arXiv:2302.01305v1 [cs.RO] 2 Feb 2023

迈向自动化车库的高效物理设计与算法设计

TengGuo Jingjin Yu

摘要—在大都市地区，停车往往耗时较长，也会影响交通模式，进而影响城市景观。为停车所需的昂贵空间不断减少，推动了自动化机械停车系统的发展。与每个停车区只有一排或两排车辆的传统车库相比，自动化车库可将多排车辆叠放在一起，以满足更大的停车需求。尽管这种多排布局减少了停车空间，却使停车和取车变得更加复杂。在本研究中，我们提出了一种自动化车库设计，可实现近100% 的停车密度。我们将多辆车的停车与取车建模为一类特殊的多机器人路径规划问题，并提出了相应算法，用于处理自动化车库中的所有常见操作，包括：(1) 用于同步停车与取车的最优算法和近优方法，以寻找可行且高效的解决方案；(2) 一种新颖的车辆调度机制，可重新安排车辆，以便在高峰时段进行预约取车。我们开展了全面的仿真研究，结果表明，所提出的方法有望应用于规模大、密度高的现实停车场景。

I. 引言

自动化停车系统（车库）的发明，有助于解决空间成本高昂地区的停车问题，例如市中心和其他人口密集地区。如今，停车位日益稀缺且价格昂贵；在曼哈顿，一个停车位的价格轻易就可能超过 200，000 美元。因此，开发能够支持高密度停车、节省空间且更加便捷的车库，出于经济和效率方面的考虑，具有很大吸引力。

在自动化车库中，人类驾驶员只需在指定的 I/O（输入/输出）港口交接（取回）车辆，无需处理停车过程。此类系统中的车辆无需预留开门所需的周边空间，因此可以停得更近。将车辆移至停车位或出入口是此类系统的核心功能。一种解决方案是使用机器人代客泊车系统移动车辆。此类系统已投入商业使用，例如香港的 HKSTP [1] 。在此类系统中[2]，，车辆的停放方式可能导致车辆相互阻挡，因此取出指定车辆时需要进行多次重新调度。遗憾的是，关于这些系统的运行表现（例如停车/取车效率），几乎找不到相关信息。其他解决方案则专注于自动驾驶车辆的停车。在此类自动化车库中，车辆能够自行驶入停车位和出入口。这使得

G. Teng 和 J. Yu 任职于美国新泽西州皮斯卡塔韦的罗格斯大学（新泽西州立大学）计算机科学系。电子邮箱：{teng.guo, jingjin.yu}@rutgers.edu。本研究部分获得美国国家科学基金会 IIS-1845888 项目和亚马逊研究奖的资助。

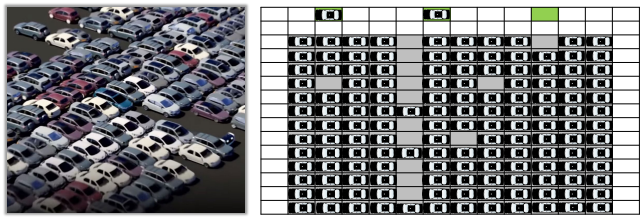

图 1。左：设想中的自动化车库系统密度水平示意图。右：基于网格的抽象，其中包含三个用于车辆送达和取回的输入/输出端口。该系统更为灵活，但节省空间的优势有限，而且要求车辆具备自主能力。

近期，许多高效的多机器人路径规划算法相继提出，使得利用多个机器人代客泊车系统来降低停车和取车成本成为可能。本文研究基于多机器人的停车与取车问题，提出一套完整的自动化车库设计，实现接近 100% 的停车密度，并开发了用于高效运行该车库的配套算法。

结果与贡献。我们的主要研究成果与贡献如下。在自动化车库设计方面，我们提出了批量车辆停放与取回（BVPR）和连续车辆停放与取回（CVPR）问题，用于建模此类车库所需的关键操作，从而推动自动化车库系统未来的理论与算法研究。

在算法方面，我们研究了一种能够在 $m_1 \times m_2$ 网格地图上实现高达$(m_1-2)(m_2-2)/m_1m_2$停车密度，并支持多辆车辆停放与取车的系统；对于大型车库，该系统的停车密度接近100%。利用该系统类似网格的规律性，我们提出了一种基于整数线性规划的最优方法，以及一种基于顺序规划的快速次优算法。我们的次优算法具有很强的可扩展性，同时能够保持较高的解质量，因此适用于大规模应用。我们还提出了一种车辆调度机制，可在非高峰时段重排车辆，以便在高峰时段快速取车，前提是能够预先确定取车顺序。在车库接近满载时，我们的重排算法执行这些打乱操作的总时间成本为 $O(m_1m_2)$。

相关工作。 研究人员提出了多种提高高密度停车效率的方案。人们研究了许多用于自动驾驶车辆的系统[3]–[5],，在这些系统中，车辆由中央控制器统一停放，可以堆叠成多排，并可能相互阻挡。这些设计可将停车容量提高多达 50%。然而，由于堵塞，车辆的取出过程会变得非常复杂，并且在很大程度上受到自动驾驶车辆机动性的影响

自动驾驶车辆的。

由于大多数车辆无法自动驾驶，基于机器人代客泊车的高密度停车系统可能是更合适的选择。基于拼图的存储系统（PBS）或基于网格的穿梭车系统最初由 [6], 提出，是最有前景的高密度存储系统之一。在此类系统中，存储单元可以是自动导引车或穿梭车，可沿四个基本方向移动。系统中必须至少有一个空单元格（护送车辆）。要取出车辆，必须利用护送车辆将目标车辆移至输入/输出端口。这与 15数码问题类似，而已知要以最优方式求解该问题是 NP难的 [7]。 [6], [8]提出了使用单个护送车辆和多个护送车辆取出一辆车的最优算法。然而，这些方法一次只考虑取出一辆车。此外，平均取车时间可能远长于传统的巷道式方案。为兼顾容量需求与取车效率，我们建议增加护送车辆和输入/输出端口，并利用最新的先进多机器人路径规划（MRPP）算法 [9], 以支持同时取出和停放多辆车。

