A*算法简介

A*算法最早由美国计算机科学家彼得·哈特和尼尔·库斯拉（Peter Hart and Nils Nilsson）于 1968 年提出[1]，它是一种应用范围较为广泛的启发式搜索算法，常应用于路径规划[2]。A*算法在 Dijkstra 算法的基础下进行了改进，结合了 Dijkstra 算法和贪心最优搜索算法的思想[3]，融合了两者的目标函数，即使用从起点到目标的实际距离以及到目标的估计距离作为其目标函数。根据这个估值函数，将启发值最小的点作为下一起点，不断迭代，从而在起始点与终点之间找到一条最优路径，是一种两全其美的算法。当贪心最优搜索算法找到正确的路径时，A*算法也能找到正确的路径，并探索相同的域；当贪心最优搜索算法找到错误的路径，即更长的路径时，A* 算法依旧会找到正确的路径，就像 Dijkstra 算法一样，但探索的区域却比它少很多。

A*算法是一种启发式算法[4]，核心是目标函数，其计算公式为

$$f(n) = g(n) + h(n) \tag{1}$$

式中：f(n) 为当前节点 n 的总代价，代价值的大小对规划的路径有不同程度的影响；g(n)为从起始点到当前节点 n 移动所需耗费的代价；h(n)为从当前节点 n 到终点移动预计所需花费的代价，决定着算法的搜索方向；基于栅格法的 h(n) 常用曼哈顿距离、欧式距离和对角线距离来表示，本文采用曼哈顿距离下作为启发函数的估值方式，曼哈顿距离 $h_m(n)$ 的表示方式为：

$$h_m(n) = abs(x_n - x_m) + abs(y_n - y_m) \tag{2}$$

式中 x_n, y_n 分别为节点 n 的横坐标和纵坐标； x_m, y_m 分别为节点 m 的横坐标和纵坐标。

A*算法实现如图所示：

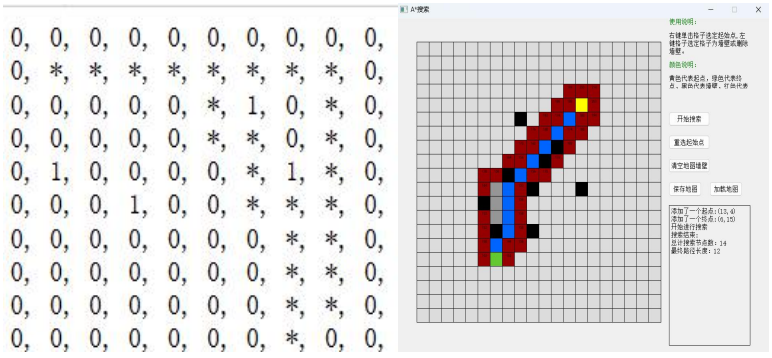

图 1.1　A*算法寻路图

D*算法简介

D*算法又称动态 A*算法，是一种动态逆向扇形搜索算法[5]，逆向的搜索机制保留了地图成本，避免了回溯的高计算成本，这也是 D*算法最大的优点[6]。D*算法计算量小，在动态环境下搜索效率更高。

D*算法的表达式如下：

$$f(n) = g(n) + h(n) \tag{3}$$

其中 n 为当前状态，$f(n)$ 为从初始状态经由状态 n 到目标状态的代价估计；$g(n)$

为初始状态到状态 n 的实际代价；$h(n)$ 为启发函数，与 A*算法类似，D*算法同样定义了两个列表合集：openlist 与 closelist；openlist 表由待考察的节点组成，closelist 表由已经考察过的节点组成，类似于 Dijkstra 算法中的 U 集合和 S 集合，当 closelist 表中出现目标节点时，程序停止[7]。

