MAPF 基础

文档作者：李 拥 祺

1. MAPF 是什么?

multi-agent pathfinding problem

1.1 定义

Multi-Agent Path Finding (MAPF) is the problem of computing collision-free paths for a team of agents from their current locations to given destinations

- MAPF 问题是多智能体规划问题中很重要的一类问题；
- 解决的问题：
  - 解决多个智能体局部路径冲突问题；
  - 传感器可以进行避障，但是如果能提前优化路径避免或降低碰撞的概率显然是更好的选择。
  - 举个例子：人在开车的时候总是会有**提前预判**危险从而规划出一条路径避免危险，这也是老司机与新手司机的一大区别。
- 核心部分：
  - 全局路径规划
  - 局部路径优化

1.2 经典 MAPF 问题数学描述

1.2.1 问题描述

k 个 agent

- **Input**： tuple $< G, s, t >$
  - 无向图 $G = (V, E)$
  - 映射 s : $[1, \cdots, k] \to V$ 表示每个 agent 都对应一个源节点；
  - 映射 t : $[1, \cdots, k] \leftarrow V$ 表示每个 agent 对应的目标节点；
- **假设** 时间是离散的，在每一个时间步长，每一个 agent 都分别位于图 G 上一个节点上，并且可以做一个动作。
- 动作 $a : V \to V$ 可以认为是一个函数，如 $a(v) = v'$ 表示节点 v 上的 agent 在一个时间点执行了动作a 之后，在下一个时间点位于节点 v' 。一般认为有两种类型的动作：
  - `move`： 表示一个 agent 从它现在的节点 v 移动到图上的相邻节点 v'；
  - `wait`：表示一个 agent 在一个时间步长不移动，即在下一个时间点任然待在现在的节点 v。
- $\pi_i[x]$ 表示一个 agent i 执行了动作序列 $\pi = (a_1, a_2, \cdots, a_n)$ 中前 x 个动作之后达到的位置，即 $\pi_i[x] = a_x(a_{x-1}(\cdots a_1(s(i))))$ ，其中 $s(i)$ 表示 i 的源节点位置。
- $\pi_i[|\pi|] = t(i)$ 表示 agent i 执行了一系列动作 π 之后到达目标节点 $t(i)$ ，我们称动作序列 π 是 agent i 的一条单智能体规划路径（**single-agent plan** ）。
- **Output**： k 条 single-agent plan 集合。

1.2.2 冲突类型

- 我们以 π_i 和 π_j 两条 single-agent plan 来说明冲突的类型

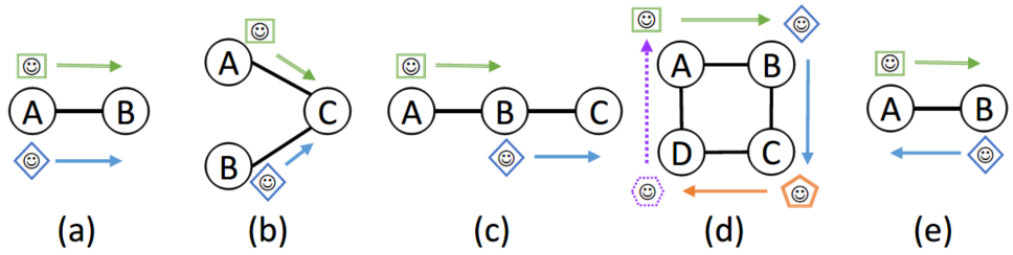

Figure 1: An illustration of common types of conflicts. From left to right: an edge conflict, a vertex conflict, a following conflict, a cycle conflict, and a swapping conflict.