MRPP 已得到广泛研究。在静态或一次性场景中，[10],给定一个图环境和若干机器人，其中每个机器人都有唯一的起始位置和目标位置，任务是为所有机器人寻找从起点到目标的无碰撞路径。已有研究证明，就最小化完工时间或成本总和而言，求解一次性 MRPP 的最优解属于 NP难问题 [11], [12]。MRPP 求解器可分为最优和次优求解器。最优求解器要么将 MRPP 规约为其他已得到充分研究的问题，例如 ILP [13], SAT [14] 和答案集编程 [15] ，要么使用搜索算法在联合状态空间中搜索最优解 [16], [17]。由于 NP难性，最优求解器不适合求解大规模问题。有界次优求解器 [18] 在仍能提供较强最优性保证的同时，实现了更好的可扩展性。然而，它们的扩展性仍然较差，尤其是在高密度环境中。也存在用于求解大规模 MRPP 的多项式时间算法 [19], ，但其代价是解的质量下降。其他 $O(1)$时间最优的多项式时间算法 [20]–[23]主要致力于最小化完工时间，这对于连续场景而言并不十分适合。

组织结构。本文其余部分的组织结构如下：第二节介绍预备知识，包括车库设计。第三节至第四节给出自动化车库的运行算法。第五节对车库系统进行全面评估和讨论，并在第六节总结全文。

II. 预备知识

A. 车库设计规范

在本研究中，基于网格的自动化车库是一个四连通的 $m_1 \times m_2$ 网格 $\mathcal{G}$（$\mathcal{V}$， $\mathcal{E}$）（见图1）。网格的顶部边界分布着 n_o 个输入 / 输出端口（下文简称为出入口），用于将车辆送入停车或取出指定的已停放车辆。某一时刻，一个出入口只能用于取车或停车中的一种用途。车辆必须停放在停车位，即下方中央（m_1-2）×（m_2-2）子网格中的一个单元格。一旦空闲车位被停放车辆，该车位便成为可移动障碍物。 $\mathcal{O}=\{o_1,...,o_{n_o}\}$ 是出入口的集合， $\mathcal{P}=\{p_1,...,p_{|\mathcal{P}|}\}$ 是停车位的集合。

B. 批量车辆停放与取回（BVPR）

批量车辆停放与取回，简称 BVPR，旨在以批量模式优化停车与取车。在单个批次中，有 n_p 辆车辆需要停放， n_r 辆车辆需要取回， n_l 辆停放车辆需要保留。记 $\mathcal{C}=\mathcal{C}_p\cup\mathcal{C}_r\cup\mathcal{C}_l$ 为所有车辆的集合。任何时刻都不能超过最大容量，即 $|\mathcal{C}|<|\mathcal{P}|$。时间被离散为多个时间步，由自动导引车/穿梭车承载的多辆车辆可以同时移动。每个时间步中，每辆车可以向左、向右、向上、向下移动，或停留在当前位置。应避免车辆之间发生碰撞：

1) 相遇碰撞。两个车辆在任何时间步都不能位于同一网格点： $\forall i,j\in\mathcal{C}, v_i(t)\neq v_j(t)$；2) 迎面碰撞。两个车辆不能沿同一条边以相反方向行驶并交换位置：$\forall i,j\in\mathcal{C}, (v_i(t)=v_j(t+1)\wedge v_i(t+1)=v_j(t))=\text{false}$；3) 垂直跟随碰撞（见图 2）。当两个车辆的移动方向相互垂直时，前一辆车辆不能被后一辆车辆跟随。将 $\hat{e_i}(t)=v_i(t+1)-v_i(t)$ 记为车辆 i 在时间步 t 的移动方向向量，则 $\forall i, j\in\mathcal{C}, (v_i(t+1)=v_j(t)\wedge\hat{e_i}(t)\perp\hat{e_j}(t))=\text{false}$。

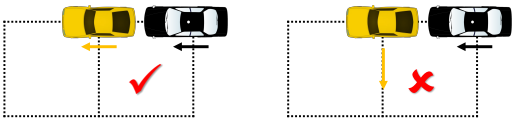

图 2. 允许车辆沿同一方向平行移动（左），但禁止车辆垂直跟随（右）。

与某些多机器人路径规划问题的表述不同 [10],，我们需要考虑垂直、跟随和碰撞，这使问题更难求解。任务是为每辆车辆找到一条无碰撞路径，将其从当前位置移动到目标位置。具体而言，对于需要取车的车辆，其目标位置是指定的港口。对于其他车辆，目标位置可以是任意一个停车位。采用以下标准评估解的质量：

1）平均停放/取回时间（APRT）：取回或停放一辆车辆所需的平均时间。2）完工时间（MKPN）：将所有车辆移动到目标位置所需的时间。3）每项任务的平均移动次数（ANM）：所有车辆移动距离之和除以 $\mathcal{C}_p\cup\mathcal{C}_r$中的车辆总数。

