# **基于区域生成的分层路径规划方法**

张建龙1,2,王紫瑶2,1,陈晨3,2，张选潮4

1. 西安电子科技大学 电子工程学院, 陕西西安 710071
2. 西安电子科技大学 杭州研究院，浙江杭州 310000
3. 西安电子科技大学 通信工程学院，陕西西安
4. 西安微纳软件有限公司 陕西西安

**摘 要：**针对栅格地图路径规划计算复杂度随地图分辨率线性增长问题，本文提出一种基于区域生成的分层路径规划方法，将全局路径搜索转化为并行区域路径搜索问题，有效减少搜索空间并降低了规划计算复杂度。本文方法主要由离线区域路径规划（ODPP）、在线动态路径规划（ORPP）和全局最优路径规划（GOPP）三部分构成。首先，离线区域路径规划根据地图障碍物分布搜索全局关键点，基于关键点完成区域划分并计算区域邻接矩阵，然后利用JPS算法获取基于区域的全局路径规划先验。其次，在线动态路径规划根据起点、终点所在区域完成并行局部路径规划，并联合全局路径规划先验构建全局动态无向图。最后，全局最优路径规划根据全局动态无向图利用Dijkstra算法完成最优路径搜索。仿真结果表明，在高分辨率复杂地图和相同路径质量条件下，相比于对比方法本文方法极大节约了时间成本，具备良好的工程实用性。

**关 键 词：**分层路径规划；区域划分；离线区域路径规划；全局动态无向图；区域路径搜索；

## 引言

路径规划是移动机器人、自动驾驶决策与控制的核心技术之一[1,2]，已经被广泛地应用于各个领域，涉及日常生活，物流运输，工业生产以及救援任务的众多方面，发挥着至关重要的作用。随着人工智能技术、通讯技术、硬件设施的不断进步完善，使得移动机器人、自动驾驶进入人们的生活中成为可能[3]。但是，现实生活中的高复杂度和不确定性的环境对导航系统的安全性、准确性、规划效率等提出了更高的要求。路径规划作为移动机器人自主导航的基础，对路径规划算法的要求也越来越高。路径规划算法研究的核心内容之一是提高规划效率、路径质量和鲁棒性并保证规划稳定性，在给定部分环境信息的基础上，需要在满足某些条件（如距离、平滑度、安全性）的两个指定位置之间自主规划最优或次优无碰撞路径[4]。因此，路径规划算法的研究对提高移动机器人以及自动驾驶的智能化水平、加快工程化应用具有重要意义。

在路径规划领域中主要分为全局和局部规划两个方面。全局路径规划主要分为基于图搜索的规划算法、基于仿生学的智能规划算法和基于随机采样的规划算法。基于图搜索的算法是机器人路径规划算法中最早也是最成熟的算法。这类算法的特点是通过一定的准则将移动机器人的运动空间分割为离散空间，再进行路径搜索。常见的离散化方法有栅格法和可视图法[5-6]，其中栅格法应用最为广泛。栅格法通过某种方法将机器人所处的环境划分为具有局部环境信息的网格单元。基于栅格的搜索算法，如 Dijkstra 算法[7]、A\*算法[8]、D\*-lite 算法[9]、theta\*算法[10]等，在建立对应的栅格地图基础上，利用图论的方法从邻近结点开始不断搜索寻找最短路径，具有算法完备性。A\*算法以其卓越的搜索路径的稳定性及出色的实时性而著称，相较于粒子群算法和遗传算法等其他全局路径规划算法，它的实现效率更高，成为机器人应用最为广泛的全局路径规划算法之一[11]。由于该算法具有良好的可扩展性与适应性，因此被广泛应用于各种环境中。

基于仿生学的智能规划算法[12]，如遗传算法[13-14]、粒子群算法[15]、蚁群算法[16]等。遗传算法是一种模拟生物种群迭代繁殖的搜索算法，通过路径编码、选择、交叉、变异等机制，不断迭代优化，最终获得近似最优解，是仿生优化领域中应用最为广泛的算法之一[17-18]。遗传算法适用范围广，但收敛速度不受控，实时性差。Yu X[19]等人将蚁群算法与 A\*算法相结合，按照粒度大小对环境信息进行分层处理，使用蚁群算法在大粒度环境下提供初始规划结果，然后利用A\*算法做进一步规划，有效改善了蚁群算法规划效率低的问题。

