XXX工程

项目名称

基于分支定界法的任务分配算法研究报告

北京大学

2022年6月

项目名称

|  |  |
| --- | --- |
| 拟 制： |  |
| 审 核： |  |
| 批 准： |  |
| 会 签： |  |

**目录**

[2 组合优化问题 1](#_Toc138080514)

[2.1 组合优化问题 1](#_Toc138080515)

[2.2 常见问题 1](#_Toc138080516)

[2.2.1 生产调度问题 1](#_Toc138080517)

[2.2.2 旅行商问题 2](#_Toc138080518)

[2.2.3 装箱问题 3](#_Toc138080519)

[2.2.4 最大割问题 5](#_Toc138080520)

[2.3 常见算法 5](#_Toc138080521)

[2.3.1 传统算法 5](#_Toc138080522)

[2.3.1.1 分支定界法 5](#_Toc138080523)

[2.3.2 启发式算法 6](#_Toc138080524)

[2.3.2.1 群算法 6](#_Toc138080525)

[2.3.2.2 进化算法 7](#_Toc138080526)

[3 基于分支定界法任务分配 9](#_Toc138080527)

[3.1 系统模型 9](#_Toc138080528)

[3.2 分支定界法 9](#_Toc138080529)

[3.3 其他分配方法 13](#_Toc138080530)

[3.3.1 混合整数线性规划 13](#_Toc138080531)

[3.3.2 局部搜索 15](#_Toc138080532)

[4 在线任务更新 17](#_Toc138080533)

[4.1 扰动在线自适应 17](#_Toc138080534)

[4.2 基于分支定界法的重规划 18](#_Toc138080535)

[5 参考文献 20](#_Toc138080536)

组合优化问题

组合优化问题

在实际工程应用中，有一类优化问题需要从集合的所有组合中找出一个最优方案或编排，这类离散空间中的优化问题称为组合最优化问题（combinatorial optimization problem，COP）[1]。组合最优化（combinatorial optimization，CO）的求解方法广泛应用于交通运输、管理、电力、航天、通信等领域[1]，其快速求解具有重要的理论意义和实用价值。例如，车辆的调度、金融资产的配置、仓库货物存储和运输路线的设计等实际问题都属于 COP 问题，随着这些优化问题实例规模的不断增大和实例中动态及随机因素的增加，传统方法的求解将耗费巨大的时间，问题结构一旦发生变化，传统方法需要重新搜索求解，计算成本也会随之提高，快速求解这些优化问题变得十分困难。

组合优化最核心的困难在于组合的可能性会随着规模的扩大而发生指数爆炸。因此，大多数组合优化问题都是NP-hard的问题[2]，无法在多项式时间内获得最优解。但是这类问题往往描述非常简单，具有很强的工程性，但是这些问题的基础解法需要极长的运行时间与极大的储存空间，以至根本不可能在先有的计算机上进行大规模的计算。这些问题的代表性和复杂性正是这个学科经久不衰的原因。

常见问题

典型的组合优化问题有很多：生产调度问题，旅行商问题，背包问题，装箱问题，图着色问题，聚类问题等等。这些问题都有着深厚的工程背景，因此不仅仅是在数学上，在工程上也有很大的发展。

生产调度问题

生产调度问题(Job Shop Schedule Problem)[3]是一个非常经典的组合优化问题，在生产制造、项目管理的计划排班上广泛存在；学界对这个课题的研究已经超过50年了，建立了各种理论模型并提出了多种算法，应该说该问题属于最难处理的组合优化问题[4]。

它的工程背景就直接来源于生产组织过程。生产调度以生产进度计划为依据，生产进度计划要通过生产调度来实现。生产调度的必要性是由工业企业生产活动的性质决定的。现代工业企业，生产环节多，协作关系复杂，生产连续性强，情况变化快，某一局部发生故障，或某一措施没有按期实现，往往会波及整个生产系统的运行[5]。因此，加强生产调度工作，对于及时了解、掌握生产进度，研究分析影响生产的各种因素，根据不同情况采取相应对策，使差距缩小或恢复正常是非常重要的。不仅仅是一个数学问题，同时也是组织学，社会学问题。

最经典的问题模型是，一个工厂需要生产n个产品，每个产品种类各不相同，有不同的加工流程，这个工厂有m个工艺设备，这些加工流程会占用不同的工艺设备不同的时间，如何规划加工的顺序使得整体完成任务的时间最短。JSP问题可以描述为在满足资源（设备，人员，时间）约束的情况下安排加工顺序，使得总消耗最少。