  - 节点冲突（Vertex conflict）

    $\pi_i[x] = \pi_j[x]$

    在同一时间点，一个节点上 agent 的数量大于等于2个。

  - 边冲突（Edge conflict）

    $\pi_i[x] = \pi_j[x]$ 并且 $\pi_i[x+1] = \pi_j[x+1]$

    在同一时间步长里，多个 agent 以同一个方向通过相同的边。

  - 跟随冲突（following conflict）

    $\pi_i[x+1] = \pi_j[x]$

    一个 agent 下一个时间点要占据的节点是另一个 agent 现在占据的节点。

  - 环绕冲突（cycle conflict）

    $$\pi_i[x+1] = \pi_{i+1}[x], \quad \pi_{i+1}[x+1] = \pi_{i+2}[x], \cdots, \pi_{j-1}[x+1] = \pi_j[x], \quad \pi_j[x+1] = \pi_i[x]$$

    多个 agent 之间存在跟随冲突，并且构成了一个环状。

  - 换位冲突（swapping confllict）

    $\pi_i[x+1] = \pi_j[x]$ 并且 $\pi_j[x+1] = \pi_i[x]$

    两个 agent 在一个时间步长里互换了位置，有些文献中将换位冲突归类于边冲突。

    换位冲突是环绕冲突的一个特例。

- 如果能给每一个 agent 都找到一条 single-agent plan ，并且在执行过程中每个 agent 之间不会发生**冲突**，我们就认为解决了这个 MAPF问题。
- 这些冲突之间的关系：
  - 禁止节点冲突意味着边冲突也被禁止；
  - 禁止跟随冲突意味着环绕冲突和换位冲突都被禁止了；
  - 禁止环绕冲突意味着换位冲突被禁止了；
  - 允许边冲突意味着节点冲突被允许了；
  - 允许换位冲突意味着环绕冲突被允许了；
  - 允许环绕冲突意味着跟随冲突被允许了；
- **针对不同的问题，禁止/允许的冲突会不一样，看论文的时候要注意。**

1.2.3 智能体到达目标点之后

因为不同的 agent 到达自己的目标点的时间不同，所以需要定义 agent 达到目标点之后的状态，一般采用以下两种假设：

- 待在目标点不动

  假设每个 agent 到达目标点之后都在目标点静止不动直到所有 agent 到达目标点，但是这种假设有一个问题就是，这些到达目标点的 agent 实际上变成了**障碍物** 。

- 在目标点消失掉

  假设每个 agent 到达目标点之后立即消失掉。

1.2.4 目标函数

一般采用目标函数来评估一个 MAPF 问题的解决方案 $\pi = \{\pi_1, \cdots, \pi_k\}$，最常用的两类目标函数：

- 完成时间（makespan）：最后完成的那个 agent 所花时间步长

  $\max\limits_{1 \leq i \leq k} |\pi_i|$

- 总成本（sum of costs）：各个 agent 完成的时间步长之和

  $\sum\limits_{1 \leq i \leq k} |\pi_i|$

1.3 经典 MAPF 问题变形

经典 MAPF 问题一般具备以下假设：

1. 将连续时间离散化为时间步长；
2. 每个时间步长执行一个动作；
3. 在每个时间步长里，每个 agent 只占据一个节点；

针对这些假设的改变产生了许多 MAPF 问题变种，具体可以看参考资料1 。

1.4 MAPF 与 多机器人系统路径规划

- 理论到工程的转化，多机器人系统路径规划是 MAPF 的一个应用案例；
- MAPF 是多机器人路径规划的理论基础；
- 多机器人系统路径规划主要是**删除理想化的约束**，在学术研究中，很多理想化假设何约束是为了方便测试；
- 工程中需要考虑各种情况，如 1）突然某一个机器人坏了；2）在线动态调度突然变成了离线无法通信；3）环境中突然闯入的人；4）机器人还需要考虑其特性，不能简单的当作一个 agent 等

1.5 应用场景与研究方向

- 应用场景：
  - 仓储系统
  - 机场自动调度
  - 车联网
  - 港口集装箱调度
  - $\cdots$
- 研究方向：
  - 改进已有算法；
  - 将 MAPF 应用到具体生活场景中，考虑实际约束；

