arXiv:2304.04217v1 [cs.AI] 9 Apr 2023

终身多智能体路径寻找的高速公路研究

李明峰　　孙敏

摘要——**在现代履约仓库中，智能体会穿行于地图，完成不断动态到来的无穷任务，这一问题被表述为终身多智能体路径寻找（终身 MAPF）问题。解决这一具有挑战性问题的目标，是在有限运行时间内为每个智能体寻找路径，同时最大限度地提高吞吐量。然而，随着地图尺寸或智能体密度增大，现有方法的运行时间会呈指数级增长，并会出现死锁和重新规划路径等不良现象。为应对终身 MAPF 中的这些挑战，我们探索了主要用于一次性多智能体路径寻找（即预先一次性寻找路径）的高速通道思想，通过鼓励智能体朝同一方向移动来降低问题的复杂度。我们采用两种方法将高速通道思想融入终身 MAPF 框架，并讨论如何利用其特性减少现有方法中的死锁和重新规划路径问题。实验结果表明，在高速通道设置下，地图尺寸增大时，运行时间显著缩短，吞吐量的衰减也逐渐变得微不足道。此外，随着智能体密度增加，利用高速通道可以显著减少死锁和重新规划路径现象。**

索引词— **多个移动机器人或智能体的路径规划、多机器人系统、运动与路径规划**

一、引言

多智能体路径寻找（MAPF）是为智能体团队规划协作路径，同时避免碰撞的问题。多智能体路径寻找（MAPF）已广泛应用于视频游戏 [1], 交通管理 [2], 和配送策略 [3] 等领域。近年来已经提出了多种多智能体路径寻找（MAPF）求解器，例如 CA* [4], CBS [5],ECBS [6],，它们以一次性的方式考虑路径规划问题，假设每个智能体只有一对起点和目标位置。通常来说，如果不进行合理的修改，这些方法并不适用于在线场景中的应用。

考虑现实世界中的应用，尤其是履约仓库中的机器人 [7], ，其并非只有一对起点和目标位置，而是会动态分配新的目标。这种场景被称为终身 MAPF。尽管当前的终身 MAPF 解决方案 [8]–[11] 能够处理一系列目标，但它们的计算效率较低或吞吐量较低（即，每超碰在线完成的任务数 per
时间步）。因此，滚动时域冲突消解（RHCR） [12] 将窗口化 MAPF [4]的理念融入终身 MAPF 框架，将终身多智能体路径寻找问题拆分为一系列一次性多智能体路径寻找问题，从而在不牺牲吞吐量的前提下提升计算效率。然而，该方法仍会遇到死锁和重新规划路径 [4], [12]等不良现象。

作者来自台湾新竹市 300 的国立清华大学（电子邮件：justin.mingfeng.li@gmail.com；sunmin@ee.nthu.edu.tw）。

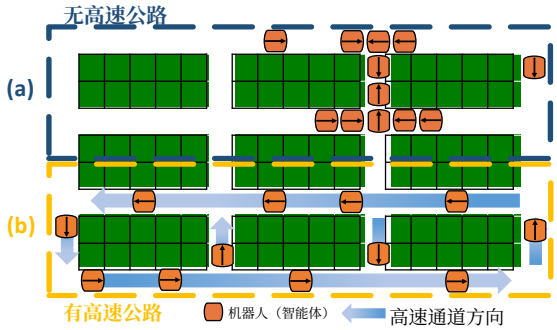


图 1. 仓库地图中的两个场景。子图 (a) 展示了没有高速公路的地图。子图 (b) 展示了设有高速公路的地图，用于解决子图 (a) 中智能体拥挤导致的死锁和重新规划路径问题。高速通道方向以蓝色箭头表示。

为了解决终身 MAPF 中的此类挑战，我们研究了一次性多智能体路径寻找 [13]–[15]中常用的高速公路 方法。首先，我们通过强制智能体沿特定方向移动来利用有向地图 [13],[14] 。这一简单而有效的策略降低了问题的复杂度，从而提高了效率。此外，我们不再强制规定移动方向，而是在规划过程中利用[15] 计算启发式值，对逆着高速通道方向的移动进行惩罚。简而言之，这两种策略都鼓励智能体朝同一方向移动，从而有效避免迎面冲突。随后，我们进一步讨论将高速公路 融入终身 MAPF 框架后的特性；如图1所示，这可以有效减少死锁和重新规划路径等不良现象。

通过多项实验，我们展示了在终身 MAPF 框架中使用高速公路 的优势，以及运行时间与吞吐量之间的权衡。实验结果表明，在障碍物占比超过 50% 的类仓库地图中，运行速度可提升数十倍，同时吞吐量衰减不到 10%。此外，当智能体密度较高时，使用高速公路后，吞吐量甚至会有所提升，因为高速公路显著减轻了死锁和重新规划路径带来的严重影响。

II. 背景与相关工作

A. 一次性多智能体路径寻找

MAPF 是一个NP难问题 [16], [17]，通常以一次性和离线的方式求解，分别指每个智能体预先已知恰好一个目标，以及一次性找到所有智能体的路径。一次性多智能体路径寻找求解器的解通常通过 代价总和 （每个智能体到达的时间之和）或最大完成时间（所有智能体到达其目标所需的时间跨度）进行评估。

存在各种一次性多智能体路径寻找方法 [18],，包括基于编译的求解器 [19], [20], 、基于规则的求解器 [21],、[22], 基于A* 的求解器 [23], [24],以及基于优先级的规划 [25], 等。尤其是，基于搜索的求解器 [6], [26]及其变体十分常见。大多数基于搜索的求解器都是两层 MAPF求解器。低层求解器为每个智能体规划路径，高层求解器处理这些路径的冲突，以保证不会发生碰撞。A* [27] 是基于搜索的求解器中低层求解器所使用的典型算法。至于高层求解器，基于冲突的搜索（CBS） [5] 是一种流行的MAPF求解器，具有完备性和最优性，而其变体 [6], [28] 则牺牲最优性以换取更快的计算速度。此外，基于优先级的搜索（PBS） [29], 通过优先级排序解决冲突，展现出卓越的计算效率，但并不完备，也不是最优的。

