随机旅行时间下多智能体路径寻找的在线重新规划与自适应参数更新

Atsuyoshi Kita 松下控股株式会社 日本大阪
kita.atsuyoshi@jp.panasonic.com

Nobuhiro Suenari 松下控股株式会社 日本大阪
suenari.nobuhiro@jp.panasonic.com

Masashi Okada 松下控股株式会社 日本大阪
okada.masashi001@jp.panasonic.com

Tadahiro Taniguchi 立命馆大学与松下控股株式会社 日本滋贺
taniguchi@ci.ritsumei.ac.jp

arXiv:2302.01489v1 [cs.RO] 3 Feb 2023

摘要

本研究探讨旅行时间连续且具有随机性、但其概率分布未知的多智能体路径寻找问题。我们的目标是管理一组自动机器人，为行人和各种机器人共存的建筑物提供包裹配送服务，例如在办公楼、医院和公寓楼中提供配送服务。在这类现实应用中，机器人通过走廊所需的时间通常是连续值且呈随机分布，而关于旅行时间概率分布的先验知识有限。多智能体路径寻找已得到广泛研究，并应用于机器人管理系统；然而，在此类环境中实现机器人操作自动化仍然十分困难。我们提出：1）在线重新规划，在机器人执行行动计划的同时对其进行更新；2）参数更新，在观测到延迟时，利用贝叶斯推断估计旅行时间的概率分布。我们采用贪心启发式，在有限的计算时间内求得解。通过仿真，我们从冲突概率和机器人的实际旅行时间两方面，将所提方法与现有方法的性能进行了实证比较。仿真结果表明，与现有方法相比，所提方法能够找到冲突至少减少 50%、且实际总旅行时间更短的行驶路径。由于参数更新优先针对路径规划中的重要边进行，所提方法只需少量试验便能达到相应性能，从而满足自动配送服务快速实施鲁棒规划的要求。

关键词

自主机器人管理系统；多智能体路径寻找；随机旅行时间；贝叶斯推断

1 引言

近年来，由于送货上门需求不断增长，以及配送行业劳动力短缺，使用自动机器人提供配送服务的需求日益增加。本研究聚焦于建筑物内的配送服务，例如办公楼、医院和公寓楼，如图1所示。建筑物内的通道狭窄，因此，处于自动运行模式的机器人无法彼此通过。此外，通道中还有移动障碍物，例如路过的人，或由其他系统控制的其他组自动机器人。机器人可以从行人身边通过，但必须减速或暂时停下。在行人众多的走廊中，穿行所需的时间会比原先预期的更长。因此，通过走廊所需的时间呈随机分布，这被称为偶然不确定性。另一方面，认知不确定性是指通常无法获知每条走廊旅行时间的概率分布。本文旨在提出一种高效方法，使自动配送机器人能够在偶然不确定性和认知不确定性并存的情况下在建筑物走廊中运行。

多智能体路径规划（MAPF）已在仓库 [25] 和机场运营 [15]中得到研究和应用。基于冲突的搜索（CBS）[22] 及其扩展方法已被提出，用于高效解决多智能体路径寻找问题（[7], [10]）。该问题的应用范围已从网格图扩展到现实世界场景。连续型CBS [1]将具有连续行驶时间的图纳入考量，而大智能体多智能体路径规划 [11] 则考虑具有物理形状的智能体。Cohen等人 [5] 将该问题扩展至运动规划任务。Liu等人 [12] 则将其扩展至取送任务。

然而，建筑物内交通状况的偶然不确定性和认知不确定性，使得机器人操作系统难以实现完全自动化。近期关于 p-Robust CBS [2] 和 STT-CBS [20] 的研究，通过使用概率分布对旅行时间的延迟进行建模，成功解决了偶然不确定性问题。STT-CBS 使用伽马分布对延迟进行建模，并生成冲突概率低于某一特定值的智能体行驶路径。然而在实际应用中，分布的参数并不明确或不稳定。因此，机器人实际上会在走廊中彼此卡住，需要远程算子监控并解决碰撞。如图1所示，远程算子会将机器人移动到附近的撤退区域（例如走廊中的凹处或障碍物之间的小空间），以便另一台机器人通过。出于安全考虑，自动运行的机器人会与障碍物和墙壁保持一定距离，因此无法进入撤退区域。不过，远程算子可以操控机器人缓慢进入撤退区域。此类监控和远程操作耗时很长，因此单个远程算子难以监控多台机器人。

本文主要介绍了应对这类情况的两种方法：1）参数更新（PU）：使用贝叶斯推断自适应更新随机旅行时间概率分布的参数，而非预先设定这些参数；2）在线重新规划（OR）：使用更新后的参数重新计算并更新机器人的行驶路径。

我们为具有连续且随机旅行时间的图，构建了一个在线版本的多智能体路径寻找，并提出一种算法，通过为 STT-CBS 引入在线重新规划和参数更新，高效地解决该问题。我们还通过在明确的不确定性假设下进行仿真，展示了该算法的重要性。

2 相关工作

2.1 不确定性下的多智能体协调

Zhou 和 Tokekar [27] 回顾了近期关于不确定性、故障和对抗攻击下多智能体协调的研究。Nam 和 Shell [16] 研究了多机器人任务分配问题，其中每种机器人-任务配对的成本均为随机分布。Yang 和 Chakraborty [26] 研究了机会约束背包问题，其中物品的权重是随机的。

若干研究探讨了不确定性下的多智能体路径寻找。其中大多数基于网格图，并将智能体建模为以一定的延迟概率停留在同一单元格，而非移动到下一个单元格，例如 Ma 等人 [14]的研究。Wagner 和 Choset [24]提出了 UM*算法，该算法在联合信念空间中搜索从起始信念状态到目标信念状态的路径。他们通过采用随机重启（置换UM*）改进了该算法。

Atzmon 等人 [2] 提出了 p-RobustCBS 算法，该算法能够找到最短的 p-鲁棒路径，并且该路径能够以大于 p 的概率无冲突地执行。他们提出了两种验证器，用于计算约束树节点中路径的全局冲突概率，并得出结论：蒙特卡洛验证器表现更好。他们将三叉树用作 CBS 最优算法的搜索树，并在启发式变体（Greedy p-Robust CBS）中使用二叉树。

