arXiv:2305.05485v2 [cs.RO] 2 Oct 2023

机器人故障情况下的弹性时序逻辑规划

Samarth Kalluraya[1], George J. Pappas[2], Yiannis Kantaros[1]

摘要—**近期已有多种任务与运动规划算法被提出，用于为移动机器人团队设计路径，以完成通过形式语言（如线性时序逻辑（LTL））指定的协作性高级任务。然而，设计出的路径往往无法灵活应对机器人能力（例如感知、移动能力或操作）失效的情况。这些失效可能由意外事件（例如人为干预或系统故障）引起，进而损害任务性能。为应对这一新挑战，本文提出一种新的韧性任务规划算法，用于执行协作性LTL任务的异构机器人团队。机器人在能力方面存在异构性，而任务要求按时序／逻辑顺序在环境中的特定区域运用这些技能。所提出的方法设计出能够适应机器人能力意外失效的路径。具体而言，该方法根据机器人当前正常运行的技能重新分配子任务，同时尽可能减少对现有团队运动规划的干扰。我们通过实验和理论保证展示了所提算法的效率与韧性。**

I. 引言

线性时序逻辑（LTL）已广泛应用于机器人运动规划，用于定义超越简单到达-避障要求的多样化任务 [1], , 例如监视、协作操作和配送 [2]–[4]。针对具有已知动力学且处于已知环境中的机器人团队，文献 [5]–[10], 提出了采用时序逻辑任务的规划算法。近期对这些工作的扩展研究在假设系统动力学已知的情况下，处理了未知且动态的环境 [11]–[15]。文献 [16]–[18] 则利用机器学习方法放宽了后一假设。这些工作缺乏对机器人故障的韧性，因为它们假设机器人在部署期间的能力始终不受影响。然而，由于机器人在实际运行中面临多种不确定性来源，这一假设可能被违反，包括恶劣天气、人为干预和组件故障。

为了提高任务规划的韧性，我们针对具有以LTL公式表达的协同任务的异构机器人团队，提出了一种新的韧性任务规划算法。机器人在能力方面具有异构性，其能力可能包括移动能力、感知或操作等，而LTL公式要求它们在特定区域和/或对象处运用这些能力。给定一个LTL任务，所提出的算法能够设计出具有韧性的计划，即这些计划能够适应机器人故障包括能力丧失（例如抓取）或机器人被完全移除（例如由于电池耗尽）。计划被定义为由机器人位置和动作组成的序列。为此，系统根据机器人当前仍可正常运行的技能（如果有）重新分配子任务。一旦任务重新分配完成，就会对先前设计的路径进行最小限度的修改，以适应能力故障。所提出的方法旨在故障发生时最大限度地减少对多机器人行为的干扰，具体通过减少重新分配的次数，并尽可能避免对整个团队进行重新规划来实现。后一种做法尤为重要，因为全局任务重新分配和从头开始重新规划在计算上代价高昂，在运行时难以实现，尤其是在机器人团队规模较大的情况下。我们通过大量涉及异构地面机器人和空中机器人团队的实验，以及理论分析，证明了所提出算法在应对多种意外机器人能力故障时的效率和韧性。

相关工作：近年来，已有多种任务分配方法被提出，用于将局部线性时序逻辑任务[19], [20]或全局协作线性时序逻辑任务 [21] 的子任务（即原子命题）分配给机器人。在这些工作中，任务分配是在离线阶段完成的，即在机器人部署之前，而未考虑任务执行过程中发生的机器人故障。当机器人发生故障时，这些方法可以在线使用，将任务全局重新分配给机器人。然而，正如前文所述，在运行时进行全局任务重新分配并不切实际。与我们工作最接近的是 [22]–[24]中提出的方法。具体而言， [22] 构建了一个乘积自动机，用于对多机器人状态空间、规范空间以及可能发生的机器人故障进行建模。利用该乘积系统，可以提取能够对故障作出反应的控制策略。然而，该工作考虑的是同构机器人，而基于乘积的方法在机器人数量、环境规模、任务复杂度以及故障数量方面缺乏可扩展性。 [23], [24]提出了概念上类似的方法。例如， [23] 考虑了具有异构能力的机器人，其中机器人能够对环境与机器人状态变化作出局部响应。然而，与我们的工作不同，当机器人技能发生故障时， [23] 需要基于团队自动机进行全局任务重新分配和重新规划。

贡献：首先，我们提出一种面向机器人故障的异构机器人团队韧性时序逻辑任务规划算法。其次，所提出的算法旨在最大限度地减少故障情况下对多机器人行为的干扰，因为它避免了全局重新分配/重新规划。第三，我们为所提出的方法提供正确性、完备性和最优性保证。第四，我们提供实验以展示该算法的效率。

[1]作者来自美国密苏里州圣路易斯市63130，圣路易斯华盛顿大学电气与系统工程系。k.samarth,ioannisk@wustl.edu， [2]作者来自美国宾夕法尼亚州费城19104，宾夕法尼亚大学电气与系统工程系。pappasg@seas.upenn.edu。本研究获得 ARL 项目 DCIST CRA（资助编号 W911NF-17-2-0181）的支持。

II. 问题定义

A. 机器人与环境建模

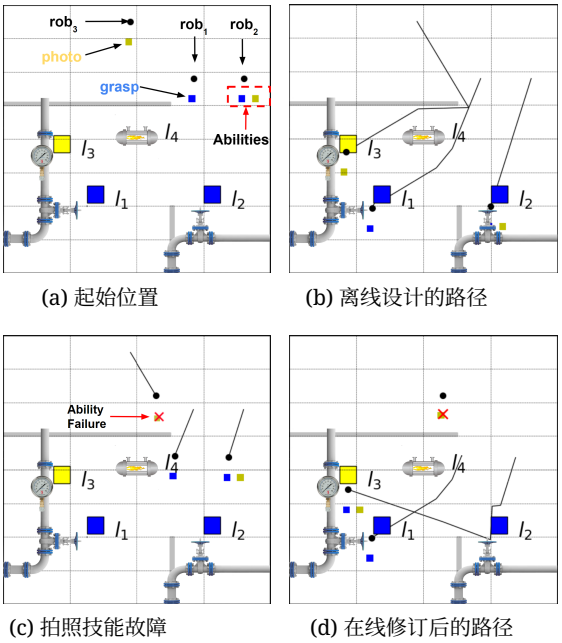
考虑一支由 $N > 0$ 移动机器人组成的团队，其遵循以下动力学：$\mathrm{p}_j(t+1) = \mathrm{f}_j(\mathrm{p}_j(t), \mathrm{u}_j(t))$, 对所有 $j \in \mathcal{R} = \{1, \ldots, N\}$均成立，其中 $\mathrm{p}_j(t) \in \mathbb{R}^n$ 表示机器人 j 在离散时间t的状态（例如位置和姿态），而 $\mathrm{u}_j(t) \in \mathbb{R}^b$ 表示控制输入。下文中，我们将所有机器人的动力学简洁地表示为：$\mathrm{p}(t+1) = \mathrm{f}(\mathrm{p}(t), \mathrm{u}(t))$, 其中 $\mathrm{p}(t) \in \mathbb{R}^{nN}$, $\forall t \geq 0$, 且 $\mathrm{u}(t) \in \mathbb{R}^{bN}$。我们假设在所有时间点 $t \geq 0$均已知机器人状态 $\mathrm{p}(t)$。机器人位于已知环境 $\Omega \subseteq \mathbb{R}^d$, $d \in \{2, 3\}$ 中，其无障碍空间表示为 $\Omega_{\text{free}} \subseteq \Omega$。我们假设 Ω_{free} 中分布着 $M > 0$ 个感兴趣区域/物体，记为 ℓ_e，其位置 $\mathrm{x}_e \in \Omega_{\text{free}}$, $e \in \{1, \ldots, M\}$已知。

B. 异构机器人能力与机器人故障

我们考虑技能各异的机器人。这些机器人总共拥有 $C > 0$ 种能力。每种能力用 $c \in \{1, \ldots, C\}$表示。例如，c可以表示移动能力、操作、灭火或各种感知技能。我们定义集合 $\mathcal{C} = [1, \ldots, c, \ldots, C]$，用于收集所有机器人能力。机器人 j 的技能表示为 $\mathrm{Z}_j(t) = [\zeta_1^j(t) \ldots, \zeta_c^j(t) \ldots, \zeta_C^j(t)]$, 其中，$\zeta_c^j(t)$等于 1 表示机器人 j 在时间 t 具备能力 c，否则为 0。注意，机器人的技能集合随时间变化。如果机器人 j 在时间 t 发生能力 c 故障，则$\zeta_c^j(t-1) = 1$ 且 $\zeta_c^j(t) = 0$。我们通过将 $\mathrm{Z}_j(t)$设为零向量来模拟机器人移除。故障可能发生在未知时刻 t，但我们假设，对于所有机器人 j，在每个时间 $t \geq 0$，向量 $\mathrm{Z}_j(t)$均已知。换言之，我们假设机器人配备了健康监测系统，以便跟踪其能力的启用或停用状态。此外，我们假设采用全对全通信。这确保团队能够获知任何能力故障。为简单起见，我们假设机器人不能同时应用多种技能。根据各机器人的能力，我们将多机器人系统划分为 C 个团队。我们将时间 t 时的机器人团队 $\mathcal{T}_c(t)$定义为由具备 $\zeta_c^j(t) = 1$的机器人组成的集合，即 $\mathcal{T}_c(t) = \{j \in \mathcal{R} \mid \zeta_c^j(t) = 1\}$；一台机器人可以同时属于多个团队。