B. 终身 MAPF

与一次性多智能体路径寻找不同，终身 MAPF 解决的是智能体可能被分配多个任务的问题，这种情况总是伴随着在线场景。在线 MAPF 中，任务会在执行过程中动态生成 [3]。因此，解决方案无法预先规划，并且应通过吞吐量 （每个时间步完成的任务数）进行评估，因为执行时间可能无限延续，分配给智能体的任务也可能无穷无尽。例如，在仓库中，任务会随时不断到来，智能体完成先前任务后会被分配新任务。因此，路径求解器需要实时持续地考虑解决方案，并应将问题作为终身 MAPF 来解决。此外，我们不仅应考虑吞吐量 ，还应考虑运行时间 （规划所需的计算时间），以确保路径规划能及时完成，从而有效利用智能体。

C. 终身解决方案

首先，如果所有任务都已提前获知，则可以将终身多智能体路径寻找问题作为一次性多智能体路径寻找问题来解决 [10]。否则，第二类方法 [9], [11]会在每个时间步为所有智能体重新规划路径，以动态处理新任务。尽管如此，这两类方法分别需要耗费大量时间离线解决整个问题，或在每个时间步持续重新规划。因此，可以采用第三类方法 [8] ，仅为获得新目标的智能体重新规划新路径；但如果所有智能体之间缺乏协同规划，其吞吐量仍然较低。

为了在最大化吞吐量的同时保持运行时间效率，RHCR [12] 将窗口化 MAPF [4]融入了终身MAPF。其理念是规划通往目标的完整路径，并仅解决前 w 个时间步内的冲突，而 w 代表时间范围。除了时间范围 w 之外，RHCR 还有另一个用户指定的参数h ，用于指定重规划周期。每经过 h 个时间步作为一个回合，系统会为已到达目标的智能体分配新的目标，并规划后续 w 个时间步内的无冲突路径，以供未来使用。智能体正在遵循上一回合的规划结果。经过 h 时间步后，智能体遵循这些新的无冲突路径，并且应规划下一回合的路径。然而，由于窗口化MAPF的不完备性以及无法保证最优性，窗口化MAPF 求解器会遇到死锁（即，智能体在当前位置空闲，相互等待对方先通过），这依赖于一个潜在函数来评估智能体的进展，并且 w 将增加以解决死锁 [12]。此外，由于窗口化 MAPF [4] 仅解决前几个时间步中的冲突，并不断重新规划路径，因此会导致智能体修改其路径方向，或重新访问它们之前访问过的位置，这种情况在 RHCR [12] 中也会发生。

D. 高速公路

在一次性多智能体路径寻找中，高速公路 是地图上的一种附加设置，用于缩短运行时间。它为移动方向制定全局规则，鼓励或强制 MAPF 求解器沿一致的方向搜索路径。禁止智能体逆着高速通道方向移动，是降低计算复杂度并避免碰撞的一种简单策略。例如，一种一次性解决方案 [14] 使用有向地图来降低多智能体路径寻找的复杂度，从而提升计算效率。此外，另一种一次性解决方案 [15] 在计算启发式值时，增加逆着高速通道方向的路径代价，从而隐式地鼓励求解器探索沿高速通道方向的路径。然而，这些方法是为一次性多智能体路径寻找提出的，而在终身 MAPF 中，死锁和重新规划路径等不良现象并不会出现。尽管 RHCR [12]（一种终身解决方案）在其实验部分使用了有向地图来降低规划复杂度，但仍未进一步讨论和实验研究高速公路 如何影响终身场景下的规划结果。

III. 问题定义

多智能体路径规划（多智能体路径寻找）问题被表述为地图 $G = (L, E)$ 的有向图以及一组智能体 $A = \{a_1, ..., a_k\}$。该图表示位置L 以及连接这些位置对的边 E 。对于每个智能体 a_i，它从其起始位置 l_s^i 出发，并以其自身的目标位置 l_g^i为目标。假设时间被离散为时间步， a_i 应在每个时间步决定通过相连的边 $e \in E$ 移动 到相邻位置 $l_i \in L$ ，或在其当前位置等待 ，且移动将在该时间步内完成。

然而，当两个智能体同时到达同一位置或经过同一条边时，可能会发生碰撞，这称为冲突 [30]。因此，多智能体路径寻找的目标是为所有智能体找到无碰撞路径 $P = \{p_1, ..., p_k\}$ ，使它们共同到达各自的目标，其中 $p_i = [l_s^i, ..., l_g^i]$ 是智能体 a_i的路径，由一系列相邻位置组成。

我们假设该场景是终身在线多智能体路径寻找。智能体从各自的位置出发，事先不知道自己的任务（即移动到目标位置），完成任务后会被分配一项新任务（即。

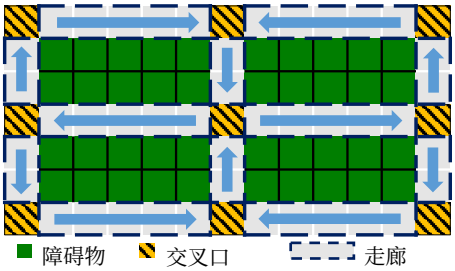


图 2. 一个示例地图，其中箭头表示每条走廊中高速公路的方向。连接走廊的交叉口不能属于任何走廊。

到达其目标位置）。MAPF 求解器的目标是持续为每个智能体规划路径，并最大化吞吐量 （每个时间步完成的任务数量）。

IV. 使用高速通道的终身 MAPF

A. 高速公路定义

在带有高速通道的一次性多智能体路径寻找中 [15], 高速公路 被表示为给定地图 $G = (L, E)$ 的图的子图 $G_H = (L_H, E_H)$，其中 L_H 包含高速公路中涉及的位置，E_H 表示沿高速公路方向的边。