与上述两项研究不同，Peltzer 等人 [20]使用非单位时间图，并将每个节点处智能体的延迟建模为服从伽马分布的连续正时间值。他们定义了冲突概率，并在 CBS 算法运行期间通过蒙特卡洛模拟计算该概率，以找到满足智能体成对冲突概率小于阈值 ϵ的解。

上述所有研究都假设不确定性能够得到准确建模；因此，它们考虑了偶然不确定性，却未考虑认知不确定性。然而，在实际情况下，有时无法在运行前获得有关延迟时间的信息，或者可用的观测数量有限。我们通过利用执行计划时观测到的延迟来更新模型参数，以应对这一问题。

尽管不能直接应用于我们的问题，但在多智能体马尔可夫决策过程（MMDP）[3],的背景下，尤其是在使用基于模型的强化学习（MBRL）解决问题时，有必要处理动力学模型学习中的认知不确定性。为提高学习效率，研究人员正在研究同时纳入偶然不确定性和认知不确定性的模型正在开展对偶然不确定性和认知不确定性的研究，以提高学习效率（ [4], [18], [17]）。

2.2 多智能体路径寻找的在线重新规划

一些研究通过在线重新规划来处理多智能体路径寻找。Švancara 等人 [23] 和 Ho 等人 [8] 定义了一类问题：在计划执行过程中，新的智能体会在未知时间出现。Ma [13]从哪些智能体可以更新当前计划，以及更新后计划的质量这两个角度，对这类问题的算法进行了理论分析。

Shahar 等人 [21] 考虑了一类多智能体路径寻找问题，其中每次移动的执行时间都有上下界。随后，他们针对这一问题设定提出了能够找到最优解的算法，并保证对于上下界之间的所有执行时间值，解中都不存在冲突；同时还提出了在线重新规划方法。他们的实验结果表明，解的代价可以降低。

Levy 等人 [9]通过考虑这样一种情形，将不确定性引入多智能体路径寻找：在每个时间步，智能体都有一定概率偏离计划行动。随后，他们提出了一种在线求解方法。当检测到当前计划中存在潜在冲突时，该方法会重新计算涉及该潜在冲突的智能体的路径，并据此更新计划。他们的实证实验表明，在线更新能够有效避免冲突。

Okumura 等人 [19] 通过在线分布式规划，研究了一类多智能体路径寻找问题：智能体以一定概率停滞而非在每个节点前进，因此会产生延迟。

上述研究表明，在线重新规划对于存在不确定性的多智能体路径寻找是有效的。然而，这些研究的结果仅适用于网格图，因此难以应用于现实情境（例如旅行时间呈连续分布的情况）。在本研究中，我们提出了一种适用于非单位时间图的在线重新规划方法。

3 在线重新规划与参数更新

3.1 问题设置

3.1.1 随机旅行时间模型。我们考虑一个连通的双向图 $G=(V, E)$，由一组 E边和一组 V顶点构成。边 e的默认旅行时间 $w(e)$ 等于边长。我们有 N个智能体，智能体 $a_i \in \{1...N\}$ 从起始顶点 $start_i \in V$移动到目标顶点 $goal_i \in V$。智能体 a_i被分配了一条路径 $p_i = \{c_1, \ldots\ldots, c_{n_i}\}$，该路径由 n_i条命令$c_j = (u_j, v_j, d_j)$ 构成，这些命令可以是移动命令，满足

$$u_j \neq v_j, (u_j, v_j) \in E, d_j = w(u_j, v_j)$$

或一个等待命令，使得

$$u_j = v_j, u_j \in V, d_j \in R^+.$$

当 $u_1^i = start_i$、 $v_{n_i}^i = goal_i$ 和 $v_{j-1}^i = u_j^i$满足 $j \in 2...n_i$时，路径有效。

我们使用伽马分布来建模随机旅行时间。每条边 $e \in E$ 都有一个形状参数 $a(e)$ 和一个尺度参数 $b(e)$。边的实际旅行时间为 $w(e) + x$，服从 $x \sim \text{Gamma}(a(e)\ b(e))$。

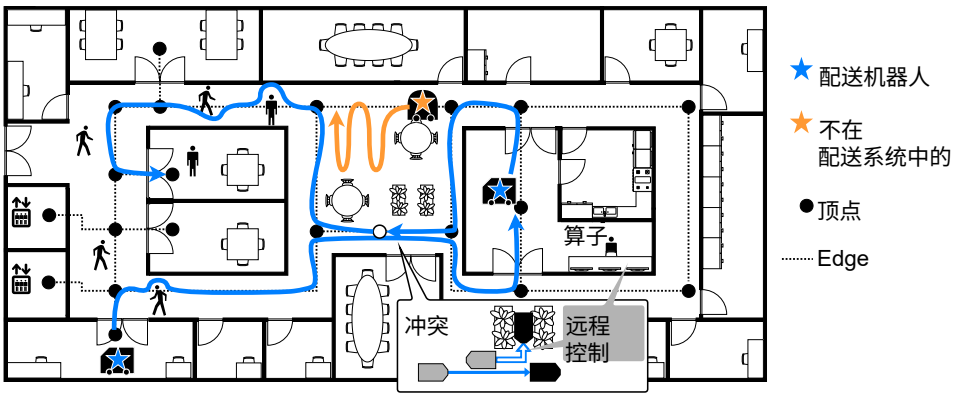

图1：机器人配送服务所提操作系统的概览。自动机器人（★）将产品从商店送到大楼会议室内的客户手中，同时由远程操作员监控多台机器人。机器人在行进时会避开行人以及由其他系统控制的清洁机器人（★）。如果由于意外延迟而发生冲突（如顶点◯所示），远程操作员会手动操控机器人，使其退至附近的避让区域（本例中为绿植之间），让另一台机器人通过。

3.1.2 交通规则与冲突解决。此处介绍适用于图 G 的交通规则。首先，每个顶点 $v \in G$ 最多只能由一个智能体占据。其次，当反向边 (v,u) 上有一个或多个智能体时，任何智能体都不能进入边 $e=(u,v) \in G$。违反这些规则即视为冲突。在路径执行过程中，随机延迟可能导致意外冲突。此类冲突将按照以下规则解决：

