ の上位 N_g 個のノードから，ランダムに 1 つ v^i_{tar} を選ぶ．この際に，学習や訪問をする v^i_{tar} の偏りを防ぐため，N_g 番目のノードと $E^i(L_t(v))$ の値が同じノードが存在する場合，そのノードをすべて含めた後，その中から v^i_{tar} をランダムに選んでいる．

Prioritizing unvisited interval (PI)
環境全体のノード集合 V 内のノード v における訪問間隔 $I^i_t(v)$ の上位 N_i 個のノードから，ランダムに 1 つ v^i_{tar} を選ぶ．この際に，学習や訪問をする v^i_{tar} の偏りを防ぐため，N_i 番目のノードと $I^i_t(v)$ の値が同じノードが存在する場合，そのノードをすべて含めた後，その中から v^i_{tar} をランダムに選んでいる．

Balanced neighbor-preferential selection (BNPS)
近隣のノードにイベント発生量が多いとエージェントが判断したとき，近隣を優先的に巡回する．v^i_{tar} の決定時にエージェントの現在地 v^i_t との距離が d_{rad} 以下のノード集合を近領域 V^i_{area} とする．ここで，V^i_{area} における 1 ステップあたりのイベント処理量の期待値 EV^i_t は以下の式で求められる．

$$EV^i_t = \frac{\sum_{v\in V^i_{area}} E^i(L_t(v))}{|V^i_{area}|} \tag{13}$$

エージェント i は近領域内のイベントを処理するか判断するための閾値 $EV_{threshold}$ と EV_t^i の値を比較し，$EV_t^i > EV_{threshold}$ の間は PGS によって近領域内から v_{tar}^i を選ぶ．その後，$EV_t^i \leq EV_{threshold}$ となった場合，環境全体を対象とし，PGS で v_{tar}^i を選ぶ．環境全体から v_{tar}^i を選択した後，V_{area}^i を更新する．更新後の V_{area}^i の 1 ステップあたりのイベント処理量の期待値を EV_{t+1}^i とし，$EV_{threshold}$ の値を以下の式に従って更新する．

$$EV_{threshold} \leftarrow EV_{threshold} + \alpha(EV_{t+1}^i - EV_{threshold}) \tag{14}$$

ここで，$\alpha(0 < \alpha < 1)$ は学習率である．また，$EV_{threshold}$ の初期値は初めに V_{area}^i を設定した際の EV_t^i の値である．

4.1.2 経路生成戦略

経路生成戦略は *gradual path generation* (GPG) を用いる．GPG では，まず v_{tar}^i までの最短経路をダイクストラ法を用いて生成し，その経路近辺でイベントが発生しやすいノードを経由するように経路を変更する．これにより，最短経路に従うよりも効率を高めることができる．しかし，経由するノードの増加によって v_{tar}^i への到着時間が遅れてしまい，逆に効率が下がってしまう．そのため，経由するノードに一定の制約をかけなければならない．そこで，GPG では経由可能なノード v を以下の式を満たすものとする．

$$\begin{cases} d(v_t^i, v) \leq d_{myopia} \\ d(v, v_{tar}^i) < k_{att}(d(v_t^i, v_{tar}^i)) \\ d(v_t^i, v) + d(v, v_{tar}^i) \leq k_{rover} d(v_t^i, v_{tar}^i) \\ \mathcal{P}^i(v_{tar}^i) + B_{drain}^i \times (d(v_t^i, v) + d(v, v_{tar}^i)) \leq b^i(t) \end{cases} \tag{15}$$

ここで，d_{myopia} はエージェントが現在地とするノードから経由地点とできるノードまでの距離の閾値であり，$k_{att}(0 < k_{att} < 1)$ は v_{tar}^i へ引き付ける力を表す係数である．また，$k_{rover}(1 < k_{rover})$ は経由地点を追加した新しい経路の距離の，最短距離からの増加率である．これらの条件を満たすノード集合を V_{sub}^i とすると，経由するノード $v_{subgoal}^i$ は以下の式で決められる．

$$v_{subgoal}^i \leftarrow \arg\max_{v \in V_{sub}^i} E^i(L_t(v)) \tag{16}$$

4.2 AMTDS with learning of dirt accumulation (AMTDS/LD)

AMTDS では，エージェントが環境内のすべてのノード v において，イベント発生確率 $p(v)$ をあらかじめ把握しているという仮定を導入した．しかし，実際の利用を想定すると，イベント発生確率が既知であることはまれである．特に，本研究のような清掃問題においては，イベントであるごみの発生確率を，エージェントが巡回しながら自ら学習するほうがより実用的である．そこで，AMTDS に環境のイベント発生確率の学習を加えた，AMTDS with learning of dirt accumulation (AMTDS/LD) [16] が提案された．

