資材搬送問題におけるロボットの動作遅延に対応したマルチエージェント自律分散アルゴリズムの提案

宮下　裕貴　山内　智貴　菅原　俊治
（技術研究所）　（早稲田大学）　（早稲田大学）

Distributed Action Planning with fluctuated robot's movement for Multi-agent Pickup and delivery problem

Yuki Miyashita, Tomoki Yamauchi and Toshiharu Sugawara

複数ロボットが資材を繰り返し運搬するマルチエージェント資材搬送問題は様々な分野に適用できるが、従来の多くの研究は自動倉庫での利用を想定し、グリッド状の環境を動作遅延なく同期して動くロボットを前提とした。一方、我々がターゲットとする建設現場や災害復旧現場のような環境は、資材置き場の数と迂回路の数が少ないため資材置き場付近で混雑が発生しやすい。加えて路面の凹凸や気象状況など外界の影響によりロボットの動作遅延が生じやすい。そこで、ロボットの動作遅延と混雑が発生しうる環境で、ロボットの衝突やデッドロックを発生させない非同期分散プランニングとその実行手法を提案する。

We propose a distributed planning method with asynchronous execution for multi-agent pickup and delivery (MAPD) problems in environments with occasional delays in agent activities. MAPD is a critical problem framework with numerous applications, yet most existing studies assume ideal conditions—such as fixed agent speeds, synchronized movements. Our proposed method relaxes some of these constraints, allowing for variable agent speeds and flexible endpoint locations, making MAPD applicable in more realistic environments. Experimental results show that our method enables agents to perform MAPD tasks efficiently in such conditions.

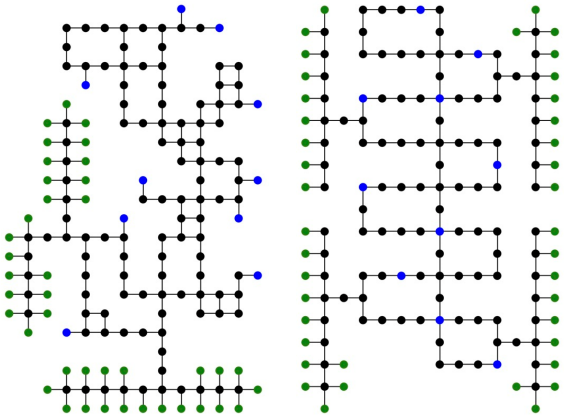
1. はじめに

複数の制御対象の協調動作を実現させるマルチエージェントシステム（multi-agent systems, MAS）がロボット制御技術の発展に伴い注目を集めている。例えば、倉庫における複数の搬送ロボットを用いた自律搬送システム [1)-3)]、複数の掃除ロボットを用いた自動清掃システムなど、複数ロボットを用いたアプリケーションの実用化が進展する中、MAS の研究はこれらシステムの基盤として寄与すると考える。これらシステムは、複数のロボット（エージェント）がそれぞれの目的地まで衝突無しに移動する multi-agent path-finding (MAPF) 問題と一般化され、エージェントの移動効率化、衝突やデッドロック回避が課題に位置付けられる．特に、倉庫における自律搬送システムでは、エージェントは指定の場所での資材の積込み/積下ろし動作を繰り返すため、MAPF 問題の繰り返しである multi-agent pickup and delivery (MAPD) 問題 [4)] と定式化される。

MAPD 問題では環境の複雑性や制約の増加により、資材搬送の効率低下、他エージェントとの衝突や頻繁なリソース競合が想定される。さらに実アプリケーションでは、エージェント動作の意図せぬ遅延が考えられ、あるエージェントの動作遅延が他エージェントの動作計画に影響しうる。このような状況に対処するには、遅延を考慮した新たな動作計画を再生成し、それを実行する必要がある。しかし、集中制御手法や、エージェントの同期動作を必要とする準分散手法では、エージェント数の増加に伴い動作計画の作成コストが指数関数に沿って増加する。さらに我々が想定する建設現場での資材搬送システムでは、天候などの影響によりロボット動作の意図せぬ遅延が十分に考えられ、動作遅延自体の解消は困難である。そのためエージェントの行動も非同期ながら個々に動作計画を作成・実行することで高い並列性を実現し、エージェントの動作遅延の影響を近接エージェントのみとの交渉によって解決する自律分散手法 [5)] が望ましい。

そこで我々は、始めに建設現場の環境を抽象化した MAPD with fluctuated movement speed (MAPDFS) 問題を提案する。MAPDFS 問題では、建設現場において運搬ロボットが建設資材を作業箇所に搬送することを想定し、資材搬送の目的地がエージェント数より少なく、さらに運搬ロボット動作には遅延などの不確実性を生じさせる。次に、MAPDFD 問題に対応した新たな自律分散型の動作計画・実行アルゴリズムを提案する。本手法は、複数の 2 重連結無向グラフからなる主領域とそれに接続したいくつかの木構造の無向グラフからなる周辺領域で構成される環境で、他エージェントのパスを考慮せず事前の通信や共有情報を必要としない低コストのパス生成アルゴリズムを使用し、事前衝突検知を用いてエージェントの衝突を防ぎ、衝突を解消するパスを低コストアルゴリズムで再生成する。