D*算法实现如图所示

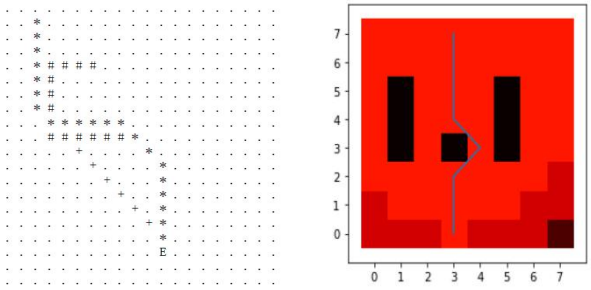

图 1.2　D*算法寻路图

LPA*算法简介

LPA*（Lifelong Planning A*）算法是一种用于解决路径规划问题的增量式搜索算法，它是对 A* 算法的扩展和优化[8]。LPA* 算法主要用于在有向图中寻找从起点到终点的最短路径。

LPA* 算法的核心思想是利用两个函数，g(s) 和 rhs(s)，来记录每个节点 s 的最短路径代价和次短路径代价。其中，g(s) 表示节点 s 的当前最短路径代价，rhs(s) 则表示节点 s 的次短路径代价。通过这两个函数的更新和比较，LPA* 算法能够动态地调整搜索方向，以便在遇到动态环境变化时快速更新路径。

rhs（s）的定义如下：

$$rhs = \begin{cases} 0, & \mathrm{s} = start \\ \min_{succ(s)}(g(t) + c(s,t)), & 其他 \end{cases} \tag{4}$$

其中，$start$ 表示第一个网格点，$succ(s)$ 表示节点 s 的后继节点集合，$g(t)$ 表示从起点到节点 t 的实际代价，$c(s,t)$ 表示从节点 s 到节点 t 的边的代价[9]。

在 LPA* 算法中，rhs(s) 的更新是动态的，当图中的边或节点发生变化时，需要重新计算节点 s 的 rhs(s) 值以确保其准确性。rhs(s) 的动态更新对于增量式路径规划非常重要，它可以帮助算法在变化的环境中快速调整路径，同时避免完全重新搜索整个图，提高了路径规划的效率。

D*Lite 算法简介

D* Lite [10]是由 Sven Koenig 和 Maxim Likhachev 于 2002 年提出的一种基于 LPA* 和 D* 的动态路径规划算法。因此 D* Lite 融入 D* 的逆向搜索思想，也结合了 LPA*算法的核心思想。D* Lite 是一种起始节点可变化的反向规划算法，因此更适合应用于未知环境或预规划路线突变障碍的情况。D*Lite 算法的次最短路径代价与 LPA*的定义大致相同，只是在 key 的取值上与 LPA*算法有不同之处，通过引入 km 因子修饰 key，使得启发估计值的 Sstart 节点是可变化的，在 LPA*算法中，节点 s 在优先列表 Key 中的定义如下[11]：

$$Key(s) = [Key_1(s), Key_2(s)] \tag{5}$$

其中:

$$\begin{cases} key_1(s) = \min(g(s), rhs(s)) + h(s) + k_m \\ key_2(s) = \min(g(s), rhs(s)) \end{cases} \tag{6}$$

式中 $h(s)$ 为起点到节点 s 的启发代价值，k_m 为最初起始点到当前起始点的 h 值

A*算法现有技术分析

1. 优化启发函数：启发函数是 A*算法的关键组成部分，直接影响算法的搜索性能[12]。常见的优化技术是使用机器学习方法，通过训练模型来预测启发函数的估计值，进而加速搜索过程，现有的技术分析主要关注如何设计更准确、更高效的启发函数。
2. 加速搜索过程：A*算法的搜索过程涉及到开放列表和闭合列表的维护，当图规模很大时，这些操作会耗费大量的时间和空间，现有的技术分析主要关注如何通过数据结构和算法的优化来加速搜索过程。
3. 实时动态路径规划：传统的 A*算法假设图的拓扑结构不会发生变化，但在实际的应用中，往往需要应对实时动态环境，现有的技术分析主要关注如何将 A*算法与感知、规划、控制等系统相结合，实现实时动态路径规划。
4. 并行化和分布式计算：A*算法的搜索过程可以通过并行化和分布式计算来加速，这些技术能够显著提高 A*算法的搜索效率，特别是在处理大规模图和复杂场景时[13]。现有的技术分析主要关注如何将 A*算法的搜索任务分解为多个子任务，并利用多核、多机等计算资源进行并行计算。

