F-016

暫定的優先順位増加による複数エージェント経路計画アルゴリズムの拡張

奥村圭祐 *, 田村康将 *, Xavier Défago*

The problem that makes multiple agents move to their destinations without collisions is called Multi-agent Path Finding (MAPF), which is receiving a lot of attention due to its high practically, e.g., warehouse applications with autonomous vehicles. Priority Inheritance with Backtracking (PIBT) is a decoupled approach to solve the iterative MAPF problem. PIBT ensures that all agents reach their destinations in finite time when the environment is given as a biconnected graph. In this paper, we extend target graphs of PIBT by introducing temporary inflation of agent priorities and discuss the effectiveness of the approach.

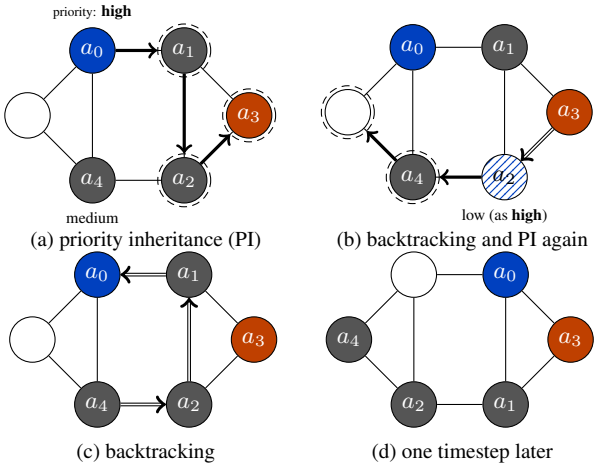
Key words: Iterative Multi-agent Path Finding, Priority Inheritance with Backtracking

1. はじめに

複数エージェントに, 衝突を回避しつつ効率の良い経路を与える問題は Multi-agent Path Finding (MAPF) と呼ばれる. MAPF はロボット群を利用した倉庫の荷物運搬 [1] への応用等から近年注目を浴びている. 最適な経路の組合せを与えることは NP 困難であり [2], 近似解法を含む多数のアルゴリズムが提案されてきた [3]. 実践的には, 連続的に発行される目的地に応じて, 膨大な数のエージェントを含む MAPF をリアルタイムに解き続ける必要がある. したがって, エージェント各々が自身の経路計画を行う自律分散的な手法が好ましく, 衝突回避のため, エージェントは局所的な通信を利用して経路の交渉を行う.

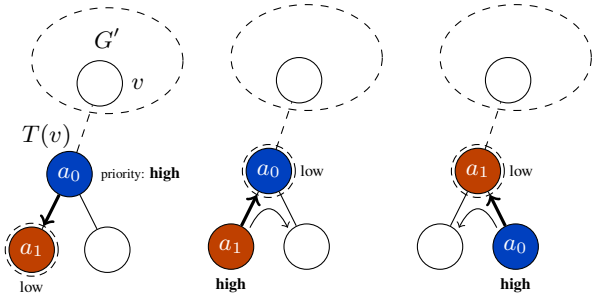
Okumura ら [4] はオペレーティングシステムにおける優先順位継承 (Priority Inheritance) [5] に着想を得て, 繰返し MAPF を求解するための準最適アルゴリズム Priority Inheritance with Backtracking (PIBT) を提案した. PIBT は限定された時間窓内での優先順位に基づいた経路計画 [6] に立脚している. 優先順位は各時刻でエージェントに動的に割当てられ, 優先順位の低いエージェント X が優先順位の高いエージェント Y の移動を妨げている場合, X は一時的に Y の優先順位を継承して経路計画を行う. 優先順位継承はその継承の妥当性を判断するバックトラッキングと組合わせて実行される. 動作例を Fig. **1** に示す. PIBT はグラフ上の距離 2 内のマルチホップ通信のみを仮定しており, 自律分散的な実装が可能である.

PIBT は二重連結グラフ等の環境で, エージェントが有限時間内に必ず目的地へ到達することを保証する. しかし, グラフに木構造が含まれる場合, エージェント同士がすれ違うことができないため, 優先順位継承では解決できない膠着状態に陥ってしまう (Fig **2**a). 本稿では, PIBT の対象グラフの拡張を目的として, トポロジに応じた暫定的な優先順位増加の導入を議論する.



(a) priority inheritance (PI) (b) backtracking and PI again (c) backtracking (d) one timestep later

Fig.**1**: Example of PIBT. Desired nodes for each agent are depicted by dashed circles. Flows of priority inheritance and backtracking are drawn as single-line and doubled-line arrows, respectively. As a result of consecutive priority inheritance ($a_0 \to a_1 \to a_2$), a_3 has no escape nodes (**1**a). To avoid collisions, a_0–a_2 have to wait backtracking before moving. In (**1**b), a_3 sends as invalid to a_2 then a_2 changes its target node (a_4). In this time, a_4 can successfully move and sends as valid to a_2. Similarly, a_1 and a_0 receive as valid (**1**c) and then they start moving (**1**d).