在这项工作中，我们聚焦于类仓库地图 [30],其中高速公路 设置可以通过指定每个走廊的移动方向来定义。位置序列$K = \{l_0, ..., l_n\} \subseteq L$ 被称为长度为 n 的走廊，当且仅当 l_i根据 E 与恰好两个位置 l_{i-1} 和 l_{i+1} 相连，且对于 $i = 1 \rightarrow i - 1$ （即每个位置只有一个入口和一个出口）。 l_0 和 l_n 可以是同一个位置，此时称为环形走廊。对于每个走廊，存在一个方向 $d \in D$，它是一个布尔值，表示走廊的方向，如图 2 所示。此外，任何作为交叉点的位置，如果连接两个或更多走廊，则不属于任何走廊。最后，给定一组方向 $D = \{d_0, ..., d_m\}$ （m是走廊的数量），该高速公路 可以表示为一个子图 $G_H = (L_H, E_H) \subseteq G$，以及与高速公路方向相反的边 $E_{H'} = \{(l_b, l_a) | (l_a, l_b) \in E_H\}$。

B. 终身 MAPF 框架

我们采用 RHCR [12]作为终身MAPF框架。给定用户指定的时间范围 w 和重规划周期h ($h \leq w$)，将在每 h 时间步为智能体规划前 w 时间步内的无冲突路径，并将新目标分配给已到达其目标的智能体。对于高层求解器，我们使用 PBS [1]，其在 RHCR 中展现出突出的计算效率。

在我们的方法中，我们使用 A* 作为低层求解器，并采用真实的最短距离启发式。最短路径启发式值忽略动态约束（即移动的智能体），可预先通过最短路径算法计算得出。这样，节点间距离的精确值便能引导低层求解器比曼哈顿距离启发式值遍历更少的位置即可到达目标。此外，在给定高速公路设置的情况下，计算出的启发式值会促使低层求解器找到沿高速公路方向行进的路径，从而减少冲突；由于去除了一些逆着高速通道方向移动的选项，低层规划的复杂度也有所降低。

C. 严格限制高速公路和软限制高速通道

有两种方法可以为低层求解器设置高速公路。第一种方法是仅允许跨越边沿一个方向移动 [14]，我们在本文中称之为 严格限制高速公路。第二种方法是仅增加启发式值，但不增加沿高速公路方向的启发式值 [15]，这会鼓励低层求解器沿着高速公路的边搜索路径，我们在此称之为 软限制高速通道。

*1)*严格限制高速公路：使用严格限制高速公路时，$G = (L, E)$ 将被替换为子图 $G_S = (L, E_S)$，其中 $E_S = (E - E_{H'}) \subseteq E$。因此，在严格限制设置下，相邻位置的连通性受到严格限制，这意味着此处无法进行逆着高速公路方向的移动，如图3 所示。严格规则迫使智能体遵循高速公路的方向，但限制了使用空闲位置的灵活性，因为低层求解器可能不再探索避开高速公路方向的相邻位置。为了在RHCR中利用具有严格限制的高速公路，应预先计算并存储基于忽略移动智能体的全对最短路径的启发式值，以供使用。使用 G_S 计算的启发式值定义为：

$$H_S(l_s, l_g) = \min_p |p| - 1 \tag{1}$$

其中，$p = [l_s, ..., l_g]$ 是从起始位置l_s 到目标位置 l_g 的一条可行路径，$|p|$ 是该路径的长度。因此，$H_S(l_s, l_g)$ 表示从起点 l_s 移动到目标位置 l_g的最短距离。随后，低层求解器遵循精确高速公路启发式的指引，使智能体沿着高速公路的方向移动。

2）软限制高速公路：在低层规划时，将限制间接加入启发式中。用户指定的参数 c 大于一。在计算启发式值时，沿高速公路方向穿过一条边的代价通常为一，而逆着高速公路方向移动时，代价会增加到 c，其定义如下：

$$H(l_s, l_g, c) = \min_p \sum_{(l_i, l_{i+1}) \in p} \begin{cases} c, & \text{if } (l_i, l_{i+1}) \in E_{H'} \\ 1, & \text{otherwise,} \end{cases} \tag{2}$$

其中，$p = [l_s, ..., l_g]$ 是从起始位置 l_s到目标位置 l_g的一条可行路径。当 c 接近1时，启发式值接近于没有任何限制时的最短路径。相反，如果 c 不断增大至无穷大，启发式值则类似于在高速公路设置下计算出的最短路径。高速公路启发式值鼓励智能体朝同一方向移动，以避免碰撞。例如，图 3 展示了一个带有 $c = 2$的示例，沿着与高速通道方向相反的路径移动时，其启发式值更高（即 8、6、4、2）。然而，由于缺乏严格限制，仍然会发生逆高速通道方向移动的情况，这意味着存在

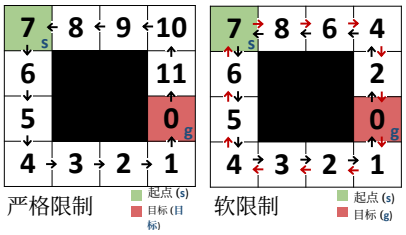


图 3. 使用 c = 2对比 严格限制 （左）和软限制 （右）的连通性。高速公路 的方向为逆时针方向，黑色箭头表示智能体在严格限制下可以采取的移动方向。软限制 允许双向移动（黑色和红色箭头），但严格限制 不允许。数字表示从当前位置到目标的启发式值，在严格限制下，这些值沿高速通道方向严格递减。在软限制下，当最短路径包含逆高速通道方向的移动时，代价会增加至 c （即 8、6、4、2）。

应解决的冲突。要在 RHCR 中使用软限制高速通道，应事先根据选定的 c 计算启发式值，然后低层求解器将遵循启发式值来寻找路径。

D. 高速公路行为

在软限制高速通道下，低层求解器使用高速公路启发式进行规划。由于逆着高速公路方向的路径具有更高的启发式值，沿高速公路方向的相邻位置具有更高的优先级，会从A* 的开放集合中被取出，从而获得更低的预期代价。因此，低层求解器找到的路径方向表现出方向一致性。智能体路径的一致性会导致更少的面对面碰撞，从而减少高层求解器为解决冲突而需要生成的节点数量。当 c 较小时，结果接近不使用高速公路时的结果。随着 c 增加，规划结果逐渐接近使用高速公路最短路径作为启发式值时的结果。然而，即使将 c 设为无穷大，当最短路径上的位置被动态障碍物阻塞时，仍会发生逆着高速公路方向的移动。这表明智能体可以灵活选择移动方向，但也可能导致需要花费时间解决的冲突。使用严格限制高速公路时，如图 4 所示，可以通过在规划期间阻塞通往相邻位置的边，保证智能体路径方向的一致性。