これを実現するため、資材を目的地に運ぶキャリアエージェント (以下 C エージェント) と、グラフ内の各ノードを管理するノードエージェント (以下 N エージェント) の 2 種類のエージェントを導入する。C エージェントは、他の C エージェントの動きを考慮せず目的地へのパスを生成し、そのパスに沿った資材運搬を試みる。一方、N エージェントは、対応するノードにおける C エージェント間の資源競合、つまり衝突の可能性を隣接する N エージェントとの通信より事前検知する。C エージェントはパスを生成後、現在滞在するノードの N エージェントとのみ通信し、そこを介して次の移動先のノードの予約を依頼する。依頼を受けた N エージェントは近隣の N エージェントと通信し、C エージェントの移動可否を確認する。近隣ノードからの返答に基づき、N エージェントは次の移動先ノードへの移動可否を C エージェントに伝え、移動できない場合は隣接する他ノードへの移動 (迂回)、もしくは滞在中のノードでの待機を提案する。返答を受け取った C エージェントは、返答に従って他の C エージェントを考慮することなく非同期に隣接ノードへの移動、もしくは現在のノードに待機する。N エージェントより迂回を提案されたときは、迂回先のノードに移動した後に目的地までの新たなパスを再生成する。低コストのパス生成アルゴリズムのみで自律分散下での N エージェントの衝突とデットロックを回避するために、2 重連結グラフからなる主領域のエッジにそれが強連結となるように向きを導入する。N エージェントが有向エッジに従ってパスを生成することで、エッジ上での衝突とデッドロックを解消する。有向エッジの導入は N エージェントに遠回りを強いるが、パス生成と衝突検知の単純化、C エージェントの動作遅延と現在ノードへの待機の許容と、資材搬送する目的地付近の混雑解消の利点を享受できる。実環境に則した実験設定のもと、従来手法との比較から提案手法の性能を評価し、提案手法の効率性を示す。



(a) 環境 1　　(b) 環境 2

図－1　実験環境

2. 関連研究

MAPD 問題の解決に取り組む研究 [4),6)-9)] は多くあるが、これらの多くは自動倉庫のなどの特別に設計された環境を想定するためエージェント数より集配場所が多く存在し、さらに遅延なく同期動作するエージェントを前提とする。また、従来研究 [4),10)] では完全性を保証するため、他エージェントの移動を妨げることなく任意の有限時間で集配場所に滞在できるよう、他のエージェントの目的地を横切らない目的地までのパスの存在を前提とする (well-formed infrastructure condition[6)]、以下、WFI 条件)。従来研究の前提は、自動倉庫環境のような単純なグリッド環境では容易に満たせるが、我々の想定する建設現場や災害現場は条件を満たすのが難しい。例えば、建設現場の運搬ロボットは、500kg の建設資材の運搬のため数台のエレベータと特定の資材保管場所を往復するため、自動倉庫に比べ運搬先の目的地の数が少ない。現場環境はアドホックでもあるため、特定の場所で局所的混雑が発生し、他のエージェントの目的地を横切らないパスの生成が困難か、存在しないこともある。よって従来研究 [4),10)] をそのまま建設現場に導入することは難しく、導入しても複数エー

ジェントの並列化が妨げられ非効率となる。そこで、本研究では、建設現場や災害現場での利用を想定した環境条件において、効率的な複数エージェントによる運搬を実現するアルゴリズムを提案する。

3. 準備

3.1 問題設定

MAPD 問題は C エージェント(以下、エージェント)の集合$I=\{1, \cdots, n\}$、**エージェントが処理するタスクの集合** $T=\{\tau_1, \cdots, \tau_{|T|}\}$と、2 次元ユークリッド空間に埋め込み可能な連結無向グラフ$G=(V, E)$から構成される。エージェントはノード$V=\{v_1, \cdots, v_{|V|}\}$上に存在でき、隣接するノード間を繋ぐエッジ$E$に沿って隣接ノードへ移動できる。以下グラフには有向エッジと無向エッジの 2 種類が存在し、v_jとv_kのエッジが無向エッジの場合は$(v_j, v_k) \in E$、有向エッジの場合は$(v_j \rightarrow v_k) \in E$と表現する。つまり、$(v_j \rightarrow v_k)$においてエージェントは$v_j$から$v_k$への移動のみ許される。本環境では、エージェントはエッジ上で止まらないとし、エージェント間の衝突は、あるノードに複数エージェントが同時に存在するか、あるエッジを両方向からすれ違うことを想定する。

タスク$\tau_k \in T$を$\tau_k=(v^l_k, v^u_k, \mu_k)$と表現し、$v^l_k$は資材$\mu_k$を積込むノード、$v^u_k$は$\mu_k$を積下ろすノードとする。資材の積込み・積下ろしノードタスクエンドポイントと呼ぶ。離散時間 $t \in N$(Nは自然数の集合)を導入し、その単位をステップとする。ある時刻 t でタスク$\tau_k \in T$を割り当てられたエージェント$i \in I$は、iが現在滞在するノード$v^i_c(t)$からv^l_kまでのパスを生成し移動を始める。v^l_kに到着し資材の積込みの後、iはv^l_kからv^u_kへのパスを生成し移動を始める。パスに沿った移動中に衝突を事前検知した場合、iはパスを修正し衝突を解消する必要がある。

