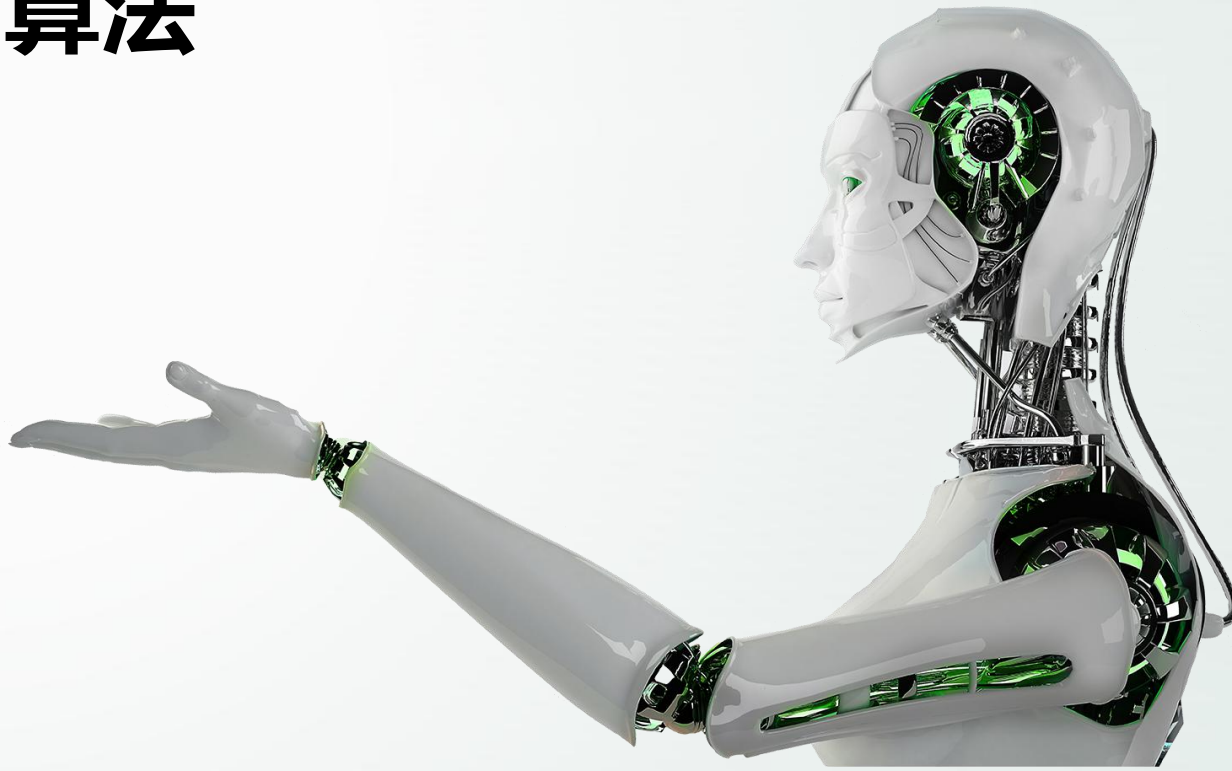
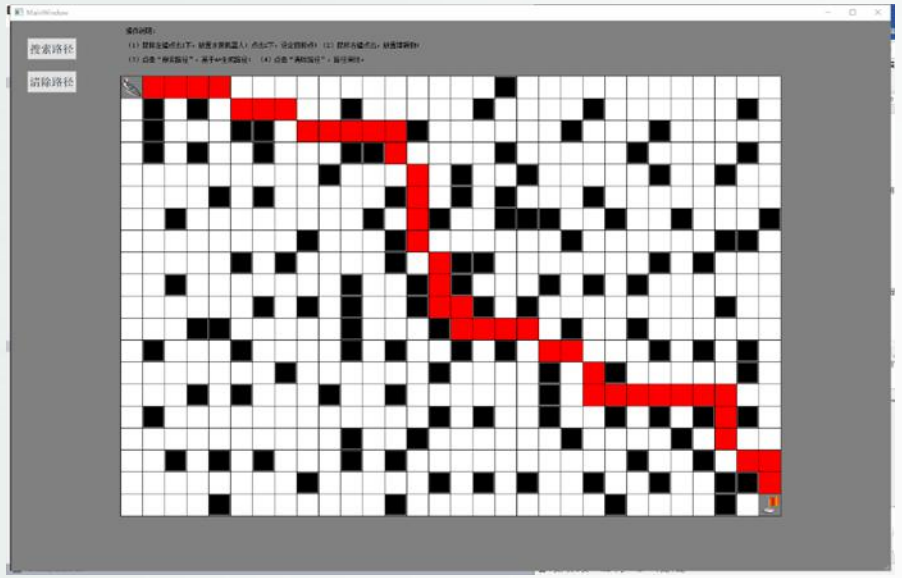
第四章