(1) 顶点冲突：如果一个智能体 a_i 在时间 t 完成对边 (u,v) 的遍历，而另一个智能体占据了顶点 v，则智能体 a_i 会停留在边上并向远程操作员求助。随后，智能体 a_i 会在时间 $t+w((u,v)) * C_{penalty}$ 从边 (u,v) 上移除，其中 $C_{penalty} \in R^+$ 是惩罚常数。如果没有智能体正在遍历边 (u,v)，则相反方向的边 (v,u) 此时变为可用。如果智能体 a_i 有待执行的下一条移动命令 $c=(u_{next},v_{next},d)$，且边 (u_{next},v_{next}) 可用，则该智能体会在时间 $t+(w(u,v)+w(u_{next},v_{next})) * C_{penalty}$ 进入该边。如果此时该边不可用，智能体 a_i 将在该边下次变为可用时进入。

（2）边冲突：当一个智能体 a_i 尝试进入边 (u,v)，而反方向的边 (v,u) 被一个或多个智能体占据时，智能体 a_i 必须等待，直到边 (v,u) 变为空闲。

图2描述了顶点冲突的消解过程。发生顶点冲突时，远程操作员会将被阻塞的机器人从机器人操作系统中分离出来，并手动控制该机器人。远程操作员将机器人移至附近的等待空间（例如走廊中的凹处），以便另一台机器人通过。由于自动运行的机器人出于安全考虑会与障碍物和墙壁保持一定距离，因此无法进入等待空间。然而，远程操作员可以操控机器人，尽管速度较慢，

进入等待空间。这种缓慢的手动操作通过惩罚 $C_{penalty}$ 建模。

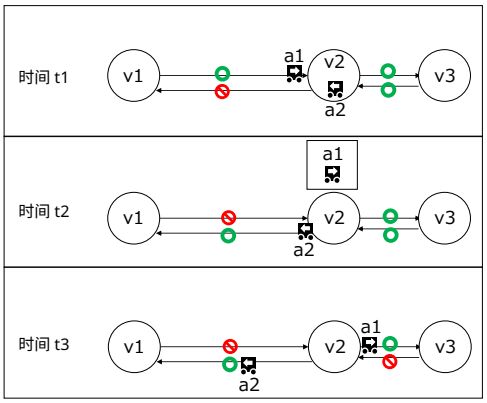

图2：解决顶点冲突的示例操作。在时间 t_1，智能体 a_1 试图进入被智能体 a_2 占据的顶点 v_2。在时间 $t_2 = t_1 + w((v_1,v_2)) * C_{penalty}$，智能体 a_1 从边 $e(v_1,v_2)$ 上移除。边 $e(v_2,v_1)$ 变为可用，因为没有其他智能体正在通过边 $e(v_1,v_2)$，智能体 a_2 开始通过 $e(v_2,v_1)$。在时间 $t_3 = t_2 + w((v_2,v_3)) * C_{penalty}$，智能体 a_1 被插入边 $e(v_2,v_3)$，而反向边 $e(v_3,v_2)$ 变为不可用。

3.1.3 在线多智能体路径规划。我们考虑一种集中式机器人操作系统。中央控制器掌握每个智能体当前正在执行的命令，并且可以随时更新所有智能体的路径；但是，当前正在运行的命令必须保持不变，更新后路径中的其他命令必须与之保持一致。

离线求解所用的离线多智能体路径规划问题实例由图 G、$start_i$ 和 $goal_i$ 组成，每个智能体为 $i \in 1 \ldots N$。智能体在时间 0 从各自的起始顶点出发。现在考虑在时间 t 进行在线重新规划的问题实例。此时，智能体 a_i 正在执行一条移动或等待命令

$c_i(t) = (u_i(t)\ v_i(t)\ d_i(t))$。对于移动命令，由于边遍历期间发生延迟，命令的完成时间不确定；但当前命令 $st_i(t)$ 的开始时间和计划完成时间（假设没有延迟时的完成时间）$ft_i(t) = st_i(t) + d_i(t) = st_i(t) + w(u_i(t)\ v_i(t))$ 均已知。我们将这些作为问题输入。智能体 a_i 的固定命令为 $(st_i(t)\ u_i(t)\ ft_i(t)\ v_i(t))$，在线更新后的路径必须以该命令为起始。

在在线多智能体路径寻找问题中，时间 t的问题实例由以下内容组成：

- 图 G;
- 固定 命令 $st_i(t)$ 的开始时间;
- 固定 命令 $u_i(t)$ 的起始顶点;
- 固定 命令 $ft_i(t)$ 的结束时间;
- 固定 命令 $v_i(t)$ 的终止顶点；• $i \in 1...N$.的目标顶点 $goal_i$；以及 • 计算时间限制 t_{limit}。

在线多智能体路径寻找问题的解由每个智能体的一系列命令组成 $path_i = c_1, c_2, ..., c_{n_i}$，这些命令从固定 命令开始，使得 $c_1 = (u_1, v_1, d_1) = (u_i(t)\ v_i(t)\ ft_i(t) - st_i\ t())$。求解算法旨在找到一个解，使包括固定 命令在内的命令均可无冲突地执行。

在在线重新规划场景中，智能体移动的同时，中央控制器会计算规划方案。如果计算完成前某个智能体的状态发生变化，那么基于旧状态生成的新规划方案可能无效；因此，问题实例为计算时间设置了时间限制。如果算法未能在时间限制内返回解，规划方案就不会更新。

3.2 求解算法

我们用于在线多智能体路径规划的更新路径算法基于 STT-CBS [20]。算法 1 着重展示并介绍了本节所述的修改。

3.2.1 贪心启发式。 在原始 STT-CBS 的最佳优先搜索中，会扩展代价最低的节点。与此不同，我们在此引入贪心 STT-CBS（GSTT-CBS），选择冲突概率最大值最小的节点。具体而言，对于路径为 $p_i = \{c_1, ..., c_{n_i}\}$、对应智能体 $a_i \in \{1 \dots N\}$ 的每个约束树节点，解中所有命令对的最大冲突概率

$$P_{max} = \max_{c_k, c_l} \{P(c_k, c_l) : c_k \in p_i, c_l \in p_j, i \neq j, (i, j) \in \{1 \dots N\}\}$$

通过蒙特卡洛模拟计算得出，其中 $P(c_k, c_l)$ 是命令 c_k 和 c_l 之间的冲突概率。具有最小 P_{max} 值的节点会从优先队列中弹出并扩展。贪心 p-鲁棒 CBS[2] 中也引入了相同的启发式。实验表明，这种方法能有效缩短计算时间。