C. 任务规范

机器人的目标是完成由全局线性时序逻辑（LTL）规范 ϕ 描述的高层、长时域协同任务，该规范要求它们以时序和逻辑的方式，在特定区域/物体 ℓ_e 应用其能力。LTL 包含一组原子命题（即布尔变量），记为 $\mathcal{AP}$，布尔运算符（即合取 $\wedge$和否定 $\neg$），以及两个时序运算符：下一时刻 $\bigcirc$ 和直到 $\mathcal{U}$。基于以下语法，可以构造关于集合 $\mathcal{AP}$ 的 LTL 公式：$\phi ::= \text{true} \mid \pi \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \bigcirc\phi \mid \phi_1 \,\mathcal{U}\, \phi_2$, 其中 $\pi \in \mathcal{AP}$。为简洁起见，我们不再介绍其他布尔运算符和时序运算符的推导，例如总是 $\Box$、最终 $\Diamond$、蕴含 $\Rightarrow$，它们可以是



(a) 起始位置　(b) 离线设计的路径

(c) 拍照技能故障　(d) 在线修订后的路径

图 1. 示例 2.1：每个机器人下方的小方块表示该机器人拥有的能力；操作阀门和拍照的能力分别用蓝色和黄色方块表示。较大的方块表示位置，其颜色表示根据 ϕ.，该位置需要使用哪种能力。

在 [1] 中发现。我们考虑基于以下基于团队的原子谓词构建的 LTL 任务：

$$\pi_{\mathcal{T}_c}(j, c, \ell_e) = \begin{cases} \text{true}, & \text{if } j \in \mathcal{T}_c \text{ applies } c \text{ at } \ell_e \\ \text{false}, & \text{otherwise} \end{cases} \tag{1}$$

当团队中的任意机器人 $\mathcal{T}_c$ 在区域/物体 ℓ_e 处应用技能 c（例如“抓取”）时，式 (1) 中的谓词为真。被分配执行此子任务/谓词的机器人在式 (1) 中记为 j。我们假设已给定机器人 j 到谓词 (1) 的初始分配，且由此得到的任务 ϕ 可行；这是大多数相关工作的常见假设。根据该谓词的定义，如果 $\mathcal{T}_c$ 中没有机器人在 ℓ_e 处应用技能 c，则 $\neg\pi_{\mathcal{T}_c}(j, c, \ell_e)$ 也成立。换句话说，如果谓词前出现否定，那么它涉及整个团队 $\mathcal{T}_c$，而不是某个特定机器人 j；也就是说，式 (1) 中的机器人 j 定义是多余的，因此由符号 $\varnothing$ 取代。我们还假设机器人具备完美技能，即一旦机器人 j 在 ℓ_e 处应用技能 c，式 (1) 就成立。示例 2.1-2.2 给出了一个简单的线性时序逻辑任务示例。给定一个线性时序逻辑公式，可以使用 [5] 等现有方法构造多机器人路径 τ。路径 τ 定义为状态的无限序列，即 $\tau = \tau(0) \ldots, \tau(t) \ldots$。在 τ 中，每个状态 $\tau(t)$ 定义为$\tau(t) = [\mathrm{p}(t), \mathrm{s}(t)]$,，其中 $\mathrm{p}(t)$ 是多机器人系统状态，$\mathrm{s}(t) = [s_1(t) \ldots, s_N(t)]$, $s_j(t) \in \mathcal{C}$。换句话说，$s_j(t)$ 决定了机器人 j 在时间 t应应用哪项技能；如果机器人 j 在时间 t 不需要应用任何技能，那么略微滥用记号，将其记为 $s_j(t) = \varnothing$。

D. 问题陈述：弹性时序逻辑规划

考虑一支承担完成任务的机器人团队ϕ。当机器人执行设计好的规划 τ时，某些机器人技能可能会意外失效；参见例2.1-例 2.2。在这种情况下， τ 可能不再可行，从而损害任务性能。我们的目标是解决以下问题：

问题 *1*：考虑一个任务 ϕ，它定义在谓词（1）之上，这些谓词的初始分配对象为机器人，且已有一个离线生成的规划 τ 满足 ϕ。当机器人技能发生故障时（可能同时发生多个故障），设计 (i) 一种在线任务重新分配方法，根据机器人的当前技能，将与故障技能相关的子任务（即原子谓词）重新分配给机器人；(ii) 修订 τ，使其满足任务重新分配后产生的 LTL 公式。

示例 2.1（场景 I）：考虑一组具有如下形式简单动力学的 $N = 3$机器人：$\mathrm{p}(t+1) = \mathrm{p}(t) + \mathrm{u}(t)$。机器人所具备的能力分别表示为 c_1、 c_2和 c_3，分别对应移动能力、抓取阀门并将其打开的能力，以及使用摄像头拍照的技能。基于技能的团队定义为 $\mathcal{T}_{c_1}(0) = \{1,2,3\}$、 $\mathcal{T}_{c_2}(0) = \{1,2\}$和 $\mathcal{T}_{c_3}(0) = \{2,3\}$。这些机器人负责完成一项管道巡检任务，该任务由以下 LTL 公式建模：$\phi = \Diamond(\pi_1 \wedge \Diamond(\pi_2)) \wedge \Diamond\pi_3 \wedge \Box\neg\pi_4$ 其中$\pi_1 = \pi_{\mathcal{T}_2}(1, c_2, \ell_1)$、$\pi_2 = \pi_{\mathcal{T}_3}(3, c_3, \ell_3)$、$\pi_3 = \pi_{\mathcal{T}_2}(2, c_2, \ell_2)$ (1)，以及 $\pi_4 = \pi_{\mathcal{R}}(\varnothing, c_1, \ell_4)$。具体而言，该任务要求机器人 1 打开管道位置 ℓ_1 处的阀门（由π_1建模），随后由机器人 3 拍摄该处流量计的照片（由 π_2建模）。最终，机器人 2 必须打开另一位置 ℓ_2 处的阀门（由 π_3建模）。在任何时刻，所有机器人都必须避免靠近 ℓ_4 (c_1)，因为该处设有一台锅炉，其高温环境不利于机器人（由 π_4建模）。机器人的初始位置和技能如图 1(a) 所示。一个可行的规划 τ 如图 1(b) 所示。在机器人执行这些规划的过程中，机器人能力可能发生故障。具体而言，图 1(c) 展示了机器人 3 的摄像头在 $t = 3$时刻发生故障的情况，即$\zeta^3_{c_3}(2) = 1$、 $\zeta^3_{c_3}(3) = 0$，以及 $\mathcal{T}_{c_3}(3) = \{2\}$。由于这一故障，原本分配给机器人 3 的 π_2无法得到满足。因此，必须将其重新分配给团队 $\mathcal{T}_{c_3}(3)$中的另一台机器人。所提出的算法将 π_2 分配给机器人 2。重新分配后满足任务要求的路径如图 1(d) 所示。我们将在第 IV 节重新讨论这一示例。

例 2.2（场景 II）：考虑一个由 $N = 5$机器人组成的团队，其技能包括 c_1、 c_2、 c_3、 c_4 和 c_5。技能 $c_1 - c_3$的定义如上。其余技能分别指拾取物体和温度传感器。基于技能的团队为 $\mathcal{T}_{c_1}(0) = \{1,2,3,4,5\}$、 $\mathcal{T}_{c_2}(0) = \{1,4\}$、 $\mathcal{T}_{c_3}(0) = \{1,2,4,5\}$、 $\mathcal{T}_{c_4}(0) = \{3\}$ 和 $\mathcal{T}_{c_5}(0) = \{4,5\}$。考虑以下 LTL 任务：$\phi = \Diamond(\pi_1 \wedge \Diamond((\pi_2 || \pi_3) \wedge \pi_5 \wedge \Diamond(\pi_1))) \wedge \Box\neg\pi_4$，其中 $\pi_1 = \pi_{\mathcal{T}_2}(1, c_2, \ell_1)$、 $\pi_2 = \pi_{\mathcal{T}_3}(2, c_3, \ell_2)$、 $\pi_3 = \pi_{\mathcal{T}_4}(3, c_4, \ell_3)$、$\pi_4 = \pi_{\mathcal{T}_2}(\varnothing, c_1, \ell_2)$ 和 $\pi_5 = \pi_{\mathcal{T}_5}(5, c_5, \ell_5)$。如果机器人 2 的技能 c_3 和机器人 3 的技能 c_4 在 $t = 1$时失效，那么机器人 $i \in \mathcal{T}_{c_3}(1) = \{1,4,5\}$ 需要接管 π_2。请注意，尽管机器人 4 未被分配任何谓词，但不能为其分配 π_2，因为由机器人 4 满足 π_2 将导致满足 π_4，从而违反 ϕ。机器人 1 也有<escape>相同的约束。因此，只有 5 能够接替 π_2。然而，需要注意的是，机器人 5 需要完成 π_5，同时 π_2 也需要得到满足。还应注意， $\mathcal{T}_{c_4}(1) = \varnothing$，因此 π_3 无法得到满足。我们提出的算法旨在处理此类具有挑战性的场景；详见第 III 节。该示例将在第 III-IV 节中再次讨论。</escape>