始めに($t=0$)、エージェントiは駐車ノードv^{v_i} $(=v^i_c(0))$に配置され、割り当てられたタスクの実行を開始する。エージェントは隣接ノードに T_{mv} ステップで移動し、T_{lu} ステップで資材を積込む/積下ろす。また、エージェントは滞在中のノードに待機もできる。我々が提案する MAPDFS 問題ではエージェントの意図しない動作遅延を想定している。そのためエージェントの隣接ノードへの移動に、偶に T_{mv} ステップより長い時間を要することもある。

3.2 環境条件

本研究では、2 重連結グラフ$G_k=(V_k, E_k)$の集合から構成される主領域$G_{main}=G_1 \cup \cdots \cup G_k$にいくつかの木構造グラフを追加した環境を想定する。ここでG_kはG_{main}の最大 2 重連結要素とする。木構造グラフG^L_{tree}で構成される部分グラフの集合を $G^1_{tree}, .., G^L_{tree}$ で表す。本研究では、以下 4 つの条件を前提とする。

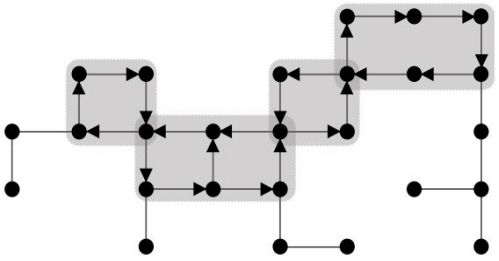
[条件.1] 主領域G_{main}は連結無向グラフである。なお、このとき 2 つの 2 重連結グラフ $G_k, G_{k'}$が$V_k \cap V_{k'} \neq \emptyset$のとき、$V_k \cap V_k$は単集合となる。

[条件.2] 主領域に含まれないノード($\forall v \in V \setminus V_{main}$)は、木構造グラフの集合 $G^1_{tree}, .., G^L_{tree}$に含まれる。ここで、$G^k_{tree} \cap G_{main}$は常に単集合とし、そのノードは G^k_{tree}の根(ルート)である。

[条件.3] 駐車ノードは木構造グラフの末端の節(葉)に配置され、駐車ノードを持つ木グラフにはタスクエンドポイントはない。

[条件.4] エージェント数は、G_{main}のノード数より 2 つ以上小さい($|I| \leq |G_{main}| - 2$)

接続ノードの集合を$G_{root}=(G^1_{tree} \cup \cdots \cup G^L_{tree}) \cap G_{main}$とし、主領域$G_{main}$外の周辺ノードの集合を$G_{mar}=G^1_{tree} \cup \cdots \cup G^L_{tree} \setminus G_{main}$と表す。**図－1** にこれら条件を満たす環境例を示す。緑色の丸は駐車ノード、青色の丸はタスクエンドポイントを表す。なおタスクエンドポイントは条件 3 に従いグラフG_{main}内のノードにも配置できる。



図－2 有向グラフの例

4. 提案手法

4.1 有向グラフの形成

我々の提案手法では、主領域のエッジに有向性をもたせ、エージェントがエッジ上で両方向からのすれ違いを防ぐ。このとき、主領域において移動不可なノードが存在しないよう、任意の 2 つのノード間を到達可能なパスが必ず存在するよう有向エッジの方向を定めることができる(one-way street theorem)グラフ理論のアルゴリズム[11)-13)]の 1 つを

使い、主領域の二重連結グラフ$G_{main}=G_1 \cup \cdots \cup G_k$のエッジに向きを決める。図－2に有向グラフの例を示す。灰色の領域が二重連結グラフで構成される主領域であり、主領域内のエッジのみ方向を有する。向き付けにより、エージェントには回り道を強いるが、主領域内でエッジ上での衝突を防げるため、複数エージェントの同じノードへの滞在を防げれば主領域での衝突を解消でき、さらにエージェントの動作の遅延にも対応できる利点がある。複数エージェントの同一ノードへの同時滞在を防ぐ仕組みは、Nエージェントとの通信により提供される。詳細は後述する。強連結性から任意の2ノード間を到達可能なパスが必ず存在し、主領域を循環する有向エッジが形成され、他エージェントのパスを考慮しないパス生成を可能とし、パス生成のコストを削減できる。

4.2 各エージェントの振る舞い

タスク$\tau_k = (v^l_k, v^u_k, \mu_k)$がCエージェント$i$に割当てられると、$i$は現在地から$v^l_k$に向かい資材$\mu_k$を積み、次に$v^u_k$に資材運んで積下ろす。そのため$i$はタスクの進捗に応じて目的地ノード$v_{dst} \in V$を定め、現在のノード$v^i_c$から$v_{dst}$までの最短距離のパス$p$を生成する。このとき、従来のMAPD問題と異なり、時間を考慮しない位置空間のみの経路探索手法（例えば、単純なA*-search）でパスpを生成する。ここで、目的地までのパスをノードの集合$p=(v^p_0, v^p_1, .., v^p_{|p|})$で表し、$v^p_{k-1}$と$v^p_k$は無向エッジ$(v^p_{k-1}, v^p_k)$か有向エッジ$(v^p_{k-1} \rightarrow v^p_k)$で連結しているとする。$i$は$p_i$を生成後、$p_i$に従って目的地への移動を試みる。なお、$i$のノードを管理するNエージェントをファシリテータと呼び、iが主領域にいる場合には、滞在しているノードのNエージェントをファシリテータとし、iが木構造グラフ内にいる場合には、そのルートのNエージェントをファシリテータとする。