算法必须在计算时间限制内返回一个解；因此，即使算法达到计算时间限制时，该解仍存在冲突，高层搜索也会返回优先队列顶端的解。

3.2.2 固定命令。低层搜索是在时间扩展图 G = (V,E) 上运行的 A* 算法。时间范围为 $T \in Z^+$, V $= V \times \{0, 1, \dots T\}$, E $= \{((v_1, \tau_1), (v_2, \tau_2)) \in$ V $\times$ V: $(v_1, v_2) \in E \wedge \tau_2 - \tau_1 = w(v_1, v_2)\}$。添加一条禁止 traversing 边 $(u, v) \in E$从时间 t到 $t + w(u, v)$ 的约束，相当于禁止（即移除）G 中的边 $((u, t), (v, t + w(u, v)))$。

算法 1 高层搜索

```
函数find_solution(在线多智能体路径查找实例)
    初始化根节点 R
    R.constraints ← ∅
    R.solution ← 使用 A* 查找 R 的低层解
    R.cost ←计算解的代价
    将 R 插入优先队列
    当 优先队列 ≠ ∅ 时
        P ← 弹出冲突概率最小-最大值的节点
        P.conflict ← 获取首个冲突(P)
        如果 P 无冲突 则
            返回 P.solution
        否则如果 超时则
            返回 P.solution
        结束 if
        对于所有 (a_i, c_i) 涉及 P.conflict执行
            如果 命令 c_i 已为智能体 a_i 固定则
                继续
            结束 if
            创建子节点 C
            C.const ← P.const + 常量(P.conflict, a_i)
            使用 A* 查找 C 低层解
            C.cost gets 查找解的代价
            将 C 加入优先队列
        结束循环
    结束循环
    返回 x
结束函数
```

由于每个智能体的路径都必须以固定 命令开头，因此低层搜索会查找从时间扩展顶点 $(v_i(t)\ ft_i(t))$ 到 $\tau \in 1...T$ 的一个时间扩展顶点 $(goal_i, \tau)$ 的最短路径，然后将固定 命令添加到路径的开头。当 CBS 节点中出现冲突，且冲突发生在固定命令与非固定 命令之间时，不会创建两个子节点，而是创建一个带有额外约束的子节点，以禁止该非固定 命令，并将其添加到搜索树中。

3.3 在线重新规划

关于重新规划的时间以及选择哪些智能体进行重新规划，可以采用多种在线重新规划策略。我们采用固定间隔策略，即每隔固定时间 t_{ci} 更新所有智能体的计划。也可以采用其他策略（例如，当命令的执行时间延迟超过某个值时更新所有智能体的计划，或者在预计两个智能体未来会发生冲突时更新这两个智能体的计划）。

3.4 参数更新

STT-CBS算法使用蒙特卡洛模拟来评估路径之间的冲突概率。我们假设真实的

延迟分布参数 a（e）和 b（e）对于 $e \in E$而言是未知的。我们只有先验知识a_{prior}（e）和 b_{prior}（e）对于$e \in E$，并根据智能体 traversing 边 e时获得的 m 延迟观测值$\mathbf{x}$ =（x_1，$\ldots x_m$）来估计这些参数。为简化记号，现省略索引 e（例如，$a = a$（e）$andb = b$（e））。令 a 和 b 的先验分布为

$$p(a, b \mid p, q, r, s) = K \frac{p^{a-1} \exp(-q/b)}{\Gamma(a)^r b^{as}}$$

其中归一化常数为 K，参数为 p、q、r、s。后验分布为

$$\begin{aligned} P(a, b \mid \mathbf{x}) &\propto \prod_{i=1}^{m} \text{Gamma}(x_i|a, b) * p(a, b|p, q, r, s) \\ &= \frac{p'^{a-1} \exp(-q'/b)}{\Gamma(a)^{r'} b^{as'}}, \end{aligned}$$

其中 $p\prime = p \prod_{i=1}^{m} x_i$、$q\prime = q + \sum_{i=1}^{m} x_i$、$r\prime = r + m$ 和 $s\prime = s + m$ 是贝叶斯更新 [6] 后的参数。对数后验分布求导并令其为零

$$\partial_a \ln P(a, b \mid \mathbf{x}) = \ln p' - r' \frac{\Gamma'(a)}{\Gamma(a)} - s' \ln b = 0$$

$$\partial_b \ln P(a, b \mid \mathbf{x}) = \frac{q'}{b^2} - \frac{as'}{b} = 0$$

可得最大后验估计量 a_{map} 和 b_{map} 的联立方程；

$$\begin{aligned} a_{map} &= \Psi^{-1}\left(\frac{\ln p' - s' \ln b_{map}}{r'}\right), \\ b_{map} &= \frac{q'}{a_{map} s'} \end{aligned}$$

其中 Ψ^{-1} 是反双伽马函数。从方程中消去 a_{map} 可得：

$$f(b) := (\ln p' - s' \ln b) - r' \Psi\left(\frac{q'}{bs'}\right).$$

我们使用牛顿法

$$b \leftarrow b - \frac{f(b)}{f'(b)}$$

以求得 b_{map} 的值，使得 $f(b_{map}) = 0$。

先验分布的超参数为 a_{prior}、b_{prior}、r 和 s。参数 p 和 q 可计算如下

$$\begin{aligned} p &= \exp\left(r\Psi(a_{prior}) - s \ln b_{prior}\right), \text{and} \\ q &= a_{prior} b_{prior} s. \end{aligned}$$

3.5 总体算法流程

算法 2 描述了参数更新和在线重新规划的总体流程。我们使用离散事件模拟器，根据第 3.1 节所述的随机延迟和交通规则，模拟智能体的移动。GSTT-CBS 求解器使用伽马概率分布模型表示每条边的延迟 e，其参数先验为 $a_{prior}(e)$ 和 $b_{prior}(e)$，最大后验估计量为 $a_{map}(e)$ 和 $b_{map}(e)$。模拟器逐一处理事件（例如，智能体到达顶点或进入边）。每当智能体进入一条边时，便根据真实参数为 $a(e)$的伽马分布随机生成延迟

算法 2 总体算法

```
使用先验知识初始化求解器
对于所有 问题列表中的问题实例 p_ini 执行
    使用 p_ini 初始化模拟器 Env
    环境.规划 ← 求解器.寻找_解(p_ini)
    当 未 环境.已_完成()时
        延迟观测 obs ← 环境.步骤()
        求解器.更新_参数(obs)
        如果 环境.重新规划_策略()则
            问题实例 p ← 环境.转换为_问题()
            Env.PLAN ←Solver.FIND_SOLUTION(p)
        结束如果
    结束当
结束循环
```