(a) deadlock (b) livelock: a_1 is high (c) livelock: a_0 is high

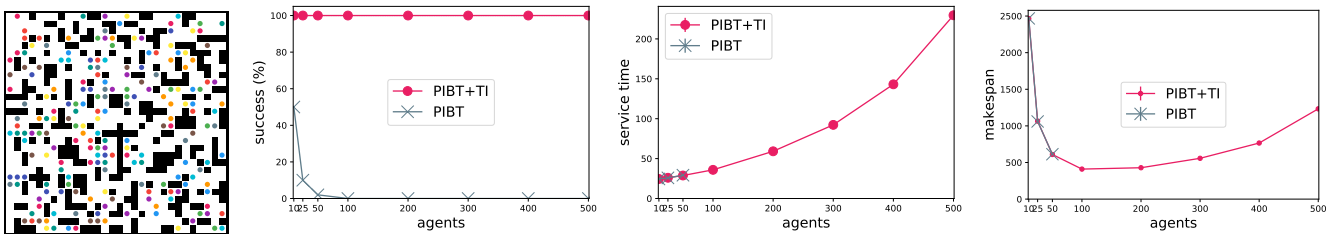
Fig.**2**: Example of deadlock/livelock situations.

2. 用語

エージェントを $a_i \in A$ と表す. 環境は有向グラフ $G(V,E)$ によって定義され, a_i の挙動は G に制限される. MAPF では G は強連結な単純グラフであり, これは連結な単純無向グラフを含む. 時刻 $t \in \mathbb{N}$ での a_i の位置を $v_i(t) \in V$ とする. $\forall t$ で a_i は $t{+}1$ でのノード $v_i(t{+}1) \in \{v | v \in V,\ (v_i(t), v) \in E\} \cup \{v_i(t)\}$ を選択し, 移動を行う. $v_i(t{+}1)$ の選択には $a_j, i \neq j$ に関して, 衝突を避けるため 1) $v_i(t{+}1) \neq v_j(t{+}1)$; 2) $v_i(t) \neq v_j(t{+}1) \vee v_i(t{+}1) \neq v_j(t)$; という制約が課される. a_i には目的地 $g_i(t) \in V,\ g_i(t) \neq v_i(t)$ が与えられる.

a_i の $t{+}1$ での最良ノードを $v_i^\star(t{+}1) \in V$ と表記する. ここで $v_i^\star(t{+}1) \neq v_i(t)$ であり, $\mathrm{cost}(v,u),\ v,u \in V$ を u から v までの最短経路長としたときに, $\mathrm{cost}(v_i^\star(t{+}1), g_i(t)) < \mathrm{cost}(v_i(t), g_i(t))$ を満たす.

* 東京工業大学 情報理工学院



(a) benchmark (b) success rate (c) service time (d) makespan

Fig.**3**: Experiments. **3**a) The map is 30×30 4-connected undirected graph, which is designed so that G' is connected. The figure contains 200 agents. **3**b–**3**d) Results. PIBT+TI (Temporary Inflation) is a proposal. Service time and makespan is average over only success cases.

G の部分グラフ $G'(V', E')$ を導入する.

$$E' = \{(v,u) \mid (v,u) \in E,\ \exists \mathbf{C} = (v,u,\dots),\ |\mathbf{C}| \geq 3\}$$
$$V' = \left\{v, u \mid (v,u) \in E'\right\}$$

$\mathbf{C}$ は単純閉路を表す. このとき, $\forall u \in V \setminus V'$ は $\exists v \in V'$ を根とする G の部分木 $T(v)$ に属することになり, $G' \cup \left\{\bigcup_v T(v)\right\}$ は G と等しい.

3. 提案手法

暫定的優先順位の増加

G' を用いて PIBT は以下の定理を与える [4].

定理 1 $G' = G$ ならば, PIBT は全エージェントに $\mathrm{diam}(G)|A|$ 時間以内に目的地へと到達する経路を与える.

以降 $G' \neq G$ であり, G' が連結である G を対象とする.

定理 1 は, $\forall v \in V'$ には閉路が存在しており, 最高優先順位のエージェントはその閉路に沿って移動することが常時可能であることに依っている. $v_i(t)$ に閉路が存在しない場合, 例えば $v_i(t)$ が $T(v)$ の葉に相当するとき, a_i より優先順位の高い a_j が $v_i(t)$ を $v_j(t{+}1)$ の候補として予約を試行すると a_i, a_j は膠着状態に陥る (Fig. **2**a). これの予防を目的として, PIBT の優先順位付けの拡張を次のように行う.

