2022年度　卒業論文

未定

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　　2022/

指導教授　　菅原 俊治

目 次

概 要

本研究では,

1　研究背景

近年, ロボット技術が発達し, 様々な分野で活躍するようになった. それに伴い,

2　関連研究

3　モデルの定義

本研究は [1] の手法を拡張し, 充電基地での待機時間を動的に決定することにより, エネルギー効率の向上を図る. そこで, 従来手法の比較を行うため, 本研究に用いるモデルは [1] と同一のものとする. この章では,[1] で用いたモデルについて詳しく説明する.

3.1　環境

エージェントの巡回環境を 2 事変ベクトル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, 各ノード上にエージェントやイベント, 障害物が存在する. また,E はノード v_i と v_j 間のエッジ $e_{i,j}$ の集合である.

本環境では, 単純化のために, エッジの長さはすべて 1 とする. v_i と v_j 間の最短距離を $d(v_i, v_j)$, ユークリッド距離を $m(v_i, v_j)$ とする. 時間は離散時間であり, その最小単位としてステップを用いる. 各エージェントは, 毎ステップ, 隣接した障害物のないノードに移動し, そのノード上のイベントを処理することができる.

すべてのノード $v \in V$ において,v 上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 のイベント発生量 $L_t(v)$ は, 以下の式で更新される.

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

任意のエージェントが時刻 t において v を訪問と,v 上のイベントが処理されるため,$L_t(v)$ の値は 0 となる.

3.2　エージェント

エージェントの集合を $A = \{1, \ldots, n\}$ と表す. 本研究では, 各エージェントは自身と他のエージェントの位置が把握できるものとする. これは, 赤外線などのセンサーや GPS を利用することによって実用可能であり, それぞれのエージェントが持つ情報の交換に比べて比較的容易に取得できるためである. しかし, エージェントの目的地や経路に関する情報は共有しない. また, 単純化のため, 同一のノードに複数のエージェントが存在できるものとし, エージェント同士の衝突は考慮しない.

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. 各エージェントはこの値を用いてイベント発生量の推定や目標ノードを決定する.

各エージェントはイベント発生量 $L(v)$ の値を直接把握することはできない. そのため, エージェント i は $p^i(v)$ の値を用いて, 毎時刻 t において, ノード v のイベント発生量の推定値 $EL_t(v)$ を以下の式で計算する.

$$EL_t^i(v) = p^i(v) \times (t - t_{vis}^v) \tag{2}$$

ここで,t_{vis}^v はノード v に任意のエージェントが最後に訪問した時刻である. 各エージェントは他のエージェントの位置が把握できるという仮定より, 時刻 t_{vis}^v が分かる.

各エージェントは, 十分な容量のバッテリを持ち, 充電基地で充電を行う. エージェント i は充電基地 v_{base} を出発点とし, バッテリを消費しながら環境内を巡回し, 再び v_{base} に戻り充電をするというサイクルを繰り返す. エージェント i のバッテリ性能を $(B_{max}^i, B_{drain}^i, k_{charge}^i)$ で表す. ここで,B_{max}^i はエージェントのバッテリ容量,B_{drain}^i は 1 ステップで消費するバッテリ消費量,k_{charge}^i 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 $b^i(t)(0 \leq b^i(t) \leq B_{max})$ 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.

$$b^i(t+1) \leftarrow b^i(t) - B_{drain}^i \tag{3}$$

$b^i(t)$ 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_{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.

$$\mathcal{P}^i(v) = d(v, v_{base}) \times B_{drain}^i \tag{4}$$

エージェント i は目標ノード v_{tar}^i 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_{tar}^i 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.

$$b^i(t) \leq \mathcal{P}^i(v) + d(v_t^i, v_{tar}^i) \times B_{drain}^i \tag{5}$$

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.

$$v_{tar}^i \leftarrow v_{base} \tag{6}$$

エージェントは充電基地に到着した後, バッテリ残量が最大になるまで充電し, 充電が完了した後, 再び環境内を巡回する.

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.

本研究のように,MACPP における清掃問題の場合, イベントはごみであり, 環境内のごみの総量と放置されている時間を減らすことが目的となる. これを示す指標として, 時刻 t_s から時刻 t_e までのイベント残存時間の総和 D_{t_s,t_e} を以下の式で定める.$(t_s < t_e)$

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

また, エージェントの総エネルギー消費量を示す指標として, 時刻 t_s から時刻 t_e までのエネルギー消費量の総和 C_{t_s,t_e} を以下の式で定める.$(t_s < t_e)$

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

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,[1] 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. ここで, 本研究では清掃問題を扱っているため,D_{t_s,t_e} が D_{req} 付近にして,C_{t_s,t_e} をより小さくすることを目指す. また, 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/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) [2] 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.

$$u_{t_0,t_0+d_{travel}} = \frac{\sum_{t_0+1\leq t\leq t_0+d_{travel}} L_t(v^i_t)}{d_{travel}} \tag{9}$$

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.

$$Q^i(s) \leftarrow (1-\alpha)Q^i(s) + \alpha \times u_{t_0,t_0+d_{travel}} \tag{10}$$

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.

$$s_{next} \leftarrow \arg\max_s Q^i(s) \tag{11}$$

4.1.1 目標決定戦略

[2] 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 $EL^i_t(v)$ 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 $EL^i_t(v)$ 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.

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.

$$EV^i_t = \frac{\sum_{v\in V^i_{area}} EL^i_t(v)}{|V^i_{area}|} \tag{12}$$

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.

$$EV_{threshold} \leftarrow EV_{threshold} + \alpha(EV_{t+1}^i - EV_{threshold}) \tag{13}$$

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.

$$\begin{cases} d(v_t^i, v) \leq d_{myopia} \\ d(v, v_{tar}^i) < k_{att}(d(v_t^i, v_{tar}^i)) \\ d(v_t^i, v) + d(v, v_{tar}^i) \leq k_{rover} d(v_t^i, v_{tar}^i) \\ \mathcal{P}^i(v_{tar}^i) + B_{drain}^i \times (d(v_t^i, v) + d(v, v_{tar}^i)) \leq b^i(t) \end{cases} \tag{14}$$

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.

$$v_{subgoal}^i \leftarrow \arg\max_{v \in V_{sub}^i} EL_t(v) \tag{15}$$

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) [3] 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.

$$I^i_t(v) = t - t^v_{vis} \tag{16}$$

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.

$$p^i(v) \leftarrow (1-\beta)p^i(v) + \beta\frac{L_t(v)}{I^i_t(v)} \tag{17}$$

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 $EL^i_t(v)$ 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 for energy saving and cleanliness (AMTDS/ESC)

AMTDS や AMTDS/LD 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 (9) 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) [1] が提案された.

$$r_{t_0,t_0+d_{travel}} = \frac{\sum_{t_0+1\leq t\leq t_0+d_{travel}} L_t(v^i_t)}{d_{travel} \times \sum_{t_0+1\leq t\leq t_0+d_{travel}} E_t(i)} \tag{18}$$

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ている. 図 1 はエージェントにおけ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.

4.3.1 要求充足の判断

エージェントは, エネルギー消費を節約すると同時に, 総イベント量が与えられた要求値 D_{req} よりも小さくしなければならないので, 次の行動を決定するには, 環境全体のイベン

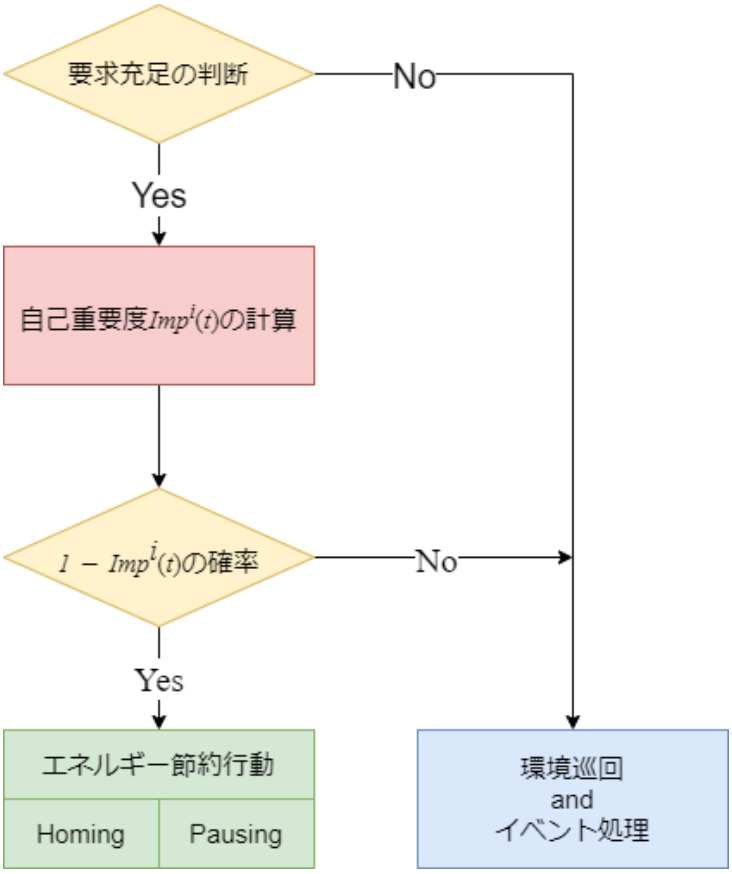

図 1: エージェントの行動選択

ト量を推定する必要がある. そこで, エージェント i は自身が持っている $p^i(v)$ の値を用いて, 以下の式に従って時刻 t における総イベント量 EV_t^i を推定する.

$$EV_t^i = \sum_{v \in V} EL_t^i(v) \tag{19}$$

その後,EV_t^i が D_{req} より小さいかを判断するために, 以下の式を用いて, これを満たしていれば i は自己重要度評価を行い, 満たしていなければ通常通りに環境循環やイベント処理を行う.

$$EV_t^i \leq D_{req} \tag{20}$$

4.3.2 自己重要度評価

式 (20) を満たしていたとき, 各エージェントは自身がどの程度重要であるかを知るために, エージェントの貢献度を表す自己重要度を評価する. エージェントは以下の二つを考慮して, 自己の重要度を求める.

(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

まず, エージェント i は, 以下の式に従って求められる, 短期的な過去の貢献度である U_s^i と, 長期的な過去の貢献度である U_l^i, 近い未来における予測される貢献度である U_f^i を計算する.

$$\begin{cases} U_s^i = u_{t_c - T_s, t_c} = \dfrac{\displaystyle\sum_{t_c - T_s < t \leq t_c} EL_t(v^i(t))}{T_s} \\ U_l^i = u_{t_c - T_l, t_c} = \dfrac{\displaystyle\sum_{t_c - T_l < t \leq t_c} EL_t(v^i(t))}{T_l} \\ U_f^i = u_{t_c, t_c + T_f} = \dfrac{\displaystyle\sum_{t_c < t \leq t_c + T_f} EL_t(v^i(t))}{T_f} \end{cases} \tag{21}$$

ここで,t_c は現在時刻,T_s と T_l,T_f はどのくらいの期間の過去や未来の貢献度を計算するかを決定する固定整数である ($T_s < T_l$). これらを用いて, 以下の式に従って自己重要度 $Imp^i(t)(0 \leq Imp^i(t) \leq 1)$ を求める.

$$Imp^i(t) = \begin{cases} \dfrac{U_s^i + U_f^i}{U_l^i} & (U_s^i + U_f^i \leq U_l^i) \\ 0 & (U_l^i = 0) \\ 1 & (\text{その他}) \end{cases} \tag{22}$$

以上のように求められた $Imp^i(t)$ を用いて, 以下の式に従って求められた $P^i(t)$ の確率で,i が Homing や Pausing といったエネルギー節約行動を選択する.

$$P^i(t) = 1 - Imp^i(t) \tag{23}$$

4.3.3　帰還動作 (Homing)

この行動は, エージェントが巡回をやめて充電基地に戻ることで, エネルギーを節約する. エージェント i は T_{check} ステップ移動した後, 要求充足の判定や自己重要度評価を行い,Homing を行うかどうかを決定する. なお,i は電池残量 $b^i(t)$ が以下の式を満たすときにしか, この行動は行わない.

$$b^i(t) < k_{homing} \times B^i_{max} \quad (0 < k_{homing} < 1) \tag{24}$$

この制約は, エージェントが十分に巡回せずに, 頻繁に充電基地に戻ることを防ぐために追加している. Homing を行うことによって,i は v^i_{tar} を v_{base} に更新し, 充電基地に着いたら充電を行う. 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評価は行わない.