時刻tのiの滞在ノードを$v^i_c(t)$と表す。エージェントiは、隣接ノードへの移動可否を判断するiのファシリテータ$v^i_{fcl}(t)$に、隣接ノードへの移動の可否を確認する。$v^i_c(t) \in V_{main}$の場合、つまりiが主領域にあるとき、$v^i_{fcl}(t) = v^i_c(t)$とする。一方、iが木構造グラフに滞在するとき（$v^i_c(t) \in G^K_{tree}$）、ファシリテータは木構造グラフのルートノードとなる（$v^i_{fcl}(t) \in G^K_{tree} \cap G_{main}$）。$v^i_c(t)$と次の移動先ノード$v^i_{next} \in p_i$が主領域に属さないとき、$i$はファシリテータに対して移動可否の確認を取らずに移動する。それ以外（$v^i_c(t) \in V_{main}$ もしくは $v^i_{next} \in V_{main}$）の場合、iは$v^i_{next} \in p_i$の滞在予約をするため、$v^i_{next} \in p_i$へ移動する前にv^i_{fcl}へリクエストメッセージとパスp_iを送り、その後、v^i_{fcl}の返答を待つ。v^i_{fcl}から了承メッセージを受け取った場合、iは時刻$t+1$でv^i_{next}への移動を開始し、$v^i_c(t)$の予約を解除する。このとき、iがv^i_{next}に到着するまで$T_{mv} \geq 1$ステップかかるが、iがv^i_{next}を離れるまで$v^i_c(t+1) = v^i_{next}$と想定する。移動先のノードに到着後、iはp_iに従って現在ノードから次の移動先ノードv^i_{next}への予約を試みる。

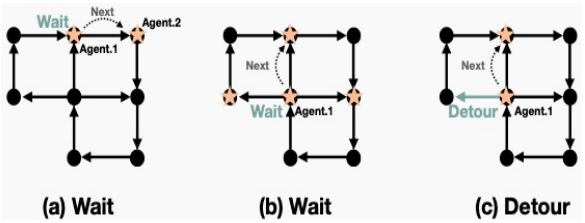
一方v^i_{fcl}から拒否メッセージを受信すると、iは拒否メッセージに含まれる動作提案(suggestion of movement、以下SOM)に従い計画を変更する。SOMには、$v^i_c(t)$での待機を提案するWAITと、$v^i_c(t)$に隣接する主領域内の他ノードへの迂回を提案するDETOURがある。DETOURを受け取るとiはDETOURで指定された隣接ノードへ移動を始め、他エージェントの動作を考慮せずに移動先から目的地までの最短迂回パスp_iを再作成する。WAITを受け取ったとき、iは$t+1$ステップまで$v^i_c(t)$に待機し、その後v^i_{fcl}にもう一度リクエストメッセージとp_iを送る。

4.3 Nエージェントによる衝突検知

主領域内のNエージェント$v(t) \in V_{main}$は、vに滞在するエージェントiからvと次のノードの滞在予約を管理する。vが滞在予約のためのリクエストメッセージをiから受け取ったとき、vはiの移動先となる次ノードへの移動の可否を確認するため、その近隣ノードと通信し衝突の可能性を確認する。このとき、2体以上のエージェントi、$j \in I$による同一ノードへの予約の重なりを確認する。また、Nエージェントはメッセージキューの先頭から順にメッセージを読むこととする。

ノードvに滞在するエージェントiから主領域内の近接ノード$v^i_{next} \in V_{main}$への移動依頼を受け取ったNエージェントvは、v^i_{next}に予約メッセージを送信し予約の可否を確認する。もしv^i_{next}が$t+1$ステップにおいて他エージェントからの予約を受けていない場合、v^i_{next}は$t+1$以降のiの予約を受け入れvに承諾メッセージを送る。その後、承諾メッセージはvを通してiにも転送される。もしv^i_{next}が$t+1$ステップですでに他エージェントに予約されていると、v^i_{next}は拒否メッセージをvに返信し、vは受け取った拒否メッセージと以下にように定めたSOMをiに送る。

DETOUR： vが複数の外向けの有向エッジを有する場合、vはv^i_{next}を除く有向エッジが繋がった主領域内の近接ノード群に順に予約メッセージを送る。近



図－3　SOM の例

接ノード群のあるノードv^i_{next*}から了承メッセージの返答を受け取ると、vは拒否メッセージと迂回先ノードv^i_{next*}を内包した DETOUR を i に送信する。i の移動のためv^i_{next*}の $t+1$ の滞在は予約される。