基于随机采样的规划算法[20]，常见的如快速搜索随机树(RRT)[21]算法和概率路线图(PRM)[22]算法。此类算法不需要对环境进行精确建模，在复杂环境及高维环境下具有明显的优势。RRT 算法可快速得到一条无碰撞路径，但存在未考虑动力学约束、树的生长无方向性、路径非最优等问题。针对这些问题，国内外进行了大量的研究[23]。针对动力学约束问题，Ghosh[24]等人提出 KB-RRT 算法，通过考虑机器人的运动学模型实现运动学约束的路径规划。针对树生长的无方向性问题，ChengRen Yuan[25]等人在 RRT的基础上，引入启发式思想提出 BG-RRT 算法，引导树向着目标点方向扩展，进而提高收敛速度。

在面对大规模搜索空间，特别是存在密集障碍物和狭窄区域的情境时，使用传统算法如A\*、JPS算法进行路径规划会出现性能瓶颈的挑战。这主要由于其计算量随栅格地图区域、分辨率以及查找空间的大小而增加，导致计算时间增加、响应速度下降、实时性能下降以及实时资源占用的增加等等[26]。针对该问题，本文提出了一种基于区域生成的分层路径规划方法，通过获取区域分布先验解耦复杂度最高的区域路径规划并实现离线处理，解决了高分辨率栅格地图复杂路径规划的实时性问题。

本文主要贡献包括：1）搜索关键点集合完成区域划分，构建包含区域信息的全局动态无向图，将全局路径规划转化为基于区域的分层局部优化问题，降低了规划复杂度。2）根据区域空间分布获取区域路径规划先验，离线处理最耗时的主路径规划，有效提高了算法实时性。

论文的结构安排如下：第二节对所提出算法的原理和实现步骤进行详细介绍；第三节为实验结果和分析；第四节对文章进行总结。

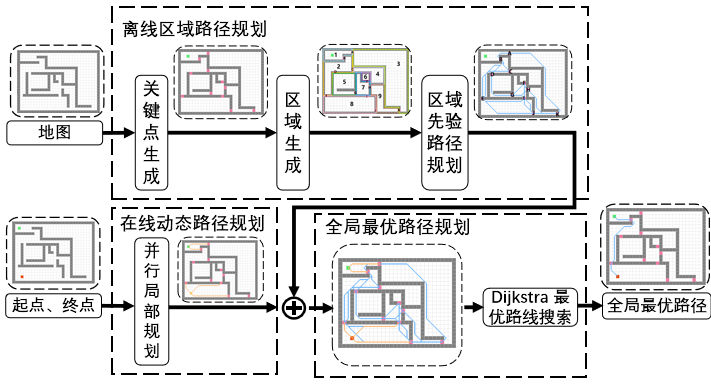


图1 本文方法的路径规划流程图

## 方法原理

本文原理结构如图1所示，主要由离线区域路径规划、在线动态路径规划以及全局最优路径规划三部分组成。离线区域路径规划和在线动态路径规划一同构成全局动态无向图，用于后续全局最优路径规划。首先，离线区域路径规划通过JPS生成的全局区域关键点以及区域障碍物分布划分空间为多个区域，并得到基于关键点的区域邻接矩阵。通过这些区域邻接矩阵，离线路径规划使用JPS算法规划区域关键点间的路径，从而生成区域先验路径规划。其次，在线动态路径规划针对起点、终点所在区域进行并行局部规划。最后，全局最优路径规划使用Dijkstra算法搜索全局动态无向图确定全局最优路径。

我们首先阐述本文算法建模过程，其次解释各模块设计原理。

### 2.1 问题定义

路径规划是指确定从起点到目标点（终点）的无碰撞最佳路径，考虑到规划时路径长度、避障等因素的影响，可将其建模成一个基于无碰撞约束的最短路径优化问题：

 (1)

其中，为包含N个节点的路径，为距离函数用于求集合内相邻两点间的距离，*v*()函数用于判断集合内相邻两点间是否无障碍物：

(2)

 (3)

而栅格法通过将求解区域离散化为一个有限的网格中，实际求解最小成本节点的评价函数为可以写为公式（4）,其中，*g*()和*h*()的具体含义及公式见2.4节：

 (4)

