2J1-5

経路探索を動的に行うオークション・アルゴリズムについて

Auction algorithms that perform on-demand pathfinding

岸本 章宏[*1] 永野 清仁[*2]
Akihiro Kishimoto Kiyohito Nagano

[*1]IBM Research, Ireland
IBM Research, Ireland

[*2]公立はこだて未来大学システム情報科学部複雑系知能学科
Department of Complex and Intelligent Systems, School of Systems Information Science, Future University-Hakodate

The routing problems such as robot and SAV routing are a representative topic for multi-agent coordination. Auctions algorithms can solve these problems. However, they suffer from high computational overhead for calculating the bidding costs based on the shortest paths among locations. We introduce auction algorithms that perform on-demand pathfinding based on contraction hierarchies to speed up the procedure of bidding cost computation. In addition, we experimentally evaluate the performance of our algorithms on actual road maps.

1. はじめに

ロボットや SAV，荷物の配送などのルーティング問題を解くアルゴリズムの研究は，コンピュータ・サイエンスやオペレーションズ・リサーチにおける重要な課題の一つである．これらのルーティング問題を解くには，あらかじめ与えられた大局的な目的を達成するために，各エージェントが与えられた仕事を局所的に解決する必要があるので，人工知能分野のマルチエージェント協調問題の代表的な例題である．

ルーティング問題は，オークション・アルゴリズムを用いれば，各エージェントが問題の大部分を局所的に解くことができる [Lagoudakis 05]．しかし，より現実世界に近い問題設定では，各エージェントが行う入札値の計算のために，対象マップ上での経路探索を何度も行う必要があるので，経路計算の計算負荷が高くなってしまうことが知られている [Kishimoto 08]．このため，実用的なオークション・アルゴリズムでは，この経路探索の計算の手間を減らすことが必要不可欠である．

本論文では，道路マップ上を走る複数の車を用いて，多くの乗客を輸送する問題を定義し，オークション・アルゴリズムに必要な経路探索のオーバーヘッドを減らすことを試みる．具体的には，代表的な最短経路探索アルゴリズムの一つである縮約階層 [Geisberger 08] を用いて経路探索を動的に行い，オークション・アルゴリズムの入札値計算を高速に行えるアルゴリズムを提案する．さらに，これらのアルゴリズムを実装し，実際の道路マップを用いて，性能評価を行う．

2. 本論文で対象にするルーティング問題

本論文では，[Kishimoto 08] で取り組まれた MRR(Multi-Robot Routing) 問題を拡張したルーティング問題 (以下，乗客輸送問題と呼ぶ) を取り扱う．乗客輸送問題は，道路マップ上の様々な場所に位置する N 台の同一性能の車と M 人の乗客からなる．MRR 問題のロボットとターゲットは，乗客輸送問題では車と乗客にそれぞれ対応する．各乗客は，1 台の車が必ず迎えに行き，その乗客の指定する目的地に必ず送り届ける．各車の出発位置，各乗客の乗車および下車場所は既知であるが，道路マップ内の地点間の車による移動時間は，経路探索で求めなければならないと仮定する．乗客輸送問題では，最後の乗客を送り届けるのに最も移動時間のかかる車の移動時間を最小にするような，各車への乗客の割り当て方を見つけることが目的である．

この乗客輸送問題自体は，VRPPD 問題 (Vehicle Routing Problem with Pickup and Delivery) などのいわゆる配送計画問題と多くの共通点があるが，本論文では，マルチエージェント研究の観点から，この問題を取り扱う．

連絡先: akihirok@ie.ibm.com

3. 提案手法

本節では，オークション・アルゴリズムと縮約階層に用いた経路探索アルゴリズムについて概説しながら，これらの手法を用いて，乗客配送問題を解く手法について説明する．

3.1 オークション・アルゴリズムの利用

MRR 問題自体は，NP 困難であることが証明されており，最適解に対する最悪性能を保証しつつ，高速に近似解の計算ができるオークション・アルゴリズムの研究が行われている [Lagoudakis 05]．特に，SSI(Sequential Single-Item) オークションが MRR 問題では良く用いられており，本論文の乗客輸送問題にも利用できる．