一般来说，这些目标会形成帕累托前沿 [12] ，而且并非总能同时优化其中任意两个目标。

C. 连续车辆停放与取回（CVPR）

CVPR 是车辆停车与取车问题的连续版本。它继承了 BVPR 的大部分结构，

但也有几个关键区别。在这一表述中，我们作出如下假设：车辆 $i \in \mathcal{C}_r$ 到达其目标出入口时，将从环境中移除。会有新车辆出现在出入口，需要将其停放（且不超过容量），也会有新的车辆取车请求。此外，当某个出入口正用于取车时，在取车任务完成之前，其他用户不能在该出入口停车。除时间范围为无限或固定的 MKPN 外，仍可使用这三项标准来评估解的质量。

D. 车辆重排问题（VSP）

在现实中的车库里，通常会有非高峰时段（例如早晨高峰时段过后），此时系统已达到满载状态，而取车请求很少。如果已知稍后某个时段（例如下午高峰时段）车辆的取车顺序，我们就可以利用这一信息重新排列车辆，以便之后取车。这个问题可建模为单次多机器人路径规划。给定初始配置 X_I 和目标配置 X_G，我们需要找到无碰撞路径来完成重新配置。目标配置根据车辆的取车时间顺序确定；预期较早取出的车辆应停在离出入口较近的位置，以免被稍后离开的其他车辆挡住。

III. 求解 BVPR

A. 整数线性规划（整数线性规划）

基于 [13], [24], 中的网络流方法，我们将 BVPR 化简为多商品最大流问题，并使用整数规划求解。 $\mathcal{C}_r$ 中的车辆有特定的目标位置（出入口），必须作为不同的商品流处理。另一方面，由于 $\mathcal{C}_p \cup \mathcal{C}_l$中的车辆可以停在任意一个停车位上，因此可将它们视为单一商品流。假设一个实例可以在 T 步内求解，我们构造一个 T步的时间扩展网络，如图3所示。

T步时间扩展网络是一个具有有向单位容量边的有向网络$\mathcal{N}_T = (\mathcal{V}_T, \mathcal{E}_T)$。该网络 $\mathcal{N}_T$ 包含原始图中各顶点 $\mathcal{V}$ 的 $T+1$ 个副本。顶点 $u \in \mathcal{V}$ 在时间步 t 的副本记作 u_t。在时间步 t，如果满足条件$u, v \in \mathcal{E}$ 或 $v = u$，则向 $\mathcal{E}_T$ 添加一条边（u_t, v_{t+1}）。对于 $i \in \mathcal{C}_r$，我们可以在顶点 s_{i0} 处提供一个单位的商品类型 i 供给量，其中 s_i 是 i的起始顶点。为确保车辆 i 到达其港口，我们添加一条反馈边，将其在 T 的目标顶点 g_i 连接到其源节点 s_{i0}。对于需要停放的车辆，我们创建一个辅助源节点 α 和一个辅助汇节点 β。对于每个 $i \in \mathcal{C}_p \cup \mathcal{C}_l$，我们添加一条单位容量边，将节点 α 连接到其起始节点 s_{i0}。由于车辆可以停放在任意一个停车位上，对于任意顶点$u \in \mathcal{P}$，我们添加一条单位容量边，将 u_T 与β连接起来。可以在节点 α处提供一个单位的商品流，其商品类型对应于 $\mathcal{C}_p \cup \mathcal{C}_l$ 中的车辆。

要使用整数线性规划求解多商品最大流，我们创建一组二元变量 $X = \{x_{iuvt}\}$, $i = 0, \ldots\ldots n_r$, $(u, v) \in \mathcal{E}$或 $u = v$, $0 \le t \le T$; 变量设为真表示最终解中使用了对应的边。整数线性规划的建模如下。

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

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

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

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

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

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

$$x_{iuvt} = \begin{cases} 0 & \text{if } i \text{ does not traverse edge } (u, v) \text{ at } t \\ 1 & \text{if } i \text{ traverses edge } (u, v) \text{ at } t \end{cases} \tag{7}$$

在公式 (1) 中，我们最小化时间范围 T 内所有车辆的移动总次数。公式 (2) 规定了每个网格点处的流守恒约束。公式 (3) 规定了用于避免相遇碰撞的顶点约束。在公式 (4) 中，不允许车辆沿同一条边反向通行。公式 (5) 规定了禁止垂直跟随冲突的约束。如果针对T步时间扩展网络的整数规划可行，则得到解。否则，我们逐步增加 T，直到得到可行解。整数规划存在解时的最小 T 即为最小完工时间。因此，ILP 通过将移动总次数作为次要目标进行最小化，求得完工时间最优解。

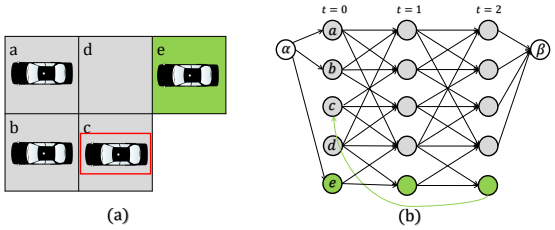

图3。（a）一个包含4个停车位和一个港口的示例 BVPR 实例。红色矩形内的车辆需要被取出，而其他车辆应停放在灰色区域内。（b）从该 BVPR 实例缩减得到的2步流网络。

B. 高密度规划的高效启发式方法

整数线性规划可以找到完工时间最优解，但其扩展性较差。我们希望设计一种快速算法，能够迅速求解密集型 BVPR 实例，同时只牺牲少量解的质量。我们提出的算法基于单车辆取车和停车运动原语。

1）单车停放/取车的运动原语：无论解的质量如何，都可以通过在 $\mathcal{C}_r \cup \mathcal{C}_p$ 中为每辆车辆进行顺序规划来求解 BVPR。具体而言，每一轮我们只考虑为给定车辆 $i \in \mathcal{C}_r \cup \mathcal{C}_p$ 完成一个单独的任务，即将其移动 i 至港口或某个停车位。车辆 i 到达