备注 *2.3*（子任务的独立性）：假设机器人 j 的某些技能发生故障。我们假设，与 j 及其故障技能相关且出现在 ϕ 中的所有原子谓词（无论位置 ℓ_e如何）都可以彼此独立地重新分配，也就是说，它们不需要分配给同一机器人 $i \neq j$。例如，在例 2.2中注意到，原子命题 π_1 在 ϕ 中出现了两次。如果 c_2 对机器人 1 失效，那么“最左侧”和“最右侧”的命题 π_1 无需由同一机器人满足。

III. 针对机器人技能故障的弹性时序逻辑规划

本节介绍一种用于解决问题1的算法。我们的解决方案由三部分组成。第一部分中，我们使用近期提出的基于采样的规划器 τ 离线生成一条满足 ϕ 的路径 [5]；详见第III-A-III-B节。第二部分中，我们设计一种在线局部任务重新分配算法，将子任务重新分配给机器人。该算法之所以称为局部算法，是因为它旨在尽量减少因技能失效而必须进行的子任务重新分配总次数；详见第 III-C-III-E节。基于修订后的 LTL 公式，我们提出一种在线重新规划方法，在局部范围内修订当前机器人路径，以设计新的可行路径。在第III-G节中，我们分析所提算法的正确性、完备性和最优性，以及其局限性。

A. 从 *LTL* 任务到自动机

给定一个 LTL 任务 ϕ，我们会在离线状态下将其转换为一个非确定性 Büchi 自动机（NBA） [1]。

定义 *3.1*（*NBA*）：定义在 $\Sigma = 2^{\mathcal{AP}}$ 上的非确定性 Büchi 自动机（NBA） B 是一个元组$B = (\mathcal{Q}_B,\ \mathcal{Q}_B^0,\ \Sigma,\ \delta_B,\ \mathcal{Q}_B^F)$，其中 $\mathcal{Q}_B$是状态集合，$\mathcal{Q}_B^0 \subseteq \mathcal{Q}_B$是初始状态集合， Σ是字母表，δ_B： $\mathcal{Q}_B \times \Sigma \to 2^{\mathcal{Q}_B}$ 是非确定性转移关系，而 $\mathcal{Q}_B^F \in \mathcal{Q}_B$ 是接受/终止状态集合。

接下来，我们讨论用于寻找满足 ϕ 的规划 τ 的 NBA 接受条件。我们定义标记函数 L： $\mathbb{R}^N \times \mathcal{C}^N \to 2^{\mathcal{AP}}$，用于确定给定多机器人状态 $\mathrm{p}(t)$ 和已应用技能 $\mathrm{s}(t)$ 时，哪些原子命题为真。在无限字 $w = \sigma_0\sigma_1\sigma_2\cdots \in \Sigma^\omega$ 上， B 的一条无限运行 $\rho_B = q_B^1 \ldots q_B^k \ldots$，其中 $\sigma_k \in \Sigma, \forall k \in \mathbb{N}$，是 NBA 状态 $q_B^k, \forall k \in \mathbb{N}$ 构成的无限序列，满足 $q_B^{k+1} \in \delta_B(q_B^k, \sigma_k)$ 且 $q_B^0 \in \mathcal{Q}_B^0$。若 $\mathrm{Inf}(\rho_B) \bigcap \mathcal{Q}_B^F \neq \emptyset$，则称无限运行 ρ_B为接受运行，其中 $\mathrm{Inf}(\rho_B)$ 表示在 ρ_B 中无限次出现的状态集合。规划 $\tau = [\mathrm{p}(0), \mathrm{s}(0)], [\mathrm{p}(1), \mathrm{s}(1)], \ldots, [\mathrm{p}(t), \mathrm{s}(t)], \ldots$ 可行，当且仅当字 $w = \sigma_0\sigma_1 \ldots \sigma_t \ldots$（其中 $\sigma_t = L([\mathrm{p}(t), \mathrm{s}(t)])$ ）至少产生一条接受运行 ρ_B。由于我们假设机器人每次不能应用多于一项技能，

因此，接下来我们假设 NBA 已按照 [5] 进行剪枝，移除了违反这一假设的转移。

B. 基于采样的规划器

给定一个 LTL 公式 ϕ，我们使用 [5] 中开发的运动规划器设计可行计划 τ，因为它具有无需抽象和可扩展性方面的优势；也可以使用其他任何运动规划器。具体而言，[5] 提出了一种基于采样的规划器，它通过增量式构建树来探索机器人运动空间和自动机状态空间。树的节点定义为 $\mathrm{q}(t) = [\mathrm{p}(t),\ \mathrm{s}(t)\ q_B(t)]$。树的根节点 $\mathrm{q}(0)$ 根据机器人初始状态 $\mathrm{p}(0)$、一个零向量 $\mathrm{s}(0)$ 以及一个初始非确定性 Büchi自动机状态 $q_B(0) \in \mathcal{Q}_B^0$定义。在算法的每次迭代中，都会对一个新状态 $\mathrm{q}(t)$进行采样；若其可行（即不会导致违反 ϕ），则将其添加到树中。这种基于采样的方法能够生成规划，即以前缀-后缀形式表示的状态序列 $\mathrm{q}(t)$。首先执行前缀，随后无限重复执行后缀。我们将该规划记为 $\tau_H = \tau_H^{\text{pre}}[\tau_H^{\text{suf}}]^\omega$，其中 ω表示无限重复。前缀部分 τ_H^{pre} 定义为 $\tau_H^{\text{pre}} = \mathrm{q}(0), \mathrm{q}(1) \ldots, \mathrm{q}(T)$，其中时域为 $T \geq 0$，且 $q_B(H) \in \mathcal{Q}_B^F$；后缀部分 τ_H^{suf} 定义为 $\tau_H^{\text{suf}} = \mathrm{q}(T+1), \mathrm{q}(T+2) \ldots, \mathrm{q}(T+K)$，其中 $K \geq 0$，且 $\mathrm{q}(T+1) = \mathrm{q}(T+K) = \mathrm{q}(T)$。为设计前缀部分，我们定义一个目标区域 $\mathcal{R}_{\text{pre}}$，其中收集所有非确定性 Büchi 自动机状态属于 $\mathcal{Q}_B^F$ 的可能节点 q，即 $\mathcal{R}_{\text{pre}} = \{\mathrm{q} = (\mathrm{p}, \mathrm{s}\ q_B) \mid q_B \in \mathcal{Q}_B^F\}$。一旦构造出一条从根节点出发并以 $\mathcal{R}_{\text{pre}}$ 为终点的树枝（即树节点序列），就可以导出一个前缀部分 τ_H^{pre}。类似地，给定一个前缀部分 τ_H^{pre}，通过构建一棵以 $\tau_H^{\text{pre}} = \mathrm{q}(T+1)$ 的最后一个状态为根的新树，可以找到其对应的后缀路径。随后定义目标区域，以便导出根节点周围的循环路径（即后缀部分）。因此，在这种情况下，目标区域定义为 $\mathcal{R}_{\text{suf}} = \{\mathrm{q} = (\mathrm{p}, \mathrm{s}\ q_B) \mid q_B(T+1) \in \delta_B(L([\mathrm{p}, \mathrm{s}])\ q_B)\}$。从 τ_H 中消去非确定性 Büchi 自动机状态，即可得到所需的规划 $\tau = \tau^{\text{pre}}[\tau^{\text{suf}}]^\omega$；更多细节见 [5]。根据 τ，我们用 $\tau(t) = [\mathrm{p}(t),\ \mathrm{s}(t)]$ 表示机器人在时间 t的状态和动作。同样的记法也适用于 τ_H。

C. 设置任务重新分配流程

假设已按照第 III-B 节生成了一个可行规划 τ。考虑这样一种情况：机器人执行规划 τ时，部分机器人能力在一个先验未知的时刻 t 失效，从而使某些机器人 $j \in \mathcal{R}$产生新的向量 $\mathrm{Z}j(t)$。令 $\mathcal{AP}_F \subseteq \mathcal{AP}$ 为集合，其中收集了在当前谓词分配给机器人的情况下，由于技能失效而永远无法为真的 ϕ 中所有原子谓词 $\pi_{\mathcal{T}_c}(j, c, \ell_e)$。[1] 稍微宽泛地使用这一术语，我们将 $\pi_{\mathcal{T}_c}(j, c, \ell_e) \in \mathcal{AP}_F$ 称为“失效”原子谓词。对于所有失效谓词，将按顺序（顺序不限）重复以下流程（算法 1，第 1 行）。

