一种用于在线取件与准时配送问题的动态异构团队非迭代方法

Shridhar Velhal[a]，Srikrishna B R[a]，Mukunda Bharatheesha[b]，Suresh Sundaram[a,1]

[a]印度科学学院航空航天工程系，印度班加罗尔[b]印度科学学院罗伯特·博世信息物理系统中心，印度班加罗尔

arXiv:2304.07124v1 [cs.MA] 14 Apr 2023

摘要

本文提出了一种非迭代方法，用于确定异构机器人的任务分配，以高效执行在线取件与准时配送（PJITD）任务，并实现最优资源利用率。PJITD任务分配问题被建模为时空多任务分配（STMTA）问题。成本建模纳入了地图中的物理约束和车辆动力学。针对异构STMTA问题，本文建立了线性和分配问题。本文对近期提出的多任务分配动态资源分配（DREAM）方法进行了改进，以求解异构PJITD问题。首先，该方法计算执行给定异构PJITD任务所需的最少机器人数量（及其类型），并将这些机器人加入团队，以确保所有PJITD任务均可行。随后，为团队分配机器人来执行PJITD任务，同时使团队执行所有PJITD任务的总成本最小化。本文通过高保真软件在环仿真和硬件实验验证了所提出的非迭代方法的性能。仿真和实验结果清楚表明，该方法具有良好的可扩展性，并能实现最优资源利用率。

关键词：时空　间任务、时间调度、异构资源分配、多智能体取货与配送　　ry,
准时制

1. 引言

随着技术的发展，机器人已应用于各种工业领域。多机器人系统为处理各类作业提供了一种分布式、可靠且可扩展的方法。在物联网和工业 4.0 技术发展的助力下，自动化行业采用准时制 [1, 2] 方法，以优化仓储和库存管理。仓库是电子商务行业供应链中的关键枢纽，仓库自动化正变得至关重要 [3, 4, 5, 6]。客户要求物品快速、准时配送，时间也日益成为电子商务行业的关键因素。对于食品和饮料等易腐物品，配送任务对时间的敏感性更高。在电子商务领域竞争持续不断的情况下，即使是非易腐物品，准时配送也已成为决定成败的关键因素。仓库管理和电子商务是需要在线解决方案的重要应用领域，而采用准时制任务可以提升这些领域的效能。

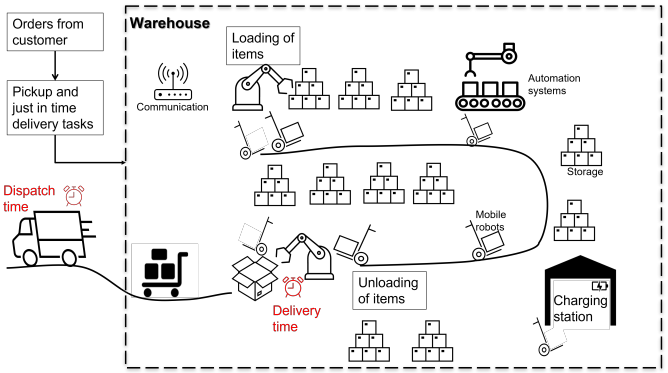


图1. 典型仓库及其运营

典型的仓库中有许多物品需要在不同位置之间取放，目前这项工作由自主机器人完成。如果物品能在准确的时间送达，后续流程便可立即开始，从而提高作业效率。此外，这还能减少或消除对本地存储空间的需求。为订单包装不同物品就是一个例子：所有物品都必须在所需时间送达包装柜台。如果所有物品都能在所需时间到达包装柜台，便无需本地存储——

；这也有助于通过减少冗余的取放操作来提高效率。制造业和汽车工业采用准时制（JIT）管理策略，使供应商的原材料订单与生产计划直接对接。准时制方法的一大隐忧是供应链可能受到干扰。本文提出在仓库作业中使用机器人执行取货与准时配送任务，从而在由机器人维护稳健供应链的同时，获得准时制方法带来的益处。

文献 [7]通过分布式资源分配解决了多机器人取送问题。其成本函数旨在最小化机器人执行

取货与配送任务时行驶的总距离。文献 [8], 提出了一种集成方法，用于解决容量受限的多智能体取送问题中的任务分配和路径规划。该方法还支持运输过程中携带多个包裹。基于边际成本和遗憾的基于边际成本的算法，在避免碰撞的同时最小化总行程延误。文献[9], 解决了使用无人机进行最后一英里配送的多任务分配问题，其中无人机必须逐件取货并配送物品。文献 [10], 提出了仓库作业中的自适应任务分配，以应对系统动态变化，例如任务位置、机器人数量、新库存补货以及机器人的电池状态。文献 [11], 将取货和配送视为独立任务，并设置优先约束，要求同一机器人先取货再配送，从而降低了组合式取货与配送问题的复杂度，以解决最后一英里配送问题。但这会使任务数量增加一倍。

文献 [12] 对具有不同时间约束的任务分配与调度进行了详细综述。截止期限问题和时间窗问题 [13, 14, 15, 16, 17], 是仓库问题研究中得到充分探讨的两类关键问题。截止期限任务要求在物品用于后续流程（该流程仅会在截止期限之后开始）之前，将物品存放在本地存储中。时间窗问题要求在时间窗内配送物品，而本地存储仅在该时间窗内可用。带时间窗的旅行商问题 [18] 为受时间约束的旅行商问题提供了数学框架。该求解方法将取货和配送视为独立任务，并对机器人施加约束：负责取走物品的机器人必须配送该物品，且只有在取货之后才能配送。这增加了优化问题的约束数量和维度。文献 [19], 研究了时间窗大小的影响，结果表明，缩短时间窗可以提高客户满意度并减少计算时间，但会延长行程持续时间。

上述所有仓库自动化研究均未考虑准时制任务，并假设给定团队能够完成任务，因此都只采用固定规模的机器人团队。因此，亟需开发一种算法，利用动态规模的异构机器人团队处理异构准时制任务，并以最少的资源（机器人）计算出可行解。

本文提出一种异构资源分配方法，用于仓库管理中的异构机器人在线取货与准时配送问题。我们对 [20] 中定义的成本函数进行了修改，使其适用于异构机器人执行异构任务。修改后的成本函数还考虑了机器人在取货和配送过程中行驶的总距离，以及机器人所需的装载时间（取货时）和卸载时间（配送时）。所提出的 PJITD任务异构资源分配方法可提供非迭代解，计算动态规模机器人团队执行给定异构时空任务的最优轨迹。首先，计算执行给定异构时空任务所需的机器人数量。从不同技能集中选取相应数量的机器人，加入活跃机器人团队，最后计算更新后的异构机器人团队的可行分配。通过这种方式，最多只需两步，便可计算出以最少资源（机器人）执行给定异构时空任务的最优分配。根据异构时空多任务分配（STMTA）的解，使用轨迹生成算法计算活跃机器人的轨迹，随后由机器人团队执行所有给定任务。本文通过仿真和硬件实验展示了DREAM算法在 PJITD任务中的运行情况。高保真仿真在ROS2-Gazebo环境中进行。通过实验室规模硬件实验，展示了所提出的 PJITD问题异构资源分配方法的运行情况。

本文其余部分安排如下：第 2 节介绍相关工作。第 3 节给出在线取货与准时制配送任务分配问题的数学建模。第 4 节介绍用于计算可行任务分配的异构资源分配方法。第 5 节展示所提出方法的运行过程。第 6 节总结全文。

2. 相关工作

本文结合准时制任务、时空多任务分配问题以及动态规模机器人团队的思路，以执行给定的时空任务。下文简要回顾这些相关工作。

2.1. 准时制（*JIT*）

准时制 [1, 2] 要求任务必须在指定时间完成；这有助于优化库存管理和仓储。近期，[21] 提出了一种名为零仓储与智能制造的新方法，其中提出了基于物联网的零仓储，以尽量减少非增值和冗余的仓库作业流程，同时缩减仓储空间。近期，[22, 23] 提出了适用于自动导引车取货与配送的准时制方法。该方法构建了成本函数，以尽量减少实际取货和配送时间与期望时间之间的偏差，因此能够灵活处理时间约束。上述关于准时制的研究均针对静态环境设计，在这类环境中，任务事先已知，解决方案可以离线计算。

2.2. 时空多任务分配（*STMTA*）

Chopra 和 Egerstedt [24, 25]提出了多机器人路径规划问题，并以音乐墙为例进行了说明：机器人到达不同音符位置，并在各自指定的精确时间演奏相应的音符。由于时空任务必须在指定时间完成，因此需要一定数量的机器人。时空任务分配的主要问题是计算所需的最少机器人数量

，以执行给定的时空任务。在 [24, 25], 中，针对给定任务，所需的最少机器人数量是通过迭代方法离线计算的。这种计算所需最少机器人数量的迭代方法，给 STMTA 的在线应用带来了很大障碍。

2.3. 面向 *S* 的动态资源分配方法 *TMTA*

分布式多任务分配动态资源分配（DREAM） [20] 方法已被提出，用于解决时空多任务分配问题。该方法提供了一种非迭代解，用于计算执行给定时空任务所需的同质机器人数量，并为这些机器人分配任务。非迭代 DREAM 方法已在 [26] 中实现，用于计算动态规模音乐演奏机器人团队的无碰撞轨迹（即由同质机器人执行的准时任务）。DREAM 方法仅适用于同质智能体，且只考虑简单的路径规划任务。PJITD任务需要针对异构机器人的解决方案，因此 DREAM 不能直接用于 PJITD任务。