4.3.4　待機動作 (Pausing)

この行動は, エージェントが充電基地で充電が完了した後も, 充電基地で待機することで, エネルギーを節約する. エージェント i は充電基地で充電が完了した後, 要求充足の判定や自己重要度評価を行い,Pausing を行うかどうかを決定する. Puasing 中は,i はイベントを処理はせずに,T_p ステップ待機する.

5　提案手法

AMTDS/ESC における Homing と Pausing を組み合わせただけでは, イベント量が要求値 D_{req} まで許容されていてもエネルギー消費量はほとんど変わらない. また, 実際に巡回ロボットに適用する場合, イベント量を最小にするよりも, ある程度の要求値 D_{req} を許容したうえでロボットのバッテリ消費を抑えたほうが, 現実的に考えた際に有用である. したがって, 本研究では充電基地での待機時間を各エージェントが未来の環境全体のイベント量の予測を行い, これを用いて動的に決定する手法を AMTDS/ESC に加えた AMTDS for energy saving under the requirement (AMTDS/ER) と, 更にイベント発生量の学習を加えた AMTDS for energy saving under the requirement with learning of event probabilities (AMTDS/ERL) を提案する.

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

この節では,AMTDS/ESC をベースに, 充電基地での待機時間を各エージェントが動的に決定する手法である AMTDS/ER について説明する.

5.1.1 Homing と Pausing の組み合わせ

AMTDS/ESC では,Homing と Pausing はどちらか一方しか実装されていなく, それぞれに対して実験, 評価を行った. 本研究では, エネルギー消費量をより小さくするために,Homing と Pausing を組み合わせ, 変更を加えた. 具体的には,Homing の動作は AMTDS/ESC と同じだが,Pausing は AMTDS/ESC のときに加えて,Homing によって充電基地へ帰還し, 充電が完了した後は必ず Pausing を行うようにした. この変更を行った理由は,Homing 単体では, 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 $\rightarrow \cdots$」のサイクルが短くなっただけで, エネルギー消費量に対してはあまり影響がない. したがって,Homing を行った後は必ず Pausing を行うことによって, エネルギー消費量を小さくすることを目指す.

5.1.2 補正係数 K の導入

未来のイベント発生量の予測は充電基地での待機中での他のエージェントによるイベント処理を考慮していないため, D_{req} を超えないように決定した待機時間だけ充電基地で待機しても, 実際のイベント残存時間は基本的に D_{req} よりも小さくなる. 従って, エージェントごとに持つ補正係数 K^i を導入する. これを用いて, イベント残存時間が $D_{req} \times K^i$ を超えないように待機時間を決定するようにする. また,K^i はエージェントが待機し終わって充電基地を出発する際に, 実際のイベント量 D_t を用いて以下の式に従って更新する.

$$\begin{cases} K^i \leftarrow (1-\alpha)K^i + \alpha \dfrac{D_{req}}{D_t} K^i & (D_t \leq D_{req}) \\ K^i \leftarrow K^i - \left(\dfrac{D_t}{D_{req}} - 1 \right) & (D_t > D_{req}) \end{cases} \tag{25}$$

5.1.3 未来のイベント発生量の予測

エージェントが持っている $p^i(v)$ を用いて,Tstep 後の未来の環境全体のイベント発生量を以下の式に従って予測する.

$$EL^i_{t+T} = \sum_{v \in V} p^i(v) \times \{(t+T) - t^v_{vis}\} \tag{26}$$

ここで, 実行時間の短縮化のため,T は $n \times T_{basic} (n \in \mathbb{N})$ である. 本研究では, これを用いて, イベント量 D_t が $D_{req} \times K^i$ を超えないステップまで充電基地で待機するように待機時間を決定する.

5.2 AMTDS for energy saving under the requirement with learning of event probabilities (AMTDS/ERL)

AMTDS/ER 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/ER に環境のイベント発生確率の学習を加えたAMTDS/ERL を提案する.AMTDS/ER との違いは以下の 2 つである.

5.2.1　イベント発生量の予測に使用するノードの範囲の変更

AMTDS/ER では最初からイベント発生確率 $p(v)$ を把握しているため, 環境全体の総イベント発生量の予測の方法は単純である. 各ノードのイベント発生量を予測し, それを全ノード計算すればよいからである. しかし,$p(v)$ が未知で学習を行う際には, 特に未訪問のノードのイベント発生確率は初期値のままであり, 予測が十分に行えない. よって, 未来のイベント発生量の予測は訪問済みのノード V_{vis} のみで行い, その値から訪問済みのノード数 N_{vis} と壁を除く全ノード数 N_{all} を用いて以下の式に従って求める. なお, 学習のため Homing と Pausing は T_{hp} ステップまでは行わない.

$$EL^i_{t+T} = \sum_{v \in V_{vis}} p^i(v) \times \{(t+T) - t^v_{vis}\} \times \frac{N_{all}}{N_{vis}} \tag{27}$$

5.2.2　補正係数 K^i の更新方法の変更

AMTDS/ER では式 (25) に従って K^i を更新していたが,AMTDS/ERL では以下の式に変更した. つまり, 式 (25) の場合分けをなくし,D_{req} や D_t の値によらず, 同じ式で K^i を更新するようにした.

$$K^i \leftarrow (1-\alpha)K^i + \alpha \frac{D_{req}}{D_t} K^i \tag{28}$$

6　評価実験

提案手法である AMTDS/ER,AMTDS/ERL と, 従来手法である AMTDS,AMTDS/LD を比較する実験を行い,D(s) と C(s) を評価指標として用いて, 提案手法がイベント量を要求値付近まで許容しつつ, エネルギー消費を抑えることに有効であることを示す. 加えて, 異なる環境においても有効であることも示す. 最後に, エージェントを途中で停止させ, エージェントのエネルギ抑制行動の変化を分析し, エージェント数の変化によって全体のイベント処理量が変化した際のエージェントの行動の変化を考察する.

6.1　実験環境

提案手法の評価実験として, 以下に示す実験環境を用意する. エージェントが巡回する環境を図で表したものが, 図 2 である. 図 2 に示す通り, 環境は 6 つの部屋で構成されており, 部屋と部屋の間には廊下が存在する. また, 黒色で表した障害物を含む 101×101 の 2 次元グリッド構造となっている. 各ノード $v \in V$ は座標 (x_v, y_v) $(-50 \leq x_v, y_v \leq 50)$ に存

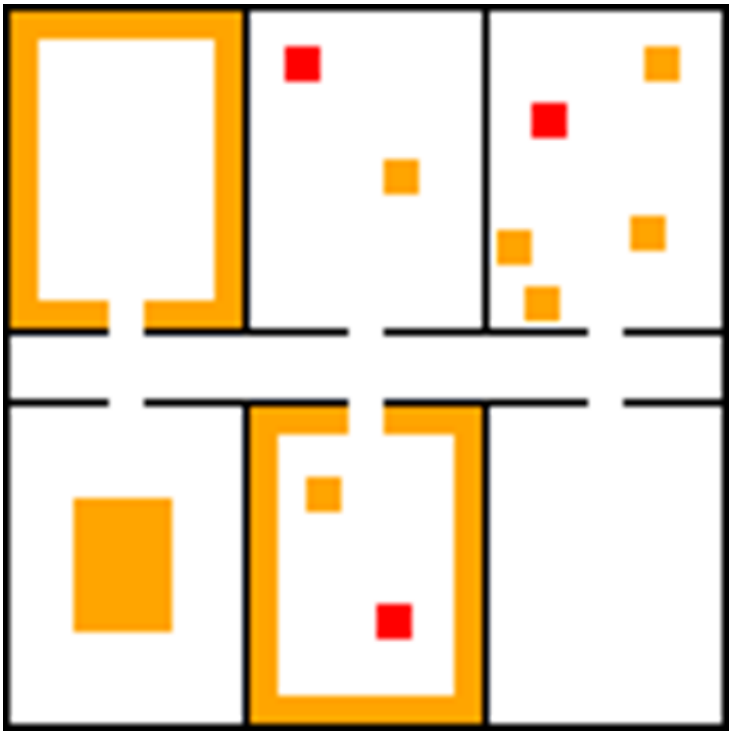

図 2: 実験環境

在する. 環境内のノード集合 V に対して, ノード $v \in V$ のイベント発生確率 $p(v)$ は図 2 の色に応じて, 以下の式で表される.

$$p(v) = \begin{cases} 10^{-3} & (\text{赤色の領域}) \\ 10^{-4} & (\text{オレンジ色の領域}) \\ 10^{-6} & (\text{白色の領域}) \end{cases} \tag{29}$$

色が濃くなるほど, イベントの発生確率が高いことを示す.

エージェントの数は $20(|A| = 20)$ とし, 充電基地 v_{base} を環境の中心である $(0, 0)$ に配置する. エージェントはバッテリ残量が最大の状態で v_{base} を出発し, 自身の戦略で環境内を循環し, バッテリ残量が 0 になる前に再び v_{base} に戻り, 充電を行う. 各エージェントはこの一連の流れを繰り返す. すべてのエージェント $i \in A$ のバッテリ性能は $(B^i_{max}, B^i_{drain}, k^i_{charge}) = (900, 1, 3)$ とする. つまり, 各エージェントは満充電の状態から最大で 900 ステップ移動でき, バッテリ残量が 0 から満充電となるまでに 2700 ステップかかる. したがって, 移動と充電のサイクルが最大で 3600 ステップとなるため, 本研究では評価指標 $D(s)$ と $C(s)$ のデータ収集間隔 $t_e - t_s$ を 3600 とした.

本実験では,1 回の試行のステップ数は評価実験によって異なるが, どの評価実験も, 試行を 50 回繰り返した平均値をとる. また, 本実験で用いるパラメータの値を表 1, 表 2, 表 3 にまとめた.

