自動化された衣服の効率的な物理的・アルゴリズム的設計に向けて

Teng Guo　　Jingjin Yu

概要- 大都市圏への駐車は、都市の造園に影響を与える交通パターンに対して、さらに示唆を与える、時間のかかる作業であることが多い。駐車場に必要なプレミアムスペースを削減することで、自動機械式駐車場システムの開発につながった。自動化されたガレージは、各島に1～2列の車両を持つ通常のガレージと比較して、複数列の車両を積み重ねることで、より高い駐車需要をサポートすることができる。この複数列のレイアウトは駐車スペースを縮小するが、駐車と検索がより複雑になる。本研究では、100%近い駐車密度をサポートする自動ガレージ設計を提案する。駐車場の駐車と複数車両の回収の問題をマルチロボット経路計画問題の特殊なクラスとしてモデル化し、自動化されたガレージのすべての一般的な操作を扱うための関連アルゴリズムを提案する。(1)駐車/回収を同時に行う実現可能で効率的な解を求める最適アルゴリズムと最適に近い方法、(2)ラッシュアワーでのスケジュール検索を容易にするために車両を再配置する新しいシャッフル機構、などである。提案手法が大規模かつ高密度の実世界駐車アプリケーションに有望であることを示す徹底的なシミュレーション研究を実施する。

I. Introduction

自動駐車システム(ガレージ)の発明は、都心部や人口の多い地域など、スペースに大きなプレミアムがある地域の駐車問題を解決するのに役立つ。現在、駐車場の面積はますます不足し、高価になってきている。マンハッタンのスポットは、簡単に200,000米ドルを超える可能性がある。高密度駐車場をサポートし、スペースを節約し、より便利なガレージを開発することは、経済的/効率的な理由から非常に魅力的である。

自動化されたガレージでは、人間のドライバーは、駐車プロセスを気にすることなく、特定のI/O(入出力)ポートで車両を落下(ピックアップ)させるだけでよい。このようなシステムでは、ドアを開けるために周囲のスペースを必要としないため、駐車がより近くなる可能性がある。駐車場や港に車両を移動させることは、このようなシステムにとって重要な機能である。解決策の一つは、ロボットのパレットを使って車両を移動させることである。このようなシステムはすでに市販されており、例えば香港のHKSTP [1]がある。このようなシステム[2]では、車両は互いにブロックし合うことができるように駐車されており、特定の車両を検索するために複数の再配置が必要である。残念ながら、これらのシステムがどの程度機能しているか、例えば駐車/回収効率に関する情報はほとんどない。その他のソリューションは、自動運転車のための駐車に焦点を当てている。このような自動化されたガレージでは、車両は駐車枠や港まで自らを運転することができる。となり

G. Teng、J. Yuは、米国ニュージャージー州ピスカタウェイのニュージャージー州立大学ラトガース校コンピュータサイエンス学部に所属している。電子メール { teng.guo, jingjin.yu}@rutgers.edu.この研究は、NSF賞IIS-1845888とAmazon Research Awardの一部支援を受けている。

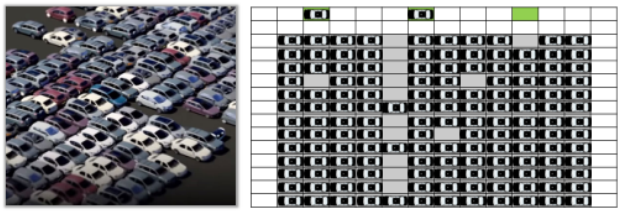

図1. 左:想定される自動ガレージシステムの密度レベルの説明図。右: 右:グリッドベースの抽象化で、3つのI/Oポートで車両をドロップオフし、検索を行う。柔軟性はあるが、限られた省スペースしか提供しない

the system more flexible but provides limited space-saving advantages, besides requiring autonomy from the vehicles.

近年、効率的なマルチロボット経路計画アルゴリズムが数多く提案され、複数のロボットパレットを用いた駐車・回収コストの低減が可能となっている。本研究では、マルチロボットに基づく駐車・検索問題を研究し、完全な自動ガレージ設計を提案し、ほぼ100%の駐車密度をサポートし、ガレージを効率的に動作させるための関連アルゴリズムを開発する。

結果と貢献。我々の研究の主な結果と貢献は以下の通りである。自動ガレージの設計において、このようなガレージが必要とする主要な操作をモデル化するバッチ式車両駐車・検索(BVPR)と連続式車両駐車・検索(CVPR)の問題を導入し、自動ガレージシステムの将来の理論的・アルゴリズム的研究を促進する。
アルゴリズム面では、$m_1 \times m_2$ の格子地図上で $(m_1 - 2)(m_2 - 2)/m_1\ m_2$ という高い駐車密度に対応し、大型ガレージの駐車密度が100%に近づく複数車両の駐車・回収が可能なシステムを研究する。グリッドのようなシステムの規則性を利用して、最適なILPベースの手法と、逐次計画に基づく高速な準最適アルゴリズムを提案する。我々の準最適アルゴリズムは、良好な解の質を維持しながら、非常にスケーラブルであり、大規模なアプリケーションに適している。さらに、検索順序が予想できる場合、ラッシュアワーに高速な車両検索を行うために、オフピーク時に車両を並べ替えるシャッフリング機構を導入する。我々の再配置アルゴリズムは、全ガレージ密度付近で$O(m_1\ m_2)$の総時間コストでこのようなシャッフルを実行する。

関連研究。研究者は、効率的な高密度駐車ソリューションに向けた多様なアプローチを提案している。自動運転車のための多くのシステムが研究されており[3]-[5]、車両は中央制御装置を使って駐車され、複数の列に積み重ねられ、互いにブロックすることができる。これらの設計は、駐車容量を最大50%増加させる。