AMTDS/LD では，$p^i(v)$ の学習アルゴリズムの提案が行われた．以下でこのアルゴリズムについて説明する．

まず，1 ステップ終わるごとにエージェント i はすべてのノードについて，そのノードで最後にイベントが処理された時刻 t^v_{vis} から現在の時刻 t までの訪問間隔 $I^i_t(v)$ を以下の式に従って更新する．

$$I^i_t(v) = t - t^v_{vis} \tag{17}$$

その後，$I^i_t(v)$ と現在時刻 t でエージェント i が処理したイベント量 $L_t(v)$ を用いて，i の v における重要度 $p^i(v)$ を以下の式に従って更新する．

$$p^i(v) \leftarrow (1-\beta)p^i(v) + \beta\frac{L_t(v)}{I^i_t(v)} \tag{18}$$

ここで，$\beta(0 < \beta \leq 1)$ は学習率である．AMTDS/LD では各エージェントが独立した $p^i(v)$ の値を保持しているため，それに伴い，イベント発生量の推定値 $E^i(L_t(v))$ もそれぞれ異なる．また，同じノード上であってもそのノードへの訪問頻度によって，エージェントのイベントの発生しやすさの認識が異なる．このことによって，エージェント間の通信を用いずに $p^i(v)$ を用いた間接的な分業が可能になる．それにより，AMTDS と比べて競合を回避し，効率も向上した．

4.3　AMTDS with learning of event probabilities and enhancing divisional cooperation (AMTDS/EDC)

後で書く AMTDS/LD では，エージェント間の交渉を用いずに，$p^i(v)$ による間接的なコミュニケーションを用いて，エージェント間の分業を促進した．しかし，間接的なコミュニケーションでの効率改善には限界があり，エージェントが停止した場合などの環境変化に対応できない．そのため，エージェントに責任ノード集合 V^i_R を導入し，そのサイズを調整するエージェント間の交渉を AMTDS/LD に追加した，AMTDS with learning of event probabilities and enhancing divisional cooperation (AMTDS/EDC) [17] が提案された．領域分割の手法と大きく異なる点は，AMTDS/EDC では，エージェントの責任ノードを交渉によって直接決定するのではなく，それぞれの環境学習によって，各自が判断する点である．このことにより，エージェントの交換する情報が少なくなり，交渉の複雑さを抑制することができる．この手法により，環境の変化に対する頑健性を高め，全体の効率をさらに改善した．この節では，新たに導入した V^i_R と，エージェント間で行う交渉について説明する．

4.3.1　エージェント間の交渉に用いる情報

前述のとおり，AMTDS/EDC では，エージェント i は自身の責任ノード集合 $V^i_R(\subset V)$ を持ち，V^i_R はそのエージェントの重要度 $p^i(v)$ の降順に N^i_R 個のノード集合と定める．V^i_R，N^i_R の初期値はそれぞれ V，$|V|$ と定める．AMTDS や AMTDS/LD とは異なり，AMTDS/EDC では，PGS や BNPS を用いて次の目標ノード v^i_{tar} を決定する際，環境全体のノード集合 V ではなく，V^i_R を用いる。これにより，選択対象のノードを減らし，分業をさらに促進

することができる．エージェントが充電基地に戻るまでの間に，充電基地に戻るまでの環境の学習により，$p^i(v)$ の値がエージェント間の交渉により N_R^i の値が更新されている．このため，i は充電基地に戻った際，自身の責任ノードの集合 V_R^i を更新された $p^i(v)$，N_R^i を用いて再定義する．ここで，$p^i(v)$ の値が同じノードが複数ある場合，ランダムに並べられる．