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}	$\frac{1}{3}$
Pausing	T_{basic}	100
AMTDS/ERL	T_{hp}	500,000

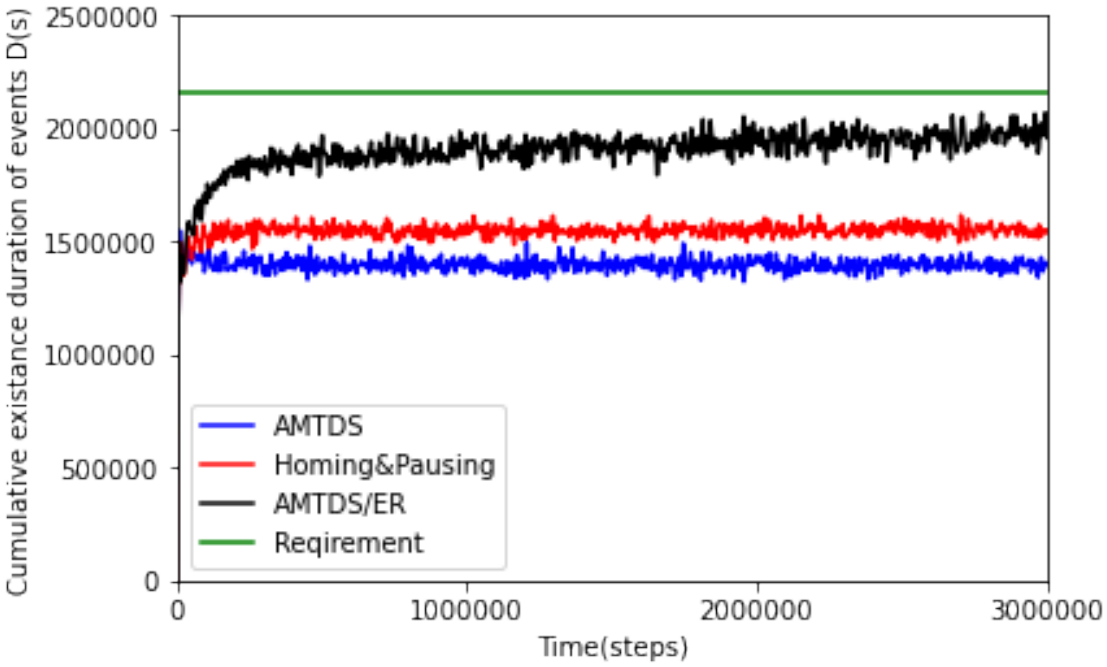

図 3: $D(s)$ の時間推移の比較 (実験 1)

6.2　AMTDS/ER についての実験結果

この節では, 提案手法である AMTDS/ER と, 従来手法であり AMTDS における評価実験結果を示す. 複数の条件下での実験結果を, 評価指標である $D(s)$ と $C(s)$, さらには, 必要に応じて他の指標も比較していく.

6.2.1　実験 1: 性能評価

従来手法である AMTDS と, 提案手法である AMTDS/ER における,D(s) と C(s) の比較結果を図 3,4 に示す. また, 本実験での総ステップ数は 3,000,000 ステップであり, 要求値 D_{req} は 600, 補正係数 K^i の初期値は 1.0 である.

まず, 提案手法である AMTDS/ER は,D(s) の値を従来手法である AMTDS よりも増加させ,D_{req} に近づけられていることが分かる. それにより,C(s) の値を AMTDS よりも減少させることができた. 具体的には,D(s) を D_{req} に近づけて約 35.5%増加させることにより,C(s) を約 28.2%削減した. つまり,AMTDS/ER のエネルギー節約行動により, 環境全体の汚れを D_{req} 付近にしつつ, より省エネな巡回が行うことができた.

次に, 環境全体ではなく, エージェントごとについて見ていく. AMTDS/ER では,$D(s)$ や $C(s)$ の変化は補正係数 K^i によって計算される Pausing による充電基地での待機時間の影響が大きい. また, これらは各エージェントによって異なるため, それに伴ってエージェントの行動も変わってくる. エージェントごとの K^i と Pausing による総待機時間をまとめたものが表 4,5,6 であり, これを散布図にまとめたものが図 5 である. なお,K^i は 3,000,000 ステップ時点の値で, 総待機時間は 2,000,000〜3,000,000 ステップでの総待機時

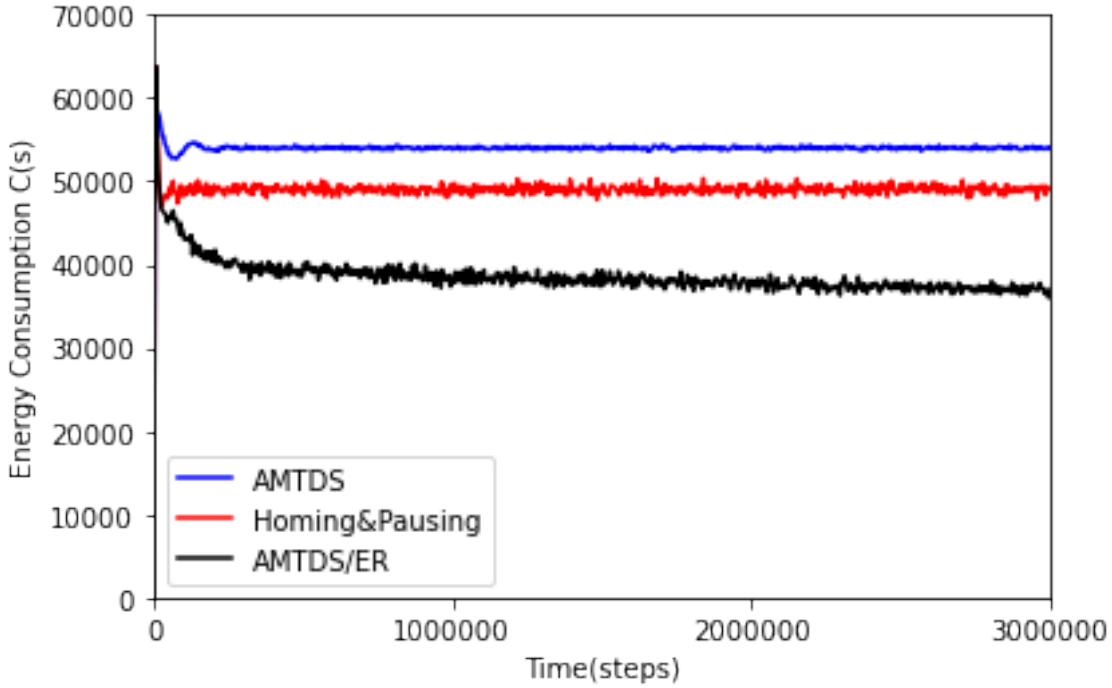

図 4: $C(s)$ の時間推移の比較 (実験 1)

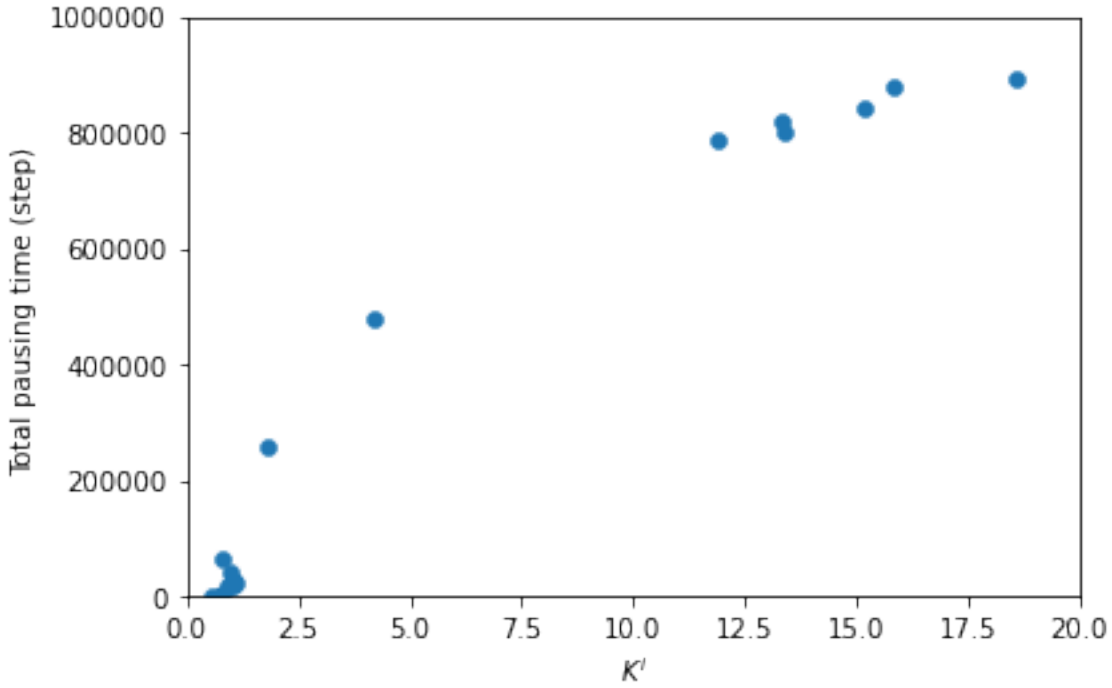

図 5: エージェントごとの K^i と総待機時間の関係 (実験 1)

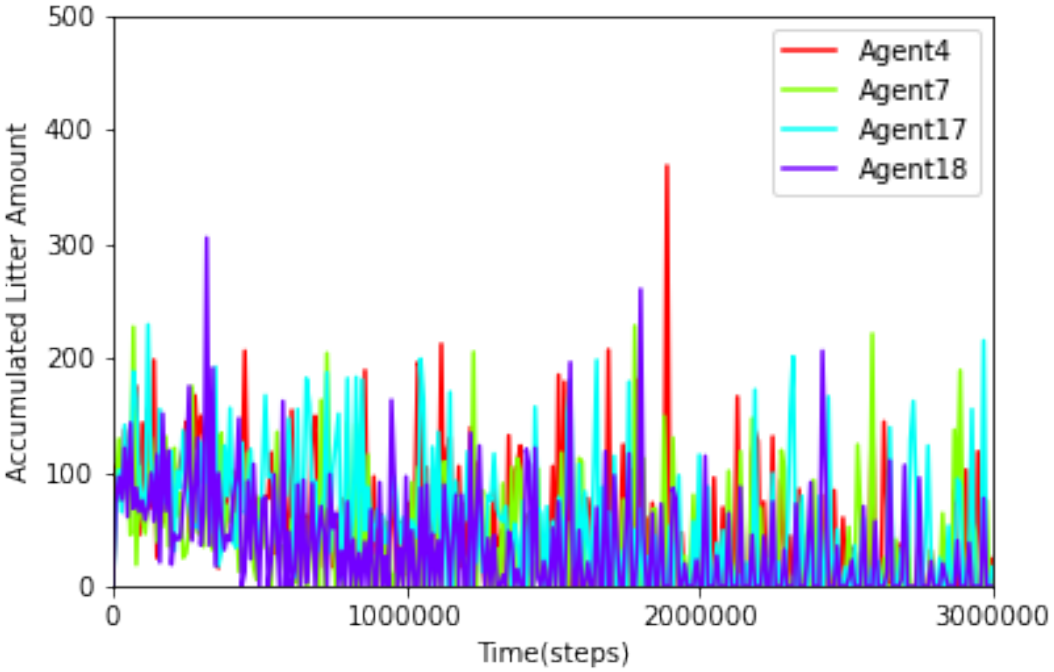

図 6: K^i 上位 4 体のエージェントによるイベント処理量の時間推移 (実験 1)

