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．
ここをもっとちゃんと書く

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．この問題においてエージェントの行動を考慮する手法には，環境を分割し，その各作業領域をエージェントに割り当て，その環境内を巡回する手法 [6, 7, 8, 9, 10] と，明示的に領域分割はせずに，各エージェントが自律的に適切な戦略を用いて環境を巡回する手法の 2 つがある．[11] では，後者の手法は前者の手法よりも少なくとも同程度に優れていると述べられている．本研究の目的であるエネルギー削減は両者の手法で考えられるが，本稿では後者の手法に着目する．

MACPP 問題への同アプローチをとった研究として，例えば，[12] 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，[13] では，巡回問題が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され，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・計画アルゴリズムが提案されている．さらに，[14] 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 を予測する方法が提案されている．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．[15] 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) という手法が提案されている．[16] では，[15] において既知としていた，各地点のイベント発生確率を学習するように拡張した手法 (AMTDS with learning of dirt accumulation, AMTDS/LD) が提案され，さらに，エージェント間の協調作業のための分業行動を促進させるために，軽い処理の交渉を加えた手法 (AMTDS with learning of event probabilities and enhancing divisional cooperation, AMTDS/EDC) が提案された．しかし，これら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, 17] 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，[17] 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．

$$b^i(t) \leq \mathcal{P}^i(v) + d(v^i_t, v^i_{tar}) \times B^i_{drain} \tag{3}$$

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

$$v^i_{tar} \leftarrow v^i_{base} \tag{4}$$

エージェントは v^i_{base} 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．

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

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 $T^{i,j}_{lst}$ を保持しており，$T^{i,j}_{lst} + B$ 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．[18, 19] 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) [15] 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．

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

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，$\alpha (0 < \alpha \le 1)$ 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

[15] 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_{tar}^i を選ぶ．この際に，学習や訪問をする v_{tar}^i の偏りを防ぐため，N_i 番目のノードと $I_t^i(v)$ 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_{tar}^i 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_{tar}^i の決定時にエージェントの現在地 v_t^i との距離が d_{rad} 以下のノード集合を近領域 V_{area}^i とする．ここで，V_{area}^i 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_t^i は以下の式で求められる．

$$EV_t^i = \frac{\sum_{v \in V_{area}^i} E^i(L_t(v))}{|V_{area}^i|} \tag{13}$$

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^i_{sub} とすると，経由するノード $v^i_{subgoal}$ は以下の式で決められる．

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

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)

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) [20] 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_R^i 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 $V_R^i(\subset V)$ を持ち，V_R^i はそのエージェントの重要度 $p^i(v)$ の降順に N_R^i 個のノード集合と定める．V_R^i，N_R^i 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_{tar}^i を決定する際，環境全体のノード集合 V ではなく，V_R^i 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．

$$1 - T_c < \frac{p^i_{sum}}{p^j_{sum}} < 1 + T_c \tag{29}$$

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．

$$e_g = \min(N^i_R - 1, N^i_{cmax}) \tag{30}$$

ここで，$N^i_{cmax}(0 < N^i_{cmax} < N^i_R)$ 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^i_{gmax} 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^i_R，N^j_R 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．

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．

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

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．

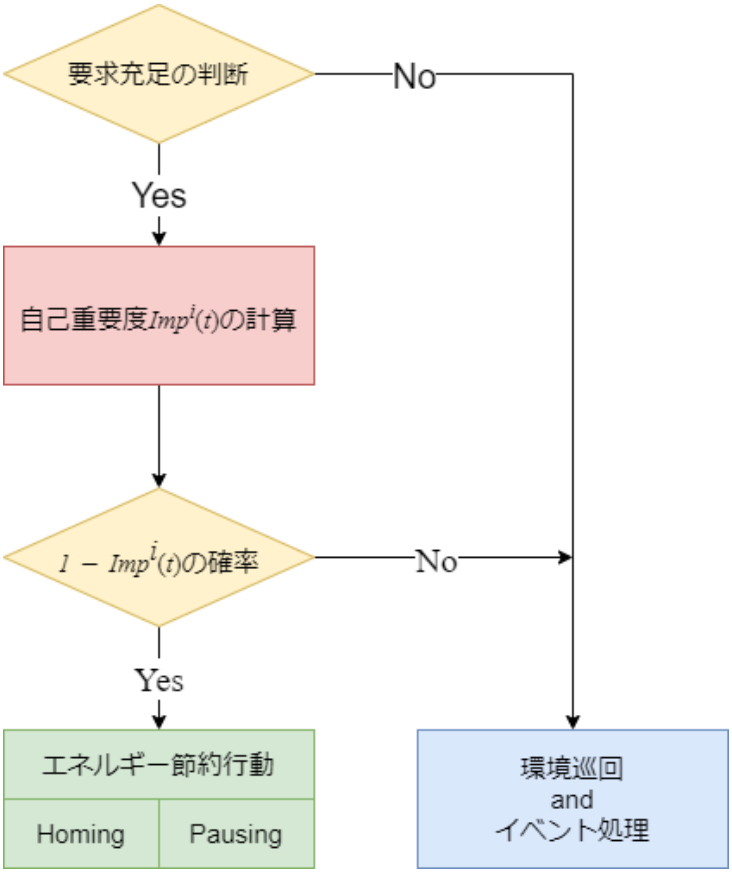


図 1: エージェントの行動選択

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，品質要求を満たした上で，不要なエージェントを停止する 2 通りの手法についても説明する．

5.1.1　未来のイベント量の予測

AMTDS/ER では，5.1.3 項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 項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．

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

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 項で説明する．

Algorithm 1 PLength: To decide pausing time-length.

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

while $\gamma_p \leq T_{\gamma_p}$ **do**
  $T \leftarrow (\gamma_p + 1) \cdot S_{pause}$ // T is used in Formula (41)
  if Formula (41) holds **then**
    $\gamma_p \leftarrow \gamma_p + 1$
  else
    break
  end if
end while
// エージェントは待機時間 $\gamma_p \cdot S_{pause}$ の *Pausing* を行う
// もし $\gamma_p = 0$ 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 $\gamma_p \cdot S_{pause}$.

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 項と 4.4.4 項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 → 充電 → イベント処理 → ・・・」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．

5.2　不要なエージェントの停止

エネルギー消費抑制の観点からは，要求条件を満たした上で，環境巡回に必要なエージェント数が十分であれば，たとえ必要最小数までではなくとも，エージェントを一時的に休止するだけでなく，一部のエージェントの巡回を停止することが有効である．さらに，これらの停止したエージェントは，他の場所で使えたり，エージェントが故障した場合のバックアップとして使用することができる．しかし，複雑でイベント発生確率の分布が一様でない環境では，どのエージェントを停止すれば要求条件を満たしたままでいられるかや，どれだけのエージェントが必要かを事前に把握することは困難である．

6.2.3 項で詳細に説明するように，K^i はエージェントによって大きなばらつきがあり，大きく分けて 2 つのグループに分かれていることが分かった．特に，K^i が比較的大きいグループに属するエージェントは，エネルギー消費を抑えるために比較的長い時間 *Pausing* を行っていたため，このような K^i が最も大きいエージェント $i_{de} \in A$ を停止させる手法を提案する．

$$i_{de} = \arg\max_{i \in A} K^i \tag{43}$$

その後，いくつかのエージェントの K^i がまだ大きい場合，品質要求が満たされているのならば，全てのエージェントの K^i が相対的に小さくなるまで同様の停止処理を繰り返す．ここで，6.2.5 項で詳細に説明するように，K^i の降順にエージェントを停止することによって，停止による $D(s)$ の一時的な悪化を防いでいる．

本実験では，不要なエージェントの停止手法として，関数 PLength が呼び出された回数を比較して停止する手法 (Deactivation on Count) と，エージェントの待機時間を比較して停止する手法 (Deactivation on Time) の 2 つの手法を提案する．

英語名については要検討

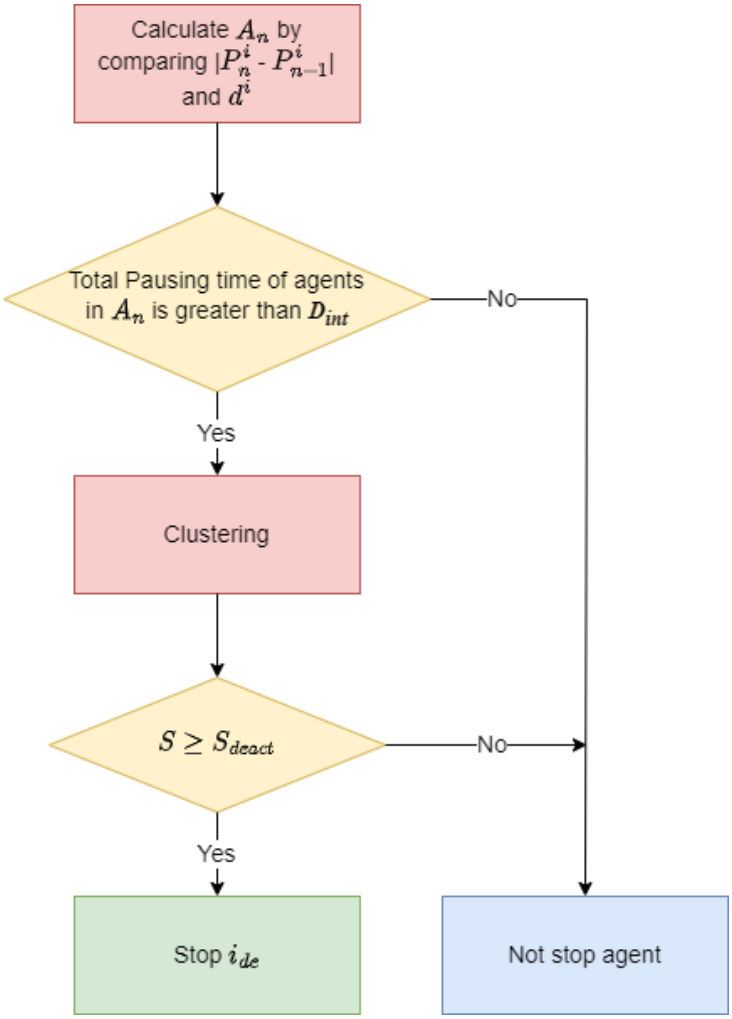


図 2: エージェントの停止選択

5.2.1　Deactivation on Count

システムは $D_{int} > 0$ ステップごとに，関数 PLength が全てのエージェントによって呼び出された回数である x_{pause} をカウントし，x_{pause} と $K^{i_{de}}$ が大きい場合，すなわち，以下の式を満たすかを確認する．

$$x_{pause} \geq N_{deact} \text{ and } \max_{i \in A} K^i \geq K_{deact}, \tag{44}$$

この式を満たす場合，i_{de} は停止する．ここで，$K_{deact} > 0$ は停止の閾値を示し，N_{deact} はシステムから見て，最近の D_{int} の間に複数の $Pausing$ が試みられたかどうかを判断する閾値を示す．ここで，関数 PLeangth によって $\gamma_p = 0$ の場合でもカウントすることに注意されたい．これは，x_{pause} と N_{deact} の式は，要求品質をみたす際に現在どの程度余裕があるのかを確認している．図 1 より，$Homing$ や $Pausing$ を行うということは，現在は品質要求を満たしているとエージェントが判断しているため，このようなカウント方法にしている．

5.2.2　Deactivation on Time

6.2.3 項で詳細に説明するように，AMTDS/ER では，イベント処理中心のグループと，エネルギー節約中心のグループの 2 グループに分かれる．Deactivation on Time は，その性質とエージェントの待機時間を用いて停止するか否かを判断する手法である．

まず，エージェント i の待機時間の通常変化量 d^i を導入する．エージェントごとに待機時間の変化量は異なり，これを学習することにより，i が品質要求を考慮して待機時間を変化させている途中なのか，均衡状態なのかを判断できるようになる．これを式 (45) のように D_{int} に対して $\beta_s(\beta_s > 1)$ で割ったステップ数である，D_{learn} ステップごとに式 (46) に従って更新する．

$$D_{learn} = \frac{D_{int}}{\beta_s} \tag{45}$$

$$d^i \leftarrow \alpha_d \times |T_n^i - T_{n-1}^i| + (1 - \alpha_d) \times d^i \tag{46}$$

ここで，T_n^i は全ての正整数 n について，$(n-1) \times D_{learn} \sim n \times D_{learn}$ ステップでの i の総待機時間である．つまり，$|T_n^i - T_{n-1}^i|$ とは，D_{learn} ステップごとの総待機時間の変化量を表す．さらに，α_l は学習率であるが，これは以下の式に従って，2 通りの値をとる $(\alpha_1 > \alpha_2)$.

$$\alpha_d = \begin{cases} \alpha_1 & (|T_n^i - T_{n-1}^i| \leq d^i \text{のとき}) \\ \alpha_2 & (|T_n^i - T_{n-1}^i| > d^i \text{のとき}) \end{cases} \tag{47}$$

図 2 は，Deactivation on Time での停止をするかどうかの判断のフローチャートである．$(n-1) \times D_{int} \sim n \times D_{int}$ ステップでの i の総待機時間である P_n^i を用いて，D_{int} ステップごとに $|P_n^i - P_{n-1}^i|$ と $\beta_s \times d^i$ を比較する．$|P_n^i - P_{n-1}^i| \leq d^i$ のエージェントの集合を A_n として，以下の式を満たすか確認する．

$$\sum_{i \in A_n} P_n^i \geq D_{int} \tag{48}$$

この式を満たせば，i_{de} を停止後も残りのエージェント達は，少なくとも $(D_{learn} - P_n^{i_{de}})$ ステップ待機しているため，この待機時間を減らすことで i_{de} の分のイベント処理をカバーできる．また，式 (46) で α_d が 2 通りの値をとる理由は，α_d が固定の場合，$D_{int} > D_{learn}$ より，i が待機時間を変化中の時でも直前の数回の更新で d^i に影響を与え，大きな値となってしまう．これにより，本当は待機時間を変更させているのに式 (48) を満たしてしまう恐れがある．これを防ぐために，α_d は 2 通りの値をとり，$\alpha_1 > \alpha_2$ としている．さらに，エージェントを停止していき，エージェント数が少なくなっていくと，品質要求を満たすために，待機時間が大きかったエージェントも，イベント処理を多く行うために待機時間を小さくしていかなければならない．最終的に，このエージェントの待機時間が 0 になり，品質要求を満たす上でこれ以上エージェントを停止できなかったはずなのに停止してしまい，残りのエージェント数では，全力でイベント処理を行っても品質要求を満たせないということは避けなければならない．そのため式 (48) では，A の総待機時間ではなく，待機時間が均衡状態であるエージェントの集合である A_n の総待機時間を用いている．