我们的目标是将 $\pi_{\mathcal{T}_c}(j, c, \ell_e) \in \mathcal{AP}_F$重新分配给一台新的机器人 $i \in \mathcal{T}_c$；同样，稍微宽泛地使用这一术语，我们也将这一过程称为“修复”失效谓词。通俗地说，这里的关键挑战在于，可能所有机器人$i \in \mathcal{T}_c$ 都在忙于其他子任务；参见例 2.2。在这种情况下，修复失效谓词可能会引发一系列任务重新分配。原因是，修复失效谓词需要另一台机器人 $i \in \mathcal{T}_c$ 接手该任务，同时，可能还需要另一台机器人接手原本分配给机器人 i 的子任务，如此直到不再有未分配的子任务。我们的目标是通过尽可能少的任务重新分配来修复失效谓词，从而尽量减少对任务规划的干扰。

假设故障发生在时间 t。设 $q_B^{\text{cur}} = q_B(t)$ 为机器人执行完其规划 τ 的前 t 步后到达的非确定性 Büchi 自动机状态。给定一个失效谓词$\pi_{\mathcal{T}_c}(j, c, \ell_e)$，我们计算从 q_B^{cur} 出发，经多跳路径可到达的所有非确定性 Büchi 自动机状态。此步骤可通过将非确定性 Büchi 自动机视为有向图，并检查从 q_B^{cur} 可到达哪些状态 $q'_B \in \mathcal{Q}_B$ 来实现。我们将这些状态（包括 q_B^{cur}）收集到集合 $\mathcal{Q}_B^{\text{cur}} \subseteq \mathcal{Q}_B$ 中。然后，在状态 q'_B, $q''_B \in \hat{}\mathcal{Q}_B^{\text{cur}}$ 之间的所有非确定性 Büchi 自动机转移中，我们将满足以下条件的转移收集到集合 $\mathcal{E}$ 中：对应布尔公式 $b_{q'_B, q''_B}$ 中出现了 $\pi_{\mathcal{T}_c}(j, c, \ell_e)$（算法 1 第 2 行）。稍微滥用一下记号，令 $e = (q'_B, q''_B)$ 表示 $\mathcal{E}$ 中的一条边（即从 q'_B 到 q''_B 的非确定性 Büchi 自动机转移）。给定边 $e \in \mathcal{E}$，我们定义布尔公式 $b_{q'_B, q''_B}$，该公式定义在 $\mathcal{AP}$ 上，并满足：对于某个$\sigma \in \Sigma$，若 $q''_B \in \delta_B(q'_B, \sigma)$，则$\sigma \models b_{q'_B, q''_B}$，反之亦然。此类布尔公式可使用 [25] 等现有工具自动构造。

D. 任务重新分配约束

我们方法的核心思想是检查所有边$e \in \mathcal{E}$，并重新分配各自公式中出现的谓词$b_{q'_B,\ q''_B}$，从而确保不存在未分配的谓词，且最终得到的 LTL 公式仍然是可行的（即机器人仍然能够生成产生接受型 NBA 运行的路径）。在任务重新分配后保持 LTL 公式可行性的一个必要条件是，$b_{q'_B,\ q''_B}$ 在任务重新分配后应当是可行的（即它可以变为“真”），$\forall e = (q'_B,\ q''_B) \in \mathcal{E}$；另请参见注释 3.8。换言之，应当存在一个由机器人生成的符号 $\sigma = L([\mathrm{p},\ \mathrm{s}]) \in \Sigma$，该符号满足任务重新分配后产生的修订后的公式$b_{q'_B,\ q''_B}$。

接下来，我们将把这一条件转化为约束，以便在边 $e = (q'_B,\ q''_B)$ 中修复失效谓词时，这些约束能够得到遵守（其中 q'_B 可能与 q''_B 重合）。为此，我们将布尔公式 $b_{q'_B,\ q''_B}$ 重新写成析取范式（DNF），即 $b_{q'_B,\ q''_B} = \bigvee_{d=1}^{D} b^d_{q'_B,\ q''_B}$，其中 $D > 0$ 为某个值。[2]对于每个布尔公式 $b^d_{q'_B,\ q''_B}$，我们定义集合 $\mathcal{R}^d_{q'_B,\ q''_B} \subseteq \mathcal{R}$，该集合收集 $b^d_{q'_B,\ q''_B}$ 中出现的所有机器人；另请参见例 3.4。给定 $b^d_{q'_B,\ q''_B}$，我们还定义集合 $\mathcal{AP}_i$，该集合收集 $b^d_{q'_B,\ q''_B}$ 中出现的所有形如 (1) 的谓词，并假设与其相关联的那些谓词.....

[1]不包含形式为 $\neg\pi_{\mathcal{T}_c}(\varnothing, c, \ell_e)$ 的否定谓词，因为此类谓词指的是整个团队 $\mathcal{T}_c$，而非特定机器人。因此，重新分配它们没有意义。

[2]失效谓词可能出现在多个子公式中$b^d_{q'_B,\ q''_B}$ 但对于每个子公式，它都会被分别／独立修复；参见备注 2.3。

技能 c 满足 $\zeta_c^i(t) = 1$ 均被分配给机器人 i。给定 $\mathcal{AP}_i$，我们可以构造字母表 $\Sigma_i = 2^{\mathcal{AP}_i}$。例如，如果 $b^d_{q'_B, q''_B} = \pi_{\mathcal{T}_c}(j,c,\ell_e) \wedge \pi_{\mathcal{T}_m}(i,m,\ell_f)$，那么 $\mathcal{AP}_i = \{\pi_{\mathcal{T}_c}(i,c,\ell_e)\ \pi_{\mathcal{T}_m}(i,m,\ell_f)\}$ 当 $i \in \mathcal{T}_m \cap \mathcal{T}_c$ 且 $\Sigma_i = \{\pi_{\mathcal{T}_c}(i,c,\ell_e)\ \pi_{\mathcal{T}_m}(i,m,\ell_f)\ \pi_{\mathcal{T}_c}(i,c,\ell_e)\pi_{\mathcal{T}_m}(i,m,\ell_f)\ \epsilon\}$，其中 ϵ 表示空符号。接下来，我们定义以下函数来建模任务重新分配约束。

定义 *3.2*（函数 $V^d_{q'_B, q''_B}$）：集合值函数 $V^d_{q'_B, q''_B}: \mathcal{R} \to \Sigma_i$，输入机器人索引 $i \in \mathcal{R}$ 后，返回一个集合，其中包含所有符号 $\sigma_i \in \Sigma_i$；如果机器人 $i \in \mathcal{R}$ 生成该符号，那么无论其他谓词的取值如何，$b^d_{q'_B, q''_B}$ 都将为“假”。对于所有机器人 $i \in \mathcal{R} \setminus \mathcal{R}^d_{q'_B, q''_B}$，我们定义 $V^d_{q'_B, q''_B}(i) = \emptyset$。

集合 $V^d_{q'_B, q''_B}(i)$ 描述了对机器人 i 可接管哪些任务/谓词的约束。具体来说，在重新分配边 $e = (q'_B, q''_B)$ 中出现的谓词时（其中 q'_B 可能与 q''_B 相同），机器人 $i \in \mathcal{R}^d_{q'_B, q''_B}$ 无法接管出现在 $V^d_{q'_B, q''_B}(i)$ 中的谓词，因为这会使 $b^d_{q'_B, q''_B}$ 为假。下一时刻，我们定义函数 $g^d_{q'_B, q''_B}$，稍后将用它来确定哪些机器人当前正忙于其他子任务。

定义 *3.3*（函数 $g^d_{q'_B, q''_B}$）：函数 $g^d_{q'_B, q''_B}: \mathcal{R} \to \mathcal{AP}$，输入一个机器人索引 $i \in \mathcal{R}$，返回一个集合，其中包含在 i 中分配给机器人 $b^d_{q'_B, q''_B}$ 的原子谓词，但不包括被否定的谓词和失效谓词。[3] 我们定义 $g(i) = \varnothing$，对所有机器人 $i \notin \mathcal{R}^d_{q'_B, q''_B}$ 以及所有出现在否定谓词或失效谓词中的机器人 $i \in \mathcal{R}^d_{q'_B, q''_B}$ 均成立。

