

# Multi-Robot Task Allocation: A Survey

## 1. 分配类型

### 1.1 基本概念

这部分内容来源于文献 [1]。

#### 1.1.1 任务和任务分解

##### 分解 (decomposition) 和可分解性 (decomposability)

如果一个任务  $t$  可以表示成一组子任务  $\sigma_i$  的集合，且这些子任务满足一些特定的组合  $\rho_i$ ，则认为  $t$  是可分解的。满足任务  $t$  的子任务组合可以表示成一组关系的集合（比如子任务间的关系和规则）。( $\sigma_i, \rho_i$ ) 称为任务  $t$  的一个分解。

##### 多可分解性 (multiple decomposability)

如果一个任务  $t$  有超过一个分解方案，则称  $t$  是多可分解的。

##### 元素任务 (elemental task)

一个元素任务是一个不可分解的任务。

##### 可分解的简单任务 (decomposable simple task)

如果一个任务可被分解成元素级任务或可分解的简单子任务，则称它是可分解的简单任务。

##### 简单任务 (simple task)

一个简单任务是一个元素级任务或可分解的简单任务。

##### 组合任务 (compound task)

如果一个任务  $t$  可被分解成一组简单子任务或组合子任务，并且只有一种固定的完全分解方案，则称  $t$  是一个组合任务。

##### 复杂任务 (complex task)

如果一个任务  $t$  是多可分解的，且每个分解是一组多可分配的 (multi-allocatable) 子任务，每个子任务可能是简单、组合或复杂的，则称  $t$  是一个复杂任务。

**讨论：**组合任务和复杂任务的关键区别在于，组合任务的最优任务分解可以在任务分配之间确定，而复杂任务在任务分配之前不知道哪个分解是最优的，任务分解和分配需要同时进行以寻找最优方案。

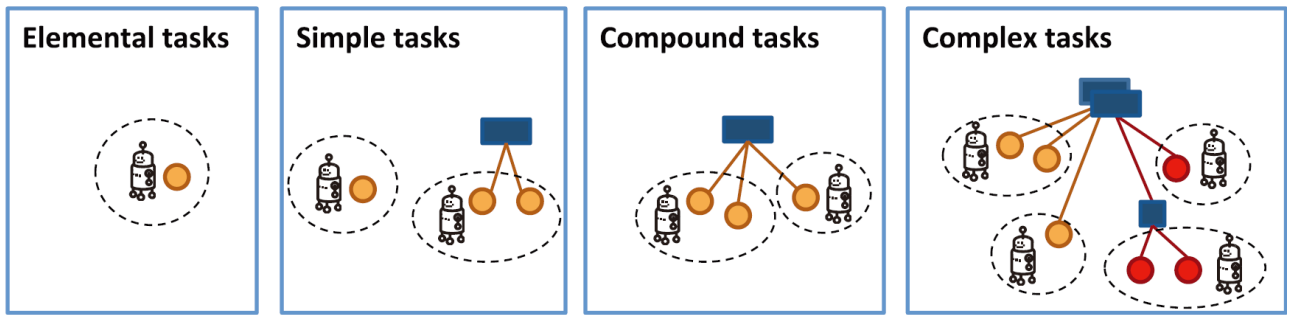


图 1 任务分类示意图。虚线圆圈表示可能有效的任务分配，实心圆圈表示元素任务，实心方块可分解任务。

最右边的图中叠加的树表示示例任务的多种可能分解方式，来自文献[1]

### 1.1.2 约束

在任务分配问题中，约束（constraints）可能是一个任意的函数，它限制了待求解问题的可行方案空间。比如，1）机器人有能力约束，意味着机器人可最多执行多少任务或只能执行哪些任务；2）同步约束，意味着两个任务必须被同步执行；3）非交叠约束，意味着两个任务不能被同步执行；4）顺序约束，意味着一个任务必须在另一个任务之前被执行。

### 1.1.3 任务分解和任务间约束的关系

举个例子，对于组合任务，任务分配可在任务分解之后执行。一个组合任务会被分解成多个简单任务，为等价于原来的组合任务，这些简单任务需要满足一些同步或顺序约束。这些约束能够让机器人的工作协调起来。因此，任务分解和任务间约束有很紧密的联系。

### 1.1.4 效用

任务分配作为一个优化问题，它总是会优化某些目标，这些目标就可以被描述为效用（utility）函数。给定一个机器人  $r$  和一个任务  $t$ ，如果  $r$  能够执行  $t$ ，则定义  $Q_r$  和  $C_r$  分别为机器人完成任务的质量（quality）和代价（cost），这样机器人的效用可表示为

$$U_r = \begin{cases} Q_r - C_r, & \text{if } r \text{ is capable of executing } t \\ -\infty, & \text{otherwise} \end{cases}$$

对于有些问题，一个机器人执行一个任务的效用独立于执行其他任务，而有任务则不是这样。举个例子，现在环境中分散着一定数量的 treasures，并有着是一群初始位置不同的机器人，机器人需要收集每个 treasure 并将其带回初始位置。机器人能力完全一致，且一次只能收集一个 treasure。在这样一个问题中，机器人必须回到初始位置后才能收集下一个 treasure，因此一个机器人完成一个任务（即带回一个 treasure）的效用独立于其它机器人-任务效用。如果一个机器人一次可收集多个 treasures，某些 treasures 的位置比较接近，那机器

人可以收集完一个 **treasure** 后，紧接着收集下一个，而不需要立刻回到初始位置。因此一个机器人完成一个任务的效用就不独立于其它机器人-任务效用了。

现在将效用定义扩展到多机器人和多任务上，令  $\mathcal{R}$  表示一组机器人中的一个子集， $\mathcal{T}$  表示一组任务中的一个子集，则这一小组机器人完成这一小组任务的效用为

$$U_{\mathcal{RT}} = \begin{cases} Q_{\mathcal{RT}} - C_{\mathcal{RT}}, & \text{if subteam } \mathcal{R} \text{ is capable of executing task subset } \mathcal{T} \\ -\infty, & \text{otherwise} \end{cases}$$

同时，定义一个机器人  $r \in \mathcal{R}$  完成一个任务  $t \in \mathcal{T}$  的有效效用（effective utility）为  ${}^eU_{rt}^{RT}$ ，则

$$U_{\mathcal{RT}} = \sum_{r \in \mathcal{R}} \sum_{t \in \mathcal{T}} {}^eU_{rt}^{RT}$$

如果一个分配问题中涉及的机器人效用是独立的，则有  ${}^eU_{rt}^{RT} = U_{rt}$ ；如果效用是相互影响的，则有  ${}^eU_{rt}^{RT} \neq U_{rt}$ 。特别地，如果这一组机器人子集和这组任务子集存在协同关系，那么有  ${}^eU_{rt}^{RT} > U_{rt}$ ，即

$$U_{\mathcal{RT}} > \sum_{r \in \mathcal{R}} \sum_{t \in \mathcal{T}} U_{rt}$$

## 1.2 问题分类

目前，多机器人任务分配（MRTA）问题存在两种分类方法，第一种是 Gerkey and Matari’c 分类，2004 年被总结 [2]；第二种是 iTax 分类，2013 年被总结 [1]。

### 1.2.1 Gerkey and Matari’c 分类

这种分类方式考虑了三个维度，第一个维度是机器人类型，机器人分为单任务（ST）机器人和多任务（MT）机器人，单任务机器人在一个时刻只能执行一个任务，多任务机器人同时可执行多个任务。第二个维度是任务类型，任务分为单机器人（SR）任务和多机器人（MR）任务，单机器人任务由一个机器人即可完成，多机器人任务需要多个机器人同时才能完成。第三个维度是时间，分为即时分配（IA）和时间扩展分配（TA），即时分配不会考虑未来的计划，而时间扩展分配会同时考虑当前和未来的计划，也即一个机器人会被分配多个任务。