对于复杂栅格地图而言，规划复杂度随地图分辨率和障碍物分布非线性增长，经典规划算法无法适应大地图实时规划的需求。地图上的一些狭窄通道不仅连接了不同区域，还天然地将地图分割成了多个部分，并且通过观察可以发现长路径规划必定穿越这些区域连接的节点，本文称它们为关键点（具体定义见2.2.1），如图2所示，起点和终点的最终路径必定经过节点B、E或节点A、F、G（其他路径）。由此，关键点集合{A，B，C...}与障碍物可以将地图划分为区域1、2、3等众多区域，所有可能的路径端点必定分布在这些区域中，规划的路径也必定穿越相互邻接的区域。事实上所有的区域连接关系可以提前获取，成为离线规划先验，从而简化规划复杂度，因此本文提出基于区域分层规划的思想，将全局路径规划目标函数J拆分为三部分：

 (5)

、、分别属于离线区域路径规划（ODPP）、在线动态路径规划（ORPP）和全局最优路径规划（GOPP）。离线区域路径规划完成栅格空间的区域划分和区域连接关系构建；在线动态路径规划完成起、终点的区域定位和区域路径规划；全局路径规划根据生成的全局无向图搜索完整最优路径。其中中的区域划分是通过关键点和障碍物栅格一同围成封闭空间来实现，如图2，关键点A~G和障碍物节点一起分别围成了区域1、2、3，区域1、3之间通过B-E路径连接。因此在线实时规划仅需规划区域1、2空间内的路径-B、E-。由图2观察可知，区域分布由关键点位置决定，与起始点位置无关，是栅格地图的本身属性和先验知识，可以离线生成。

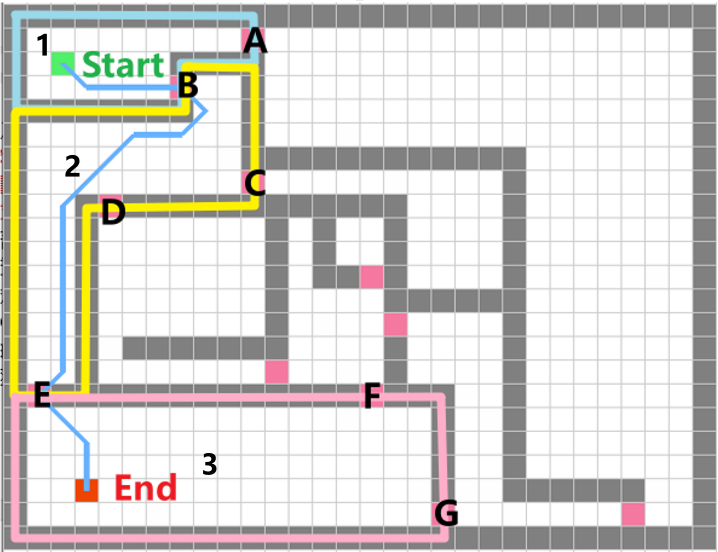


图2 区域划分和路径规划结果

因此利用区域分层规划解耦全局动态规划，并分离出计算复杂度最高部分，目标函数可等价如公式(6)，其具体定义和展开式见2.4内容。

 （6）

### 2.2 离线区域路径规划

一幅栅格地图一旦确定，就可以事先合理划分成若干子区域，起始点分布在这些区域中，点对点的路径规划问题就可以转换为点-区域-点的分层规划问题，由于区域划分可以离线处理，因此我们可以获取区域规划先验，极大节省实际规划时间。

离线区域路径规划首先使用JPS遍历全图生成关键点并划分区域，并基于关键点分布确定区域间邻接关系，最后进行区域内离线局部规划，得到基于关键点的区域局部路径，从而将全局路径规划转化为并行区域路径规划问题。我们将给出区域定义，并阐明区域划分原理以及区域邻接关系的生成原理。

##### 2.2.1 关键点生成

如图3，在地图上特定的狭窄空间（节点A~K）起到了连接地图不同部分的重要作用，同时这些节点也将地图自然划分成了若干个区域。识别这些关键节点和区域对于地图的分层处理至关重要。我们将使用JPS算法遍历全图并结合地图障碍物分布等特征进行初始关键点生成，然后进行关键点滤波，从而得到最终的关键点集合。