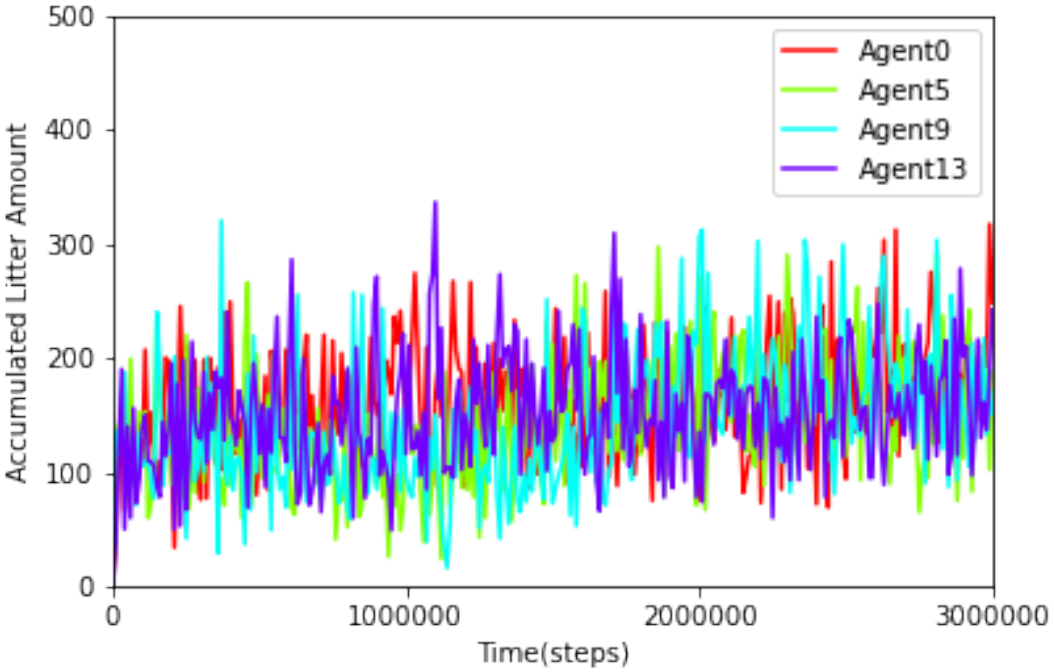

図 7: K^i 下位 4 体のエージェントによるイベント処理量の時間推移 (実験 1)

表 4: エージェント 0〜6 における K^i と総待機時間の比較 (実験 1)

エージェント	0	1	2	3	4	5	6
K^i	11.0250	0.5079	0.8457	0.5082	6.0391	7.7298	0.6987
総待機時間	786,800	0	28,100	0	609,700	698,600	0

表 5: エージェント 7〜13 における K^i と総待機時間の比較 (実験 1)

エージェント	7	8	9	10	11	12	13
K^i	0.7758	0.5405	0.4826	0.7002	9.5671	0.7050	0.8753
総待機時間	144,000	0	0	1,000	771,300	3,700	1,800

間である. これらの表から, エージェントによって K^i は大きく異なり, それによって総待機時間も異なっていることがわかる. また, エージェント通しで K^i の値や総待機時間といった情報の交換は行われていないが,Pausing を行わずに通常通り環境を巡回するエージェントと,1,000,000 ステップの間, 充電基地で待機している時間が長いエージェントとの役割分担が自然に行われたということが分かる. 特に, エージェント 1,3,6,8,9 は 1,000,000 ステップのうち,Pausing による充電基地での待機は行っていないのに対し, エージェント 17 は 810,900 ステップも待機している. 巡回と充電のサイクルが最大で 3,600 ステップであり, 通常はこのサイクルを 1,000,000 ステップでは少なくとも 277 回行うが, エージェント 17 は 225.25 サイクル分の巡回を行わずに充電基地で待機していたということが分かる.

さらに, 各エージェントのイベント処理量の時間推移のうち,K^i が上位 4 体を図 6, 下位 4 体を図 7 に示す. まず, 図 6 より, 時間が経つにつれて K^i が大きくなったエージェントは, それに伴って充電基地での待機時間が長くなるため, 環境を巡回する時間が短くなり, イベント処理量がだんだんと少なくなっていくことが分かる. 逆に, 図 7 より,K^i が小さいエージェントはだんだんとイベント処理量が多くなっていることが分かる. これは, 複数のエージェントの充電基地での待機時間が長くなると, ノード v が任意のエージェントに訪問される間隔が長くなり, それによってイベントが溜まっていくので,K^i が小さいエージェントのイベント処理量がだんだん多くなっていると考えられる.

以上のことから,AMTDS/ER は,AMTDS のように環境全体のイベント発生確率をあらかじめ把握した中で, イベント量を与えられた要求値付近にすることで, エネルギーを節約しながら巡回することができる手法といえる.

6.2.2　実験 2: 環境の変化による性能の違い

本実験では, 図 2 とは異なる環境である, 環境全体のイベント発生確率がより小さい環境を新しく用意し, 実験を行った. その環境を図 8 に示す. ここでは, 環境全体のイベント発生量が異なっても, 提案手法が有効であるかを検証するため, 図の色の濃さに対応するイベント発生確率は式 (29) と同じであるが, 色のついたノードの数を減らしている. また, 比

表 6: エージェント 14〜19 における K^i と総待機時間の比較 (実験 1)

エージェント	14	15	16	17	18	19
K^i	9.1121	10.1859	0.6054	11.8902	11.6212	0.8252
総待機時間	770,800	759,800	400	810,900	798,700	12,200

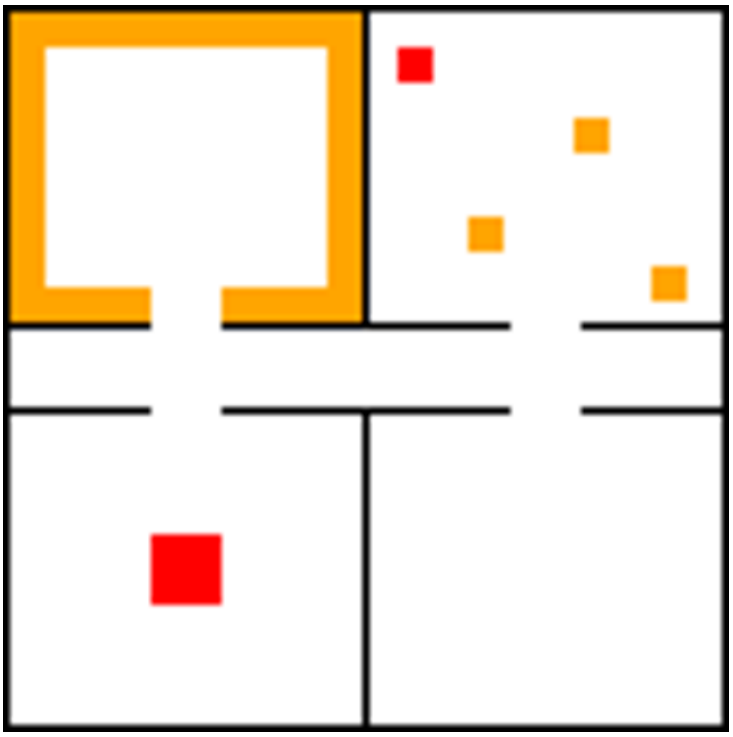

図 8: 実験環境 (全体のイベント発生確率が小さい場合)

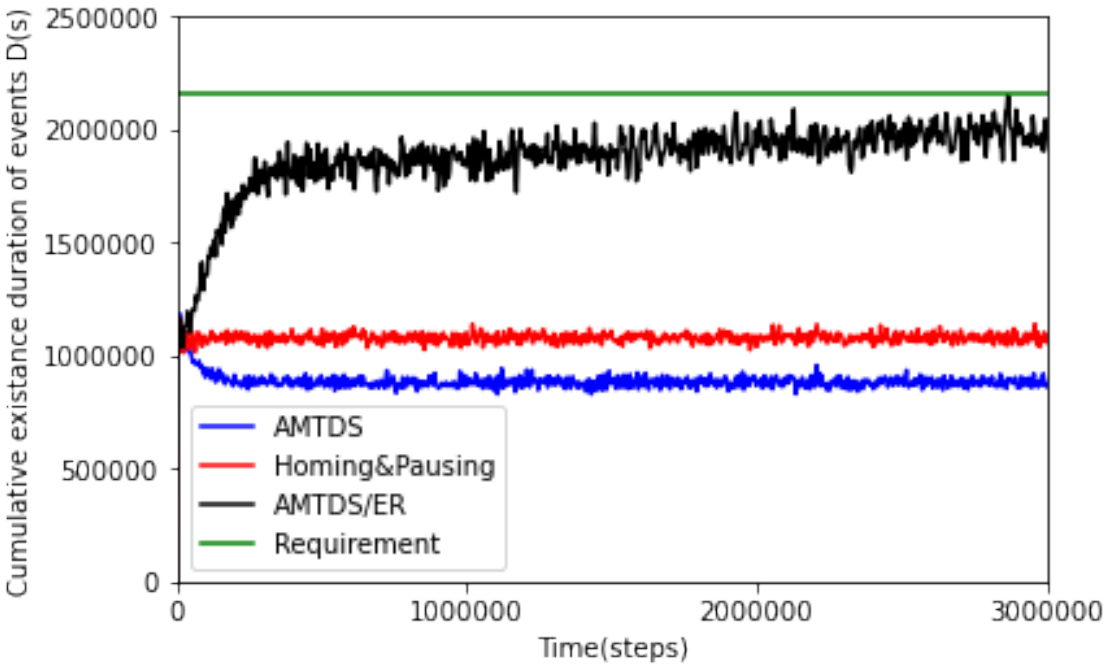

図 9: $D(s)$ の時間推移の比較 (実験 2)

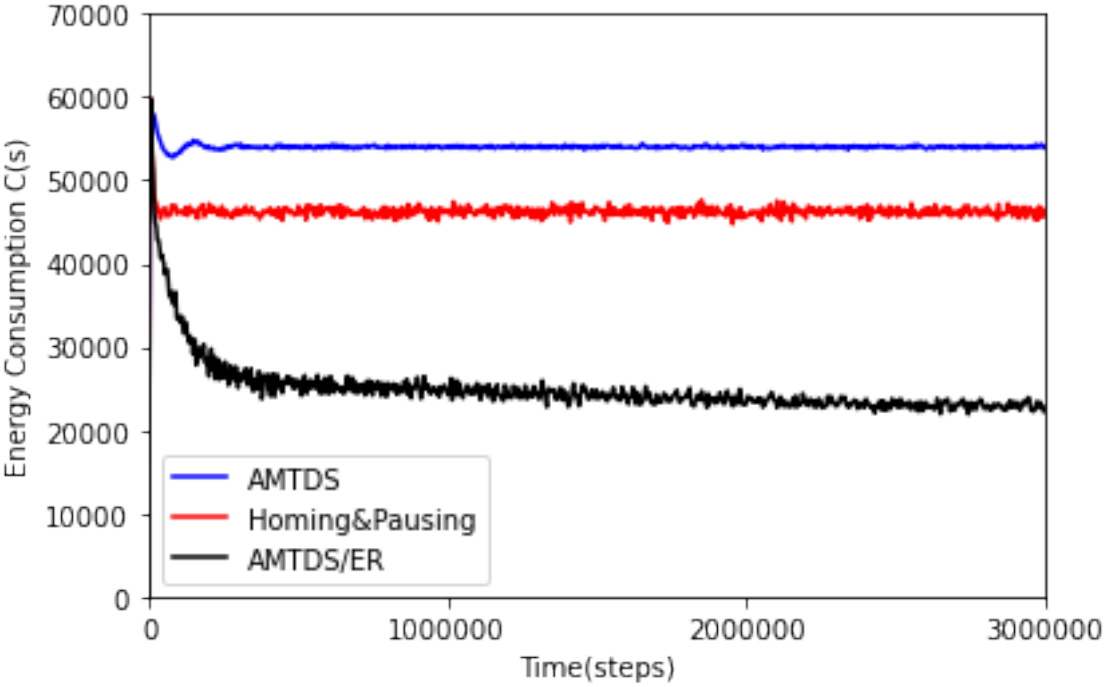

図 10: $C(s)$ の時間推移の比較 (実験 2)