旅行商问题

旅行推销员问题（英语：Travelling salesman problem, TSP）[6]是这样一个问题：给定一系列城市和每对城市之间的距离，求解访问每一座城市一次并回到起始城市的最短回路。它是组合优化中的一个NP难问题，在运筹学和理论计算机科学中非常重要。经典的TSP可以描述为：一个商品推销员要去若干个城市推销商品，该推销员从一个城市出发，需要经过所有城市后，回到出发地。应如何选择行进路线，以使总的行程最短。从图论的角度来看[7]，该问题实质是在一个带权完全无向图中，找一个权值最小的Hamilton回路。由于该问题的可行解是所有顶点的全排列，随着顶点数的增加，会产生组合爆炸，它是一个NP完全问题。由于其在交通运输、电路板线路设计以及物流配送等领域内有着广泛的应用，国内外学者对其进行了大量的研究。早期的研究者使用精确算法求解该问题，常用的方法包括：分枝定界法、线性规划法、动态规划法等[8]。但是，随着问题规模的增大，精确算法将变得无能为力，因此，在后来的研究中，国内外学者重点使用近似算法或启发式算法，主要有遗传算法、模拟退火法、蚁群算法、禁忌搜索算法、贪婪算法和神经网络等[9]。

最早的旅行商问题的数学规划是由Dantzig（1959）等人提出[10]，并且是在最优化领域中进行了深入研究。许多优化方法都用它作为一个测试基准。尽管问题在计算上很困难，但已经有了大量的启发式算法和精确方法来求解数量上万的实例，并且能将误差控制在1%内。

TSP的研究历史很久，最早的描述是1759年欧拉研究的骑士环游问题，即对于国际象棋棋盘中的64个方格，走访64个方格一次且仅一次，并且最终返回到起始点。1954年，Geo~eDanzig等人用线性规划的方法取得了旅行商问题的历史性的突破——解决了美国49个城市的巡回问题。这就是割平面法，这种方法在整数规划问题上也广泛应用。后来还提出了一种方法叫做分枝定界法，所谓定界，就是求出问题解的上、下界，通过当前得到的限界值排除一些次优解，为最终获得最优解提示方向。每次搜索下界最小的分枝，可以减小计算量。

装箱问题

箱问题是复杂的离散组合最优化问题之一[11]。装箱问题同许多组合最优化问题一样, 如旅行商问题、图的划分问题等一样属于NP-hard问题。经典的装箱问题要求把一定数量的物品放入容量相同的一些箱子中,使得每个箱子中的物品大小之和不超过箱子容量并使所用的箱子数目最少。

从20世纪70年代初开始,装箱问题就引起了广泛的探讨和研究。然而装箱问题可以追溯到1831年高斯(Gauss)开始研究布局问题，因为装箱问题和布局问题本质上是一样的,到现在已有百余年的历史。虽然经过几代人的努力,但迄今尚无成熟的理论和有效的数值计算方法。由于目前NP完全问题不存在有效时间内求得精确解的算法,装箱问题的求解极为困难,因此,从70~80年代开始,陆续提出的装箱算法都是各种近似算法,如下次适应、首次适应、降序下次适应和调和算法等。

装箱问题广泛存在于工业生产,包括服装行业的面料裁剪、运输行业的集装箱货物装载、加工行业的板材型材下料、印刷行业的排样和现实生活中包装、整理物件等。在计算机科学中,多处理器任务调度、资源分配、文件分配、内存管理等底层操作均是装箱问题的实际应用,甚至还出现在一些棋盘形、方块形的数学智力游戏中。装箱问题的研究文献分布面很广,在运筹学、计算机辅助设计、计算机图形学、人工智能、图像处理、大规模集成电路逻辑布线设计、计算机应用科学等诸多领域都有装箱问题最新的研究动态和成果出现,从这个角度来讲,布局问题涉及到了工业生产的方方面面,也足以证明了装箱问题的应用前景日趋广泛而重要[12-14]。

装箱问题可分为一维装箱问题，二维装箱问题，三维装箱问题三种。现实生活中常见的应该是三维装箱问题。

一维装箱问题只考虑一个因素，比如重量、体积、长度等。

二维装箱问题考虑两个因素——给定一张矩形的纸（布料、皮革），要求从这张纸上剪出给定的大小不一的形状，求一种剪法使得剪出的废料的面积总和最小。常见问题包括堆场中考虑长和宽进行各功能区域划分、停车场区位划分、包装材料裁切时考虑怎样裁切使得材料浪费最少、服装布料裁切、皮鞋制作中的皮革裁切等。

三维装箱问题考虑三个因素——一般指长、宽、高。装车、装船、装集装箱等要考虑这三个维度都不能超。

根据目标的不同,三维装箱问题可分成以下几类 [2] ：

箱柜装载问题(three-dimensional bin packing problem，简称3D-BPP)：给定一些不同类型的方型箱子和一些规格统一的方型容器，问题是要把所有箱子装入最少数量的容器中。

容器装载问题(three-dimensional container-packing problems，简称3D-CPP)：在该问题中，所有箱子要装入一个不限尺寸的容器中，目标是要找一个装填，使得容器体积最小。

背包装载问题(three-dimensional knapsack loading problems，简称3D-KLP)：每个箱子有一定的价值，背包装载是选择一部分箱子装入容器中，使得装入容器中的箱子总价值最大。如果把箱子的体积作为价值，则目标转化为使容器浪费的体积最小。

最大割问题

最大割问题（英语：Maximum Cut）是NP完备问题[15, 16]。给定一张图，求一种分割方法，将所有顶点（Vertex）分割成两群，同时使得被切断的边（Edge）数量最大。当每条边都有权重的时候，那么我们要保证最后的切掉的边权重之和最大。最大割常被用于芯片制造过程中，便于找到能使得戒断线路最少的截面。