和 $b(e)$。在每个仿真步骤中，求解器利用观测到的延迟更新模型参数 $a_{map}(e)$ 和 $b_{map}(e)$。当满足重新规划策略时（即自上次更新以来经过了一定时间段），模拟器将智能体的当前状态转换为在线多智能体路径规划问题实例，将其输入求解器，然后使用求解器返回的解更新智能体的行动计划。

我们考虑这样一种情形：针对同一张地图，有一系列任务（即每个智能体的起点和目标位置），并按顺序完成这些任务。求解器利用上一个任务中观察到的延迟时间更新概率分布参数，并据此为下一个任务生成计划。简言之，每当智能体经过一条边并获取信息后，规划所依据的模型就会变得稍微准确一些。

4 实验

本节介绍实验设计及其结果。首先，我们在具有不确定随机旅行时间的非网格图上评估我们的方法，并比较贪心启发式、在线重新规划（OR）和参数更新（PU）的使用效果。接下来，通过使用不同的参数值或额外的问题实例，分析每种技术的影响。算法使用 C++ 实现，并在一台配备 32 GB RAM、搭载英特尔酷睿 i7-10850H 处理器（2.7 GHz）的笔记本电脑上运行。

4.1 实验设计

4.1.1 问题实例。我们为实验随机生成非网格图。顶点的坐标为（x，y），度为 δ，满足

$$\begin{aligned} x &= \text{Random}(\{0, \ldots, 99\}), \\ y &= \text{Random}(\{0, \ldots, 99\}), \\ \delta &= \text{Random}(\{2, \ldots, 4\}). \end{aligned}$$

对于每个顶点 v，在 v 与其 δ 个最近顶点之间生成边。图3展示了生成的图。

边e 的延迟概率分布参数随机生成如下：

- 均值：m_e = 随机($\{3, \ldots, 9\}$) $* 1.0$,
- 方差：v_e = 随机($\{1, \ldots, 4\}$) $* 0.1$,

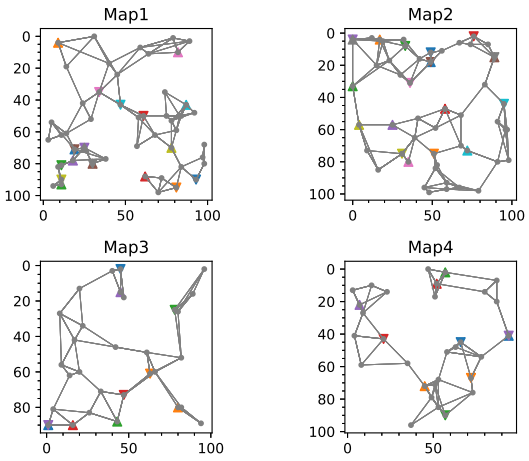

图3：实验中使用的图。Map1 和地图2有50个节点和10个智能体。地图3和地图4有30个节点和5个智能体。三角形表示智能体起点和目标顶点的一个示例。

- 真实形状参数： $a(e) = m_e^2 v_e$，
- e 真实尺度参数： $b(e) = v_e / m_e$。

如果得到的图不是连通图，则重复相同的过程，直到得到连通图。每张地图随机生成 100 个任务。起点和目标顶点从顶点 V 中随机选取，确保智能体之间不共用起点和目标顶点。

4.1.2 超参数。 其他超参数设置如下：STT-CBS 和 GSTT-CBS 中两个智能体之间成对冲突概率的阈值 ϵ 为 0.01；在线重新规划 t_{ci} 的时间间隔为 100；算子 $C_{penalty}$ 进行冲突消解时的惩罚常数设为 1.0；初始规划和在线重新规划 t_{limit} 的计算时间限制设为 10 秒；先验分布的参数为 $[a_{prior}, b_{prior}, r, s] = [1.0, 0.2, 0.1, 0.1]$ r s 0 1 。

将 和 设置为 。意味着参数估计时会比先验知识更侧重于观测值。

4.2 实验结果

表1展示了执行100项任务期间的平均冲突数和平均流时间。流时间定义为智能体到达其目标顶点所需时间的总和。表中还展示了初始解的平均计算时间（秒），以及在计算初始解期间超时的任务比例。

从冲突数量和流时间来看，实验结果表明，我们提出的方法优于现有方法（请注意高亮显示的列）。三种方法（即贪心启发式、在线重新规划和参数更新）均被证实有效，因为缺少其中任何一种都会导致性能下降。在下一节中，我们将详细分析每种方法的作用，以回答这些方法为何以及如何有效减少冲突数量和流时间。

从初始解的计算时间和超时率来看，我们提出的方法优于现有方法。计算时间足够短，能够满足假定的系统要求。

4.3 分析

4.3.1 贪心启发式的作用。我们在此考察表1中Map1上的模拟结果，以阐明贪心启发式为何有效。我们将100项任务分为“简单”和“困难”两类。前者包括使用原始CBS算法能在时间限制内找到无冲突解的任务；后者包括CBS算法运行超时的任务。表2展示了各算法在这两组任务上的表现。

对于“简单”组中的10项任务，STT-CBS未能找到无冲突解。这是因为与CBS相比，STT-CBS检测冲突所需时间更长，因此在时间限制内只能扩展较少的约束树节点。这解释了为什么在平均处理100项任务时，STT-CBS有时不如CBS。贪心启发式提升了算法在“困难”任务组中的表现，使其能在更多任务中找到无冲突解（GSTT-CBS为14项，而STT-CBS为0项）。

4.3.2 PU 的影响。我们开展了一项补充实验，以明确如何更新延迟概率分布的参数。实验使用一张包含 50 个节点的地图和 1000 个任务，这些任务由 10 个智能体执行。任务按顺序求解并仿真。我们比较了三种情况：（1）带参数更新的 GSTT-CBS；（2）不带参数更新的 GSTT-CBS；（3）无误差的 GSTT-CBS，这是一种假设情形，即从一开始就已获得真实参数。每次仿真后，我们都会计算地图中所有边的估计参数 a_{map} 和 b_{map} 的均方根误差 (RMSE)，并计算每条边 $E_a = \frac{|a - a_{map}|}{|a - a_{prior}|}$ 和 $E_b = \frac{|b - b_{map}|}{|b - b_{prior}|}$ 的误差率。