乗客数を M とすると，SSI オークションでは M ラウンドの入札を行う．その際に，車と乗客をそれぞれ入札への参加者と商品であるとみなす．各ラウンドでは，それぞれの車は，まだどの車にも割り当てられていない乗客 1 人にのみ入札できる．さらに，各ラウンドでは，1 人の乗客のみを割り当てることとし，最小入札値を投票した車に希望通りの乗客を割り当てる．最小入札値を投票した車が 2 台以上ある場合には，その中のどの車を選んでもよい．この乗客割り当て過程を M 回繰り返し，すべての乗客を車に割り当てる．

オークション・アルゴリズムを利用してルーティング問題を解くことの利点の一つは，各車が他の車の情報を知ることなく，局所的に入札値の計算ができることである．さらに，入札値を受け取るサーバーは，各車からの入札コストと乗客情報のみを受け取り，割り当てる乗客と車の組み合わせを決めればよいので，サーバーの計算負荷が少ない．

本論文では，[Lagoudakis 05] の BIDMAXPATH 戦略に

基づき，各車の入札値の計算はその車が開始地点を出発してから，最後の乗客を送り届けるまでにかかる移動時間を利用する．P_i を車 i が獲得した乗客の集合，$p_1, p_2, \cdots, p_k$ を未割り当ての乗客とする．さらに，$path(i, P)$ を車 i の出発始点から，P に属する最後の乗客の乗り降りを終了するまでにかかる時間とすると，車 i の入札値 $b(i)$ は次の通りである．

$$b(i) = \min_{p_j \in \{p_1, p_2, \cdots, p_k\}} path(i, P \cup \{p_j\})$$

v, w を車の出発地点，乗客の乗車または降車地点とし，$c(v, w)$ を v から w への移動に要する時間とする．$path$ の計算には，$c(v, w)$ の値が必要があるが，$c(v, w)$ を求めるには，道路マップ上での経路探索を行わなければならない．たとえ，1 回あたりの $c(v, w)$ の計算時間自体は短くても，この経路探索のオーバーヘッドは，マップが複雑でかつ乗客数が多ければ，経路探索アルゴリズムの呼び出し回数が増加にするので，$path$ の計算時間に無視できない影響を与えうる．例えば，すべての組み合わせ (v, w) について $c(v, w)$ を計算する単純な方法は，最悪の場合，$O(|P^2|)$ 回の経路探索を行う必要がある．

さらに，$c(v, w)$ があらかじめ与えられている場合でも，$P \cup \{p_j\}$ の中で最適な乗降順を見つけるのは NP 困難問題である．

本論文では，[Lagoudakis 05] のように， TSP 問題でよく用いられている，インサーション・ヒューリスティックを用いて $path$ を計算する．$path(i, P \cup \{p_j\})$ の計算には，$path(i, P)$ の経路内に乗客 p_j の乗降位置を追加する場合のみに特化し，その $path$ の中で最小のものを $b(i)$ として利用する．

3.2 縮約階層に基づく経路計算

本論文が利用する [Geisberger 08] の手法では，縮約階層という空間を定義し，経路探索は Dijkstra 法で双方向探索を行なっている．この方法では，前処理として，縮約階層グラフを構築する．この前処理では，まず，道路マップを有向グラフ G に変換し，G の節点に全順序を付けて並べ替える．*1次に，$x > y$ と $z > y$ を満たす節点 x, y, z に対して，枝 (x, y) と (y, z) が存在し，かつ $x \to y \to z$ が唯一の最短経路である場合には，ショートカットと呼ばれる枝 (x, z) を追加することを繰り返す．(x, y) のコストは，(x, y) と (y, z) のコストの和である．ショートカットの追加は，双方向探索が生成する子節点を制限することから生じる，$x \to y \to z$ が最適解である場合を見逃さないようにするためである．