エージェントは交渉に用いる次の 2 つの情報を計算する．1 つは，エージェントの責任ノード集合の重要度の総和 p_{sum}^i である．p_{sum}^i の値は以下の式で計算される．

$$p_{sum}^i = \sum_{v \in V_R^i} p^i(v) \tag{19}$$

この値は，責任ノードのイベント発生確率の総和であるため，この値が大きいほどそのエージェントはタスクを多く持っているといえる．

もう 1 つは，エージェントの責任ノード集合 V_R^i の重心 $C^i = (x_c^i, y_c^i)$ である．C^i は以下の式で x_c^i, y_c^i を計算し，これに最も近いノードを重心ノードと定める．

$$x_c^i = \sum_{v \in V_R^i} \frac{p^i(v)}{p_{sum}^i} x_v \tag{20}$$

$$y_c^i = \sum_{v \in V_R^i} \frac{p^i(v)}{p_{sum}^i} y_v \tag{21}$$

最短距離 $d(v_i, v_j)$ に関して，$d(C^i, v) < d(C^j, v)$ のとき，ノード v を訪れるコストは，エージェント i の方がエージェント j よりも小さい，つまり，この場合は，i が v を担当した方が移動のコストが少なく，望ましいといえる．

4.3.2　エージェント間の交渉

各エージェントは 4.3.1 で示した情報を用いて，エージェント間で Negotiation for Balancing Tasks (公平性のための交渉) と Negotiation for Trade-Off of Reaponsibility (改善のための交渉) の 2 種類の交渉を行う．エージェント間の交渉は，通信コストの抑制のため，1 対 1 の通信のみとし，マネージャーなどを介した複数のエージェント間での交渉は用いない．エージェントは以下に述べる 2 種類の交渉を行うことで，V_R^i の改善による性能向上と，$p^i(v)$ の最適化による分業の促進を図る．以下で，2 種類の交渉について詳細に説明する．

Negotiation for Balancing Tasks (公平性のための交渉)

この交渉は，エージェント間のタスクを公平にし，一部のエージェントへのタスクの偏りを抑制することを目的としている．このため，交渉相手のエージェントよりも自身の重要度の総和が大きい場合，自身の担当ノードの重要度の小さいノードの重要度の一部を相手に受け渡す．具体的な手段としては，まずエージェント i と j が以下の式を満たすかを判断する．

$$1 + T_c < \frac{P_{sum}^i}{p_{sum}^j} \tag{22}$$

この式を満たした場合，i と j は公平性のための交渉を行う．ここで，$T_c(0 < T_c \ll 1)$ はエージェント間のタスクの差の閾値である．i は，i よりも j の方が距離が近いノードを $p^i(v)$ の降順に並べたノード集合 $V_R^{i,j}$ を以下の式で定義する．

$$V_R^{i,j} = v \in V_R^i | d(C^i, v) > d(C^j, v) \tag{23}$$

i は $V_R^{i,j}$ から，$p^i(v)$ の小さい e_g 個のノードにおいて，以下の式に従って i の重要度の一部を j に受け渡す．

$$p^j(v) \leftarrow p^j(v) + p^i(v) \times \delta \tag{24}$$

$$p^i(v) \leftarrow p^i(v) \times \delta \tag{25}$$

ここで，$\delta(0 < \delta < 1)$ は重要度を受け渡す割合である．また，i から j に受け渡す個数を表す e_g を，以下の式に従って求める．

$$e_g = \min(N_R^{i,j} - 1, N_{gmax}^i, \lfloor \frac{p_{sum}^i}{p_{sum}^j} \times \gamma \rfloor) \tag{26}$$

ここで，$N_{gmax}^i(0 < N_{gmax}^i < N_R^i)$ は，急激な変化を抑制するための受け渡すノード数の上限であり，γ は受け渡すノード数を調整するためのパラメータである．重要度の受け渡し後，i と j のノード集合のサイズ N_R^i，N_R^j を以下の式に従って更新する．

$$N_R^i \leftarrow N_R^i - e_g \tag{27}$$

$$N_R^j \leftarrow \min(|V|, N_R^j + e_g) \tag{28}$$

Negotiation for Trade-Off of Reaponsibility (改善のための交渉)

この交渉は，エージェントのタスクを交換し，全体への影響を抑えながらエージェントの巡回コストを減らし，全体の効率を高めることを目的としている．このため，自身とほぼ同じ重要度を持つエージェントに対し，自身よりも移動距離が短いノードをお互いに受け渡す．具体的な手順としては，まずエージェント i と j が以下の式を満たすかを判断する．

$$1 - T_c < \frac{p_{sum}^i}{p_{sum}^j} < 1 + T_c \tag{29}$$

この式を満たした場合，i と j は改善のための交渉を行う，ここで，$T_c(0 < T_C \ll 1)$ はエージェント間のタスクの差の基準値である．i の重要度の一部を j 受け渡す．i から j に受け渡す個数を示す e_g は以下の式に従って求められる．