图4总结了分析结果。随着已执行任务数量的增加，估计值的均方根误差会降低。尽管执行 1000 个任务后，均方根误差仍未降至 0，但带 PU 的 GSTT-CBS（第三、第四行中的蓝线）的冲突数量和流时间在约 200 个任务后已达到最小值。第五行中各边观测数量的直方图通常显示，只有少数边被频繁使用，而许多其他边被经过的概率较低，因此观测到其延迟的机会也较少。图中最下面两行表明，频繁使用的边的误差会迅速减小。这些边上的估计值（红色所示）仅在 100 次迭代后才变得准确。一些不重要的边即使经过 1000 次迭代后仍不准确（蓝色所示）。

结果表明，尽管经过 1000 次试验后所有边的均方根误差仍然存在，但仅经过 100 到 200 次试验，路径规划中频繁使用的边的参数就已更新至准确值。这样，参数更新便可将数据采集的工作集中于重要的边，使得即使在不了解所有边的真实延迟概率分布的情况下，也能在少量试验后生成无冲突的解决方案。

表1：贪心启发式、在线重新规划和参数更新的效果。粗体和下划线表示最佳结果，下划线表示次佳结果。

		GSTT-CBS				STT-CBS				CBS	
		采用在线重新规划 ,PU（我们的方法）	采用参数更新	采用在线重新规划		采用在线重新规划 ,PU	采用参数更新	采用在线重新规划	[20]	采用在线重新规划	
贪心启发式		✓	✓	✓	✓						
在线重新规划		✓		✓		✓		✓		✓	
参数更新		✓	✓			✓	✓				
冲突数量	Map1	**1.81**	1.97	2.68	2.92	2.96	3.21	3.08	3.56	2.90	3.11
	Map2	**0.87**	0.96	2.05	1.97	2.15	1.96	2.34	2.38	2.22	2.27
	Map3	**0.05**	**0.05**	0.12	0.14	0.08	0.09	0.26	0.31	0.27	0.28
	Map4	**0.15**	0.22	0.91	0.93	0.59	0.64	1.15	1.32	1.27	1.35
流时间	Map1	**1334.91**	1342.72	1383.96	1390.50	1418.99	1423.95	1408.13	1431.77	1388.45	1396.91
	Map2	**1175.94**	1182.06	1273.48	1247.17	1274.46	1249.78	1287.13	1276.63	1279.06	1267.58
	Map3	**478.22**	478.67	481.60	481.16	477.09	477.92	489.94	493.28	493.70	492.07
	Map4	**620.71**	626.25	669.83	665.34	648.10	647.03	690.42	696.03	700.34	698.03
计算时间（秒）	Map1	3.70	3.67	**3.04**	3.07	6.23	6.26	5.05	5.10	4.03	4.04
	Map2	**2.65**	2.71	3.08	3.04	5.14	5.02	4.50	4.42	4.06	4.07
	Map3	**0.17**	0.18	0.18	0.19	0.49	0.49	0.40	0.40	0.36	0.36
	Map4	0.66	**0.50**	1.04	0.95	2.34	2.43	2.93	2.97	2.91	2.89
超时率	Map1	0.27	0.24	**0.22**	**0.22**	0.54	0.54	0.44	0.44	0.34	0.34
	Map2	**0.16**	0.17	0.21	0.20	0.44	0.41	0.39	0.39	0.35	0.35
	Map3	**0.01**	**0.01**	**0.01**	**0.01**	0.03	0.03	0.03	0.03	0.03	0.03
	Map4	0.03	**0.01**	0.07	0.06	0.17	0.17	0.24	0.26	0.26	0.26

表2：贪心启发式对 Map1 的影响。

		GSTT-CBS	STT-CBS	CBS
Easy	找到的数量	64	56	66
	超时数量	2	10	0
	流时间	1190.56	1191.18	1166.62
困难	找到数量	14	0	0
	超时数量	20	34	34
	流时间	1778.59	1898.78	1843.93

表3：在线重新规划间隔对 Map1 的影响。

t_{ci}	10	25	50	75
流时间	1242.75	1264.72	1312.24	1353.99
冲突数量	0.71	1.06	1.75	2.33
初始计算时间（秒）	3.02	3.03	3.01	3.20
在线计算时间（秒）	4.38	1.23	0.50	0.28

然而，与理想规划相比，仍有一定的改进空间（第三行和第四行中蓝线与绿线之间的差距）。结果表明，路径规划会受到不常经过的边的影响，因此有必要通过探索来获取这些边的延迟信息。我们将在第6节讨论这一点。

4.3.3 在线重新规划的效果。我们在 Map1 上重复实验，采用带有在线重新规划的 GSTT-CBS，并设置不同的在线重新规划时间间隔 $t_{ci} = \{10, 25, 50, 75\}$。本实验旨在揭示在线重新规划的效果。结果见表3。表中的在线计算时间是重规划阶段总计算时间的平均值（即，如果一项任务中更新了 10 次计划，则显示这 10 次计算的总时间）。结果表明，较短的更新间隔会带来更少的冲突和更短的流时间。请注意，此处不进行参数更新，因此规划基于存在建模误差的延迟概率模型。即使延迟模型不准确，频繁重规划仍可减少冲突数量。然而，这会增加在线重新规划阶段的计算时间，并提高因计算过程中智能体状态发生变化而导致计算结果无效的风险。我们将计算时间限制设为 $t_{limit} = 10$ 秒，并假设智能体状态在此期间不会发生变化。在该假设成立的情况下，频繁重规划可以有效减少碰撞。

4.4 大规模实例上的性能

我们开展了一项额外实验，以回答所提出方法的可扩展性问题。按照前文所述的方法生成一个包含100个顶点的图。进行仿真，其中包含100个任务，每个任务由 $t_{ci} = \{10、100\}$个智能体组成，任务数量分别为10、15和20。测试了采用在线重新规划和参数更新的GSTT-CBS，其中固定更新间隔取不同值。图5展示了100个任务的平均流时间和平均冲突数量。随着智能体数量增加，在时间限制内完成计算变得更加困难。使用15个智能体的任务中，超过60%发生超时；使用20个智能体的任务中，超过90%发生超时。然而，频繁的重新规划 $t_{ci} = 10$ 使碰撞数量保持在较低水平。