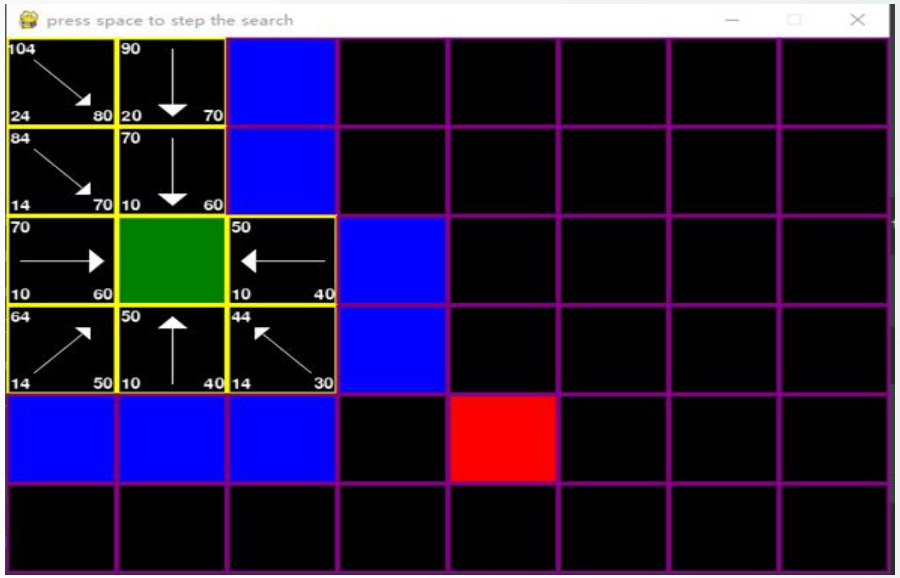
设计自己的全局规划算法

1. 作业点评
2. 全局算法设计思路
3. 模型建立和特征分析
4. 算法设计和仿真



作业点评



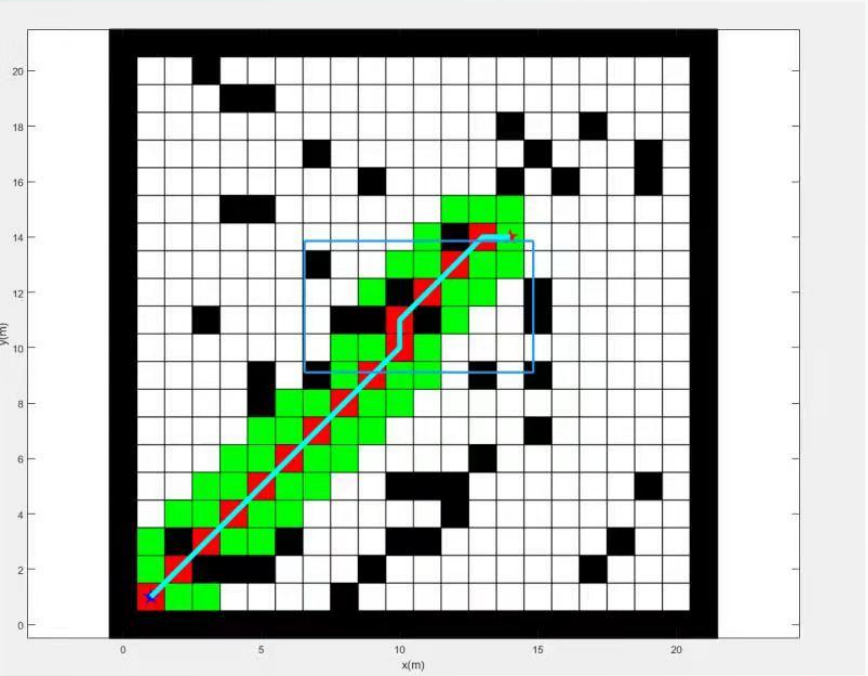


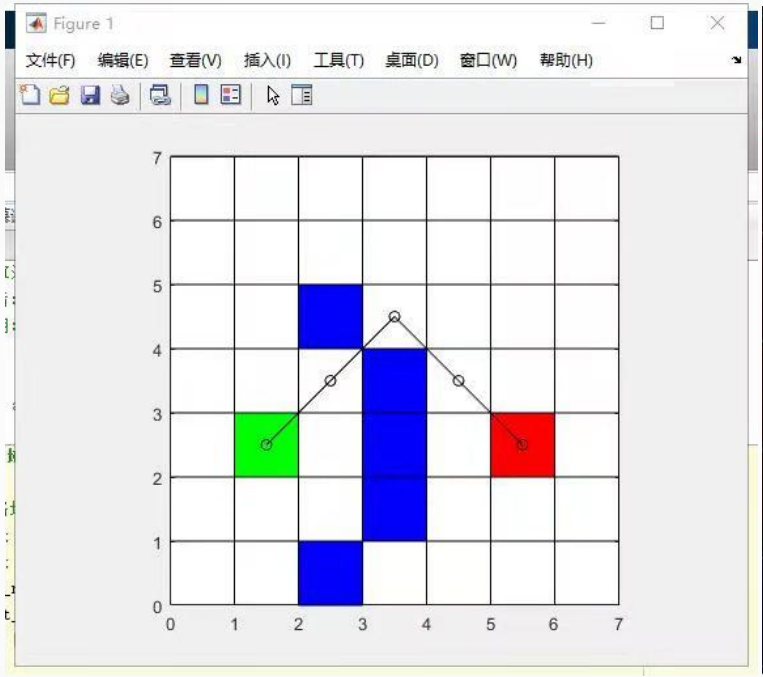
测试方便，设计逻辑清晰，路径对比色明显

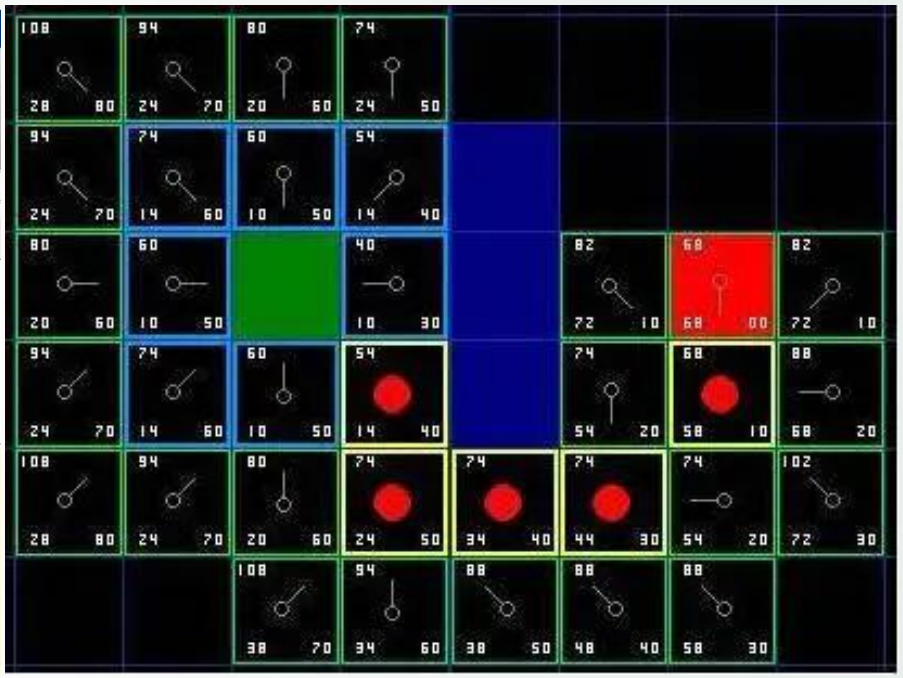
文档全面，运行简单，分步骤操作也便于学习算法流程，页面简洁。

用JS写的网页版，过程简洁明了，代码简洁，值得研究，分步骤求解，可用于学习。

作业点评

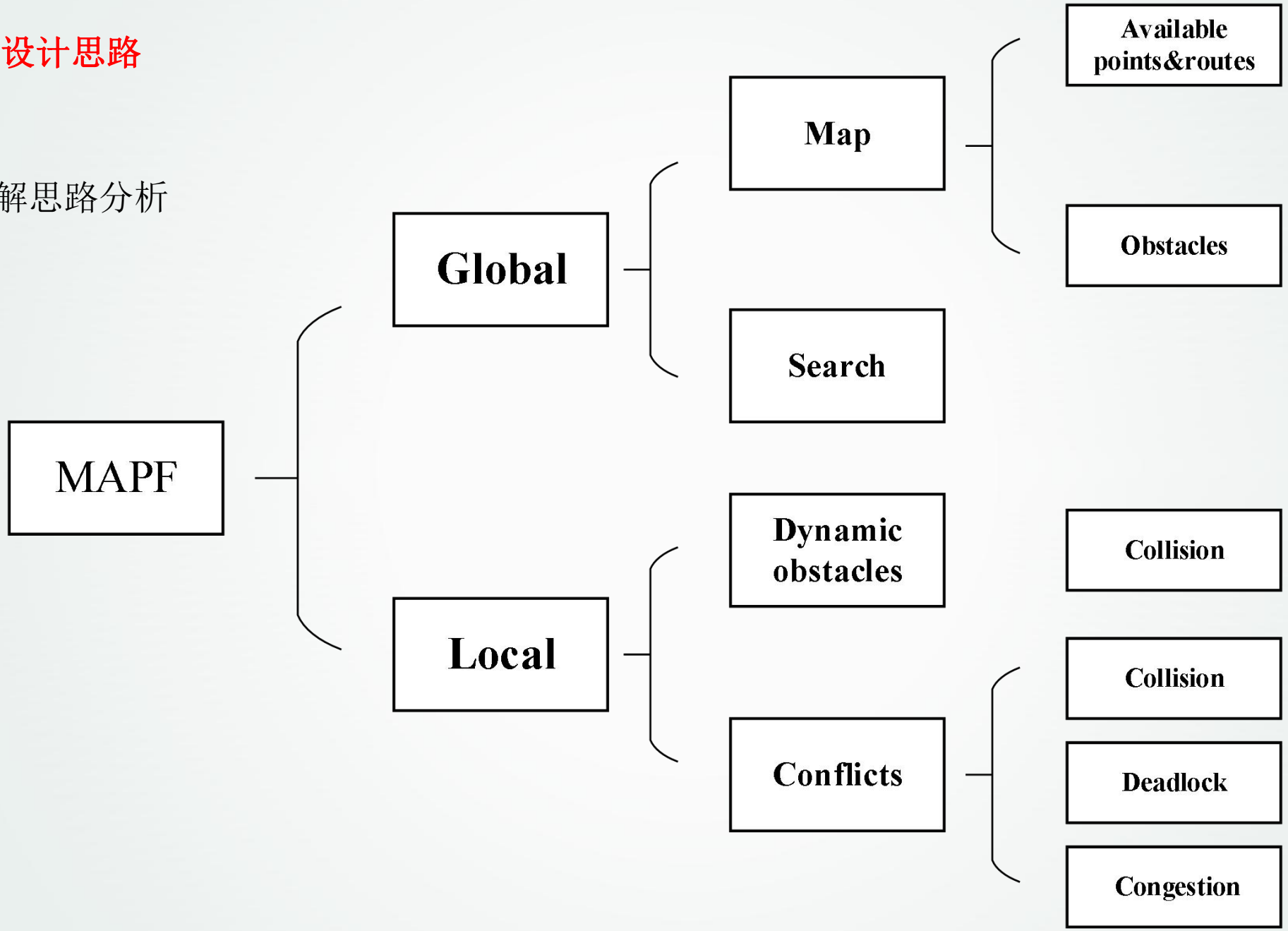






全局算法设计思路

MAPF求解思路分析



全局算法设计思路

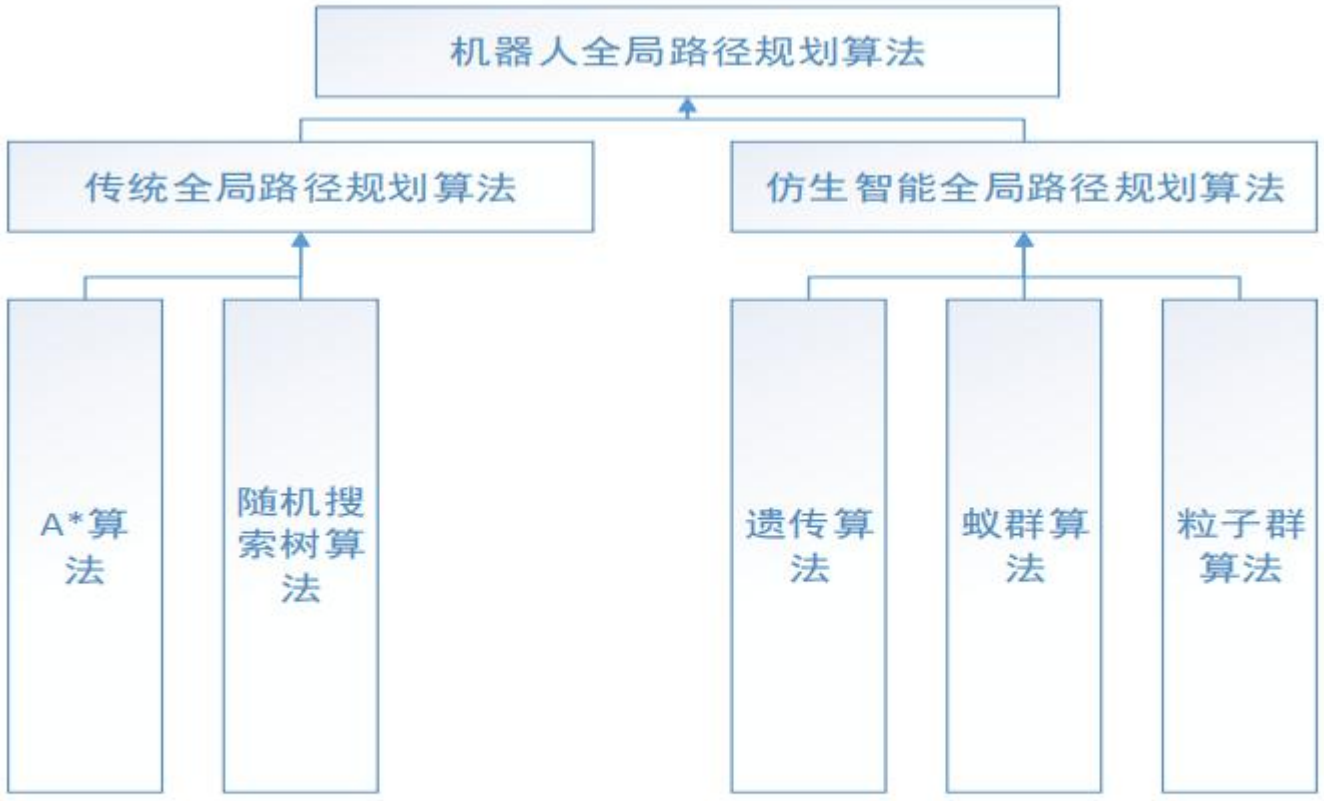


图 1 全局路径规划算法

Fig.1 Global path planning algorithm

系统模型的建立

结合任务分配（调度），得到起止点

设计算法搜寻路径

考虑冲突消除

全局算法设计思路

在第二章和第三章，我们分别讲解了A*算法和RRT算法求解迷宫问题。

但只是针对单个智能体进行路径寻优。

如何从单个智能体扩展到多智能体？

两个关键点 （以智能仓储为例）：

1. 需要和task scheduling/allocation 结合---目的是确定起始点及相应的目标点
2. 需要稍微考虑部分冲突的消除---相当于提前做一些局部避障措施

第 24 卷第 2 期
2 0 1 8 年 2 月

计算机集成制造系统
Computer Integrated Manufacturing Systems

Vol. 24 No. 2
Feb. 2 0 1 8

DOI:10. 13196/j. cims. 2018. 02. 013

智能仓库中的多机器人协同路径规划方法

张丹露[1],孙小勇[2,3],傅　顺[2],郑　彬[2+]

(1. 中国科学院大学 计算机控制学院,北京　101400;2. 中国科学院 重庆绿色智能技术研究所,重庆　400714;
3. 重庆德领科技有限公司,重庆　400714)