使用高速公路 的目的是确保智能体的路径在局部遵循共同的协作规则，从而通过解决冲突缓解拥堵，并保持相邻时间范围内规划结果的一致性。得益于路径方向的一致性，严格限制高速公路具有以下特性，可解决死锁和重新规划路径的问题。此外，在实验部分，我们将展示软限制高速通道如何根据给定的 c 利用这些特性。

E. 避免死锁的特性

死锁 是由窗口化 *MAPF* 未考虑整个时间范围 [12] 所导致的现象。当两个智能体迎面移动，且目标位于彼此相对的一侧时，就会发生死锁。由于 w较小，它们都不能由于存在对换冲突，智能体无法直接通行；也无法沿反方向通行，因为高层求解器更倾向于让智能体留在原位置等待，以获得更低的总代价，而不是选择更长的路径。例如，如果时间范围为 $w = 2$、重规划周期为 $h = 2$，并使用 CBS 作为高层求解器，那么窗口化 MAPF 求解器会返回图5-(a)中的路径，让智能体在原位置停留 w 个时间步，然后再开始移动到目标位置。这种情况是由窗口化 MAPF 求解器有限的协同规划能力造成的：它只解决前 w 个时间步内的冲突，而忽略之后的冲突。因此，由于在原位置等待时总代价更低，低层求解器无法考虑更远的位置。要让一个智能体沿上侧移动，另一个沿下侧移动，w 至少应设为 3，使求解器考虑更远处的路径，而不是像图5-(b)那样，通过在整个时间范围内等待来避开冲突的取巧解。更糟的是，如果走廊更长或目标位置更远，就需要检查更大的 w 值才能找到解，这会增加运行时间。

一般而言，当 MAPF 求解器无法为智能体找到可行的移动路径，且这些路径的总代价 低于等待路径的代价时，就会发生死锁。也就是说，MAPF 求解器无法为智能体找到下一 w 时间步内的新位置，使其到目标的距离总和更小。给定 $A_d \subseteq A$ 表示可能导致死锁的智能体子集。智能体 $a_i \in A_d$ 请求$p_i = [l_c^i, ..., l_n^i]$ ($|p_i| - 1 = w$, 其中 w 为时间范围），即一条从当前位置 l_i^c到下一 w 时间步内下一位置 l_n^i 的无碰撞可行路径。利用严格限制高速公路中表示到目标位置距离的最短路径启发式值，可能发生死锁的情形可简化为：

$$\sum_{a_i \in A_d} H_S(l_c^i, l_g^i) \leq \sum_{a_i \in A_d} \min_{p_i} H_S(l_n^i, l_g^i) \tag{3}$$

其中，l_g^i 是 a_i的目标位置。

在严格限制高速公路中，如果智能体 a_i 从任意位置 l_a^i 移动到其相邻位置 l_b^i，并且 $(l_a^i, l_b^i) \in E_H$始终成立，那么 $H_S(l_b^i, l_g^i) = H_S(l_a^i, l_g^i) - 1$ 在 $l_a^i \neq l_g^i$时成立，其中l_g^i 是其目标位置。因此，式（3）可以改写为：

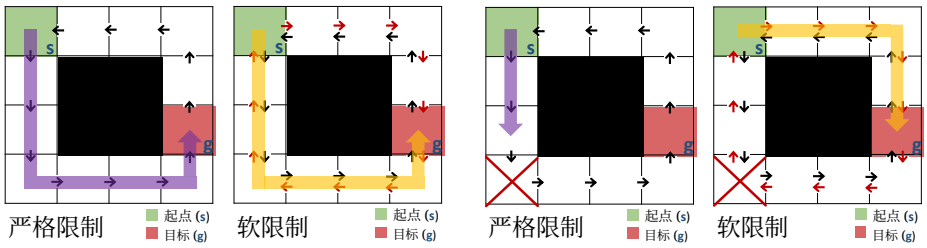
$$\sum_{a_i \in A_d} H_S(l_c^i, l_g^i) \leq \sum_{a_i \in A_d} \min_{p_i} H_S(l_c^i, l_g^i) - |\bar{p}_i| + 1 \tag{4}$$

其中，$\bar{p}_i$ 是 p_i中不重复位置的集合，$|\bar{p}_i|$ 反映了 p_i中不重复位置的数量。

当 $\exists a_i \in A_d, |\bar{p}_i| > 1$时，式（4）不成立。也就是说，在严格限制高速公路中，如果所有智能体都沿着边$\in E_H$移动，那么除非每个智能体都没有可移动到的相邻位置，否则不会发生死锁。

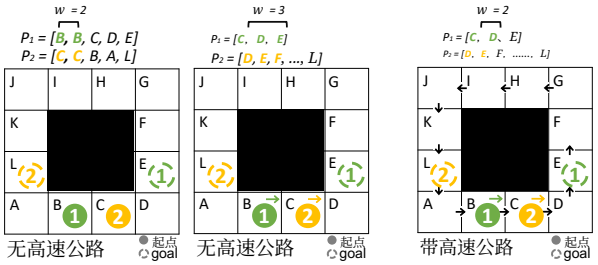
F. 避免重新规划路径的性质

重新规划路径是由于窗口化 *MAPF* [4], 缺乏长期考量而产生的现象，这会导致智能体改变路径方向，或重新经过之前访问过的位置。窗口化的想法



(a) 没有动态障碍物　　(b) 存在动态障碍物

图 4. 严格限制和软限制高速公路下的不同表现。箭头表示接下来n 个时间步内的路径。前方道路畅通时，严格限制和软限制高速公路下的智能体都会沿高速公路方向行进，如图 (a) 所示。当出现动态障碍物时，严格限制高速公路下的智能体仍严格保持高速公路方向，而软限制高速公路下的智能体则选择重新规划路径，如图 (b) 所示。



(a) 双向，w = 2和 3　　(b) 双向，w = 2

图5. 死锁死锁 情形。 P_1， P_2 表示由 MAPF 求解器找到的 a_1， a_2 路径，w 为时间范围。在不使用高速公路的面板 (a) 中，应将 w 设为 3，以避免死锁。在面板 (b) 中，使用高速公路 时，无论 w为何，都不会发生死锁 。