常见算法

传统算法

分支定界法

分支定界法（branch and bound）[17, 18]是一种求解整数规划问题的最常用算法。这种方法不但可以求解纯整数规划，还可以求解混合整数规划问题。分支定界法是一种搜索与迭代的方法，选择不同的分支变量和子问题进行分支。对于两个变量的整数规划问题，使用网格的方法有时更为简单。通常，把全部可行解空间反复地分割为越来越小的子集，称为分支；并且对每个子集内的解集计算一个目标下界（对于最小值问题），这称为定界。在每次分枝后，凡是界限超出已知可行解集目标值的那些子集不再进一步分枝，这样，许多子集可不予考虑，这称剪枝。这就是分枝定界法的主要思路。

分枝界限法是由三栖学者理查德·卡普（Richard M.Karp）[19]在20世纪60年代发明，成功求解含有65个城市的旅行商问题，创当时的记录。“分枝界限法”把问题的可行解展开如树的分枝，再经由各个分枝中寻找最佳解。分枝界限法也能够使用在混合整数规划问题上，其为一种系统化的解法，以一般线性规划之单形法解得最佳解后，将非整数值之决策变量分割成为最接近的两个整数，分列条件，加入原问题中，形成两个子问题(或分枝)分别求解，如此便可求得目标函数值的上限（上界）或下限（下界），从其中寻得最佳解。

在每次分支后，对凡是界限超出已知可行解值那些子集不再做进一步分支。这样，解的许多子集（即搜索树上的许多结点）就可以不予考虑了，从而缩小了搜索范围。这一过程一直进行到找出可行解为止，该可行解的值不大于任何子集的界限。因此这种算法一般可以求得最优解。将问题分枝为子问题并对这些子问题定界的步骤称为分枝定界法。

对搜索树上的某些点必须作出分枝决策，即凡是界限小于迄今为止所有可行解最小下界的任何子集（节点），都有可能作为分枝的选择对象（对求最小值问题而言）。怎样选择搜索树上的节点作为下次分枝的节点呢？有两个原则：

1）从最小下界分枝（优先队列式分枝限界法）：每次算完界限后，把搜索树上当前所有叶节点的界限进行比较。找出限界最小的节点，此结点即为下次分枝的结点。

·优点：检查子问题较少，能较快地求得最佳解；

·缺点：要存储很多叶节点的界限及对应的耗费矩阵，花费很多内存空间。

2）从最新产生的最小下界分枝（队列式（FIFO）分枝限界法）：从最新产生的各子集中按顺序选择各结点进行分枝，对于下届比上届还大的节点不进行分枝。

优点：节省了空间；缺点：需要较多的分枝运算，耗费的时间较多。

这两个原则更进一步说明了，在算法设计中的时空转换概念。

分枝定界法已经成功地应用于求解整数规划问题、生产进度表问题、货郎担问题、选址问题、背包问题以及可行解的数目为有限的许多其它问题。对于不同的问题，分枝与界限的步骤和内容可能不同，但基本原理是一样的。

启发式算法

群算法

群智能理论研究领域主要有两种算法：蚁群算法和粒子群算法[20, 21]。蚁群算法是对蚂蚁群落食物采集过程的模拟，已成功应用于许多离散优化问题。粒子群优化算法也是起源于对简单社会系统的模拟，最初是模拟鸟群觅食的过程，但后来发现它是一种很好的优化工具。

蚁群在寻找食物时通常可以找到巢穴到食物之间最短的路径，每只蚂蚁个体的智能度很低，但是通过彼此之间的简单交流，蚂蚁群体的智能度很高，这一现象给我们在搜索问题最优解中带来了直接的启发。每只蚂蚁在寻找道路时是随机的，并且会在路上散发信息素，那么在单位时间内最优道路上单位时间通过的蚂蚁数量最多，信息素的浓度最高，新来的蚂蚁和返回的蚂蚁会按着信息素浓度高的路径走，这样最优路径上的信息素浓度会越来越高，而其他路上的信息素随着挥发浓度越来越低，这样就形成了一个正反馈，时最优路径最终被几乎所有蚂蚁使用。蚁群算法即模拟这一现象。

近年来，各种各样的群算法被提出，包括鲸群[22]，狼群[23]，蜂群[24]等等。这些算法考虑不同生物搜索通讯的模式，进而在不同的算法上体现出各自的优点。

进化算法

进化算法也是一类常被用于处理组合优化问题的算法[25]。进化算法，或称“演化算法”（evolutionary algorithms）是一个“算法簇”，尽管它有很多的变化，有不同的遗传基因表达方式，不同的交叉和变异算子，特殊算子的引用，以及不同的再生和选择方法，但它们产生的灵感都来自于大自然的生物进化。与传统的基于微积分的方法和穷举法等优化算法相比，进化计算是一种成熟的具有高鲁棒性和广泛适用性的全局优化方法，具有自组织、自适应、自学习的特性，能够不受问题性质的限制，有效地处理传统优化算法难以解决的复杂问题。进化计算包括遗传算法（Genetic Algorithms）、遗传规划（Genetic Programming）、进化策略（Evolution Strategies）和进化规划（Evolution Programming）4种典型方法。第一类方法比较成熟，现已广泛应用，进化策略和进化规划在科研和实际问题中的应用也越来越广泛。