仓库自动化需要一种非迭代（在线）解决方案，将多个复杂任务（由若干子任务和等待组成）分配给规模最优的异构机器人团队（机器人具有不同的速度和载荷能力）。DREAM算法提供在线解决方案，但只考虑由同质机器人执行的简单路径规划任务。因此，需要开发一种可在线实现的算法，以处理仓库环境中的在线、动态和异构复杂任务。

3. 取货与准时交付任务

典型的仓库作业如图 1 所示。仓库的主要运营目标是在收到订单后，尽可能缩短物品从仓库发运所需的时间。订单中的物品从仓库发运至客户附近的某个本地枢纽。在计划将物品运往该枢纽/客户的车辆出发前，必须优先收集所有属于同一本地枢纽的物品。可采用线性时序逻辑方法[27] 生成自动化子任务；使用装箱算法 [28] 计算物品的打包顺序。由于同一包裹中的所有物品都应一起打包，准确的交付时间有助于加快打包流程。同一包裹中的所有物品可以同时到达并直接打包，无需本地存储，也不会产生时间延迟。这还可以减少为本地存储而进行的冗余取放操作。通过这种方式，可以提高作业效率。

在仓库作业中，需要由人工操作员（在取货点附近）或机械臂系统（安装在每台机器人上）取出多个物品并运送它们。为避免协同工作环境带来的复杂性，本文假设每台机器人一次只能执行一项任务。也就是说，如果机器人取走了一件物品，就必须先将其配送到目的地，然后才能取走另一件物品。机器人可以提前规划未来的任务，但必须先完成当前任务，才能开始下一项任务。

本文假设，一旦收到客户的订单，便会定义子任务，并生成取货与准时配送（PJITD）问题。本文提出了PJITD问题的解决方案，旨在尽量减少所需机器人数量以及动态规模机器人团队为按时完成所有PJITD任务而行驶的总距离。PJITD任务要求具备所需技能集的机器人在指定时间将物品取出并配送至指定位置；因此，这类任务也称为异构时空任务。需要注意的是，给定的准时/时空任务需要至少一定数量的机器人才能按时完成。本文的目标是计算机器人执行在线异构任务的分配方案，同时尽量减少资源（即实际投入使用的机器人数量）和总行驶距离。

首先，我们定义本文中使用的符号。$p = (x, y) \in \mathbb{R}^2$: 笛卡尔坐标中的位置 p^R: 机器人的位置 p^P ： 取件位置 p^D: 交付地点 R_i ： 机器人编号 i，（i 用于表示机器人）V^R i: R_i 的最大速度 Q_ℓ: 质量/技能集 ℓ, $\ell \in \mathcal{L} = \{1, 2, \cdots, n_\ell\}$, $Q(R_i)$: 机器人 R_i 的质量/技能集 τ^l ： 装载时间τ^u ： 卸载时间 t_j^D: 任务 T_j 的交付时间 Q(T*j*) = : 执行任务 T j 所需的质量/技能集。$T j(p_j^P, \tau_j^l, p_j^D, \tau_j^u, t_j^D, Q(T j))$ 或 T j: 第 j^{th} 个任务（索引 j 和 k 用于表示任务）μ_i: 分配给机器人 R_i 的任务序列 c f,Q(R_i) i j: 机器人 R_i（其质量为 $Q(R_i)$）将 T j 作为首个任务执行的成本 c^{s,Q_ℓ}k j: 质量为 Q_ℓ 的机器人在任务 T_k 之后立即执行任务 T j 的成本（后续任务）。δ f,Q(R_i) i j: 决策变量，用于表示质量为 $Q(R_i)$ 的机器人是否将任务 T_j 作为首个任务执行。δ^{s,Q_ℓ}k j: 决策变量，用于表示质量为 Q_ℓ 的机器人是否在任务 T_k 之后立即执行任务 T j。

3.1. 数学公式

考虑一组由 $\mathcal{R}$, $\mathcal{R}$ ={$R_1, R2, \cdots, R_N$}表示的 N 个机器人。机器人 R_i 的位置表示为 $p_i^R = (x_i^R, y_i^R)$。通常，机器人具有不同的有限技能（例如，载重能力、尺寸承载能力），并用集合 lambda$_e$ll 表示。机器人将被分配执行取货与配送任务。任务 T_j 的取件位置表示为 $p_j^P = (x_j^P, y_j^P)$，交付位置表示为 $p_j^D = (x_j^D, y_j^D)$。充电站位于 $S_i = (x_i^C, y_i^C)$。仓库机器人有两种工作模式：活动模式（机器人被分配执行任务时）和休息模式。在休息模式下，机器人要么处于空闲状态，要么在充电站为电池充电。

3.1.1. 取货与准时制配送任务

取件与准时配送（PJITD）任务包括装载物品（机器人必须在取件站停留一段装载时间）、前往交付地点，以及卸载物品（机器人必须等待一段卸载时间）。这一执行过程必须在期望配送时间内完成。如果机器人开始执行任务并在期望配送时间内完成，则该 PJITD 任务完成。如果机器人到达交付地点时已晚于期望配送时间，则任务执行失败。考虑一个 PJITD 任务（T_j），机器人必须前往取件位置（$\boldsymbol{p}_j^P$），等待装载时间（τ_j^l）以将物品装载到机器人上。取件后，机器人应到达位于（$\boldsymbol{p}_j^D$）的期望包装/处理（交付）柜台，并在交付时间（t_j^D）之前或当时完成物品卸载，卸载时间为（τ_j^u）。该任务表示为$T_j(\boldsymbol{p}_j^P, \tau_j^l, \boldsymbol{p}_j^D, t_j^D, \tau_j^u, Q(T_j))$；在本文其余部分中，该 PJITD 任务记作 T_j。

一个任务由以下子任务组成：

(a) 选择具备所需技能的机器人（即$Q(T_j) \subseteq Q(R_i)$）；(b) 机器人应到达取件位置$\boldsymbol{p}_j^P$；(c) 等待 τ_j^l 装载物品；(d) 到达配送/卸货位置$\boldsymbol{p}_j^D$；(e) 等待至交付时间t_j^D；(f) 等待 τ_j^u 卸下物品

执行任务时行驶的空间距离，是机器人当前位置到取件位置（$\boldsymbol{p}_j^P$）的距离，再加上前往$\boldsymbol{p}_j^D$处配送站的距离。此任务必须满足配送时间约束。

收到客户订单后，系统会生成任务，任务分配器会将任务分配给机器人。机器人必须按顺序执行分配任务。假设分配给机器人R_i 的任务为 μ_i、$\mu_i = \{T_a$、$T_b\}$，那么机器人会从其初始位置出发，前往任务T_a的位置取货（即$\boldsymbol{p}_a^P$），然后在交付时间t_a^D或之前将物品送至任务T_a 的交付地点（即$\boldsymbol{p}_a^D$）。接着，机器人从任务T_a的交付地点前往任务T_b 的取件位置，领取任务中的物品，并将其送至交付地点$\boldsymbol{p}_b^D$。简而言之，机器人从前一个任务的交付地点前往下一个分配任务的取件位置，并按顺序执行任务。

我们考虑任意给定时刻t可用的 PJITD 任务：$\left\{T_1\left(\boldsymbol{p}_1^T、t_1\right)、T_2\left(\boldsymbol{p}_2^T、t_2\right)、\cdots、T_{M_t}\left(\boldsymbol{p}_{M_t}^T、t_{M_t}\right)\right\}$。通常，任务数量（$M_t$）多于机器人数量（$N$），即$M_t > N$。请注意，任务数量取决于客户的订单，每收到一份新订单，就会增加一个新任务。不过，在给定时刻t，任务数量为M_t。假设M_t 个任务均可由N 个机器人组成的团队完成，那么这些N 个机器人需要协同规划各自的轨迹（由取件点和配送点及其对应的交付时间组成）$\{\mu_1, \mu_2、\cdots、\mu_N\}$，使所有机器人能够共同完成所有 PJITD 任务。

该 PJITD 问题的主要目标是为机器人寻找多个任务的最优分配，使所有 PJITD任务都能执行，同时最小化总行驶距离并优化资源利用率。该 PJITD任务分配的主要挑战是计算执行所有给定 PJITD任务所需的最少机器人数量。确定最少机器人数量后，再求解异构时空多任务分配问题，以计算机器人的可行最优轨迹。下一节将介绍该方法的详细内容。

4. PJITD任务分配问题的异构资源分配方法

上一节已讨论PJITD任务。本节介绍将机器人分配给这些PJITD任务并执行它们的算法。为执行给定的PJITD任务，机器人需要在特定时间依次前往各个位置。对于物品取货，机器人可以在任意时间前往取货地点；但对于配送，机器人必须在期望配送时间到达交付地点。由于时间约束，需要一定数量的机器人才能执行所有给定的异构时空任务。因此，我们采用规模动态变化的机器人团队，并使用执行给定PJITD任务所需的最少机器人数量。所提出的算法会计算执行给定PJITD任务所需的最少机器人数量。随后，算法会计算更新后的团队中各机器人的最优轨迹，使其能够执行所有给定的PJITD任务。为使总成本最小，算法会为机器人分配任务；PJITD任务的执行成本将在本小节中说明。