例 3.4（函数 $g^d_{q'_B,q''_B}$, $V^d_{q'_B,q''_B}$ 和集合 $\mathcal{R}^d_{q'_B,q''_B}$）：考虑例 2.2 中给出的线性时序逻辑公式。我们聚焦于一个具有以下内容的非确定性 Büchi 自动机转移：$b^d_{q'_B,q''_B} = \pi_2 \wedge \pi_5 \wedge \neg\pi_4$，其中回顾一下 $\pi_2 = \pi_{\mathcal{T}_3}(2,c_3,\ell_2)$、$\pi_5 = \pi_{\mathcal{T}_5}(5,c_5,\ell_5)$，以及 $\pi_4 = \pi_{\mathcal{T}_2}(\varnothing,c_1,\ell_2)$。于是，$\mathcal{R}^d_{q'_B,q''_B} = \{2,5\} \cup \mathcal{T}_2$。此外，对于所有机器人 $i \in \mathcal{R}^d_{q'_B, q''_B} \setminus \{2,5\}$，有 $g^d_{q'_B,q''_B}(i) = \varnothing$，且 $g(2) = \pi_2$，$g(5) = \pi_5$。我们还有 $V^d_{q'_B, q''_B}(i) = \emptyset$，对于所有 $i \notin \mathcal{T}_2$ 和 $\{\pi_{\mathcal{T}_2} i\ c_1\ \ell_2\ \pi_{\mathcal{T}_3} i\ c_3\ \ell_2, \pi_{\mathcal{T}_2} i\ c_1\ \ell_2\ \pi_{\mathcal{T}_3} i\ c_3\ \ell_2\}$ 对所有 $i \in \mathcal{T}_2 \cap \mathcal{T}_3$ 成立。注意，$\pi_{\mathcal{T}_2}(i,c_1,\ell_2)$ 和 $\pi_{\mathcal{T}_2}(i,c_1,\ell_2)\pi_{\mathcal{T}_3}(i,c_3,\ell_2)$ 被包含在内，是因为 $\neg\pi_{\mathcal{T}_2}(\varnothing,c_1,\ell_2)$ 位于 $b^d_{q'_B,q''_B}$ 中。此外，$\pi_{\mathcal{T}_3}(i,c_3,\ell_2)$ 被包含在内，因为如果机器人 1 或 4 满足它，那么它们将靠近位置 ℓ_2；因此，$\pi_{\mathcal{T}_2}(i,c_1,\ell_2)$ 也将得到满足，从而违反 $b^d_{q'_B,q''_B}$。

E. 局部任务重新分配算法

本节介绍所提出的任务重新分配算法。构造集合 $\mathcal{E}$ 后，我们对每条边 $e \in \mathcal{E}$ 并行重复以下步骤（第3行，算法1）。给定 e，我们将其对应的布尔公式表示为析取范式 $b_{q'_B, q''_B} = \bigvee_{d=1}^{D} b^d_{q'_B, q''_B}$（无论 $q'_B = q''_B$ 与否）（第4行，算法1）。然后，我们对每个子公式 $b^d_{q'_B, q''_B}$ 并行重复以下步骤（第5行，算法1）。

[3]分配给机器人的谓词至多有一个 $i \in \mathcal{R}^d_{q'_B, q''_B}$ 因为所有要求机器人同时满足多个谓词的 NBA 转移都已被剪枝；参见 III-A。

算法 1： 局部任务重新分配

输入： (i) 非确定性 Büchi 自动机 B，(ii) 当前非确定性 Büchi 自动机状态 q''_B；(iii) 失效谓词集合 $\mathcal{AP}_F$

输出： 修订后的 NBA

1 **对于**每个 $\pi \in \mathcal{AP}_F$ **执行**
2 定义有序边集 $\mathcal{E}$;
3 **对于**每个 $e = (q'_B, q''_B) \in \mathcal{E}$ **执行**
4 重写：$b_{q'_B,q''_B} = \bigvee_{d=1}^{D} b^d_{q'_B,q''_B}$;
5 **对于** $d = 1, \ldots, D$ **执行**
6 定义 $\mathcal{G}$ 和函数 $V^d_{q'_B,q''_B}$ $g^d_{q'_B,q''_B}$;
7 应用算法2 计算以下序列：重新分配 $p = p(0), \ldots, p(P)$;
8 **如果** $\exists p$ **则**
9 根据 p 重新分配原子谓词;
10 **else**
11 将 $\pi =$ 赋为 False;
12 修订 $b^d_{q'_B,q''_B}$;

首先，我们定义以下有向图，以捕获 $b^d_{q'_B,q''_B}$ 中所有可能的重新分配（包括违反第 III-D 节所讨论约束的重新分配）。然后，我们将在该图上搜索，以找到 $b^d_{q'_B,q''_B}$ 中出现的子任务/谓词的重新分配，使得不存在未分配的谓词，且所有分配均满足 $V^d_{q'_B,q'_B}$ 所捕获的约束。我们用 $\mathcal{G} = \{\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G}, w_\mathcal{G}\}$ 表示该图，其中 $\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G}, w_\mathcal{G}$ 分别表示节点集、边集和成本函数 $w_\mathcal{G}: \mathcal{E}_\mathcal{G} \to \mathbb{R}$。节点集定义为 $\mathcal{V}_\mathcal{G} = \mathcal{R}$。若 $a' \in \mathcal{T}_c$，则存在一条从节点 a 到 $a' \neq a$ 的边，其中 c 是满足谓词 $g^d_{q'_B,q''_B}(a)$ 所需的技能。若 $g^d_{q'_B,q''_B}(a) = \varnothing$，则从 a 出发不存在出边。一条边的物理含义是，机器人 a' 可以接管在 $b^d_{q'_B,q''_B}$ 中分配给 a 的谓词（如果有）。成本函数 $w_\mathcal{G}$ 的定义使得每条边的成本均为 1。我们强调，我们不会显式构建该图；相反，任务重新分配算法只需知道从任意节点 a 出发一步可达的节点 $a' \in \mathcal{T}_c$（算法 1 第 6 行）。

然后，我们在 $\mathcal{G}$ 上应用受约束的广度优先搜索（BFS）算法，重新分配子任务（即出现在 $b^d_{q'_B,q''_B}$ 中的谓词）给机器人（第 7 行，算法 1）。我们的目标是在 $\mathcal{G}$ 中寻找一条路径，从与失效谓词相关联的机器人（记为 a_{root}）出发，到达任意满足 $g^d_{q'_B, q''_B}(a') = \varnothing$ 的节点 a'（即 a' 在 $b^d_{q'_B,q''_B}$ 中没有被分配任何任务），同时遵守第 III-D 节中讨论的约束。我们定义集合 $\mathcal{A} = \{a \in \mathcal{R} \mid g^d_{q'_B,q''_B}($ a $p =$ $p\ 0\ \ p\ 1\ \ \ldots\ p\ P$ $\mathcal{G}$ $p\ 0\ =) = \varnothing\}$。令 $(\ )\ (\ )$, $(\ )$ 表示在 上的这类路径，其中 $(\ )\ a_{\text{root}}$, $p(P) \in \mathcal{A}$ 且 $p(k) \notin \mathcal{A}$，对所有 $k \in \{2, \ldots, P-1\}$ 均成立。这条路径将决定修复失效谓词所需的任务重新分配。具体来说，机器人 $p(k+1)$ 接管机器人 $p(k)$ 在 $b^d_{q'_B,q''_B}$ 中的子任务（即原子谓词 $g^d_{q'_B,q''_B}(p(k))$）。请注意，这意味着机器人 $p(k+1)$ 放弃其当前的子任务（这些任务将由机器人 $p(k+2)$ 接手）。此外，由于此

算法 2： 广度优先搜索

输入： (i) 失效谓词 $\pi_{\mathcal{T}_c}(j,\ c,\ \ell_e)$，(ii) $V^d_{q'_B,q''_B}$，(iii) $g^d_{q'_B,q''_B}$，(iv) 团队 $\mathcal{T}_c(t),\ \forall c \in \mathcal{C}$

输出： 路径 p

1 $a_{\text{root}} = j$;
2 $\mathcal{Q} = [a_{\text{root}}]$;
3 标记$_{\text{root}}$ =真;
4 **当** $\sim$*empty*($\mathcal{Q}$) **执行**
5 $a \leftarrow$ 弹出($\mathcal{Q}$);
6 **如果** $a \in \mathcal{A}$ & $\sim$标记$_{root}$ **则**
7 使用父函数返回路径 p;
8 标记$_{\text{root}}$=False;
9 **对于** a' 与 a相邻，且位于 $\mathcal{G}$ **则**
10 **如果** $g^d_{q'_B,q''_B}(a) \notin V^d_{q'_B,q''_B}(a')$& a' 未探索 **then**
11 将 a' 标记为已探索;
12 Parent(a') $= a$;
13 追加 a' 至 $\mathcal{Q}$;
14 返回 ($\emptyset$)

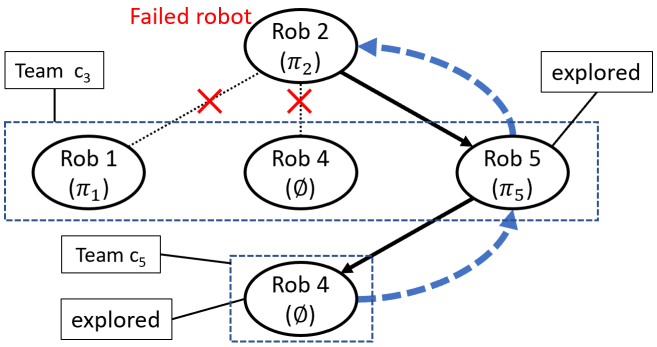


图 2。考虑示例2.2 中机器人 2 的技能 c_3失效的情况，即失效谓词为 π_2。我们展示了为修复由 $b^d_{q'_B,q''_B} = \pi_1 \wedge \pi_2 \wedge \pi_5 \wedge \neg\pi_4$启用的非确定性 Büchi 自动机转移 π_2 而构建的广度优先搜索树（算法2）。集合 $\mathcal{A}$ $\mathcal{A} = \{4\}$ 2 1, 4, 定义为，树 5 2 1 4 2 的根节点是机器人。机器人 与机器人 在 中相邻。机器人 和 不与机器人 相连，因为它们不满足 $g^d_{q'_B,q''_B}(2) \notin V^d_{q'_B,q''_B}(a') = \{\pi_2, \pi_4, \pi_2\pi_4\}$，对于 $a' \in \{1,4\}$。机器人 5 与机器人 2 相连，随后机器人 4 与机器人 5 相连。蓝色虚线箭头表示沿计算出的路径 p进行的重新分配过程，即机器人 4 将接替 π_5，机器人 5 将接替失效谓词。