$c(v, w)$ を求める際には，前処理で計算した縮約階層グラフ上を，v から順方向に w へと向かう探索と w から逆方向に v へと向かう探索を交互に行う．順方向探索が節点 n を展開する場合には，縮約階層グラフ内の枝 (n, m) の中で $n < m$ を満たす子節点 m のみを生成する．一方，逆方向探索が n を展開する際には，枝 (m, n) に対して，$n < m$ を満たす子節点 m のみを生成する．n が順方向と逆方向探索の両方で生成されていれば，v から n 経由で w に到達する経路があることが分かる．しかし，この経路が最小のコストを持つとは限らないので，最適解であると証明されるまで，双方向探索を繰り返す．

$c(v, w)$ の計算の際に，現在 v からその節点に至るまでの枝コストの和の小さな順番に，未展開の節点を並べ替える順序付きキューを用いる順方向 Dijkstra 法を利用すると仮定し，キューの先頭にある節点 n の値を $g_1(n)$ とする．同様に，逆方向 Dijkstra 法の優先度キューの先頭にある節点 m の値を $g_2(m)$ とし，ub を現在の双方向探索で見つけた最善解である．Dijkstra 法の性質により，$\min(g_1(n), g_2(m)) \geq ub$ を満たせば，ub が最適解であることが保証できる．

3.3 動的な経路探索

縮約階層を用いる経路探索は，Dijkstra 法ベースであるので，1 開始地点から多終了地点の探索を自然に行える．$b(i)$ の計算では，乗客の乗降車地点からの順方向と逆方向探索，および車 i からの順方向探索に関する情報をすべて保持しておく．順方向探索の 1 つが節点 n を展開する前に，すべての逆方向探索が n をすでに生成済みかどうかを確認し，生成済みの場合には，その順方向および逆方向探索で計算した，現在の最善経路コストを更新する．逆方向探索の節点を展開する場合にも同様の処理を行う．

本論文では，$b(i)$ で利用する $path$ の計算にインサーション・ヒューリスティックを利用しているので，この性質を利用すれば，経路探索の呼び出し頻度を大幅に減らせる．

車 i が第 j ラウンドのために計算した経路 $current\_path = v_{i0} \to l_{i1} \to l_{i2} \cdots \to l_{ik}$ （v_{i0} は i の出発地点，l_{im} を乗客の乗車または降車位置）とし，$p_1, p_2, \cdots, p_n$ をまだ割り当てられていない乗客とする．第 $j+1$ ラウンドの入札では，未割り当ての各乗客 1 名の乗り降り位置を $current\_path$ に挿入した際に必要となる地点間のみの経路コストを経路探索で求めればよい．そのため，例えば，乗客 p_1 の乗車位置から乗客 p_2 の乗車位置の経路をこのラウンドでは計算する必要がない．

各 $p_r (1 \leq r \leq n)$ について，p_r の乗降地をインサーション・ヒューリステックが挿入できる，合法な $current\_path$ のインデックスのペア q（p_r の降車位置を乗車位置より前に挿入する場合を考えない）とする．q に p_r の乗降値を挿入して生成する乗客輸送経路の候補 $path\_candi(q)$ を，すべての q について列挙した後，$path\_candi(q)$ に必要な $c(v, w)$ をすべて正確に計算し，$b(i) = \min_q path\_candi(q)$ を求める方法が，手法の一つとして提案できる．この手法を**アルゴリズム A** と呼ぶ．

さらに，BIDMAXPATH 戦略では，$path\_candi(q)$ が最適解でない場合には，$path\_candi(q)$ の下限が最適解 opt 以上であることを示せばよい．つまり，$path\_candi(q)$ で新たに追加する $c(v, w)$ の下限を $path\_candi(q) \geq opt$ を満たすまで，双方向探索で上昇させればよい．このようにすれば，不必要な $c(v, w)$ の計算を減らすことができる．