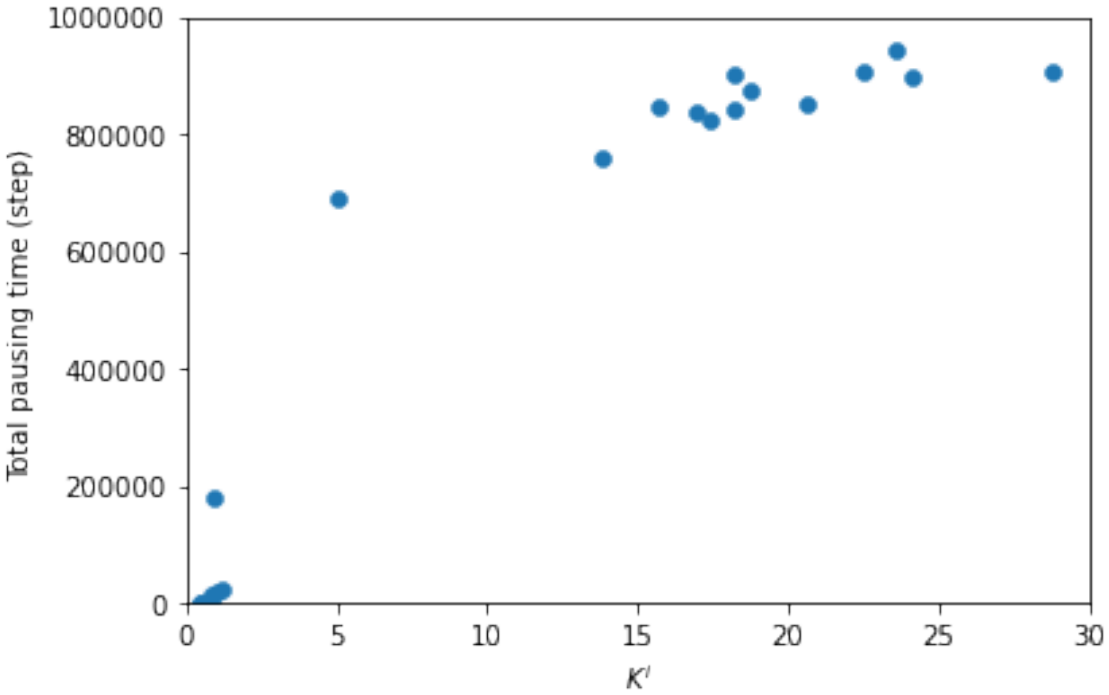

図 11: エージェントごとの K^i と総待機時間の関係 (実験 2)

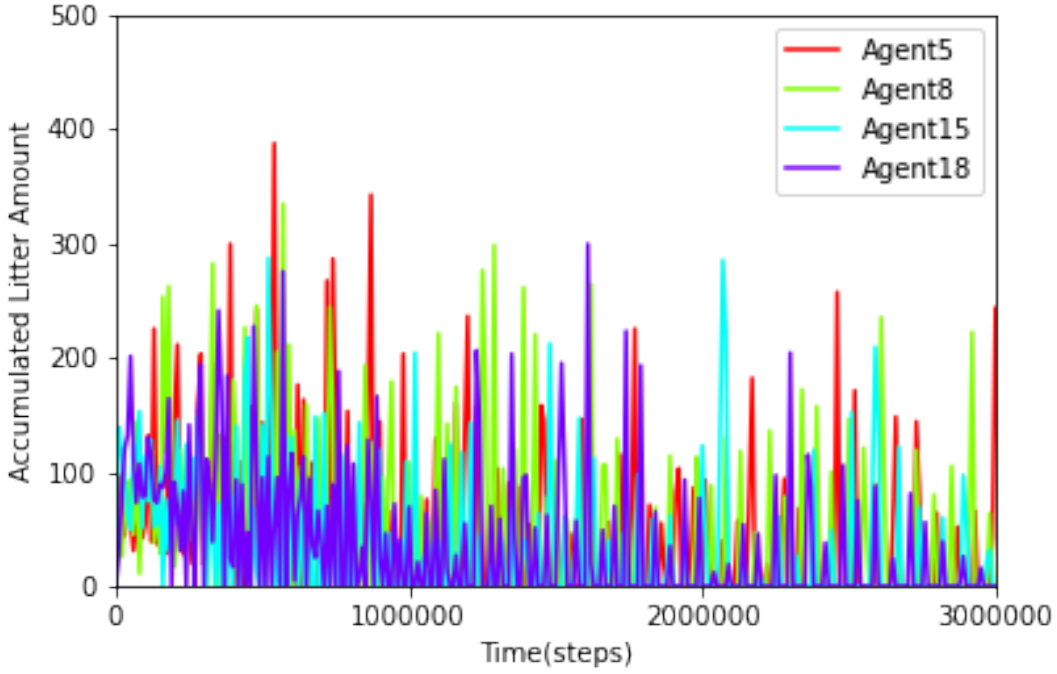

図 12: K^i 上位 4 体のエージェントによるイベント処理量の時間推移 (実験 2)

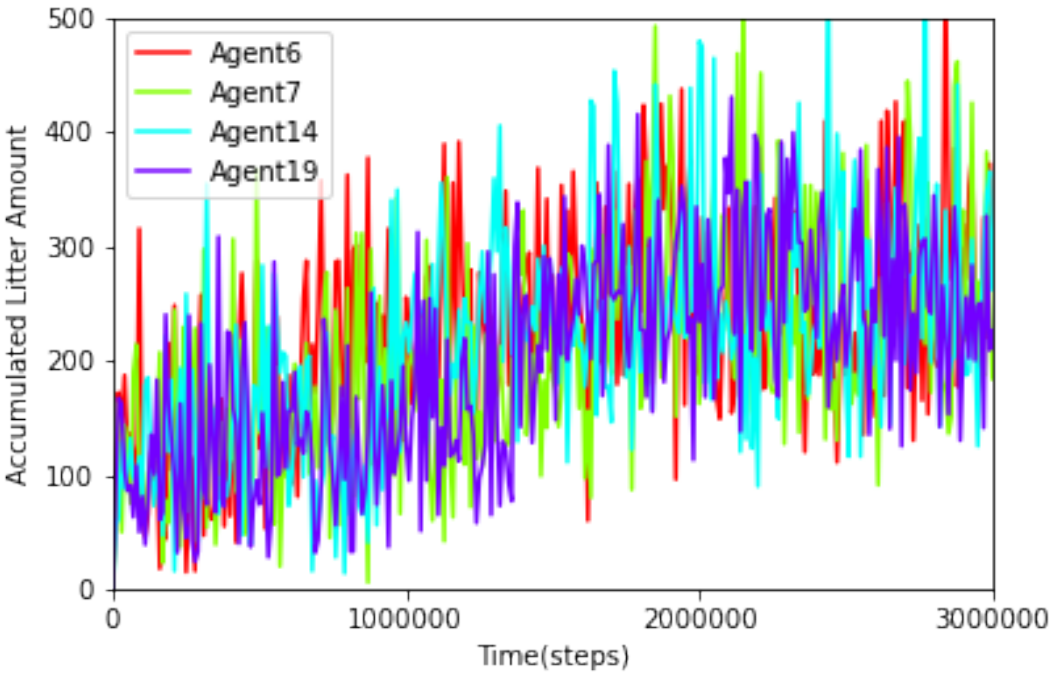

図 13: K^i 下位 4 体のエージェントによるイベント処理量の時間推移 (実験 2)

較を容易にするために,6.2.1 と同様に, 本実験での総ステップ数は 3,000,000 ステップであり, 要求値 D_{req} は 600, 補正係数 K^i の初期値は 1.0 である.

まず, 提案手法である AMTDS/ER は, 異なる環境においても,D(s) の値を従来手法である AMTDS よりも増加させ,D_{req} に近づけ,C(s) の値を AMTDS よりも減少させることができた. 具体的には,D(s) を D_{req} に近づけて約 112.5%増加させることにより,C(s) を約 54.1%削減した. つまり,AMTDS/ER のエネルギー節約行動により, 環境全体の汚れを D_{req} 付近にしつつ, より省エネな巡回が行うことができた. また,6.2.1 と比べると, 環境全体のイベント発生量は小さくても要求値 D_{req} を同じであるので,D(s) の増加率が大きくなるため, それに伴い C(s) の減少率も大きくなった.

次に, 環境全体ではなく, エージェントごとについて見ていく. エージェントごとの K^i と Pausing による総待機時間をまとめたものが表 7,8,9 であり, これを散布図にまとめたものが図 11 である. なお,6.2.1 と同様に,K^i は 3,000,000 ステップ時点の値で, 総待機時間は 2,000,000〜3,000,000 ステップでの総待機時間である. これらの表から, エージェントによって K^i は大きく異なり, それによって総待機時間も異なっていることがわかる. また, エージェント通しで K^i の値や総待機時間といった情報の交換は行われていないが,Pausing を行わずに通常通り環境を巡回するエージェントと,1,000,000 ステップの間, 充電基地で待機している時間が長いエージェントとの役割分担が自然に行われたということが分かる. 特に, エージェント 17,18, は 1,000,000 ステップのうち,Pausing による充電基地での待機は行っていないのに対し, エージェント 9 は 905,600 ステップも待機している. 巡回と充電のサイクルが最大で 3,600 ステップであり, 通常はこのサイクルを 1,000,000 ステップで

表 7: エージェント 0〜6 における K^i と総待機時間の比較 (実験 2)

エージェント	0	1	2	3	4	5	6
K^i	14.4864	14.2702	21.0839	14.2977	0.6541	12.2702	11.0955
総待機時間	843,300	846,800	877,400	840,800	4,200	808,600	861,400

表 8: エージェント 7〜13 における K^i と総待機時間の比較 (実験 2)

エージェント	7	8	9	10	11	12	13
K^i	18.9945	14.6801	21.5276	8.5475	18.3532	19.2192	0.5416
総待機時間	905,500	855,200	905,600	771,000	875,400	883,600	10,400

は少なくとも 277 回行うが, エージェント 9 は約 251.6 サイクル分の巡回を行わずに充電基地で待機していたということが分かる. さらに, 図 5 では充電基地に待機している時間が長いエージェントは 8 体だが, 図 11 では 14 体に増えている. これは, 本実験の環境である図 8 は図 2 と比べて環境全体のイベント発生確率が小さいため, 同じ要求値 D_{req} が与えられると,D(s) 要求値付近にするためには, 巡回を行わずに充電基地で待機する時間を長くしなければいけないので, 待機するエージェントが増加したと考えられる.

さらに, 各エージェントのイベント処理量の時間推移のうち,K^i が上位 4 体を図 12, 下位 4 体を図 13 に示す. まず, 図 12 より, 時間が経つにつれて K^i が大きくなったエージェントは, それに伴って充電基地での待機時間が長くなるため, 環境を巡回する時間が短くなり, イベント処理量がだんだんと少なくなっていくことが分かる. 逆に, 図 13 より,K^i が小さいエージェントはだんだんとイベント処理量が多くなっていることが分かる. これは, 複数のエージェントの充電基地での待機時間が長くなると, ノード v が任意のエージェントに訪問される間隔が長くなり, それによってイベントが溜まっていくので,K^i が小さいエージェントのイベント処理量がだんだん多くなっていると考えられる.