从二十世纪40年代起，生物模拟就构成了计算科学的一个组成部分，像早期的自动机理论，就是假设机器是由类似于神经元的基本元素组成的，它向人们展示了第一个自复制机模型。这些年来诸如机器能否思维、基于规则的专家系统是否能胜任人类的工作，以及神经网络可否使机器具有看和听的功能等有关生物类比的问题已成为人工智能关注的焦点。最近生物计算在机器昆虫和种群动态系统模拟上所取得的成功激励越来越多的人们致力于人工生命领域的研究。当今计算机科学家和分子生物学家已开始携手进行合作研究，并且类比也得到了更为广泛的应用。

自然界生物体通过自身的演化就能使问题得到完美的解决。这种能力让最好的计算机程序也相形见拙。计算机科学家为了某个算法可能要耗费数月甚至几年的努力，而生物体通过进化和自然选择这种非定向机制就达到了这个目的。

近三十年的不断的研究和应用已经清楚地表明了模拟自然进化的搜索过程可以产生非常鲁棒的计算机算法，虽然这些模型还只是自然界生物体的粗糙简化。进化算法就是基于这种思想发展起来的一类随机搜索技术，它们是模拟由个体组成的群体的集体学习过程。其中每个个体表示给定问题搜索空间中的一点。进化算法从任一初始的群体出发，通过随机选择（在某些算法中是确定的）、变异和重组（在某些算法中被完全省去）过程，使群体进化到搜索空间中越来越好的区域。选择过程使群体中适应性好的个体比适应性差的个体有更多的复制机会，重组算子将父辈信息结合在一起并将他们传到子代个体，变异在群体中引人了新的变种。

目前研究的进化算法主要有三种典型的算法：遗传算法、进化规划和进化策略。这三种算法是彼此独立发展起来的，遗传算法由美国J.Holand创建，后由K.De Jong，J.Grefenstette，D.Goldberg和L.navis等人进行了改进；进化规划最早由美国的L·J·Fogel，A.J.Owens和M.J.walsh提出，最近又由D.B.Fogel进行了完善；进化策略是由德国的I·Reehenberg和H.p.Sehwefel建立的。

群体搜索策略和群体中个体之间的信息交换是进化算法的两大特点。它们的优越性主要表现在:首先，进化算法在搜索过程中不容易陷入局部最优，即使在所定义的适应度函数是不连续的，非规则的或有噪声的情况下，它们也能以很大的概率找到全局最优解；其次，由于它们固有的并行性，进化算法非常适合于巨量并行机；再者，进化算法采用自然进化机制来表现复杂的现象，能够快速可靠地解决非常困难的问题；此外，由于它们容易介人到已有的模型中并且具有可扩展性，以及易于同别的技术混和等因素，进化算法目前已经在最优化、机器学习和并行处理等领域得到了越来越广泛的应用。1993年德国Dortmound大学的等人在一份研究报告中搜集了篇有关进化算法的应用的科技文献。

# 基于分支定界法任务分配

## 系统模型

任务被描述为一个偏序集，由一组子任务和对应的偏序约束构成。子任务为

（3-1）

每个指向一个具体的组合任务。偏序约束表明不同任务之间的时序关系，表示任务1需要在任务2执行前开始。表示任务1和任务2的执行时间不允许有交集。

在给定的偏序集合后，我们将在本节介绍如何使用该集合来计算这些子任务的最优分配。更具体地说，我们考虑以下任务分配子问题：给定任意偏序集，为*P*中所有子任务对多智能体系统*N*的最优分配。该最优分配*S*具有两个特点，一是满足偏序中所有的部分排序需求，二是最小化所有子任务的最大完成时间。将*m*个任务以一定约束分配给*n*个智能体，这是一类典型的规划问题。在工程中常用被称为流水线模型或者工件模型，其中常见的约束有成本约束，功能约束，以及顺序约束等等。而且智能体需要考虑移动距离的约束，这是常见的多旅行商问题。这些问题都是典型的NP难问题，对应的解法有混合整数线性规划方法MILP，局部搜索方法（local search），和具有anytime性质的分支定界法。我们给出了三种方法的算法内容，并测试了对应的性能。结果表明，混合整数规划方法速度最慢，局部搜索速度较快但是不能保证最优解，分支定界法能较快的获取当前最优解，并继续迭代获取最优解，是理想的解决方法。

## 分支定界法

本小节提出了一种基于分支定界法（branch and bound）[26]搜索方法任意时间分配算法，BnB是一种求解整数规划问题的最常用算法。这种方法不但可以求解纯整数规划，还可以求解混合整数规划问题。它不仅是完整的和最优的，而且也意味着在任何给定的时间预算内都可以查询到一个好的解决方案。通常，把全部可行解空间反复地分割为越来越小的子集，称为分支；并且对每个子集内的解集计算一个目标下界（对于最小值问题），这称为定界。在每次分枝后，凡是界限超出已知可行解集目标值的那些子集不再进一步分枝，这样，许多子集可不予考虑，这称剪枝。这就是分枝定界法的主要思路。