#### ST-SR-IA 问题

这类问题是最简单的，它的特点是一个机器人只能执行一个任务，一个任务只能被分配给一个机器人。最优分配问题就属于这一类型，目前已有统一的数学模型来描述此类问题，也有成熟的算法来解决此类问题，比如 Hungarian 算法（集中式方法，求解最优分配的时间在  $O(mn^2)$  量级，其中  $m$  表示机器人数量， $n$  表示任务数量）、拍卖算法（集中式或分布式算

法，分布式情况下，求解最优分配的时间正比于最大效用，反比于最小竞价增量）。这类问题的求解是 polynomial time 的。

ST-SR-IA 问题有两类常见的变体，第一类是迭代式分配（Iterated Assignment），第二类是在线分配（Online Assignment）。

迭代式分配指的是迭代进行多次 ST-SR-IA 分配，有两个常见的具体问题。第一个是协同多目标跟踪。在该问题中，一群机器人需要观察一些不可预测的移动目标，当接收到新的传感器输入（如相机图像）和效用估计（如感知到的离目标的距离）时，整个系统需要确定哪个机器人跟踪哪个目标。第二个是机器人足球。在该问题中，许多机器人的身份是可交换的，允许任何一个机器人发挥任何一个角色。它们需要周期性的评估场上态势，进行任务分配。迭代式分配问题常见解法是 BLE（broadcast of local eligibility）算法，它分为三步：第一步，如果机器人  $i$  没有被分配任务，则找到该机器人所有的任务对  $(i, j)$ ，找出最高效用的机器人-任务对；如果分配了，则跳过；第二步，将最高效用对应的任务  $j$  分配给机器人  $i$ ；第三步，返回第一步。

在线分配指的是一组任务不会全部被机器人群事先知道，换句话说，在机器人执行任务期间可能会出现一个新任务。此时，一个已经被分配任务的机器人不能被重新分配了（如果能被重新分配，又变成迭代式分配问题了）。在线分配问题的常见解法是 MURDOCH 算法，它的核心思想是当一个新任务出现时，将它分配给一个最合适的可用的机器人。

### ST-SR-TA 问题

这类问题会为每个机器人建立一个时间扩展的任务 schedule，即一个任务列表，求解上是 NP-hard 的。一类处理 ST-SR-TA 问题的方法是忽略时间扩展属性，将它估计成一个 ST-SR-IA 问题，具体来说是一个在线分配问题。比如，现在有  $m$  个机器人和  $n$  个任务，满足  $n > m$ ，则估计算法分为两步：第一步，求解一个初始的  $m \times n$  最优分配问题；第二步，以在线的方式，用贪婪算法将剩余的任务分配给机器人，此时所有机器人都是可用的。

### ST-MR-IA 问题

这类问题指的就是联盟形成（coalition formation）问题，即将一组机器人分成一些 task-specific 的联盟。后者可以用集合分区（set partition）方法来求解，但求解上是 NP-hard 的。目前已有不少工作研究了这类问题，如常见的机组人员排班问题。

集合分区问题（SPP）可定义为：给定一个有限集合  $E$ ，一组可接受的子集合  $F$  和一个效用函数  $u: F \rightarrow \mathbb{R}$ ，找到  $F$  中的一组元素  $X$  能够最大化效用函数，且  $X$  是  $E$  的一个分区。

根据集合分区问题，ST-MR-IA 问题可被看成： $E$  是机器人集合， $F$  是所有可行的联盟-任务对， $u$  是每一个联盟-任务对的估计效用。

## ST-MR-TA 问题

这类问题同时包括联盟形成和调度，求解上是 NP-hard 的。举个例子，现在在中心站有一些不同尺寸的包裹需要被配送到不同的目的地，目的地和包裹数量都事先知道。现在要求一群机器人建立一个配送 schedule，确保每个尺寸大小的包裹都有一群合适大小的机器人配送。如果要获得这样一个分配问题的最优解，那得分析所有可行的联盟-任务对。如果联盟给定，每个任务只能分配给一个联盟，那就变成了一个多处理调度（multi-processor scheduling）问题：

$$MPTm \parallel \sum w_j C_j$$

即使  $m = 2$ ，即两个 processor，这样一个问题也是 NP-hard 的。

目前有两个思路解决这类问题。第一个思路是忽略 ST-MR-TA 问题的时间扩展属性，将它估计成一个 ST-MR-IA 问题，具体来说是一个迭代式 ST-MR-IA 问题。这样就可用贪婪估计算法来求解，只是估计的质量受很多因素影响，不好量化。第二个思路是采用一种 leader-based 机制，leader 会动态地形成联盟并为每个联盟建立一个 schedule，有一个工作便是这么做的。

## MT-SR-IA 和 MT-SR-TA 问题

这两类问题很少遇到，因为它要求每个机器人可同时执行多个任务。但在数学上讲，求解 MT-SR-IA 问题等价于求解 ST-MR-IA 问题，只需将任务和机器人的身份交换即可。类似地，求解 MT-SR-TA 问题等价于求解 ST-MR-TA 问题。两类问题都是 NP-hard 的。

## MT-MR-IA 问题

这类问题的目标是确定机器人联盟来执行每个任务，但一个机器人可能被同时分配到多个联盟里。举个例子，现在有一群机器人在办公楼里执行监视任务，每个机器人可以连续监视这栋楼的固定部分，由于资源有限，每个机器人只能同时监视一定数量的环境事件（如可疑人、烟雾）。现在有一些事件需要被监视，它们大致的位置是事先知道的，问题是确定哪个机器人去监视哪个事件呢？该问题可由集合覆盖（set covering）方法来求解，求解上是 NP-hard 的。

集合覆盖问题（SCP）可定义为：给定一个有限集合  $E$ ，一组可接受的子集合  $F$  和一个效用函数  $c: F \rightarrow \mathbb{R}_+$ ，找到  $F$  中的一组元素  $X$  能够最小化代价函数，且  $X$  是  $E$  的一个覆盖。这里的覆盖指的是  $X$  的并集等于  $E$ 。尽管 SCP 表面上与 SPP 相似，但实际上它是一个“远亲”，SCP 的解空间受到的限制要少得多。

根据集合覆盖问题，MT-MR-IA 问题可被看成： $E$  是机器人集合， $F$  是所有可行的（可能有重合）联盟-任务对， $c$  是每一个联盟-任务对的估计代价。现在很少有工作在 MRTA 问题中研究了集合覆盖问题。

### MT-MR-TA 问题

这类问题在 MT-MR-IA 问题基础上考虑了调度。举个例子，在前一类问题的监视任务基础上，考虑某些事件不需要被即时地或者连续地监视，而是按照事先定义好的计划监视，比如每隔一个小时监视某一扇门的情况。这样的问题就属于 MT-MR-TA 类分配问题，它属于多处理器（multi-processor）多目的机（multi-purpose machine）调度问题：

$$MPTmMPMn \parallel \sum w_j C_j$$

这类问题是强 NP-hard 的。目前没有发现有解决这类问题的研究工作。

### 讨论：

Gerkey and Mataric 分类认为三个维度是独立的，因此组合起来有八种类型。明显不足是没有考虑任务间相联系的效用和约束。比如在多旅行商问题（m-TSP）中，机器人需要到达多个地点执行空间上分布的任务，效用函数是与路线代价相关的。此时，那些分布相近的任务表现出协同性，机器人执行这些任务的总效用不等于单独执行每个任务的效用和，而是前者大于后者。

### 1.2.2 iTax 分类

这种分类方式考虑了任务间有关联的效用和约束。MRTA 问题涉及了任务分解、任务分配及调度，而任务间效用的依赖性决定了一个 MRTA 问题的耦合程度。iTax 分类考虑了一个维度，即依赖类型。它包括以下四类：

#### 无依赖（No Dependencies, ND）

一些简单或组合任务有着独立的机器人-任务效用，此时，一个机器人完成一个任务的有效效用不依赖于其它机器人或任务。在这些问题中，任务分解和任务分配是解耦的，而且不存在调度优化问题。这类问题可被描述成线性分配问题，求解上是 polynomial time 的。