模型建立

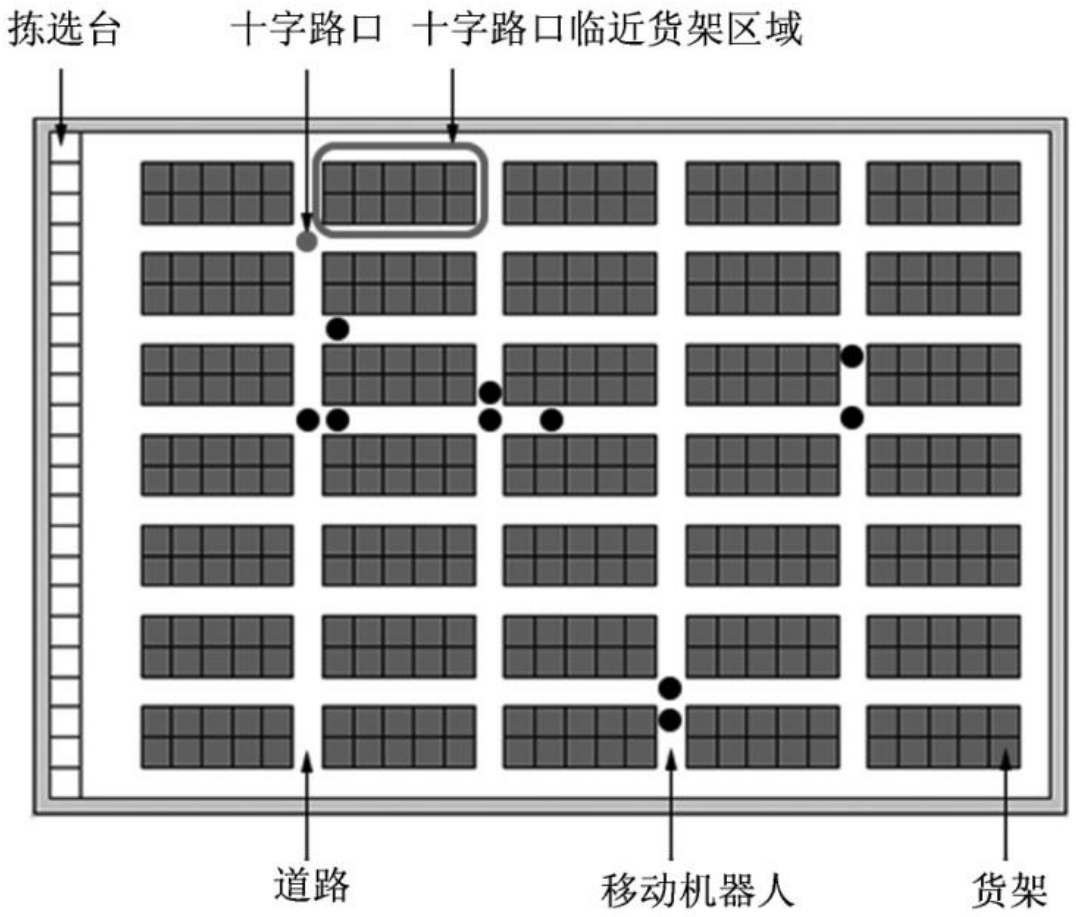


图1　智能仓库模型

1. 用栅格地图表示智能仓库环境

2. 位于栅格地图左侧的拣选台 S ＝ {S_1，…，S_m}（m 为拣选台的个数）

3. 栅格地图中由相邻栅格连续构成的路径，货架间的路径都只允许一台机器人通过，即为单行道，两条交叉的路径形成了一个十字路口

4. 由货架 L ＝ {l_1,…,l_e} (e为货架个数)构成的货架区域，每个货架区由两条横向的道路和两条纵向的道路包围。

5. 一群机器人 R ＝ {r_1，…，r_n}（n 为机器人的个数）以统一的速度v_{r_i}在仓库中行走并搬运货物

模型建立和特征分析

机器人系统控制方法采用集中式控制还是分布式控制？

特征分析

集中控制方法由中央控制系统统一处理机器人间的碰撞协调问题，并为机器人找到一个全局最优路径

优势：安全稳定并且容易实现，已被广泛用

缺陷：随着机器人数目的增长，计算复杂度将大幅提升而效率却逐渐降低

分布式控制方法是由机器人利用自己获得的实时信息为自己进行路径规划的方法

优势：适应快速增长的机器人数目和快速变化的仓库环境

缺陷：受通信水平、信息量等限制，只能为机器人找到局部最优路径，并且可能使机器人间发生碰撞、交通堵塞和死锁等问题

结合两者的优点，优势互补，融合使用是更为实用高效的控制策略

模型建立和特征分析

机器人系统控制方法采用集中式控制还是分布式控制？

结合两者的优点，优势互补，融合使用是更为实用高效的控制策略

特征分析

一方面通过集中式控制方式获得仓库的全局信息（如机器人的状态、位置等信息），并利用全局信息构建预约表和动态加权地图，为机器人实时提供智能仓库中的交通信息

另一方面，机器人系统采用分布式控制方式，每个机器人根据中央控制系统提供的信息自行进行路径规划

既减少了中央控制系统的负担，又为多机器人路径规划提供了全局信息，使机器人能更加动态地规划路径

特征分析

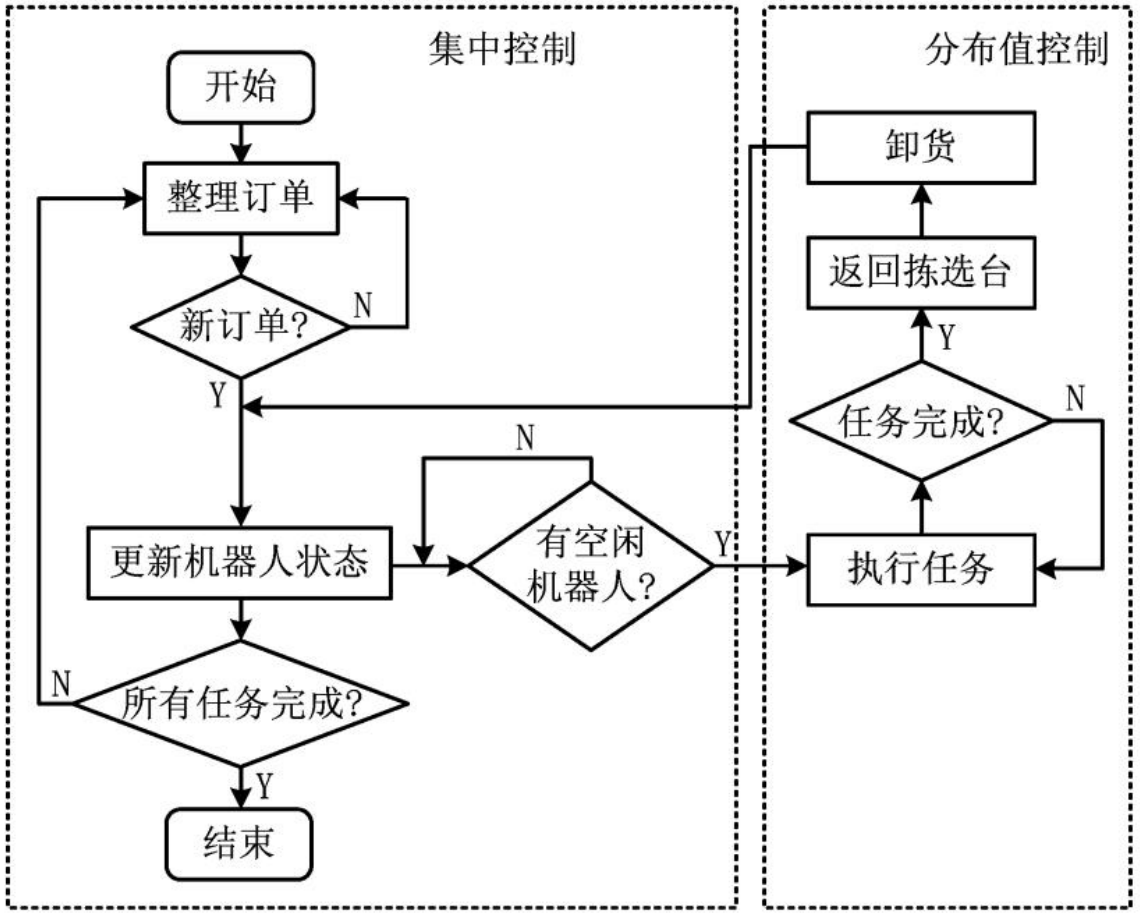


图2　任务流程

1. 订单到达，按照时间、位置、品类等分组。（订单预处理）
2. 订单派发至各拣选台，拣选台查询是否有空闲状态的机器人（订单下发）
3. 选择离货物位置最近的机器人执行任务（任务指派）
4. 机器人规划路径执行任务（路径规划）
5. 任务完成，机器人返回拣选台

系统模型的建立

结合任务分配（调度），得到起止点

设计算法搜寻路径

考虑冲突消除

交通规则

预约机制

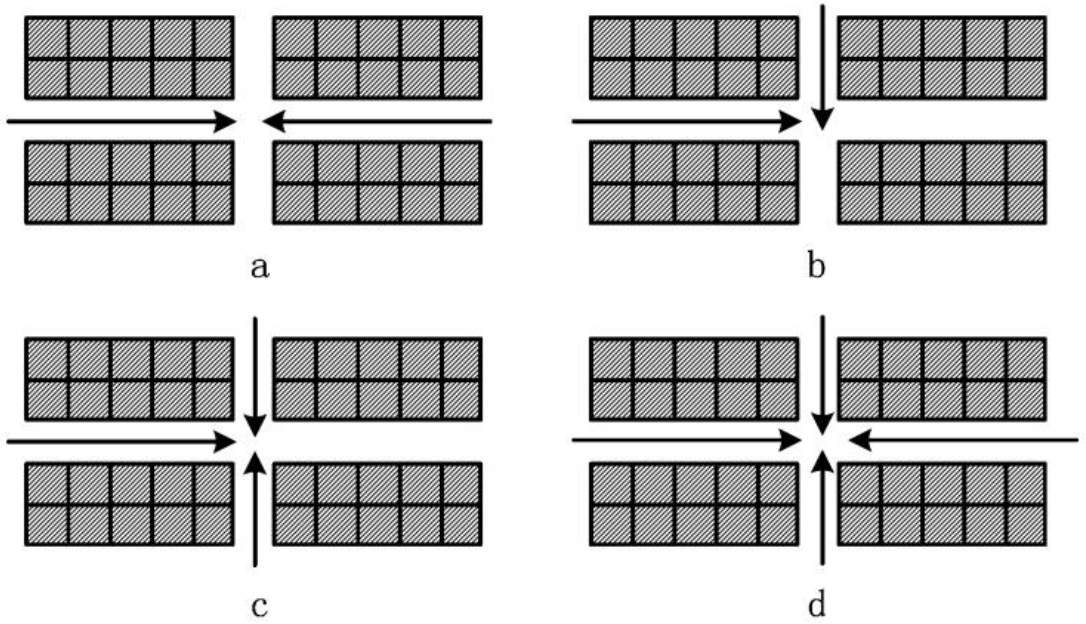


图3　几种碰撞类型

交通规则

1. 单行道规则
2. 相邻道路方向相反规则
3. 机器人不可逆行规则

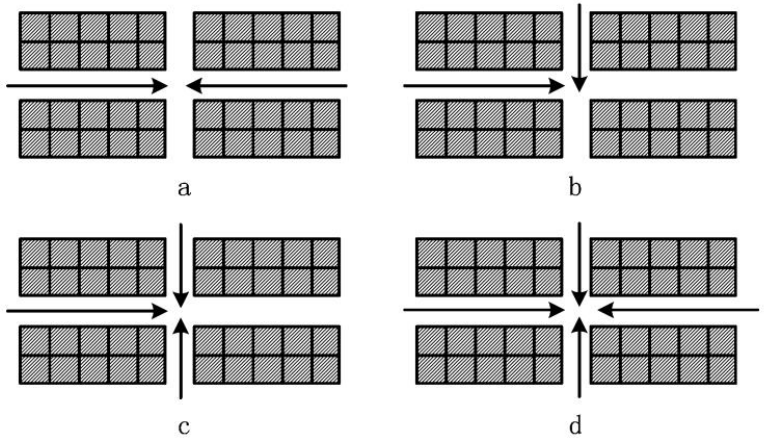


图3　几种碰撞类型

预约机制

1. 造成冲突的根本原因是同一时刻一个栅格最多只能被一个机器人占据
2. 设计与栅格地图大小相同的预约表，用于记录地图中每个栅格的状态
3. 从t时刻开始，每隔Δt更新一次
4. 如果栅格被占用，记录占用机器人ID，其他机器人查询当前的预约表来决策