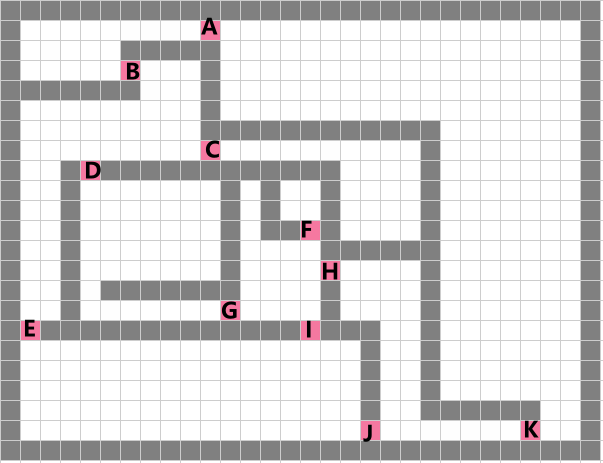


图3 区域关键点

关键点集合将满足以下条件：

1. (7)

2. (8)

3. (9)

4. (10)

其中，公式(7)得到存在冗余点的初始关键点，通过公式（8）~（10）对其进行筛选得到最终关键点集合。公式7中的为JPS算法遍历全图得到的“跳点”集合，、为节点的x坐标和y坐标。*H*、*V*集合如图4所示的1、2范围内的节点。、分别表示节点 n 的水平和垂直邻域方向上的点集。我们定义*iswall*(*x*) 函数来确定当前位置*x*是否为障碍物，若为1表明该集合内的点均为障碍物，若为0则不全为障碍物。因此得到、：

 (11)

 (12)

|  |  |
| --- | --- |
|  |  |
| (a)水平方向 | (b)垂直方向 |

图4 关键点间水平、垂直方向集合

算法1给出了关键生成的核心思想。通过利用JPS算法遍历全图跳点（successors），从起点开始搜索得到符合定义公式（7）的初始关键点，然后遍历初始关键点，而后根据地图分布特征进行关键点的滤波，滤波近距离关键点实现公式（8），通过沿着狭窄区域向四向搜索，删除可能存在的冗余关键点实现公式(9)、(10)。最终效果，如图5展示了从跳点到关键点生成的整个过程，其中图5（a）蓝色节点为JPS搜索的跳点，图5（c）（d）中黄色节点分别为公式(9)、(10)滤除的节点，最后图5（d）保留的粉红色节点为关键点集合。

|  |  |
| --- | --- |
|  |  |
| （a）跳点 | （b）初始关键点 |
|  |  |
| （c）滤波1 | （d）滤波2 |

图5 关键点生成过程

|  |
| --- |
| Algorithm 1**：Keys Generation** |
| **Input：***map*: a grid map contains inforation about wallkey andobstacles, *map*[]=1 represents an obstacle at that position, *start*，*goal*，Jump(): forward jump function, d():distance calculation function, isVaild():effective node determination function. |
| **Output：***keys*: key points |
| **1**：Initialize ,, |
| **2**：**for** all *n* ∈ *lst* **do** |
| **3**： *successors* = Jump(*n*) |
| **4：** add *successors* to *lst* |
| **5**：**for** all (*x*, *y*) ∈ *succsessors* **then** |
| **6**： **if** (and) or ( and)**then** |
| **7**： add(*x*, *y*)to *keys* |
| **8**：**for**∈*keys* **do** |
| **9**： **if**  **then** |
| **10**： remove  from *keys* |
| **11:** **for** all *dir* ∈ *dirs* **do** |
| **12:** |
| **13: while** isValid() and  and  **do** |
| **14: if** ∈*keys* **then** |
| **15:** remove from *keys* |
| **16**：**return** *keys* |

##### 2.2.2 区域生成

全局关键点集合生成后，我们根据关键点分布以及全局障碍物的特征，将全局地图共划分为M个区域，用于后续区域局部规划。

我们可将由障碍物节点和关键点一同围成的封闭空间定义为区域，如图6彩色框表示了障碍物、关键点围成各个封闭空间，具体而言，左上角蓝色狂内的封闭空间由障碍物节点和关键点集一同围成。我们定义由关键点和障碍物节点围成的封闭空间为区域，它由封闭空间区域内无障碍物点集和区域关键点集合构成，障碍物集合称为。区域（也称区域点集合）的具体定义如下。

**定义**：区域，满足：

 (13)

其中，区域和全局区域*R*满足关系：

 (14)