目的地后，再解决下一个任务。图4和图5中的示例展示了该方法的操作过程，其中停车位已达到最大容量。

对于图4中的检索场景，需要取出红色矩形标记的车辆。要实现左侧虚线所示的直观路径，应先将挡路的车辆移开。利用两列空位，只需将这些车辆向左或向右移动一步即可轻松做到。采用这种运动原语，总能成功取出车辆，且不会发生死锁。

对于图5中的停车场景，我们需要将车辆停入绿色端口。为此，我们首先采用贪心策略寻找一个空位（护送车辆）。利用车辆并行移动机制，护送车辆可以先在一个时间步内移动到待停车车辆所在的列，然后再用一个时间步移动到该车辆正下方的位置。之后，该车辆便可直接驶向作为其目的地的护送车辆。

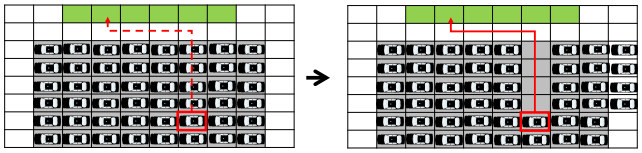

图4. 取车运动原语。

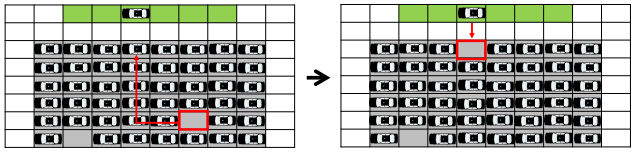

图5. 车辆停车运动原语。

顺序规划总能在存在解时返回一个解；然而，当需要执行多个停车与检索请求时，顺序规划得到的解表现不佳，具体体现为 MKPN 和 ARPT 指标较差。

2）通过 MCP （CSMP）耦合单个运动原语：最小通信策略（MCP） [25] 是一种鲁棒的多机器人执行策略，可在处理意外延迟时避免让未受影响的机器人停下。在执行过程中，MCP 保持机器人访问各个顶点的顺序与原始规划一致。当机器人 i 即将执行一个移动动作并进入顶点 v 时，MCP 会检查该机器人是否是按照原始规划下一个进入该顶点的机器人。如果计划由另一台机器人 j下一个进入顶点 v，那么 i 会在当前位置等待，直到 j 离开 v。

我们使用 MCP 为通过顺序规划得到的计划引入并发。算法如算法1和算法2所述。算法1介绍了 CSMP 的框架。首先，我们依次为 $\mathcal{C}_r \cup \mathcal{C}_p$ 中的所有车辆寻找初始计划（第4–6行）。获得路径后，我们移除所有等待状态，并用队列列表记录每个顶点处车辆访问的顺序（第7行）。然后，我们进入循环，使用 MCP 执行计划，直到所有车辆完成任务并到达目的地（第8–15行）。在算法2中，如果根据 v_i 的原始顺序， i 是下一个进入顶点的车辆，我们会检查当前是否有车辆位于 v_i。如果没有，就让 i 进入 v_i。如果另一辆车辆j 当前占据着 v_j，我们会通过递归调用函数 MCPMove，检查 j 下一步是否要移动到其下一个顶点 v_j。如果 j 下一步要移动到 v_j ，且 i, j 的移动方向不垂直，就让车辆 i 进入顶点 v_i。否则，i 应在 u_i等待。该算法从构造上保证无死锁；由于篇幅限制，我们省略相对直接的证明。

命题 III.1. CSMP 无死锁，并且只要存在可行解，就总能在有限时间内找到一个。

算法 1： CSMP

```
1  函数 CSMP():
2      对每个 v ∈ V, VOrder[v] ← Queue()
3      InitialPlans ← {}
4      对于 i ∈ C_p ∪ C_r，执行
5          SingleMP(i,InitialPlans)
6      预处理(InitialPlans, VOrder)
7      当为真 时执行
8          对于 i ∈ C，执行
9              mcpMoved ← Dict()
10             MCPMove (i)
11         如果 AllReachedGoal()=true则
12             跳出
```

算法 2： MCPMove

```
1  函数MCPMove (i):
2      如果 i属于 mcpMoved 则
3          返回 mcpMoved[i]
4      u_i ← i的当前位置
5      v_i ← i的下一个位置
6      如果 i = VOrder[v_i].front() 则
7          j ← 当前位于 v_i 的车辆
8          如果 j=为 None 或 (MCPMove(j)=为 true 且
             (u_i, v_i) ⊥̸ (u_j, v_j)) 那么
9              移动 i到 v_i
10             VOrder[v_i].popfront()
11             mcpMoved[i]=真
12             返回 真
13     令 i 在 u_i等待
14     mcpMoved[i]=假
15     返回 假
```

3) 优先级排序：顺序规划无论规划顺序如何，都能找到一个解。不过，优先级会影响解的质量。我们可以尽可能先规划停车，而不是按照随机的优先级顺序进行规划（算法1第4—6行），因为单车停放只需两步。等 $\mathcal{C}_p$ 中的所有车辆都停放完成后，我们再应用 SingleMP 来取出车辆。对于 $\mathcal{C}_r$中的车辆，我们先对距离港口较近的车辆应用 SingleMP，使它们能够更早到达目标位置，并且不会阻挡下排车辆。

C. 复杂度分析

本节分析 CSMP 的时间复杂度和解的最大完成时间上界。在 SingleMP 中，为了停放/取出一辆车辆，我们假设 n_b 辆车辆可能造成堵塞，需要移开。显然，$n_b < n$ 其中 $n = n_p + n_r + n_l$。因此，