WAIT: DETOUR に失敗するとvは i の滞在を $t+1$ まで延ばし、現在滞在しているノード$v^i_c(t)=v$に待機することを提案する。i が次ノードを確保するまでvは他エージェントからの予約を承諾しないため、i のvでの待機は常に可能となる。

WAIT と DETOUR を送る例を図－3 に示す。図－3 内の(a)、(b)で Agent1 が次ノードの予約を依頼するが、次ノードは他エージェントにすでに予約され、 Agent1 のファシリテータの隣接ノードにも移動可能なノードがないため、ファシリテータは Agent1 に WAIT を送る。一方、図－3 (c)では Agent1 のファシリテータの近隣ノードに移動可能なノードがあり、Agent1 に DETOUR を送る。DETOUR を受け取った Agent1 は行き先ノードを変更し、変更先ノードへの移動を試みる。

4.4　主領域外の衝突検知

主領域外の衝突検知について、まず駐車ノードを含まない木構造グラフを考える。木構造グラフ G^k_{tree} のルートとなる N エージェント $v \in V_{root} \subset V_{main}$ は、その木構造グラフのノード群 $V^k_{tree} \setminus \{v\}$に入るエージェントの数を 1 体に制限するように、その木構造グラフへの入退場を管理する。例えば、vに滞在するエージェント i が木構造グラフ G^k_{tree} 内のノード $v^i_{next} \in V^k_{tree}$ を次ノードとするとき、まず i はvにリクエストメッセージを送る。ここで $G^k_{tree} \cup G_{mar}$ に他エージェントがいなければ、vは i に了承メッセージを送る。一方、他エージェントがすでに $G^k_{tree} \cup G_{mar}$ にいれば、vは i に拒否メッセージを送る。ここでも同様に、迂回先となるノードがあれば DETOUR を、迂回先となるノードがなければ WAIT を拒否メッセージと共に送る。

一方、$v^i_{next}=v$かつ$v^i_c(t) \in V^k_{tree} \cap V_{mar}$の場合、$i$ はファシリテータ v $(= v^i_{fcl} = v^i_{next})$に$v^i_{next}$へ移動するためのリクエストメッセージを送る。ここで、$t+1$ における i のvでの滞在が可能な場合、vは了承メッセージを i に送り、可能でない場合は拒否メッセージと共に WAIT を送る。$v^i_{next} \notin V_{main}$かつ$v^i_c \in G^i_{tree} \cap G_{mar}$の場合、$i$ はファシリテータにリクエストメッセージを送らずv^i_{next}に移動する。

G^k_{tree}が駐車ノードを含む場合、MAPDFS インスタンスの始まりに G^k_{tree} のルートノード$v \in V_{root}$は G^k_{tree} から主領域への移動のみを許し、その後間もなく、G^k_{tree} 内部への移動のみを許す。つまり、エージェントは駐車ノードに主領域から戻ると、その後の主領域への移動は制限される。

5. 実験・議論

5.1　実験設定

MAPDFS 問題における提案手法の性能を評価するため、提案手法の求める環境条件を満たす 2 つの環境(図－1)で、既存手法との性能の比較実験をした。環境 1 では、木構造グラフの葉に 10 個のタスクエンドポイントを配置した。さらに既存手法と性能を比較するため、既存手法の環境条件である WFI 条件を満たす環境とした。環境 2 も同様に 10 個のタスクエンドポイントが存在するが、主領域内に配置されるため WFI 条件を満たさず、比較手法は適用できない。環境 2 ではエージェントが資材の積下ろしを実行中、他のエージェントの移動はそれらの動作により妨げられ、積下ろしの完了を待つ状況が生じる。WFI 条件を満たさない環境 2 は建設現場での自動搬送において十分に想定されるため、このような環境での提案手法の性能も評価する。

既存手法に holding task endpoint(HTE) [4]と rolling-horizon collision resolution(RHCR) [7]を用いて提案手法との性能を比較する。HTE は、タスクやエージェントの動作計画の情報を含む同期した共有メモリブロックを用いて、順に参照しながらそれぞれのエージェントが衝突の無い動作計画を生成する。また、HTE は WFI 条件を満たす必要がある。RHCR では、MAPD 問題を「繰り返し windowed MAPF 問題」と捉え、エージェントは w ステップ

表－1　実験に使用するパラメータ値

説明	パラメータ	値
次ノードへの移動時間	T_{mv}	3
積込み/積下ろし時間	T_{lu}	3 or 6
動作遅延発生確率	ν	0 to 0.2
動作遅延による追加時間	T_{nse}	1 or 2

表－2　実験 1 におけるインスタンス成功割合

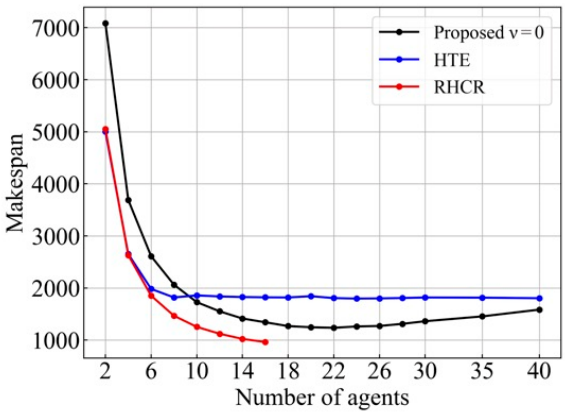
Alg.	Number of agents																
	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	35	40
提案手法 $\nu = 0$	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
HTE	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
RHCR	1.0	0.92	0.86	0.78	0.52	0.28	0.18	0.02	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