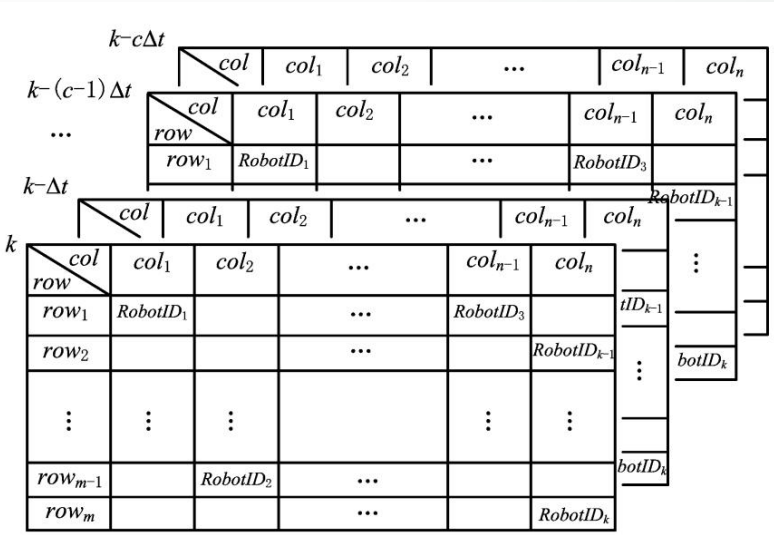


图4　不同时刻预约表

增强多agentA*算法

从开始栅格出发，机器人每次选择下一步将要扩展的栅格时，都会估算与自己所在栅格相邻的栅格路径代价。

得到周围栅格的估算代价后，将其放入一个带扩展节点候选表格，然后在表格中选择一个估算代价最小的栅格进行扩展。

机器人重复上述步骤，直到扩展到目标栅格。

$$f^*(n) = g(n) + h^*(n)$$

增强多agentA*算法

由于交通规则的限定，机器人只能根据智能仓库中的交通规则有序移动

(1) 从某一非十字路口栅格向十字路口栅格扩展。
(2) 从十字路口栅格扩展到另一个十字路口栅格。
(3) 从十字路口栅格扩展到目标栅格。

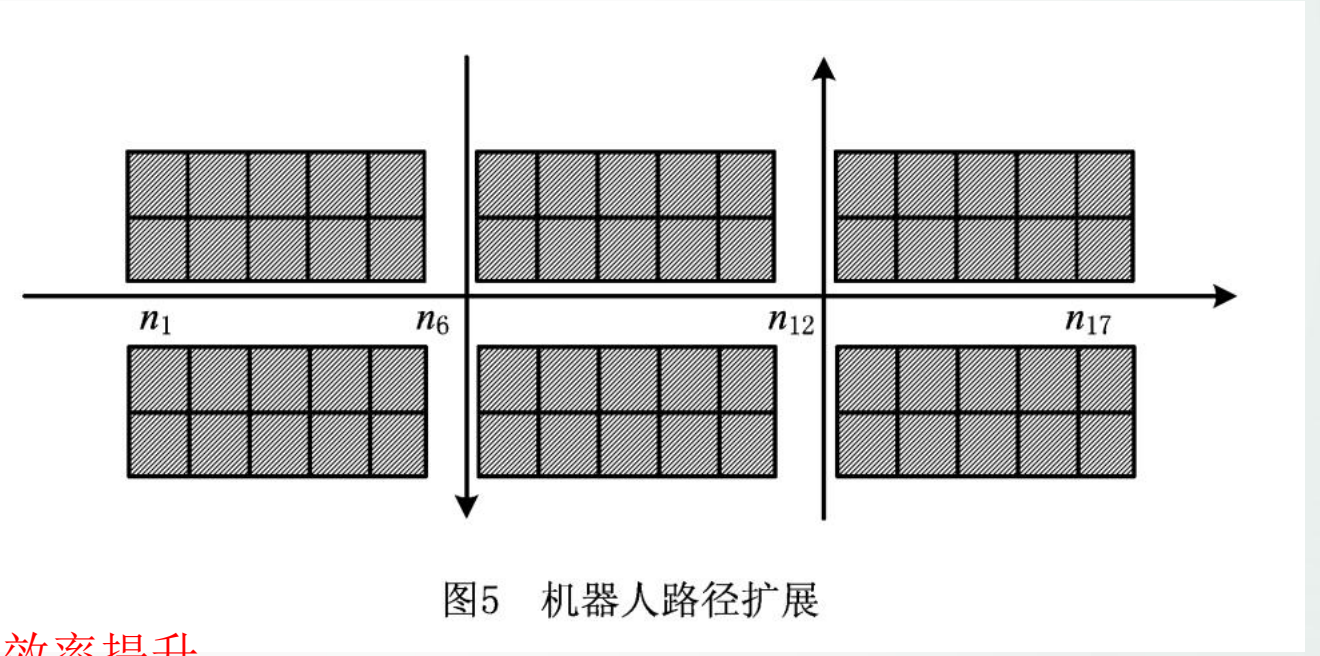


图5　机器人路径扩展

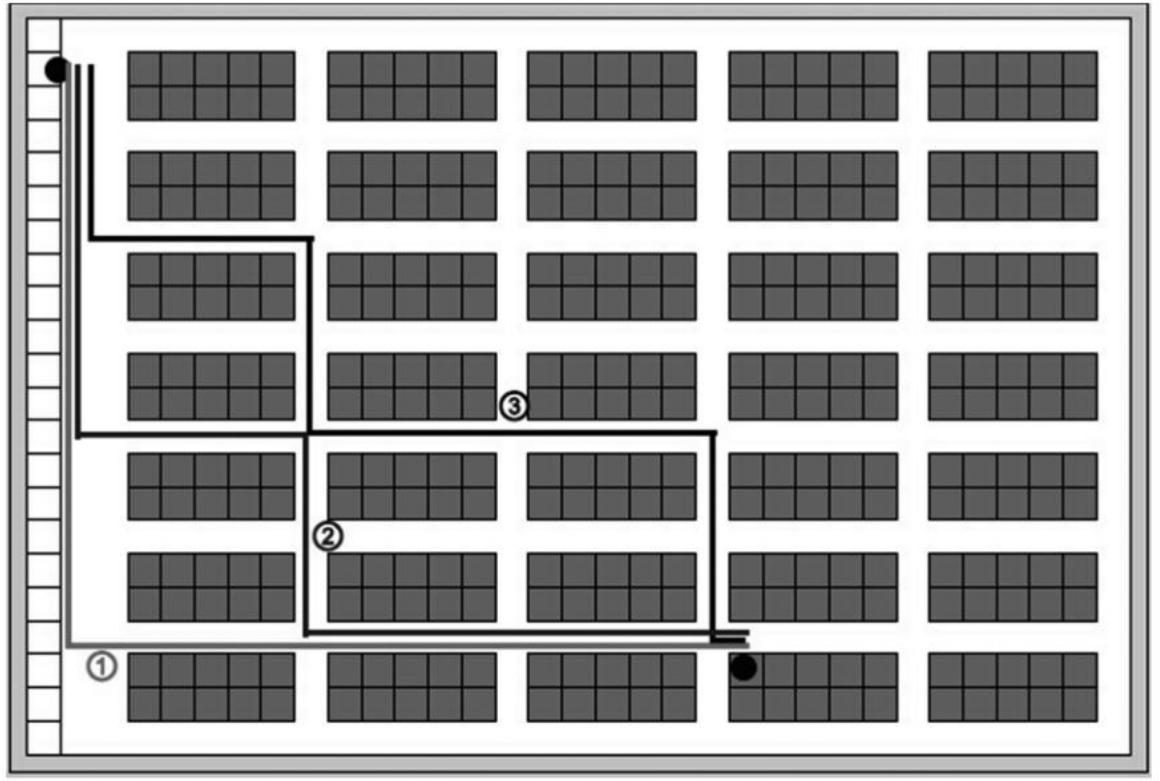
减少了无效搜索，效率提升

增强多agentA*算法

在真实的仓库环境中，机器人会在转角处花费更多的时间用于转弯，因此当机器人沿着相同路径长度的不同路径行走时会花费不同的时间。

$$f^*(n) = g(n) + h^*(n)$$

$$f^*(n) = g(n) + h^*(n) + \sum_{i=1}^{m} t_{\text{turn}}$$



增强多agentA*算法

目前为止我们设计了两处改进

1. 交通规则的加入，通过限定行进方向减少了无效搜索，缩小了搜索范围，提升了搜索效率
2. 结合工程实际，在A*原有的代价函数中加入累积转弯惩罚，进一步减小总体代价