本論文では，[Kishimoto 08] にある方法を利用する．この手法を**アルゴリズム B** と呼ぶ．車 i は，$path\_candi(q)$ の現在の下限値 $path\_lb(q)$ を用いて，昇べき順に q を並べる優先度キュー Q を用意する．また，δ を Q への要素挿入・削除の回数を減らすための小さな定数とする．Q の先頭と 2 番目の要素をそれぞれ q_1 と q_2 とすると，アルゴリズム B では，q_1 を Q から取り出し，$path\_lb(q_1) \geq path\_lb(q_2) + \delta$ になるまで，q_1 に関連する $c(v, w)$ の探索を行う．この条件を満たすか $path\_lb(q_1)$ が q_1 に関する最適解になれば，q_1 を Q に再度挿入する．この作業を Q の先頭要素の $path\_lb$ が最適解になるまで繰り返す．

[Kishimoto 08] では，最小全域木を構築後，経路に変換し，入札を行っている．また，元々のグリッドマップで構成される探索空間では，経路探索に時間がかかりすぎることを指摘している．そのため，アルゴリズムの近似性能と実行時間とのトレードオフとして，$c(v, w)$ の計算に，[Sturtevant 05] で提案された，クリークに基づいて簡約化した探索空間上でのみ経路探索を行い，その結果を $c(v, w)$ で代用する方法が最も有効であると結論付けている．一方，本論文では，$c(v, w)$ の値自体は最適解であり，さらに最小全域木を経由せずに直接経

*1 理論的には，節点の順序の付け方は任意でよいが，効率の良い縮約階層を構築するためには，様々なヒューリスティックスがある．

表 1: 実行時間と生成節点数（乗車数制限なし）

乗客数	20	50	100	150
アルゴリズム A	1.06	7.10	32.13	73.52
アルゴリズム B	1.02	6.72	26.88	59.38
アルゴリズム C	1.00	5.72	24.06	55.78
アルゴリズム A	1,042,582	2,799,167	5,650,365	8,656,244
アルゴリズム B	833,685	1,956,407	3,598,918	5,223,272
アルゴリズム C	784,652	1,658,517	3,060,067	4,266,791

表 2: 実行時間と生成節点数（乗車数制限あり）

乗客数	20	50	100	150
アルゴリズム A	1.03	6.86	30.77	70.08
アルゴリズム B	1.00	6.67	29.16	67.11
アルゴリズム C	0.95	5.77	24.55	54.13
アルゴリズム A	1,043,017	2,741,496	5,626,395	8,642,276
アルゴリズム B	835,275	2,029,158	4,016,277	6,096,531
アルゴリズム C	784,373	1,717,355	3,325,307	4,954,936

路を生成している．MRR 問題でも，インサーション・ヒューリステックに基づき，ロボットのルーティング経路を直接計算し，入札値に利用する方が，最小全域木を経由する方法よりも，実験的に性能が良いことが知られている [Lagoudakis 05]．さらに，乗合タクシーや SAV のルーティング問題でも，オークションを利用するか否かにかかわらず，経路を直接生成するアプローチが盛んに研究されている [Ma 13, Nakashima 14]．

3.4 サーバー上の入札情報の利用

サーバーは第 k ラウンドの入札が終わったとき，各車の入札値と入札した乗客の情報をもとに乗客割り当てを決定する．[Kishimoto 08] では，第 k ラウンドで割り当てられなかった乗客リストの中で，最小の入札値は，第 $k+1$ ラウンドの入札値の上限値 ub であることを利用し，このラウンドでの入札値の計算の手間を減らしている．本論文のアルゴリズムでも，この方法を用いて，車 i の $path\_lb(q) \geq ub$ が証明できた時点で，$path\_lb(q)$ の計算を終了するようにした．また，車 i のすべての q で $path\_lb \geq ub$ を満たせば，i は入札を放棄することをサーバーに知らせるようにした．アルゴリズム B にこの改良を施したものを**アルゴリズム C** と呼ぶ．

4. 実験結果

提案したアルゴリズム A と B および C を実装し， Intel Core i5-2520M（クロック数 2.50GHz，4CPU コア）と 4GB のメモリで構成される PC 上で実験を行った．実装には，縮約階層を利用した高性能なルーティング・プログラムである Open Source Route Machine[*2]のソースを利用し，実験には 1 コアのみを利用した．