以上のことから,AMTDS/ER は, 環境が異なっても, 充電基地での待機時間を変化させることによってイベント量を要求値付近にして, エネルギーを節約しながら巡回することができる手法といえる.

表 9: エージェント 14〜19 における K^i と総待機時間の比較 (実験 2)

エージェント	14	15	16	17	18	19
K^i	0.8083	12.1195	0.7170	0.4826	0.3332	9.3735
総待機時間	1,100	824,900	10,200	0	0	790,600

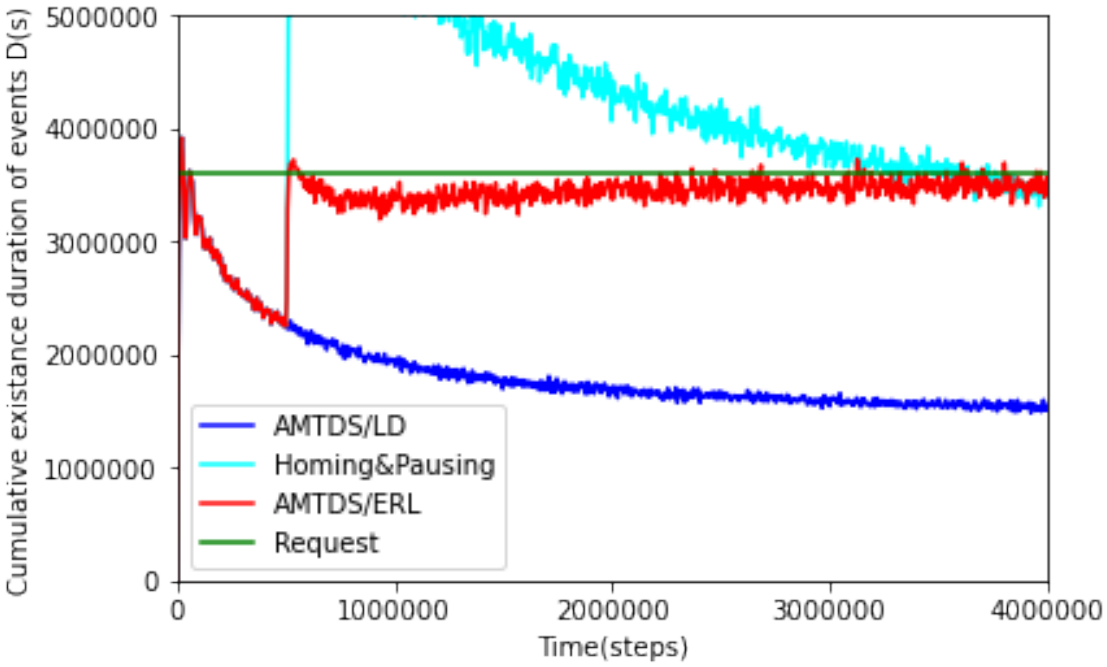

図 14: $D(s)$ の時間推移の比較 (実験 5)

6.2.3　実験 3: エージェント数減少による性能の変化

6.2.4　実験 4: K^i の降順にエージェント数を減少させたときの性能の変化

6.3　AMTDS/ERL についての実験結果

この節では, 提案手法である AMTDS/ERL と, 従来手法であり AMTDS/LD における評価実験結果を示す. 節 6.2 と同様に, 複数の条件下での実験結果を, 評価指標である $D(s)$ と $C(s)$, さらには, 必要に応じて他の指標も比較していく. なお,AMTDS/ERL はイベント発生確率の学習のため, エネルギー節約行動は 500,000 ステップから行っている.

6.3.1　実験 5: 性能評価

従来手法である AMTDS/LD と, 提案手法である AMTDS/ERL における,D(s) と C(s) の比較結果を図 14,15 に示す. また, 本実験での総ステップ数は 4,000,000 ステップであり, 要求値 D_{req} は 1,000, 補正係数 K^i の初期値は 0.5 である.

まず, 提案手法である AMTDS/ERL は,D(s) の値を従来手法である AMTDS/LD よりも増加させ,D_{req} に近づけられていることが分かる. それにより,C(s) の値を AMTDS/LD よりも減少させることができた. 具体的には,D(s) を D_{req} に近づけて約 82.4%増加させることにより,C(s) を約 42.2%削減した. また, 図 3 と比較すると,AMTDS/ERL の方が要求値により近づけられている. これは,5.2.2 で K^i の更新方法を変更したためである. つまり,AMTDS/ERL のエネルギー節約行動により, 環境全体の汚れを D_{req} 付近にしつつ, より省エネな巡回が行うことができた.

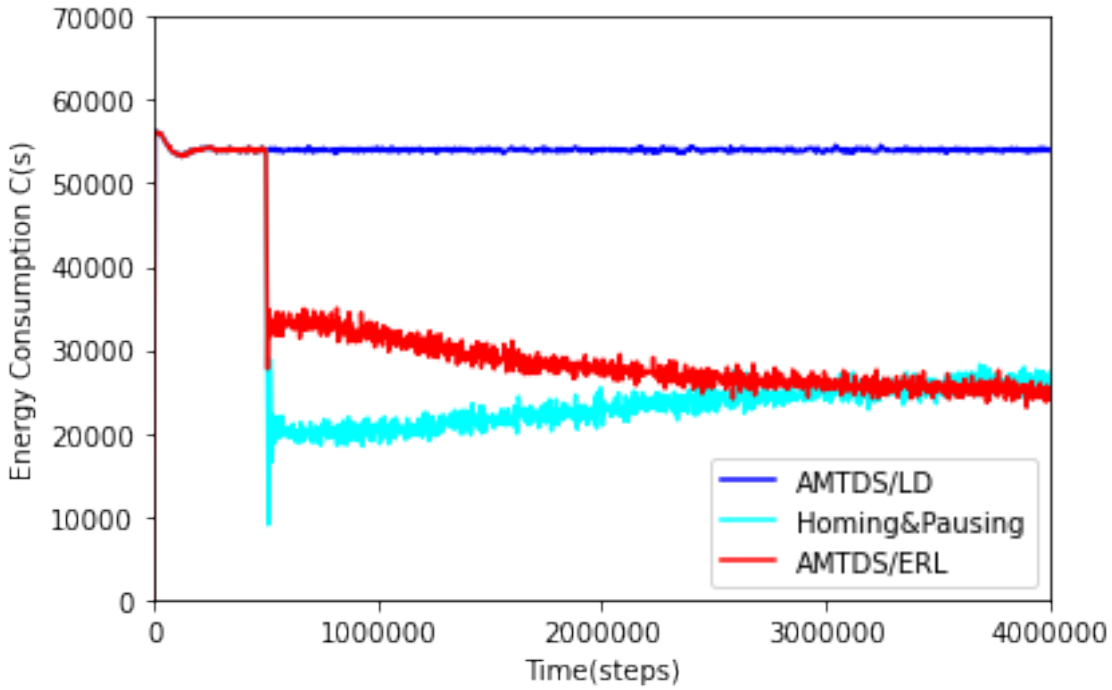

図 15: $C(s)$ の時間推移の比較 (実験 5)

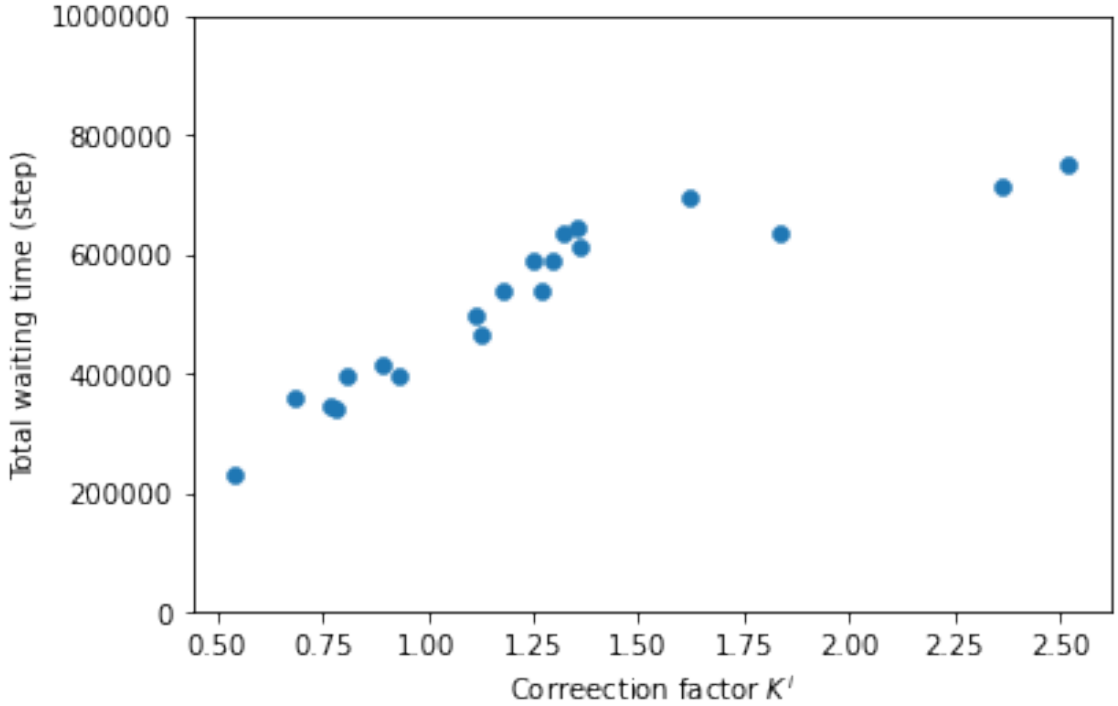

図 16: エージェントごとの K^i と総待機時間の関係 (実験 5)

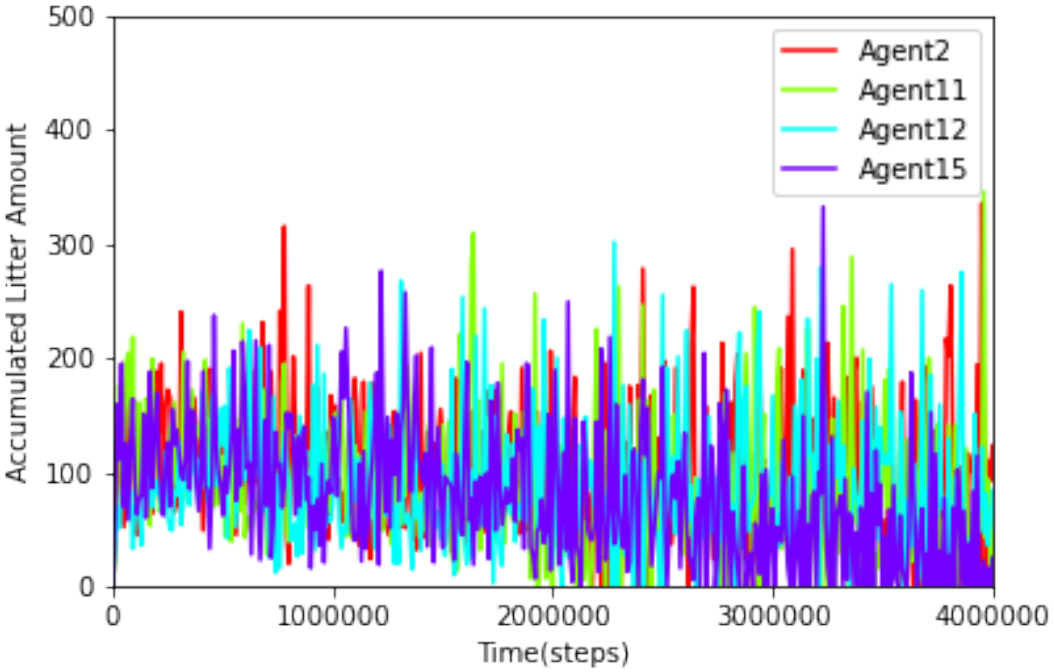

図 17: K^i 上位 4 体のエージェントによるイベント処理量の時間推移 (実験 5)