$$e_g = \min(N_R^i - 1, N_{cmax}^i) \tag{30}$$

ここで，$N_{cmax}^i(0 < N_{cmax}^i < N_R^i)$ は，急激な変化を抑制するための受け渡すノード数の上限である．この交渉は責任ノードの調整をする目的で行うことから，$N - i_{cmax}$ の値は N_{gmax}^i よりも小さい．重要度の受け渡し後，公平性のための交渉と同様に，N_R^i，N_R^j を更新する．この交渉は双方向に行われるため，j から i にも責任ノードの一部が受け渡される．

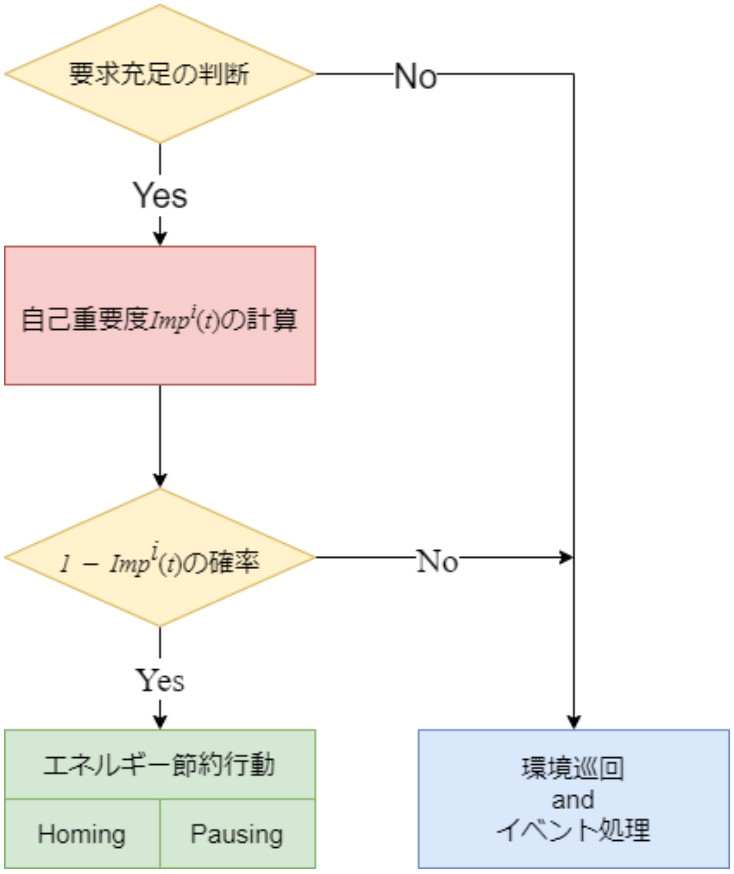

図 1: エージェントの行動選択

この手法により，AMTDS/LD に比べ，タスクが均一化されたことや，エージェントの担当領域の偏りが増し，分業が促進されたことで，全体としての効率は向上した．これは，エージェントのタスク量が同等になったことで，カバーしきれないタスクが減少したことに加え，エージェントの担当領域の偏りの増加により，移動コストが減少したためである．また，この手法により，エージェントが自律的に複数エージェントのチームを組み，より広い領域を巡回する generalists と，より狭い領域を巡回する specialists に分かれた．このことにより，AMTDS/LD に比べ，環境の変化に対し，より柔軟に対応できるようになった．

4.4　AMTDS for energy saving and cleanliness (AMTDS/ESC)

AMTDS や AMTDS/LD，AMTDS/EDC では，イベント残存時間をより小さくすることが求められた．しかし，エネルギー消費量も同時に小さくすることができればより有用である．つまり，イベント残存時間だけではなく，エネルギー消費を節約することが求められることもある．したがって，目標決定戦略を選択する Q 学習における報酬の計算方法を，式 (10) から以下の式に変更し，報酬を $r_{t_0,t_0+d_{travel}}$ とした，AMTDS for energy saving

and cleanliness (AMTDS/ESC) [5] が提案された．

$$r_{t_0,t_0+d_{travel}} = \frac{\sum_{t_0+1\leq t\leq t_0+d_{travel}} L_t(v_t^i)}{d_{travel} \times \sum_{t_0+1\leq t\leq t_0+d_{travel}} E_t(i)} \tag{31}$$