每台机器人执行分配给自己的任务；从当前位置出发，从取货点取走物品，并在交付地点的期望配送时间准确交付。随后，机器人前往下一个分配任务的取货点。机器人按顺序执行分配给自己的任务；该顺序既包含空间位置，也包含时间，这种顺序称为该机器人的轨迹（μ）（因为它同时包含空间位置和时间）。所有机器人的可行轨迹根据机器人从当前位置出发、继而前往分配任务的交付地点所采取的行动计算得出。

二元决策向量 $\boldsymbol{\delta}$有两个组成部分，即第一个分配任务（$\boldsymbol{\delta}^f$, $_{Q(R_i)}\mathrm{i} \in \mathbb{R}^{M_t}$）和随后分配的任务（$\boldsymbol{\delta}^s$,$_{Q_\ell}\mathrm{k} \in \mathbb{R}^{M_t}$）。决策变量表示为 $\boldsymbol{\delta}^f$,$_{Q(R_i)}\mathrm{i} = \left[\delta_{\mathrm{f,Q(R_i)\,i1}}, \delta_{\mathrm{f,Q(R_i)\,i2}}, \cdots, \delta_{\mathrm{f,Q(R_i)\,iM}}\right], \mathrm{i} = \{1, 2, \cdots, \mathrm{N}\}$, 以及 $\boldsymbol{\delta}^s$,$_{Q_\ell}\mathrm{k} = \left[\delta^{s,Q_\ell}_{1}\mathrm{k}, \delta^{s,Q_\ell}\mathrm{k2}, \cdots, \delta^{s,Q_\ell}\mathrm{kM}_t\right], \mathrm{k} = \{1, 2, \cdots, M_t - 1\}$，$\ell = \{1, 2, \cdots, n_\ell\}$。

第一个决策变量 $\delta^{f,\ Q(R_i)\,ij} \in \{1, 0\}$表示机器人 R_i 是否首先执行任务 T_j。后续决策变量 $\delta^{s,\ Q_\ell\,kj} \in \{1, 0\}$表示机器人 R_i是否紧接在执行任务 T_k 之后执行任务 T_j。决策变量 $\boldsymbol{\delta}^{f,\ Q(R_i)}$ 和 $\boldsymbol{\delta}^{s,\ Q_\ell}$ 经过优化以使成本（燃料消耗）最小化；该成本取决于机器人为在各自的配送时间执行所有 PJITD任务而行驶的距离。

4.1. 成本函数

任务的成本定义为：机器人从其所在位置前往取件位置，再前往交付地点，并在期望配送时间之前完成任务所需行驶的距离；此外，机器人还必须具备执行该任务所需的技能。对于具备所需技能、从初始位置执行首个任务的机器人，首个任务的成本（$\mathbf{C}^{f,\ Q(R_i)}$）是机器人 R_i 从当前位置前往取件位置，再从取件位置前往交付地点所行驶的距离，且必须在PJITD任务的配送时间或之前完成。数学上，

$$d_1(R_i, T_j) = d(\boldsymbol{p}_i^R, \boldsymbol{p}_j^P) + d(\boldsymbol{p}_j^P, \boldsymbol{p}_j^D) \tag{1}$$

$$C_{ij}^{f,Q(R_i)} = \begin{cases} d_1(R_i, T_j) & \text{if } \dfrac{d_1(R_i, T_j)}{\overline{V}_i^R} \le t_j^D - \tau_j^l \\ \kappa & \text{if } \dfrac{d_1(R_i, T_j)}{\overline{V}_i^R} > t_j^D - \tau_j^l \\ \kappa & \text{if } Q_i^R \nsubseteq Q_j^T \end{cases} \tag{2}$$

$$\text{for } i \in \mathcal{I} = \{1, 2, \cdots, N\}, \quad j \in \mathcal{J} = \{1, 2, \cdots, M_t\}$$

其中 κ 是一个很大的值，$d(A,B)$ 表示从点 A 到点 B 的最短可行路径距离。

机器人R_i执行后续任务（$\mathbf{C}^{s,Q_\ell}$）的成本，是机器人从上一次配送地点出发，到达当前取件位置，再从当前取件位置前往当前交付地点，并在该后续任务的交付时间之前或准时到达所行驶的距离。如果机器人不具备执行该任务所需的技能集，则成本设为 κ。如果任务要求在负时间内执行，则成本设为 ∞。

$$d_2(T_k, T_j) = d(\boldsymbol{p}_k^D, \boldsymbol{p}_j^P) + d(\boldsymbol{p}_j^P, \boldsymbol{p}_j^D) \tag{3}$$

$$C_{kj}^{s,Q_\ell} = \begin{cases} d_2(T_k, T_j) & \text{if } t_{k,j}^\ell \le t_j^D - (t_k^D + \tau_k^u + \tau_j^l) \\ \kappa & \text{if } t_{k,j}^\ell > \left(t_j^D - (t_k^D + \tau_k^u + \tau_j^l)\right) > 0 \\ \kappa & \text{if } Q(T_j) \nsubseteq Q_\ell \\ \infty & \text{if } \left(t_j^D - (t_k^D + \tau_k^u + \tau_j^l)\right) \le 0 \end{cases} \tag{4}$$

$$\text{for } k \in \mathcal{K} = \{1, 2, \cdots, M_t - 1\};\ j \in \mathcal{J};\ \ell \in \mathcal{L}$$

其中 $t^\ell{}_{k,j}$ 表示质量为 Q_ℓ 的机器人 R 从任务 T_k 的位置前往任务 T_j 所需的最短时间，计算方式如下

$$t_{k,j}^\ell = \frac{d_2(T_k, T_j)}{\overline{V}_\ell} \tag{5}$$

4.2. 优化问题

异构资源分配算法将机器人分配给多个 PJITD 任务。机器人将按顺序执行任务，我们用 μ_i表示分配给机器人R_i的任务序列。在这里，计算序列 μ_i的问题已转化为计算单个机器人从一个位置移动到另一个位置的每一步；将所有移动组合起来，即可得到任务序列。每个机器人计算自己的序列，以便在执行所有 PJITD 任务的同时，使行驶距离最短。第一个决策变量 $\delta_{f,Q(R_i)\,ij}$用于表示机器人R_i 是否从位置$\boldsymbol{p}_i^R$移动到位置$\boldsymbol{p}_j^P$、取货任务并将其配送至配送站$\boldsymbol{p}_j^D$，时间为t_j' 和$t_j' \le t_j^D$。后续决策变量 $\delta^{s,Q_\ell}{}_{kj}$ 用于表示机器人R_i 是否从上一项任务的配送位置$\boldsymbol{p}_k^D$（时间为t_k'）移动到下一项任务的取货位置$\boldsymbol{p}_j^P$，并在交付时间t_j^D之前或当时将物品配送至配送站（$\boldsymbol{p}_j^D$）。整数规划问题定义如下：

$$\begin{aligned} \min_{\delta_{ij}^{f,Q(R_i)}\ \delta_{kj}^{s,Q_\ell}} & \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} C_{ij}^{f,Q(R_i)}\delta_{ij}^{f,Q(R_i)} \\ & + \sum_{\ell\in\mathcal{L}}\sum_{k\in\mathcal{K}}\sum_{j\in\mathcal{J}} C_{kj}^{s,Q_\ell}\delta_{kj}^{s,Q_\ell} \end{aligned} \tag{6}$$

$$\text{s. t. } \delta_{ij}^{f,Q(R_i)} \in \{0,1\} \qquad \forall (i,j) \in \mathcal{I}\times\mathcal{J} \tag{6a}$$

$$\delta_{kj}^{s,Q_\ell} \in \{0,1\} \qquad \forall (\ell,k,j) \in \mathcal{L}\times\mathcal{K}\times\mathcal{J} \tag{6b}$$

$$\sum_{i\in\mathcal{I}} \delta_{ij}^{f,Q(R_i)} + \sum_{\ell\in\mathcal{L}}\sum_{k\in\mathcal{K}} \delta_{kj}^{s,Q_\ell} = 1 \quad \forall j \in \mathcal{J} \tag{6c}$$

$$\sum_{j\in\mathcal{J}} \delta_{ij}^{f,Q(R_i)} \le 1 \quad \forall i \in \mathcal{I} \tag{6d}$$

$$\sum_{\ell\in\mathcal{L}}\sum_{j\in\mathcal{J}} \delta_{kj}^{s,Q_\ell} \le 1 \quad \forall k \in \mathcal{K} \tag{6e}$$

所有任务都必须恰好分配给一台机器人，作为其首个或后续任务；该约束由式（6c）给出。机器人从当前位置出发后，最多只能移动到一个任务位置，这一约束由式（6d）和（6e）限定。