さらに，全ての巡回エージェントの K^i と P_n^i をそれぞれ平均 0，標準偏差 1 に標準化し，k-means 法により 2 つのクラスターにクラスタリングする．その後，以下の方法で i のシルエット係数 s^i を計算し，それの平均シルエット係数 S を求める．

(1) クラスタ内の凝集度として，i が属するクラスタ C_{in} の他の点までの平均距離 c_{in}^i を計算

(2) 別クラスタからの乖離度として，i が属しないクラスタ C_{out} に属する点までの平均距離 c_{out}^i を計算

(3) 以下の式に従って，i のシルエット係数 s^i を計算

$$s^i = \frac{c_{out}^i - c_{in}^i}{(\max(c_{in}^i, c_{out}^i))} \tag{49}$$

なお，C_{in} に属するエージェントが i しかいない場合，$c_{in}^i = 0$ とする．このようにして求めた S を用いて，以下の式を満たすか確認する．

$$S \geq S_{deact} \tag{50}$$

ここで，S_{deact} 平均シルエット係数の閾値である．一般に，シルエット係数は，複数のクラスターに対して計算し，最適なクラスター数を決めるために用いられる．しかし，6.2.3 項で詳細に説明するように，提案手法の AMTDS/ER は，エージェントを 2 つのグループに分けることができる．この 2 つのグループに分かれる前，つまりエージェントの役割分担がしっかり行われる前に停止を行わないようにするため，本来の使用用途とは異なるが式 (50) のようにシルエット係数を用いている．

5.3 AMTDS for energy saving under the requirement with communications (AMTDS/ERC)

subsection などの構成も含めて再考

この節では，従来手法の AMTDS/EDC と，AMTDS/ER を組み合わせた手法である．しかし，AMTDS/EDC にエネルギー節約行動や学習パラメータ K^i を導入しただけではうまくいかない．それの一番大きな理由は，AMTDS/ER では既知であったイベント発生確率を学習し，さらにそれを 4.3 で述べたように，交渉を行うため，その値はエージェントによって異なる．そのため，AMTDS/ER のように，式 (40) では実際の総イベント量との差が大きい．つまり，現在の環境全体のイベント量を予測することが困難になった．これを解決するため，以下で説明する変更・工夫を行った．なお，$p^i(v)$ の学習のため，エネルギー節約行動は T_{hp} ステップから開始する．

5.3.1 学習パラメータ C_{rate}^i の導入

まず，エージェント i のイベント発生確率の学習がどの程度ずれているのかを示す学習パラメータ C_{rate}^i を導入する．i は充電基地を出発時に，イベントの期待値の総和 S_{est}^i と実測値の総和 S_{real}^i を 0 にリセットする．時刻 t にノード $v \in V_R^i$ を訪問した際に，期待値 $E^i(L_t(v))$ の計算とイベント処理による実測値 $L_t(v)$ の取得を行い，それぞれを S_{est}^i と S_{real}^i に加算する．その後，充電基地に戻った際に，以下の式に従って C_{rate}^i を更新する．

$$C_{rate}^i \leftarrow \alpha_{rate} \times \frac{S_{real}^i}{S_{est}^i} + (1 - \alpha_{rate}) \times C_{rate}^i \tag{51}$$

5.3.2　K^i の更新方法の変更

C^i_{rate} の導入に伴い，K^i の更新方法も式 (42) から変更する．まず，以下の式から

$$K^i \leftarrow (1-\alpha_k) \times K^i + \alpha_k \times \frac{D_{req}}{E^i(D_t(V^i_R)) \times C^i_{rate}} \tag{52}$$

ここで，i の t における総イベント量の推定値 $E^i(D_t(V^i_R))$ は，V から壁などの障害物を表すノードを除いたノード集合 V_{access} と V^i_R を用いて，以下の式に従って求められる．

$$E^i(D_t(V^i_R)) = \sum_{v \in V^i_R} E^i(L_t(v)) \times \frac{|V_{access}|}{|V^i_R|} \tag{53}$$

5.3.3　未来のイベント量の予測の変更

AMTDS/ER では，式 (40) を用いて T ステップ後のイベント量を予測していたが，AMTDS/ERC ではイベント発生確率が実際とは異なるため，このままでは品質要求を満たせなくなってしまう可能性がある．そのため，未来のイベント量の予測を C^i_{rate} を用いて以下の式に変更した．

$$E^i(L_{t_c+T}(v)) = p(v) \times \{(t_c+T) - t^v_{vis}\} \times \frac{|V_{access}|}{|V^i_R|} \times C^i_{rate} \tag{54}$$

6　評価実験

この章では，提案手法である，AMTDS/ER や AMTDS/ERC と，エージェントがエネルギー節約行動をとらない手法である AMTDS[15] と AMTDS/EDC[20] や，エージェントがエネルギー節約行動を調整するパラメータを学習しない手法である AMTDS/ESC[5] を比較する実験を行い，提案手法の有効性を検証する．評価指標 $D(s)$ と $C(s)$ を調査することにより，提案手法が他の手法と比べて，エージェントが品質要求を満たしつつ，エネルギー消費を削減できることを実証する．また，$\forall i \in A$ の K^i の分布を分析し，その値に応じてエージェントを 2 つのグループに分けた．さらに，要求値 D_{req} を変化させたときのエージェントの対応を検証も行う．加えて，エージェントの停止を行う際に，停止方法による要求充足度の違いを検証し，それを基に，品質要求を満たしながら巡回エージェントの数を減らすことができることを実証する．実験に用いたパラメータ値を表 1~3 に示す．

6.1　実験環境

本研究では，図 3 に示すように，大きさが 101×101 の 2 次元グリッド環境を 2 つ用意した．図 3a の環境 $G_1 = (V_1, E_1)$ は，比較のため [5] の実験で用いられていた環境と同じである．この環境は，6 つの独立した部屋と，それらをつなぐ廊下が中央にある．黒線は壁 (エージェントが通過できない障害物) を表す．各ノード $v \in V_1$ は (x_v, y_v) の整数座標

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

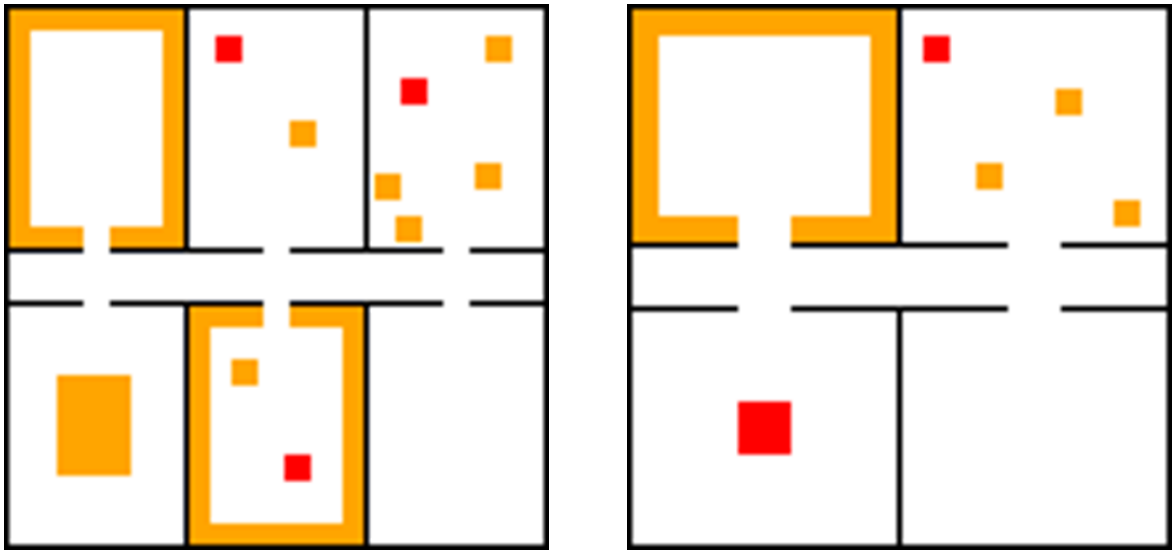
目標決定戦略	パラメーター	値
PGS	N_g	5
PI	N_i	5
BNPS	α	0.1
	d_{rad}	15
AMTDS	α	0.1
	ε	0.05
AMTDS/LD	β	0.1
AMTDS/EDC	N_{gmax}	100
	N_{cmax}	10
	T_c	0.05
	γ	10
	δ	0.5

表 3: エネルギー節約行動に関するパラメータ

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_{homing}	100
	k_{homing}	1/3
Pausing	S_{pause}	100
	T_{γ_p}	1,000
AMTDS/ER	α_k	0.1
AMTDS/ERC	T_{hp}	1,000,000
	α_{rate}	0.1

表 4: エージェントの停止に関するパラメータ

種類	パラメーター	値
Deactivation on Count	D_{int}	250,000
	N_{deact}	100
	K_{deact}	1.0
Deactivation on Time	D_{int}	250,000
	β_s	10
	α_1	0.1
	α_2	0.05
	S_{deact}	0.7



(a) 環境 1　　(b) 環境 2

図 3: 実験環境

で表され，$-50 \leq x_v, y_v \leq 50$ である．図 3 の色に従って，ノード $\forall v \in V_1$ のイベント発生確率 $p(v)$ を次のように設定した．

$$p(v) = \begin{cases} 10^{-3} & (v \text{ が赤色の領域内の場合}) \\ 10^{-4} & (v \text{ がオレンジ色の領域内の場合}) \\ 10^{-6} & (v \text{ が白色の領域内の場合}) \end{cases} \tag{55}$$

したがって，色が濃いほどイベント発生確率は高くなる．

図 3b の環境 $G_2 = (V_2, E_2)$ は 4 つの部屋があるが，各部屋の大きさは少し広くなっている．イベント発生確率も図 3a と同様に，色を使って式 (55) で指定する．この環境は図 3a と比べると，イベント発生確率の総和は少ないが，その分エージェントがエネルギー節約行動を行う必要がある．

エージェント数 $|A|$ は 20 であり，全てのエージェントの充電基地は中心 $(0, 0)$ に配置する．エージェントはバッテリを満タンにしてから充電基地を出発し，周囲を巡回して，バッテリ残量が 0 になる前に充電基地に戻るというサイクルを繰り返す．また，バッテリの性能を $(B^i_{max}, B^i_{drain}, k^i_{charge}) = (900, 1, 3)$ とする．したがって，0 から充電が完了するまでの時間は 2700 ステップとなり，最大活動時間は 3,600 ステップとなる．これにより，$D(s)$ と $C(s)$ のデータ収集間隔 $t_e - t_s$ を 3,600 とした．さらに，品質要求値を $D_{req} = 600$ とし，学習パラメータ K^i の初期値を 1.0 とした．

本実験では，1 回の試行ステップ数は評価実験によって異なるが，全ての評価実験の $D(s)$ と $C(s)$ の値は，独立した 50 回の試行による平均値である．

6.2 AMTDS/ER についての実験結果・考察

提案手法である AMTDS/ER とについての評価実験を，エネルギー節約行動を行わない AMTDS や，エネルギー節約行動を調整するパラメータを学習しない AMTDS/ESC と比較しながら行った．この節では，複数の条件下での実験結果を，評価指標である $D(s)$ や $C(s)$，さらには，必要に応じて他の指標とも比較していく．

6.2.1 実験 1: 性能評価

図 4，5 は，環境 1 で実験を行い，それぞれ 3,600 ステップごとのイベント残存時間の総和 $D(s)$ と，エージェントの総エネルギー消費量 $C(s)$ の時間推移を示したものである．なお，図 4 では，$D_{req} = 600$ なので，エージェントは $D(s) \leq 2,160,000$ $(= 3,600 D_{req})$ を保つことが要求条件であることに注意されたい．図 4 に示すように，全ての手法が品質要求を満たしていることがわかる．特に，従来手法である AMTDS と AMTDS/ESC は，$D(s)$ が要求値よりもかなり小さい値を維持している．

しかし，これは要求値以上の巡回による過剰なエネルギー消費を示唆しており，これを図 5 で確認することができる．この図から，提案手法 AMTDS/ER は，AMTDS に比べて $D(s)$ は約 40.3% 増加しているが，$C(s)$ を約 30.9% 削減することができた．なお，このデータは 2,000,000 ステップから 3,000,000 ステップの平均値である．AMTDS/ESC もエネルギー消費量を削減することはできたが，その削減効果は限られており，AMTDS/ER は AMTDS/ESC よりも，$D(s)$ を約 26.2% 増加させ，$C(s)$ を約 24.0% 削減することができた．これは，学習パラメータ K^i の導入により，各エージェントがエネルギー節約行動中の他のエージェントの巡回による効果を，ある程度予測できるようになったためであると考えられる．

6.2.2 実験 2: 異なる環境による性能評価

図 4，5 は，環境 2 で実験を行い，それぞれ 3,600 ステップごとの $D(s)$ と $C(s)$ の時間推移を示したものである．実験 2 でも，実験 1 と同様な結果がみられる．前述のように，環境 2 では，イベント発生確率の総和は環境 1 よりはるかに小さく，エージェントは実験 1 よりも多くのエネルギーを削減するために，より多くのエネルギー節約行動を行うことができるかを確認する．図 6 と図 7 より，エージェントは，要求条件を満たしながらエネルギー節約行動を徐々に増加させ，エネルギー消費を削減したことが分かり，期待通りの行動をとることができた．これにより，AMTDS/ER は AMTDS/ESC よりも，エネルギーを約半分しか消費していないことが分かる．