#### 调度内依赖（In-schedule Dependencies, ID）

一些简单或组合任务有着 intra-schedule 依赖的机器人-任务效用，此时，一个机器人完成一个任务的有效效用取决于该机器人需要执行的其他任务。一个机器人的 schedule 中的任务存在约束。在这些问题中，任务分解和任务分配是解耦的，一个机器人需要考虑调度优化问题，但它的调度优化问题和其他机器人是解耦的。这类问题求解上是 NP-hard 的。

## 调度间依赖（Cross-schedule Dependencies, XD）

一些简单或组合任务有着 inter-schedule 依赖的机器人-任务效用，此时，一个机器人完成一个任务的有效效用不仅取决于该机器人自己的 schedule，还取决于其他机器人的 schedule。多个机器人的 schedule 中的任务可能存在约束。在这类问题中，任务分解和任务分配是解耦的，一个机器人需要考虑调度优化问题，且它的调度优化问题不能与其他机器人解耦。这类问题求解上是 NP-hard 的。

## 复杂依赖（Complex Dependencies, CD）

一些复杂任务有着 inter-schedule 依赖的机器人-任务效用，此时，一个机器人完成一个任务的有效效用不仅取决于该机器人自己的 schedule，还取决于其他机器人的 schedule。不同机器人的 schedule 中的任务可能存在约束。在这类问题中，任务分解和任务分配是耦合的，最优任务分解必须和任务分配同时确定。一个机器人的调度优化问题不能与其他机器人解耦。这类问题求解上是 NP-hard 的。

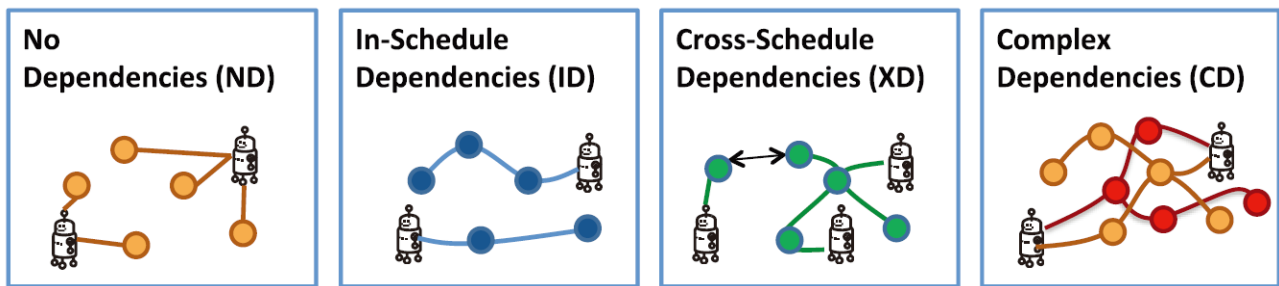


图 2 iTax 分类示意图。实心圆圈表示任务，实线表示机器人路线，任务之间的箭头表示约束。第四个图中有阴影的路线表示多个任务分解，来自文献[1]。

### 1.2.3 组合分类

#### 1) ND 类问题

在这一类问题中，一个机器人完成一个任务的有效效用只依赖于该机器人和该任务。常见的情况是效用函数只与机器人能力相关。这类问题一般包括单任务机器人（ST）和单机器人任务（SR），而不包括：（1）多任务机器人（MT），MT 属性问题属于 ND 类问题的一点条件就是一个机器人可同时执行的任务数量不受限制，显然这是不切实际的。因此，MT 属性常出现在 ID 类问题中；（2）多机器人任务（MR），因为 MR 属性涉及到多机器人合作，此时一个机器人完成一个任务的有效效用就取决于其他机器人。

#### a) ND[ST-SR-IA]问题

#### 数学模型

这类问题建立了独立的单机器人任务与独立的单任务机器人之间的 one-to-one 分配，它可表示为一个线性分配问题，其数学形式为

$$\begin{aligned} & \text{Max} \sum_{i \in N} \sum_{j \in M} u_{ij} x_{ij} \\ & \text{s.t.}: \\ & \sum_{i \in N} x_{ij} = 1, \quad \forall i \in N \\ & \sum_{j \in M} x_{ij} = 1, \quad \forall j \in M \end{aligned}$$

其中， $N$  表示机器人数量， $M$  表示任务数量， $x_{ij}$  是一个二值函数，表示任务  $j$  是否分配给机器人  $i$ ，如果分配，则为 1，否之则为 0。 $u_{ij}$  表示任务  $j$  分配给机器人  $i$  的效用值。

为获得一个可行解，机器人的数量必须等于任务的数量。如果不相等，可以将多余的机器人或任务认为是“dummy”属性的，为它们设置一个很小的效用值。另外。效用值可以根据机器人-任务约束来设置，比如一个机器人不能执行某一个任务，则可为该机器人-任务对设置一个很小的效用值。

### 求解方法

线性分配问题求解上是 polynomial time 的，过去有大量工作研究了这类问题的解法，如 Hungarian 算法，potential fields，拍卖算法。

#### b) ND[ST-SR-TA]问题

在这类问题中，每个机器人会被分配多个任务，最后拥有一个任务列表（即一个任务 schedule）。出现这种情况的原因可能是任务数大于机器人数量，或者某些机器人直接不执行任务。因为没有效用依赖，所以分配给一个机器人的任务的顺序不影响总的效用。举个例子，再次讨论前文的机器人收集 treasures 的问题，一个机器人收集一个 treasure 后就需要带回初始位置，且执行任务没有时间限制。那这样一个 treasure 收集问题就是一个 ND[ST-SR-TA] 问题。

### 数学模型

因为机器人-任务效用与其他机器人和任务是独立的，所以这类问题也是一个线性分配问题。

#### 2) ID 类问题

在这类问题中，一个机器人完成一个任务的效用取决于分配给该机器人的其他任务。这类任务经常出现在以下两种情况：（1）出现在时间扩展（TA）分配问题中，因为 TA 分配的效用函数经常涉及路线代价或任务完成时间。而且机器人-任务效用主要取决于那些排在机器人 schedule 中靠前的任务（靠前的任务的顺序合理了才有利于后面的任务被完成）。（2）出现在多任务机器人（MT）分配问题中，一个机器人的资源或能力限制了这个机器人可以

同时执行的任务数量，这会影响任务执行质量或时间。比如一个机器人不能同时到达 A 点和 B 点，但能在导航到 A 点的同时监视某一个位置。

这类问题肯定不包括 ST-SR-IA 类分配问题，因为后者要求机器人不能被分配多个任务，也自然没有 ID 依赖了。也不包括 MR 类分配问题，因为 MR 问题存在多机器人合作，因此通常都存在 XD 依赖。相反，ID 类问题主要包括 ST-SR-TA，MT-SR-IA，和 MT-SR-TA 类分配问题。通用分配问题、多机调度问题、多旅行商问题、车辆路线问题基本都可归结为这些类别。

#### a) ID[ST-SR-TA]问题

##### 数学模型

这类问题可表示为一个通用分配问题，每个机器人可以被分配一个以上的任务，同时存在在一个约束（通常是预算或时间约束），该约束限制了机器人可被分配的任务数量，其数学形式为

$$\begin{aligned} & \text{Max} \sum_{i \in N} \sum_{j \in M} u_{ij} x_{ij} \\ & \text{s.t. :} \\ & \sum_{i \in N} x_{ij} \leq 1, \quad \forall j \in M \\ & \sum_{j \in M} t_{ij} x_{ij} = T_i, \quad \forall i \in N \end{aligned}$$

其中， $N$  表示机器人数量， $M$  表示任务数量， $x_{ij}$  是一个二值函数，表示任务  $j$  是否分配给机器人  $i$ ，如果分配，则为 1，否之则为 0。 $u_{ij}$  表示任务  $j$  分配给机器人  $i$  的效用值。 $t_{ij}$  表示机器人  $i$  执行任务  $j$  的时间。 $T_i$  表示机器人  $i$  的任务执行时间限制。