先までの衝突を解消した動作計画を生成し、h ステップごとに全エージェントの動作計画を再生成する。RHCR は HTE と違い WFI 条件を必要としないが、RHCR の衝突の解消は不完全である。また、本実験では RHCR の MAPF ソルバに priority-based search[14]を用い、RHCR のハイパーパラメータを $w=60$ と h=15 に設定した。また提案手法における低コストのパス生成には、A*アルゴリズムを採用した。

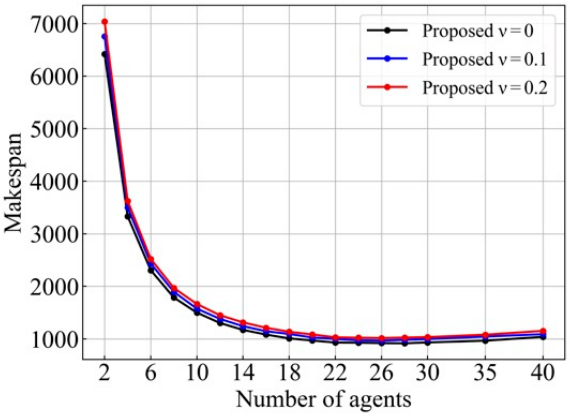
実験 1 では、WFI 条件を満たす環境 1 で提案手法と 2 つの既存手法の性能比較を行う。このとき、既存手法は動作遅延に対応できないため、エージェントの移動速度は一定とする。実験 2 では、WFI 条件を満たさずエージェントの移動速度に遅延が発生する環境 2 において、提案手法が全てのタスクを処理できることを確認する。実験 1 では資材積下ろし時間を T_{lu}=3、実験 2 では T_{lu}=6 と設定し、実験 2 では他エージェントの動作をより長く妨げる。手法の評価として、インスタンスの成功割合、全タスクの完了に必要な時間（makespan）、合計 CPU 時間を用いる。

全エージェントは駐車ノードから開始する。タスクは、資材の積込み積下ろし位置をタスクエンドポイントからランダムに選び、生成される。エージェントがタスクを完了すると、新たなタスクが割り当てられ、予め決められた全タスクが完了するまで繰り返される。全てのタスクが完了し、タスクのないエージェントは初期位置の駐車ノードに戻る。

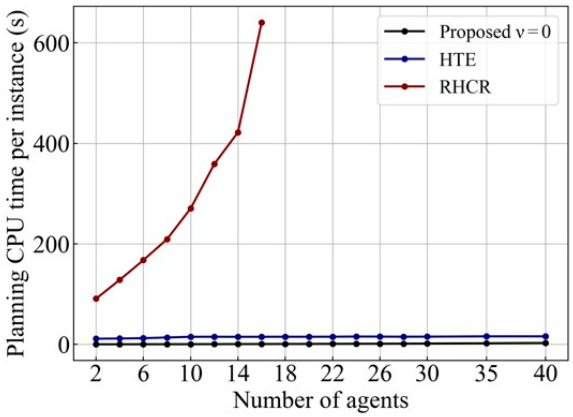
実験 2 ではエージェントが次ノードに移動するとき、移動に遅延を導入する。具体的には、通常は T_{mv}=3 ステップで次ノードに移動するが、確率 $\nu\ (\leq \nu \leq 1)$ の確率で次ノードへの移動に $T_{mv} + T_{nse}$ ステップを要する。T_{nse} はランダムに 1 か 2 とする。エージェントの数 n を 2 から 40、タスク数を $|T|$ =100 とした。表－1 にその他のパラメータ値を示す。以下の実験結果は異なる乱数シードによる 50 試行の平均より算出し、apple M1 Max CPU with 64 GB RAM を用いた。