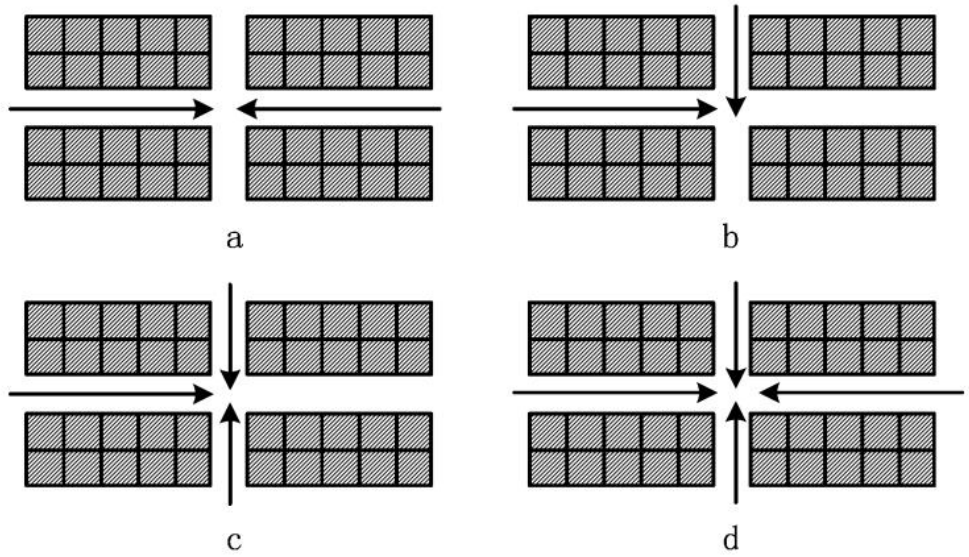


图3　几种碰撞类型

存在的问题

仍然无法避免3-b所示的碰撞

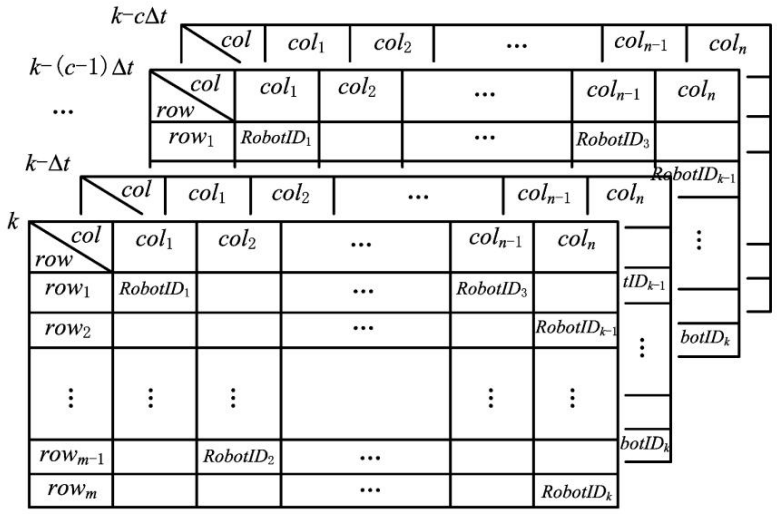


图4　不同时刻预约表

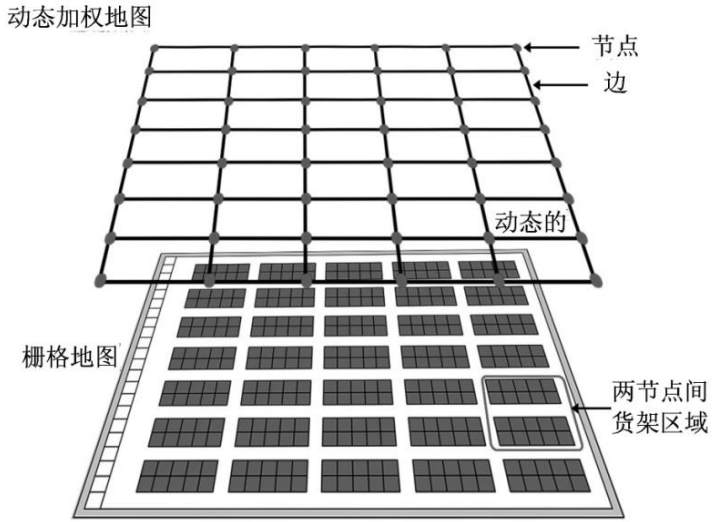


图7　动态加权地图

预约机制和动态加权地图

每个节点表示一个十字路口，两个节点之间的边表示十字路口之间的路径。连接两个节点的路径和路径两旁的货架组成两个节点间的货架区域，每两个节点间的货架区域和拣选台组成整个智能仓库。动态加权地图中的每条边表示一个相邻节点间的货架区域，每条边的权重随着货架区域拥塞程度的变化而变化，机器人可以根据权重动态规划路径来避让交通堵塞区域。

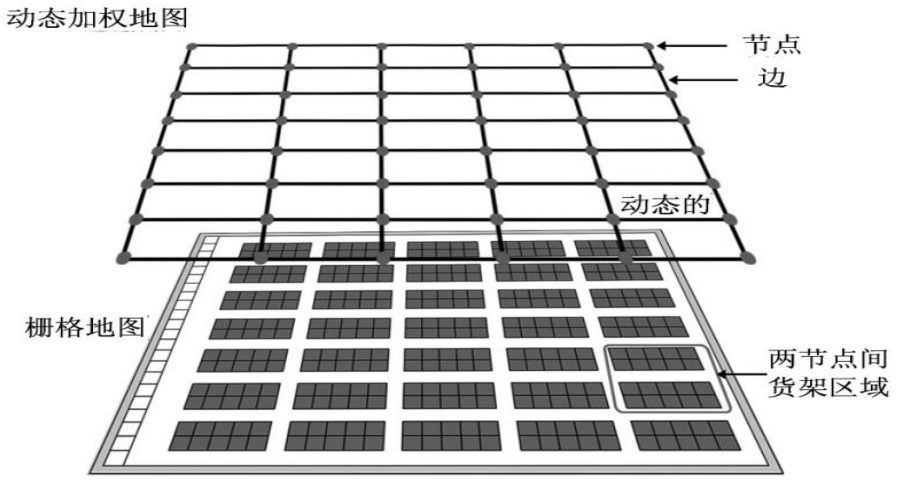


图7　动态加权地图

每隔一定时间 Δt，中央控制系统更新一次预约表，预约表中存储了某一时刻每个机器人在仓库中的位置信息，而中央控制系统则存储了从 k 时刻开始之前一段时间 $c \cdot \Delta t$ 内更新过的预约表（c 表示更新次数）。机器人在 k 时刻到达十字路口时，通过中央控制系统获得 $k-c \cdot \Delta t \sim k$ 时刻将要进入的货架区域内其他机器人的位置信息，通过对比该时间段内其他机器人在货架区域内的位置信息获得该区域内的交通堵塞情况。权重计算公式为

$$w_{\text{dynamic}} = N \cdot \frac{N_{\text{estimate}}(k - c \cdot \Delta t, k)}{N_{\text{real}}(k - c \cdot \Delta t, k)}$$

若机器人在某一货架区域内出现故障不能移动，则 N 的值为无穷大，否则为 1。$N_{\text{estimate}}(k-c \cdot \Delta t, k)$ 表示在交通顺畅的情况下，从 $k-c \cdot \Delta t \sim k$ 时刻，货架区域应该通过的机器人数量的估算值。首先机器人得到 $k-c \cdot \Delta t$ 时刻该货架区域内其他机器人的位置信息，估算经过时间 $c \cdot \Delta t$ 后在 $k-c \cdot \Delta t$ 时刻该区域的机器人应该通过的数目；然后机器人得到 $k-(c-1) \cdot \Delta t$ 时刻该货架区域内新进入的其他机器人的位置信息，估算经过时间 $(c-1) \cdot \Delta t$ 后，从 $k-(c-1) \cdot \Delta t$ 时刻以后进入该区域的机器人应该通过的数目，以此类推，直到估算 $k-\Delta t \sim k$ 时刻应该通过的机器人数目；最后，每个时间段估算值的总和为 $k-c \cdot \Delta t \sim k$ 时刻货架区域应该通过的机器人数量的估算值。$N_{\text{real}}(k-c \cdot \Delta t, k)$ 表示 $k-c \cdot \Delta t \sim k$ 时刻，该货架区域内的真实通过的机器人数目，当 $N_{\text{real}}(k-c \cdot \Delta t, k)=0$ 且 $N_{\text{estimate}}(k-c \cdot \Delta t, k) \neq 0$ 时，估算值 w_{dynamic} 为无穷大。

增强多agentA*算法

机器人先通过栅格地图和改进 A* 算法进行路径规划。机器人每进入一个十字路口均需查看最新的动态加权地图。如果预计通过货架区域在地图中对应边的权重为 0 或 1，则表示该区域没有交通堵塞，可以继续通过该区域；否则，机器人将利用动态加权地图估算通过该区域的额外时间：

$$t_{\text{extra}} = w_{\text{dynamic}} \cdot t_{\text{wait}}$$

$$f'^{*}(n) = g(n) + h'^{*}(n) + \sum t_{\text{turn}}$$

$$h'^{*}(n) = \frac{d_n}{v_{r_i}} + t_{\text{extra}}$$

$$f^{*}(n) = g(n) + h^{*}(n)$$

$$f^{*}(n) = g(n) + h^{*}(n) + \sum_{i=1}^{m} t_{\text{turn}}$$

$$f'^{*}(n) = g(n) + h'^{*}(n) + \sum t_{\text{turn}}$$

增强多agentA*算法

目前为止我们设计了三处改进

1. 交通规则的加入，通过限定行进方向减少了无效搜索，缩小了搜索范围，提升了搜索效率
2. 结合工程实际，在A*原有的代价函数中加入累积转弯惩罚，进一步减小总体代价
3. 预约表和动态加权地图的设计，赋予机器人提前感知十字路口交通状况的能力，提前规避可能出现的拥堵和碰撞。