しかし、検索は閉塞により非常に複雑になり、自動
運転車の操縦性に大きく影響される。
ほとんどの車両が自動運転できないため、ロボット・バレットベースの高密度駐車システムの方がより適切な選択となりうる。パズルベースドストレージ(PBS)システムまたはグリッドベースシャトルシステムは、もともと[6]によって提案されたもので、最も有望な高密度ストレージシステムの一つである。このようなシステムでは、AGVやシャトルなどの蓄電ユニットは4つの枢軸方向に可動である。空のセル(エスコート)が少なくとも1つなければならない。車両を回収するためには、エスコートを利用して、希望する車両をI/Oポートに移動させなければならない。これは、NP困難であることが知られている15パズルに似ています[7]。単一のエスコートと複数のエスコートを持つ1台の車両を検索するための最適なアルゴリズムが[6], [8]で提案されている。しかし、これらの方法は一度に1台の車両を検索することしか考慮していない。また、平均検索時間は従来の通路ベースのソリューションよりもはるかに長くなる可能性がある。容量需要と検索効率のトレードオフを達成するために、最近の先進的なマルチロボット経路計画(MRPP)アルゴリズム[9]を利用することで、複数の車両の検索と駐車を同時に可能にする、より多くのエスコートとI/Oポートを使用することを提案する。

MRPPは広く研究されている。静的またはワンショット設定[10]では、グラフ環境と、各ロボットがユニークな開始位置とゴール位置を持つ多数のロボットが与えられたとき、タスクは、開始からゴールまでの全てのロボットの衝突のない経路を見つけることである。ワンショットMRPPをメイクスパンかコスト和のどちらかを最小化するという観点から最適に解くことはNP困難であることが証明されている[11], [12]。MRPPのソルバーは最適と準最適に分類できる。最適ソルバーは、MRPPをILP [13]、SAT [14]、ASP [15]などのよく研究された問題に還元するか、探索アルゴリズムを用いて最適解を見つけるために結合空間を探索する [16], [17]。NPhardnessのため、最適ソルバーは大きな問題を解くのに適していない。境界付き準最適ソルバー[18]は、強い最適性保証を持ちながら、より良いスケーラビリティを達成する。しかし、特に高密度環境では、まだスケールが小さい。大規模MRPPを解くための多項式時間アルゴリズム[19]があるが、これは解の質を犠牲にしている。他のO(1)時間最適多項式時間アルゴリズム[20]-[23]は、主にメイクスパンの最小化に焦点を当てており、連続的な設定にはあまり適していない。
組織である。本稿の残りの部分は以下のように構成されている。第II章では、ガレージ設計を含む前置きを説明する。第III-第IV章では、自動化されたガレージを操作するためのアルゴリズムを示す。第V章では、ガレージシステムの徹底的な評価と考察を行い、第VI章で結論を述べる。

II. Preliminaries

A. ガレージ設計仕様

本研究では、自動格子ベースのガレージは、4連結$m_1 \times m_2$格子G(V, E)である(図1参照)。グリッドの最上縁には、駐車車両や特定の駐車車両を検索するためのn_o個のI/Oポート(ここでは単にポートと呼ぶ)が分散している。ポートは、検索または駐車のいずれかにのみ使用できる。
車両は駐車場(下部中央($m_1 - 2$)×($m_2 - 2$)サブグリッドのセル)に駐車しなければならない。空き地が駐車したら、それは移動可能な障害物となる。$O = \{o_1, \cdots, o_{no}\}$ は港の集合、$P = \{p_1, \cdots, p_{|P|}\}$ は駐車場の集合である。

B. バッチ車両駐車・検索(BVPR)

バッチ式車両駐車・回収(BVPR)は、バッチ式駐車・回収の最適化を目指す。1バッチで、駐車する車両はn_p台、回収する車両はn_r台、駐車する車両はn 1台である。C = C p ～C r ～C l を全車両の集合とする。すなわち、¦C¦ < ¦P¦である。時間はタイムステップに離散化され、AGV/シャトルが運ぶ複数の車両が同時に移動することができる。各タイムステップにおいて、各車両は現在の位置で左、右、上、下、または待機することができる。車両間の衝突は避けるべきである:

1) Meet collision. Two vehicles cannot be at the same grid point at any timestep: $\forall i, j \in \mathcal{C}, v_i(t) \neq v_j(t)$;
2) Head-on collision. Two vehicles cannot swap locations by traversing the same edge in the opposite direction: $\forall_i$によって、(v_iは、∧v in$_i$ (t their + (see l) 反対側の ılse;
3) P_i移動 = 図v_j (t))のとき、次の=同じv reacher$_j$ (t + edg ve-hie collisions l)に従う。2).方向: l = 偽; veare 垂直。ions arタイムステップtにおける車両iの移動方向ベクトルを$e^_i$ the $m^(t) = v_i$ (t + l) − v_i (t)とすると、∀i, j∈C, (v_i (t :hen $\forall_i$ + l) = v_j (t) ∧ $e^_i$ (t) ≒ $e^_j$ (t)) = false.

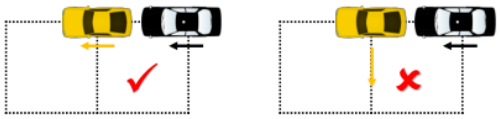

図2. 同じ方向への車両の平行移動(左)は許容されるが、車両の垂直追従(右)は禁止される。

ある種のMRPP定式化[10]とは異なり、垂直な追従衝突を考慮する必要があり、問題を解くのが難しくなる。タスクは、各車両の無衝突経路を見つけ、現在位置から所望の位置に移動させることである。具体的には、車両が回収されるためには、その目標が指定された港である。他の車両については、所望の位置は駐車場のいずれか1つである。解決策の質を評価するために、以下の基準が用いられる:

1) 平均駐車/検索時間(APRT):車両の回収または駐車に必要な平均時間。
2) Makespan (MKPN): すべての車両を希望する位置に移動させるのに必要な時間。
3) Average number of moves per task (ANM): the sum of the distance of all vehicles divided by the total number of vehicles in $\mathcal{C}_p \cup \mathcal{C}_r$.