路径应遵守第 III-D 节中讨论的约束，因此对于所有 $k \in \{2, \dots, P-1\}$，还必须满足 $g^d_{q'_B,\ q''_B}(p(k)) \notin V^d_{q'_B,\ q''_B}(p(k+1))$。

我们应用广度优先搜索算法来查找 p在算法 2 中总结的最短路径；参见图 2。请注意，这与标准广度优先搜索算法存在两个关键区别。第一个区别涉及何时将节点添加到 $\mathcal{Q}$ 广度优先搜索算法的队列数据结构中。具体而言，当一个节点a 从 $\mathcal{Q}$中弹出/移除后，如果相邻节点 a'满足以下条件，则将其添加到 $\mathcal{Q}$ 中：（1）尚未被探索（与标准广度优先搜索相同）；并且（2）$g^d_{q'_B、\ q''_B}(a) \notin V^d_{q'_B、\ q''_B}(a')$（算法 2 第 10 行）；第一个约束避免了单个机器人必须替代两个机器人的情况，因为这可能导致布尔公式$b^d_{q'_B、\ q''_B}$ 在某个机器人同时应用多个技能时得到满足，而这是不可行的（参见第 II 节）。第二个约束确保满足第 III-D 节中讨论的约束。与标准广度优先搜索的第二个区别在于，根节点最初不会被标记为“已探索”。这使得具有失效技能的机器人（根节点）能够利用其剩余的活动技能（如果有）接管其他子任务。最后，我们注意到，当找到从 a_{root} 到 $\mathcal{A}$ 中任意节点的第一条可行路径时，图搜索过程即终止（算法 2 第 6–7 行）。原因是，这条路径对应于修复$b^d_{q'_B、\ q''_B}$中失效谓词所需进行的最少重新分配次数；参见第 III-G 节。

修复所有失效谓词后，相应的公式$b_{q'_B,\ q''_B}$ 也会随之修订，生成一个新的 NBA（第9行，算法 1）。如果某个失效谓词无法修复（即算法2 无法找到可行路径），则在所有相关公式中将该失效谓词替换为逻辑值“false”$b_{q'_B,\ q''_B}$因为没有机器人能够接替它（第11行，算法 1）；请注意，这并不一定意味着该 LTL 公式不可行。这个修订后的 NBA 是在线规划器的输入，在线规划器据此设计新路径；参见第III-F节。

F. 在线重新规划

假设多机器人系统处于状态 $\tau_H(t) = [\mathbf{p}(t), \mathbf{s}(t)\ q_B(t)]$ 时，发生了能力故障，且算法 1 已将任务重新分配给机器人。修订机器人路径的一种直接方法是应用第 III-B 节讨论的基于采样的规划器，从头开始构建一个新的前缀-后缀计划。然而，鉴于算法 *1* 所执行的局部任务重新分配，对所有 机器人从头开始重新规划可能并无必要，而且对于大型机器人团队而言，这种方法可能不切实际。受 [11], 的启发，我们提出了一种替代性的重新规划方法，旨在对多机器人计划进行局部修订；另请参见图 3。

首先，我们定义路径 $\hat{}\tau_H = \tau\text{pre}_H\ \tau_H^{\text{suf}} = [\mathbf{q}(1) \dots, \mathbf{q}(T)], [\mathbf{q}(T+1) \dots, \mathbf{q}(T+K)]$，该路径将当前规划的前缀和后缀部分拼接起来 τ_H（不重复 τ_H^{suf}）；另请参见第 III-B 节。其次，我们计算满足 $\hat{}\tau_H(k) = \tau_H(t)$ 的状态 $\hat{}\tau_H(k)$。[4]第三，我们定义有序集合 $\mathcal{T}_{\text{pre}}$，其中收集了 $\hat{}\tau_H$ 中所有满足以下要求的状态 $\hat{}\tau_H(k') = [\mathbf{p}(k'), \mathbf{s}(k')\ q_B(k')]$：i) $k' \geq k$；ii) $(q_B(k') \neq q_B(k'+1)$ 且 $(q_B(k')\ q_B(k'+1)) \in \mathcal{E})$ 或 $(q_B(k') \neq q_B(k'-1)$ 且 $(q_B(k'-1)\ q_B(k')) \in \mathcal{E})$；以及 iii) $k' \leq T$。我们用 $\mathcal{T}_{\text{pre}}(e)$ 表示 $\mathcal{T}_{\text{pre}}$ 中的第 e 个条目。我们类似地定义 $\mathcal{T}_{\text{suf}}$，其中第三个要求替换为 $T+1 \leq k' \leq T+K$。这些集合中的条目按索引 k' 的递增顺序排列（即，$\hat{}\tau_H(k)$ 总是最先出现）。

假设机器人在执行 τ_H中的前缀部分时发生故障。为了在局部修订前缀部分，我们使用第III-B节讨论的基于采样的规划器构建一棵以 $\mathcal{T}_{\text{pre}}(r)$为根节点的树，其中 $r = 1$，目标区域定义为 $\mathcal{R}^{\text{rev}}_{\text{pre}} = \{\mathbf{q}(T)\} \cup \{\mathcal{T}_{\text{pre}}(s)\}_{s=r+1}^{|\mathcal{T}_{\text{pre}}|} \cup \mathcal{R}_{\text{pre}}$，也就是说，目标是到达任意一个后续状态

[4]可能出现这样的情况：$t > T+K$ 因为状态 $\tau_H(t)$ 可能属于后缀部分，而 τ_H 包含后缀部分的无限重复。在$\hat{\tau}_H(k)$ 中，k 指向$\hat{\tau}_H$ 中的第 k 个条目，其中 $\hat{}\tau_H(k) = \tau_H(t)$。

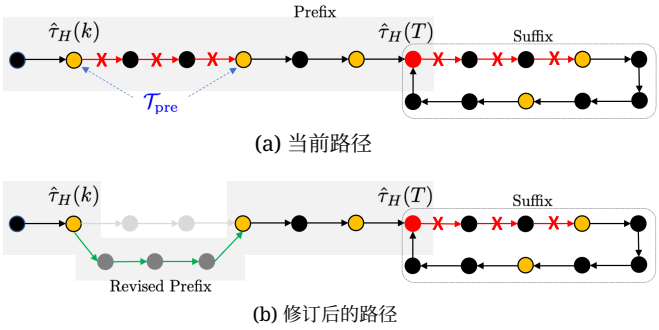


图 3. 前缀路径在线修订示例。圆盘表示 $\hat{}$
τ_H τ_H k' q_B k' $\neq$ 中的状态。q_B $k'-1$
黄色状态 $\hat{}$ ( ) 表示满足 ( ) ( ) 的状态。红色状态对应一个终止状态，围绕该状态设计了一个后缀部分。连接 NBA 状态 q_B 和 q'_B 的$\mathcal{T}_H$ $\hat{}$ 部分，其中 $e=(q_B,q'_B)\in\mathcal{E}$（见算法 1），用红色标记，并以红色“X”表示需要修订。在此示例中，以 $\mathcal{T}_{\text{pre}}$(1) $=\hat{\tau}_H(k)$ 为根构建树后，找到了路径。计算得到的路径将 $\hat{\tau}_H(k)$ 连接到 $\mathcal{T}_{\text{pre}}(2)$。修订后的前缀部分通过用先前计算的路径替换 $\hat{\tau}_H(k)$ 中红色标记的边来构造。

在 $\mathcal{T}_{\text{pre}}$ 中沿当前路径 $\hat{}\tau_H$前进，或到达与当前路径 $\hat{}\tau_H$ 相关联的终止状态 q(T)，或到达第 III-B 节中定义的集合 $\mathcal{R}_{\text{pre}}$ 中的任一终止状态。一旦基于采样的规划器返回首条可行路径，我们便终止该规划器。如果该路径将根节点连接到 $\mathcal{R}_{\text{pre}}$ 或 $\hat{}\tau_H(T)$，则此路径对应于新的／修订后的前缀部分；其他所有树的构建均终止。否则，如果它找到一条通往状态 $\mathcal{T}_{\text{pre}}(s)$、 $s>r$的路径，我们便像之前一样构建一棵新树，以 $\mathcal{T}_{\text{pre}}(r)$ 为根，其中 r 重新定义为 $r=s$。随后重复上述过程，直到找到一条通往 $\mathcal{R}_{\text{pre}}$或 $\hat{}\tau_H(T)$ 的路径。通过用修订后的部分替换 τ_H^{pre} 中受影响的部分来构造修订后的前缀路径，如 $\mathcal{T}_{\text{pre}}$ 所示；另见图 3。如果找不到前缀路径，则触发全局重新规划，并从头设计新的前缀-后缀规划，如第 III-B 节所述；唯一的区别是，前缀树的根节点基于当前机器人位置 p(t) 和当前非确定性 Büchi 自动机状态 $q_B(t)$。前缀部分设计完成后，接着构造后缀部分。如果修订后的前缀部分终止于状态 $\hat{}\tau_H(T)$，则按照与上述类似的逻辑构造后缀部分。否则，如果它终止于状态 $\mathcal{R}_{\text{pre}}\setminus\hat{}\tau_H(T)$，则需要从头构造后缀部分，即构建一棵以该前缀路径终止状态为根的新树。如果机器人在 τ_H^{suf}中执行后缀部分时发生故障，则采用类似的方法，尝试在局部修订后缀部分，即围绕 $\hat{}\tau_H(T)$ 的循环。如果找不到围绕 $\hat{}\tau_H(T)$ 的循环路径，则从头构造一条新的前缀-后缀路径。