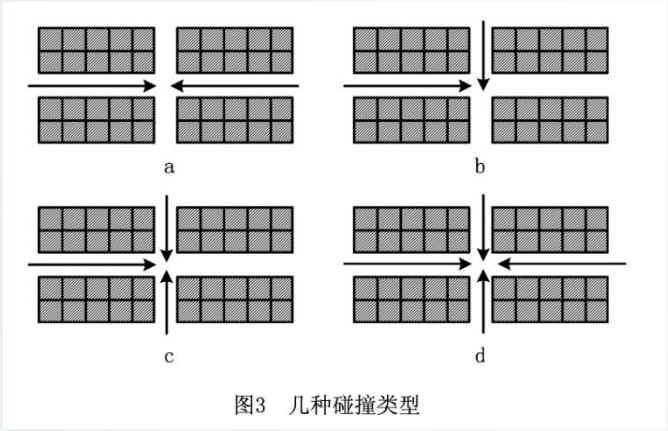


图3　几种碰撞类型

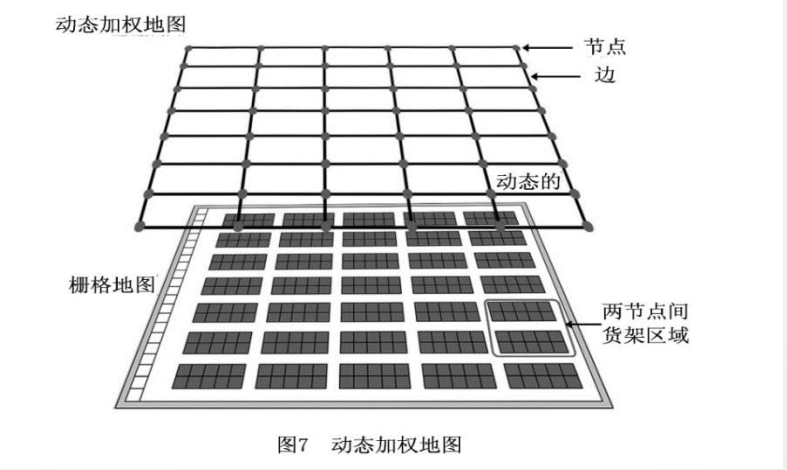
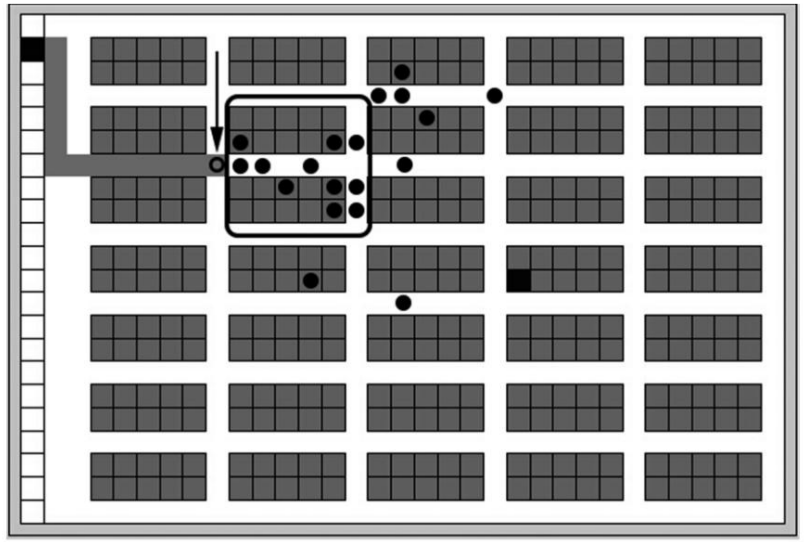
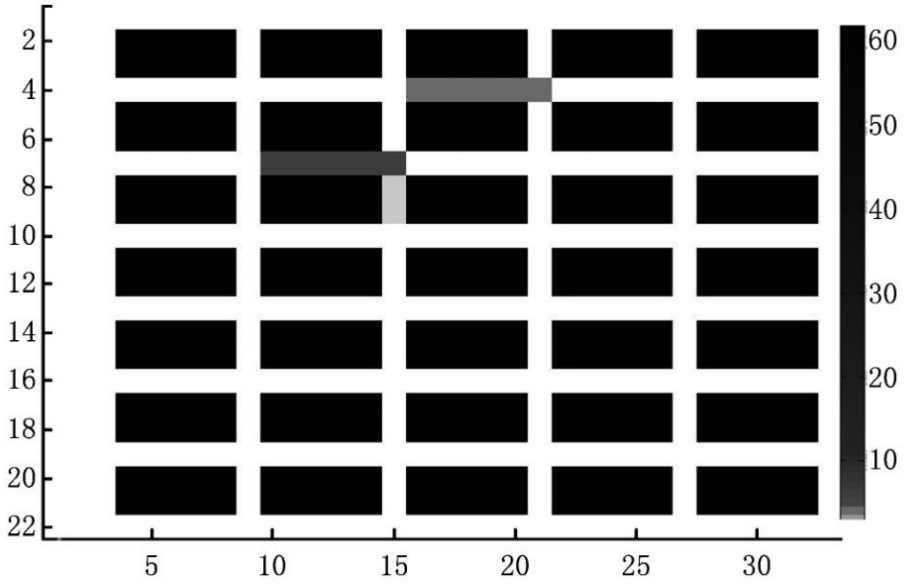