4.3. 异构资源分配方法 *ch*

PJITD任务本质上具有时空属性。时空任务需要在时间约束内执行；由于时间约束，给定数量的机器人可能无法执行所有时空任务。团队中需要配备一定数量的机器人，才能执行给定的所有异构时空任务。由于时间约束，配备给定数量的N 个活动机器人时，异构时空任务可能可行，也可能不可行。因此，首先对一种动态资源分配算法 [20] 进行修改，使其适用于异构任务（下文称为异构资源分配算法）。异构资源分配算法计算具备不同技能集的机器人所需的最少活动机器人数量。一旦所有技能集所需的这些最少活动机器人均可用，就可以保证更新后的异构时空多任务分配问题是可行的。然后，通过将机器人分配给多个任务，求解这一可行的异构时空多任务分配（STMTA）问题。在需要时，可以将机器人从休息模式切换为活动模式。此外，如果执行任务时不需要某个机器人，则可以将其设置为休息模式。

在异构资源分配算法中，算法首先求解优化问题（不保证得到可行解）（算法1中的步骤2）。根据计算得到的解（可能可行，也可能不可行），识别出所有不可行分配（成本等于 κ），并将相同数量的休息模式机器人切换到活动模式（算法1中的步骤7）。在添加这些备用资源时，应选择

算法1 异构资源分配算法

使用当前的N个机器人和M个任务进行初始化
求解优化问题（公式（6））
q = 个不可行分配（即 分配成本等于 κ）
若$q > 0$ **则**
  查找q个靠近以下位置的不同机器人（具备所需技能）：不可行任务的取货位置。即 $Q(R_i^{reserve}) \supseteq Q(T_{\text{不可行 } i}) \quad \forall i = \{1, 2, \cdots, q\}$
  将这些q个机器人加入团队
  $N = N + q$
  转到第 2 步
结束 if
如果 所有 N 个机器人均未分配给任务**则**
  p = 未分配机器人的数量
  未分配的机器人将被设置为休息模式
  $N = N - p$
结束条件
使用以下方法计算分配给每个机器人的任务序列 算法 2
将团队中未分配任何任务的机器人设置为休息模式

算法 2 轨迹计算

输入：任务分配解
对于 i = 1:N**执行**
  如果 $\sum_j \delta^f_{,Q(R_i)\,i,j} = 1$ **则**
    $k^* = arg_k(\delta^f_{,Q(R_i)\,i,k} = 1)$
    $\mu_i = \{(p^P_{k^*}, \cdot), (p^D_{k^*}, t^D_{k^*})\}$
    当 $\sum_j \delta^s_{,Q_\ell k^*,j} = 1$ **do**
      $j^* = arg_j(\delta^s_{,Q_\ell k^*,j} = 1)$
      $\mu_i = \{\mu_i, (p^D_{j^*}, \cdot), (p^D_{j^*}, t^D_{j^*})\}$
      $k^* = j^*$
    结束 while 循环
  结束 if 条件
结束 for 循环

具备执行所考虑的不可行任务所需技能集的备用资源。使用更新后的异构机器人团队再次求解任务分配问题。如果得到的解已经可行，则按照该解将机器人分配给任务。随后，机器人使用轨迹计算算法计算各自的轨迹。如果有机器人未被分配任务，则将该机器人设为休息模式。

4.3.1. 轨迹计算

根据 STMTA 的可行解，使用算法 2 所示的轨迹生成算法计算分配给机器人的任务序列。每个机器人独立计算自己的轨迹，并按照该轨迹执行分配给自己的 PJITD任务。轨迹生成算法从机器人自身的位置出发，计算分配给该机器人的任务序列。第一个分配任务定义前两个航点（即取货和配送），如步骤 5 所示。请注意，取货任务没有精确的时间约束；因此，取货时间表示为 $\cdot$(dot)，表示在前一个任务之后、当前任务交付时间之前的可行时间（考虑所需的行驶时间）。

取货后，下一个航点是时空配送点，它同时定义了交付地点和交付时间。将第一个任务添加到轨迹后，算法会检查是否有任务从当前位置的交付地点分配出去。如果有后续任务从该交付地点分配出去，就将该任务追加到轨迹中（算法 2 第 9 行）。这一追加后续任务以更新轨迹的步骤会不断重复，直到没有任务从更新后轨迹的最后一个交付地点分配出去。

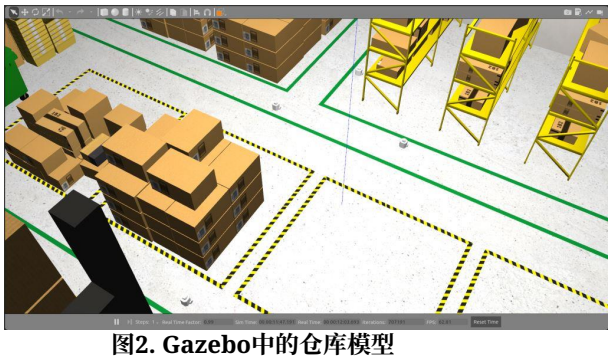


图2. Gazebo中的仓库模型

5. 性能评估

本文提出的异构资源分配方法在 PJITD任务分配问题中的运行过程，通过 ROS2-Gazebo 仿真和实验室规模硬件实验进行了说明。

5.1. 高保真仿真研究

5.1.1. 仿真设置

通过仿真环境展示所提出的DREAM算法在取货与准时交付问题中的应用。使用带有*RViz2* 插件的Gazebo仿真器，并通过ROS2（galactic）和Python运行。使用*Nav2* 插件实现仿真中机器人的导航。仿真在配备i7-8700 CPU、16GB RAM和NVIDIA GT710 GPU的Ubuntu系统上进行。

仿真针对在Gazebo中设计的小型仓库环境进行；图2展示了该仓库。仿真中共考虑了四台机器人和七个PJITD任务。

5.1.2. 仿真架构

图 3 展示了仿真软件的功能模块。该架构由 4 个模块组成：环境、时空任务分配器、多机器人导航器和通用接口。

环境。 环境模块对应仿真环境，由机器人及仿真中的所有其他物体组成。它还提供各机器人的感知数据。所有机器人都使用 SLAM 算法进行定位，并获取环境的实时地图。机器人将在导航器的指示下于环境中运行以执行任务。

中央系统。该软件模块充当环境、多机器人导航模块和时空任务分配生成模块之间的通用接口。它接收用户/客户提交的任务，并将其发送给任务分配生成模块。计算出任务分配方案后，中央系统会接收这些方案。随后，这些方案会共享给导航模块，以执行任务。同时，如果收到新任务，中央系统会检查正在执行的任务状态，然后更新机器人未来时刻的位置，接着调用时空任务分配例程。

时空任务分配服务。中央系统会向任务分配服务提供所有任务信息，即相应的取货和配送地点以及配送时间。在时空任务分配服务中，机器人会计算导航距离，以生成成本矩阵。*Nav2* 内置的*ComputePathToPose* 动作客户端服务（使用 Dijkstra 算法计算机器人从一个位置到另一个位置的可行路径）用于生成可行轨迹。接下来，对生成的轨迹进行线积分即可得到距离。然后，根据式（2）和式（4）考虑期望时间内的可行性，计算成本矩阵。接着，使用 SciPy 中 optimize 库的线性和分配函数求解式（6）定义的优化问题。随后，使用轨迹生成算法计算各机器人在获得的任务分配方案下的轨迹。这些轨迹将作为输出返回给中央系统。

多机器人导航器。中央系统收到每个机器人的轨迹后，会将其发送至导航服务。

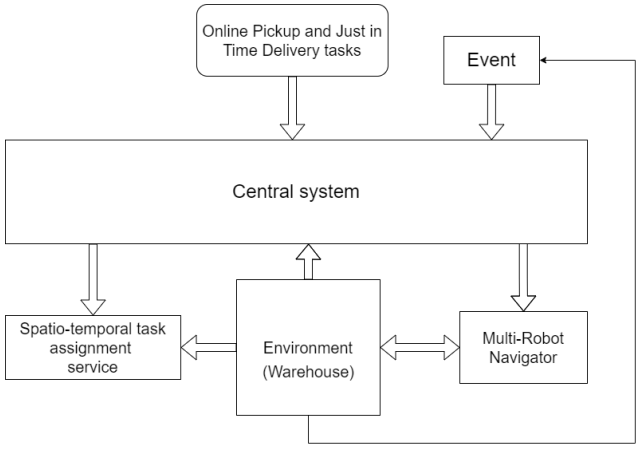


图 3. 仿真架构

	T_0	T_1	T_2	T_3	T_4	T_5	T_6
取货位置	4.0,1.0	1.0,-3.5	-4.0,0.5	1.5,5.0	1.0,3.5	2.0,1.0	2.0,2.0
配送位置	1.0,2.5	-2.5,-2.5	1.0,-2.0	-4.0,2.5	4.0,-2.5	0.0, 2.5	4.0, 2.0
配送 y time	35	50	60	75	100	120	140
装载时间（τ^l）	1	1	1	2	2	1	1
卸载时间（τ^u）	1	2	1	1	2	1	1