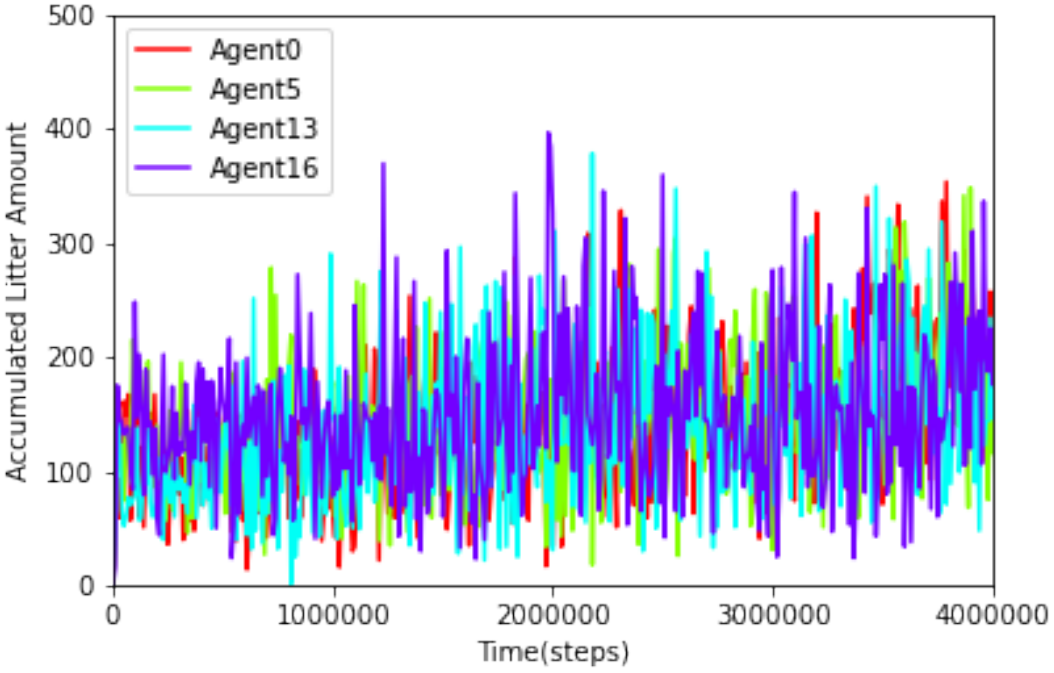

図 18: K^i 下位 4 体のエージェントによるイベント処理量の時間推移 (実験 5)

表 10: エージェント 0〜6 における K^i と総待機時間の比較 (実験 5)

エージェント	0	1	2	3	4	5	6
K^i	0.5414	1.2494	1.3542	1.1793	1.2972	0.7833	2.3616
総待機時間	228,000	586,400	643,800	538,500	587,700	339,800	712,300

表 11: エージェント 7〜13 における K^i と総待機時間の比較 (実験 5)

エージェント	7	8	9	10	11	12	13
K^i	0.8060	0.8948	1.8351	0.9304	1.6217	1.3595	0.7680
総待機時間	393,900	416,100	636,500	395,500	696,500	611,600	343,800

次に, 環境全体ではなく, エージェントごとについて見ていく. AMTDS/ERL も AMTDS/ER と同様に,$D(s)$ や $C(s)$ の変化は補正係数 K^i によって計算される Pausing による充電基地での待機時間の影響が大きい. また, これらは各エージェントによって異なるため, それに伴ってエージェントの行動も変わってくる. エージェントごとの K^i と Pausing による総待機時間をまとめたものが表 10,11,12 であり, これを散布図にまとめたものが図 16 である. なお,K^i は 4,000,000 ステップ時点の値で, 総待機時間は 3,000,000〜4,000,000 ステップでの総待機時間である. これらの表から, エージェントによって K^i は異なり, それによって総待機時間も異なっていて, 基本的に充電基地で待機しているエージェントと, あまり待機しないエージェントが存在していることがわかる. しかし, 図 5 と比較すると,K^i にしっかりとしたグループ分けのようなものは存在していない. これも,5.2.2 での K^i の更新方法を変更したためであると考えられる. また,AMTSD/ER はあらかじめノード v におけるイベント発生確率 $p(v)$ が与えられ, これが変更されることはないが,AMTDS/ERL は $p^i(v)$ の学習を行っているので, 巡回するたびに環境全体のイベント発生確率の予測値が変化するということも影響していると考えられる.

さらに, 各エージェントのイベント処理量の時間推移のうち,K^i が上位 4 体を図 17, 下位 4 体を図 18 に示す. まず, 図 17 より, 時間が経つにつれて K^i が大きくなったエージェントは, それに伴って充電基地での待機時間が長くなるため, 環境を巡回する時間が短くなり, イベント処理量がだんだんと少なくなっていくことが分かる. 逆に, 図 18 より,K^i が小さいエージェントはだんだんとイベント処理量が多くなっていることが分かる. これは, 複数のエージェントの充電基地での待機時間が長くなると, ノード v が任意のエージェントに訪問される間隔が長くなり, それによってイベントが溜まっていくので,K^i が小さいエージェントのイベント処理量がだんだん多くなっていると考えられる.

以上のことから,AMTDS/ERL は AMTDS/LD のように, 各ノードのイベント発生確率を学習しながら巡回し, イベント量を与えられた要求値付近にすることで, エネルギーを節約しながら巡回することができる手法といえる.

表 12: エージェント 14〜19 における K^i と総待機時間の比較 (実験 5)

エージェント	14	15	16	17	18	19
K^i	1.3230	2.5208	0.6828	1.1285	1.2685	1.1136
総待機時間	632,700	747,400	357,500	465,500	540,000	496,000

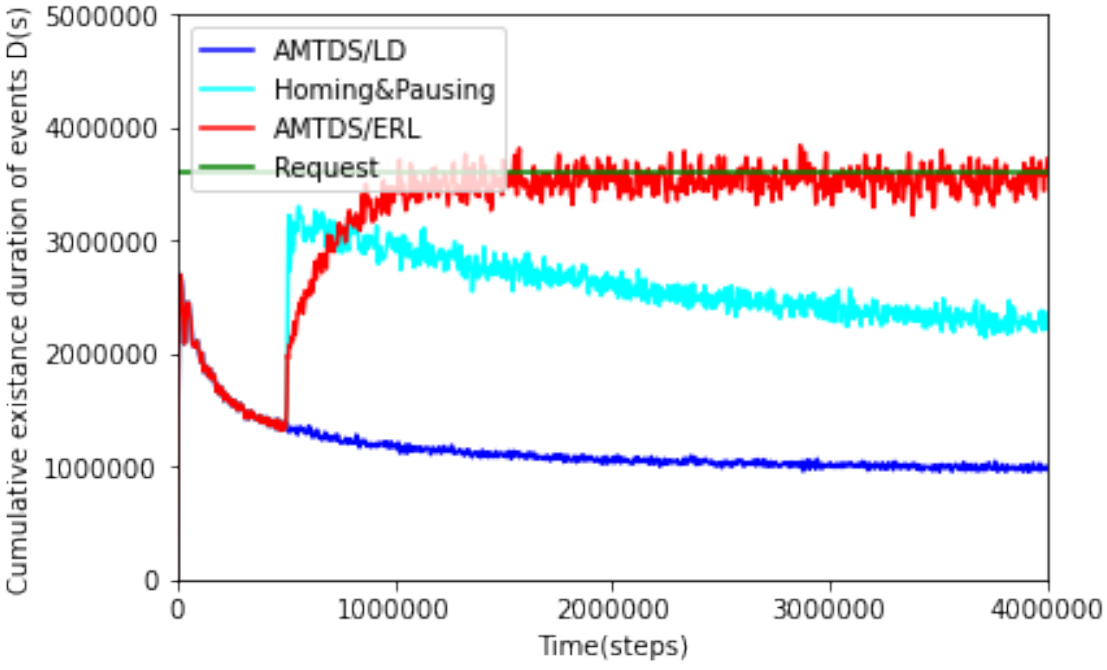

図 19: D(s) の時間推移の比較 (実験 6)

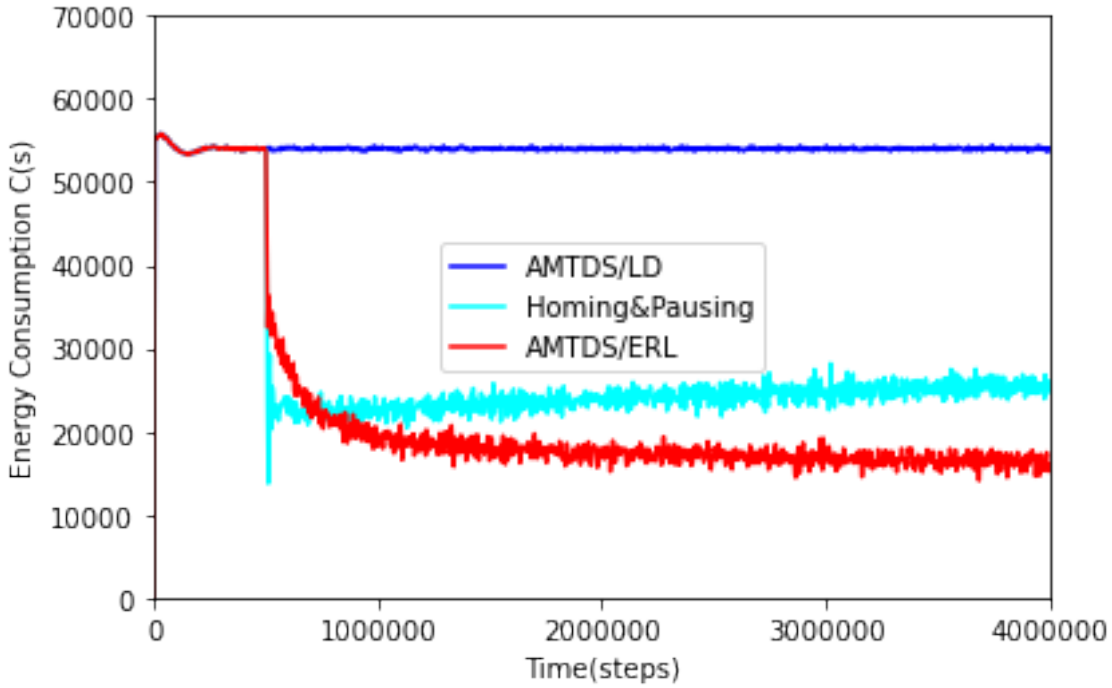

図 20: C(s) の時間推移の比較 (実験 6)