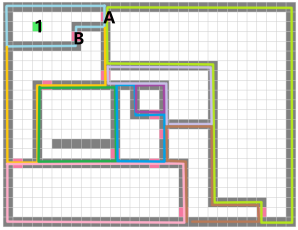
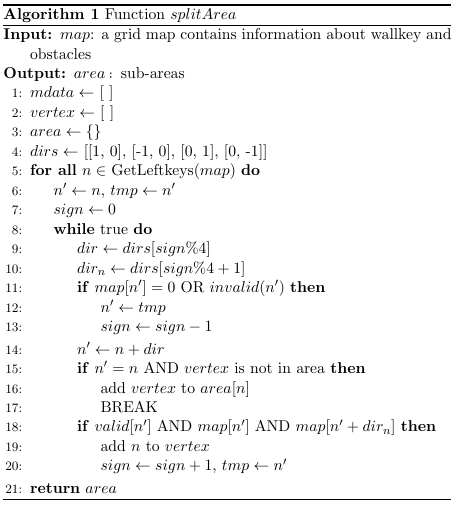


图6 区域

算法2给出了利用关键点集合划分区域的过程。整个过程可以分为三部分，确定遍历区域左上顶点，区域各顶点探索及区域更新。首先，确定遍历区域的起点*Leftkeys*集合，这些点集是基于障碍物分布特征确定的，如图7，绿色节点为区域的预选左上顶点。其次，遍历区域各顶点，从左上顶点开始，按照逆时针方向遍历地图上的障碍物和关键点。通过一个标记变量（*sign*）来控制遍历的方向，在遍历过程中，如果遇到下一个方向上是障碍物点，则转换遍历方向并继续遍历。如果遇到一个非障碍物点，说明该方向的障碍物和关键点无法围成封闭区域，需要返回原出发点并沿原方向重新探索。最后，得到区域顶点集合后，我们会将未存在于区域集合内的区域更新到该集合中。



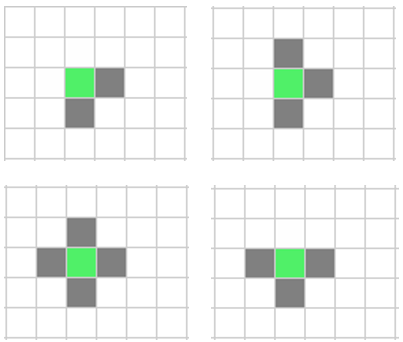


图7 区域的预选左上顶点的几种类型

##### 2.2.3 区域先验路径规划

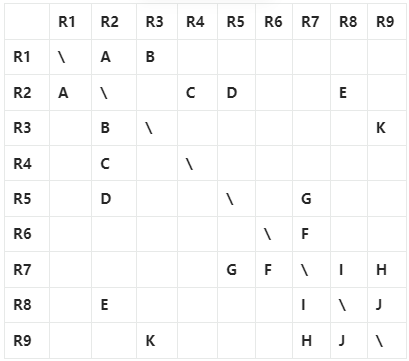
基于关键点将整个地图划分为多个区域后，我们可以确定各个区域间的连接关系，从而得到区域邻接矩阵，并以区域关键点作为引导，在每个区域内部进行局部规划，从而实现基于关键点的区域先验路径规划。

**A. 区域邻接关系**

区域邻接关系确保了整个地图的连接性，可以在规划时指导机器人快速到达不同区域，避免不必要的绕行，同时根据邻接关系，可以确定哪些区域需要更多的资源或关注，从而进行合理的资源分配和管理。全局地图由算法1划分为多个区域后，可通过区域间的公共点情况来获取区域间的位置关系，得到基于关键点的区域邻接矩阵，邻接矩阵计算方式如公式15所示：

 (15)

其中表示区域，公有的关键点集合，并唯一确定了区域间连通的关系，即区域和仅能通过中的关键点通过，因此可得到基于关键点的区域邻接矩阵。如图8，上图中矩阵中存放关键点信息，空处表示不相连，下图可视化相邻关系，和间关键点为B，并且它们之间通过B关键点相连。至此，区域间的位置关系可通过基于关键点集合的邻接矩阵来表示。