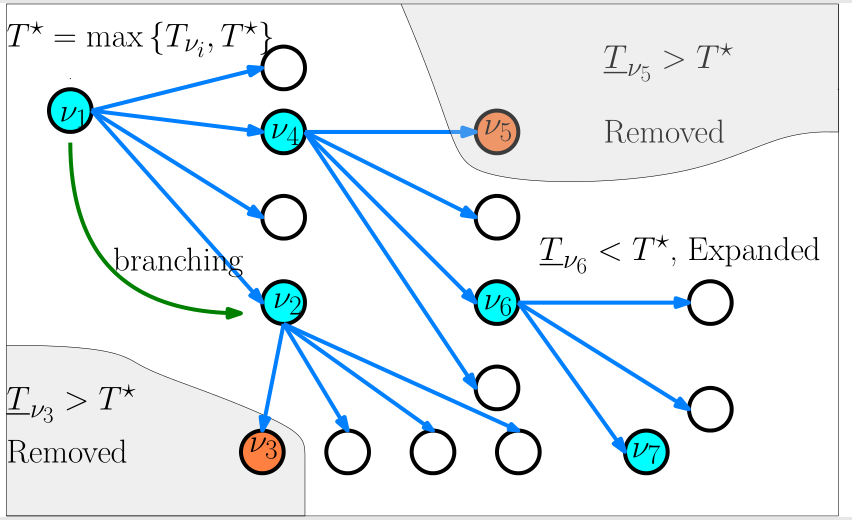


图6.13 分支定界法流程图

BnB算法的四个典型组成部分是节点扩展、分支方法和上下界的设计，如图6.13所示。下面，我们将详细描述这些部分。

**1）节点扩展**

首先BnB算法是开始于一个空白的搜索树，搜索树中的每个节点表示子任务的一个部分赋值

（3-2）

其中，是一个代表着任务的分配给智能体n的有序序列。例如，对于三个智能体的系统，，这意味着两个子任务，被分配给智能体1，而智能体2和智能体3没有任务。因此从零分配的空白根节点开始，树通过生成子节点展开，其中包含的子任务将按照偏序所限制的关系，在每次的节点扩展中分给所有潜在可行的智能体，直到分配完所有的子任务后。当该节点无法再拓展出其他的节点时，这个节点将终结。

**2）分支方法**

分支方法决定访问这些子节点的顺序。可以使用多种搜索方法，如广度优先搜索(BFS)、深度优先搜索(DFS)或A\*搜索。我们这里使用A\*搜索，因为启发式函数与下面介绍的下界算法有相似之处，且预先设定的分配偏好能更快的选择出更好的节点。更具体地说，子节点集按照整个计划在给定其当前赋值的情况下估计完成时间的顺序展开。考虑到启发式算法在此环节的应用，利用强化学习对分支方法进行估计也是可行的。以节点分配的任务作为强化学习的输入，采用对每次分配求得的上界值作为当前分配的奖励，能够构建出深度强化学习网络，更为快速的估计出当前节点分配是否能获取一个更为优秀的解，使得分支快速收敛到最优解附近。

**3）下界算法**

下界方法通常忽略大部分的约束，获取一个不可行的解，但是理论上能保证最优解将不会小于下界。当下界算法得到的值仍然超越了当前的最优解时，这意味着该分支不会再出现比最优解更好的结果了，此时能在分支定界法中将该节点除去，跳过节点扩张这一步因为不可能在这个分支找到最优解。

这里下界算法采用一个极简的线性规划问题，采用类似局部搜索的算法获取当前分配的边界，其中已经分配的任务集为，未分配的任务集为：

（3-3）

（3-4）

（3-5）

不同的是，这里第二个算式只能选取内的元素，因为单纯的≠约束是非凸的，这里的最优值可能会影响到下界的分配。由于忽略的运动规划，且一个智能体可能被要求在执行多个任务，实际上这是不可能的，因此下界将永远小于上界。能保证下界大于最优值时，所对应的分支不存在最优解。

**4）上界算法**

上界算法采用一类启发式的*A\**算法，其核心是基于贪婪的分配方法，最坏情况能在N\*N-1步内获取到解。

|  |
| --- |
| 输入：当前节点，偏序*P* |
| 输出：上界解，以及对应最小执行时间上界 |
| 1 初始化，节点序列*V*  2 while 还有子任务未分配  3 根据序列*A*，从*V*中取出函数值最大的节点*v*  4 选取可行的子任务，随机在*v*上分配*n*次，生成子节点集  5 计算子节点集对应的值  6 将加入*V*，将加入序列  7 if 此次分配后生成完整的节点  8 循环结束  9 返回当前最优节点及对应值 |

上界的算法核心在于对*A\**函数的估计，这里采用的是执行效率，即每次都选择执行效率最高的分配情况，进行下一次的分配。值得注意的是，上界算法一般能获得为最优值的200%以内解，即在第一次计算上界后，就能获得性能相对可以接受的解。