これにより，エージェントはより少ないエネルギーでより多くのイベントを処理できる目標決定戦略を決定する．なお，目標決定戦略 s_{next} を ε-Greedy 法によって決定するのは，4.1 と同様である．

AMTDS/ESC ではさらに，エネルギー節約行動として，Homing と Pausing の 2 つの行動が導入された．図 1 はエージェントにおけるエネルギー節約行動にかかわる行動選択のフローチャートである．このフローチャートが示すように，2 つの条件が満たされた場合，エージェントはエネルギー節約行動を行う．その際に，エージェントは 2 つのエネルギー節約行動のどちらかを行う．エージェントは，不要と思われる行動を排除するために，*Homing* または *Pausing* の 2 つのエネルギー節約行動をとる．以下でそれぞれの詳細を説明する．

4.4.1 要求充足の判断

エージェントは，環境の現状を把握し，必要なタスクの品質と比較することで，エネルギー消費を抑える必要がある．しかし，どのエージェントも実際の状態を把握することができないため，定期的に推定を行う必要がある．そこで，エージェント i は，自身が持っている $p^i(v)$ を用いて，以下の式に従って，時刻 t における総イベント量 $E^i(V_t)$ を推定する．

$$E^i(V_t) = \sum_{v\in V} E^i(L_t(v)) \tag{32}$$

その後，要求条件を満たしているのかを判断するために，以下の式を用いている．

$$E^i(V_t) \leq D_{req} \tag{33}$$

この式を満たしていれば i は自己重要度評価を行い，満たしていなければ，通常の環境巡回とイベント処理を行う．

4.4.2 自己重要度評価

エージェントは，4.4.1 で述べた方法で要求条件を満たしているかを判断し，満たしていた場合，自分の貢献度を評価し，自分のエネルギー節約行動が環境の将来に与える影響を予測する．そのために，エージェント i の時刻 t における自己重要度 ${S_{ass}}^i(t)$ を導入する．これは，i が協調タスクのために巡回を継続すべきか，エネルギー節約行動をとって巡回を停止すべきかを決定するための，i の最近の貢献度合いを表している．

$SelfAss^i(t)$ を計算するために，以下の 3 つのパラメータを $\forall i \in A$ に対して定義する．

$$U_s^i(t_c) = \frac{\sum_{t_c - T_s < t \leq t_c} L_t(v^i(t))}{T_s} \tag{34}$$

$$U_l^i(t_c) = \frac{\sum_{t_c - T_l < t \leq t_c} L_t(v^i(t))}{T_l} \tag{35}$$

$$U_f^i(t_c) = \frac{\sum_{t_c < t \leq t_c + T_f} E^i(L_t(v^i(t)))}{T_f}, \tag{36}$$

ここで，t_c は現在の時刻，T_s と T_l $(0 < T_s < T_l)$ は過去の短期・長期の評価期間，T_f は将来の評価期間，$v^i(t)$ は t において i がいた，またはいる予定のノードを表す．直感的には，$U_s^i(t_c)$ と $U_l^i(t_c)$ は過去に i が処理したイベント数を表し，$U_f^i(t_c)$ は将来 $t_c + T_f$ までに i が処理するであろうイベント数の推定値であるといえる．つまり，エージェントは以下の 2 つを考慮して自己重要度を求めている．

(1) ある期間における過去の貢献度

(2) 重要な領域を発見し, その後の行動で多くのイベントを処理できるか

なお，式 (36) でのみ，$L_t(v^i(t))$ の代わりに期待値 $E^i(L_t(v^i(t)))$ が使用されていることに注意されたい．

ここで，自己重要度 ${S_{ass}}^i(t)$ $(0 \leq {S_{ass}}^i(t) \leq 1)$ を次の式のように定義する．

$${S_{ass}}^i(t) = \begin{cases} 0 & (U_l^i = 0 \text{ のとき}) \\ \dfrac{U_s^i + U_f^i}{U_l^i} & (U_s^i + U_f^i \leq U_l^i \text{ のとき}) \\ 1 & (\text{その他}) \end{cases} \tag{37}$$

その後，i は以下の式で求められる確率 $P^i(t)$ でエネルギー節約行動を選択する．

$$P^i(t) = 1 - {S_{ass}}^i(t) \tag{38}$$

したがって，自己重要度が低いと，エージェントのエネルギー節約行動が促進される．

4.4.3 帰還動作 (*Homing*)