一般に、これらの目的はパレートフロント[12]を作り出し、これら2つを同時に最適化することは必ずしも不可能である。

C. 連続車両駐車・検索(CVPR)

CVPR は、駐車場と検索問題の連続バージョンである。BVPRの構造の大部分を継承しているが、いくつかの重要な違いがある。

この定式化では、以下の仮定を置く。車両i∈C_rが所望の港に到着すると、環境から除去される。港に(容量内で)駐車する必要がある新しい車両が出現し、車両の回収に新たな要求が発生する。また、ある港が車両の回収に使用されている場合、他のユーザは回収タスクが終了するまで、その港に車両を駐車することができない。時間地平が無限大または固定である場合を除き、解の質を評価するために3つの基準を使用することができる。

D. 車両シャッフリング問題(VSP)

実際のガレージでは、システムが容量に達するオフピーク時(朝のラッシュアワーなど)にいることが多く、回収要求も少ない。後の時間帯(例えば、午後のラッシュアワー)における車両の検索順序がわかっている場合、その情報を利用して車両を再シャッフルし、後での検索を容易にすることができる。この問題はワンショットMRPPとして定式化される。開始構成X_Iとゴール構成X_Gが与えられたとき、再構成を達成するために衝突のない経路を見つける必要がある。ゴールの構成は、車両の検索時間順序に従って決定される。先に検索されると予想される車両は、後で出発する他の車両に遮られないように、港の近くに駐車されるべきである。

III. Solving BVPR

A. 整数線形計画法(ILP)

13]、[24]のネットワークフローに基づくアイデアを基に、BVPRを多商品マックスフロー問題に還元し、整数計画法を用いて解く。C_rの車両は特定のゴール位置(港)を持っており、異なる商品として扱われなければならない。一方、C_p △C_Iの車両は駐車枠のどれかに駐車できるので、1つの商品と見なすことができる。インスタンスがTステップで解けると仮定して、図3に示すようなTステップの時間拡張ネットワークを構築する。

Tステップの時間拡張ネットワークは、有向ネットワークN_T = (V_T, E_T) で、有向、単位容量のエッジを持つ。ネットワークN_Tは、元のグラフの頂点VのT + 1個のコピーを含む。タイムステップtにおける頂点u∈Vのコピーをu_tとする。タイムステップtにおいて、(u, v) ∈ E または v = u の場合、エッジ (u_t , v_{t+1}) がE_Tに追加される。i∈C_rに対して、頂点s_{i0}(s_iはiの開始頂点)に商品タイプiの1ユニットの供給を与えることができる。車両iがポートに到着することを保証するために、Tにおけるゴール頂点g_iとソースノードs_{i0}を結ぶフィードバックエッジを追加する。駐車が必要な車両については、補助ソースノードαと補助シンクノードβを作成する。各i∈C_p △C_Iに対して、ノードαとその開始ノードs_{i0}を結ぶ単位容量の辺を追加する。車両は駐車スロットのどれかに駐車できるので、任意の頂点u∈Pに対して、u_Tとβを結ぶ容量が1の辺を追加する。C_p ～ C_I の車両に対する n_p + n_I 単位の商品の供給は、ノードαで与えることができる。

ILPを用いて多商品マックスフローを解くために、バイナリ変数X = {x_{iuvt} }, i = 0, ・・・, n_{r}, (u, v) ∈ E または u = v, 0 ≦ t ≦ T ; 変数を真に設定すると、対応する辺が最終解に使われることを意味する。ILPの定式化は以下のように与えられる。

$$\text{Minimize} \quad \sum_{i,t,u\neq v} x_{iuvt} \tag{1}$$

$$\text{subject to} \quad \forall t,v,i \quad \sum_{u} x_{iuv(t-1)} = \sum_{w} x_{ivwt} \tag{2}$$

$$\forall t,i,v \quad \sum_{v} x_{iuvt} \leq 1 \tag{3}$$

$$\forall t,i,(u,v)\in\mathcal{E} \quad \sum_{i}(x_{iuvt}+x_{ivut}) \leq 1 \tag{4}$$

$$\forall t,i,(u,v)\perp(v,w) \sum_{i}(x_{iuvt}+x_{ivwt}) \leq 1 \tag{5}$$

$$\sum_{i=0}^{n_r-1} x_{ig_is_iT} + \sum_{u\in\mathcal{P}} x_{n_ru\beta T} = n_p + n_r + n_l \tag{6}$$

$$x_{iuvt} = \begin{cases} 0 & \text{iがtで辺(u, v)を横切らない場合} \\ 1 & \text{if } i \text{ traverses edge } (u,v) \text{ at } t \end{cases} \tag{7}$$

式(1)において、時間地平T内の全車両の移動回数を最小化する。式(2)は、各格子点における流れの保存制約を指定する。式(3)は、meet-collisionsを避けるための頂点制約を指定する。式(4)において、車両は同じエッジを反対方向に通過することは許されない。式(5)は、以下のコンフリクトを垂直に禁止する制約を指定する。Tステップの時間拡張ネットワークのプログラミングが実行可能であれば、解を求めることができる。そうでなければ、実行可能な解が得られるまでTを段階的に増加させる。整数計画法が解を持つ最小のTは、最小のメイクスパンである。その結果、ILPは二次目的として、総手数を最小化するメイクスパン最適解を求める。