表1：所考虑任务的取货和配送地点，以及相应的配送、装载和卸载时间

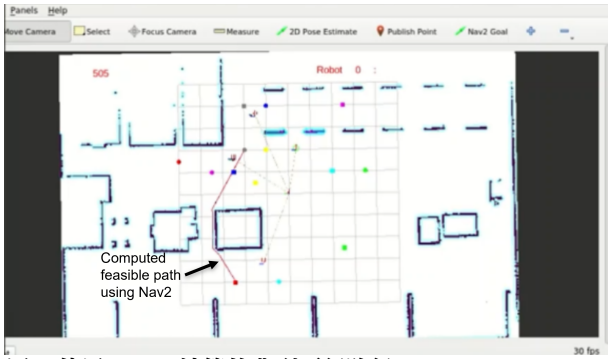


图4. 使用 Nav2 计算的典型可行路径

导航节点使用 *Nav2* 插件为每个机器人导航。为使多个机器人能够同时运行，*Nav2*服务会以异步方式调用。每个机器人都会沿着分配给自己的轨迹行进。对于单个机器人，一项任务被细分为四个子任务：前往取货站、装载物品、前往配送站，以及在期望配送时间卸载物品。所有这些子任务都按顺序（同步）执行。但从团队整体来看，所有机器人会异步运行，同时执行各自的任务。

事件。 假设环境中发生了某些突发情况，例如某个机器人发生故障或部分道路关闭。此时将触发该事件。事件发生后，中央系统会检查所有活动任务的状态，并针对事件发生后的场景重新求解任务分配问题。导航服务随之更新，并根据更新后的分配结果将机器人分配给任务

5.1.3. 仿真结果

通过七项任务（T_0 至T_6）和负责这些任务的四台机器人（R_0 至R_3）进行仿真，以展示所提出的 PJITD 任务异构资源分配方法的运行过程。PJITD任务见表1。机器人初始位置分别为$R_0(0) = (2, -0.35)$、$R_1(0) = (1.6, 2.5)$、$R_2(0) = (-3.0, 1.2)$和$R_3(0) = (3.6, 1.5)$。为直观显示取货点和配送点的坐标，每项任务均以不同颜色标示，其中圆圈表示取货点，方形点表示配送点，并附有其交付时间。当前仿真时间显示在右上角。仿真视频见 https://www.youtube.com/watch?v=gNC0hG4CG2A。

每个 RViz2 窗口专用于一台机器人，用于显示其操作和状态。每台机器人的动作由所提出的算法决定，并显示在相应 RViz2 窗口的顶部。机器人计算出的局部导航轨迹以红色曲线显示。导航启动后，状态会指明机器人正在执行哪项任务，并显示其正在执行取货、配送、装载还是卸载子任务。每次操作完成后都会更新机器人的状态。分配给特定机器人的所有任务完成后，状态中会对此进行标示。

使用所提出的方法为给定的 PJITD任务分配任务。为此，机器人会计算给定任务的路径距离。使用 *ComputePathToPose* 函数（来自*Nav2* 软件包）计算场地内两点之间的可行路径。图 4 展示了仓库的 RViz2 截图，其中红色曲线也标出了两点之间的可行路径。通过计算沿此可行路径的线积分来求得距离。

利用式（2）和式（4），针对给定任务和机器人位置计算成本矩阵，结果分别如下式（7）和式（8）所示。仿真中使用的机器人均为同质机器人，并且都具备执行所有任务所需的全部技能集；因此，后续的成本矩阵对于全部四台机器人均相同，即$C^{s,0} = C^{s,1} = C^{s,2} = C^{s,3} = C^{s,\bullet}$。采用所提出的异构资源分配方法得到的解是一个决策变量，表示分配给各项任务的机器人位置。成本矩阵中的方框突出显示了任务分配解，上标表示被分配执行该任务的机器人。

$$\begin{bmatrix} \mathbf{C}^{f,0} \\ \mathbf{C}^{f,1} \\ \mathbf{C}^{f,2} \\ \mathbf{C}^{f,3} \end{bmatrix} = \begin{bmatrix} 5.66 & \boxed{6.95}^{0} & 11.73 & 11.44 & 10.61 & 2.84 & 4.25 \\ 6.13 & 9.67 & 22.72 & \boxed{8.56}^{1} & 7.77 & 3.11 & 2.60 \\ 10.54 & 9.78 & \boxed{6.75}^{2} & 12.77 & 12.01 & 6.69 & 7.21 \\ \boxed{3.94}^{3} & 9.28 & 13.41 & 10.30 & 9.88 & 3.25 & 3.58 \end{bmatrix} \quad (7)$$

$$\mathbf{C}^{s,\bullet} = \begin{bmatrix} \infty & 9.62 & 11.12 & 8.60 & 7.62 & 3.36 & 3.08 \\ \infty & \infty & 8.91 & 14.56 & 13.56 & 7.17 & 8.24 \\ \infty & \infty & \infty & 1.31 & 12.12 & \boxed{4.65}^{2} & 6.02 \\ \infty & \infty & \infty & \infty & \boxed{11.96}^{3} & 8.15 & 8.67 \\ \infty & \infty & \infty & \infty & \infty & 100 & 6.87 \\ \infty & \infty & \infty & \infty & \infty & \infty & \boxed{3.49}^{2} \end{bmatrix} \quad (8)$$

接下来，使用轨迹计算算法（算法 2）获取每台机器人的轨迹。获得的轨迹为：R_0 的轨迹是 $\mu_0 = \{T_1\}$，R_1 的轨迹是 $\mu_1 = \{T_3\}$，R_2 的轨迹是 $\mu_2 = \{T_2, T_5, T_6\}$，而$R_3$的轨迹是 $\mu_3 = \{T_0, T_4\}$。

任务执行从仿真时间 505 秒开始，因此所有交付时间均已更新。图 5 展示了$t = 505$秒时的初始场景，此时四台机器人均已分配各自的任务。图中还展示了各机器人在对应 RViz2 窗口中的操作。

每个机器人都必须执行分配给自己的任务，包括前往取件位置、装载物品、前往配送地点、等待至期望配送时间，然后卸载物品。每个机器人按顺序执行这些子任务，之后再执行随后分配的任务。

机器人 R_0 已被分配任务T_1。R_0 正前往取件位置P_1（以绿点表示）。与此同时，R_1 被分配了任务T_3，，因此正前往P_3取货。同样，R_2 正前往P_2，取货，而R_3 正前往P_0取货。图 5 展示了 t= 505 时的快照（505 秒时将任务交给算法），此时机器人正在寻找前往各自取件位置的可行轨迹。

机器人R_3 在$t = 513$ 秒到达其取件位置P_0，并在那里等待 1 秒以装载物品。接着，R_3 向其交付地点D_0 移动，并在$t = 531$ 秒到达D_0。任务T_0 的交付时间为 540，因此它还要等待 9 秒。与此同时，其他机器人R_0、R_1, 和R_2 已到达各自的取货地点，装载物品后正向各自的交付地点行进。在$t = 540$ 秒时，R_3 用 1 秒卸下物品。因此R_3 在 541 秒完成任务T_0。完成分配的第一个任务后，R_2 开始执行下一个任务T_4。为此，它前往P_4 取货。

R_0 于546秒到达交付地点D_1，而T_1 的交付时间为555秒，因此它等待了9秒。之后，它在1秒内卸下物品。由于R_0 只被分配了一项任务，完成任务后便进入休息模式。R_2 于$t = 551$ 到达第一个交付地点D_2，并等待至交付时间，即565秒，然后在接下来的1秒内卸下物品。因此，R_2 于$t = 566$完成任务T_2。完成分配的第一项任务T_2, 后，R_2 开始执行下一项任务，即任务T_5。与此同时，R_1 于$t = 556$到达第一个交付地点D_3，并在那里等到$t = 580$，随后在1秒内卸下物品。由于R_1 只被分配了一项任务，完成所有任务后便进入休息模式。

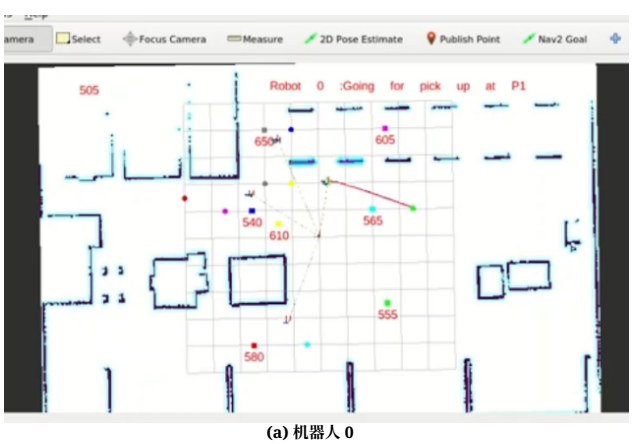
现在只有两台机器人，R_2 和R_3, 处于活动状态并正在执行任务。R_0 和R_1 处于休息模式。R_3 在 $t = 591$ 时到达分配给它的配送D_4，并等待至指定的 605 秒配送时间。$t = 605$之后，R_3 用 2 秒卸载物品，完成最后一项任务T_4，，并进入休息模式。R_2 在$t = 599$时到达配送地点D_5，等待至配送时间；在$t = 610$ 之后，它卸载物品并开始新的任务T_6。R_2 从P_6 取起物品，并在$t = 645$时到达D_6。等待 5 秒后，它卸下物品并完成最后一项任务。从仿真视频中可以观察到，机器人能够执行所有给定的 PJITD任务。