Homing とは，バッテリ残量にかかわらず，巡回を中止して充電基地に戻ることで，エネルギーを節約する動作のことである．エージェント i は，充電残量が少なくなり，以下の式を満たしたとき，T_{homing} ステップ移動するごとに要求充足度 (式 (32)) をチェックする．

$$b^i(t) < k_{homing} \cdot B_{max} \tag{39}$$

ここで，正の整数 $HomingCheck$ は *Homing* による頻繁な充電基地への帰還を避けるためのパラメータ，k_{homing} $(0 < k_{homing} < 1)$ はバッテリ残量低下を判断するためのパラメータである．式 (32) を満たさない場合，i は巡回を継続し，満たす場合は，確率 $P_i(t)$ で i が現在の v_{tar}^i を v_{charge} に変更する *Homing* を行い，充電基地についたら充電を行う．また，*Homing* 中はイベント処理は行うが，要求充足の判定や自己重要度の計算は行わない．なお，充電基地から遠い地点を巡回しているエージェントがこのチェックをする前に充電基地に帰還することもありうる．

4.4.4　待機動作 (*Pausing*)

Pausing とは，充電完了後，エージェントはすぐに巡回を再開せずに，待機する動作のことである．エージェント i が時刻 t に充電が完了したとき，要求充足度 (式 (32)) をチェックし，満たす場合は一定の休止時間 S_{pause} ステップ充電基地で待機する．満たさない場合は，巡回を再開する．

5　提案手法

AMTDS/ESC では，要求値 D_{req} とエネルギー節約行動の導入によって，品質要求を満たしつつ，エネルギー消費量を削減することができた．しかし，*Homing* と *Pausing* の 2 つのエネルギー節約行動のうち，どちらか片方しか行うことができなかった．また，全てのエージェントが同一の視点で要求充足の判断を行っていたため，環境の状態の推定が不十分であり，要求条件を満たす上で不必要な巡回を行っているため，エネルギーを余分に消費してしまっている．これらの問題を解消するため，本研究では，AMTDS for energy saving under the requirement (AMTDS/ER) と，AMTDS/EDC に加えた AMTDS for energy saving under the requirement with communications (AMTDS/ERC) を提案する．以下では，ベースとしている手法との違いに絞って説明する．

5.1　AMTDS for energy saving under the requirement (AMTDS/ER)

この節では，AMTDS/ESC をベースに，エージェントがそれぞれの視点で将来の環境状態を予測する方法を学習し，充電基地についた際に，未来の環境状態を予測し，待機時間を動的に決定する手法である，AMTDS/ER について説明する．また，エージェント数を減らす手法についても説明するここをもっとちゃんと書く

5.1.1　未来のイベント量の予測

AMTDS/ER では，5.1.3 で詳しく説明するが，*Pausing* による待機時間を可変に変更する．その際に，未来のイベント量の予測をする必要があるため，その方法について説明する．

エージェント i は，将来の時刻 $t_c + T$ における $\forall v \in V$ の期待値 $E^i(L_{t_c+T}(v))$ を，以下の式に従って求める．

$$E^i(L_{t_c+T}(v)) = p(v) \times \{(t_c + T) - t^v_{vis}\} \tag{40}$$

ここで，$T > 0$ は i がどの程度先の未来を推定するかを指定するパラメータである．

5.1.2　予測の自律的補正学習

5.1.1 で説明した未来のイベント量の予測を行った後，エージェント i は式 (32) を確認する．しかし，この式 (32) は T に対して単調増加であるため，この推定は，均一な作業量に基づく理想的な場合であり，他エージェントによる努力 (巡回) は無視されている．特に，i がエネルギー節約や充電のために巡回を停止していても，他エージェントは要求値 D_{req} を維持しようとする．さらに，各エージェントは異なる特徴を持つ場所を訪問するため，他エージェントによる努力は，個々の視点によって明らかに異なる．したがって，エージェントは，エネルギー節約行動中における他エージェントの努力の可能性を学習する必要がある．

これを調整するために，学習パラメータ K^i を $\forall i \in A$ に導入する．そして，エージェント i は，式 (32) の代わりに，以下の式を用いて要求条件を満たしているかを判断する．

$$\sum_{v \in V} E^i(L_{t_c+T}(v)) \div K^i \leq D_{req} \tag{41}$$

さらに，K^i は個々に次の式に従って更新される．

$$\begin{cases} K^i \leftarrow (1-\alpha_k)K^i + \alpha_k \dfrac{D_{req}}{E^i(D_t)} K^i & (E^i(D_t) \leq D_{req} \text{ のとき}) \\ K^i \leftarrow K^i - \left(\dfrac{E^i(D_t)}{D_{req}} - 1 \right) & (E^i(D_t) > D_{req} \text{ のとき}) \end{cases} \tag{42}$$