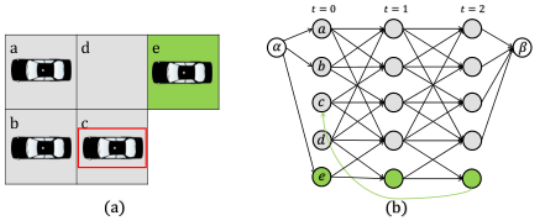

図3 (a) 4つの駐車場と1つの港を持つBVPRインスタンスの例。赤い長方形内の車両は回収する必要があり、他の車両は灰色の領域に駐車する必要がある。(b) BVPRインスタンスから削減された2ステップフローネットワーク。

B. 高密度プランニングのための効率的なヒューリスティック

ILPはメイクスパン最適解を見つけることができるが、スケールが小さい。我々は、解の質の小さなコストで密なBVPRインスタンスを素早く解くことができる高速なアルゴリズムを求めている。我々が提案するアルゴリズムは、検索と駐車の単一車両運動プリミティブに基づいて構築されている。

1) 単一車両の駐車/回収のためのモーションプリミティブ: 解の質に関係なく、BVPRはC_r △C_p の各車両を逐次計画することで解くことができる。具体的には、各ラウンドにおいて、与えられた車両i∈C_r ≒ C_p に対して、iをその港に移動させるか、

駐車スポットのいずれかに移動する1つのタスクを完了することのみを考える。車両iが目的地に到着した後、次のタスクが解決される。図4と図5の例では、最大容量に達しているが、この手法の動作を示している。

図4の検索シナリオでは、赤い四角で示された車両が検索される。左の破線で示した直感的な経路を実現するためには、経路を遮断する車両を途中で排除する必要があり、そのためには、遮断する車両を2つの空の列を利用して左または右に1ステップ移動させることで容易に実現できる。このようなモーションプリミティブは、常にデッドロックのない車両の取得に成功する。

図5の駐車シナリオでは、車両をグリーンポートに駐車する必要がある。まず、空点(エスコート)を貪欲に探索することで、これを行う。車両の平行移動のメカニズムを用いると、まず1タイムステップで駐車車両の柱に移動させ、次に1タイムステップで車両の直下の位置に移動させることができる。その後、車両は目的地であるエスコートに直接移動することができる。

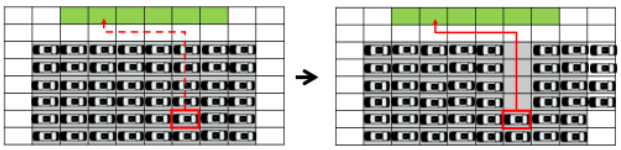

図4. 車両を検索するためのモーションプリミティブ。

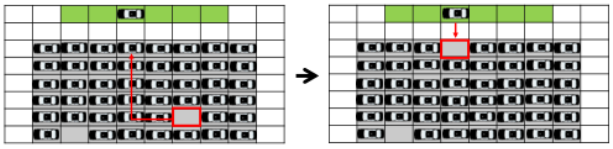

図5. 車両駐車のためのモーションプリミティブ。

逐次計画では、解があれば常に戻るが、複数の駐車場や検索要求を実行する場合、逐次解はMKPNやARPTで測定される性能が低下する。

2) MCPによる単一動作プリミティブの結合(CSMP): 最小通信政策(MCP)[25]は、影響を受けていないロボットを止めることなく、予期せぬ遅延を処理するためのロバストなマルチロボット実行政策である。実行中、MCPは元の計画と同様にロボットが各頂点を訪問する順序を保持する。ロボットiが移動動作を行い、頂点vを入力しようとしているとき、MCPはロボットiが次にその頂点に入るかどうかを元の計画によってチェックする。次に別のロボットjがvに入るように計画されている場合、iはjがvを離れるまで現在の頂点で待機する。

MCPを用いて、逐次計画によって発見された計画に並行性を導入する。アルゴリズムはAlg. 1とAlg. 2に記述されている。1はCSMPのフレームワークを記述している。まず、$C_r \sim C_p$ の全ての車両の初期計画を一つずつ求める(4-6行目)。パスを取得した後、すべての待ち状態を削除し、各頂点の車両訪問順序をキューのリストに記録する(7行目)。次に、すべての車両がタスクを終了し、目的地に到達するまで、MCPを使用して計画を実行するループを入力する(8-15行目)。Alg.2において、iが元の順序に従って頂点v_iに入る次の車両である場合、現在v_iにいる車両があるかどうかをチェックする。

もしそうでなければ、iをv_iに入れる。現在、別の車両jがv_jを占有している場合、次のステップでjが次の頂点v_jに移動しているかどうかを、再帰的に関数MCPMoveと呼ぶことで調べる。次のステップでjがv_jに移動し、i,jの移動方向が垂直でない場合、車両iを頂点v_iに入力する。そうでなければ、iはu_iで待つべきである。このアルゴリズムは構成上デッドロックフリーであるが、ページ数の関係上、比較的簡単な証明は省略する。

命題III.1. CSMPはデッドロックフリーであり、有限時間で常に実行可能解が存在する場合、その解を求める。

Algorithm 1: CSMP

```
1  Function CSMP ():
2      foreach v ∈ V, VOrder[v] ← Queue()
3      InitialPlans ← {}
4      for i ∈ C_p ∪ C_r do
5          SingleMP(i, InitialPlans)
6      Preprocess(InitialPlans, VOrder)
7      while True do
8          for i ∈ C do
9              mcpMoved ← Dict()
10             MCPMove (i)
11         if AllReachedGoal ()=true then
12             break
```