MAPF 在 RHCR 中的用法是按时间将终身多智能体路径寻找问题拆分为一系列子问题，这些子问题称为回合。在每个回合中，低层求解器为每个智能体规划完整路径，然后高层求解器在有限的时间范围内解决路径之间的冲突 w。规划完成后，每个智能体沿路径移动前 h 个时间步（小于或等于 w）。然而，尽管规划的是通往目标的完整路径，每个智能体实际只沿着前 h 个时间步的部分路径移动。因此，智能体可能停在启发式值高于原位置的位置；也就是说，等待比移动更明智。此外，在大多数基于搜索的 MAPF 求解器中，高层规划会采用特殊机制来确定优先级排序，以决定发生冲突时哪个智能体先行动（例如，CBS [5] 会将约束添加到某个智能体上，形成一个*CT*节点；PBS [29] 则维护优先级排序对）。然而，后续高层规划中并不保证智能体优先级排序的一致性，而优先级排序的变化可能导致规划结果出现明显差异。这些情况可能会要求智能体重新规划路径，甚至向后移动，从而增加智能体到达目标所需的距离。

我们正式定义，当智能体 a_i 被 MAPF 求解器分配的路径 $p_i = [l_c^i, ..., l_n^i]$ 使其 a_i 远离其目标 l_g^i时，该智能体正在重新规划路径 ，这可以通过最短路径启发式表示:

$$H_S(l_c^i, l_g^i) < H_S(l_n^i, l_g^i) \tag{5}$$

其中 l_c^i 是 a_i 的当前位置，而 l_n^i 表示 a_i 经过 h 个时间步后将到达的位置（h 是小于或等于时间范围w的重规划周期）。在 严格限制高速公路中，给定智能体 a_i 及其目标位置 l_g^i，$\forall (l_a, l_b) \in E_H$， $H_S(l_a, l_g^i) = H_S(l_b, l_g^i) + 1$ 如果 $l_a \neq l_g^i$。因此，当 a_i 沿着边 $\in E_H$移动时，公式 (5) 不成立，因为 $H_S(l_c^i, l_g^i) \geq H_S(l_n^i, l_g^i)$ 。总之，使用严格限制高速公路时，这些性质保证了当智能体沿着属于高速公路的边移动时，不会出现死锁 和重新规划路径 。

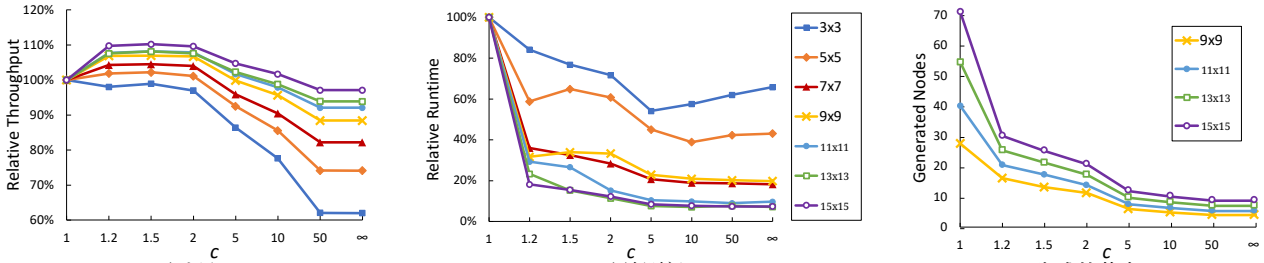
V. 实验

我们的实验重点比较将高速公路纳入前和纳入后 终身 MAPF 的表现，并分析其优势与不足。首先，我们评估了采用高速公路后吞吐量和运行时间的变化。其次，我们分析了在地图尺寸或智能体密度增大时，使用高速公路的优势。

A. 环境

我们使用 $w = 5$、 $h = 5$ 实现了 RHCR，采用 PBS 作为终身 MAPF 求解器，并在 Python 中使用标准的位置-时间 A* 作为低层求解器；所有实验均在配备 16 GB 内存的 Intel Core i9-9980XE 上运行。在每轮实验开始前，智能体的起始位置和目标位置会从地图上不同的空位置中随机选择。当智能体到达其目标位置时，会为其分配一个不属于任何其他智能体目标位置的位置。此外，我们为每项实验随机初始化 100 个回合，并在每个回合中模拟 100 次规划迭代。每次迭代的时间限制为 60 秒，如果任意一次迭代超时，该回合将被标记为失败用例 。所有显示的指标均为平均值，且不包括失败用例。

通常，吞吐量和运行时间是终身 MAPF 的两项主要指标。为了评估高速公路能否有效应对现有的死锁和重新规划路径现象，我们在实验中记录了移动时间步、空闲时间步和重新规划率。重新规划率是一次迭代中重新规划路径的智能体所占的百分比。此外，移动时间步表示智能体执行任务期间移动的时间步数，而空闲时间步则表示智能体停留在同一位置的时间步数



(a) 吞吐量 (b) 运行时间 (c) 生成的节点

图 6. 不同地图尺寸下吞吐量、运行时间和生成节点数量（即 PBS 在找到解之前生成的高层节点数量）的相对比率。比较采用不同 c 的软高速公路 所得结果与不采用高速公路（即采用 $c = 1$）所得结果。随着c 增加，逆着高速公路方向移动的代价也随之增加，这会逐渐促使智能体沿高速公路方向移动，并导致吞吐量降低、运行时间缩短，因为规划时生成的节点更少。此外，随着地图尺寸增大，对吞吐量的影响会减弱，这一点可从面板 (a) 中由下至上的曲线看出。

在任务期间未移动。为了观察增大c 对智能体的影响，高速公路规避率表示智能体逆着高速通道方向移动的概率。与 [5] 和 [15], 类似，生成的节点用于解释我们的方法为何运行时间更短；它指的是 MAPF 求解器在一次规划过程中找到解之前生成的高层节点数。