ここで，t におけるイベント量の推定値は $E^i(D_t) = \sum_{v \in V} E^i(L_t(v))$ で定義され，$\alpha_k > 0$ は学習率である．K^i がいつ更新されるかは，5.1.3 で説明する．

Algorithm 1 PLength: To decide pausing time-length.

Require: $S_{pause} > 0$, $\gamma_p = 0$, $T_{\gamma_p} > 0$

```
while γ_p ≤ T_{γ_p} do
    T ← (γ_p + 1) · S_pause // T is used in Formula (41)
    if Formula (41) holds then
        γ_p ← γ_p + 1
    else
        break
    end if
end while
// エージェントは待機時間 γ_p · S_pause の Pausing を行う
// もし γ_p = 0 の場合, エージェントは即座に充電基地を出発し，巡回を再開する
return γ_p · S_pause.
```

5.1.3　エネルギー節約行動の修正

ここでは，より多くのエネルギーを削減するための，4.4.3 と 4.4.4 で説明したエネルギー節約行動の修正点について説明する．AMTDS/ESC では，時刻 t に充電が完了する

と，エージェント i が要求条件 (式 (32)) を満たす場合，i は固定の待機時間 $PausingInt$ ステップ $Pausing$ を行っていた．

しかし，提案手法では，エージェントは推定状態の条件に応じて可変の待機時間をとる．i は以下で説明する可変長の $Pausing$ を確率 $P_i(t)$ で行い，t で充電を完了する．ただし，例外として，i は $Homing$ によって充電基地 v_{base} に戻ったときのみ，$P_i(t)$ を計算せずに，常に可変長の $Pausing$ を行う．これは，$Homing$ のみでは，エージェントによる,「イベント処理 $\rightarrow$ 充電 $\rightarrow$ イベント処理 $\rightarrow \cdots$」のサイクルが短縮しただけで，エネルギー消費量にはあまり影響しない．既に環境が要求条件を満たしていて，自分は余力があっても充電基地に戻るべきと判断し $Homing$ を行っているため，$Pausing$ を行うことは自然であり，エネルギー消費量の減少が見込める．

可変長の $Pausing$ について説明する．まず，i は待機時間を以下のように決定する (Algorithm 1 の PLength 参照)．初期状態では，エージェント i は $\gamma_p = 0$，$T = S_{pause}$ と設定する．時刻 t にバッテリが満充電になると，i は要求充足度をチェックし (式 (41))，満たしていない場合は，i は直ちに充電基地から出発し，巡回を再開する．つまり，待機時間は 0 となる．満たしている場合は，γ_p を 1 だけインクリメントし，$T \leftarrow (\gamma_p + 1) \cdot S_{pause}$ とセットする．次に，i は再び要求度をチェックし，待機時間を決定するか，$\gamma_p = T_{\gamma_p}$ となるまでこの操作を繰り返す．ただし，T_{γ_p} はこの反復の最大値である．その後，i は $\gamma_p \cdot S_{pause}$ ステップだけ $Pausing$ を行う．エージェントはこの可変長の $Pausing$ の後は，必ず充電基地を出発する．

前述したように，エージェント i が $Homing$ で充電基地に戻る場合，AMTDS/ESC とは異なり，充電完了後に必ず可変長の $Pausing$ を行う．この際に，上記のアルゴリズムによって待機時間が 0 になる可能性もなる．以下では，可変長の $Pausing$ を単に $Pausing$ と呼ぶ．

表 1: エージェントに関するパラメータ

種類	パラメーター	値
エージェント数	—A—	20
バッテリ	B^i_{max}	900
	B^i_{drain}	1
	k^i_{charge}	3
経路生成戦略	d_{myopia}	10
	k_{att}	1.0
	k_{rover}	1.2

表 2: 目標決定戦略のパラメーター

目標決定戦略	パラメーター	値
PGS	N_g	5
PI	N_i	5
BNPS	α	0.1
	d_{rad}	15
AMTDS	α	0.1
	ε	0.05
AMTDS/LD	β	0.1

表 3: エネルギー節約行動に関するパラメーター

種類	パラメーター	値
自己重要度評価	T_s	20
	T_l	50
	T_f	10
Homing	T_{check}	100
	k_{homing}	1/3
Pausing	T_{basic}	100
AMTDS/ERL	T_{hp}	500,000

参考文献

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

[2] Abdenour Benkrid, Abdelaziz Benallegue, and Noura Achour. Multi-robot coordination for energy-efficient exploration. *Journal of Control, Automation and Electrical Systems*, Vol. 30, No. 6, pp. 911–920, 2019.