Algorithm 2: MCPMove

```
1  Function MCPMove (i):
2      if i in mcpMoved then
3          return mcpMoved[i]
4      u_i ← current position of i
5      v_i ← next position of i
6      if i = VOrder[v_i].front() then
7          j ← the vehicle currently at v_i
8          if j=None or (MCPMove (j)=true and
             (u_i, v_i) ⊀ (u_j, v_j)) then
9              move i to v_i
10             VOrder[v_i].popfront()
11             mcpMoved[i]=true
12             return true
13     let i wait at u_i
14     mcpMoved[i]=false
15     return false
```

3) 優先順位付け: 逐次計画では、計画順序に関係なく、常に解を見つけることができる。しかし、優先順位はソリューションの質に影響を与える。ランダムな優先順位(Alg.1 4-6行目)で計画を立てる代わりに、単車駐車は2ステップしかかからないので、可能な限りまず駐車計画を立てることができる。C_p内の全ての車両が駐車された後、SingleMPを適用して車両を検索する。C_r 内の車両のうち、まず、より早く目標に到達し、下段の車両をブロックしないように、港に近い車両に SingleMP を適用する。

C. 複雑さ解析

本節では、CSMPの時間複雑性と解のメイクスパン上限を解析する。SingleMPでは、1台の車両を駐車/回収するために、n_b台の車両が閉塞の原因となり、移動の必要性があると仮定する。$n = n_p + n_r + n_l$. したがって、$C_r \Delta C_p$ の全車両に対して SingleMP を用いて経路を計算する複雑さは、$(n_p + n_r)n$ で境界される。

SingleMPによって計算された各単一車両経路の経路長は、$m_1 + m_2$以下である。すべての単一車両経路を連結して得られる経路のメイクスパンは、n_r ($m_1 + m_2$)+$2n_p$ で境界される。これは、MCPがn_r ($m_1 + m_2$) + $2n_p$回以上の反復を要しないことを意味する。したがって、解のメイクスパンは n_r ($m_1 + m_2$) + $2n_p$ で上界される。MCPの各ループにおいて、基本的に$n = n_p + n_r + n_l$ 個のノードを持つグラフ上でDFSを実行し、全てのノードを横断する。したがって、CSMPの時間複雑度は$O(n(n_r m_1 + n_r m_2 + 2n_p))$となる。まとめると、CSMPの時間複雑度は$O(n(n_r m_1 + n_r m_2 + 2n_p))$で境界があり、メイクスパンは$n_r$ ($m_1 + m_2$) + $2n_p$ で上限がある。

D. CSMPのCVPRへの拡張

CSMPはCVPRを解くのに容易に適応できる。BVPR版と同様に、各タイムステップで各車両に対してMCPMoveを呼び出す。新しいリクエストがあるタイムステップに来ると、SingleMPを使用して関連する車両のパスを計算し、頂点の訪問順序の情報を更新する。すなわち、車両$i \in C_p \Delta C_r$にSingleMPを適用するとき、車両jがタイムステップt´で頂点uを訪問する場合、iをキューV Order[u]にプッシュする。このようにして、MCPは訪問順序を維持したまま計画を実行する。以前に計画された車両は、新しい要求の影響を受けず、当初の計画を実行し続ける。この方法の主な欠点は、訪問順序が固定されているため、通常、再計画よりも解の質が悪いことである。

IV. ライブ・ケーブルによる対照解答

VSPは本質的に静的/ワンショットMRPPを解いている。$m_1 \times m_2$ グリッド上では、$2(m_2 -2)$列シャッフルと$(m_1 -2)$行シャッフルを超えないルービックテーブルアルゴリズム[26]を適用することで解くことができる。図6に示す例として、近傍の2つの列を使って、与えられた列の車両をかなり効率的にシャッフルすることができ、$O(m_1)$ステップしか必要としない[20]。行シャッフルも同様である。駐車車両の数に応じて、1つ以上の複数の行/列シャッフルを同時に実行することができる。となる(スペースが限られているため、ストレートフォワード証明は省略)。

命題IV.1. ガレージ容量がフルの場合、$O(m_1 m_2)$のメイクスパンと、ガレージに$\Theta(m_1 m_2)$の空きスポットと$\Omega(m_1 m_2)$の駐車車両がある場合の$O(m_1 + m_2)$のメイクスパンを用いてVSPを解くことができる。一方、$\Omega(m_1 m_2)$の駐車車両では、VSPを解くために必要なメイクスパンは$\Omega(m_1 + m_2)$である。

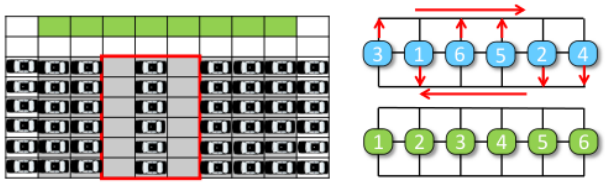

図6. 1行/1列を「シャッフル」する仕組みの説明図。

V. Evaluation

本節では、提案アルゴリズムを評価する。すべての実験はIntel® Core™ i7-9700 CPU、3.0GHzで行った。各データポイントは、特に断りのない限り、ランダムに生成されたインスタンスに対する20回の実行の平均である。ILPはC++で実装され、その他のアルゴリズムはCPythonで実装されている。シミュレーションの動画は https://youtu.be/XPpOB5f7CzA にある。

A. BVPRにおけるアルゴリズム性能

グリッドサイズを変化させる。最初の実験では、グリッドの辺の長さを変化させたm×mグリッドで、最も密なシナリオの下で提案アルゴリズムを評価する:システムには$(m - 2)^2$台の車両があり、すべてのポートは駐車または検索に使用される($n_p + n_r = n_o$)。その結果は図7にある。CONCATは、単純に単車経路を連結する手法である。rCSMPでは、優先順位がランダムな車両に対してSingleMPを適用し、pCSMPでは、優先順位付け戦略を適用する。