鉴于启发式值是根据最短路径计算的（例如，采用严格限制 高速公路和软限制 高速公路，并结合 $c = 1$ 或 ∞），可以使用先前工作 [4] 中提到的一种技巧来简化规划。具体而言，在窗口化 MAPF 中，我们无需规划通往目标的完整路径，而是可以直接忽略 w 时间步之后的规划，这样可得到相同的结果，同时省去规划完整路径所需的时间。我们在表 II 中列出此技巧以供比较。

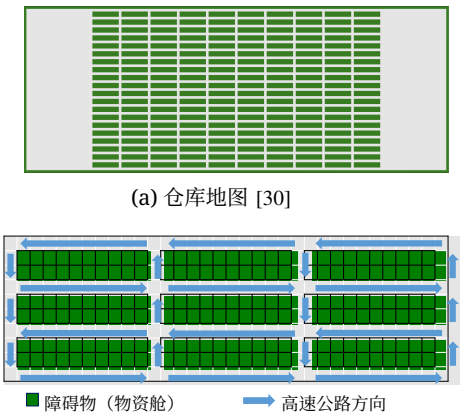
B. 履约仓库

在自主仓库中，一组库存货架被紧密地摆放成一个矩形街区，街区之间留有走廊，并以网格形式排列，如图7所示。在我们的实验中，我们遵循图7-(a)中的仓库地图 设置 [30]: (1) 每个街区包含 10×2 个货架；(2) 走廊为单行。为简单起见，我们测试包含 N×N 个街区的地图，其中 N 为奇数（即 3、5、7 等），并将库存货架视为智能体的障碍物。例如，图7-(b)展示了地图的最小版本 3×3 。此外，整个地图中障碍物所占的百分比表示地图的拥挤程度。我们的最小地图（图7-(b)）中障碍物占比超过 50%，并且随着地图尺寸随 N 增大而扩大，障碍物占比也逐渐上升。高障碍物密度和相邻走廊适合用于测试

高速公路.

C. 从无高速公路到高速公路

<|...|>



(a) 仓库地图 [30]

(b) 带有高速公路方向的仓库地图

图 7。（a）显示了 [30] 中包含 10×20 个库存货架街区的仓库地图。每个矩形表示一个街区。（b）显示了包含 3×3 个库存货架街区的仓库地图，以及高速公路的方向（箭头）。两幅图中的每个街区（绿色矩形）均由 10×2 个库存货架（绿色方块）组成。

当地图较小时，图 6-(a) 中吞吐量会随着 c 的增加而下降，不过图 6-(b) 中运行时间略有减少。然而，在地图较大的情况下，随着 c 增加，吞吐量下降得慢得多，而运行时间则显著缩短。令人惊讶的是，当 $c = 1.5$ 时，相对吞吐量甚至超过了不使用 高速公路 时的吞吐量。此外，随着地图尺寸增大，无高速公路 （$c = 1$）与高速公路 （$c = \infty$）之间的吞吐量差距呈缩小趋势。

运行时间的加速主要得益于在高速公路下更高效的规划，因为必须遵循方向的全局规则，因此规划结果更加一致，从而在找到解决方案之前产生的冲突更少。参见图 6，如果使用更高的 c ，则需要生成更少的节点来找到解决方案，而且在更大的地图中，这一优势更加明显。

表 I 中的结果验证了使用高速公路可减少平均空闲时间步（即智能体在任务期间停留在同一位置的时间步），这反映了路径方向更加一致、死锁更少；而且地图越大，平均空闲时间步的减少幅度也越大。此外，随着c 增大，智能体逆高速通道方向移动的概率以及重新规划路径问题都会减少，

表 I 不同地图尺寸、c以及相同智能体密度（5%）的软限制高速通道 设置下的平均空闲时间步、移动时间步、高速公路规避率和重新规划率。

	c	地图尺寸 3×3	5×5	7×7
Idle 时间步	1	2.49	2.66	2.83
	2	2.42（-2.5%）	2.55（-4.1%）	2.61（-7.9%）
	5	2.32 (-6.8%)	2.38 (-10.5%)	2.42（-14.7%）
	50	2.22 (-10.8%)	2.29 (-13.8%)	2.33 (-17.9%)
移动 时间步	1	17.65	27.76	37.69
	2	18.27 (3.5%)	27.49 (-0.9%)	36.42 (-3.4%)
	5	20.90 (18.4%)	30.61（10.3%）	39.96 (6.0%)
	50	29.84（69.1%）	38.45 (38.5%)	46.72 (23.9%)
高速公路 规避 / 重新规划路径 Rate	1	43.2% / 3.8%	44.3% / 4.5%	44.9% / 4.7%
	2	32.3% / 2.5%	28.3% / 1.9%	25.5% / 1.5%
	5	18.7% / 0.7%	14.9% / 0.4%	12.5% / 0.3%
	50	2.2% / 0.0%	2.0% / 0.0%	1.8% / 0.0%

为比较使用高速公路 (c $\neq$ 1 $\neq$) 后空闲时间步减少和移动时间步增加的幅度，括号中表示 c 与无高速公路 ( ) 相比的差异百分比。

这一点体现在表 I 中的高速公路规避率（智能体逆高速通道方向移动的概率）和重新规划率（一次规划迭代中需要重新规划路径的智能体所占比例）上。此外，沿高速公路行进会使智能体有更多移动时间步（即智能体到达目标所需的时间步）。不过，在较大的地图中使用高速公路会导致更少的额外移动时间步，这也解释了为什么使用高速公路时，较大的地图更有优势。例如，表 I 显示，在 3x3 地图中使用高速公路($c = 50$)会导致移动时间步额外增加 69.1%，而在 7x7 地图中使用高速公路 ($c = 50$)仅会导致移动时间步额外增加 23.9%。

D. 扩展规模

为确保运行时间足够高效，能够及时分配路径而不让智能体闲置，问题在于，仓库地图扩展后，高速公路的优势是否仍然存在。根据前文提到的结果，答案显然是肯定的。随后，我们将讨论并量化随着地图尺寸扩大和智能体密度增加，高速公路所带来的益处。

1) 地图尺寸：为了扩大地图规模，街区保持与图 7-(b)中相同，每个街区包含 10×2 个货架，走廊仍为单行。随后，街区数量从 3×3、5×5、7×7 个.....增加到 15×15 个，智能体密度固定为5%。使用严格限制高速公路、软限制高速通道（$c = \infty$）以及不使用高速公路 的基线所得结果如图所示。