使用 SingleMP 为 $\mathcal{C}_r \cup \mathcal{C}_p$ 中的所有车辆计算路径的复杂度上界为 $(n_p+n_r)n$。SingleMP 计算出的每条单车路径的路径长度不超过 m_1+m_2。通过串联所有单车路径得到的路径，其最大完成时间上界为$n_r(m_1+m_2)+2n_p$。这意味着 MCP 至多需要 $n_r(m_1+m_2)+2n_p$次迭代。因此，解的最大完成时间上界为 $n_r(m_1+m_2)+2n_p$。在 MCP 的每次循环中，我们实质上都会在一个包含 $n=n_p+n_r+n_l$ 个节点的图上运行深度优先搜索，并遍历所有节点，其时间复杂度为 $O(n)$，因此 CSMP 的时间复杂度为 $O(n(n_rm_1+n_rm_2+2n_p))$。总之，CSMP 的时间复杂度上界为$O(n(n_rm_1+n_rm_2+2n_p))$，而最大完成时间上界为 $n_r(m_1+m_2)+2n_p$。

D. 将CSMP 扩展至CVPR

CSMP 可以轻松调整以解决 CVPR。与 BVPR 版本类似，我们在每个时间步为每辆车辆调用 MCPMove。当某个时间步出现新请求时，我们使用 SingleMP 计算相关车辆的路径，并更新顶点访问顺序的信息。也就是说，当我们对车辆 $i \in \mathcal{C}_p \cup \mathcal{C}_r$ 应用 SingleMP 时，如果车辆 j 将在时间步 t' 访问顶点 u，那么我们就将 i 推入队列 $VOrder[u]$，其中队列始终按照 u 的进入时间排序。这样一来，MCP 就会在维持访问顺序的同时执行计划。此前已规划的车辆不会受到新请求的影响，并会继续执行原计划。这种方法的主要缺点是，由于访问顺序固定，其解的质量通常不如重新规划。

IV. 通过 RUBIK TABLES 求解 VSP

VSP 本质上是在求解静态/单次多机器人路径规划。在一个$m_1 \times m_2$ 网格上，可以使用 Rubik Table 算法 [26], 求解，最多需要 $2(m_2-2)$ 次列打乱和 (m_1-2) 次行打乱。如图 6 所示，我们可以利用两列相邻的列，以相当高效的方式打乱给定列中的车辆，仅需 $O(m_1)$ 步 [20]。行打乱同理。根据停放车辆的数量，可以同时执行一次或多次行/列打乱。我们有（由于篇幅有限，省略了直接证明）

命题 IV.1。VSP 在车库容量满载时可通过 $O(m_1m_2)$完工时间求解；当车库有 $\Theta(m_1m_2)$ 空位且停放了$\Omega(m_1m_2)$辆车辆时，可通过 $O(m_1+m_2)$ 完工时间求解。相比之下，当停放$\Omega(m_1m_2)$辆车辆时，求解所需的完工时间

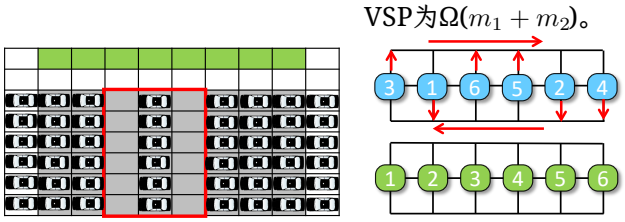

图6. 单行/列“重排”机制示意图。

V. 评估

本节对所提出的算法进行评估。所有实验均在主频为 3.0GHz 的 Intel® Core™ i7-9700 中央处理器上进行。除非另有说明，每个数据点均为随机生成的实例运行 20 次后的平均值。整数线性规划使用 C++ 实现，其他算法则使用 CPython 实现。模拟视频可在 https://youtu.be/XPpOB5f7CzA 查看。

A. BVPR 上的算法性能

网格大小变化。在第一个实验中，我们在网格边长各异的 $m \times m$ 网格上评估所提出的算法，并采用最密集的场景：系统中有 $(m-2)^2$辆车，且所有出入口均用于停车或取车（$n_p+n_r=n_o$）。结果见图7。CONCAT 是一种简单拼接单车路径的方法。在 rCSMP 中，我们按照随机优先级顺序对车辆应用 SingleMP；而在 pCSMP 中，我们应用优先级策略。