6.2.3 行動の解析

この項では，実験 1 と実験 2 でのエージェントの行動を解析し，提案手法である AMTDS/ER を定性的に評価する．まず，エージェントのエージェントのエネルギー節約行動の特徴，特に各エージェントの学習パラメータ K^i に一部影響される，*Pausing* による待機時間の

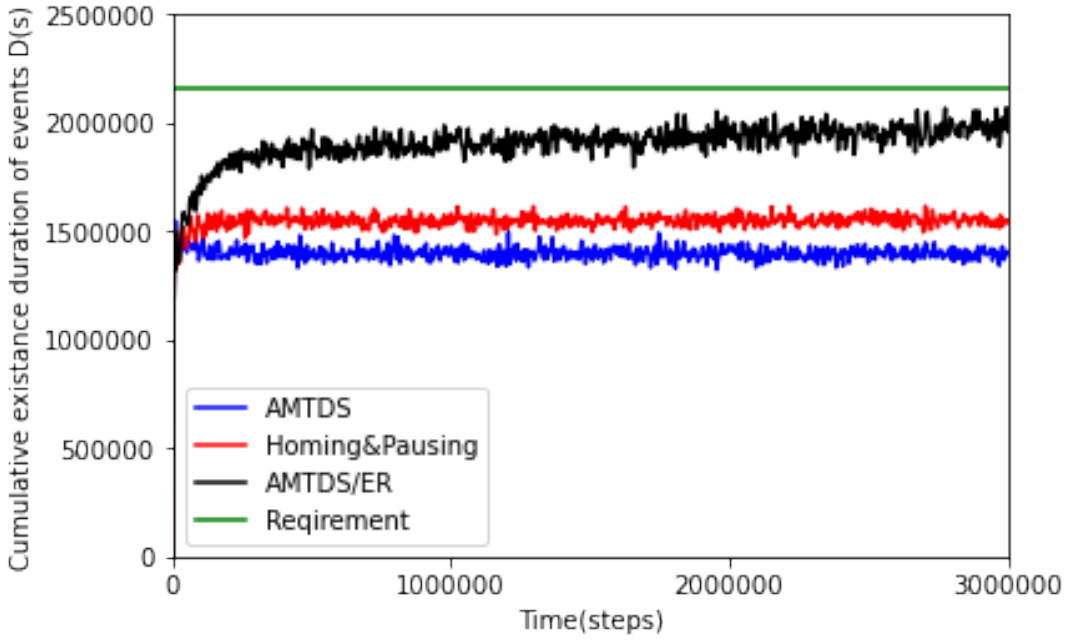


図 4: $D(s)$ の時間推移 (実験 1)

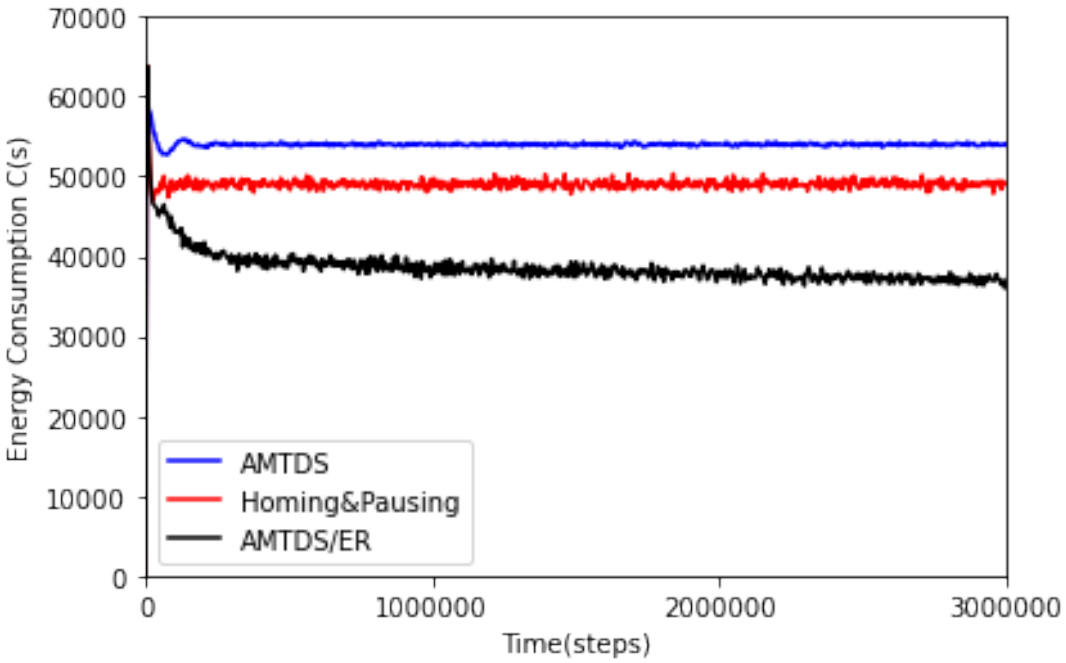


図 5: $C(s)$ の時間推移 (実験 1)

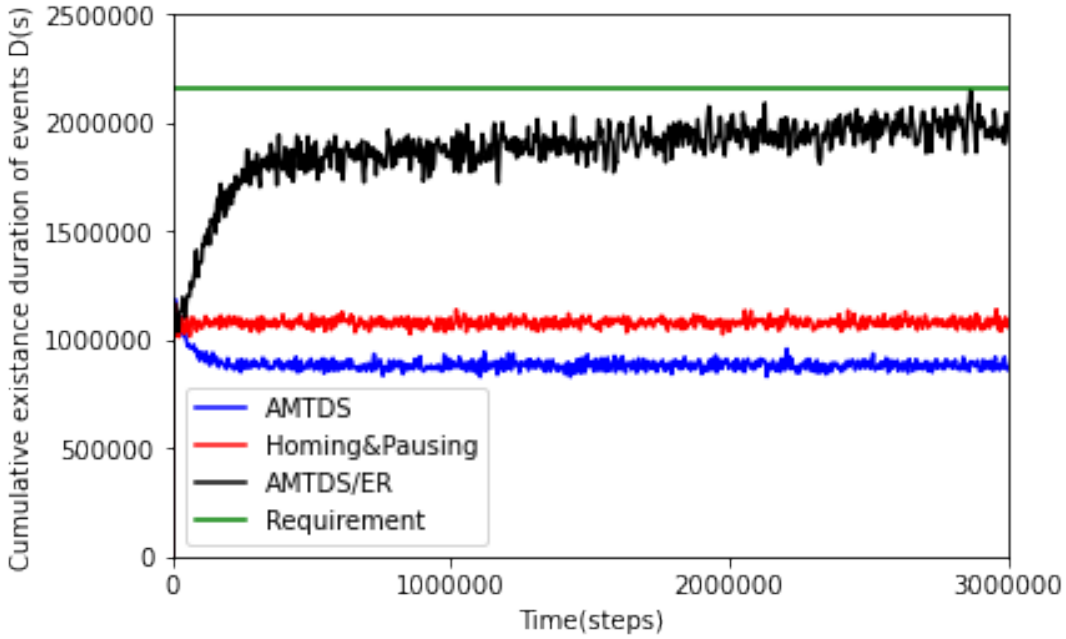


図 6: $D(s)$ の時間推移 (実験 2)

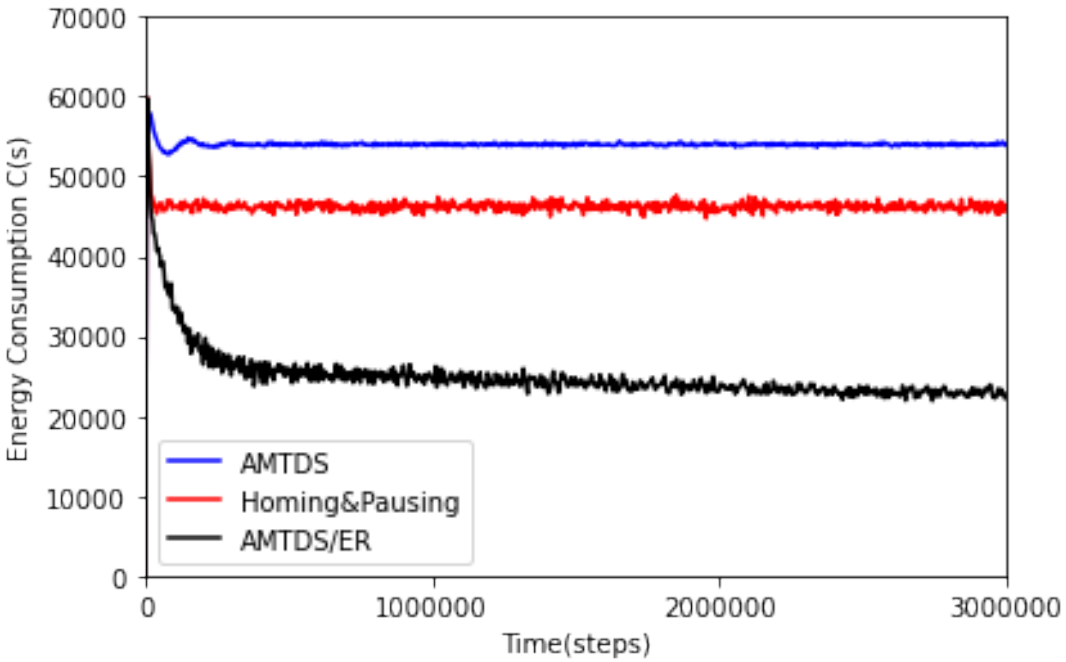


図 7: $C(s)$ の時間推移 (実験 2)

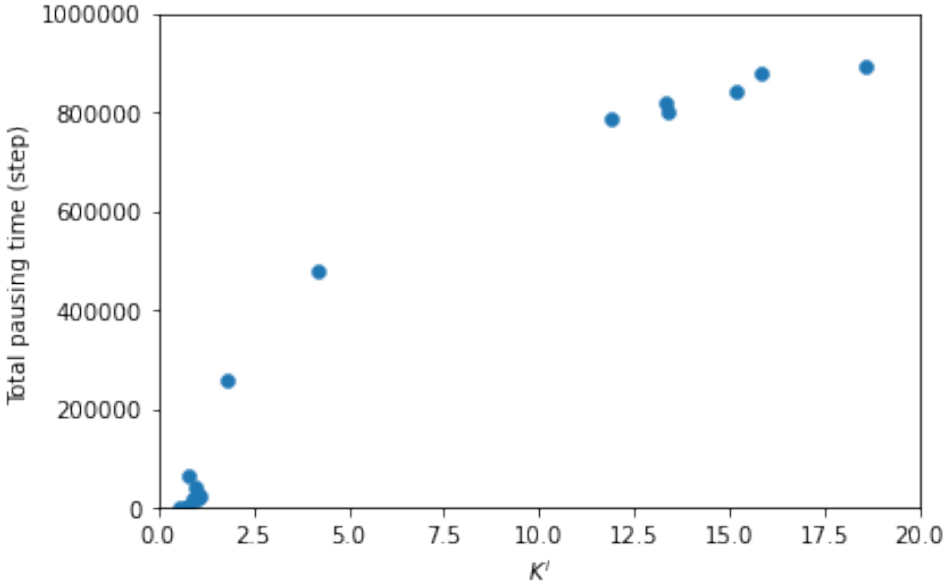


図 8: エージェントごとの K^i と待機時間の関係 (実験 1)

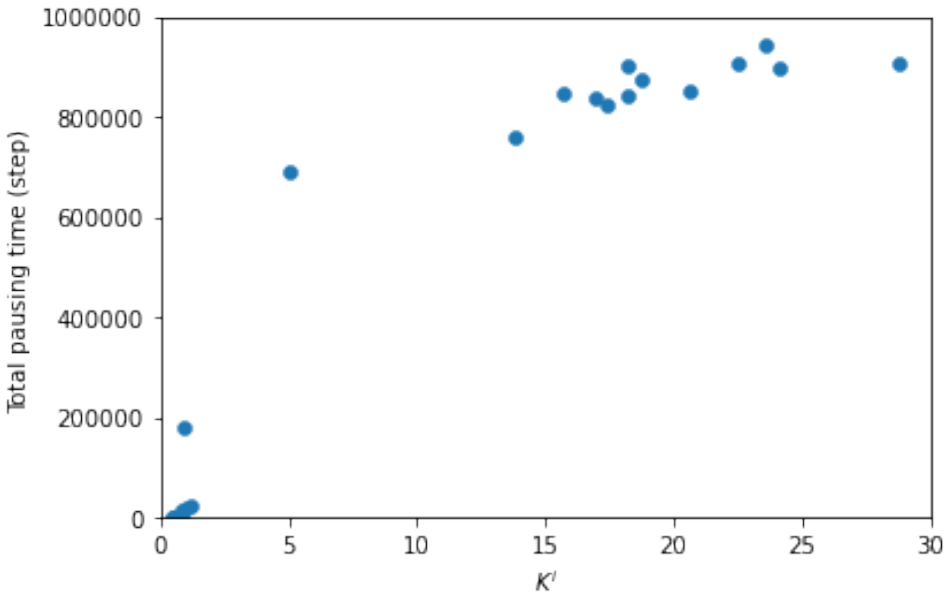
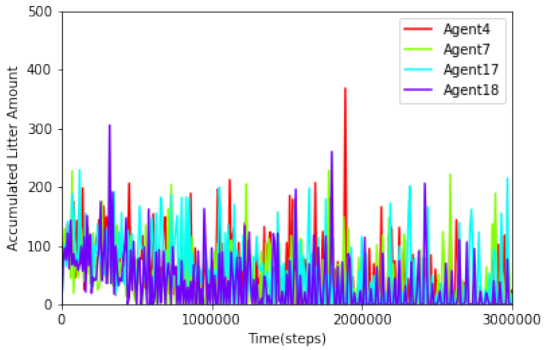
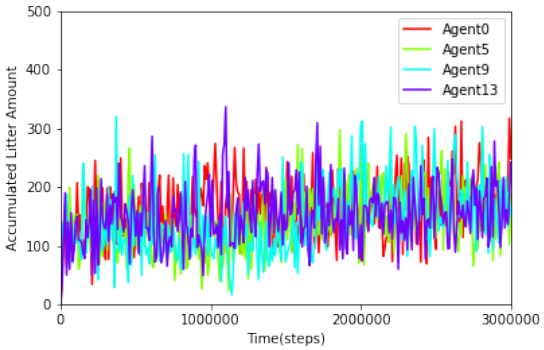