表 II。当地图变得更大时，高速公路 与基线之间的吞吐量差距变小，而使用高速公路 所获得的加速比却不断提高。吞吐量差距减小的主要原因是，随着地图变大，为高速公路额外执行的移动次数相对于移动总次数而言变得微不足道。此外，与使用高速公路的设置相比，在没有高速公路的情况下，较大的地图上会出现失败用例。

此外，严格高速公路的吞吐量差距

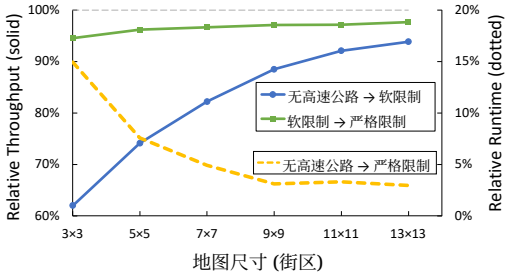
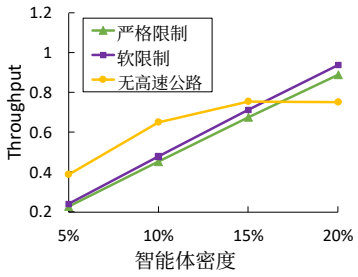
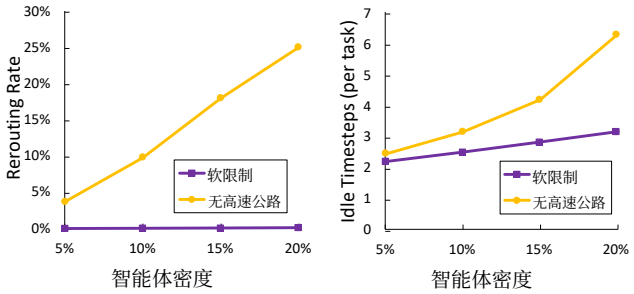


图 8. 将“无高速公路转换为 软限制高速通道”以及将“软限制高速通道 转换为严格限制高速公路”的相对吞吐量。点线表示将“无高速公路 替换为严格限制高速公路”的相对运行时间，显示了运行时间的减少。随着地图扩大，这两种设置的相对吞吐量趋近于 100%（实线），运行时间持续减少（黄色点线）。



(a) 吞吐量



(b) 重新规划率和空闲步数

图 9。比较不同智能体密度下高速公路 与无高速公路 的吞吐量、重新规划率和平均空闲时间步数。当智能体密度较高时，使用高速公路 甚至比无高速公路获得更高的吞吐量，如面板 (a) 所示。如面板 (b) 所示，这是因为重新规划现象（实线）和空闲智能体（虚线）十分严重。

与软高速公路 之间的差距在图 8 中也在缩小。因此，将软高速公路 转换为严格高速公路以获得更快的运行时间，成本效益也越来越高（见表 II）。

2）智能体数量：为了测试不同智能体密度对地图的影响，在图7-(b)所示的 3×3 地图上，将智能体密度从5%逐步提高到20%。根据图9中的结果，使用高速公路时，吞吐量持续上升。然而，不使用高速公路时，吞吐量在密度增加后开始下降。因此，我们评估了重新规划率和平均空闲时间步。在高速公路设置下，重新规划路径很少发生，且较少的平均空闲时间步可能表明，由于智能体路径方向一致，死锁和相互等待的现象显著减少。

表 II 不同地图尺寸和相同智能体密度（5%）下的吞吐量和运行时间。

	类型 \ 地图尺寸	3×3	5×5	7×7	9×9	11×11	13×13	15×15
吞吐量 (tasks/step)	严格限制	0.23（-41.4%）	0.45（-28.7%）	0.69（-20.5%）	0.93（-14.1%）	**1.17 (-10.6%)**	**1.43 (-8.3%)**	1.68（-3.8%）
	软限制	**0.24 (-38.0%)**	0.47（-25.9%）	0.71（-17.8%）	**0.96 (-11.5%)**	1.21（-7.9%）	1.46（-6.2%）	1.71（-2.2%）
	无高速公路	0.39	0.63	0.87	1.09 [20]	1.31 [47]	1.56 [77]	1.75 [95]
运行时间 (秒)	严格限制 (P)	0.002 (6.7)	0.010 (13.2)	0.032 (20.4)	0.069 (32.1)	0.130 (30.4)	0.284 (34.0)	0.553 (39.8)
	严格限制	0.007 (1.8)	0.034 (3.9)	0.097 (6.7)	0.241 (9.1)	0.339 (11.7)	0.681 (14.2)	1.323 (16.6)
	软限制	0.008 (1.5)	0.041 (3.2)	0.111 (5.8)	0.321 (6.9)	0.439 (9.0)	0.870（11.1）	2.016 (10.9)
	无高速公路	0.012	0.133	0.646	2.208	3.955	9.638	21.983

每个数值都是从排除失败用例（即超时）的测试用例中计算得出的平均值。使用严格限制高速公路、软限制高速通道 ($c = \infty$)以及不使用高速公路时的基准结果分别显示，使用部分规划技巧（第五节A 部分中提及）来加速的结果标记为"(P)"。使用高速公路时相较于无高速公路的吞吐量衰减百分比显示在括号中，如果存在失败用例，则其数量会标记在顶栏的方括号中（总计 100 个测试用例）。使用高速公路时相较于无高速公路 的加速比显示在底栏的括号中。

VI. 结论

在这项工作中，我们研究了高速公路这一理念（此前提出用于单次多智能体路径寻找），并将其应用于终身多智能体路径寻找场景，以权衡运行时间和吞吐量。此外，我们讨论了将高速公路与终身多智能体路径寻找框架相结合的特性，该框架能够缓解死锁和重新规划路径等现有问题。最后，我们通过一系列实验对高速公路进行了评估。根据实验结果，使用高速公路能够显著缩短运行时间；随着地图尺寸或智能体密度增大，吞吐量的衰减会逐渐减弱。