アルゴリズムの性能評価には，アイルランドの道路マップを利用し，ダブリン県内に車を 5 台と乗客をランダムに配置し，乗客数が 20-150 名の場合を調べた．さらに，1 台の車に同時に搭乗できる乗客数に制限がない場合と，5 名までに制限したときのアルゴリズムの性能を調べた．各設定について，それぞれの手法に解かせる共通問題を 10 題用意し，平均実行時間と平均生成節点数を求めた．

表 1 と 2 に，各アルゴリズムの性能を示す．各表の上から 3 行の数値が，それぞれのアルゴリズムの平均解答時間 (秒) であり，残りの 3 行は平均生成節点数である．これらの表より，インサーション・ヒューリステックの性質から，どのアルゴリズムも実行時間が大幅に増加していることが読み取れる．特に，乗客数を 20 名から 150 名に増やした場合には，アルゴリズム A の実行時間が約 1 秒から約 70 秒に増加している．例えば，SAV ルーティング [Nakashima 14] を大都市で利用する場合には，オークション・アルゴリズムは，多数の車と乗客を取り扱う必要があるので，入札値の計算オーバーヘッド (経路探索に要する時間) の削減が強く望まれる．

$path\_lb$ による優先度キューの導入によって，アルゴリズム B は，アルゴリズム A と比べて，最大で生成節点数を約 40%削減し，実行時間は約 20%少なくなっている．この結果は，優先度キューの管理の手間をかけても経路探索の無駄な節点展開を避けることの重要性を示唆している．アルゴリズム C は，サーバーが持っている，割り当てられなかった乗客への入札値の情報を有効に利用することで，アルゴリズム B の実行時間をさらに $10-20\%$ 減らすことに成功している．

各アルゴリズムの振る舞いは，各車の乗客数に制限数を設けた場合も，乗客数が無制限の場合とほぼ同様である．ただし，アルゴリズム B の効果は，乗客数が無制限の場合よりも少なかった．

5. まとめと今後の課題

本論文では，縮約階層に基づく経路探索とオークション・アルゴリズムを組み合わせた手法を提案し，乗客輸送問題に適用した．今後の課題としては，アルゴリズムの近似性能の理論的解析や経路探索のオーバーヘッドのさらなる削減，および実世界でのアルゴリズムの性能評価 [Nakashima 14] などが挙げられる．

参考文献

[Geisberger 08] Geisberger, R.: Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks, Master's thesis, Universität Karlsruhe (2008)

[Kishimoto 08] Kishimoto, A. and Sturtevant, N.: Optimized Algorithms for Multi-Agent Routing, in *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems*, pp. 1585–1588 (2008)

[Lagoudakis 05] Lagoudakis, M., Markakis, V., Kempe, D., Keskinocak, P., Koenig, S., Kleywegt, A., Tovey, C., Meyerson, A., and Jain, S.: Auction-Based Multi-Robot Routing, in *Proceedings of the International Conference on Robotics: Science and Systems*, pp. 343–350 (2005)

[Ma 13] Ma, S., Zheng, Y., and Wolfson, O.: T-Share: A Large-Scale Dynamic Taxi Ridesharing Service, in *Proceedings of the 29th IEEE International Conference on Data Engineering (ICDE)*, pp. 410–421 (2013)

[Nakashima 14] Nakashima, H., Sano, S., Hirata, K., Shiraishi, Y., Matsubara, H., Kanamori, R., Koshiba, H., and Noda, I.: One Cycle of Smart Access Vehicle Service Development, in *Proceedings of the 2nd International Conference on Serviceology*, pp. 152–157 (2014)

[Sturtevant 05] Sturtevant, N. and Buro, M.: Partial Pathfinding Using Map Abstraction and Refinement, in *AAAI*, pp. 1392–1397, AAAI Press (2005)

*2 http://project-osrm.org/