図 9: エージェントごとの K^i と待機時間の関係 (実験 2)

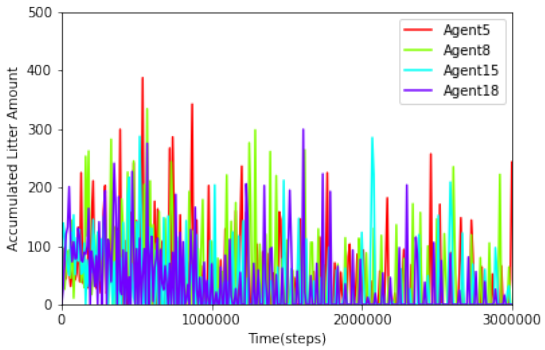


(a) K^i の上位 4 体

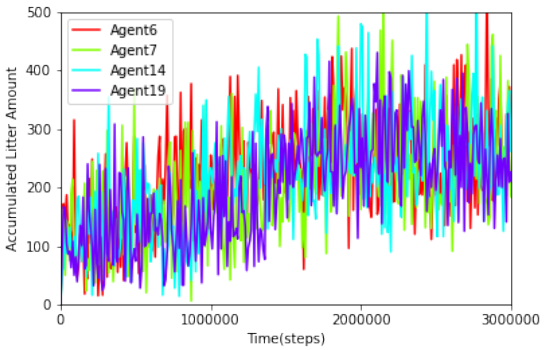


(b) K^i 下位 4 体

図 10: エージェントごとのイベント処理量の時間推移 (実験 1)



(a) K^i 上位 4 体



(b) K^i 下位 4 体

図 11: エージェントごとのイベント処理量の時間推移 (実験 2)

総和を解析する．図 8 と図 9 に，2,000,000 ステップから 3,000,000 ステップでの総待機時間を計算し，$\forall i \in A$ の K^i との関係をプロットした．なお，これらの散布図は，実験と実験 2 からランダムに選んだ 1 つの実験施行で得られたデータを用いており，特別な意図はない．

これらのグラフから，エージェントが K^i が比較的大きく (例えば $K^i \geq 10$)，総待機時間が大きいものと，K^i が小さく (例えば $K^i \leq 3$)，総待機時間が比較的小さいものとに大別できることが分かる．前者のグループを Energy save グループ (ES-group と呼ぶ)，後者のグループを Busy グループ (B-group と呼ぶ) と名付けることにする．まず，$2,000,000 \sim 3,000,000$ の間の長さは 1,000,000 であるので，800,000 ステップ以上 *Pausing* を行った ES-group のエージェントはほとんど環境を巡回していないことが分かる．200,000 ステップ以下は *Pausing* を行っていないことになるが，$k_{charge} = 3$ より，このエージェントの実際の巡回は 50,000 ステップ以下である．

一方，B-group のほとんどのエージェントは，*Pausing* を行わなかった．なお，*Homing* は残量に関係なく充電基地に戻るだけであり，充電時間を短縮することができる．そのため，*Homing* は直接的にはエネルギー節約に貢献しなかった．B-group のエージェントは，

要求条件を満たすために必要な巡回をほとんど行ったのに対し，ES-group のエージェントは待機時間を増やしており，このような分化が個人学習により発生したことが分かる．

ここで図 10a と図 11a は，ES-group に属し K^i が大きい上位 4 体を，図 10b と図 11b は，B-group に属す K^i が小さい下位 4 体の行動である．これらの図から，どちらの環境でも，Pausing を時間と共に多くとるようになり，散発的にのみ行動してイベントを処理するようになったグループと，充電時間を除いて継続的に行動し，イベント処理数を時間と共に増加させたグループの行動の違いが分かる．

なお，実験 1 と比べ実験 2 のほうが，推移の分散が大きい．これは環境の大きさに比べ，B-group のエージェント数が相対的に小さいことによるものと考えられる．実際に，図 8 と図 9 における B-group の数の最頻値 (mode) を調べると，実験 1 では 12 体，実験 2 では 7 体のエージェントがこれに対応している．相対的にエージェント数が多い実験 2 では，多くのエージェントが ES-group となり，多くのエネルギー抑制行動がとられたことを反映している．なお，上記の最頻値は，エージェント間の分業の結果により，1 体ほど前後することがある．以上より，提案手法の AMTDS/ER は，環境と品質要求を考慮した 2 つのグループの構成比に分かれることができる手法であるといえる．

6.2.4　実験 3: D_{req} の変化による性能の変化

この項では，要求値 D_{req} が途中で変化した際に，提案手法は対応できるのかを検証する．図 12，13 は環境 2 で実験を行い，それぞれ 3,600 ステップごとの $D(s)$ と $C(s)$ の時間推移を示したものである．ここで，$0 \sim 2,000,000$ ステップの期間を P1，$4,000,000 \sim 6,000,000$ ステップの期間を P2，$4,000,000 \sim 6,000,000$ ステップの期間を P3 として，D_{req} は P1 と P3 では 1,000 であり，P2 では 600 と設定している．つまり，エージェントは P1 と P3 では $D(s) \leq 3,600,000 (= 3600 D_{req})$ を保ち，P2 では $D(s) \leq 2,160,000 (= 3600 D_{req})$ を保つことが要求条件であることに注意されたい．図 12 より，提案手法である AMTDS/ER は，途中で品質要求が変化してもしっかり対応できていることがわかる．従来手法である AMTDS/ESC も D_{req} が変化した際に $D(s)$ を変化させているが，その変化量は D_{req} の変化量と比べると小さい．このことから，エネルギー節約行動を調整する学習パラメータ K^i の導入により，品質要求の変化にも対応できたといえる．

次に，D_{req} が変化したときの各エージェントの学習パラメータ K^i と，$Pausing$ による待機時間の関係をまとめた散布図が図 14 ~16 である．なお，それぞれは 2,000,000 ステップ，4,000,000 ステップ，6,000,000 ステップ時点での K^i と，直前 1,000,000 ステップにおける待機時間の総和をプロットしてある．これらの図より，品質要求によってプロットの散らばりぐらいが異なることが分かる．D_{req} が小さくなると厳しくなった品質要求を満たすために，K^i が小さくなり，待機時間が短くなって巡回を多く行うようになったエージェントが存在し，D_{req} が大きくなると品質要求に対して巡回が過剰であるため，K^i が大きくなり，待機時間が長くなったエージェントが存在することが分かる．また，それぞれの図での B-goup と ES-group のエージェント数は，図 14 では 6 体と 14 体，図 15 では 11 体と 9 体，図 16 では 6 体と 14 体であった．

さらに，図 17~19 に，P1~P3 の期間で B-group と ES-group のうち，属するグループを変更したエージェントのイベント処理数の時間推移を示す．なお，図 17 は P1~P2 と

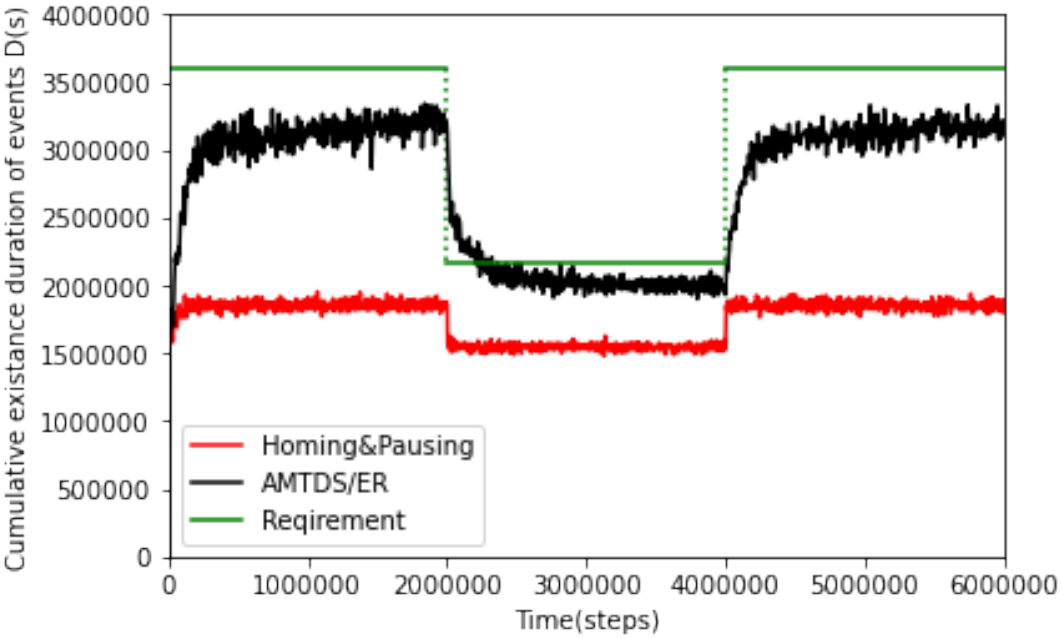


図 12: $D(s)$ の時間推移 (実験 3)

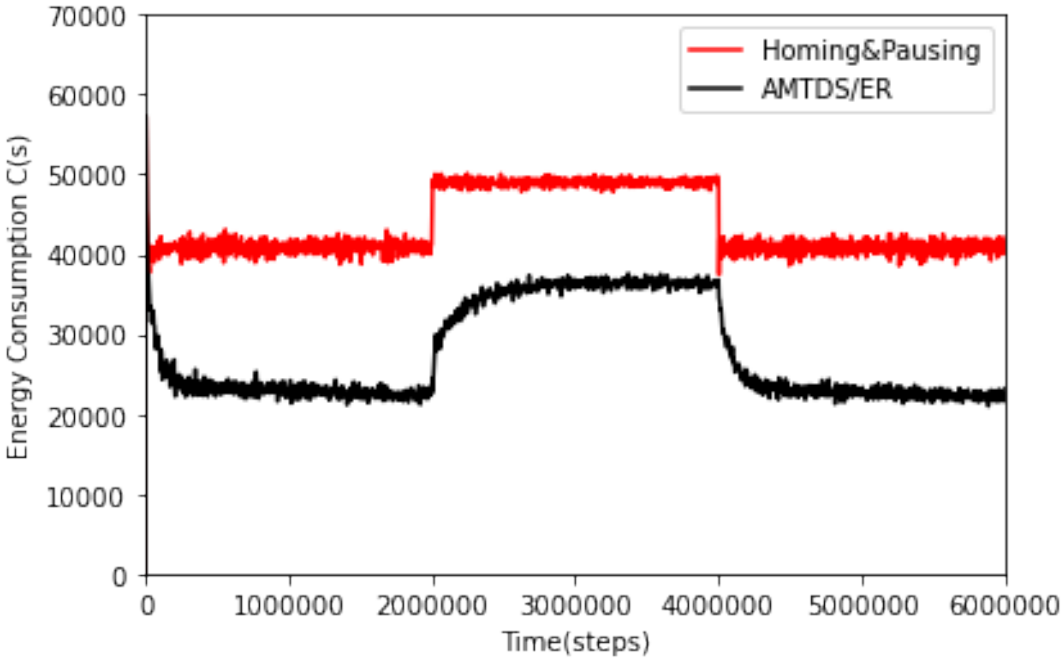


図 13: $C(s)$ の時間推移 (実験 3)

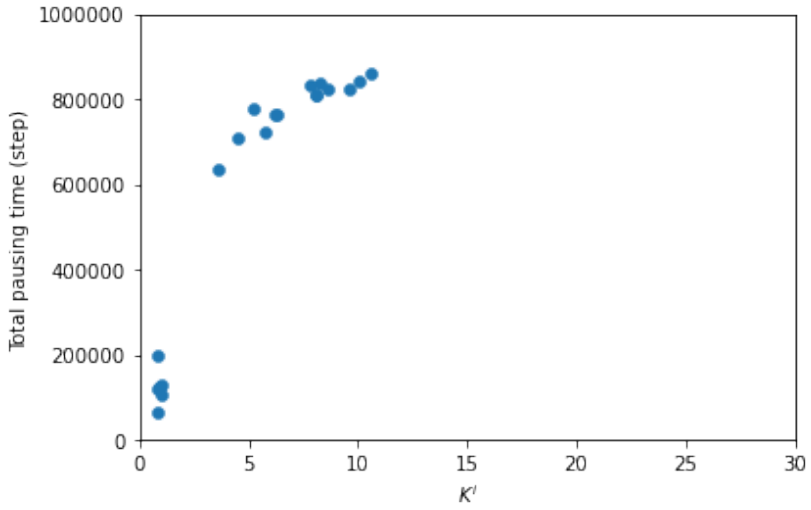


図 14: エージェントごとの K^i と待機時間の関係 (実験 3：2,000,000)

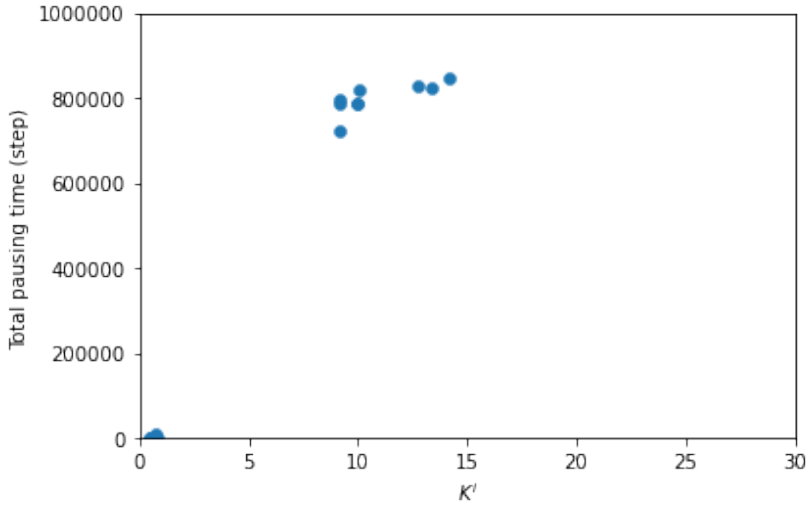


図 15: エージェントごとの K^i と待機時間の関係 (実験 3：4,000,000)