PIBT は a_i に固有の値 $\epsilon_i \in [0,1)$ と, 最後に g_i が更新されてからの経過時間 $\eta_i(t) \in \mathbb{N}$ を用いて, a_i の優先順位を $p_i(t) \leftarrow \eta_i(t) + \epsilon_i$ とする. これを拡張して $\boldsymbol{p}_i(t) = \left(p_i^1(t), p_i^2(t)\right)$ という優先順位のもとで PIBT を実行する. $p_i^1(t) \in \{0,1\}$ は暫定的な優先順位増加に対応し, 以下の条件を満たす時に 1, そうでない場合に 0 とする.

$$v_i(t) \in T(v) \ \wedge\ (\mathrm{cost}(v_i^\star(t+1), v) < \mathrm{cost}(v_i(t), v))$$

$p_i^2(t) \leftarrow \eta_i(t) + \epsilon_i$ として, 優先順位の大小を次に定義する.

$$\boldsymbol{p}_i(t) > \boldsymbol{p}_j(t)$$
$$\Leftrightarrow \left(p_i^1(t) > p_j^1(t)\right)\ \vee\ \left(p_i^1(t) = p_j^1(t) \wedge p_i^2(t) > p_j^2(t)\right)$$

上記は, a_i が $T(v)$ の根 v の方向に進もうとする場合に暫定的な優先順位増加を与えていることになる.

タイブレーク

PIBT では, a_i は次時刻で移動可能なノード全てを目的地までの最短距離で評価した後, その中で最高評価値をもつノード v_i^* の予約を試行する. $\boldsymbol{p}_i(t)$ による優先順位付けでは, a_i, a_j が $T(v)$ にいる場合, Fig. **2**b–**2**c に示すような膠着状態が生じる可能性がある.

これを未然に防ぐために, $v_i(t) \in T(v)$ となる a_i の v_i^* の決定に関して, タイブレークを以下のように定める. まず, $v_i(t) = v$, つまり a_i が根にいる場合, $u \notin \forall T(w)$ を優先し, 次に $u \notin T(v)$ となる u を優先的に選択する. $v_i(t) \neq v$ の場合, $\mathrm{cost}(u,v) < \mathrm{cost}(v_i(t), v)$ となる u を優先的に選択する. 上記でタイブレークがなされない場合, 他エージェントの所在の有無を利用する. これにより, エージェントが G' へ優先的に移動することになり, 膠着状態の予防が期待される.

4. 評価実験

Fig. **3**a に示す環境で評価実験を行った. 各エージェントには目的地が連続的に与えられ, $v_i(t{+}1) = g_i(t)$ のとき g_i が更新される. 全エージェントによる目的地の更新回数の総和が規定の値 (1000) に達した後, それまでに発行された目的地全てが割当てられたエージェントによって訪問された時点を終了状態とした. 上限となる時刻 (5000) を経過しても終了状態に達しなかった場合, その試行は失敗とみなした. これはエージェントが膠着状態に陥ったことを示唆する. エージェント数を 10 から 500 まで変化させながら, それぞれ 50 回ずつ実験を行った.

PIBT との比較結果を Fig. **3**b–**3**d に示す. makespan は終了状態に達した時刻を表し, service time は目的地が与えられてからそこに到達するまでに要した時間を表している. 実験結果は, 提案手法は PIBT と経路効率の観点で競合し, PIBT が膠着状態に陥る状況に対処可能であることを示唆している.

5. 結び

本稿では繰返し MAPF を解くためのアルゴリズム PIBT の対象グラフを拡張するために, トポロジに応じた暫定的な優先順位増加を導入を検討した. 今後の課題として G' が非連結である G への拡張や, 定理 1 に相当する, エージェントの到達可能性の理論保証を与えることが挙げられる.

■**謝辞** 本研究の一部は JSPS 科研費 JP17K00019, 17K12734 および JST-SICORP の助成を受けたものである.

参 考 文 献

1) Peter R Wurman, Raffaello D'Andrea, and Mick Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI magazine*, Vol. 29, No. 1, p. 9, 2008.
2) Jingjin Yu and Steven M LaValle. Structure and intractability of optimal multi-robot path planning on graphs. In *AAAI*, 2013.
3) Ariel Felner, Roni Stern, Solomon Eyal Shimony, Eli Boyarski, Meir Goldenberg, Guni Sharon, Nathan Sturtevant, Glenn Wagner, and Pavel Surynek. Search-based optimal solvers for the multi-agent pathfinding problem: Summary and challenges. In *Tenth Annual Symposium on Combinatorial Search*, 2017.
4) Keisuke Okumura, Manao Machida, Xavier Défago, and Yasumasa Tamura. Priority inheritance with backtracking for iterative multi-agent path finding. In *IJCAI*, 2019.
5) Lui Sha, Ragunathan Rajkumar, and John P Lehoczky. Priority inheritance protocols: An approach to real-time synchronization. *IEEE Transactions on Computers*, Vol. 39, No. 9, pp. 1175–1185, 1990.
6) David Silver. Cooperative pathfinding. *AIIDE*, Vol. 1, pp. 117–122, 2005.