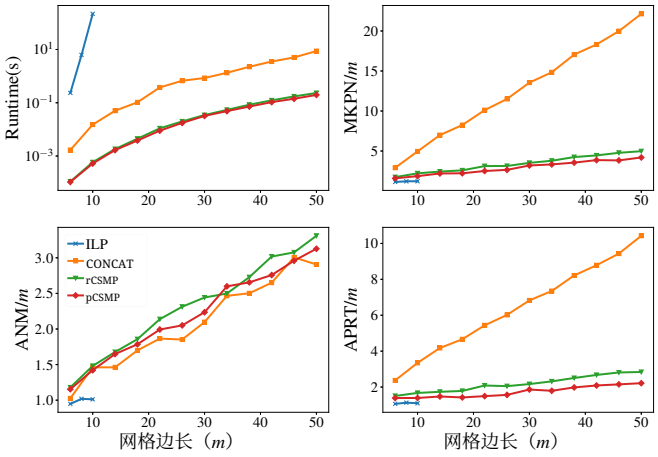

图7. 所提出方法在最密集场景下、针对 $m \times m$ 网格的运行时间、MKPN、APRT 和 AVN 数据。

在这些方法中，整数线性规划在 MKPN、ANM 和 APRT 方面的解的质量最佳，这在预料之中，因为它能保证最优性。然而，整数线性规划的可扩展性最差，在 $m \leq 10$ 和 $n \leq 64$时便达到极限。CONCAT、rCSMP 和 pCSMP 的可扩展性强得多，能够在几秒内求解包含 2304 辆车的 50×50 网格实例。由于 CONCAT 只是拼接单车路径，因此与 rCSMP 和 pCSMP 相比，生成的路径非常长；我们观察到，MCP过程大幅提高了并发能力，从而显著改善了解的质量。CONCAT 得到的路径，其 MKPN 和 APRT 可达 $10m$-$20m$，而 rCSMP 和 pCSMP 得到的路径，其 MKPN 和 APRT 为 $2m$-$4m$。采用优先级策略的 pCSMP，其 MKPN 和 APRT 比 rCSMP 低约 20%。

车辆密度的影响。在第二个实验中，我们研究了车辆密度变化时算法的表现，并将网格大小固定为 20×20。我们仍然令 $n_p+n_r=n_o$。结果如图8所示。与前一种情况一样，整数线性规划只能求解密度低于 20% 的实例，

在合理时间内，而另外三种算法都能应对最密集的场景。对于所有算法，$\mathcal{C}_l$ 中的车辆密度对 MKPN 和 APRT 的影响有限，而 rCSMP 和 pCSMP 的效果远优于 CONCAT。在低密度场景中，需要移动的车辆较少，这可能会导致取车/停车时发生堵塞。因此，ANM 会随着车辆密度的增加而增加。

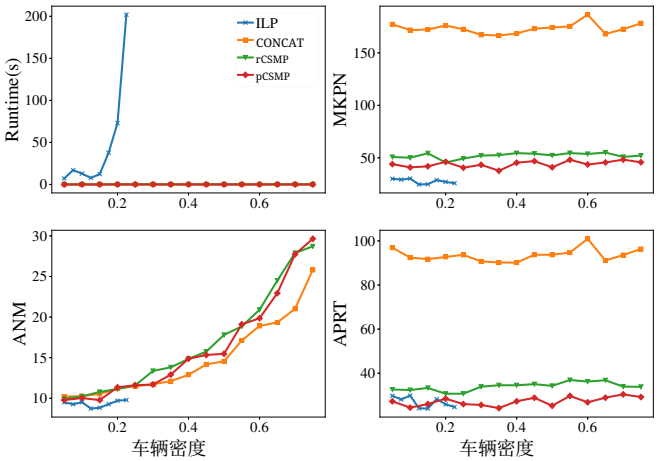

图8. 所提出方法在车辆密度不同的 20×20 个网格上的运行时间、MKPN、APRT 和 ANM 数据。

B. CVPR 上的算法性能

随机取车和停车。我们在一个包含 10 个出入口的 12×12 网格上测试连续 CSMP。每个时间步中，如果某个出入口可用，则以 p_p 的概率有一辆需要停放的新车辆出现在该出入口（前提是车辆数量未超过容量）。此外，以 p_r 的概率，该出入口会被用于取出一辆随机停放的车辆（如果有车辆停放）。我们模拟以下三种情形：

(i).早晨高峰时段。初始时，车库内没有停放车辆。停车请求远多于取车请求：$p_p = 0.6$，$p_r = 0.01$。

(ii). 工作日时段。初始时，车库已满。停车和取车请求数量相等：$p_p = p_r = 0.05$。

(iii). 晚高峰。初始时，车库已满。取车请求远多于停车请求：$p_p = 0.01$，$p_r = 0.6$。

最大时间步数设为 500。在这些场景下，我们评估平均取车时间、平均停车时间和移动总次数。结果如图 9 所示。由于取车次数较少，OnlineCSMP 在早晨的表现最佳。另一方面，所有三种场景下的平均取车时间均小于$2m$，停车时间均小于 m。这表明，即使在最密集的场景和高峰时段，该算法也能够规划出解的质量良好的路径。

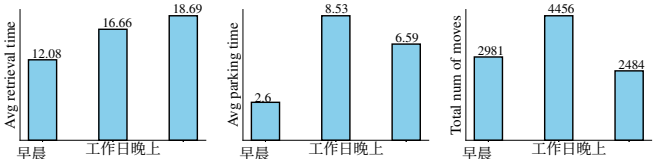

图 9. CVPR 在三种车库交通模式下的性能统计。**重排的优势**。在本实验中，我们考察重排的效果（用于求解 VSP）。我们假设每辆车都被分配了一个取车优先级顺序，这符合晚高峰时的预期，因为有些人会比其他人更早回家。我们对车辆执行列打乱操作，以便于取车。图 10(a)-(c) 展示了在最密集设置下，对具有不同网格大小的 $m \times m$ 网格执行列打乱操作的完工时间、平均移动次数和计算时间。虽然可以计算重排路径