[3] Gennaro Notomista, Siddharth Mayya, Yousef Emam, Christopher Kroninger, Addison Bohannon, Seth Hutchinson, and Magnus Egerstedt. A resilient and energy-aware task allocation framework for heterogeneous multirobot systems. *IEEE Transactions on Robotics*, Vol. 38, No. 1, pp. 159–179, 2022.

[4] Lingying Wu and Toshiharu Sugawara. Strategies for Energy-Aware Multi-agent Continuous Cooperative Patrolling Problems Subject to Requirements. In Matteo Baldoni, Mehdi Dastani, Beishui Liao, Yuko Sakurai, and Rym Zalila Wenkstern, editors, *PRIMA 2019: Principles and Practice of Multi-Agent Systems*, pp. 585–593, Cham, 2019. Springer International Publishing.

[5] Lingying Wu, Ayumi Sugiyama, and Toshiharu Sugawara. Energy-efficient strategies for multi-agent continuous cooperative patrolling problems. *Procedia Computer Science*, Vol. 159, pp. 465–474, 2019.

[6] Katsuya Hattori and Toshiharu Sugawara. 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,*, pp. 281–288. INSTICC, SciTePress, 2021.

[7] Gleb K Tevyashov, Mark V Mamchenko, Andrey N Migachev, Rinat R Galin, Konstantin A Kulagin, Petr M Trefilov, Rodion O Onisimov, and Nikolay V Goloburdin. Algorithm for multi-drone path planning and coverage of agricultural fields. In *Agriculture Digitalization and Organic Production*, pp. 299–310. Springer, 2022.

[8] Bernát Wiandt and Vilmos Simon. Autonomous graph partitioning for multi-agent patrolling problems. In *2018 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pp. 261–268. IEEE, 2018.

[9] Mehdi Othmani-Guibourg, Amal El Fallah-Seghrouchni, Jean-Loup Farges, and Maria Potop-Butucaru. Multi-agent patrolling in dynamic environments. In *2017 IEEE international conference on agents (ICA)*, pp. 72–77. IEEE, 2017.

[10] Xin Zhou, Weiping Wang, Tao Wang, Yonglin Lei, and Fangcheng Zhong. Bayesian reinforcement learning for multi-robot decentralized patrolling in uncertain environ-

ments. *IEEE Transactions on Vehicular Technology*, Vol. 68, No. 12, pp. 11691–11703, 2019.

[11] Mehdi Othmani-Guibourg, Amal El Fallah-Seghrouchni, and Jean-Loup Farges. Decentralized multi-agent patrolling strategies using global idleness estimation. In *International Conference on Principles and Practice of Multi-Agent Systems*, pp. 603–611. Springer, 2018.

[12] Keisuke Yoneda, Chihiro Kato, and Toshiharu Sugawara. 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)*, Vol. 2, pp. 216–223, 2013.

[13] Ehsan Latif, Yikang Gui, Aiman Munir, and Ramviyas Parasuraman. Energy-aware multi-robot task allocation in persistent tasks, 2021.

[14] Tomoki Yamauchi, Yuki Miyashita, and Toshiharu Sugawara. 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*, pp. 1427–1435, Richland, SC, 2022. IFAAMAS.

[15] Hang Ma Jiaoyang Li TK Satish and Kumar Sven Koenig. Lifelong multi-agent path finding for online pickup and delivery tasks. 2017.

[16] Ayumi Sugiyama and Toshiharu Sugawara. Meta-strategy for cooperative tasks with learning of environments in multi-agent continuous tasks. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pp. 494–500, 2015.

[17] Ayumi Sugiyama, Vourchteang Sea, and Toshiharu Sugawara. Emergence of divisional cooperation with negotiation and re-learning and evaluation of flexibility in continuous cooperative patrol problem. *Knowledge and Information Systems*, Vol. 60, No. 3, pp. 1587–1609, 2019.