举个例子，再次讨论前文的机器人收集 treasures 的问题。假设一个机器人收集一个 treasure 后必须回到初始位置，才能收集下一个 treasure。同时假设机器人执行任务有一个时间限制。因为机器人执行任务的时间可由速度和到 treasure 的距离确定。所以，一个机器人是否执行一个任务取决于任务时间限制和它的 schedule 里面的其他任务。这样一个问题就属于 ID 类问题。

##### 求解方法

通用分配问题求解上是 NP-hard 的，过去有些工作研究了这类问题的解法，比如 GRAMMPS 算法（集中式算法，用暴力和随机搜索算法求解了旅行商和多旅行商问题）、auction 或 market-based 算法（集中式或分布式算法）

#### b) ID[MT-SR-IA]问题

##### 数学模型

这类问题只考虑即时信息，每个任务只需一个机器人即可完成，但一个机器人可同时执行多个任务。理论上，这类问题可被表示成通用分配问题，同时存在一个约束（通常是能力约束，而不是时间约束，一般的能力约束指的是一个机器人执行的任务数量是有限的）。

### 求解方法

目前，很少有 MRTA 工作研究这一类问题。

#### c) ID[MT-SR-TA]问题

### 数学模型

这类问题考虑将一组单机器人任务分配给多任务机器人，同时考虑 TA 属性，目前没有发现有合适的数学模型表示这类问题。但有一些车辆路线问题的变体属于这一类。举个例子，取送货问题（pick-up and delivery problems, PDPs）和打车问题（dial-a-ride problems, DARPs），在这两个问题中，车辆需要去取货点（上车点）取货（接人），然后在放货点（下车点）卸货（送人）。车辆可以根据自己的能力，在这个过程中同时运送多个货物（乘客）。因此，这两类问题属于 ID[MT-SR-TA]问题。

### 求解方法

目前，很少有 MRTA 工作研究这一类问题。

#### 3) XD 类问题

在这类问题中，机器人完成一个任务的效用取决于机器人自己的 schedule 和其他机器人的 schedule。这类任务经常出现在以下两种情况：（1）两个或多个单机器人任务存在 inter-task 约束，如 proximity、顺序或同步约束，这些任务可被分配给不同的机器人；（2）存在一些必须多个机器人才能完成的任务，涉及联盟形成问题。

#### a) XD [ST-SR-IA], XD [ST-SR-TA], XD[MT-SR-IA]和 XD [MT-SR-TA]问题

这类问题最简单的情况是 XD [ST-SR-IA]。举个例子，再次讨论前文的机器人收集 treasures 的问题。现在机器人收集 treasures 后不是带回自己的初始位置，而是将其放到两个存储箱中的一个，每个 treasure 具体放在哪个存储箱根据最小化目标函数确定，目标函数可能是走过的距离。我们现在指定两个位置接近的特定 treasures 必须最终放在同一个存储箱中，且它们可能被不同的机器人收集到，那这就会造成了一个 XD 依赖的分配问题，这个 cross-schedule 依赖联系了这两个机器人的动作。

相似的 XD 依赖可能出现在计算机器人的任务 schedule 时，此时存在 inter-task 约束。举个例子，再次讨论前文的机器人收集 treasures 的问题。假设一些 treasures 位置很接近，以至于它们堆叠在一起，那么堆叠在最上面的 treasures 将需要先于下面的 treasures 移出来。由

于这些任务（收集一个 treasure）中的每一个都可能被分配给不同的机器人，因此两个任务之间的顺序约束可能会导致 XD 依赖。

## 数学模型

目前很少有数学模型描述 XD-SR 类型的分配问题。这里只讨论最简单的即时分配情况，此时可用进一步通用分配问题表示，同时存在一个联合的（而不是单个机器人的）约束，数学形式为

$$\begin{aligned} & \text{Max} \sum_{i \in N} \sum_{j \in M} u_{ij} x_{ij} \\ & \text{s.t.:} \\ & \sum_{i \in N} x_{ij} \leq 1, \quad \forall j \in M \\ & \sum_{i \in N} \sum_{j \in M} t_{ij} x_{ij} = T_k, \quad \forall k \in K \end{aligned}$$

其中， $N$  表示机器人数量， $M$  表示任务数量， $K$  是一组联合约束， $x_{ij}$  是一个二值函数，表示任务  $j$  是否分配给机器人  $i$ ，如果分配，则为 1，否之则为 0。 $u_{ij}$  表示任务  $j$  分配给机器人  $i$  的效用值。 $t_{ij}$  表示机器人  $i$  执行任务  $j$  的时间。 $T_k$  表示机器人  $i$  的任务执行时间限制。

## 求解方案

目前，只有少数 MRTA 工作研究了 XD 依赖。如 M+ 算法（支持迭代式的即时分配和顺序约束）。

b) XD [ST-MR-IA], XD [ST-MR-TA], XD [MT-MR-IA] 和 XD [MT-MR-TA] 问题

## 数学模型

XD [ST-MR-IA]，这类问题考虑即时分配情况下的联盟形成问题，每个机器人在一个时刻只能执行一个任务，即只能作为一个联盟的成员，这类问题可被建模为集合分区（set partition）问题。

XD [MT-MR-IA]，这类问题考虑即时分配情况下的联盟形成问题，每个机器人在一个时刻能执行多个任务，即可作为多个联盟的成员，这类问题可被建模为集合覆盖（set covering）问题。

XD [ST-MR-TA]，这类问题考虑时间扩展条件下的联盟形成问题，每个机器人在一个时刻只能执行一个任务，但随着时间的推移可成为不同联盟的成员。这类问题可近似被建模成 multi-mode multi-processor machine schedule 问题，也有研究用混合整数规划将其建模成 Coalition Formation with Spatial and Temporal Constraints 问题（CFSTP）

XD [MT-MR-TA]，目前很少有数学模型建模这类问题。

## 求解方案

目前有很多工作解决了联盟形成问题，如 Kraus 用贪婪、分布式的集合分区算法解决了一个 XD [ST-MR-IA]问题（搬运一些不同重量和大小的货物，有些货物可由一个机器人搬运，有些货物需要多个机器人搬运，因此机器人群需要形成联盟）。Guerrero and Oliver 用类似拍卖的算法解决了一个 XD [ST-MR-IA]问题（一个机器人发现一项任务并成为其领导者，并举行拍卖会，让其他机器人组成联盟来执行任务）。

#### 4) CD 类问题

在这类问题中，机器人完成一个任务的效用取决于机器人自己的 schedule 和其他机器人的 schedule，且任务分解具有多种形式。因此，分配复杂任务除了回答“谁”在“什么时间”执行每个任务外，还需要回答“哪一组子任务需要被分配”（即使用哪一个分解）。目前，没有发现合适的数学模型来描述 CD 类问题，但存在少数工作解决这类问题。

##### a) CD [ST-SR-IA], CD [ST-SR-TA], CD [MTSR-IA]和 CD [MT-SR-TA]问题

Jones 等人解决了一个考虑了“intra-path”约束的时间扩展多机器人协调问题。他们考虑了一个火灾救援场景。现在有一些火灾发生点，需要一群消防车机器人去救援，但在路线上可能存在一些碎石堆，它们会阻挡消防车的通过。这些碎石堆可以用推土机机器人清理。显然，并不是所有的碎石堆都需要清理；如果知道消防车的路线，那么只清理消防车沿线的碎石堆就足够了。然而，消防车每条路线的成本，以及路线的选择，又取决于要清理哪些碎石堆。在最基本的情况下，每个火灾点只需要一辆消防车，每堆碎石只需要一辆推土机清理，因此，这是一个 CD[ST-SR-TA]问题。它不仅要考虑火灾点和消防车之间的分配，还要考虑消防车走哪条路线以及推土机清理哪些碎石堆。作者用了分层拍卖+聚类，遗传算法求解了这个问题。