5.1.4. 计算复杂度

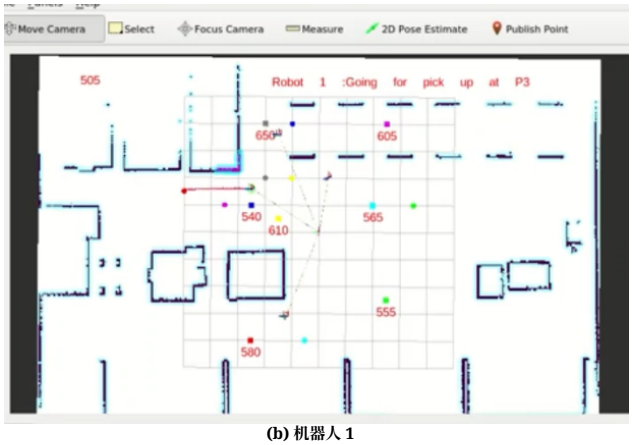
表 2 展示了所提出的方法用于 PJITD任务分配所需的计算时间。任务成本根据给定仓库的地图计算；成本矩阵的计算耗时较长。该算法的计算效率较高，所需时间几乎比成本矩阵计算少三个数量级。从表中可以观察到，随着任务数量增加，总计算时间也随之增加，且总计算时间与任务数量呈二次比例关系。

5.2. 资源利用率

所提出的方法需要动态数量的机器人来执行给定的时空任务。为此，我们分析了执行不同任务到达率下给定时空任务所需的资源（机器人）。



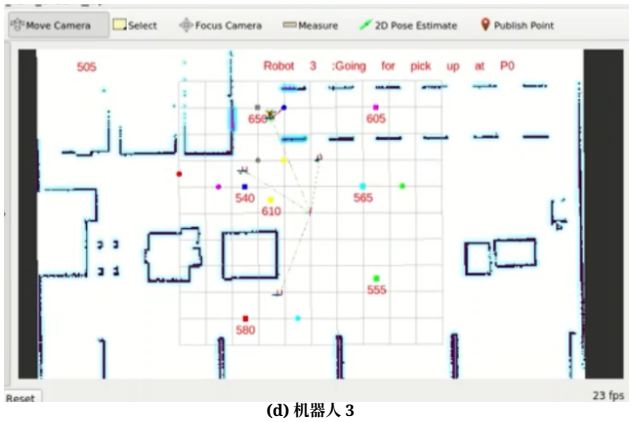
(a) 机器人 0



(b) 机器人 1



(c) 机器人 2



(d) 机器人 3

图 5. 四台机器人的初始分配 r 四台机器人。 每个子图都显示了机器人从其初始位置移动到第一个分配的取货点模拟开始时的位置 时刻 t = 505 秒。

任务数量	10	20	30	40	50	60	70	80	90	100	200	500
平均计算时间（毫秒）	0.0481	0.1283	0.2837	0.5294	0.8929	1.3869	2.0116	2.7918	3.7553	4.9230	31.5588	408.9476

T表 2：针对不同任务数量，求解优化问题（式 (6)）的平均计算时间

在不同任务到达率下的任务。此处，PJITD任务在随机的取货位置和配送位置生成，其交付时间服从泊松分布。给定团队规模（n）时，资源利用率因子（RUF）定义为规模为n 的机器人团队处于活动状态的总时间间隔与总仿真时间之比，定义如下：

$$\mathrm{RUF}(n) = \frac{\text{time interval with n robots}}{\text{Total simulation time}} \times 100 \qquad (9)$$

图 6 展示了不同时空任务到达率（Q）下的 RUF。当到达率为 0.1 时，最多需要 5 台机器人；其中，只有 1.15% 的情况下需要五台机器人，2.68% 的情况下需要四台机器人，48.35% 的情况下需要三台机器人，25.61% 的情况下需要两台机器人，而 22.19% 的情况下只需要一台机器人。从图 6 可以看出，随着到达率的提高，机器人数量也随之增加。

5.3. 可实施性研究

本文提出的方法用于计算执行给定任务所需的异构机器人数量及其任务分配。为展示所提出的、适用于异构 PJITD任务的非迭代且可在线计算的解决方案的可实现性，我们进行了实验室规模的实验。图 8 展示了实验中所用的场地。场地尺寸为 3.6米 × 2米，原点位于右上角。在场地内的 $\boldsymbol{O}_1 = (1.37, 0.96)$ 和 $\boldsymbol{O}_2 = (2.78, 0.96)$ 处放置了两个长方体障碍物。硬件实验使用了两台 TurtleBot3 Burger 机器人，它们配备 Raspberry Pi 3B+ 和搭载 ROS Melodic 的 Ubuntu 18.04 Server [1]

[1]由于 ROS2 的硬件支持仍在开发中，因此我们使用 ROS melodic 框架进行硬件实现

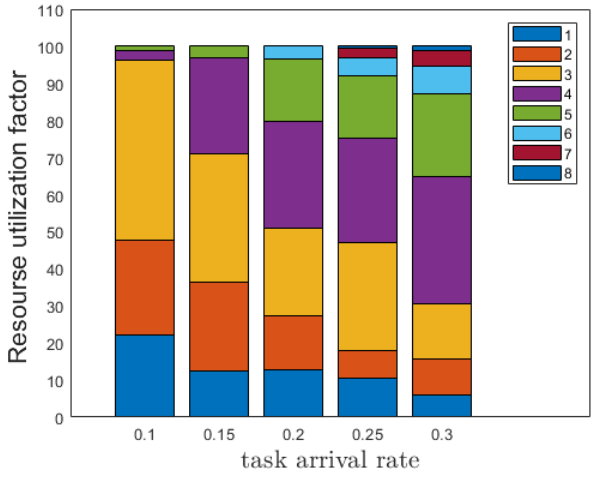


图 6. PJITD任务在不同到达率下的资源利用率

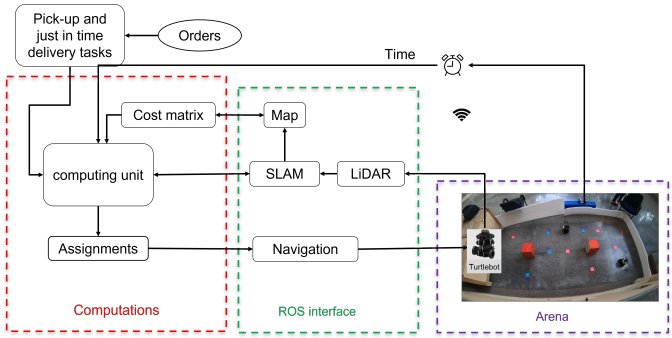


图 7. 硬件实验架构

实验。场地地图由 TurtleBot 在场地内遥操作时采集的 LDS-01 激光雷达传感器输出，通过同步定位与建图（SLAM）算法生成。对于 TurtleBot 的导航，每台 TurtleBot 均使用内置的 *Nav*包。

实验开始时，将 Turtlebot 机器人放置在场地中，并定义 PJITD任务。集中式算法利用地图和任务信息，计算分配给每台 Turtlebot 机器人的任务序列，并将任务序列传达给各台 Turtlebot 机器人。导航 ROS 节点用于按照正确顺序引导机器人前往目标航点，同时确保满足时间约束。

图 8 展示了设有取货点和配送点的场地。此外，场地右上角附近放置了一个实时时钟，以突出实验的时间特性。实验中的取货配送任务见表 3。机器人 1 和机器人 2 分别初始化于 (1.7,0.5) 和 (0.4,1.12)。Turtlebot3 的最高速度可达 0.22米/秒。然而，实际观察发现，对于这种大小的场地，由于加速和减速限制，实际速度为 0.12米/秒 。本实验中，机器人的速度设为 0.12米/秒。

现在，机器人计算给定任务的路径距离。*Nav* package 已用于计算可行路径

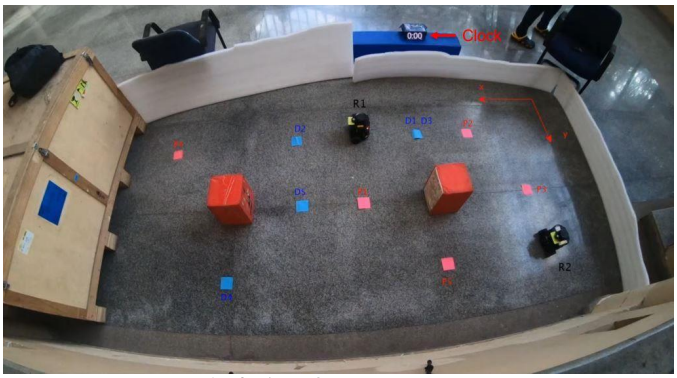


图 8. 硬件实验场地

	T_1	T_2	T_3	T_4	T_5
取件位置	(1.9,0.85)	(0.8,0.35)	(0.5,0.85)	(3.0,0.4)	(1.2,1.3)
配送位置	(1.2, 0.35)	(2.0,0.35)	(1.2,0.35)	(2.5,1.25)	(2.0,0.85)
配送 time	35	50	90	110	120