**5）整体算法**

|  |
| --- |
| 输入：智能体*N*，偏序*P*，时间预算 |
| 输出： 当前最优解*J\**，以及对应最小执行时间*T\** |
| 1 初始化树初始节点，以及生成树  2 初始化最优值，  3 while 或  *T\**  4 从中根据 选取分支  5 根据上述公式计算下界  6 if >  7 continue  8 else  9 计算上界,  10 If ,  11 拓展当前节点,将新节点加入搜索树中 |

## 其他分配方法

混合整数线性规划

整数规划，或者离散优化（Discrete Optimization），是指数学规划问题中自变量存在整数的一类问题。一个混合整数线性规划问题，首先目标方程和约束方程都是线性的，其次自变量既有连续变量，又有整数变量。与线性规划连续的可行域（可行解组成的集合）不同，整数规划的可行域是离散的。这里将设置一系列值域为0,1的布尔变量，每个布尔变量表示第*i*个智能体，是否在其第*j*个任务中，计划以功能*l*的形式执行子任务*k*。该方法存在两个主要缺点：一是计算复杂度和时间随问题规模呈指数增长；二是只能在最终得到解后才能处理，在生成最优解之前没有中间解，通常是通过MILP求解器，如CPLEX。这两个缺点阻碍了这种方法在实时应用程序中的使用。

**数学模型** 给定一个偏序集和对应的锚定函数，定义域为，其作用是给每个任务一个序号，即用来表示任务*i*,*j*，避免同类型的任务重复。如果，则意味着必须在之前执行。我们用来表示具有不同动作的智能体数量，作为感兴趣区域。目标是找到执行时间最小的任务分配方案。

进一步，我们给出这些问题的规范定义：

（3-7）

满足如下约束

·任务的时序约束

（3-8）

·为任务提供足够的服务（servers）

（3-9）

·智能体只能提供其拥有的服务

（3-10）

·智能体只能执行不超过一次任务

（3-11）

·智能体在任何时间只能执行不超过一次任务

（3-12）

·当智能体已经执行完次任务后，才可执行第次任务

（3-13）

·智能体需要遵守运动约束

（3-14）

（3-15）

通过上述约束条件，我们定义出混合整数线性规划问题，相关变量如表6.2所示。该问题复杂度为, 随着智能体数量和任务数量的增加，计算时间会显著增加，出现维数爆炸问题。

表6.2 变量定义

|  |  |  |
| --- | --- | --- |
| **变量** | **变量定义** | |
| **名称** | **定义** | **范围** |
|  | 偏序集合 |  |
|  | 智能体数量 |  |
|  | 智能体集合 |  |
|  | 分配任务集合 |  |
|  | 未分配任务集合 |  |
|  | 智能体分配的任务 |  |
|  | 未分配任务数量 |  |
|  | 分配任务完成时间 |  |
|  | 锚函数 |  |
|  | 智能体执行任务 |  |
|  | 智能体开始时间 |  |
|  | 智能体结束时间 |  |
|  | 任务持续时间 |  |
|  | 任务估计持续时间 |  |
|  | 任务需要的智能体数量 |  |
|  | 智能体的速度 |  |
|  | 从任务到任务的距离 |  |
|  | 从任务到任务的距离 |  |

局部搜索

局部搜索（Local Search）是离散受限优化问题的一种常见解法，它的主要思路是从一个初始解开始，在该结果的邻域内搜索下个解，然后重复此过程，直到当前解的质量无法再被提升。其中邻域的定义、搜索方法和解的质量评估根据不同问题的特点有很多不同的选择。常见的邻域是指在一个解的基础上只改变很少的部分，例如改变一个或少数几个子任务的分配。在邻域中的搜索方法可分为贪婪方法和随机方法，其中贪婪方法是指只有在被搜索到的解要优于当前解时，当前解才会被替代。这样做的缺点是解常常会陷入局部最优而非全局最优，为了解决这个问题，随机方法会随机的选择较差质量的解。另外，解的质量定义也不同，比如对约束条件的满足程度和对应的代价。当地搜索已被用来解决一些著名的组合优化的问题，例如一阶逻辑表达式的满足性问题，旅行商问题和资源调度问题。其最大的优点是能够快速得到可行解，并在允许时间内不断提高当前解的质量。相比之下，基于混合整数优化的方法往往需要等待商业解法器给出最优解，在此之前无法得到任何反馈。

在本问题中，局部搜索的核心有两个，一是获取当前分配解的最优值，这是一个简单的线性优化问题，二是定义当前分配解的领域。解的最优值我们通过求解一个凸优化函数来获得

（3-16）

对于一个任务分配情况，我们设智能体*i*对应的任务序列为。 任务开始时间满足偏序的时序约束

（3-17）

（3-18）

值得注意的是，上式是非凸的，在实际的凸优化代码中需要额外添加一个布尔变量来计算，在这一点上来说这不算一个严格的线性规划问题，但是此类情况出现的很少，大多数时刻少数的布尔变量不会影响计算速度。

智能体满足路径上的运动约束，即智能体在路径中移动的时间需要加入考虑中，这有两类约束，一类是从初始点移动到第一个任务的区域，第二类是从当前任务移动到下一个任务的执行地点，如下式所示。