##### b) CD [ST-MR-IA], CD [ST-MR-TA], CD [MT-MRIA]和 CD [MT-MR-TA]问题

待补充，这部分有个代表工作还没看明白。

## 2. 分配方法

### 2.1 按方法分类

这部分内容来源于文献 [3], [4], [5], [6]。

表 1 不同 MRTA 方法的对比

|  | 基于聚类的方法 | 基于优化的方法 | 基于市场的方法 | 基于学习的方法 |
|--|---------|---------|---------|---------|
|--|---------|---------|---------|---------|

|      |              |            |                          |              |
|------|--------------|------------|--------------------------|--------------|
| 优势   | 简化任务分配，降低复杂度 | 提供最优解      | 分布式版本灵活、可扩展              | 处理复杂情况       |
| 局限   | 动态任务不好处理     | 计算量大，适应性有限 | 复杂任务不好处理，分布式版本可能实现不了全局最优 | 需要大量训练，适应性有限 |
| 擅长场景 | 逻辑型、简单的任务    | 好定义的，静态的任务 | 动态变化的任务                  | 复杂，不确定的任务    |
| 未来方向 | 动态聚类         | 混合模型，实时优化  | 自适应市场机制                  | 迁移学习         |

### 2.1.1 基于聚类的方法（clustering-based）

任务分配问题通常可建模为组合优化问题，但直接求解大规模任务分配需要指数级计算时间。基于聚类的方法通过将任务划分为若干 **clusters**，将全局优化转化为分阶段优化。其核心原理是通过对任务或环境的特征进行模式识别和分组，将复杂的大规模分配问题转化为多个局部优化问题，从而降低计算复杂度。比如输入是一组任务，它们会在某些标准下被分成一定数量的 **clusters**，接着这些 **clusters** 会被分配给机器人。这么做的好处是更少的任务需要被分配，计算复杂度降低了。

聚类的核心目标与 **MRTA** 的典型目标一致，比如，1）最小化路径成本（如总行驶距离、能耗），此时可通过空间邻近性聚类（如 **K-means**），使簇内任务紧凑，降低单个机器人的路径成本；2）最大化任务覆盖率，在探索、监测任务中，通过密度聚类（如 **DBSCAN**）覆盖更多关键区域；3）均衡负载，约束每个簇的任务数量或资源需求，避免机器人过载。

基于聚类的方法的研究重点在于怎么将任务聚类和怎么实现最优聚类。目前，常见的聚类方法有 **K-means** 聚类。

### 2.1.2 基于优化的方法（optimization-based）

优化是应用数学的一个分支，它关注于在一个问题的可行解空间中寻找一个最优解。可行解空间受到一些约束的限制，最优解会根据这些约束和一个标准被选取。这个标准就是整个系统的目标函数。目前，基于优化的方法可广泛的分为两大类，第一类是确定性优化方法，第二类是随机优化方法。

确定性优化方法会遵守一个严格的执行流程，变量、函数的数值都是可重复的，即相同的输入肯定会有相同的输出。确定性优化方法又可分为三类，第一类是数值方法，这类方法可通过数值计算工具（如迭代算法、数值逼近）直接求解问题。第二类是经典优化方法，这类方法基于数学规划理论，以严格的数学规则和收敛性为基础，比如混合整数规划、基于梯度的方法、二次型规划、**derivative-free** 方法等等。第三类方法是基于图的方法，这类方法将问题建模为图结构，通过搜索算法（如广度优先搜索、**A\***算法）探索解空间，比如 **blind/uninformed** 搜索。随机优化方法会存在一定的随机性，它又可被分为两类，第一类是 **trajectory-based** 方法，这类方法每次迭代仅维护单个候选解，通过随机扰动（如邻域搜索）逐步优化解的质量，类似于在解空间中沿特定轨迹搜索最优解，如模拟退火算法。第二类是 **population-based** 方法，这类方法维护多个候选解（种群），通过种群内个体间的协作（如交叉、变异、信息共享）探索解空间，比如遗传算法、蚁群算法、粒子群优化算法等等。

目前，应用在 MRTA 问题中的常见优化方法主要有三类，第一类是 **Hungarian** 算法，这类方法最常见，其核心原理是通过构建二分图匹配模型寻找最优任务分配方案，这种算法求解上是 **polynomial time** 的，保证找到全局最优解，适用于需要精确分配的静态任务场景。劣势在于算法要求任务和机器人数量严格相等，且任务信息预先已知。在动态任务分配（如新任务加入或机器人故障）中需频繁重新计算，效率较低。另外，该算法难以直接处理复杂约束（如任务优先级、时间窗口、资源动态变化）。总之，**Hungarian** 算法比较适用于求解 **ST-SR-IA** 类型分配问题。

第二类是混合整数规划，这类方法也很常见，它将任务分配问题转化为包含整数变量（如机器人选择、任务分配状态）和连续变量（如路径坐标、能耗）的线性或非线性数学模型。通过严格的数学规划保证解的最优性，且可以处理更复杂的约束（比如机器人的资源限制，任务时间限制和任务之间的顺序约束）。劣势在于混合整数规划属于 **NP-hard** 问题，求解时间随分配问题规模（机器人、任务数量）呈指数级增长。对突发任务增减或机器人故障的响应较慢，需重新建模和求解，实时性可能不够好。

第三类是生物启发式方法，这类方法比较具有前景，遗传算法、蚁群算法、粒子群优化算法就属于这一类。这类方法通过模拟自然界生物群体的智能行为（如进化、群体协作、觅

食等）来求解分配问题。遗传算法擅长处理复杂约束与全局优化，但实时性差；蚁群算法在动态规划中表现优异，适用于动态分配、旅行商等问题，但收敛慢，可能会出现局部最优，粒子群算法的优势在于收敛速度较快，实时性可以。但对任务优先级、时间窗等复杂约束的建模能力较弱。

因此，不少 MRTA 的研究工作都是将几类方法结合，发挥各自的优势。比如混合整数规划嵌入遗传算法、模拟退火等启发式方法，生成高质量初始解以加速收敛。又比如将问题分解为全局规划（粗粒度）和局部调整（细粒度）两层，全局层用图搜索或聚类算法划分任务组，局部层对每组任务使用 MIP 分配机器人。

### 2.1.3 基于市场的方法（market-based）

基于市场的方法的灵感来源于经济学，它提供了一种协调机器人之间活动的一种方式。这类方法主要基于了拍卖的概念。在经济学中，拍卖是根据竞价和某个标准将一些商品或服务分配给一群竞标者的过程。交易规则可以自由定义，但一般来讲，拍卖行都会将商品分配给竞价最高的投标者。

在 MRTA 问题中，每个机器人可表示为投标者，每个任务可表示为商品，机器人知道任务的信息（比如这个任务对机器人的价值，事先估算出机器人-任务效用），需要根据自己的能力给某个任务出价，给到拍卖行，拍卖行将任务分配给出价最高的那个机器人。