表3：取货与准时配送任务

从场地中的一个点到另一个点。沿这条可行路径的线积分用于计算距离。使用公式 (2) 和 (4) 计算得到的成本矩阵如下：

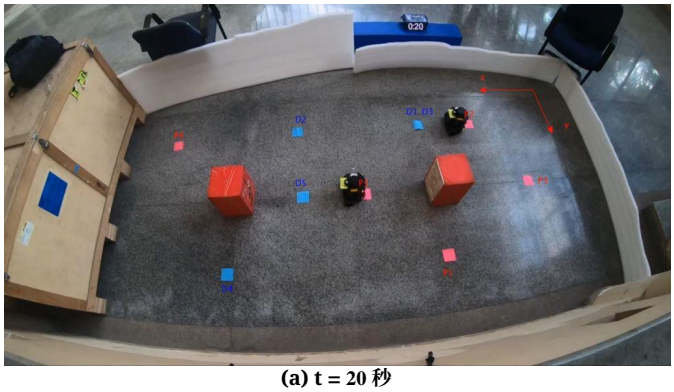
$$\begin{bmatrix}\mathbf{C}^{f,1}\\ \mathbf{C}^{f,2}\end{bmatrix}=\begin{bmatrix}\boxed{1.7278}^{1} & 3.0593 & 2.5471 & 4.3748 & 2.3141\\ 2.4100 & \boxed{3.0920}^{2} & 1.4627 & 5.9802 & 2.1106\end{bmatrix} \quad (10)$$

$$\mathbf{C}^{s,1}=\begin{bmatrix}\infty & 1000 & \boxed{1.9985}^{1} & 4.9158 & 2.6336\\ \infty & \infty & 2.8275 & 4.0760 & 2.6032\\ \infty & \infty & \infty & 1000 & \boxed{2.6336}^{1}\\ \infty & \infty & \infty & \infty & 1000\\ \infty & \infty & \infty & \infty & \infty\end{bmatrix} \quad (11)$$

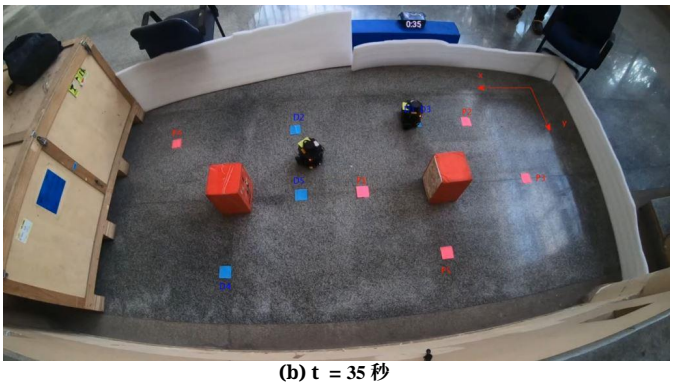
$$\mathbf{C}^{s,2}=\begin{bmatrix}\infty & 1000 & 1.9985 & 4.9158 & 2.6336\\ \infty & \infty & 2.8275 & \boxed{4.0760}^{2} & 2.6032\\ \infty & \infty & \infty & 1000 & 2.6336\\ \infty & \infty & \infty & \infty & 1000\\ \infty & \infty & \infty & \infty & \infty\end{bmatrix} \quad (12)$$

请注意，成本矩阵 $C^{s,1}$ 和$C^{s,2}$ 的对角线元素和下三角元素均设为 ∞，因为这些任务已在过去发生。对于沿时间正向不可行的分配，其值设为较大的 1000。对于元素 $C^{s,1}(1, 2)$，即机器人R_1 在执行任务T_2 后执行任务T_1，可用总时间为（$t_2^D - t_1^D = 15$），需要行驶的距离为：取货 0.48，配送 2.05786。需要行驶的总距离为 2.5378，至少需要 21.48秒。但可用时间（15秒）小于所需的最短时间，因此成本设为 1000。同理，成本 $C^{s,1}(3, 4)$ 和 $C^{s,2}(4, 5)$ 也设为 1000。

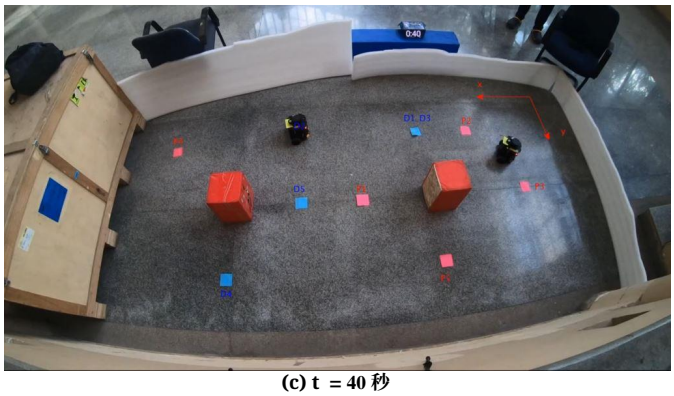
通过DREAM方法获得的解是一个决策变量，表示分配给该



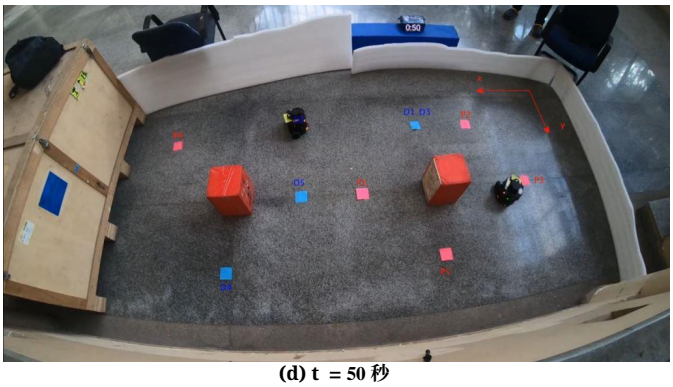
(a) t = 20 秒



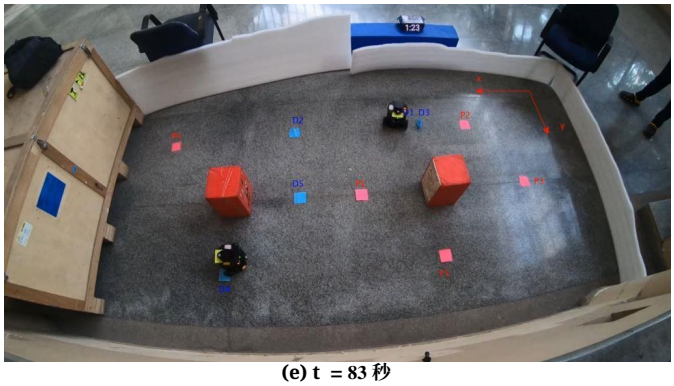
(b) t = 35 秒



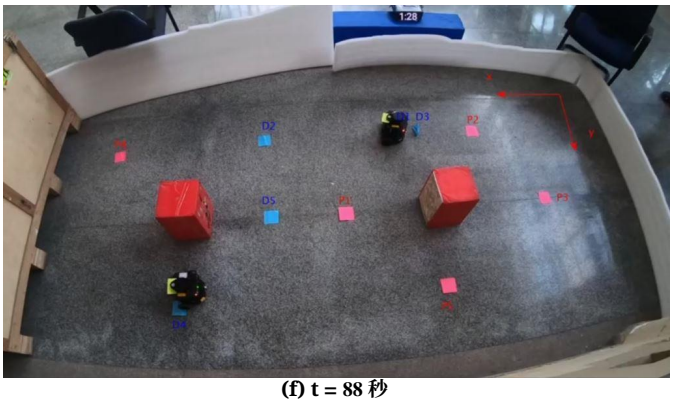
(c) t = 40 秒



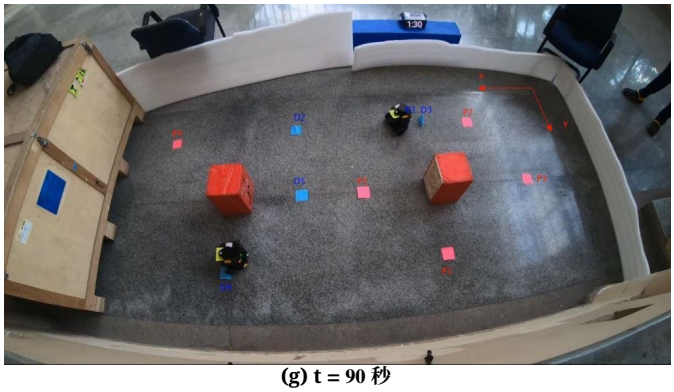
(d) t = 50 秒



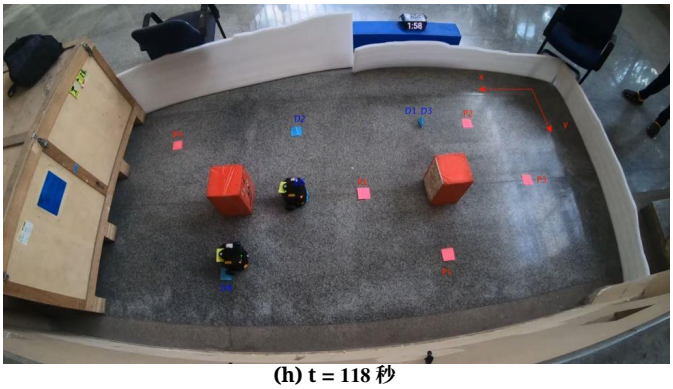
(e) t = 83 秒



(f) t = 88 秒



(g) t = 90 秒



(h) t = 118 秒

图 9. Turtlebot 执行 PJITD 任务时不同时间点的快照。a) t = 20，R_1 正在取货点 P_1 取货，R_2 正在取货点 P_2取货。b) t = 35，R_1 在 D_1完成配送。c) t = 40，R_2 到达 D_2 并等待，直至交付时间达到 50 秒。d) t = 50，R_2 在 D_2完成配送。e) t = 83，R_2 到达 D_4 并等待。f) t = 88，R_1 到达 D_3 并等待。g) t = 90，R_1 在 D_3完成配送。h) t = 118，R_1 到达 D_5。