不到 1 秒内，完成打乱操作的完工时间相对于 m^2 呈线性增长，平均移动次数相对于 m 呈线性增长。

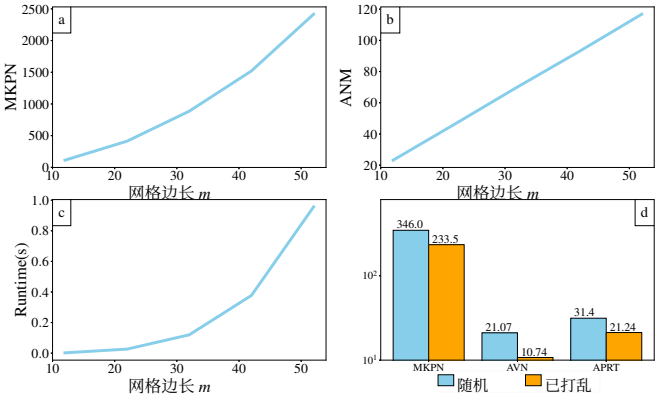

图 10. 在网格大小各异的 $m \times m$ 网格上执行列打乱操作的统计结果。

打乱后，采用 $p_r = 1$，$p_p = 0$ 连续 CSMP 取出所有车辆，并与未执行打乱操作的情况进行比较。结果完工时间、每辆车的平均移动次数以及每辆车的平均取车时间如图 10(d) 所示。与未打乱的配置相比，CSMP 能够以少 30%–50% 的移动次数和取车时间取出所有车辆（注意，图中采用了对数刻度以容纳所有数据），这表明，为应对高峰时段取车而提前重新排列车辆能够带来显著益处。

六、结论与讨论

本文提出了自动化车库系统完整的物理与算法设计，旨在使大都市地区能够以高速度和高效率实现车辆的高密度停车。我们将取车与停车问题建模为多机器人路径规划问题，使系统能够支持接近 100% 的车辆密度。所提出的整数线性规划算法能够提供完工时间最优解，而 CSMP 算法具有很强的可扩展性，且解的质量良好。研究还明确表明，在非高峰时段重新排列车辆，以便之后按序执行取车操作，能够带来显著益处；这是我们自动化车库设计的一项独特且实用性很高的功能。

未来的工作中，我们计划进一步提升 CSMP 的可扩展性和灵活性，并可能借助 MRPP/MAPF 研究的最新进展。我们还计划将自动化车库设计从二维扩展到三维，以支持多层停车。最后，我们希望建造小型测试平台，以实现这些物理和算法设计。

VII. 附录

命题 *IV.1* 的证明。 对于满密度情形，可以在此应用 Rubik Table 算法。要打乱一列，可以利用两列空列应用高速公路运动基元，或利用一列空列应用行归并排序运动基元 [20]。每列打乱需要 $O(m_1)$ 步，共有 m_2 列，因此可以使用一列或两列空列依次打乱所有列。打乱一行也可以采用相同的方法。因此，任何重配置都可以在 $O(m_1m_2)$ 的完工时间内完成。

接下来，我们证明，使用 Rubik Table 算法，当有 $\Theta(m_1m_2)$辆护送车辆时，重新配置需要 $O(m_1+m_2)$ 的完工时间。关键在于找到一种方法，在 $O(m_1+m_2)$ 中打乱所有行/列。

假设我们有 $\lambda m_1m_2(0<\lambda<1)$辆车辆和 $(1-\lambda)m_1m_2$ 辆护送车辆。为此，我们首先应用无标签多机器人路径规划，将起始配置和目标配置转换为由若干块组成的块配置 （图 11(d)），且每个块至少有一行或一列空位。为简单起见，我们假设块中的每个黑色单元格都由一辆车辆占据；否则，就用虚拟车辆填充。不难看出，每个块中的所有列都可以在 $O(Wm_2)$ 步内完成打乱，其中 $W=\frac{1}{1-\lambda}=O(1)$ 是块的宽度，并且不同块中的打乱操作可以并行进行。 行打乱同理，可以在 $O(m_1)$ 内完成。

因此，我们只需证明无标记转换可以在 $O(m_1+m_2)$ 内完成。转换可以按照图 11 所示的以下方式进行。首先，从初始配置出发，将所有车辆尽可能向右移动。（图 11(a) 至图 11(b)）。第二步，将车辆向上移动（图 11(b) 至图 11(c) ）。要从配置 (c) 得到配置 (d)，车辆较多的行中多出的车辆可以沿列向下移动到车辆较少的行中。当每一行的车辆数量都相同时，所有车辆都可以沿各行向左移动，从而形成配置 (d)。在整个过程中，所有的

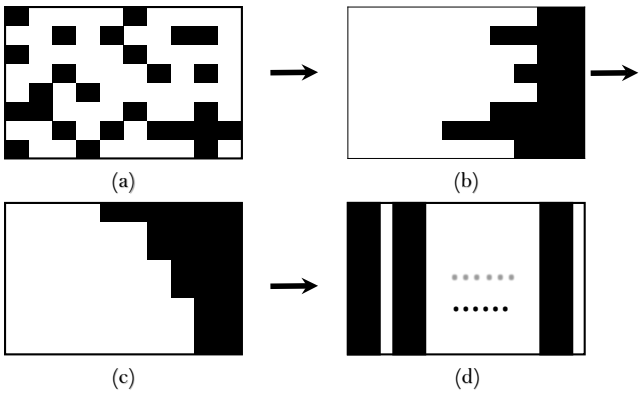

图 11. 在 $O(m_1+m_2)$ 步内将任意配置转换为“块”配置，以便应用 Rubik Table 的结果。(a) 初始配置。(b) 将所有车辆从 (a) 向右移动后得到的配置。(c) 将所有车辆从 (b) 向上移动后得到的配置。(d) 所需的规则“块”配置。

车辆始终朝一个方向行驶，因此不会发生相遇、迎面冲突或垂直跟随冲突。因此，无标记转换可在 $O(m_1+m_2)$ 内完成。综合以上结果，我们得出，当有 $\Theta(m_1m_2)$ 辆车辆时，VSP 的重新配置需要 $O(m_2+m_1)$。

下界可通过计算起点与目标点之间的最大曼哈顿距离得到。对于具有 $\Omega(m_1m_2)$ 辆车辆的随机起点和目标点，下界为 $m_1+m_2-o(m_1+m_2)$ [20]。因此，当车辆数量为 $\Omega(m_1m_2)$ 时，重配置所需的步数为 $\Omega(m_1+m_2)$。