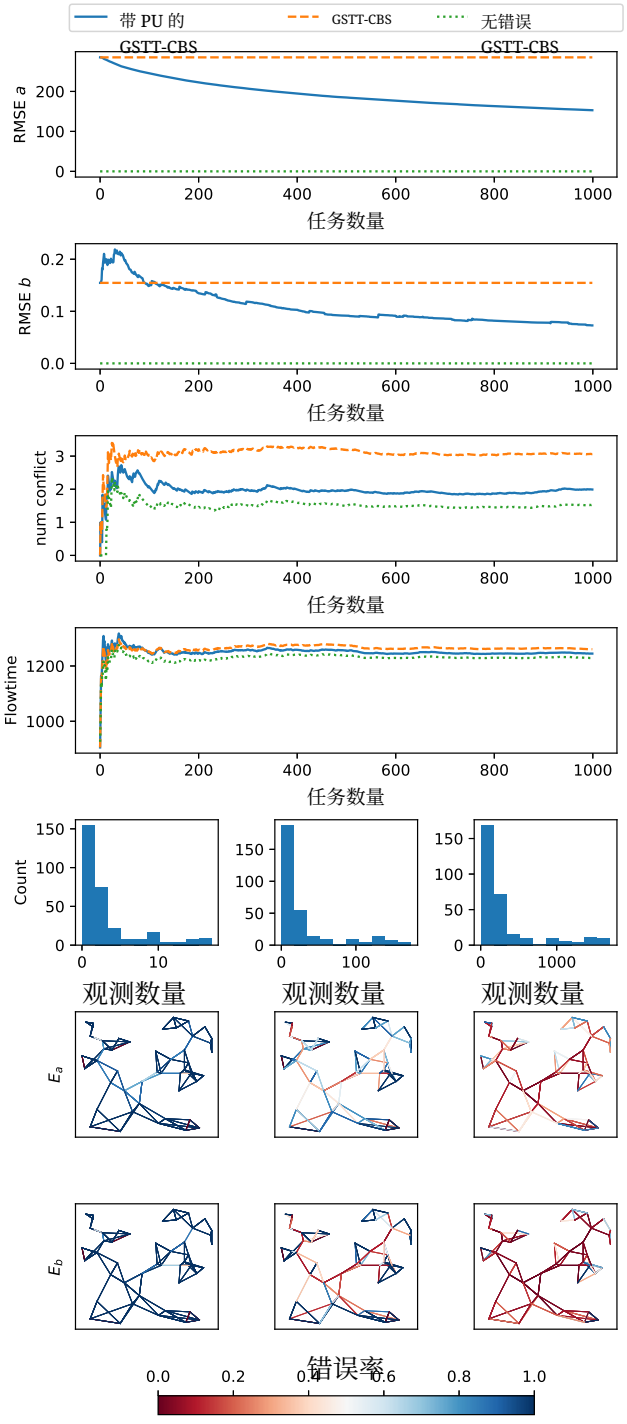

图4：第一行和第二行：每次任务执行后，所有边上估计的 a 和 b 值的均方根误差。
第三行和第四行：每次执行任务前的冲突数量和流时间平均值。

执行 10 个（左）、100 个（中）和 1000 个（右）任务后，每条边的观测数量直方图。

底部两行：地图上绘制的每条边的错误率，分别对应10、100和1000个任务之后的结果（从左到右）。

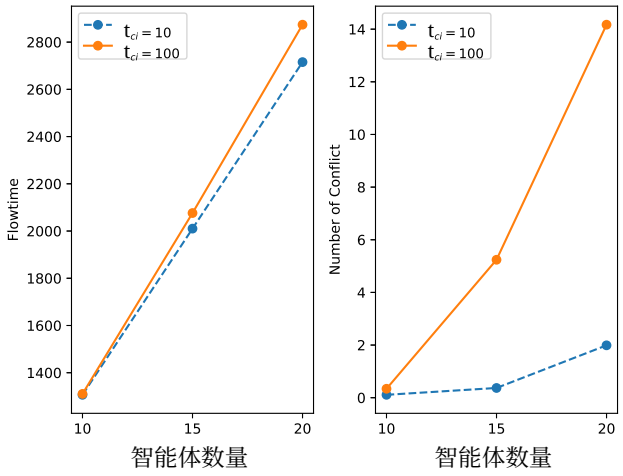

图 5：在大型实例上，GSTT-CBS 结合 OR 和 PU 的结果。

5 结论

在本工作中，我们研究了非单位时间图上具有随机旅行时间的多智能体路径寻找问题，其中智能体的旅行时间呈随机分布（即偶然不确定性），但真实分布未知（即认知不确定性），必须根据计划执行期间获取的数据进行估计。为应对这一问题，我们提出了在线重新规划和参数更新方法。据我们所知，这是首项通过引入这两类不确定性来扩展多智能体路径寻找问题，以应对建筑物内自动化配送系统实际情况的研究。

我们对在线多智能体路径寻找问题进行了形式化描述。在该问题中，更新后的计划必须与当前已执行的命令保持一致，并且必须在一定的时间限制内返回解，以模拟自动化配送服务的现实场景。我们使用贪心启发式，在有限的计算时间内寻找发生碰撞可能性较低的解。

随后，我们使用边的旅行时间为连续值且具有延迟概率分布的非网格图，将我们的方法与现有方法进行了比较。结果表明，我们提出的方法能够减少执行过程中发生的冲突数量以及智能体实际旅行时间的总和。由于参数更新优先针对频繁使用的边进行，因此只需少量学习任务即可取得这一性能。

6 未来工作

未来工作有许多可能的方向。

第 4.3.2 节的分析结果表明，有必要进一步研究主动探索方法：在权衡探索与利用的同时，使用不在最优路径上的边，以获取有关该边延迟概率的信息。

在必须于有限时间内制定计划的现实世界场景中，优先避免超时而非追求最优性是有效的。因此，采用其他启发式求解器可能具有较好的前景。利用先前计算结果来应对在线重新规划情况的专用算法也可能具有研究价值。另一种可能的方法是通过调整命令执行时间，在计划执行期间避免碰撞，而不是重新计算所有智能体的计划。