任务。在成本矩阵中，任务分配方案以方框标出。接下来，使用轨迹生成算法计算每个机器人的轨迹。turtlebot 1 和 turtlebot 2 的轨迹分别为 $\mu_1 = \{T_1, T_3, T_5\}$ 和 $\mu_2 = \{T_2, T_4\}$。

硬件实验过程已录制成视频，可通过 https://www.youtube.com/watch?v=wuwL5-OVjyM 查看。下面结合图 9 说明该实验。R_1被分配了任务 T_1,T_3, T_5，R2 被分配了任务 T_2,T_4。机器人需要按顺序执行这些任务。根据分配结果，在 $t = 0$ 时，R_1 开始按照分配结果执行 T_1，R_2 则开始执行任务 T_2。R_1 导航至 P_1，R_2 导航至 P_2 取货。到达取件位置后，turtlebot 将机头朝向正 x 方向（视频中的左侧）。取货后，turtlebot 移动至相应的交付地点。turtlebot 1 在 $t = 35$ 秒时到达交付地点 D_1，并将机头对准正 x 方向。D_1 处的期望配送时间也是 35 秒；turtlebot 准时到达，任务 T_1 成功执行。

接下来，turtlebot R_1 开始执行其下一个任务 T_3，并向取件位置 P_3移动。与此同时，R_2 在 $t = 40$到达其交付地点 D_2；但 T_2 的期望配送时间为 50 秒，因此 R_2 在 D_2 等待 10 秒，并在 $t = 50$ 由 R_2 完成任务 T_2。随后，R_2 开始执行下一个任务 T_4，向取件位置 P_4移动。R_2 在 $t = 83$ 秒到达其交付地点 D_4。T_4 的配送时间为 110 秒，因此它在 D_4 等待接下来的 27 秒，然后完成任务 T_4。

与此同时，正在执行任务 R_1 的机器人在 $t = 88$秒到达交付地点 D_3，等待 2 秒直至期望配送时间，并在 $t = 90$秒完成任务 T_3。完成任务 T_3, 后，R_1 开始执行其下一个也是最后一个任务 T_5, : 它前往 P_5 取货，然后向交付地点D_5移动。R_1 在 $t = 119$秒到达D_5。R_1 等待 1 秒，并在期望配送时间 120秒完成最后一个任务。可以看到，所提出的算法将可行任务分配给了 turtlebot，而 turtlebot 按时执行这些任务。

6. 结论

本文提出了一种用于多任务分配的非迭代异构资源分配方法，以处理异构机器人执行在线取件与准时配送（PJITD）任务的问题。在PJITD任务中，交付时间受到期望时间的约束，因此，PJITD问题被表述为异构时空多任务分配（STMTA）问题。STMTA的成本函数经过修改，纳入了任务所需的行驶时间、操作时间和异构技能。所提出的异构资源分配方法以最少的机器人执行所有给定的异构PJITD任务，并且所得任务分配方案是最优的（使机器人团队行驶的总距离最小）。该方法的运行效果通过高保真仿真和硬件实现进行了验证。通过在线计算，可以将PJITD任务分配给机器人/智能体，并且同样通过高保真仿真和硬件实验进行了验证。未来的工作将探索将时空任务分配表述应用于各种因缺乏在线解决方案而被弃用的应用场景。此外，未来还将研究所提出的方法，以结合调度问题与时空任务分配问题。

致谢

作者谨此感谢印度科学研究所（IISc）班加罗尔 Nokia 网络机器人卓越中心以及 Nokia 企业社会责任基金的支持。

参考文献

[1] Y. Monden，《丰田生产方式：准时制的集成方法》，CRc Press，2011年。[2] S. Emde，N. Boysen，《针对混流装配线准时制供料的牵引列车最优路径规划与调度》，《欧洲运筹学杂志》217 (2) (2012) 287–299。[3] N. Boysen，R. De Koster，F. Weidinger，《电子商务时代的仓储：综述》，《欧洲运筹学杂志》277 (2) (2019) 396–411。[4] P. R. Wurman，R. D'Andrea，M. Mountz，《在仓库中协调数百辆协作型自主车辆》，《AI magazine》29 (1) (2008) 9–9。[5] J. Fischer，C. Lieberoth-Leden，J. Fottner，B. Vogel-Heuser，《物流领域多智能体系统的设计、应用与评估》，《IEEE自动化科学与工程汇刊》17 (3) (2020) 1283–1296。[6] ´I. R. da Costa Barros，T. P. Nascimento，《机器人移动履约系统：近期进展与研究机遇综述》，《机器人与自主系统》137 (2021) 103729。[7] A. Camisa，A. Testa，G. Notarstefano，《通过分布式资源分配实现多机器人取送》，《IEEE机器人学汇刊》(2022) 1–13doi:10.1109/TRO.2022.3216801。[8] Z. Chen，J. Alonso-Mora，X. Bai，D. D. Harabor，P. J. Stuckey，《容量受限的多智能体取送中的任务分配与路径规划集成》，《IEEE机器人与自动化快报》6 (3) (2021) 5816–5823。[9] A. Ham，《机器人移动履约中心中的基于无人机的物料转运系统》，《IEEE自动化科学与工程汇刊》17 (2) (2019) 957–965。[10] A. Bolu，¨O. Korc¸ak，《面向多机器人智能仓库的自适应任务规划》，《IEEE Access》9 (2021) 27346–27358。[11] X. Bai，M. Cao，W. Yan，S. S. Ge，《异构车辆有序约束包裹配送的高效路径规划》，《IEEE自动化科学与工程汇刊》17 (1) (2019) 248–260。[12] E. Nunes，M. Manner，H. Mitiche，M. Gini，《具有时间与顺序约束的任务分配问题分类法》，《机器人与自主系统》90 (2017) 55–70。[13] D. M. Vu，M. Hewitt，N. Boland，M. Savelsbergh，《求解时间依赖的带时间窗旅行商问题》，《Optimization Online》6640 (2018) 785。[14] L. Jiang，M. Dhiaf，J. Dong，C. Liang，S. Zhao，《网络购物最后一公里配送中的带时间窗旅行商问题》，《International Journal of Production Research》58 (16) (2020) 5077–5088。[15] D. Jiang，X. Li，《网络零售中产生的带时间窗与同步约束的订单履约问题》，《International Journal of Production Research》59 (4) (2021) 1187–1215。[16] J. Gao，S. Jia，J. S. Mitchell，L. Zhao，《带时间窗旅行商问题和奖赏收集旅行商问题的近似算法》，载于：《机器人学算法基础 XII》，Springer，2020年，第560–575页。[17] F. Semiz，F. Polat，《无人机区域覆盖问题求解：带时间窗的车辆路径问题的一种变体》，《机器人与自主系统》126 (2020) 103435。

[18] D. M. Vu，M. Hewitt，N. Boland，M. Savelsbergh，《通过动态离散化发现求解带时间窗的时间依赖旅行商问题》，《Transportation Science》54 (3) (2020) 703–720。[19] G. Budak，X. Chen，《配送作业中旅行商问题时间窗大小的评估》，《Complex & Intelligent Systems》6 (3) (2020) 681–695。[20] S. Velhal，S. Sundaram，N. Sundararajan，《面向周界防御问题的分布式多任务分配动态资源分配方法》，《IEEE航空航天与电子系统汇刊》58 (4) (2022) 3313–3325。[21] Z. Lyu，P. Lin，D. Guo，G. Q. Huang，《从零库存准时制生产迈向零仓储智能制造》，《机器人与计算机集成制造》64 (2020) 101932。[22] K. Nishida，T. Nishi，H. Kaname，K. Kumagai，T. Higashi，《多辆自动导引车的准时制路径规划与调度》，载于：2019年IEEE系统、人类与控制论国际会议（SMC），IEEE，2019年，第841–846页。[23] K. Nishida，T. Nishi，《准时交付中自动导引车无冲突路径规划的动态优化》，《IEEE自动化科学与工程汇刊》(2022)。[24] S. Chopra，M. Egerstedt，《异构多机器人路径规划》，载于：2014年美国控制会议，IEEE，2014年，第5390–5395页。[25] S. Chopra，M. Egerstedt，《时空多机器人路径规划》，《Automatica》60 (2015) 173–181。[26] S. Velhal，K. K. VS，S. Sundaram，《基于非迭代时空多任务分配的音乐演奏机器人无碰撞轨迹》，arXiv预印本 arXiv:2210.07653 (2022)。[27] C. Baier，J.-P. Katoen，《模型检测原理》，MIT Press，2008年。[28] A. A. Leao，F. M. Toledo，J. F. Oliveira，M. A. Carravilla，R. Alvarez- Vald´es，《不规则装箱问题：数学模型综述》，《欧洲运筹学杂志》282 (3) (2020) 803–822。