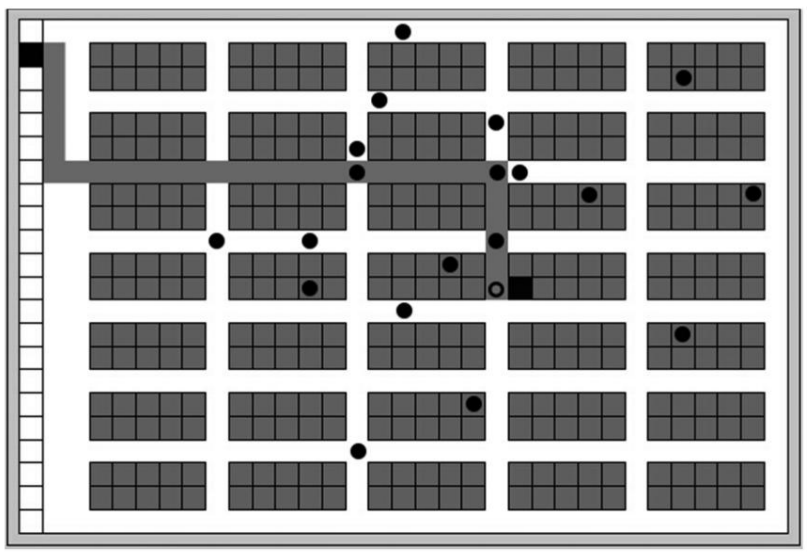
图7　动态加权地图

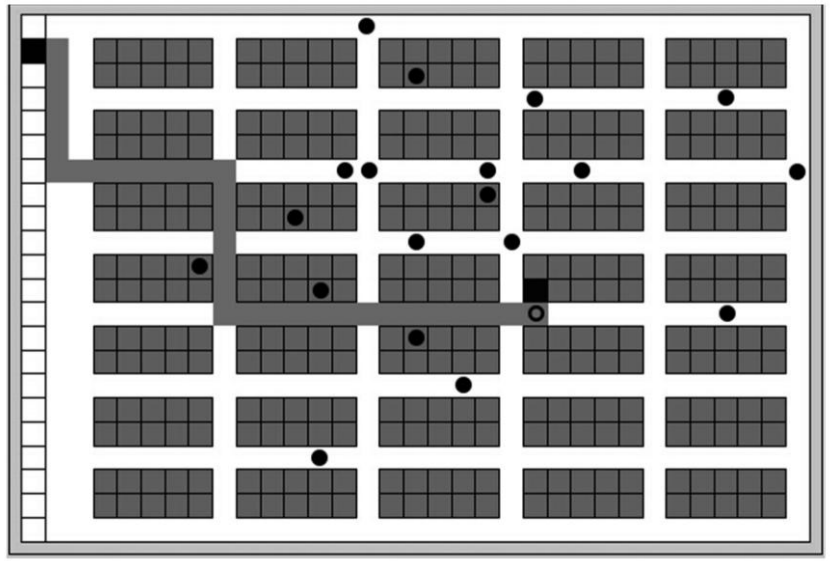
增强多agentA*算法



案例展示







算法对比

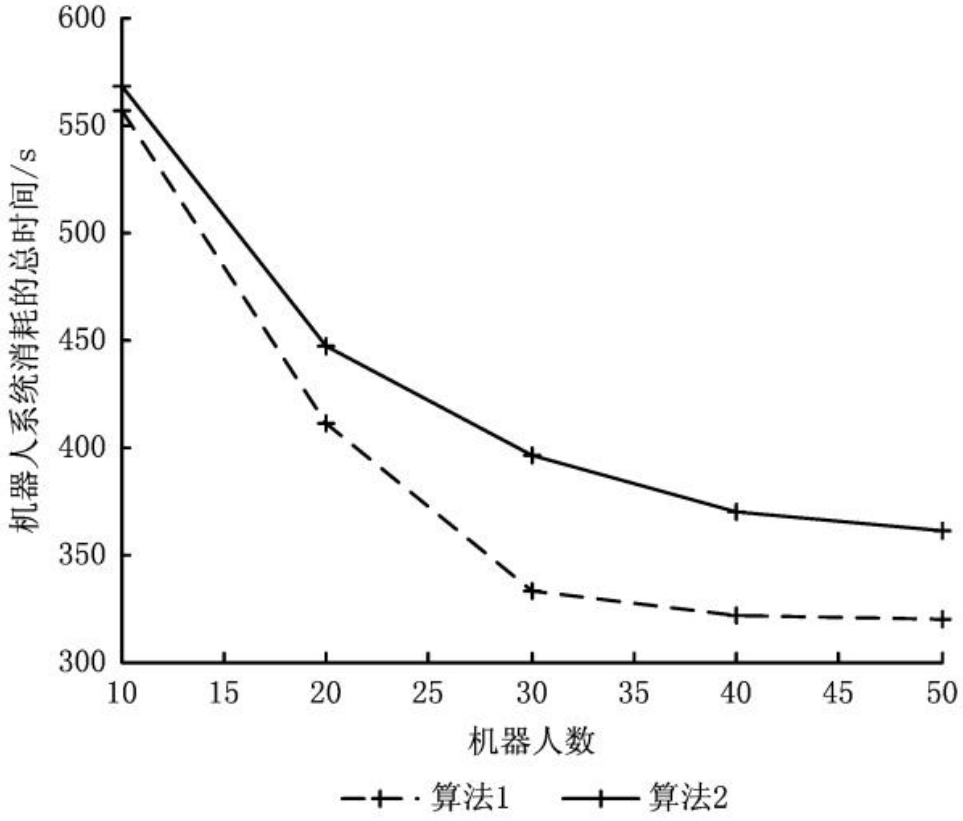


图12　机器人系统执行任务消耗的总时间

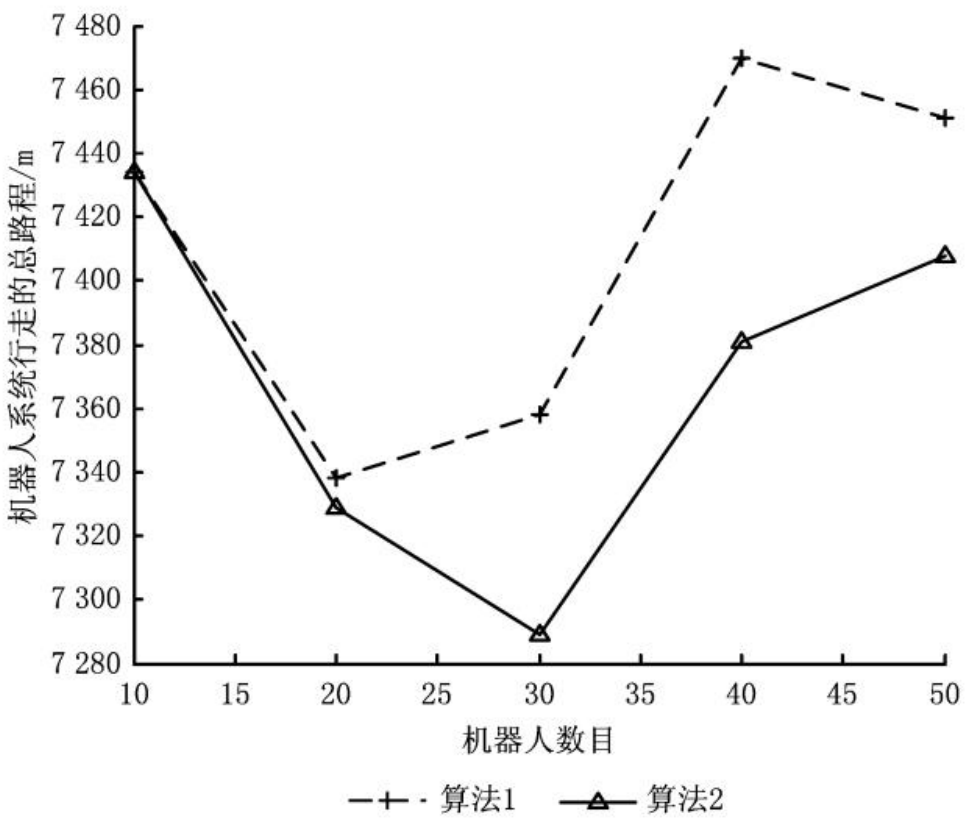


图13　机器人系统执行任务行走的总路程

不同数目机器人的多机器人搬运300个货物

算法对比

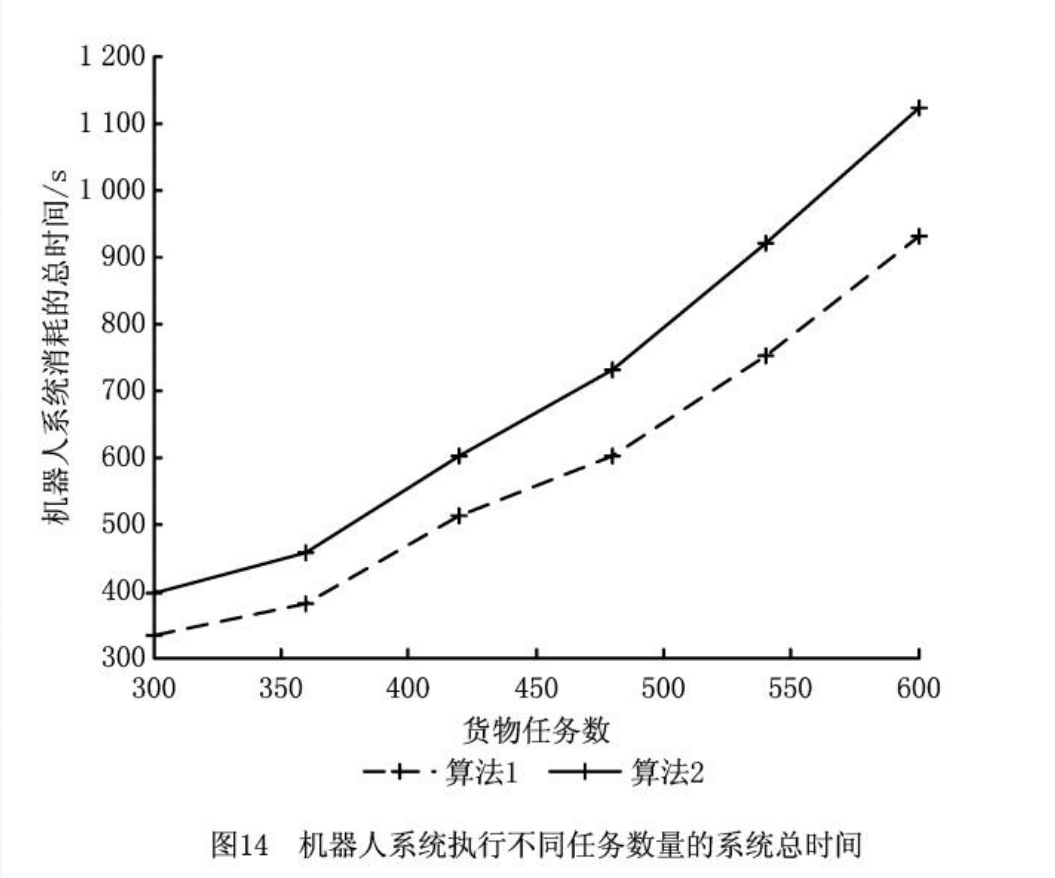


图14　机器人系统执行不同任务数量的系统总时间

固定机器人数目为30，执行不同数目的任务

总结

MAPF全局算法的设计较为简单，局部优化才是MAPF的重中之重。通过参考文献压缩包中的几篇学位论文可以明显感觉到，虽然大多数论文中都用了一定篇幅介绍全局算法，但其重点还是在局部。

本章作业

1. 阅读本章讲解案例的文章
2. 阅读参考文献压缩包中的其他文献
3. 有能力的学员可尝试复现本章算法