（3-19）

（3-20）

解的邻域我们采用结合偏序集的方式，进行受限的领域拓展。一个邻域视为将一个任务从当前任务分配解中取出，并重新按照偏序的约束分配给所有可行的智能体。通常来说，由于智能体功能，偏序集等的约束，一个任务的领域解的数量大致为，其中*n*为智能体数量，*m*为任务数量。在实践中表明，若智能体集群和任务序列足够复杂，局部搜索方法很可能陷入局部最优解，使得最终只能得到次优解。

# 在线任务更新

## 扰动在线自适应

在非确定性的环境中，由于系统模型中经常存在扰动，如智能体可能会因为受到干扰而移动速度变快或变慢，某个子任务可能会提前或延迟完成，或者在执行任务过程中出现故障等，在初始状态下生成的最优计划在在线执行过程中可能会失效，也会导致整体任务无限制的推延。因此，在本节中，我们分析执行时间和路径变化等的不确定性，并针对此类不确定性提出在线自适应方法。

执行时间的不确定性会导致子任务的延迟或提前终止。若没有对任务进行适当的同步，执行后果可能是灾难性的。例如，一个协同任务在没有等到一个迟到的协作者的情况下直接启动，或者一个子任务在另一个子任务完成之前启动，这违反了偏序约束的协议。这些情况都会导致任务执行失败。为了克服这些可能出现的问题，我们提出了一个基于在线同步和分布式通信的自适应算法。

具体地说，考虑最优分配和智能体所被分配的任务方案。不失一般性，假设智能体刚刚结束执行任务，并且正转移到下一个子任务的指定区域。不论此次转移发生了何种延迟或者提早到达，都可以执行以下同步程序，以确保即使在执行和转移时长都不确定的情况下也能正确执行计划。

1）在任务执行之前：在开始执行任务之时，对于每一个满足偏序约束, 智能体将发送一个“开始”信息给智能体。这个信息意味着任务开始执行，因此对应的偏序约束满足，任务也可以开始执行。另一方面，对于每个满足, 智能体将会等待来自智能体的任务开始信息，在获得这个信息后才能执行任务。最后，对于每个任务具有,智能体将等待来着智能体的“停止”信号。这个信号意味着任务已经结束了，因此任务可以开始执行。

2）在任务执行时：若任务是一个协同动作，那么智能体将发送同步信息给每一个协作者来开始执行这个动作；若任务是局部的无协同的动作，那么智能体将直接开始执行这个任务。

3）在任务执行之后：在任务结束后，对于每一个满足约束, 智能体将发送“停止”消息给智能体。这个信息意味着任务已经完成因此可以执行。

上述过程如图4.1所示。

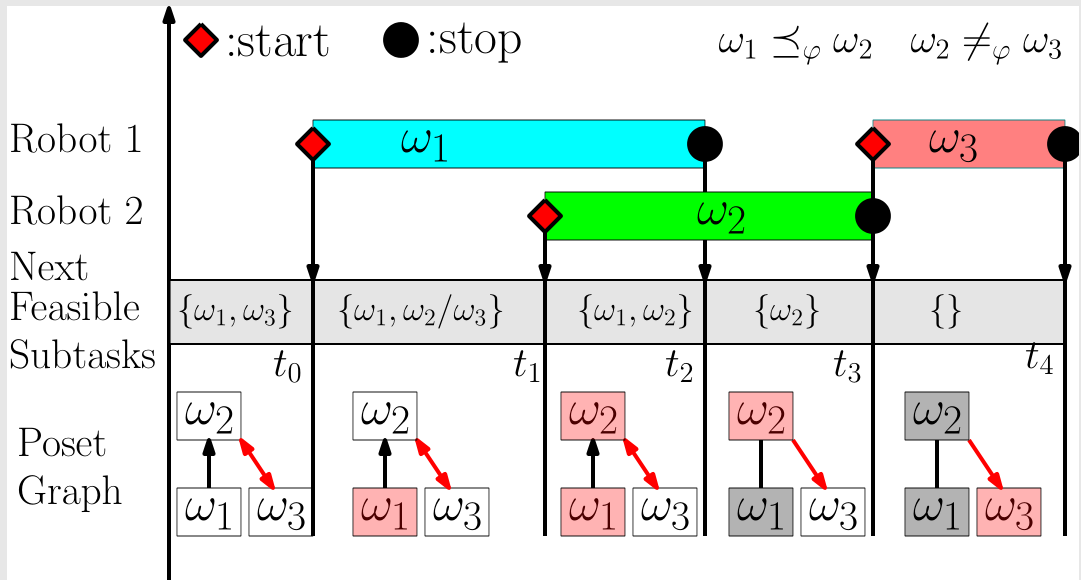


图4.1 整体规划流程

## 基于分支定界法的重规划

与环境扰动相比，智能体的故障将会造成更为严重的事故。因为，显而易见的，这个智能体所负责的一系列任务都无法继续执行了，这就需要一种更复杂的适应方法，将未完成的子任务需要重新分配给其他智能体，避免当前算法在执行过程中。假设智能体的最优方案为。在时刻，智能体发生故障，剩余的任务序列为 （4-1）