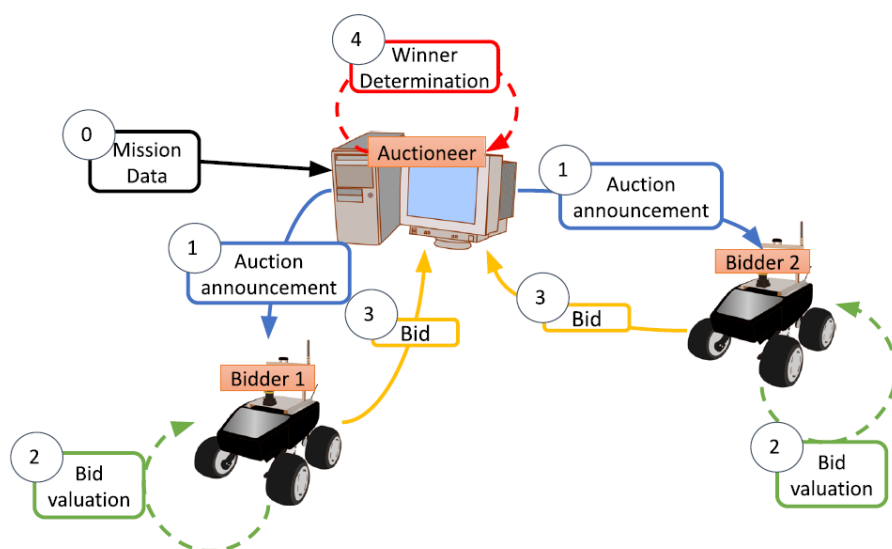


图 3 简单拍卖设置。实线表示消息传递，虚线表示计算过程，数字表示流程顺序。

目前，应用在 MRTA 问题中的拍卖策略大体上分为两类，第一类是单项拍卖（Single-Item Auctions），这类策略是最简单、常见的拍卖策略，即每个拍卖轮回只考虑对一个任务进行拍卖，每个机器人独立地对这个任务进行投标且仅能获得单一任务分配权（一对一分配）。典型形式包括英式拍卖（公开加价）或维克瑞拍卖（密封次价）。在单项拍卖中，机器

人对任务的竞价不会考虑以后的计划，因此单项拍卖很适合 ND 依赖类型的分配任务，又因为是一对一分配，所以适用于 ND[ST-SR-IA]类型分配问题。这类拍卖方式计算复杂度低，有利于实时分配，但可能因忽略任务协同效应导致全局次优解。单项拍卖有两类变体，

第一类变体是并行单项拍卖（Parallel Single-Item Auctions），在这种情况下，所有任务会一次性被拍卖，机器人会对所有任务进行投标（即发送一个投标列表，列表中每一项对应一个任务），接着拍卖行会在一个回合内完成所有任务分配。这类变体拍卖策略虽然想法简单，但实际上可能会获得一个很差的分配方案，即这类拍卖策略不稳定。

第二类变体是顺序单项拍卖（Sequential Single-Item Auctions），在这种情况下，每个拍卖轮回只拍卖一个任务，机器人需要对该任务进行投标，但是该投标值（即出价值）需要考虑以后的计划（或者说考虑已有的任务 plan），这种方式考虑了已有的任务 plan 的边际代价，换句话说，考虑了 plan 中的任务之间的协同因素。因此，这类拍卖策略适用于 ID 依赖类型的分配任务。

第二类是组合拍卖（Combinatorial Auctions），这类策略较复杂，允许机器人对任务组合进行联合投标，支持“全有或全无”（OR/XOR 逻辑）的竞标策略，可捕获任务间的协同效应或资源约束。它需要机器人考虑对所有的任务组合进行投标，因此这等价于考虑所有的分配情况，理论上确保了最优分配。出于以上原因，组合拍卖方式很适用于 ID 依赖类型的分配问题。组合投标的方式可以减少冲突，但投标空间随任务数呈指数增长，计算量很大，属于强 NP-hard 问题。

因此，有些 MRTA 工作采用了混合的拍卖策略，在动态环境中将两者结合，例如全局用组合拍卖预分配任务组，局部用单物品拍卖处理突发任务。

#### 2.1.4 基于学习的方法（learning-based）

得益于机器学习，神经网络的发展，基于学习的方法近些年开始受到关注。这类方法基于图、图神经网络、图卷积网络、强化学习等算法，来解决动态环境下的复杂 MRTA 问题。比如图结构方法的原理可能是将任务、机器人及其时间、空间等约束表示为图的节点与边，通过图卷积聚合邻域信息，提取全局特征。在这个图里可以实时更新节点属性（如机器人状态）和边权重（如任务优先级），支持动态调度。强化学习方法的原理可能是，观测向量被设计为包括任务信息和其他机器人的任务选择情况，动作向量被设计为任务的索引对应的概率，概率最大的任务会被优先分配，奖励被设计为用户的目标，比如完成任务的时间，这个目标和基于优化的方法中的优化目标是一致的。难点可能在于网络的设计，不同类型的分配任务需要的策略网络可能不同。

目前，还没有统一的分类标准将基于学习的方法进一步细分。但现在已有的方法整体上表现出两个特点，第一个特点是能够处理较复杂的分配问题，比如动态分配。第二个特点是很少考虑泛化，比如在一个新的场景中，机器人和任务的规模变了，学习到的分配方法可能就表现不足。

## 2.2 按通信架构分类

为了实现机器人之间的协作，通信是有必要的。一般分为集中式分配和分布式分配。

### 2.2.1 集中式方法

这种方法通过一个全局领导者（如中央控制器）收集所有机器人状态、任务需求及环境信息，基于全局效用函数（如最小化总能耗、最大化任务完成率）进行统一规划，常见的问题有工厂装配线调度、多无人机协同检测等等。

这种方法有两点优势，第一点是能实现全局最优，中央节点掌握完整信息，可生成全局最优任务分配方案，常见的混合整数规划就拥有这点性质。第二点优势是统一调度可有效避免机器人路径冲突。

同时，这类方法存在三点劣势，第一点是计算量大，机器人数量增加时，中央节点计算负荷呈指数级增长，因此能处理的分配问题规模是有限的。第二点是中心节点故障将导致系统瘫痪，鲁棒性差。第三点是需重新计算全局方案以响应突发任务或环境变化，计算量大导致实时性不足。

在有些工作中还提到，集中式方法又分为弱集中式方法和强集中式方法。弱集中式方法在任务执行过程中，可能根据某些条件变换全局领导者，比如环境变化、机器人资源变化等等。强集中式方法中，全局领导者则不变。因此，在鲁棒性方面，弱集中式方法相比强集中式方法更有优势。

### 2.2.2 分布式方法

这类方法中，每个机器人独立决策，基于局部感知信息（如自身状态、邻近机器人通信数据）通过协商机制（如合同网协议（Contract Net Protocol）、博弈论（Game Theory））完成任务分配。比如在灾害救援场景中，机器人通过本地交互动态调整搜救区域。

这类方法有两点优势，第一点是鲁棒性好，局部故障不影响整体系统，支持动态调整任务优先级。第二点是可扩展性好，机器人数量的增减无需重构全局模型，适合大规模异构系统（如物流仓库的 AGV 集群）。

这类方法有两点劣势，第一点是出现局部次优，缺乏全局信息可能导致任务重复或资源浪费，例如多个机器人争抢同一低优先级任务。第二点是可能存在分配冲突，需要设计分布式协商规则来解决，这点也是分布式方法的难点。

### 3. 常见问题

#### 3.1 车辆路线问题

##### 3.1.1 基本车辆路线问题（VRP）

###### 任务描述

现在有一个中央调度仓库和  $n$  个客户点，中央仓库管理着  $m$  辆车，这  $m$  辆车被要求向  $n$  个客户点运送货物，且在完成配送任务后需要返回中央仓库。所有车辆载重不限，所有客户点的货物需求量和种类都相同。任务目标（通常）是优化运输路径，以最小化总行驶距离或时间。

###### 类型分析

该问题属于 ID[ST-SR-TA]问题。

- 1) 单个车辆只能在某一时刻给一个客户送货，完成后才能再前往下一个客户点，所以是 ST 类型。
- 2) 车辆的载重不限，所以每个客户点的货物需求单个车辆即可满足，因此是 SR 类型。
- 3) 车辆需要规划一条完整的路径，而不是完成一个再决定下一个，因此是 TA 类型。
- 4) 每辆车会规划一条完整路径，客户点的送货顺序会影响整体的效用，因此是 ID 类型。