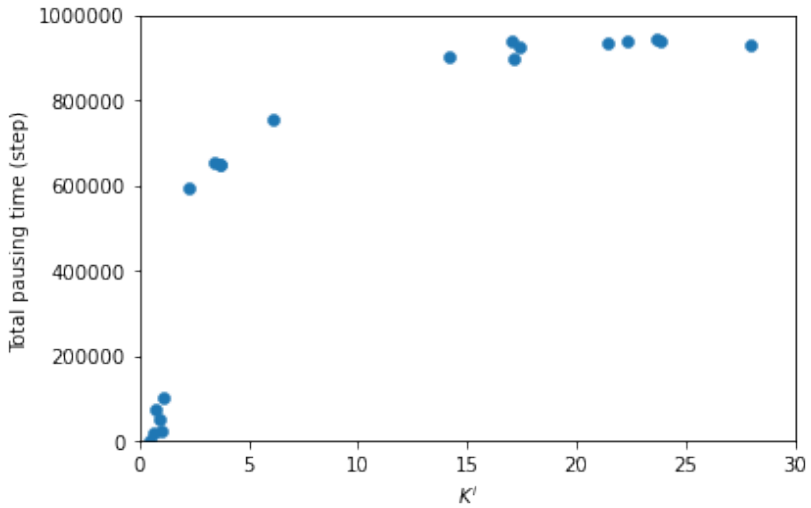


図 16: エージェントごとの K^i と待機時間の関係 (実験 3：6,000,000)

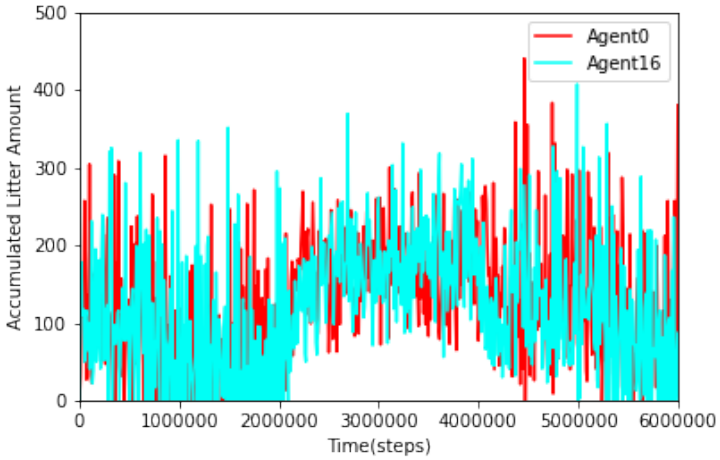


図 17: エージェントのイベント処理数の時間推移 (全期間ともグループ変更)

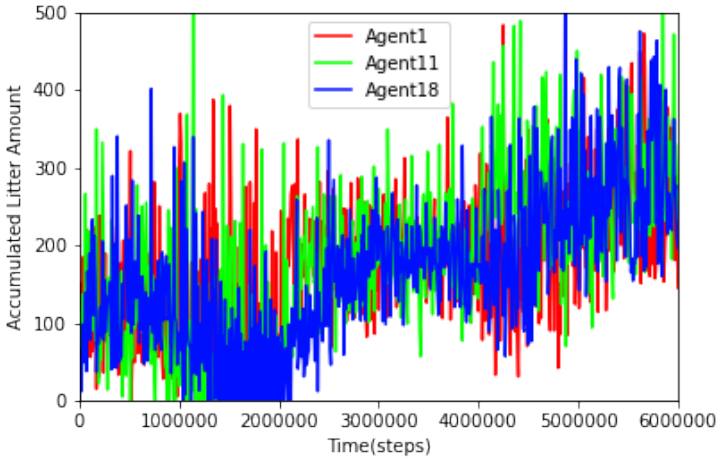


図 18: エージェントのイベント処理数の時間推移 (P1∼P2 でグループ変更)

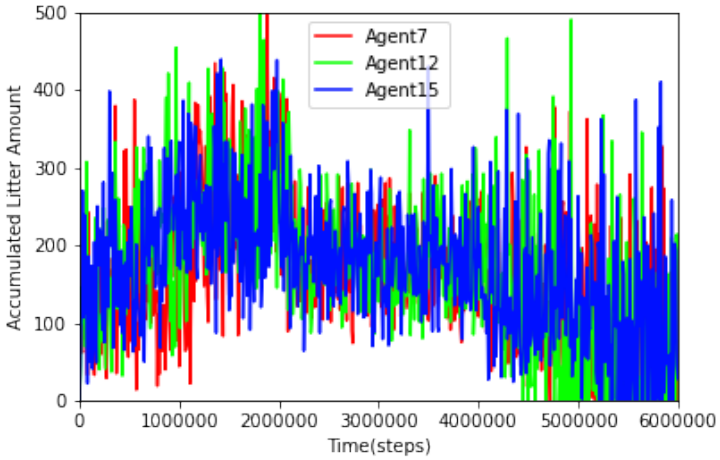


図 19: エージェントのイベント処理数の時間推移 (P2∼P3 でグループ変更)

P2∼P3 のどちらでも属するグループを変更したエージェントであり，図 18 は P1∼P2 で，図 19 は P1∼P2 で属するグループを変更したエージェントを示している．まず，図 17 にプロットしたエージェントは，P1∼P2 では ES-group から B-group に，P2∼P3 では B-group から ES-group に変更した．これは，イベント処理数の時間推移からも確認することができる．P1 と P3 では ES-group に属しているため，図 10a のように，時間とともに待機時間が増え，イベント処理数が減少しているが，P2 では B-group に属しているため，図 10b のように，ES-group の分もイベント処理している．次に，図 18 にプロットしたエージェントは，P1∼P2 で ES-group から B-group に変更した．実際に，図??では，P1 はイベント処理数が減少しているが，P2 と P3 では ES-group の分もイベントを処理するようになり，イベント処理数を増加させている．さらに，図 19 にプロットしたエージェントは，P2∼P3 で B-group から ES-group ひ変更した．これまでと同様に，図 19 より，P1 ではイベント処理数を増加させ，P2 では品質要求が厳しくなったことによって，B-group に属するエージェントが増えたため，イベント処理数は P1 よりは少ないが，一定量を維持し，P3 ではイベント処理数が減少していることを確認することができる．

以上より，提案手法である AMTDS/ER は，品質要求が世中で変化しても対応でき，それに応じて 2 つのグループの構成比を変化させることができる手法であるといえる．

6.2.5　実験 4: エージェント数減少による性能の変化

この節では，途中でエージェントが複数体停止したとき，提案手法の AMTDS/ER が品質要求を満たさせるかを検証する．ここでは，2,000,000 ステップで 5 体停止させ，ランダムに 5 体停止と K^i の降順に 5 体停止の 2 つの停止手法を比較する．前者は巡回中のエージェントの故障を想定し，後者はエージェント数を減らしたいときに *Pausing* による待機を多めにしているエージェントから停止する状況を想定している．

図 20，21 は，環境 1 でランダムに 5 体停止を行い，それぞれ 3,600 ステップごとの $D(s)$ と $C(s)$ を示したものである．図 20 より，2,000,000 ステップでランダムに 5 体停止したため，$D(s)$ の一時的な悪化が生じてしまい，品質要求を満たすことができなくなってしまっている．しかし，その後徐々に品質要求を満たせるようになった．

図 22a は 2,000,000 ステップ時点での K^i と，1,000,000∼2,000,000 ステップでの総待機時間をプロットしたものであり，この時に停止するエージェントを赤でプロットしている．この図より，停止するエージェントは，待機時間が少ないものもあれば，多いものもあることが分かる．つまり，停止エージェントは B-group や ES-group に偏らずに選択された．次に，図 22b は 4,000,000 ステップ時点での K^i と，3,000,000∼4,000,000 ステップでの総待機時間をプロットしたものである．図 22a，22b より，品質要求を満たすために待機時間が短く，巡回中心であったエージェントが停止し，このままでは品質要求をみたせないため，その時は待機時間が長かったエージェントも K^i が小さくなり，待機時間も短くなったことが分かる．

また，図 23 はエージェントの K^i の時間推移を示したものである．図 23a はランダム停止後に，品質要求を満すために ES-group から B-goup になったエージェントであり，図 23b はランダム停止後も ES-group のままであったエージェントである．図 23a より，このエージェントは停止後に品質要求が満たさなくなったと判断し，元々は K^i が大きかっ

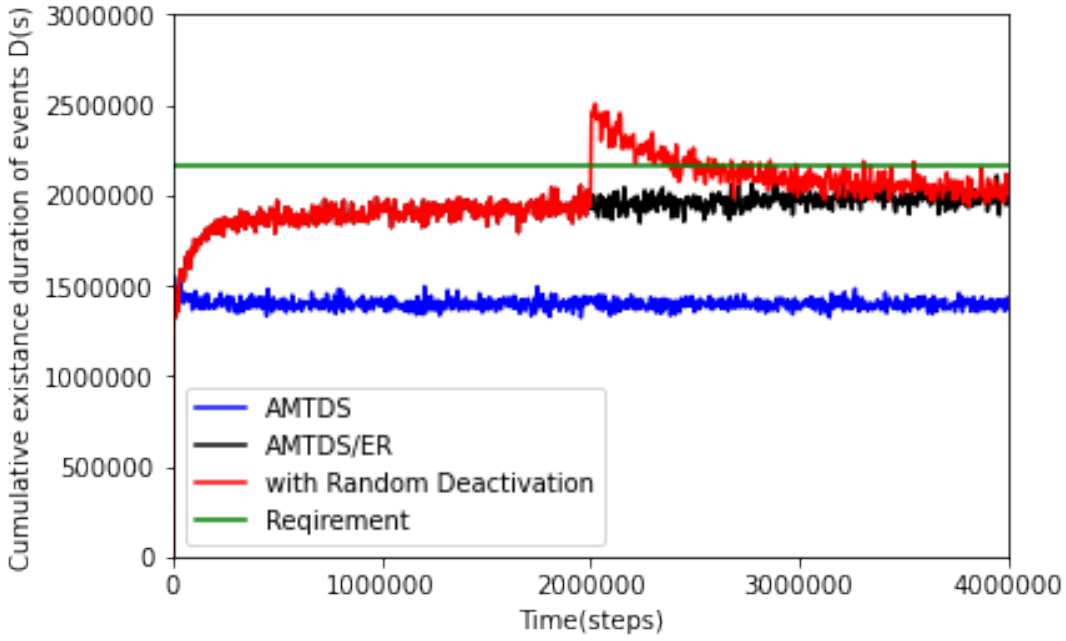


図 20: ランダム停止における $D(s)$ の時間推移 (実験 4)

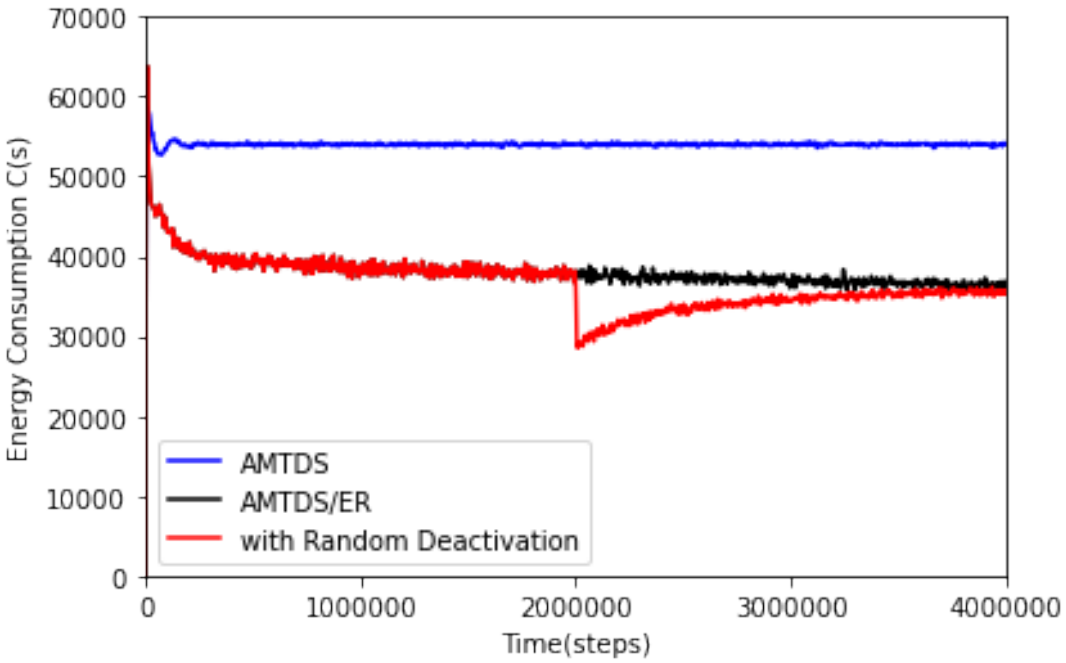
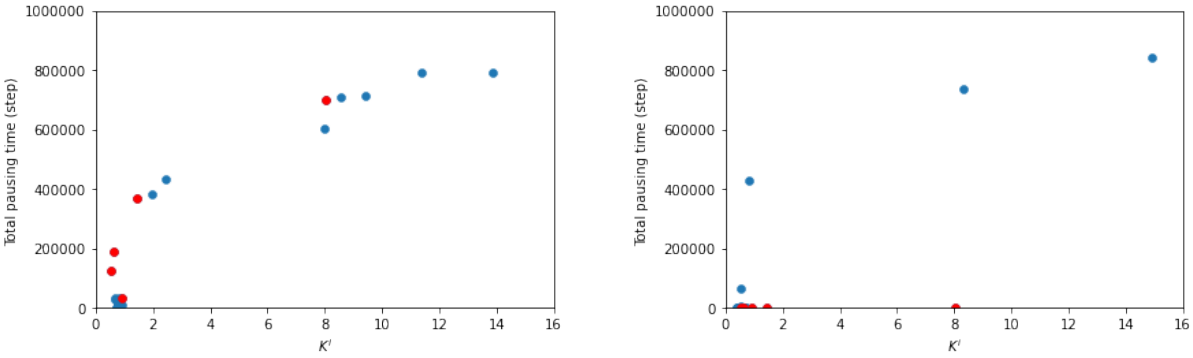
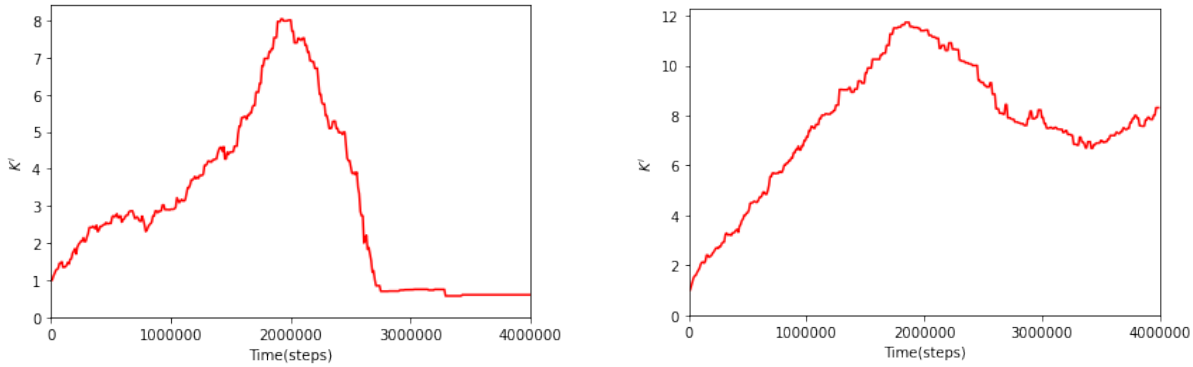