1.6 研究团队

http://mapf.info/index.php/Main/Researchers

2. MAPF 算法

MAPF 问题是一个 NP-hard 问题

- 明白了 MAPF 问题的定义之后，很自然就会想到一种解决方案：
  - 给每一个 agent 单独规划路径；
  - 利用一些简单的规则来避免冲突；

  这种方案在简单的系统中可以工作的比较好，但是随着问题规模的增大，效率会降低，如果可以将所以 agent 一起考虑进行规划将大幅度提高效率，而这类方法主要有以下几种：
- 基于搜索的方法（**参考资料2-3**）

表 1　最优多智能体路径规划算法对比

Table 1　Comparison of optimal MAPP algorithms

算法分类	优点	缺点	适用问题规模	适用问题特点
A* 搜索算法	实现简单，在智能体密集问题中表现良好	时间代价和空间代价高，速度最慢	中小规模（2 个 ~ 30 个智能体）	在智能体密度较高时效果较好，密度越高效果越好，对密度敏感性较低
代价增长树搜索算法	实现比较简单，分为 2 层搜索，速度较快	速度偏慢，高层次搜索冗余度高	中等规模（2 个 ~ 60 个智能体）	在智能体密度较低时效果好，较高时计算代价高
基于冲突搜索的算法	速度相对较快，求解问题规模较大	实现难度略高	中等规模（2 个 ~ 60 个智能体）	适用于智能体稀疏问题，在加入合并操作后适用于任意密度问题
规约算法	速度一般很快，只要完成证明就可以找到高效的求解器	规约证明需要极强的数理功底	中等规模（2 个 ~ 60 个智能体）	由规约后的模型特点决定

  - Reduction-based Optimal Solvers
  - A*-based Optimal Solvers
  - Increasing Cost Tree Search
  - Conflict Based Search(CBS)

  基于搜索的方法每一类里面都有很多算法，以及改进算法。
- 近似算法（**参考资料3**）
  - 有边界

表 4　有边界次优的多智能体路径规划算法对比

Table 4　Comparison of bounded suboptimal MAPP algorithms

算法分类	引入次优性的方式	优缺点	求解问题规模
A* 类算法	修改启发式函数	求解速度最慢，但是易于实现	针对 60 个 ~ 120 个智能体的问题时效果较好
基于代价增长树搜索的算法	不存在经典问题下的有边界次优算法	—	—
基于冲突的搜索类算法	在高层次搜索和低层次搜索中都可以修改启发式函数	需要根据经验来确定合并的阈值	目前在 120 个 ~ 200 个智能体的问题中比较适用
基于规约的算法	由规约后问题的有边界次优算法引入次优性	规约证明困难，优缺点由规约后采用的算法决定	由规约后采用的算法决定

  - 无边界

表 3　各类无边界次优的多智能体路径规划算法对比

Table 3　Comparison of each kind of unbounded suboptimal MAPP algorithms

算法分类	优点	缺点	适用问题的特点
基于搜索的算法	易于实现，结果质量一般较好	求解速度较慢，一般无法保证完整性	适用于智能体数量较少的问题，在智能体密度较低时效果较好
基于规则的算法	求解速度很快	求解的结果可能比最优解差很多，只有在特定的地图上才有完整性	能够求解大规模智能体问题，但是在对结果质量要求较高时不适用
基于规约的算法	可以利用已有的一些求解算法	规约的证明比较困难	规约的模型决定算法适合求解问题的类型
其他算法	求解结果的质量较好	无法保证完整性	能够求解大规模智能体问题

参考资料

1. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks
2. Search-Based Optimal Solvers for the Multi-Agent Pathfinding Problem: Summary and Challenges
3. 多智能体路径规划研究进展
4. 睿慕课《多机器人系统路径规划技术与实践》课程
5. https://www.bilibili.com/read/cv10556167 李娇阳