###### 讨论

VRP 问题等同于多旅行商问题，车辆相当于旅行商，客户点相当于城市。

##### 3.1.2 动态车辆路线问题（DVRP）

###### 任务描述

任务设置基本与 VRP 相同，差别在于车辆在配送过程中会出现突发状况（比如车辆出现抛锚故障、出现新客户、原来的客户改变需求、交通堵塞）。

###### 类型分析

该问题属于 ID[ST-SR-TA]问题，分析与 VRP 问题一致。

###### 讨论

求解 DVRP 问题时，当出现突发状况后，可能有部分车辆已经完成配送，即所有车辆并不在同一个起点，那么这时候问题就变成了下文的 OVRP 问题。“尚未服务的老客户”或“车辆出现抛锚问题”都可以当成新增客户来处理。

### 3.1.3 容量限制车辆路线问题（CVRP）

#### 任务描述

任务设置基本与 VRP 相同，差别在于现在的车辆载重相同，但有限。

#### 类型分析

该问题属于 ID[ST-SR-TA]问题或 XD[ST-MR-TA]问题。

- 1) 单个车辆只能在某一时刻给一个客户送货，完成后才能再前往下一个客户点，所以是 ST 类型。
- 2) 如果车量载重足够大，那和 VRP 一样，是 SR 类型。如果车辆载重不能满足一个客户点需求，那需要多辆车的协作，因此是 MR 类型。
- 3) 车辆需要规划一条完整的路径，而不是完成一个再决定下一个，因此是 TA 类型。
- 4) 如果任务类型是 SR，则依赖关系应属于 ID，如果任务类型是 MR，则依赖关系为 XD。

### 3.1.4 时间窗口车辆路线问题（VRP-TW）

#### 任务描述

任务设置与 CVRP 基本一致，差别在于每个客户有限制的访问时间段，这些时间段可以被中央调度仓知道。

#### 类型分析

该问题属于 ID[ST-SR-TA]问题或 XD[ST-MR-TA]问题，分析与 CVRP 问题一致。

### 3.1.5 混合车辆路线问题（HVRP）

#### 任务描述

任务设置相比基本的 VRP 问题有以下几点不同：1) 所有车辆的速度不同；2) 所有车辆载重不同，且有限；3) 每个客户点的需求量和货物种类不同。

#### 类型分析

该问题属于 ID[ST-SR-TA]问题或 XD[ST-MR-TA]问题。

- 1) 单个车辆只能在某一时刻给一个客户送货，完成后才能再前往下一个客户点，所以是 ST 类型。

- 2) 尽管车辆载重有限且客户点货物需求不一样，但适当组合还是能实现单辆车完成任务，所以是 SR 类型。如果车辆载重不能满足一个客户点需求，那需要多辆车的协作，因此是 MR 类型。
- 3) 车辆需要规划一条完整的路径，而不是完成一个再决定下一个，因此是 TA 类型。
- 4) 如果任务类型是 SR，则依赖关系应属于 ID，如果任务类型是 MR，则依赖关系为 XD。

### 3.1.6 开放式车辆路线问题（OVRP）

#### 任务描述

现在有  $m$  辆车， $n$  个客户点，这些车辆被要求以相同速度向  $n$  个客户点运输货物。所有车辆载重不限，每个客户点的需求量和种类均相同。车辆的起点可以不同，完成配送任务后的终点也可以不同。任务目标（通常）是优化运输路径，以最小化总行驶距离或时间。

#### 类型分析

该问题属于 ID[ST-SR-TA]问题。

- 1) 单个车辆只能在某一时刻给一个客户送货，完成后才能再前往下一个客户点，所以是 ST 类型。
- 2) 车辆的载重不限，所以每个客户点的货物需求单个车辆即可满足，因此是 SR 类型。
- 3) 车辆需要规划一条完整的路径，而不是完成一个再决定下一个，因此是 TA 类型。
- 4) 每辆车会规划一条完整路径，客户点的送货顺序会影响整体的效用，因此是 ID 类型。

#### 讨论

与 VRP 的不同点是车辆运送完之后不需要返回起点，因此第一次执行之后，问题就会变成多起点车辆路线问题。典型应用实例：外卖配送，从某个停车场出发单向行驶的公交车。

## 3.2 取送货问题

### 3.2.1 取送货问题（GPDP）

#### 任务描述

现在有  $m$  辆车， $n$  个客户点，这些车辆被要求向  $n$  个客户点送货。所有车辆载重有限，且运送货物的能力可能相同，也可能不同。每个客户点的需求量可能相同也可能不同。车辆在完成配送任务后需要前往另一个地方进行取货。取（送）货地点可以是中央仓库，也可以是客户点。任务目标通常是优化运输路径，以最小化总行驶距离或时间。

#### 类型分析

该问题属于 ID[ST-SR-TA]问题或 XD[ST-MR-TA]问题，分析与 HVRP 问题一致。

## 讨论

GPDP 是 VRP 问题的一种扩展，包括了送货和取货，而不只是送货。GPDP 又可分为多种类型的取送货问题，具体分类如下图所示。比如，1) VRPB 问题的特点是中心仓库与客户点之间进行取货运货，运货到客户点并在客户点取货后送回中心仓库（车辆出发的起点为中心仓库，最后取货后输送的终点也是中心仓库），通常情况下是先执行完所有配送再取货。路径：仓库（Depot）→客户点（Delivery）→回程取货（Backhaul）→仓库（Depot）；2) VRPPD 问题的特点是在客户点之间进行取货送货，与 VRPB 的区别是配送和取货是配对的，也就是执行一次配送就执行一次取货，其中，unpaired 指的是取送货地点不是固定的，如超市补货，仓库可能向多个点送货，多个点向仓库回收货物，而 paired 指的是从某一地点取货，只能送到指定的送货地点，如网约车、包裹快递。

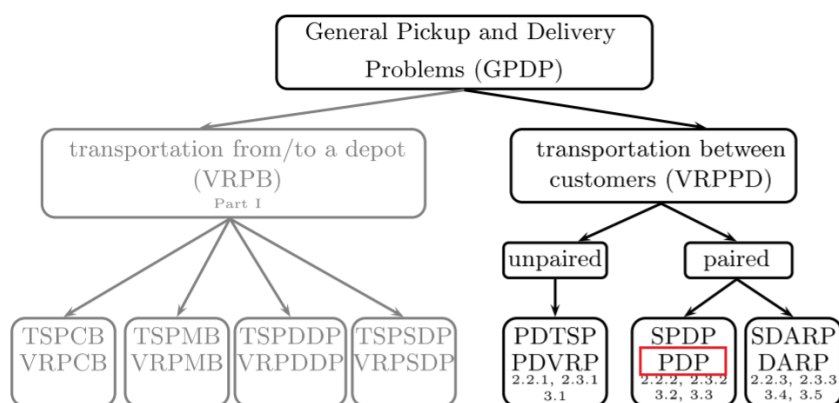


图 4 GPDP 问题分类

## 3.3 多旅行商问题

### 3.3.1 旅行商问题（TSP）

#### 任务描述

现在有  $n$  个城市和一个推销员，这个推销员需要访问这  $n$  个城市，且每个城市只能被访问一次，访问完后推销员需要回到出发城市，任务目标是最小化旅行距离。

#### 类型分析

该问题属于 ID[ST-SR-TA]问题。

- 1) 旅行商同一时间只能访问一个城市，所以是 ST 类型。
- 2) 一个城市只需要被一个旅行商访问，所以是 SR 类型。
- 3) 旅行商需要规划一条完整的路线，包含多个城市，所以是 TA 类型。
- 4) 旅行商访问的城市顺序会影响整体的效用，所以是 ID 依赖类型。

讨论

一些生活中的例子：1）物流，比如一家快递公司需要向某城市的 10 个客户分别送货，每个客户的地址已知，配送员从快递中心出发，需要确定一个最短路径顺序一次将货物送达每个客户，并最终回到出发点。2）生产，在生产过程中，机器人需要依次访问不同的位置进行组装或检查。