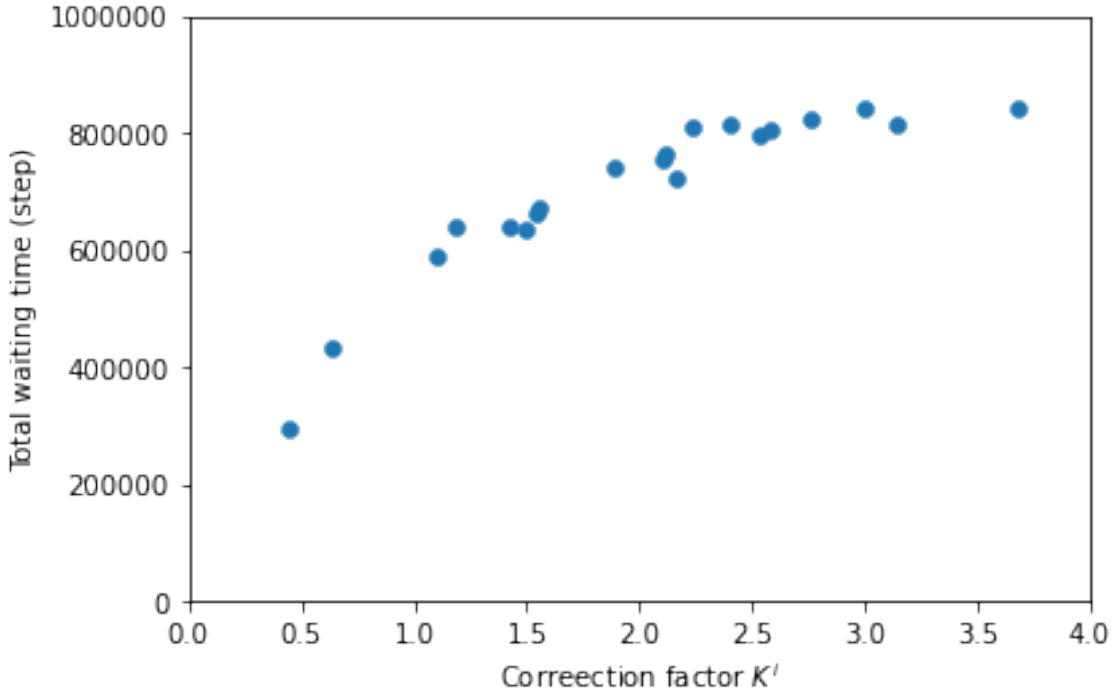

図 21: エージェントごとの K^i と総待機時間の関係 (実験 6)

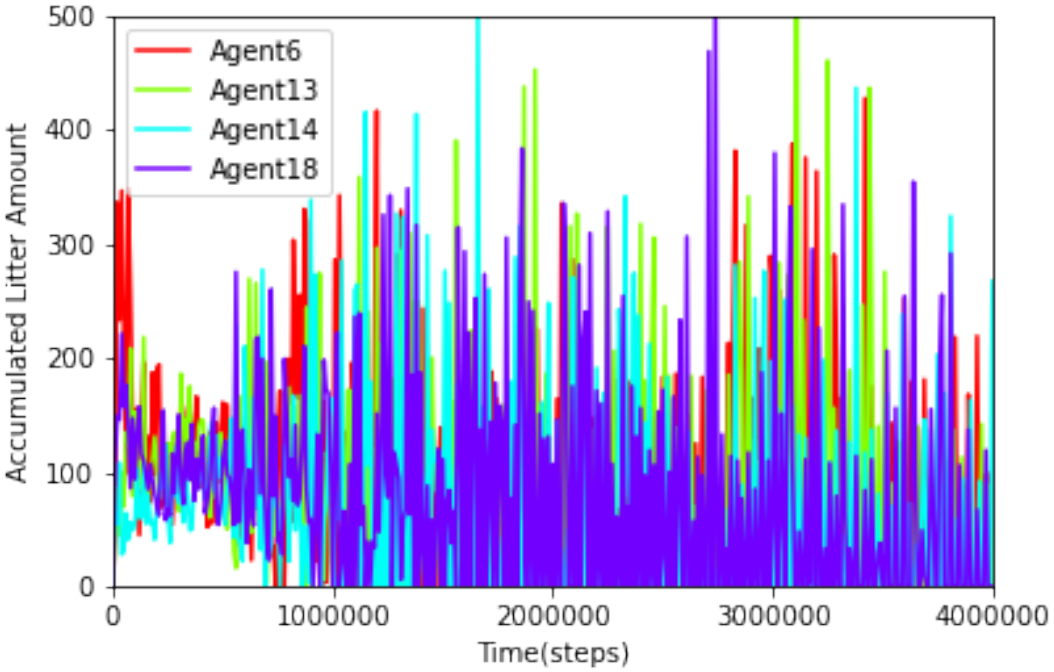

図 22: K^i 上位 4 体のエージェントによるイベント処理量の時間推移 (実験 6)

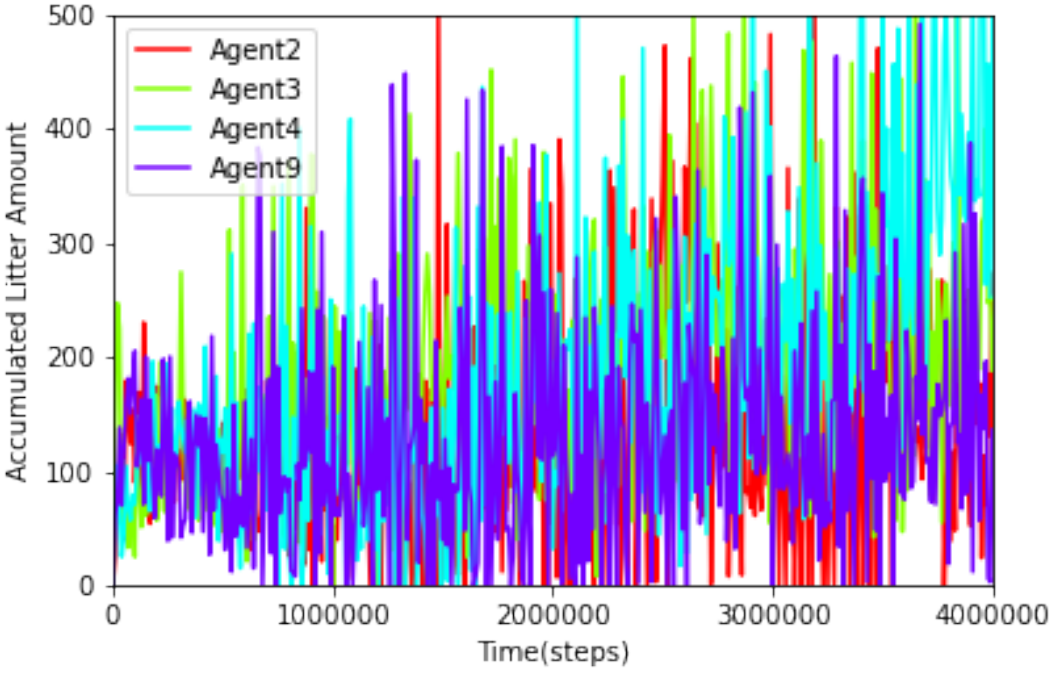

図 23: K^i 下位 4 体のエージェントによるイベント処理量の時間推移 (実験 6)

6.3.2 実験 6: 環境の変化による性能の違い

本実験では,6.2.2 と同様に, 図 8 の環境で実験を行った. また, 比較を容易にするために,6.3.1 と同様に, 本実験での総ステップ数は 4,000,000 ステップであり, 要求値 D_{req} は 1,000, 補正係数 K^i の初期値は 0.5 である.

まず, 提案手法である AMTDS/ERL は, 異なる環境においても,D(s) の値を従来手法である AMTDS/LD よりも増加させ,D_{req} に近づけ,C(s) の値を AMTDS よりも減少させることができた. 具体的には,D(s) を D_{req} に近づけて約 179.9%増加させることにより,C(s) を約 58.1%削減した. つまり,AMTDS/ER のエネルギー節約行動により, 環境全体の汚れを D_{req} 付近にしつつ, より省エネな巡回が行うことができた. また,6.3.1 と比べると, 環境全体のイベント発生量は小さくても要求値 D_{req} を同じであるので,D(s) の増加率が大きくなるため, それに伴い C(s) の減少率も大きくなった.

次に, 環境全体ではなく, エージェントごとについて見ていく. エージェントごとの K^i と Pausing による総待機時間をまとめたものが表 13,14,15 であり, これを散布図にまとめたものが図 21 である. なお,6.3.1 と同様に,K^i は 4,000,000 ステップ時点の値で, 総待機時間は 3,000,000〜4,000,000 ステップでの総待機時間である.

これらの表から, エージェントによって K^i は異なり, それによって総待機時間も異なっていることがわかる. また, 図 16 と比較すると, 全体的に充電基地での待機時間が大きくなっていることが分かる. これは, 本実験の環境である図 8 は図 2 と比べて環境全体のイベント発生確率が小さいため, 同じ要求値 D_{req} が与えられると,D(s) 要求値付近にするためには, 巡回を行わずに充電基地で待機する時間を長くしなければいけないが, 待機するエージェントと待機しないエージェントという役割分担はされていないので, 全体的に待機時間が増加したと考えられる.

さらに, 各エージェントのイベント処理量の時間推移のうち,K^i が上位 4 体を図 12, 下位 4 体を図 13 に示す. まず, 図 12 より, 時間が経つにつれて K^i が大きくなったエージェントは, それに伴って充電基地での待機時間が長くなるため, 環境を巡回する時間が短くなり, イベント処理量がだんだんと少なくなっていくことが分かる. 逆に, 図 13 より,K^i が小さいエージェントは, 充電基地で待機することもあるのでイベント処理量が 0 の時もあるが, だんだんとイベント処理量が多くなっていることが分かる. これは, 複数のエージェントの充電基地での待機時間が長くなると, ノード v が任意のエージェントに訪問される間隔が長くなり, それによってイベントが溜まっていくので,K^i が小さいエージェントのイベント処理量がだんだん多くなっていると考えられる.

以上のことから,AMTDS/ERL は, 環境が異なっても, 充電基地での待機時間を変化させることによってイベント量を要求値付近にして, エネルギーを節約しながら巡回することができる手法といえる.

6.3.3 実験 7: エージェント数減少による性能の変化

6.3.4 実験 8: K^i の降順にエージェント数を減少させたときの性能の変化

7 結論

表 13: エージェント 0〜6 における K^i と総待機時間の比較 (実験 6)

エージェント	0	1	2	3	4	5	6
K^i	2.1088	2.3994	1.1821	0.6392	0.4433	1.4969	2.7605
総待機時間	752,000	815,300	639,100	430,300	294,800	636,900	824,800

表 14: エージェント 7〜13 における K^i と総待機時間の比較 (実験 6)

エージェント	7	8	9	10	11	12	13
K^i	1.8867	2.5781	1.0963	2.1660	2.5299	2.2422	3.1407
総待機時間	738,600	806,500	589,200	723,400	796,800	809,500	813,600

表 15: エージェント 14〜19 における K^i と総待機時間の比較 (実験 6)

エージェント	14	15	16	17	18	19
K^i	2.9975	2.1116	1.5375	1.4290	3.6800	1.5534
総待機時間	841,600	763,000	660,400	637,800	839,700	669,600

参考文献

[1] 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.

[2] 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.

[3] 杉山歩未, 菅原俊治. 動的環境におけるマルチエージェント巡回清掃の自律的な戦略学習の提案. 電子情報通信学会論文誌 D, Vol. 98, No. 6, pp. 862–872, 2015.