D*算法现有技术分析：

1. 双向搜索：D* 算法采用了双向搜索的方式，同时从起点和终点进行搜索。这种方式不但可以缩小搜索空间，提高搜索效率，还可以更快地响应环境变化，更新路径。
2.增量路径规划：D* 算法是一种增量路径规划算法，能够实时处理环境变化，适用于动态环境或需要频繁更新路径的场景[14]。
3. 解决无法确定代价值的问题：相比于其他路径规划算法，D* 算法考虑了代价值的不确定性，因此能够更加准确地表示路径的代价。
4. 依赖图的数据结构：D* 算法采用了依赖图的数据结构，能够有效地处理有大量空的地图，减少搜索空间，提高搜索效率。

LPA*算法现有技术分析

1. 增量式规划：LPA 算法通过维护节点的最短路径代价和次短路径代价两个函数，实现了增量式的路径规划。在环境变化时，算法可以快速更新部分节点的路径代价而不必重复计算整个图的路径，节省了计算资源。
2. 动态路径调整： LPA 算法能够动态调整搜索方向，根据当前节点的最短路径和次短路径代价情况选择扩展最优节点，以适应动态环境变化。这种动态路径调整能够使算法更加灵活地应对实时变化的情况。

3. 效率与优化：现有技术针对 LPA*算法进行了多方面的优化和改进，包括对更新策略的优化、对数据结构的改进以及对搜索过程的加速等，提高了算法的效率和性能。
4. 实际应用： LPA*算法在机器人路径规划、智能交通系统等领域有着广泛的应用。现有技术分析也可以结合具体的实际案例和场景，分析 LPA*算法在实际应用中的优势、局限性和改进空间。

D*Lite 算法现有技术分析
1. 增量路径规划：D*Lite 算法是一种增量路径规划算法，通过在每次搜索迭代中更新启发式值，可以快速地适应环境的变化
2. 启发式搜索：D*Lite 算法利用 A*算法的思想，在搜索过程中利用启发式函数估计到目标的成本，并据此来指导搜索方向，从而高效地找到路径。
3. 双向搜索：D*Lite 算法通常采用双向搜索的策略，同时从起点和终点开始搜索路径，直到两个搜索队列相遇。这样可以加快搜索速度，特别是在大规模地图中应用时效果更为明显。
4. 重规划策略：D*Lite 算法具有基于局部改变的重规划策略，当环境发生变化时，只需对附近受影响的路径段进行重新规划，而不需要重新规划整条路径。这种策略减少了计算量，提高了算法的实时性。

实验结果与分析
本次实验，我们将研究新的并行算法，以提高处理大规模动态图的效率。我们将使用多线程、多进程、进行并行计算。本次实验中，采用 python 库中的 joblib 包，joblib 主要用于将 Python 对象存储在磁盘上，从而可以提高计算效率，以适应大规模复杂动态图的特性。同时使用模拟数据和真实数据进行串行并行以及大规模图分割算法效率对比。

实验一
当选取相同的起始点和终点位置，选取 8 个障碍物点后，对单个节点进行寻路，A*算法，D*算法，LPA*算法和 D*Lite 算法时间对比如下：

地图大小	10*10	20*20	30*30	40*40	50*50	100*100
A*算法用时	0.0145	0.0429	0.2019	0.3353	0.6521	5.3618
D*算法用时	0.0025	0.0099	0.0269	0.0932	0.1294	0.6309
LPA*算法用时	0.0094	0.0184	0.0281	0.0518	0.2102	0.2400
D*Lite 算法用时	0.0062	0.0497	0.1137	0.1332	0.2192	0.9710

实验二
在随机生成五个坐标点之后，在不同地图大小下，对多个节点进行寻路，串行和并行的时间对比如下（其中每个地图下的时间是测试 10 次的平均值）

地图大小	10*10	20*20	30*30	40*40	50*50	60*60	70*70	80*80	100*100

A* 算法串行	2.55ms	37.7ms	0.17s	0.54s	1.32s	2.74s	4.96s	8.4s	20.3s
A* 算法并行	3.58ms	41.2ms	0.19s	0.54s	1.32s	2.88s	5.43s	9.0s	22.5s
A* 算法并行 joblib	5.42ms	60.9ms	0.11s	0.32s	0.76s	1.53s	2.86s	4.63s	11.4s