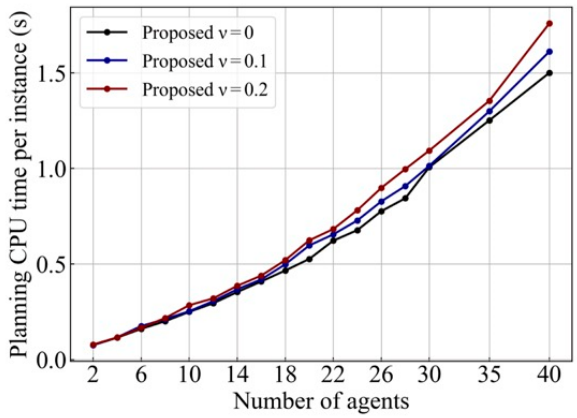
図－4　Makespans（実験 1）



図－5　Makespans（実験 2）



図－6　CPU の合計時間（実験 1）



図－7　CPU の合計時間（実験 2）

5.2　性能比較

5.2.1　MAPDFS インスタンスの成功割合

表－2 に、実験 1 におけるエージェント数ごとの MAPDFS インスタンスの成功割合を示す。ここで、上限ステップ（10000 ステップ）の超過、衝突の発生、もしくは衝突を解消するパスを見つけられない場合を失敗とした。表－2 より、提案手法は全てのインスタンスで全タスクの処理に成功した。また、環境 1 は WFI 条件を満たすため HTE も全インスタンスを完了できた。一方、RHCR ではエージェント数の増加に従って成功割合が減少し、エージェント数が $n \geq 18$ の場合では成功割合が 0 になった。複数のエージェントが目的地に同じタスクエンドポイントを選択するとそのタスクエンドポイント付近で混雑が生じやすい。Prioritized planning を用いる RHCR では、このような混雑する状況下において衝突を回避するパスを生成できなかった。表には示さないが、実験 2 においても提案手法は全インスタンスに成功した。

5.2.2　makespan

実験 1 のエージェント数ごとの makespan の平均を図－4 に示す。ここで、失敗した MAPDFS インスタンスは makespan の算出から除外した。図－4 より、エージェント数が $n \geq 8$ で手法ごとに makespan に違いが表れ、提案手法が最も良い性能を示した。提案手法では、エージェント数が 22 を超えるまで makespan を徐々に減少させたが、$n \geq 24$ のとき makespans はわずかに上昇した。makespan の上昇は、エージェント数の増加によるタスクエンドポイント付近のエリアでの混雑が要因であった。一方、$2 \leq n \leq 6$ のとき、提案手法より既存手法が良い性能を示した。タスクエンドポイントの数よりエージェント数が少ないとき、既存手法は並列性を保った効率的な動作計画を生成できるが、一方で提案手法は有向グラフによりエージェントに遠回りを強要し、運搬効率に差が生じて makespan に影響を及ぼした。

一方 HTE の makespan は、$n \geq 8$ で一定であった。HTE は、エージェントは同じタスクエンドポイントを持つタスクを同時に実行できない。したがって、エージェント数がタスクエンドポイント数より多いとき並列実行するエージェント数を増やせず、性能は頭打ちになる。

RHCR は $n \geq 16$ では他手法を上回る性能を示したが、$n>16$ の場合にインスタンスを完了できず、$n>16$ の makespan を算出できなかった。表－2 より、$n=12$ でも RHCR の成功割合は 0.28 であり非常に低い。そのため、我々のアプリケーションへの RHCR の導入は難しい。

MAPDFS 問題において移動速度の変動が makespan に与える影響を調査するため、実験 2 の makespan の平均を図－5 に示す。図－5 より、移動ノイズの追加確率 ν の増加に従って性能は減少するが、これら性能差は軽微である。したがって、提案手法は移動速度の突発的な遅延に対して頑強と言える。

5.2.3　CPU 時間

実験 1 におけるインスタンスあたりの全エージェントの合計 CPU 時間の平均を図－6 に示す。図－6 より、エージェント数に関わらず提案手法の合計 CPU 時間は既存手法に比べて小さい。提案手法は、HTE のように他エージェントのパスを考慮せず、時間を考慮しない位置空間のみの経路探索を行うため、既存手法に比べて CPU 時間が小さくなった。

図－6 を注視すると、HTE では $n \geq 10$ のとき合計 CPU 時間が一定になる。これは並列実行するエージェントの数が限られたことが要因である。一方、提案手法ではエージェント数の増加に伴い並列実行するエージェント数が増加し、合計 CPU 時間も徐々に上昇する。しかし $n=40$ でも合計 CPU 時間は 2.78 秒で HTE より小さく、エージェント数の多いアプリケーションへの導入も可能なことを示唆する。RHCR は、$h=15$ ステップごとに全エージェントの動作計画を再計画するため、他手法と比べて非常に大きな合計 CPU 時間を要した。

図－7 に、実験 2 において移動ノイズの追加確率を変えた場合のインスタンス当たりの CPU 時間を示す。実験 1 と異なり、実験 2 の環境では他のエージェントによる遅延や積み下ろし動作により、同じノードに待機せざるを得ない場合が多い。そのため、エージェントが拒否メッセージを受け取る頻度が増

えるが、図－7より移動ノイズがCPU時間に与える影響は小さいことがわかった。

6. 考察

我々の提案手法は、建設現場で想定されうる環境条件において、非同期動作による分散制御手法を用いて、衝突やデッドロック状態に陥ることなく全タスクを完遂した。衝突やデッドロックが発生しなかったのは、エージェントの速度変動による遅延に関係なく、既に他のエージェントによって予約・滞在されている次のノードへのエージェントの移動をノードエージェントが防ぎ、対向方向での交差を防ぐための有向エッジが機能したためである。

さらに、提案手法は、エージェント数が10以上の際、タスクエンドポイントの数が少ない環境において比較手法(HTEおよびRHCR)を上回る性能を示した。提案手法がタスク実行の並列性を高め、混雑した領域においてもパフォーマンスの低下を軽減できることを示した。HTEでは、タスクエンドポイントの数がエージェント数よりも少ないため、稼働するエージェント数を増やすことができず、タスクの並列実行ができなかった。一方、RHCRは全エージェントでの同期した行動計画と行動実行の実現に、たびたび行動の再計画が必要となるため、計算コストが増大し、合理的な時間内にすべてのタスクを完了できなかった。