□

参考文献

[1] Yeefung Automation, 《由 Yeefung 开发的香港首个 AGV 机器人停车系统》, 2021年。 [在线]。网址: https://www.youtube.com/watch?v=M225gM7rplw [2] A. K. Nayak、H. Akash 和 G. Prakash, 《机器人代客泊车系统》, 载于*2013*年德州仪器印度教育工作者会议。IEEE, 2013年, 第311–315页。[3] M. Ferreira、L. Damas、H. Conceicao、P. M. d'Orey、R. Fernandes、P. Steenkiste 和 P. Gomes, 《基于车载自组织网络的自动驾驶车辆自主停车场》, 载于*2014*年*IEEE*智能车辆研讨会论文集。IEEE, 2014年, 第472–479页。[4] J. Timpner、S. Friedrichs、J. Van Balen 和 L. Wolf, 《K-堆栈: 自动驾驶车辆的高密度代客泊车》, 载于*2015*年*IEEE*智能车辆研讨会 (*IV*)。IEEE, 2015年, 第895–900页。[5] M. Nourinejad、S. Bahrami 和 M. J. Roorda, 《自动驾驶车辆停车设施设计》, 《交通研究*B*辑: 方法论》, 第109卷, 第110–127页, 2018年。[6] K. R. Gue 和 B.-S. Kim, 《基于拼图的存储系统》, 《海军研究后勤》 (*NRL*), 第54卷, 2007年。[7] D. Ratner 和 M. K. Warmuth, 《求解 n× n 扩展版 15数码问题的最短解是棘手的》。载于*AAAI*, 1986年, 第168–172页。[8] H. Yu、Y. Yu 和 M. de Koster, 《在基于拼图的存储系统中调度多个可同时移动的空单元格以取出载荷的最优算法》, 可于 *SSRN 3506480* 获取, 2016年。[9] A. Okoso、K. Otaki、S. Koide 和 T. Nishi, 《通过多智能体路径搜索实现高密度自动代客泊车》, 载于*2022*年*IEEE*第*25*届智能交通系统国际会议 (*ITSC*)。IEEE, 2022年, 第2146–2153页。[10] R. Stern、N. R. Sturtevant、A. Felner、S. Koenig、H. Ma、T. T. Walker、J. Li、D. Atzmon、L. Cohen、T. S. Kumar、等, 《多智能体路径搜索: 定义、变体与基准》, 载于第十二届组合搜索年度研讨会, 2019年。[11] P. Surynek, 《一种在双连通图中进行多机器人路径规划的新方法》, 载于*2009*年*IEEE*机器人与自动化国际会议。IEEE, 2009年, 第3613–3619页。[12] J. Yu 和 S. M. LaValle, 《图上最优多机器人路径规划的结构与棘手性》, 载于第二十七届*AAAI*人工智能会议, 2013年。[13] J. Yu 和 S. M. LaValle, 《图上的最优多机器人路径规划: 完备算法与有效启发式方法》, *IEEE*机器人学汇刊, 第32卷, 第5期, 第1163–1177页, 2016年。[14] P. Surynek, 《多机器人路径规划的一种优化变体是棘手的》, 载于*AAAI*人工智能会议论文集, 第24卷, 第1期, 2010年。[15] E. Erdem、D. G. Kisa、U. Oztok 和 P. Sch¨uller, 《面向多智能体寻路问题的通用形式化框架》, 载于第二十七届*AAAI*人工智能会议, 2013年。[16] G. Sharon、R. Stern、A. Felner 和 N. R. Sturtevant, 《用于最优多智能体路径查找的基于冲突的搜索》, 人工智能, 第219卷, 第40–66页, 2015年。[17] G. Sharon、R. Stern、M. Goldenberg 和 A. Felner, 《用于最优多智能体路径查找的递增代价树搜索》, 人工智能, 第195卷, 第470–495页, 2013年。

[18] M. Barer、G. Sharon、R. Stern 和 A. Felner，《多智能体路径查找问题的基于冲突的搜索算法的次优变体》，载于第七届组合搜索年度研讨会，2014年。[19] R. J. Luna 和 K. E. Bekris，《推送与交换：具有完备性保证的快速协作式寻路》，载于第二十二届国际人工智能联合会议，2011年。[20] T. Guo 和 J. Yu，《多项式时间内网格上的次于1.5倍时间最优多机器人路径规划》，载于机器人学：科学与系统，2022年。[21] T. Guo、S. W. Feng 和 J. Yu，《面向大规模无人机协调应用的三维空间多项式时间近时间最优多机器人路径规划》，载于*2022*年*IEEE/RSJ*智能机器人与系统国际会议（*IROS*），2022年。[22] S. D. Han、E. J. Rodriguez 和 J. Yu，《Sear：一种具有期望常数因子最优性保证的多项式时间多机器人路径规划算法》，载于*2018*年*IEEE/RSJ*智能机器人与系统国际会议（*IROS*）。IEEE，2018年，第1–9页。[23] J. Yu，《高维网格上的常数因子时间最优多机器人路径规划》，*2018*年机器人学：科学与系统，2018年。[24] H. Ma 和 S. Koenig，《智能体团队的最优目标分配与路径查找》，载于*AAMAS*，2016年。[25] H. Ma、T. S. Kumar 和 S. Koenig，《考虑延迟概率的多智能体路径查找》，载于*AAAI*人工智能会议论文集，第31卷，第1期，2017年。[26] M. Szegedy 和 J. Yu，《堆栈中存储物品的重新排列》，载于第十四届机器人算法基础国际研讨会，2020年。