注：为最大范围进行寻路，起点坐标和终点坐标的设置分别为左上角和右下角。

实验三

对庙算数据地图进行 A*寻路和并行的测试。庙算地图数据选择分队级的庙算杯人机对抗，该数据是 2021 年 7 月庙算杯决赛人机对抗数据,共有 88 局比赛实时制信息。该数据中地图障碍物信息有两组，分别为 15 个障碍物点和 9 个障碍物点。夺控点信息有两组，其中地图大小为 92*77，算子坐标有 6 组。对庙算数据地图进行 A*寻路和并行的测试结果如下：(其中每个地图的时间是测试 10 次取的平均值）

地图信息	地图 1	地图 2
A*算法串行	17.2ms	23.1ms
A*算法并行	14.9ms	21.0ms
A*算法并行（joblib)	15.5ms	16.5ms

实验四：

在庙算地图上对大规模图进行分割后寻路时间与未分割时间对比（注：为最大范围寻路，起始点设置为（0,0）,目标点为（91,76））

地图信息	地图 1	地图 2
A*算法	4.6134	4.6147
A*算法（分割后）	0.9562	0.4762

由上述数据分析可得

1.在 100*100 大小的地图上、8 个障碍物情况下，单个算子寻路算法的搜索时间达到秒级。

2.支持并行加速，在庙算数据 92*77 大小的地图 2 上、9 个障碍物，6 个算子的情况下，对比串行和并行、加入 joblib 包、分割后的并行加速比分别为 1.1，1.4，和 9.69，寻路效率分别增加 9.09%，28.6%，89.6%。

参考文献

[1]CANDRAA,BUDIMANMA,POHANRI.ApplicationofA-staralgorithmonpathfinding game[J].JournalofPhys ics:ConferenceSeries,2021,1898(1):012047.
[2]车建涛,高方玉,解玉文,等.基于 Dijkstra 算法的水下机器人路径规划[J].机械设计与研究,2020,36(1):44-48.
[3]赵春宇,姜皓,徐茂竹,等.改进 A* 算法在无人船路径规划中的应用[J]. 浙江工业大学学报,2022,50(6):615-620.
[4]秦玉鑫，王红旗，杜翠杰. 基于双层 A*算法的移动机器人路径规划[J].制造业自动化,2014，36(12):21- 25.
[5]李凯业.基于改进分层 D*搜索算法的室内路径规划问题研究［D］.合肥：合肥工业大学.2015
[6]胡粒琪,曾维,陈才华等.基于改进 D*Lite-APF 算法的巡检机器人路径规划[J].现代电子技术，2024，47（5）：155-159.
[7]李季，孙秀霞. 基于改进 A-Star 算法的无人机航迹规划算法研究［J］. 兵工学报,2008,29（7）: 788-792.
[8]郭晓宇，鲁旭涛. 基于 LPA*优化算法的察打一体无人机远距离飞行航线规划［J］.战术导弹技术，2023（4）:148-155
[9]李生，李明宇，方舟等.基于改进 LPA*算法的虚拟人路径规划[J].装备制造技术，2022（08）:82-84
[10]SVEN K，MAXIM L，DAVID F. Lifelong Planning A［J］.Artificial Intelligence: An International Journal，2004，155(1/2) : 93－146.
[11]吴宏鑫，解永春，李智斌，等. 基于对象特征模型描述的智能 控制[J]. 自动化学报，1999，25(1) : 9-17.
[12]ZHOU Tao，ZHAO Jin，HU Qiuxia，et al. Global path planning and tracking for mobile robot in cluttered environment［J］.Computer Engineering，2018，44(12) : 208-214
[13]徐唐剑.A*寻路算法的并行化设计及改进[J].现代计算机(专业版)2018(21):44-49
[14]秦 旭,黄晓华,马东明等.基于改进 D*算法的巡检机器人路径规划[J].组合机床与自动化加工技术，2022（06）：10-13