提案手法では、目的地までの経路を生成した後、エージェントはローカル通信を通じて次のノードの利用可能性を確認し、必要に応じて経路を修正しながら移動する。したがって、従来のローカルリペアA*[15]やその拡張アルゴリズム、RHCRのように、限定されたウィンドウサイズを持つローカルリペアアルゴリズムに類似している。これらアルゴリズムの欠点は、多くのエージェント特定のエリアに集中すると、衝突やライブロック状態が発生しやすくなり、混雑による高い計算コストを必要とする経路再計画が頻発する可能性がある。しかし、我々の提案手法はグラフに有向性を導入することで、エージェントが左右から来る他のエージェントに挟まれる状況を防ぐことを可能とした。さらに、エージェントが移動可能なノードに誘導されることで、混雑している場合でも、エージェントを周辺エリアに誘導することで、混雑を解消した。

7. むすび

本研究では、動作遅延が生じる自律搬送問題において効率的で頑強な非同期分散アルゴリズムを提案した。提案手法は単純な手法であり、エージェント数がタスクエンドポイントより小さく、エージェントの動作にノイズが生じる環境に有効である。実験より、MAPD問題において提案手法は既存手法を上回る性能を示した。また、既存手法が対応できないエージェントの動作にノイズが生じる環境においても、提案手法は衝突を生じさせず効率的に全てのタスクの処理に成功した。

今後は提案した手法の拡張を進める所存である。例えば、提案手法が求める環境条件の緩和、資材運搬の効率性を向上させる有向グラフの向きの提案や、エレベータと運搬ロボットの連携による複雑なタスクの処理の実現などを検討する。

＜参考文献＞

1) O. Salzman and R. Stern, “Research challenges and opportunities in multi-agent path finding and multi-agent pickup and delivery problems,” Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems, pp.1711-1715, 2020.
2) J.S. Jennings, G. Whelan, and W.F. Evans, “Cooperative search and rescue with a team of mobile robots,” 1997 8th International Conference on Advanced Robotics. Proceedings. ICAR’97, pp.193-200, 1997.
3) M.P˘echou˘cek, D. ˘ Si˘sl´ak, D. Pavl´ı˘cek. and M.Uller, “Autonomous agents for air-traffic deconfliction,” Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, pp.1498-1505, AAMAS ’06, Association for Computing Machinery, New York, NY, USA, 2006.
4) H. Ma, J. Li, T.S. Kumar, and S. Koenig, “Lifelong multi-agent path finding for online pickup and delivery tasks,” Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, pp.837–845, AAMAS ’17.
5) Z. Yan, N. Jouandeau, and A.A. Cherif, “A survey and analysis of multi-robot coordination,” International Journal of Advanced Robotic Systems, vol.10, no.12, p.399, 2013.
6) M. ˘C´ap, J.Vok˘r´ınek, and A. Kleiner, “Complete decentralized method for on-line multi-robot trajectory planning in well-formed infrastructures,” Proceedings of the Twenty-Fifth International Conference on

International Conference on Automated Planning and Scheduling, pp.324–332, ICAPS'15, AAAI Press, 2015.

7) J. Li, A. Tinka, S. Kiesel, J.W. Durham, T.K.S. Kumar, and S. Koenig, "Lifelong multi-agent path finding in large-scale warehouses," Proceedings of the AAAI Conference on Artificial Intelligence, vol.35, no.13, pp.11272–11281, May 2021.

8) M. Liu, H. Ma, J. Li, and S. Koenig, "Task and path planning for multi-agent pickup and delivery," Proceedings of the 18th International Conference on Autonomous Agents and Multi Agent Systems International Foundation for Autonomous Agents and Multiagent Systems, pp.1152–1160 2019.

9) T. Yamauchi, Y. Miyashita, and T. Sugawara, "Standby-based deadlock avoidance method for multi-agent pickup and delivery tasks," Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems, pp.1427–1435, AAMAS '22,

10) H. Ma, W. H¨onig, T.S. Kumar, N. Ayanian, and S. Koenig, "Lifelong path planning with kinematic constraints for multi-agent pickup and delivery," Proceedings of the AAAI Conference on Artificial Intelligence, vol.33, pp.7651–7658, 2019.

11) M.J. Atallah, "Parallel strong orientation of an undirected graph," Information Processing Letters, vol.18, no.1, pp.37–39, 1984.

12) E.W. Dijkstra, A Discipline of Programming., Prentice-Hall, 1976.

13) R. Tarjan, "Depth-first search and linear graph algorithms," SIAM Journal on Computing, vol.1, no.2, pp.146–160, 1972.

14) H. Ma, D. Harabor, P.J. Stuckey, J. Li, and S. Koenig, "Searching with consistent prioritization for multi-agent path finding," Proceedings of the AAAI Conference on Artificial Intelligence, vol.33, no.01, pp.7643–7650, Jul. 2019.

15) A. Zelinsky. 1992. A mobile robot exploration algorithm. IEEE Transactions on Robotics and Automation 8, 6 (1992), 707–717.