这部分信息会在损毁的时刻广播给全局。值得一提的时候，当智能体正在执行某个任务的时候损毁，则该任务也需要被重新分配，并重新完整的执行一遍。

对于这组子任务，最简单的恢复方法是招募另一个具有与相同能力的且完全空闲的新智能体，并接管其中的所有任务。然而，这并不总是可行的，也不是高效的，这意味着需要分配给团队中其他现有的智能体。这里，对之前提到的BnB搜索算法进行修改使之能够在当前执行的任务基础上重新分配。首先，BnB算法搜索树的初始根节点由已经被完成的子任务所组成，即

（4-2）

其中，是智能体在*t*时刻已经完成的子任务，在节点拓展过程中，考虑的智能体为，即损毁的智能体将在节点拓展过程中被除去。其次，节点拓展步骤将重新把所有未完成的任务分配出去：，其中是智能体未结束的任务，而且行的分配方案需要遵循相同的偏序约束。最后，根据相同的分支规则，和的计算下界和上界的方法，我们可以从同一个分支定界法算法框架下得到一个新的自适应的方案*J\**。需要说明的是，当有多个损毁的智能体时，可以对上述过程进行少量修改，例如，节点定义排除所有损毁的智能体而不只是一个。

对于复杂的战场环境，这样的重新规划可能不止要在智能体损毁的时候执行。在环境发生变化积累到一定程度后，需要对任务的特性重新进行更新，此时也需要对任务进行重新的规划。

# 参考文献

1. 刘振宏, 蔡茂诚. 组合最优化算法和复杂性: 组合最优化算法和复杂性; 1988.

2. Hochba, Dorit SJASN. Approximation Algorithms for NP-Hard Problems. 1997;28(2):40-52.

3. 刘学智. 生产调度问题. 2006.

4. 熊锐, 清华大学学报：自然科学版 吴J. 车间生产调度问题的技术现状与发展趋势. 1998;38(10):6.

5. 赵小强, 化工自动化及仪表 荣J. 流程工业生产调度问题综述. 2004;31(6):6.

6. 俞庆生, 林冬梅, 价值工程 王J. 多旅行商问题研究综述. 2012;31(2):3.

7. 殷剑宏, 吴开亚. 图论及其算法: 图论及其算法; 2003.

8. 田贵超, 黎明, 计算机仿真 韦J. 旅行商问题(TSP)的几种求解方法. 2006;23(8):5.

9. 霍佳震, 同济大学学报：自然科学版 张J. 求解配送\\收集旅行商问题的启发式算法. 2006;34(1):5.

10. 科技创新与应用 李J. 遗传算法(GA)在旅行商问题(TSP)中的应用. 2015(10):2.

11. 李建华, 计算机光盘软件与应用 李J. 多约束三维装箱问题的研究综述. 2012(17):3.

12. 曹先彬 刘, 王煦法 %J 小型微型计算机系统. 基于免疫遗传算法的装箱问题求解. 2000;21(4):3.

13. 田冉, 孙林夫, 王楠, 计算机工程 李J. 求解多卸载点车载装箱问题的多信息素蚁群算法. 2015.

14. 吴蓓, 丁文英, 杜彦华, 计算机集成制造系统 赵J. 基于重力装载的自适应随机算法求解多箱型三维装箱问题. 2020;26(11):3084-93.

15. 刘运龙, 高技术通讯 王J. 带权最大割问题的一种基于划分技术的固定参数可解算法. 2010;20(3):264-9.

16. 张爱君, 秦新强, 计算机应用 龚J. 求解最大割问题的多启动禁忌搜索算法. 2014;34(5):4.

17. 范宏, 电力自动化设备 韦J. 基于扰动KKT条件的原始-对偶内点法和分支定界法的最优潮流研究. 2004;24(5):5.

18. 张丽华, 电力系统保护与控制 韦J. 内点-分支定界法在最优机组投入中的应用. 2006;34(18):18-21.

19. 科技信息 潘J. 整数规划的分支定界法及其MATLAB实现. 2008(7):2.

20. 闫雪丽, 王学武, 华东理工大学学报：自然科学版 连J. 结合历史全局最优与局部最优的粒子群算法. 2011;37(4):6.

21. 汪镭, 控制与决策 吴J. 蚁群算法在连续空间寻优问题求解中的应用. 2003;18(1):5.

22. 蒋华伟, 郭陶, 杨震, 电子与信息学报 赵J. 基于离散鲸鱼群算法的物资应急调度研究. 2022;44(4):11.

23. 吴虎胜, 张凤鸣, 系统工程与电子技术 吴J. 一种新的群体智能算法——狼群算法. 2013;35(11):9.

24. 刘三阳, 张平, 控制与决策 朱J. 基于局部搜索的人工蜂群算法. 2014;29(1):6.

25. 云庆夏. 进化算法: 进化算法; 2000.

26. 汪祖柱, 安徽大学学报：自然科学版 程J. 求解组合优化问题的一种方法——分枝定界法. 2004;28(1):5.