在随机旅行时间不确定的情况下，更新涉及预计近期将发生冲突的有限数量智能体的计划，可以作为未来的研究任务。

参考文献

[1] Anton Andreychuk、Konstantin Yakovlev、Dor Atzmon 和 Roni Stern。2019。《连续时间多智能体寻路》。载于 第二十八届国际人工智能联合会议论文集，*IJCAI-19*。国际人工智能联合会议组织，39–45。https://doi.org/10 24963 2019 6 。/ijcai。/ Dor Atzmon、Roni Stern、[2] Ariel Felner、Nathan R. Sturtevant 和 Sven Koenig。2020。《概率鲁棒多智能体寻路》。国际自动规划与调度会议论文集30，1（2020年6月），29–37。https://ojs.aaai.org/index.php/ICAPS/article/view/6842 [3] 1996 Craig Boutilier。.《多智能体决策过程中的规划、学习与协调》。载于 第六届理性与知识的理论方面会议论文集（荷兰）*(TARK '96)*。Morgan Kaufmann Publishers Inc.，美国加利福尼亚州旧金山，195–210。[4] Kurtland Chua、Roberto Calandra、Rowan McAllister 和 Sergey Levine。2018。通过概率动力学模型，仅用少量试验进行深度强化学习。神经信息处理系统进展31（2018）。[5] Liron Cohen、Tansel Uras、TK Satish Kumar 和 Sven Koenig。2019。最优与有界次优多智能体运动规划。载于 第十二届组合搜索年会。[6] Daniel Fink。1997。《共轭先验纲要》。技术报告。（01 1997），25–26。https://www.researchgate.net/publication/238622435_A_共轭先验纲要[7] Graeme Gange、Daniel Harabor 和 Peter J. Stuckey。2021。惰性CBS：使用惰性子句生成进行隐式基于冲突的搜索。国际自动规划与调度会议论文集29，1（2021年5月），155–162。https://ojs.aaai.org/index.php/ICAPS/article/view/3471[8] Florence Ho、Artur Goncalves、Ana Salta、Marc Cavazza、Ruben Geraldes 和 Helmut Prendinger。2019。用于无人机交通管理的多智能体寻路：机器人赛道。（2019）。[9] Elad Levy、Guy Shani 和 Roni Stern。2022。《运动不确定条件下多智能体寻路的在线方法》。载于 国际组合搜索研讨会论文集，第15卷。299–301。[10] Jiaoyang Li、Daniel Harabor、Peter J Stuckey、Hang Ma、Graeme Gange 和 Sven Koenig。2021。多智能体寻路搜索的成对对称性推理。人工智能 301 （2021），103574。[11] Jiaoyang Li、Pavel Surynek、Ariel Felner、Hang Ma、TK Satish Kumar 和 Sven Koenig。2019。大型智能体的多智能体寻路。载于 论文集 *AAAI*人工智能会议，第33卷。7627–7634。[12] Minghua Liu、Hang Ma、Jiaoyang Li 和 Sven Koenig。2019。多智能体取送的任务与路径规划。载于 自主智能体与多智能体系统国际联合会议（*AAMAS*）论文集。[13] Hang Ma。2021。在线多智能体寻路的竞争分析。载于国际自动规划与调度会议论文集，第31卷。234–242。[14] Hang Ma、TK Satish Kumar 和 Sven Koenig。2017。具有延迟概率的多智能体寻路。载于*AAAI*人工智能会议论文集，第31卷。[15] Robert Morris、Corina S Pasareanu、Kasper Luckow、Waqar Malik、Hang Ma、TK Satish Kumar 和 Sven Koenig。2016。机场地面运行的规划、调度与监控。载于第三十届*AAAI*人工智能会议研讨会。[16] Changjoo Nam 和 Dylan A. Shell。2017。分析概率多机器人任务分配中最优分配的敏感性。*IEEE*机器人与自动化快报 2, 1 2017 193 200 10 1109 2016 2588138 ( )，–。[17] 2020 https://doi.org/。/LRA。。Masashi Okada、Norio Kosaka 和 Tadahiro Taniguchi。.贝叶斯学派的PlaNet：通过纳入贝叶斯推断重新审视并改进深度规划网络。载于*2020*年*IEEE/RSJ*智能机器人与系统国际会议（*IROS*）。IEEE，5611–5618。[18] Masashi Okada 和 Tadahiro Taniguchi。2020。用于贝叶斯模型强化学习的变分推断模型预测控制。载于机器人学习会议。PMLR，258–272。[19] Keisuke Okumura、Yasumasa Tamura 和 Xavier Défago。2021。多个移动智能体的时间无关规划。载于*AAAI*人工智能会议论文集，第35卷。11299–11307。[20] Oriana Peltzer、Kyle Brown、Mac Schwager、Mykel J. Kochenderfer 和 Martin Sehr。2020。STT-CBS：随机旅行时间下多智能体寻路的基于冲突的搜索算法。https://doi.org/10.48550/ARXIV.2004. 08025[21] Tomer Shahar、Shashank Shekhar、Dor Atzmon、Abdallah Saffidine、Brendan Juba 和 Roni Stern。2021。具有时间不确定性的安全多智能体寻路。人工智能研究期刊 70（2021），923–954。[22] Guni Sharon、Roni Stern、Ariel Felner 和 Nathan R Sturtevant。2015。用于最优多智能体寻路的基于冲突的搜索。人工智能219（2015），40–66。[23] Jiří Švancara、Marek Vlk、Roni Stern、Dor Atzmon 和 Roman Barták。2019。在线多智能体寻路。载于*AAAI*人工智能会议论文集，第33卷。7732–7739。[24] Glenn Wagner 和 Howie Choset。2017。《不确定环境下的多智能体路径规划》。国际自动规划与调度会议论文集27，1（2017年6月），577–585。https://ojs.aaai.org/index.php/ICAPS/article/view/13866[25] Peter R Wurman、Raffaello D'Andrea 和 Mick Mountz。2008。协调数百辆仓库中的协作式自主车辆。《*AI Magazine*》29，1（2008），9–9。[26] Fan Yang 和 Nilanjan Chakraborty。2018。带有应用于多机器人协作组队的最优机会约束背包问题算法。载于*2018*年*IEEE*机器人与自动化国际会议（*ICRA*）。1043–1049。https://doi.org/10 1109 2018 8461040 。/ICRA。。Lifeng Zhou 和 Pratap [27] 2021 Tokekar。.不确定和对抗性环境中的多机器人协调与规划。《*Current Robotics Reports*》2，2（2021），147–157。