図 21: ランダム停止における $C(s)$ の時間推移 (実験 4)



(a) 停止時　　(b) 停止後

図 22: エージェントごとの K^i と待機時間の関係 (ランダム停止)



(a) 停止後に B-group になったエージェント　　(b) 停止後も ES-group であったエージェント

図 23: エージェントの K^i の時間推移 (ランダム停止)

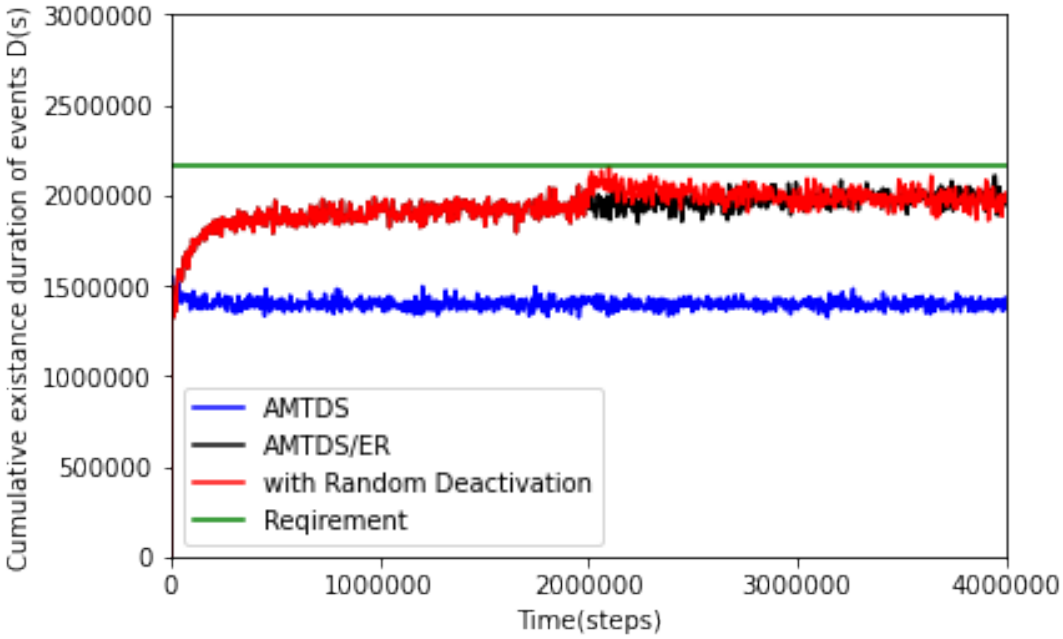


図 24: 降順停止における $D(s)$ の時間推移 (実験 4)

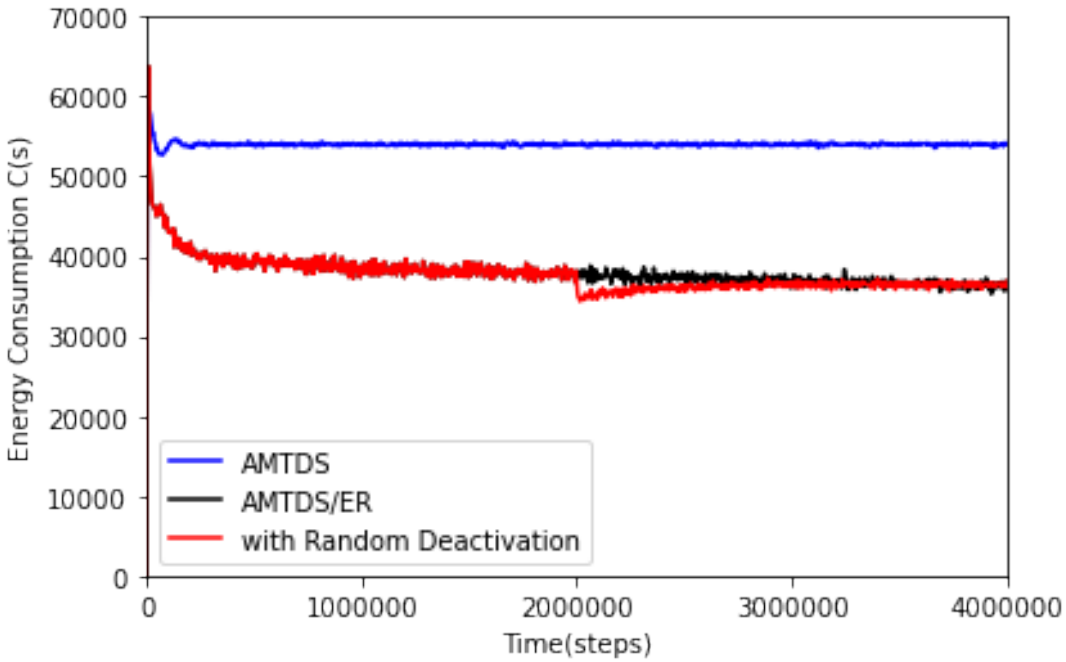
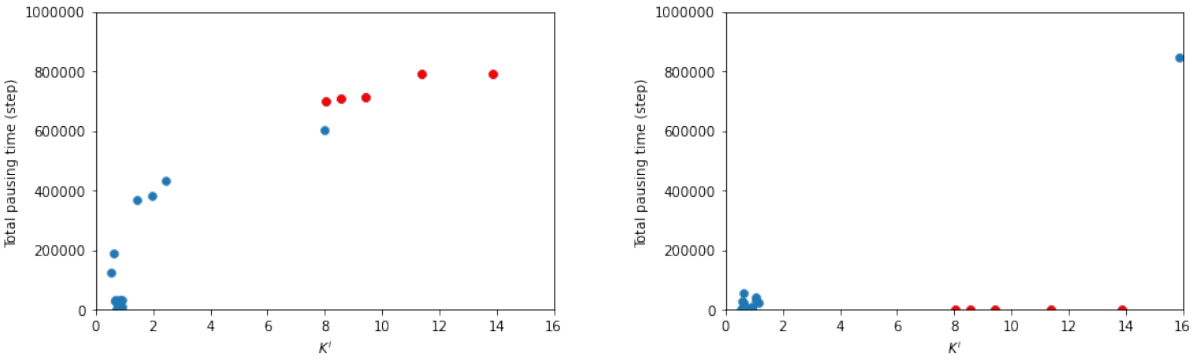
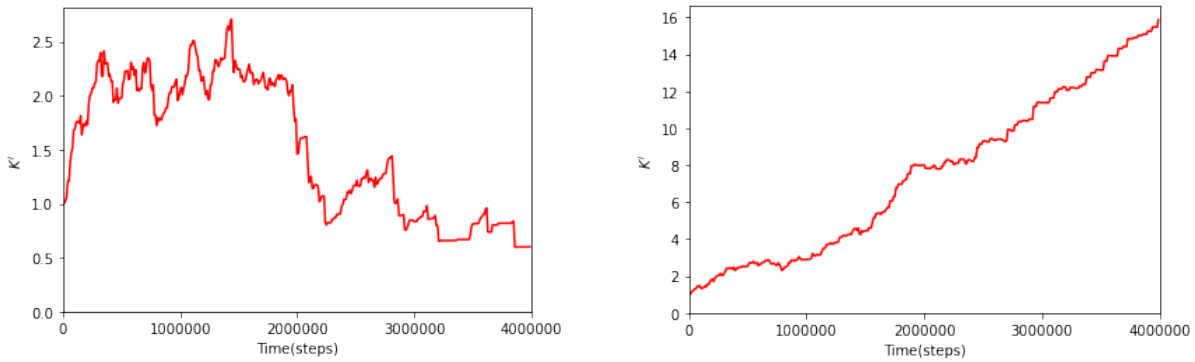


図 25: 降順停止における $C(s)$ の時間推移 (実験 4)



(a) 停止時　　(b) 停止後

図 26: エージェントごとの K^i と待機時間の関係 (降順停止)



(a) 停止後も B-group であったエージェント　　(b) 停止後も ES-group であったエージェント

図 27: エージェントの K^i の時間推移 (降順停止)

たのに，停止後に急激に減少し，B-goup になり，巡回中心になったことが分かる．一方，図 23b より，このエージェントも停止後は品質要求が満たさなくなったと判断し，K^i を減少させたが，他のエージェントも巡回を多くしたため品質要求が満たさせるようになり，エネルギー消費抑制の観点から，ES-group のままであったということが分かる．

以上より，ランダムにエージェントを停止した際，$D(s)$ の一時的な悪化は起きてしまうものの，その後にエージェントが品質要求を満たさせていないと判断し，K^i を変化させ，B-group と ES-group の比率を変化させることにより，品質要求を満たせるようになった．つまり，提案手法の AMTDS/ER は，巡回中にエージェントが故障し停止してしまっても，学習パラメータ K^i を変化させることにより，対応できるということが分かった．

図 24，25 は，環境 1 で K^i の降順に 5 体停止を行い，それぞれ 3,600 ステップごとの $D(s)$ と $C(s)$ を示したものである．図 20 と図 24 を比較すると，前述したランダム停止と比べて，停止時の $D(s)$ の一時的な悪化がかなり緩和されていることが分かる．具体的には，ランダム停止時のピークから降順停止のピークは，約 13.9%削減することができた．ランダム停止では品質要求を大幅に超えてしまっていたが，降順停止では停止による悪化をかなり抑え，品質要求も満たせたままでいることができた．

次に，図 26a は 2,000,000 ステップ時点での K^i と，1,000,000~2,000,000 ステップでの総待機時間をプロットしたものであり，この時に停止するエージェントは赤でプロットしている．この図より，停止するエージェントは，K^i が大きいものから 5 体選べていることが分かる．また，図 26b は 4,000,000 ステップ時点での K^i と，3,000,000~4,000,000 ステップでの総待機時間をプロットしたものである．図 26a，26b より，K^i の大きなエージェントが停止したため，停止後に ES-group から B-group に変更したエージェントは少なかった．これは，元々待機時間が多くイベント処理をあまり行っていなかったエージェントが停止したため，それをカバーするのは容易であり，ランダム停止に比べて，待機時間を減らして巡回を多くやる必要があまりなかったからだと考えられる．

また，図 27 はエージェントの K^i の時間推移を示したものである．降順停止では，前述したランダム停止と比べて，停止後に大きな役割の変更はなかった．しかし，同じ役割でも K^i の変化から，行動の変化を読み取ることができる．図 27a は，停止前後でも B-group であったエージェントであり，図 27b は，停止前後でも ES-group であったエージェントである．図 27a より，このエージェントは停止前も K^i が小さく，B-group であったが，2,000,000 ステップで降順停止後はさらに K^i を小さくしている．これにより，停止前は待機時間をある程度とっていたが，停止後はほとんど待機せずに巡回するようになった．一方，図 27b より，このエージェントは停止前から K^i が大きく，停止後も K^i が増加している．しかし，2,000,000 ステップでの停止直後はほとんど増加していない．つまり，この期間では品質要求を満たすために，停止していないエージェント達は図 27a のように K^i を減少させ，停止したエージェントの分も巡回を行おうとする．それにより品質要求は十分に満たされ，これ以上待機時間を減らして巡回しなくても良くなり，エネルギー抑制の観点から，K^i を減らさずに増加を続け，待機時間を多くするということを判断していると考えられる．

以上より，エージェント数を減らしたいときには，提案手法で導入した学習パラメータ K^i の降順に停止することにより，$D(s)$ の一時的な悪化を最小限に留めることができるということが分かった．