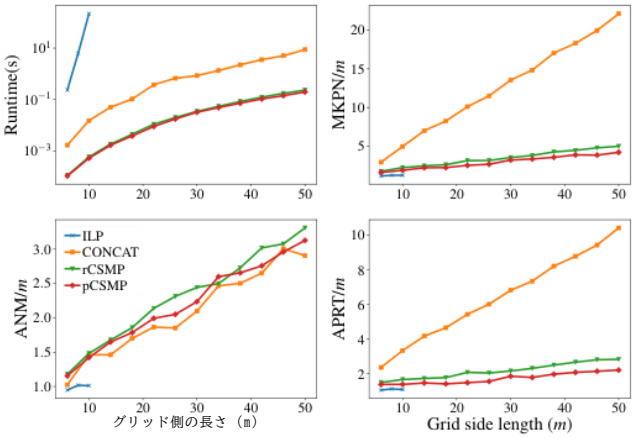

図7. 提案手法のm×mグリッドにおけるランタイム、MKPN、APRT、AVNデータ。

手法の中で、ILPはMKPN、ANM、APRTの点で最も優れた解の質を持っており、その最適性が保証されていることから予想される。しかし、ILPはm≦10、n≦64の極限に達し、スケーラビリティが最も低い。CONCAT、rCSMP、pCSMPはよりスケーラブルで、50×50グリッド、2304台のインスタンスを数秒で解くことができる。CONCATは単車経路を連結しているだけなので、rCSMPやpCSMPに比べて非常に長い経路になる。MCP手順は同時実行性を大幅に改善し、より優れた解の質につながることが観察される。CONCATで得られた経路のMKPNとAPRTは10m-20m、rCSMPとpCSMPで得られた経路のMKPNとAPRTは2m-4mである。優先順位付け戦略を用いたpCSMPのMKPNとAPRTは、rCSMPのそれよりも約20%低い。

車両密度の影響。2つ目の実験では、グリッドサイズを20×20に固定し、車両密度の変化に対するアルゴリズムの挙動を調べる。なお、$n_p + n_r = n_o$ とする。その結果を図8に示す。前回と同様に、ILPは密度が20%以下のインスタンスを合理的な時間でしか解くことができないが、

他の3つのアルゴリズムは全て最も密度の高いシナリオに取り組むことができる。すべてのアルゴリズムにおいて、C_lの車両密度はMKPNとAPRTに限定的な影響しか与えず、rCSMPとpCSMPはCONCATよりもはるかに品質が良い。低密度シナリオでは、移動に必要な車両が少なくて済むため、車両の回収/駐車に支障をきたす可能性がある。その結果、車両密度が高くなるにつれてANMが増加する。

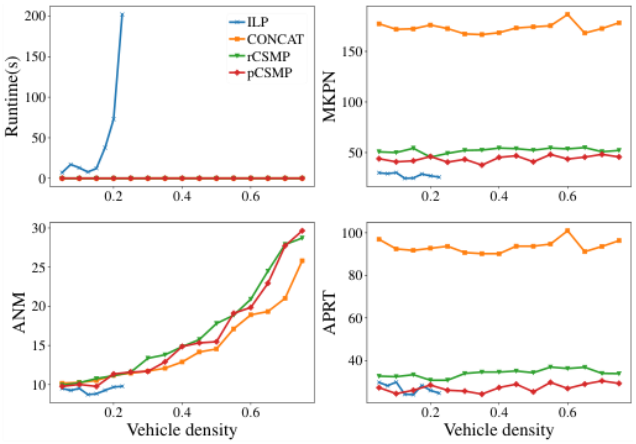

図8. 提案手法の20×20グリッド、車両密度を変化させた場合の実行時間、MKPN、APRT、ANMデータ。

B. CVPRにおけるアルゴリズム性能

ランダム検索とパーキング。連続CSMPを10ポートの12×12グリッドでテストする。各時間ステップにおいて、ポートが利用可能であれば、容量を超えなければ確率p_pでこのポートに駐車する必要がある新車両が存在することになる。そして、確率p_rで、このポートは、ランダムな駐車車両がある場合、その車両を検索するために使用される。以下の3つのシナリオをシミュレートする:

(i). 朝のラッシュアワー。当初は駐車していない。$p_p = 0.6$, $p_r = 0.01$。

(ii). Workday hours. Initially, the garage is full. Request for parking and retrieval are equal: $p_p = p_r = 0.05$.

(iii). 夕方のラッシュアワー。最初はガレージは満杯。検索要求が駐車場を支配する: $p_p = 0.01$, $p_r = 0.6$.

タイムステップの最大数は500とした。これらのシナリオの下で、平均検索時間、平均駐車時間、総移動回数を評価する。その結果を図9に示す。オンラインCSMPは、検索回数が少ないため、朝最高の性能を達成している。一方、3つのシナリオの平均検索時間は2m以下、駐車時間はm以下である。これは、最も密なシナリオとラッシュアワーでも、アルゴリズムが良好な解の質でパスを計画できることを示している。

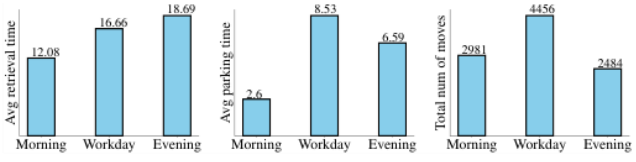

図9. 3つのガレージトラフィックパターンにおけるCVPR性能統計。

シャッフリングの利点。本実験では、シャッフリングの効果(VSPを解くため)を検証する。各車両には、ある人が他の人より早く帰宅する夕方のラッシュアワーに期待される検索優先順序が割り当てられていると仮定する。検索を容易にするために、車両に対して列シャッフル操作を行う。図10(a)～(c)に、最も密な設定における、グリッドサイズの異なるm×mグリッドに対する列シャッフル演算のメイクスパン、平均移動回数、計算時間を示す。シャッフルの経路は1秒以内に計算できるが、シャッフル完了のメイクスパンはm^2に対して線形にスケールし、平均移動回数はmに対して線形にスケールする。2500 120