G. 算法分析

在本节中，我们讨论所提出算法的正确性、完备性和最优性。

命题 *3.5*（算法的完备性/最优性*2*）：考虑一个失效谓词 $\pi\in\mathcal{AP}_F$ 和布尔公式$b^d_{q'_B,\ q''_B}$ 其中包含 π。算法2 是完备的，即，若存在对出现在 $b^d_{q'_B,\ q''_B}$ 中的所有谓词进行重新分配的方案（即，一个路径 p) 满足第 III-D 节讨论的约束，那么算法 2 将会找到它。算法 2 也是最优的，即如果存在多个可行的重新分配，它将找到重新分配次数最少的那一种。

证明：这一结果归因于广度优先搜索算法的完备性和最优性。图 $\mathcal{G}$ 通过构造涵盖了所有可能的重新分配。所提出的算法依赖广度优先搜索，在该图中寻找一条路径 p，以此对满足第 III-D 节所述约束的任务重新分配序列进行建模。由于广度优先搜索具有完备性和最优性，因此，算法 2 在求解布尔公式 $b^d_{q'_B,q''_B}$ 方面也具有完备性和最优性。■

命题*3.6*（算法*1*的完备性/最优性）：考虑一个失效谓词 $\pi\in\mathcal{AP}_F$ 以及一组非确定性 Büchi 自动机边 $\mathcal{E}$，其激活取决于 π。如果对于给定边 p，存在为所有布尔公式$b^d_{q'_B}$分配所有谓词的可行重新分配（即路径）q''_B，并且对于边 $e\in\mathcal{E}$ 以及所有 $e\in\mathcal{E}$，则算法1 将找到这些重新分配。它还将找到跨所有子公式和边的重新分配次数最少的解。

证明：请注意，给定一个失效谓词，所有边e 均由算法独立修复。 1。原因在于，第 $V^d_{q'_B}$节中的函数 q''_B 对修复 III-D 中 $b^d_{q'_B,\ q''_B}$ 内失效谓词施加的约束，在各个子公式 $b^d_{q'_B,\ q''_B}$ 之间完全独立；对于给定的一条边以及不同边之间也是如此。因此，由命题 3.5 可得该结论。■

命题 *3.7*（重新规划的可靠性/完备性）：考虑通过算法将谓词重新分配给机器人。1。如果重新规划算法（第 III-F 节）返回一个新的前缀-后缀规划，则该规划满足 ϕ。此外，如果存在满足修订后公式的可行前缀-后缀规划，则该规划算法将会找到它。

证明：在线设计的规划 τ_H 满足 ϕ，因为根据构造，其后缀部分包含一个状态$\mathbf{q}=(\mathbf{p},\mathbf{s}\ q_B)$，其中 q_B 是一个最终非确定性 Büchi 自动机状态。因此，沿该规划生成的词满足非确定性 Büchi 自动机的接受条件。此外，如果存在一个可行的规划 τ_H，所提出的算法将能够找到它。原因在于，在最坏情况下，即当前规划无法进行局部修订时，算法会从头开始，通过使用基于采样的规划器 [5]构建一棵新树，从而构造一个新的前缀-后缀规划；参见第III-F节。所采用的基于采样的规划器具有（概率意义上的）完备性，因此，在这种最坏情况下，如果存在一个可行的前缀-后缀规划 τ_H，那么 [5] 就能找到它，从而完成证明。■

备注 *3.8*（局限性）：第 III-D 节中定义的约束无法确保修订后的 LTL 公式可行。事实上，满足这些约束是确保存在能够产生接受型 NBA 运行的机器人路径的必要条件，但并非充分条件。我们注意到，这也是相关任务分配算法中的一个常见局限。在任务重新分配过程中，可以引入额外约束，对确保某个转换在物理上可实现的必要条件进行建模；例如参见[12]中的定义 3.4。

IV. 实验验证

在本节中，我们通过实验展示所提出的算法在存在意外多重故障情况下的性能。我们在一台配备 Intel Core i7-8565U 1.80GHz 处理器和 16GB RAM 的计算机上，使用 Python 3 进行了实验。相关视频可在 [26] 中查看。

A. 管道巡检任务 - 单次故障

我们重新考察了例 2.1 中讨论的管道巡检任务。所考虑的线性时序逻辑任务对应一个包含 8 个状态的非确定性 Büchi 自动机。由于机器人 3 的摄像头发生故障，最初分配给机器人 3 的 π_2无法完成。此时，调用算法 1 修复与 π_2 关联的所有非确定性 Büchi 自动机边。受影响的边总数为 6。在所有边中，π_2 被分配给机器人 2。此次重新分配耗时 0.0009 秒，新路径在 0.26 秒内生成。请注意，在图 1(d)中，机器人 2 完成了原本在 ℓ_2 操作阀门的任务，随后接替机器人 3 的角色，前往 ℓ_3 拍照。例 2.2 中讨论的案例研究结果见 [26]。

B. 空中感知任务——多重故障

我们考虑一项涉及 $N = 7$架无人机的任务，其动力学如 [27]所示。无人机的能力定义为 c_1、 c_2、 c_3和 c_4，分别对应移动能力、数据传输能力、使用摄像头拍照的技能以及红外成像能力。无人机1、2和3具备 c_1 和c_2能力。无人机4和5具备 c_1、 c_2和 c_3能力，无人机6和7具备 c_1、 c_3、 c_4能力。这些无人机负责完成一项空中感知任务，同时将收集到的数据传输至基站。

具体而言，无人机1、2和3应分别最终总是位于 ℓ_1、ℓ_2和 ℓ_3这些位置，以便以多跳方式将其他无人机收集的数据传输至基站（靠近 ℓ_1）。其余每架无人机都需要依次完成两项任务。无人机4和5需要分别在两个位置拍照，而无人机5和6需要分别在两个位置进行红外成像。该任务由以下公式表示：$\phi = \Diamond(\Box\xi_1 \wedge \pi_1 \wedge \Diamond(\pi_2)) \wedge \Diamond(\Box\xi_1 \wedge \pi_3 \wedge \Diamond(\pi_4)) \wedge \Diamond(\Box\xi_1 \wedge \pi_5 \wedge \Diamond(\pi_6)) \wedge \Diamond(\Box\xi_1 \wedge \pi_7 \wedge \Diamond(\pi_8))$，其中 ξ_1 是对传输目标建模的布尔公式，定义为 $\xi_1 = \pi_{\mathcal{T}_2}(1, c_2, \ell_1) \wedge \pi_{\mathcal{T}_2}(2, c_2, \ell_2) \wedge \pi_{\mathcal{T}_2}(3, c_2, \ell_3)$，以及 $\pi_1 = \pi_{\mathcal{T}_3}(4, c_3, \ell_4)$、 $\pi_2 = \pi_{\mathcal{T}_3}(4, c_3, \ell_5)$、 $\pi_3 = \pi_{\mathcal{T}_3}(5, c_3, \ell_6)$、$\pi_4 = \pi_{\mathcal{T}_3}(5, c_3, \ell_7)$、 $\pi_5 = \pi_{\mathcal{T}_4}(6, c_4, \ell_8)$、 $\pi_6 = \pi_{\mathcal{T}_4}(6, c_4, \ell_9)$、$\pi_7 = \pi_{\mathcal{T}_4}(7, c_4, \ell_{10})$、 $\pi_8 = \pi_{\mathcal{T}_4}(7, c_4, \ell_{11})$。该任务对应一个具有132个状态的非确定性 Büchi 自动机。

我们仿真了两架无人机 2 和4 在 $t = 9$ 时（机器人当前处于初始非确定性Büchi自动机状态），所有技能均失效的情况；这要求重新分配与其关联的原子谓词 ϕ。因此，$|\mathcal{E}| = 892$ 需要通过算法 1修复非确定性 Büchi 自动机边。算法1 将无人机2的传输任务重新分配给无人机 5 ，并将无人机 4 和 5原有的拍照任务分别分配给无人机 6 和 7。这一重新分配过程耗时 2.28 秒，而规划新路径耗时 1.1 秒。任务接近结束时，在 $t = 58$，机器人 7 也完全失效了。在这种情况下，