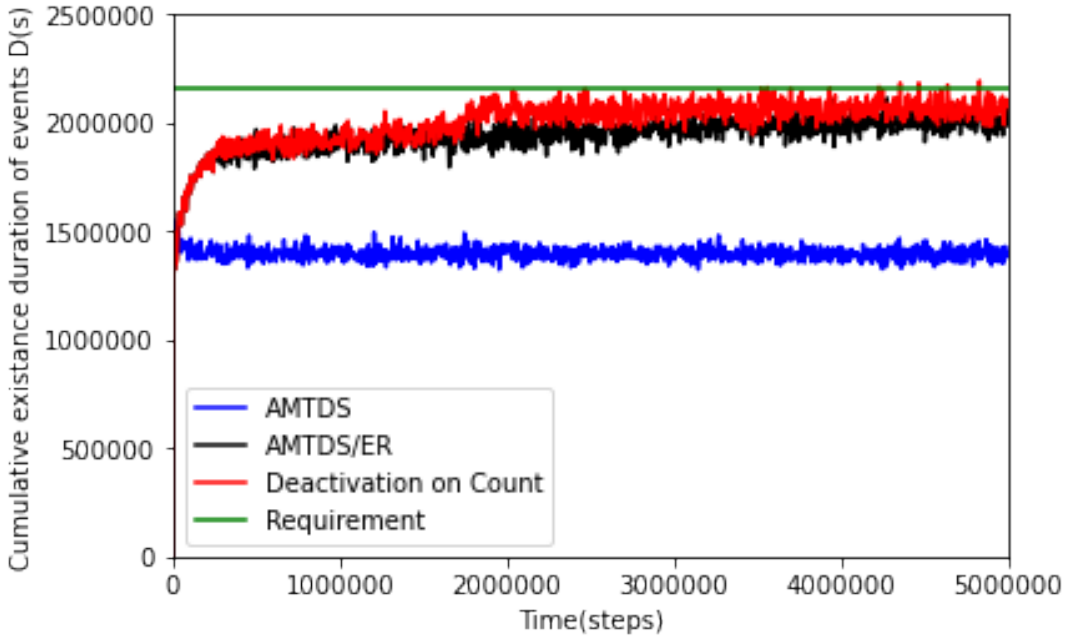


図 28: $D(s)$ の時間推移 (実験 5)

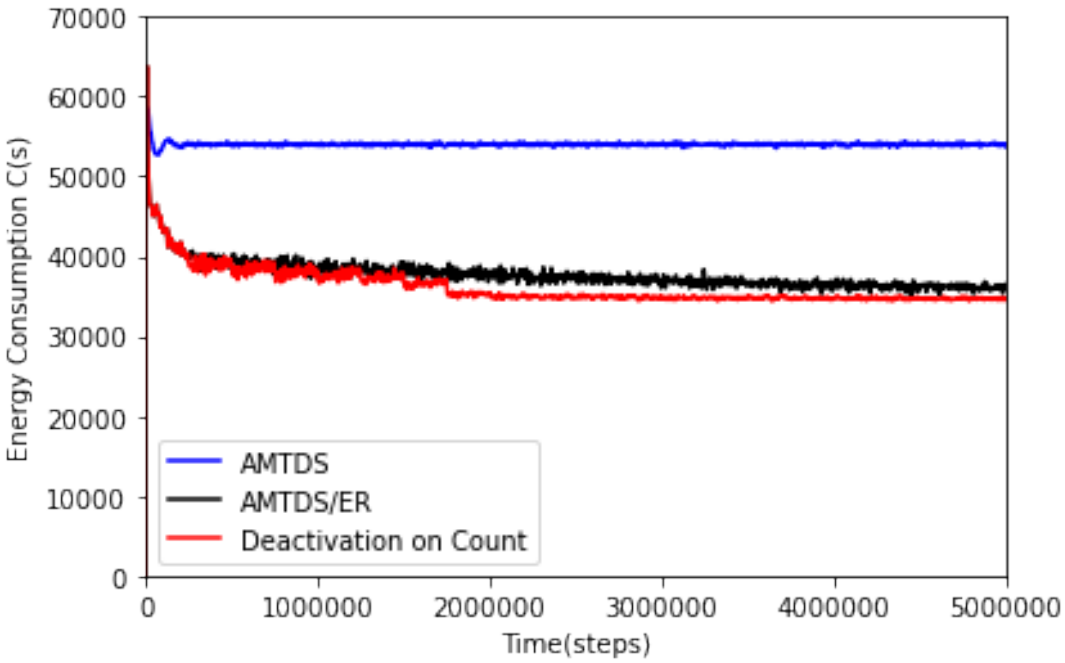


図 29: $C(s)$ の時間推移 (実験 5)

6.3 不要なエージェントの停止

この節では，5.2 節で述べた Deactivation on Count と Deactivation on Time それぞれの手法による，品質要求を満たす上で不要なエージェントを停止する実験を停止なしの AMTDS/ER と比較する．

6.3.1 実験 5: Deactivation on Count による不要なエージェントの停止

この項では，6.2.3 項での分析に基づき，5.2.1 項で述べた Deactivation on Count による不要なエージェントの停止手法を評価するために，実験 1 と同じ設定で実験を行った．すなわち，品質要求を満たしながら，巡回エージェントの数を減らし，エネルギー消費を抑えられるかを検証する．表 4 に示すように，$D_{int} = 250,000$ ごとにエージェントを停止できるかを確認した．

図 28，29 はそれぞれ $D(s)$ と $C(s)$ の 5,000,000 ステップまでの時間推移を示しており，図内の Deactivation on Count というラベルが Deactivation on Count による停止手法を用いた AMTDS/ER を示している．この実験では，停止したエージェントの数は 6 から 8 の間であり，ほとんどが 7 であった．ここで，$D(s)$ と $C(s)$ の値は小さい方が良いことに再度注意されたい．

図 28 より，Deactivation on Count を用いた AMTDS/ER は，停止手法を用いない AMTDS/ER と比較して，差は小さいが，品質要求を満たしつつ $D(s)$ をより大きいことが分かる．これは，AMTDS/ER が既に $D(s)$ の値を要求品質付近に維持できるためである．また，それにより図 29 より，エネルギー消費が若干減少した．しかし，停止手法ありと停止手法なしとの大きな違いは，停止したエージェントが巡回のタスクを他のエージェントに完全に移譲できている点である．さらに，$D(s)$ と $C(s)$ の性能差が大きくないにもかかわらず，以下のようなメリットがあることを強調したい．

(1) 停止したエージェントを他の環境で利用できる

(2) それらのエージェントを待機させ，巡回エージェントが故障時に利用できる

(3) 定期点検や交代運用を行ってシステムを延命できる

6.3.2 実験 5 における停止エージェントと非停止エージェントの行動の解析

この項では，実験 5 における K^i の推移を調べることで，Deactivation on Count による停止エージェントと非停止エージェントの行動を解析する．図 30 に，実験 5 からランダムに選んだ，独立した 1 つの実験における K^i と i の *Pausing* による待機時間の関係をプロットする．ここで，K^i は 5,000,000 ステップでの値であり，待機時間は実験の最後の 1,000,000 ステップでの待機時間の総和である．したがって，停止したエージェント i の K^i は停止したときの値であり，待機時間は 0 である．図 30 での 7 つの赤いプロットが停止エージェントに対応する．この図より，停止エージェントは K^i が比較的大きく，非停止エージェントは非常に短い待機時間で巡回を続けていることが分かる．このことから，この環境では 13 体のエージェントで要求品質を満たすことができたと考えられる．

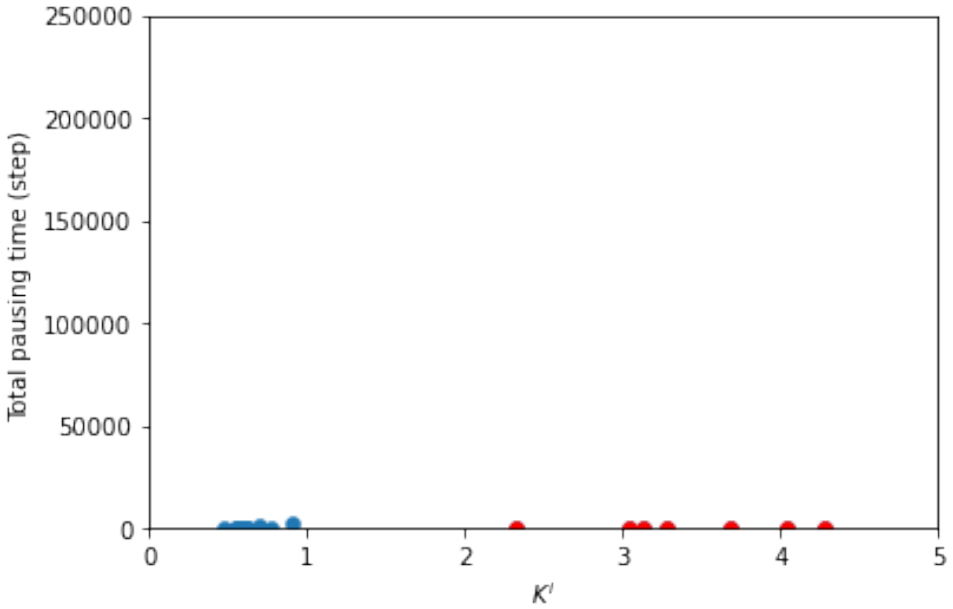
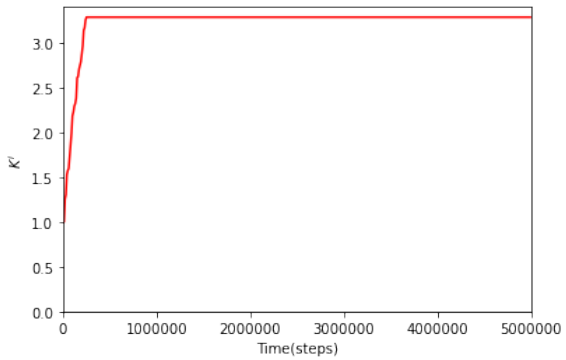
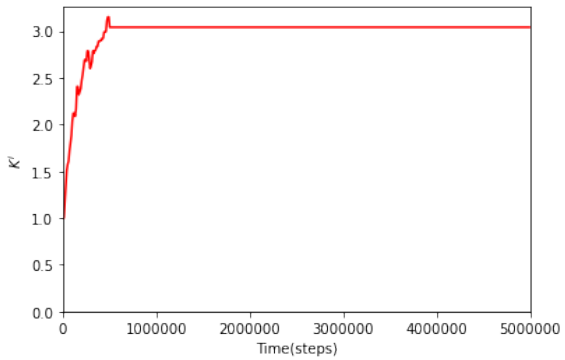


図 30: エージェントごとの K^i と待機時間の関係 (実験 5)

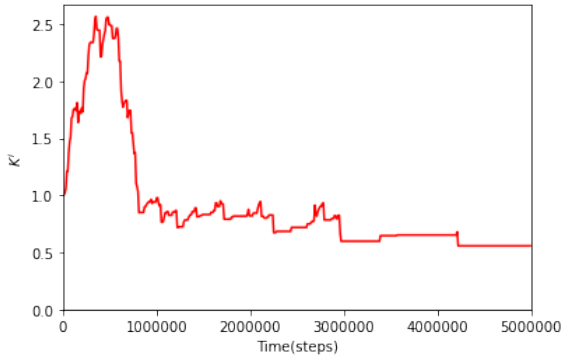


(a) $D = 250,000$ ステップ

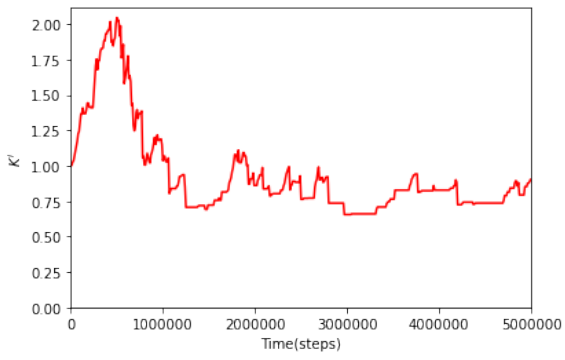


(b) $D = 500,000$ ステップ

図 31: D ステップで停止したエージェントの K^i の時間推移 (実験 5)



(a) 例 1



(b) 例 2

図 32: 非停止エージェントの K^i の時間推移 (実験 5)

図 31 と図 32 は，250,000 ステップと 500,000 ステップで停止したエージェント (図 31) と，非停止エージェント (図 32) の K^i の時間推移をプロットしたものである．図 31 より，これらのエージェントの K^i は急激に増加し，待機時間がその時点で最大であったため，停止の対象に選ばれたことが分かる．停止後は K^i は変化しない．一方，非停止エージェントは一時的に K^i を増加させたが，その後，一部のエージェントが停止したため，K^i が減少し，比較的小さい値を維持した．つまり，非停止エージェントは品質要求を満たすために，停止エージェントの分も巡回しなければならなくなった．

実験の結果，AMTDS/ER で導入した学習パラメータ K^i により，この値に基づき，さらに，関数 PLength が呼び出された回数を比較してすることによって，巡回エージェントの数を削減することができた．

6.3.3　実験 6: Deactivation on Time による不要なエージェントの停止

この項では，6.2.3 項での分析に基づき，5.2.2 で述べた Deactivation on Time による不要なエージェントの停止手法を評価するために，実験 1 と同じ設定で実験を行った．すなわち，品質要求を満たしながら，巡回エージェントの数を減らし，エネルギー消費を抑えられるかを検証する．表 4 に示すように，$D_{int} = 250,000$ ごとにエージェントを停止できるかを確認した．

図 33，34 はそれぞれ $D(s)$，$C(s)$ の 5,000,000 ステップまでの時間推移を示しており，図内の Deactivation on Time というラベルが Deactivation on Time による停止手法を用いた AMTDS/ER を示している．この実験では，停止したエージェントの数は 5 から 7 の間であり，ほとんどが 6 であった．ここで，$D(s)$ と $C(s)$ の値は小さい方がよいことに再度注意されたい．

図 33 より，Deactivation on Time を用いた AMTDS/ER は，停止手法を用いない AMTDS/ER とほとんど差がなかった．それに伴い，図 34 でも差が小さいのが分かる．つまり，システム全体の巡回性能をほとんど変化させることなく，エージェントの数を減らすことができた．ここでも，6.3.1 項で述べた，停止手法によるメリットを再度強調したい．