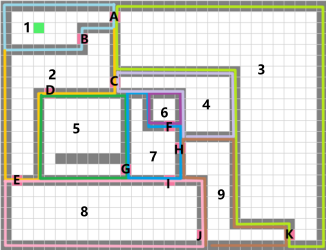


图8 区域邻接关系图

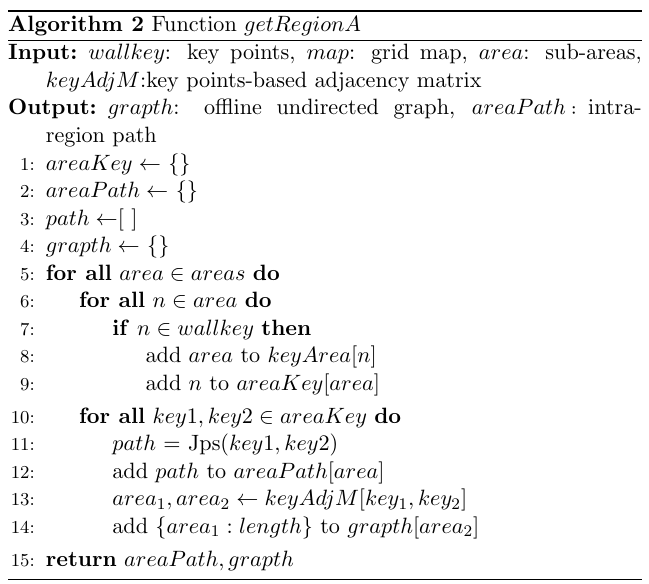
**B. 离线局部路径规划**

离线局部路径规划的目的在于预先规划了区域内基于关键点的局部路径，避免了在实时路径规划中进行大量的计算和搜索。根据区域关键点邻接矩阵得到的区域间连通关系后，我们可对区域内部关键点间进行两两路径规划，从而得到非相邻区域间的路径规划，如图9（a）区域、间由关键点B连通，区域、间由关键点E连通。预先规划的区域内的BE路径联通区域和区域，并且该路径长度最优，通过区域最少。因此，我们使用JPS算法生成区域局部路径，用于后续全局区域路径的生成，有效提高规划效率并减少规划时间。

算法3给出区域局部路径、离线无向图的生成过程。通过区域内关键点，使用JPS算法规划出区域内两两关键点间的路径，结合区域邻接矩阵创建区域间的离线无向图(区域局部路径)，如图9（b）所示。

|  |  |
| --- | --- |
|  |  |
| (a) 区域路径规划 | (b) 区域局部路径 |

图9 局部路径规划结果



### 2.3 在线动态路径规划

离线区域路径规划可获得区域连接路径先验，因此在线动态路径规划实时应用中仅需定位起点、终点所在区域，并实施起终点与该区域的关键点的局部路径规划即可。

##### 2.2.1 起点、终点区域确定

射线法[27]是判断点与多边形内外关系的最基本有效的方法，因此本节采用该算法来快速判断起点、终点所在区域，并用于后续全局最优路径规划。如图10，起点位于区域，终点位于区域。

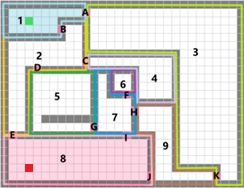


图10 起点、终点区域的确定

##### 2.2.2 起点、终点连接区域

利用JPS算法规划起点、终点和区域内关键点集合间的路径，从而实现起点、终点所在区域的内部路径最优搜索，由于两种规划无内在相关性，因此实际规划中两个区域的路径规划可以并行实施。如图11，橙色路径为起点、终点和其各自所在区域的相邻区域间的路径规划，并同离线规划的区域局部路径（蓝色路径）一同构成全局动态无向图[28]。

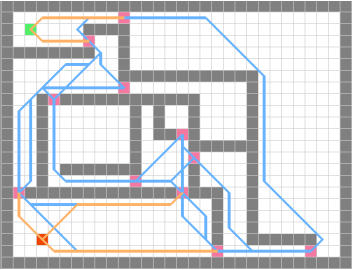


图11 全局动态无向图

### 2.3 全局最优路经规划

全局最优规划基于在线动态路径规划和离线先验路径拼接生成的全局动态无向图。Dijkstra算法是图论中典型的单源最短路径算法，其具有全局最优性。通过采用Dijkstra算法搜索起终点之间的最优路径，从而实现全局最优路径规划。如图12，我们离线规划得到区域到区域之间的路径BE，在线规划得到路径-B、E-。最后，使用Dijkstra搜索确定了全局最优路径：-B-E-。