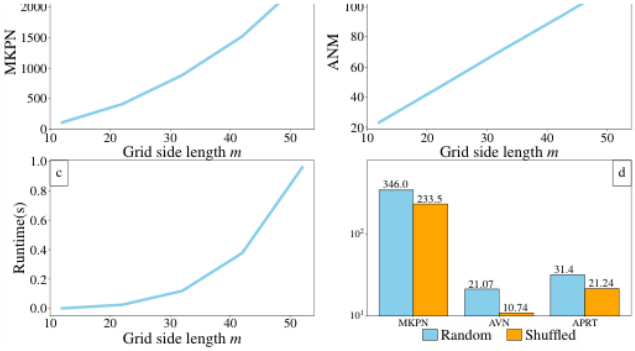

図10. グリッドサイズを変化させたm×mグリッドに対して列シャッフル操作を行った統計量。

シャッフル後、$p_r = 1$, $p_p = 0$の連続CSMPを適用して全車両を取得し、シャッフルを行わない場合と比較する。結果のメイクスパン、1台あたりの平均移動回数、1台あたりの平均検索時間を図10(d)に示す。シャッフルなし構成と比較して、CSMPは30%～50%少ない移動回数と検索時間ですべての車両を検索することができ(すべてのデータに適合させるために対数スケールが使用されていることに注意)、ラッシュアワー検索を予期して車両を再配置することが大きな利点を提供することを示している。

VI. 結論と考察

本研究では、大都市圏における高速/高効率での車両の高密度駐車を可能にすることを目的とした、自動化されたガレージシステムの完全な物理的かつアルゴリズム的な設計を紹介する。我々は、検索と駐車問題をマルチロボットの経路計画問題としてモデル化し、我々のシステムがほぼ100%の車両密度をサポートすることを可能にする。提案するILPアルゴリズムはメイクスパン最適解を提供できるが、CSMPアルゴリズムは高いスケーラビリティを持ち、解の品質も良い。また、後で順序付けされた検索操作のために、非ラッシュ時間中に車両再配置を実行することは非常に有益であることが明確に示されており、これは我々の自動化されたガレージ設計のユニークな高実用性の特徴である。

今後の課題として、MRPP/MAPF研究の最新の進歩を活用し、CSMPのスケーラビリティと柔軟性をさらに向上させる予定である。また、自動化されたガレージの設計を2Dから3Dに拡張し、複数のレベルの駐車をサポートする予定である。最後に、物理的・アルゴリズム的設計を実現する小規模なテストベッドを構築したい。

VII. Appendix

命題IV.1の証明。全密度に対して、Rubik Tableアルゴリズムを適用することができる。列をシャッフルするには、2つの空列を利用することでハイウェイモーションプリミティブを適用し、1つの空列を利用することでラインマージソートモーションプリミティブを適用することができます[20]。各列シャッフルには$O(m_1)$ステップが必要であり、そのような列はm_2個あり、全ての列は1/2個の空列を用いて逐次シャッフルできる。行のシャッフルにも同様のことができる。したがって、どのような再構成も$O(m_1\ m_2)$のメイクスパンを使って行うことができる。

次に、$\Theta(m_1\ m_2)$のエスコートがある場合、ルービックテーブルアルゴリズムを用いて、再構成に$O(m_1 + m_2)$のメイクスパンが必要であることを証明する。重要な点は、すべての行/列を$O(m_1 + m_2)$でシャッフルする方法を見つけることである。

$\lambda m_1\ m_2$ $(0 < \lambda < 1)$ 車両と$(1 - \lambda)m_1\ m_2$ エスコートがあると仮定する。そのために、まずラベルなしMRPPを適用して、開始とゴールの構成を、複数のブロックから構成され、各ブロックが少なくとも1つの空の列または行を持つブロック構成に変換する(図11(d))。簡単のため、ブロックの各黒セルは車両で満たされていると仮定する。そうでない場合は、仮想車両で埋める。ここで、$W = \frac{1}{1-\lambda} = O(1)$ はブロックの幅であり、異なるブロックのシャッフルは並列に実行できる。行シャッフルも同様に$O(m_1)$で行うことができる。

そして、ラベルなし変換が$O(m_1 + m_2)$で完了できることを証明すればよい。変換は、図11に示すように、以下のように行うことができる。まず、初期設定から始めて、すべての車両をできるだけ右方向に動かす。(図11(a)～図11(b))。第2ステップでは、車両を上方に移動させる(図11(b)～図11(c))。構成(c)から構成(d)を得るためには、より多くの車両を持つ行の追加車両は、より少ない車両を持つ行まで列に沿って下方に移動することができる。各列が同じ数の車両を持つ後、すべての車両は左向きの行に沿って移動し、構成(d)を形成することができる。全体として、以下のすべての会議、正面衝突、または垂直衝突が発生する。