6.3.4　実験 6 における停止エージェントと非停止エージェントの行動の解析

ここで，Time の方がよいということ、30 台の追加実験について述べる

この項では，Deactivation on Time による停止エージェントと非停止エージェントの行動を解析し，実験 5 の Deactivation on Count と比較する．図 35 に，実験 6 からランダムに選んだ，独立した 1 つの実験における K^i と i の $Pausing$ による待機時間の関係をプロットする．ここで，K^i は 5,000,000 ステップでの値であり，待機時間は実験の最後の 1,000,000 ステップでの待機時間の総和である．したがって，停止したエージェント i の K^i は停止したときの値であり，待機時間は 0 である．図 35 での 6 つの赤いプロットが停止エージェントに対応する．この図でも，停止エージェントは K^i が大きく，非停止エージェントは非常に短い待機時間で巡回を続けていることが分かる．また，待機時間が 500,000 以上であるが停止していないエージェントが 1 体確認することができる．これは，実験 5 では停止していたエージェントである．他の独立した実験でも，実験設定は同じであるが，停止

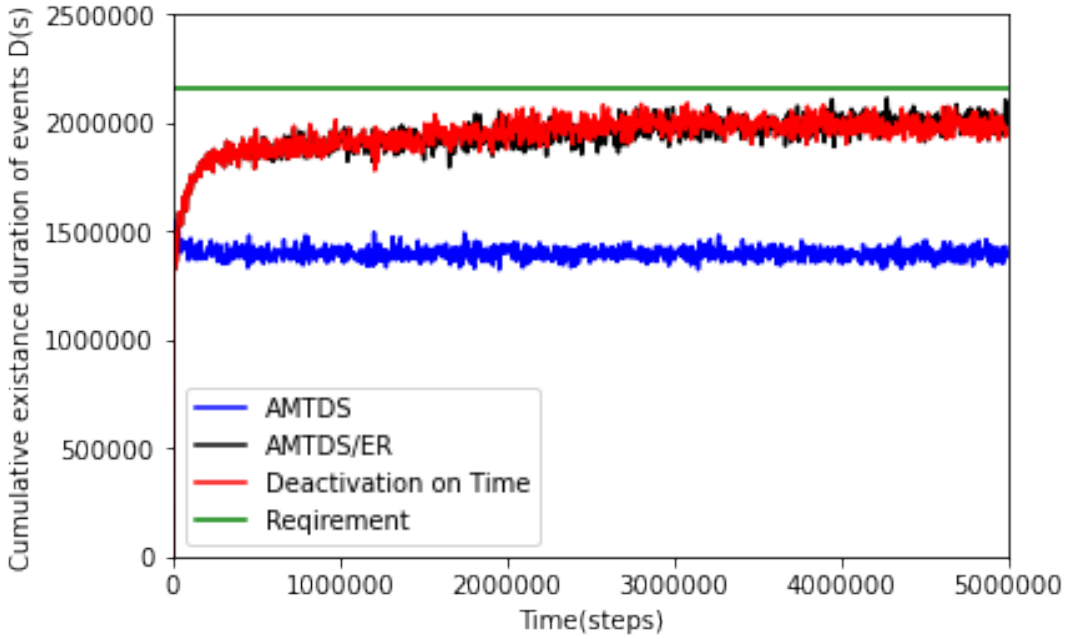


図 33: $D(s)$ の時間推移 (実験 6)

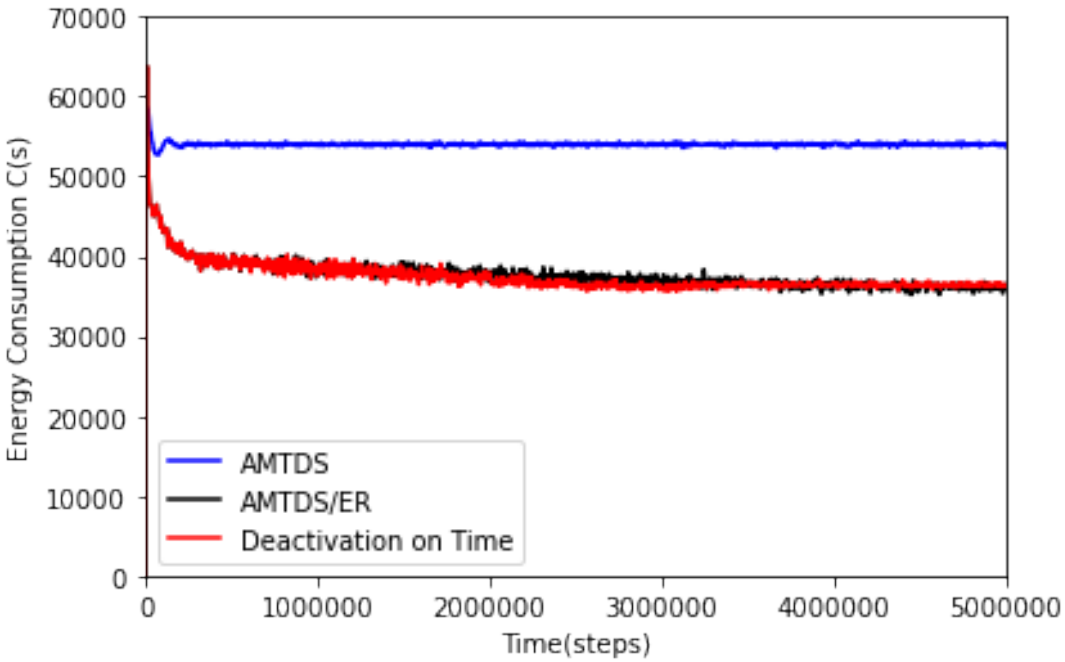


図 34: $C(s)$ の時間推移 (実験 6)

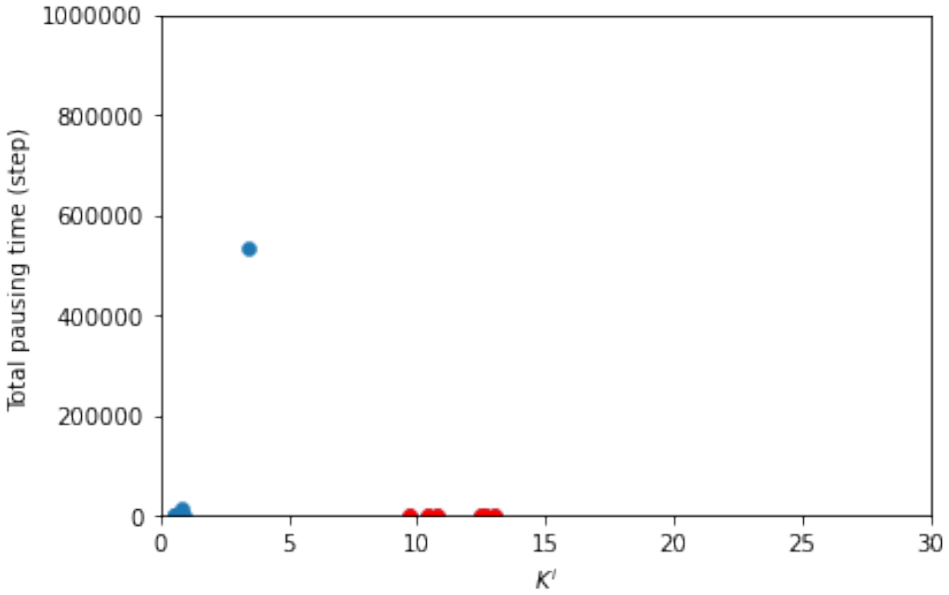
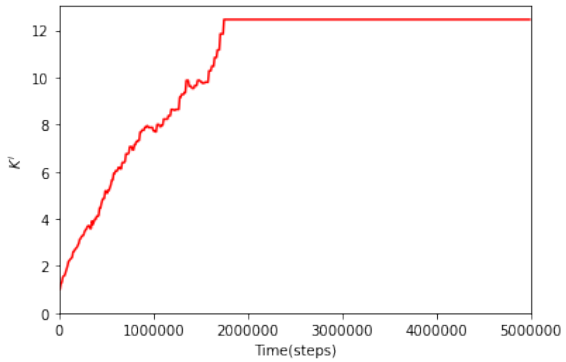
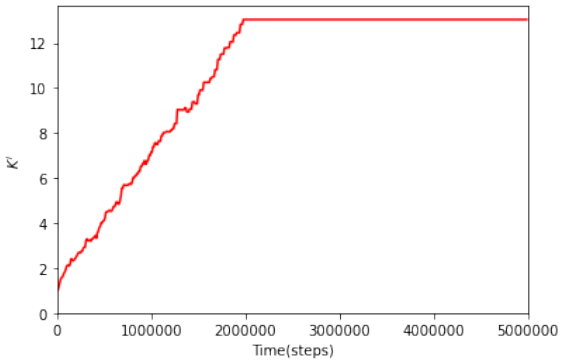


図 35: エージェントごとの K^i と待機時間の関係 (実験 6)

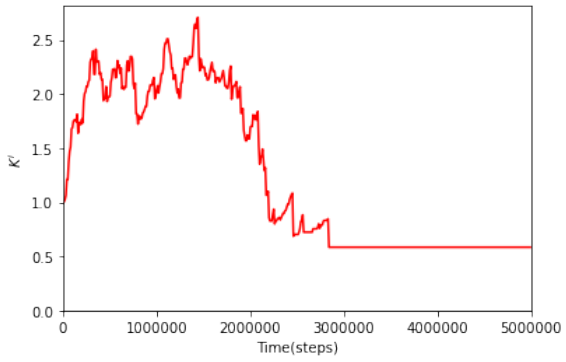


(a) $D = 1,750,000$ ステップ

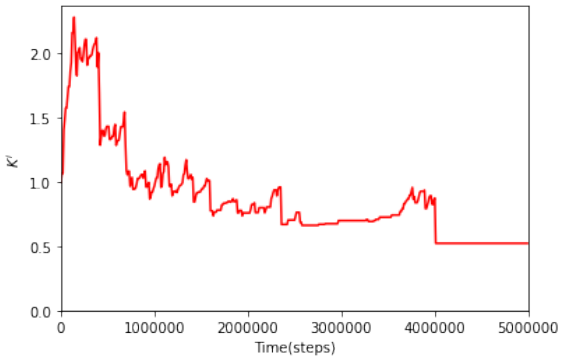


(b) $D = 2,000,000$ ステップ

図 36: D ステップで停止したエージェントの K^i の時間推移 (実験 6)



(a) 例 1



(b) 例 2

図 37: 非停止エージェントの K^i の時間推移 (実験 6)

したエージェント数は Deactivation の方が Deactivation on Time 以上であり，50 試行の平均では，1.1 体多かった．つまり，Deactivation on Count では，図 30 のように，非停止エージェントの待機時間は 0 に近かったが，Deactivation on Time では，比較的待機時間が大きいエージェントが 1 体残ったという結果になった．しかし，停止手法の優劣を決める際に，停止したエージェント数が多いという点が必ずしも優先されるべきではない．実験 5 と実験 6 では，停止したエージェントが巡回を再開することはないため，停止をしすぎて品質要求を満たせないということはあってはならない．このような条件の際に，Deactivation on Time では 5.2.2 項で述べたように，しっかり役割分担が行われてから停止を始めるようにしており，エージェントを停止しても，そのエージェントの分の巡回を残りのエージェントでカバーすることができることが保証されてから停止をするようになっている．しかし，Deactivation on Count では，現在のエージェント数では要求品質を十分に満たせていることは確認できているが，エージェントの停止によって，どの程度の影響があるかは確認できずに停止している．つまり，Deactivation on Count の方では，エージェントを停止しすぎてしまう可能性もある．図 CountStop と TimeStop とのグラフを合わせる，C(s) のグラフに Deactivation on Count と Deactivation on Time の $D(s)$ と $C(s)$ の時間推移をプロットした．これらの図から分かるように，両者の巡回効率やエネルギー消費量はほとんど差がないことを考慮すると，停止手法としてより望ましいのは Deactivation on Time の方であると考えられる．

次に，実験 6 における K^i の推移を調べた．図 36 と図 37 は，1,750,000 ステップと 2,000,000 ステップで停止したエージェント (図 36) と，非停止エージェント (図 37) の K^i の時間推移をプロットしたものである．式 (50) での平均シルエット係数 S と閾値 S_{deact} の比較のため，停止し始めた時刻は実験 5 よりも遅いが，大まかな結果は 6.3.2 項と同様である．停止エージェントは K^i が急激に増加し停止の対象に選ばれた一方で，非停止エージェントは一時的に K^i を増加させたが，一部のエージェントが停止したため，K^i が減少した．

さらに，Deactivation on Time でエージェント数 $|A|$ を 30 にして実験を行った．

6.4　AMTDS/ERC についての実験結果

6.4.1　実験 5: 性能評価

6.4.2　実験 6: 環境の変化による性能の違い

6.4.3　実験 7: エージェント数減少による性能の変化

7　結論

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] M. Ahmadi and P. Stone. Continuous area sweeping: a task definition and initial approach. In *ICAR '05. Proceedings., 12th International Conference on Advanced Robotics, 2005.*, pp. 316–323, 2005.

[8] Noa Agmon, Daniel Urieli, and Peter Stone. Multiagent patrol generalized to complex environmental conditions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 25, pp. 1090–1095, 2011.

[9] 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.

[10] 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.

[11] Yann Chevaleyre, Francois Sempe, and Geber Ramalho. A theoretical analysis of multi-agent patrolling strategies. In *AAMAS*, Vol. 4, pp. 1524–1525, 2004.

[12] 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.

[13] Xin Zhou, Weiping Wang, Tao Wang, Yonglin Lei, and Fangcheng Zhong. Bayesian reinforcement learning for multi-robot decentralized patrolling in uncertain environments. *IEEE Transactions on Vehicular Technology*, Vol. 68, No. 12, pp. 11691–11703, 2019.

[14] 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.

[15] 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.

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

[18] 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.

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

[20] 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.