3.3.2 多旅行商问题（MTSP）

任务描述

现在有  $n$  个城市和  $m$  个推销员，这些推销员需要访问这  $n$  个城市，且每个城市只能被访问一次，访问完后推销员需要回到出发城市，任务目标是最小化旅行距离。

类型分析

该问题属于 ID[ST-SR-TA]问题，分析与 TSP 问题一致。

讨论

MTSP 问题存在不少变体，下面图片展示了可能改变的条件。比如，1）单仓库闭合路径的 MTSP 指的是  $m$  个旅行商从同一起点出发，最终返回起点；2）单仓库开放路径的 MTSP 指的是  $m$  个旅行商在最后一处访问城市任务终止，不再回到起点；3）多仓库闭合路径的 MTSP 指的是  $m$  个旅行商从不同起点出发，遍历了目标城市后最终返回到各自的起点；4）多仓库闭合路径的 MTSP 指的是  $m$  个旅行商从不同起点出发，任务结束后停留在当前城市，无需返回各自起点。

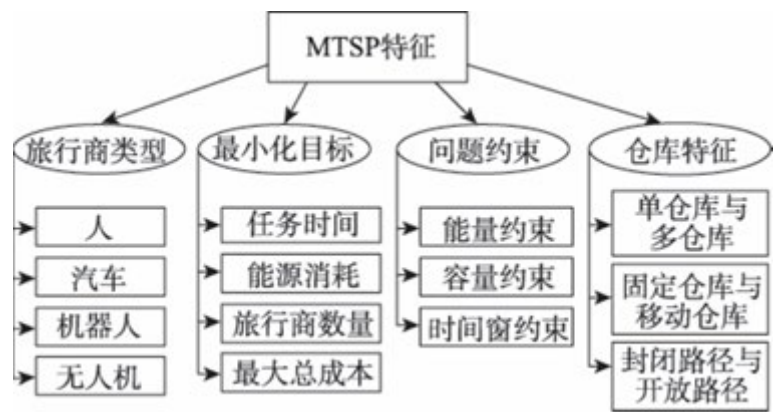


图 5 MTSP 问题分类

3.4 作业调度问题

3.4.1 单机调度问题（SMP）

任务描述

现在有一台机器和  $n$  个作业，每个作业只有一道工序，所有工序都可在该机床上加工。每道工序加工时间确定。任务目标是最小化作业时间。

### 类型分析

该问题属于 ND[ST-SR-TA]或 ID[ST-SR-TA]问题。

- 1) 加工机器在同一时间只能加工一道工序，所以是 ST 类型。
- 2) 一道工序被一台机器加工即可完成，所以是 SR 类型。
- 3) 该机器需要加工一系列作业，所以是 TA 类型。
- 4) 如果每个作业之间不存在依赖关系，那加工顺序不影响任务目标，所以是 ND 类型；如果某些作业需要先后完成，那顺序会影响目标，所以是 ID 依赖类型。

## 3.4.2 并行机调度问题（PMP）

### 任务描述

现在有  $m$  台机器和  $n$  个作业，每个作业只有一道工序，每道工序可以在任一台机器上被加工，加工时间确定。任务目标是最小化作业时间。

### 类型分析

该问题属于 ND[ST-SR-TA]或 ID[ST-SR-TA]或 XD[ST-SR-TA]问题

- 1) 加工机器在同一时间只能加工一道工序，所以是 ST 类型。
- 2) 一道工序被一台机器加工即可完成，所以是 SR 类型。
- 3) 每台机器可能需要加工一系列作业，所以是 TA 类型。
- 4) 如果每个作业之间不存在依赖关系，那加工顺序不影响任务目标，所以是 ND 类型；如果某些作业需要先后完成，那顺序会影响目标，但这些有顺序的工序在一台机器上被完成，则是 ID 依赖类型；如果某些作业需要先后完成，那顺序会影响目标，且这些有顺序的工序在不同机器上被完成，则是 XD 依赖类型。

## 3.4.3 开放车间调度问题（OSP）

### 任务描述

现在有  $m$  台机器和  $n$  个作业，每个作业有  $L_i$  道工序。每个工件的工序之间没有约束，即工件的加工可以从任何一道工序开始，在任何一道工序结束。每道工序可在任一台机器上被加工，加工时间确定。任务目标是最小化加工时间。

### 类型分析

该问题属于 ND[ST-SR-TA]或 ID[ST-SR-TA]或 XD[ST-SR-TA]问题，分析与 PMP 问题一致。

### 3.4.4 流水车间调度问题（FSP）

#### 任务描述

现在有  $m$  台机器和  $n$  个相同的作业，每个作业有  $L_i$  道工序。每个作业的多道工序之间有顺序约束，每道工序可在一台机器上被加工，加工时间确定。不同机器的加工功能可能不同。任务目标是最小化加工时间。

#### 类型分析

该问题属于 ID[ST-SR-TA]或 XD[ST-SR-TA]问题。

- 1) 加工机器在同一时间只能加工一道工序，所以是 ST 类型。
- 2) 一道工序被一台机器加工即可完成，所以是 SR 类型。
- 3) 每台机器可能需要加工一系列作业，所以是 TA 类型。
- 4) 因为一个作业的工序之间有顺序约束，如果一个整个作业都在一台机器上被加工，则不同机器之间的加工计划互不影响，所以是 ID 依赖类型；如果一个作业的不同工序由不同机器完成，则不同机器之间的加工计划有影响，所以是 XD 依赖类型。

### 3.4.5 作业车间调度问题（JSP）

#### 任务描述

现在有  $m$  台机器和  $n$  个不同的作业，每个作业有  $L_i$  道工序。每个作业的多道工序之间有顺序约束，每道工序可在一台机器上被加工，加工时间确定。不同机器的加工功能可能不同。任务目标是最小化加工时间。

#### 类型分析

该问题属于 ID[ST-SR-TA]或 XD[ST-SR-TA]问题，分析与 FSP 问题一致。

## 3.5 背包问题

#### 任务描述

现在有一个容量为  $C$  的背包和  $n$  个物品，每个物品具有不同的重量和价值，要求在不超过背包容量的前提下，选择一些物品放入背包中，使得这些物品的总价值最大。

#### 类型分析

该问题属于 ID[ST-SR-TA]问题。

## 参考文献

- [1] “A comprehensive taxonomy for multi-robot task allocation - G. Ayorkor Korsah, Anthony Stentz, M. Bernardine Dias, 2013.” Accessed: Mar. 20, 2025. [Online]. Available: <https://journals.sagepub.com/doi/abs/10.1177/0278364913496484>
- [2] B. P. Gerkey and M. J. Mataric, “A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems,” *Int. J. Robot. Res.*, vol. 23, no. 9, pp. 939–954, Sep. 2004, doi: 10.1177/0278364904045564.
- [3] A. K A *et al.*, “A Systematic Literature Review on Multi-Robot Task Allocation,” *ACM Comput. Surv.*, vol. 57, no. 3, pp. 1–28, Nov. 2024, doi: 10.1145/3700591.
- [4] H. Chakraa, F. Guérin, E. Leclercq, and D. Lefebvre, “Optimization techniques for Multi-Robot Task Allocation problems: Review on the state-of-the-art,” *Robot. Auton. Syst.*, vol. 168, p. 104492, Oct. 2023, doi: 10.1016/j.robot.2023.104492.
- [5] A. Khamis, A. Hussein, and A. Elmogy, “Multi-robot Task Allocation: A Review of the State-of-the-Art,” in *Cooperative Robots and Sensor Networks 2015*, A. Koubâa and J. R. Martínez-de Dios, Eds., Cham: Springer International Publishing, 2015, pp. 31–51. doi: 10.1007/978-3-319-18299-5\_2.
- [6] F. Quinton, C. Grand, and C. Lesire, “Market Approaches to the Multi-Robot Task Allocation Problem: a Survey,” *J. Intell. Robot. Syst.*, vol. 107, no. 2, p. 29, Feb. 2023, doi: 10.1007/s10846-022-01803-0.