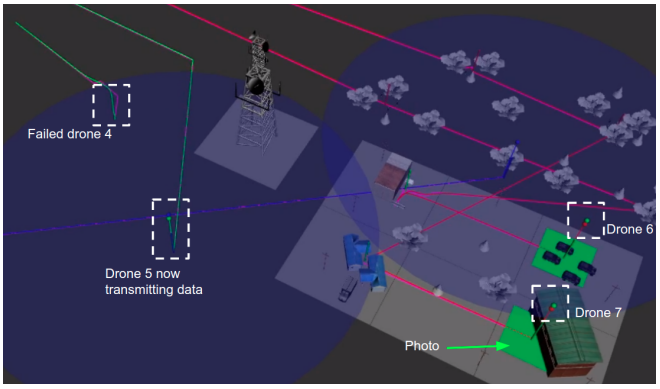


图 4. 无人机 5 执行原本分配给故障无人机 4 的通信任务（蓝色圆圈表示通信范围）。无人机 6 和 7 拍摄照片，完成最初分配给故障无人机 4 和 5 的任务。

算法1 必须修复一个失效谓词 π_2 （该谓词最初分配给机器人 4，后来由机器人 7接管）以及一条边。算法 1 在 6秒内将该谓词重新分配给机器人 0.0013 秒。这里需要指出，由于非确定性 Büchi 自动机状态总数很大（132），在任务初期（机器人处于初始非确定性Büchi自动机状态时）发生故障，会产生一个很大的集合 $\mathcal{Q}_B^{\text{cur}}$（见第 III-C 节），因此会产生一个很大的集合 $\mathcal{E}$ 需要修复的非确定性 Büchi 自动机边。因此，在这种情况下，修复失效谓词所需的运行时间比任务接近结束时发生故障所需的时间更长；在任务接近结束时，只需修复少数几条边。仿真截图见图 4。

C. 大型机器人团队中的故障

最后，我们考虑一支由 $N = 24$具备$|\mathcal{C}| = 6$ 技能的地面机器人组成的团队。这些技能涉及移动能力、关闭阀门、灭火、采集样本以及拍摄热成像和普通图像。灾害发生后，这些机器人驻留在一座工厂内，需要完成一系列不同的任务，以控制局势，具体如以下内容所示：$\phi = \Diamond(\xi_1 \wedge \Diamond(\xi_2 \wedge \Diamond\xi_3)) \wedge \Diamond(\xi_4 \wedge \Diamond(\xi_5 \wedge \Diamond\xi_6))$，其中每个 ξ_i 均为一个布尔公式，其定义在形如（1）的原子谓词之上，适用于不同的机器人，具体如第 IV-B节所述。所考虑的公式对应一个具有 25 个状态的非确定性 Büchi 自动机。在时刻 $t = 20$，8 个机器人的所有技能均完全失效，因此需要修复 $|\mathcal{E}| = 19$ 条非确定性 Büchi 自动机边。重新分配和重新规划过程分别耗时 0.004 秒和17 秒。

V. 结论

本文提出了一种适用于异构机器人团队的时序逻辑任务规划算法，该算法能够抵御机器人能力失效。所提出算法的目标已通过理论和实验得到验证。

参考文献

[1] C. Baier 和 J.-P. Katoen，《模型检测原理》。麻省理工学院出版社，剑桥，2008年，第 26202649 卷。[2] D. Aksaray、K. Leahy 和 C. Belta，“时序逻辑约束下的分布式多智能体持续监视”，*IFAC-PapersOnLine*，第 48 卷，第 22 期，第 174–179 页，2015年。[3] C. K. Verginis 和 D. V. Dimarogonas，“高层任务与不确定动力学下协作机械臂系统的控制”，*Annual Reviews in Control*，2022年。

[4] A. S. Asarkaya、D. Aksaray 和 Y. Yazıcıo˘glu，“利用受时序逻辑约束的混合强化学习，通过配送无人机执行最优空中监测”，载于《2021年国际无人机系统会议（ICUAS）》。IEEE，2021年，第285–294 页。[5] X. Luo、Y. Kantaros 和 M. M. Zavlanos，“一种用于多机器人时序逻辑最优控制综合的无抽象方法”，《IEEE 机器人学汇刊》，2021年。[6] Y. Kantaros 和 M. M. Zavlanos，“Stylus*：一种面向大规模多机器人系统的时序逻辑最优控制综合算法”，《国际机器人研究杂志》，第 39 卷，第 7 期，第 812–836 页，2020年。[7] T. Wongpiromsarn、U. Topcu 和 R. M. Murray，“滚动时域时序逻辑规划”，《IEEE 自动控制汇刊》，第 57 卷，第 11 期，第 2817–2830 页，2012年。[8] C. I. Vasile 和 C. Belta，“基于采样的时序逻辑路径规划”，载于《IEEE/RSJ 智能机器人与系统国际会议》，日本东京，2013年11月，第 4817–4822 页。[9] M. Kloetzer 和 C. Belta，“根据时序逻辑规范控制线性系统的全自动框架”，《IEEE 自动控制汇刊》，第 53 卷，第 1 期，第 287–297 页，2008年。[10] D. Gujarathi 和 I. Saha，“Mt*：针对时序逻辑规范的多机器人路径规划”，载于《IEEE/RSJ 智能机器人与系统国际会议（IROS）》，2022年，第 13 692–13 699 页。[11] M. Guo、K. H. Johansson 和 D. V. Dimarogonas，“在部分已知工作空间中根据线性时序逻辑规范修订运动规划”，载于《IEEE 机器人与自动化国际会议（ICRA）》，德国卡尔斯鲁厄，2013年，第 5025–5032 页。[12] Y. Kantaros、M. Malencia、V. Kumar 和 G. J. Pappas，“未知环境中多机器人的反应式时序逻辑规划”，载于《IEEE 机器人与自动化国际会议（ICRA）》，法国巴黎，2020年6月，第 11 479–11 485 页。[13] S. Kalluraya、G. J. Pappas 和 Y. Kantaros，“动态语义环境中的多机器人任务规划”，载于《IEEE 机器人与自动化国际会议（ICRA）》，2023年。[14] M. Lahijanian、M. R. Maly、D. Fried、L. E. Kavraki、H. Kress-Gazit 和 M. Y. Vardi，“具有部分满足保证的不确定环境迭代时序规划”，《IEEE 机器人学汇刊》，第 32 卷，第 3 期，第 583–599 页，2016年。[15] Z. Li、M. Cai、S. Xiao 和 Z. Kan，“未知动态环境中基于软度量区间时序逻辑的在线运动规划”，《IEEE 控制系统快报》，第 6 卷，第 2293–2298 页，2022 年。[16] M. Hasanbeig 、 Y. Kantaros、A. Abate、D. Kroening、G. J. Pappas 和 I. Lee，“具有概率满足保证的时序逻辑控制综合强化学习”，载于《IEEE 决策与控制会议（CDC）》，法国尼斯，2019年12月。[17] M. Cai、M. Mann、Z. Serlin、K. Leahy 和 C.-I. Vasile，“针对不可行线性时序逻辑规范学习最小违反的连续控制”，载于《美国控制会议》，2023年。[18] X. S un 和 Y. Shoukry，“面向线性时序逻辑任务的神经符号运动与任务规划”，arXiv 预印本 arXiv:2210.05180，2022年。[19] P. Schillinger、M. B¨urger 和 D. Dimarogonas，“用于高效多智能体规划的有限 LTL 规范分解”，载于《第 13 届分布式自主机器人系统国际研讨会》，英国伦敦，2016年11月。[20] C. Banks、S. Wilson、S. Coogan 和 M. Egerstedt，“使用交叉熵时序逻辑优化的多智能体任务分配”，载于《IEEE 机器人与自动化国际会议（ICRA）》，2020年，第 7712–7718 页。[21] X. Luo 和 M. M. Zavlanos，“异构多机器人系统中的时序逻辑任务分配”，《IEEE 机器人学汇刊》，第 38 卷，第 6 期，第 3602–3621 页，2022年。[22] F. Huang、X. Yin 和 S. Li，“线性时序逻辑规范下的故障鲁棒多机器人任务规划”，载于《第 13 届亚洲控制会议（ASCC 2022）》。IEEE，2022年。[23] Z. Zhou、D. J. Lee、Y. Yoshinaga、S. Balakirsky、D. Guo 和 Y. Zhao，“四足机器人与轮式机器人协作的反应式任务分配与规划”，载于《2022年 IEEE 第 18 届自动化科学与工程国际会议（CASE）》，2022年，第 2110–2117 页。[24] F. Faruq、D. Parker、B. Laccrda 和 N. Hawes，“不确定性下的任务分配与规划同步进行”，载于《2018年 IEEE/RSJ 智能机器人与系统国际会议（IROS）》，2018年，第 3559–3564 页。[25] P. Gastin 和 D. Oddoux，“快速将 LTL 翻译为 B¨uchi 自动机”，载于《计算机辅助验证国际会议》。Springer，2001年，第 53–65 页。

[26] “仿真视频：面向异构机器人团队的弹性时序逻辑任务规划”，https://vimeo.com/813432282。[27] F. Furrer、M. Burri、M. Achtelik 和 R. Siegwart， 机器人操作系统（*ROS*）：完整参考（第*1*卷）。Cham：施普林格国际出版公司，2016年，第 RotorS——模块化 Gazebo MAV 仿真器框架章，第595–625页。 [在线]。可访问：http://dx.doi.org/10.1007/978-3-319-26054-9 23