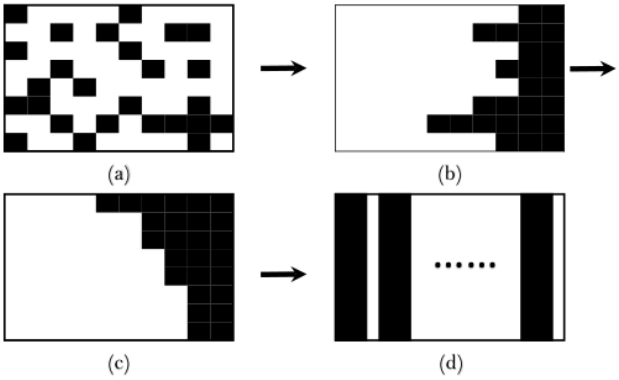

図11. $O(m_1 + m_2)$ ステップで任意の構成を「ブロック」構成にするステップ。(a) 初期設定。(b) (a)から全車両を右方向に移動させた場合の構成。(c) (b)から全車両を上方に移動させて得られる構成。(d) 望ましい通常の「ブロック」構成。

vehicles always move in one direction, and there would be no

したがって、ラベルなし変換は$O(m_1 + m_2)$で完了する。これらを合わせると、$\Theta(m_1\ m_2)$ 個の車両がある場合、VSP の再構成には $O(m_2 + m_1)$ かかると結論づけられる。

下限は、スタートとゴールの間のマンハッタン距離の最大値を計算することで求めることができる。$\Omega(m_1\ m_2)$ 車両のランダムなスタートとゴールの場合、下界は $m_1 + m_2 - o(m_1 + m_2)$ [20] である。したがって、$\Omega(m_1\ m_2)$ 個の車両がある場合、再構成に必要なステップ数は $\Omega(m_1 + m_2)$ となる。

□

References

[1] Yeefung Automation, "First AGV Robotic Parking System in Hong Kong developed by Yeefung," 2021. [Online]. Available: https://www.youtube.com/watch?v=M225gM7rplw

[2] A. K. Nayak, H. Akash, and G. Prakash, "Robotic valet parking system," in *2013 Texas Instruments India Educators' Conference*. IEEE, 2013, pp. 311–315.

[3] M. Ferreira, L. Damas, H. Conceicao, P. M. d'Orey, R. Fernandes, P. Steenkiste, and P. Gomes, "Self-automated parking lots for autonomous vehicles based on vehicular ad hoc networking," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*. IEEE, 2014, pp. 472–479.

[4] J. Timpner, S. Friedrichs, J. Van Balen, and L. Wolf, "K-stacks: high-density valet parking for automated vehicles," in *2015 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2015, pp. 895–900.

[5] M. Nourinejad, S. Bahrami, and M. J. Roorda, "Designing parking facilities for autonomous vehicles," *Transportation Research Part B: Methodological*, vol. 109, pp. 110–127, 2018.

[6] K. R. Gue and B.-S. Kim, "Puzzle-based storage systems," *Naval Research Logistics (NRL)*, vol. 54, 2007.

[7] D. Ratner and M. K. Warmuth, "Finding a shortest solution for the n× n extension of the 15-puzzle is intractable." in *AAAI*, 1986, pp. 168–172.

[8] H. Yu, Y. Yu, and M. de Koster, "Optimal algorithms for scheduling multiple simultaneously movable empty cells to retrieve a load in puzzle-based storage systems," *Available at SSRN 3506480*, 2016.

[9] A. Okoso, K. Otaki, S. Koide, and T. Nishi, "High density automated valet parking via multi-agent path finding," in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2022, pp. 2146–2153.

[10] R. Stern, N. R. Sturtevant, A. Felner, S. Koenig, H. Ma, T. T. Walker, J. Li, D. Atzmon, L. Cohen, T. S. Kumar, *et al.*, "Multi-agent pathfinding: Definitions, variants, and benchmarks," in *Twelfth Annual Symposium on Combinatorial Search*, 2019.

[11] P. Surynek, "A novel approach to path planning for multiple robots in bi-connected graphs," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 3613–3619.

[12] J. Yu and S. M. LaValle, "Structure and intractability of optimal multi-robot path planning on graphs," in *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[13] J. Yu and S. M. LaValle, "Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1163–1177, 2016.

[14] P. Surynek, "An optimization variant of multi-robot path planning is intractable," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 24, no. 1, 2010.

[15] E. Erdem, D. G. Kisa, U. Oztok, and P. Schüller, "A general formal framework for pathfinding problems with multiple agents," in *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[16] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015.

[17] G. Sharon, R. Stern, M. Goldenberg, and A. Felner, "The increasing cost tree search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 195, pp. 470–495, 2013.

[18] M. Barer, G. Sharon, R. Stern, and A. Felner, "Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem," in *Seventh Annual Symposium on Combinatorial Search*, 2014.

[19] R. J. Luna and K. E. Bekris, "Push and swap: Fast cooperative path-finding with completeness guarantees," in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[20] T. Guo and J. Yu, "Sub-1.5 time-optimal multi-robot path planning on grids in polynomial time," in *Robotics: Sciences and Systems*, 2022.

[21] T. Guo, S. W. Feng, and J. Yu, "Polynomial Time Near-Time-Optimal Multi-Robot Path Planning in Three Dimensions with Applications to Large-Scale UAV Coordination," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[22] S. D. Han, E. J. Rodriguez, and J. Yu, "Sear: A polynomial-time multi-robot path planning algorithm with expected constant-factor optimality guarantee," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–9.

[23] J. Yu, "Constant factor time optimal multi-robot routing on high-dimensional grids," *2018 Robotics: Science and Systems*, 2018.

[24] H. Ma and S. Koenig, "Optimal target assignment and path finding for teams of agents," in *AAMAS*, 2016.

[25] H. Ma, T. S. Kumar, and S. Koenig, "Multi-agent path finding with delay probabilities," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[26] M. Szegedy and J. Yu, "On rearrangement of items stored in stacks," in *The 14th International Workshop on the Algorithmic Foundations of Robotics*, 2020.