参考文献

[1] H. Ma、J. Yang、L. Cohen、T. S. Kumar 和 S. Koenig，"可行性研究：在拥挤的电子游戏环境中移动异质团队"，发表于第十三届人工智能与交互式数字娱乐会议，2017年。[2] F. Ho、A. Goncalves、A. Salta、M. Cavazza、R. Geraldes 和 H. Prendinger，"用于无人机交通管理的多智能体路径规划：机器人赛道"，2019年。[3] H. Ma、J. Li、T. Kumar 和 S. Koenig，"面向在线取送任务的终身多智能体路径寻找"，*arXiv* 预印本 *arXiv:1705.10868*，2017年。[4] D. Silver，"协作式寻路"，发表于人工智能与交互式数字娱乐 *AAAI* 会议论文集，第1卷，第1期，2005年，第117–122页。[5] G. Sharon、R. Stern、A. Felner 和 N. R. Sturtevant，"用于最优多智能体路径寻找的基于冲突的搜索"，《人工智能》，第219卷，第40–66页，2015年。[6] M. Barer、G. Sharon、R. Stern 和 A. Felner，"多智能体路径寻找问题的基于冲突的搜索算法的次优变体"，发表于第七届年度组合搜索研讨会，2014年。[7] P. R. Wurman、R. D'Andrea 和 M. Mountz，"在仓库中协调数百辆协作式自主车辆"，《AI Magazine》，第29卷，第1期，第9–9页，2008年。[8] M. ˇC´ap、J. Vokˇr´ınek 和 A. Kleiner，"在规范基础设施中进行在线多机器人轨迹规划的完备分布式方法"，发表于国际自动规划与调度会议论文集，第25卷，2015年，第324–332页。[9] Q. Wan、C. Gu、S. Sun、M. Chen、H. Huang 和 X. Jia，"动态环境中的终身多智能体路径寻找"，发表于*2018*年 *IEEE* 第*15*届国际控制、自动化、机器人与视觉会议（*ICARCV*），IEEE，2018年，第875–882页。[10] V. Nguyen、P. Obermeier、T. C. Son、T. Schaub 和 W. Yeoh，"使用答案集编程进行广义目标分配与寻路"，发表于第十二届年度组合搜索研讨会，2019年。[11] F. Grenouilleau、W.-J. van Hoeve 和 J. N. Hooker，"用于多智能体寻路的多标签 A* 算法"，发表于国际自动规划与调度会议论文集，第29卷，2019年，第181–185页。[12] J. Li、A. Tinka、S. Kiesel、J. W. Durham、T. S. Kumar 和 S. Koenig，《大型仓库中的终身多智能体路径寻找》，载于*AAA*人工智能会议论文集，2021年。[13] M. R. J. M. R. Jansen 和 N. Sturtevant，《协作式寻路的方向图》，载于*AAA*人工智能与互动数字娱乐会议论文集，第4卷第1期，2008年，第185–190页。[14] K.-H. C. Wang、A. Botea、等，《快速且节省内存的多智能体寻路》，载于*ICAPS*，2008年，第380–387页。[15] L. Cohen、T. Uras 和 S. Koenig，《可行性研究：使用高速通道进行有界次优多智能体路径寻找》，载于国际组合搜索研讨会，第6卷第1期，2015年。[16] P. Surynek，《多机器人路径规划的一种优化变体是不可处理的》，载于*AAA*人工智能会议论文集，第24卷第1期，2010年，第1261–1263页。[17] J. Yu 和 S. M. LaValle，《图上最优多机器人路径规划的结构与不可处理性》，载于第二十七届*AAAI*人工智能会议，2013年。[18] A. Felner、R. Stern、S. Shimony、E. Boyarski、M. Goldenberg、G. Sharon、N. Sturtevant、G. Wagner 和 P. Surynek，《多智能体路径寻找问题的基于搜索的最优求解器：总结与挑战》，载于国际组合搜索研讨会，第8卷第1期，2017年。[19] P. Surynek，《通过模理论可满足性统一多智能体路径寻找的基于搜索和基于编译的方法》，载于国际组合搜索研讨会，第10卷，1，2019年，第。[20] E. Lam、P. Le Bodic、D. Harabor 和 P. J. Stuckey，《多智能体路径寻找的分支切割定价法》，《计算机与运筹学》，第144卷，第105809页，2022年。[21] 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。[22] B. De Wilde、A. W. Ter Mors 和 C. Witteveen，《推送与旋转：协作式多智能体路径规划》，载于*2013*年国际自主智能体与多智能体系统会议论文集，2013年，第87–94页。[23] M. Goldenberg、A. Felner、R. Stern、G. Sharon、N. Sturtevant、R. C. Holte 和 J. Schaeffer，《增强型部分扩展A》，《人工智能研究杂志》，第50卷，第141–187页，2014年。[24] G. Wagner，《子维度扩展：计算上可处理的多机器人路径规划框架》，2015年。[25] K. Okumura、M. Machida、X. D´efago 和 Y. Tamura，《用于迭代式多智能体路径寻找的带回溯的优先级继承》，《人工智能》，第310卷，第103752页，2022年。[26] G. Sharon、R. Stern、M. Goldenberg 和 A. Felner，《用于最优多智能体路径寻找的递增代价树搜索》，《人工智能》，第195卷，第470–495页，2013年。[27] P. E. Hart、N. J. Nilsson 和 B. Raphael，《最小代价路径启发式判定的形式基础》，《*IEEE*系统科学与控制论汇刊》，第4卷第2期，第100–107页，1968年。[28] J. Li、W. Ruml 和 S. Koenig，《EECBS：用于多智能体路径寻找的有界次优搜索》，载于*AAA*人工智能会议论文集，第35卷第14期，2021年，第12 353–12 362页。[29] H. Ma、D. Harabor、P. J. Stuckey、J. Li 和 S. Koenig，《基于一致优先级的多智能体路径寻找搜索》，载于论文集

AAAI《人工智能》会议，第33卷，第 01 期，2019年，第 7643–7650页。[30] R. Stern、N. R.Sturtevant、A. Felner、S.Koenig、H.Ma、T. T.Walker、J. Li、D. Atzmon、L. Cohen、T.S.Kumar、等，“多智能体路径规划：定义、变体与基准”，载于 第十二届年度组合搜索研讨会，2019年。