当起点、终点位置改变时，进行在线规划时只需重新进行起点、终点区域内的局部规划，并使用Dijkstra算法重新搜索全局最优路径，不涉及时间复杂度过高的空间搜索，避免了重新计算导致的时间和资源成本浪费。

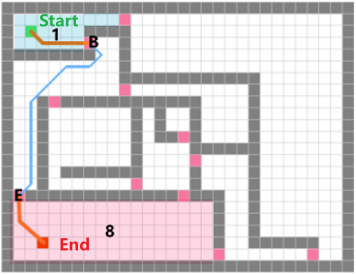


图12 全局最优路径

### 2.4 代价函数及复杂度分析

##### 2.4.1 代价函数计算

由2.1可知我们优化的代价函数*J*可以拆分为、、，并可给出具体的计算公式[29]如下：

 (16)

 (17)

 (18)

其中，、分别为区域内在线、离线扩展节点，如图13终点、起点所在区域和内无障碍物节点为在线扩展节点，其余区域内为离线扩展节点，关键点集合节点（图中节点A~K）为。

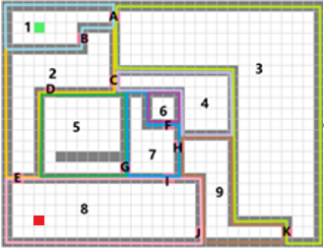


图13 搜索节点

、均为从起点到当前节点n的实际代价,而由于离线、在线和全局规划搜索节点类型不同，其实际计算公式如下：

 (15)

其中L、O、G为离线、在线节点及全局规划搜索节点数量。表示关键点和的边权重。为栅格地图中两节点间距离，、为所求节点和的x、y坐标的绝对差。其具体计算公式如公式(18)。为启发函数，计算当前节点到终点的预估距离如公式(19)。

 (18)

 (19)

##### 2.4.2 时间复杂度

设JPS算法的时间复杂度为，Dijkstra算法的时间复杂度为,由于*n*为搜索尺度(此处则为搜索空间大小)，因此本文方法全局路径规划的时间复杂度为：

 (20)

其中为全局区域(也为全局区域无障碍节点集合)，，和分别为起点、终点区域，实际使用时的在线全局搜索时间复杂度为：

 (21)

而随着地图分辨率的提升，区域数量增多，并且全局区域节点数量会远大于子区域内节点数量，即，同时子区域节点数量也大于关键点数量，例如图13中的节点数量。因而即便Dijkstra算法时间复杂度高于JPS算法，但随着地图分辨率的提升，在线搜索的时间复杂度将远低于JPS和Dijkstra算法：

 (22)

## **实验结果与分析**

##### 3.1 实验设置和评价指标

为了充分验证本文方法的有效性，本文构造了两组栅格地图，第一组地图是如图14的地图的尺寸从23×30到30×40不等的低分辨率栅格地图，；第二组地图如图15所示的通过纵向拼接构造的高分辨率栅格地图，地图分辨率从100×100到1594×200不等。对比算法选择实时性较好的基于图搜索的A\*算法、JPS算法以及基于采样原理的GSLST算法。对比方法采取分组对比以及随机抽样方式，同一地图随机抽取100组不同的起点、终点，每组进行10次规划计算性能指标平均值。对比性能指标包括平均规划时间、平均路径长度、规划成功率、扩展节点数量等。仿真实验环境包括CPU Intel(R) Core i7-7700 3.60GHZ、RAM 16 GB、Windows 10、python3.8。

|  |  |
| --- | --- |
|  |  |
| (a) 23×30 | （b）28×30 |
|  |  |
| （c）30×30 | （d）30×40 |

图14 低分辨率栅格地图

|  |  |  |
| --- | --- | --- |
|  |  |  |
| (a)100×100 | (b)140×151 | (c)200×200 |
|  |  |  |
| (d)199×151 | (e)381×151 | (f)399×200 |

图15 高分辨率栅格地图

##### 3.2 仿真结果与分析

|  |
| --- |
|  |
| （a）28×30地图中三种算法规划时间比较 |
|  |
| （b）200×200地图中两种算法规划时间比较 |
|  |
| （c）28×30地图中三种算法扩展节点数量比较 |
|  |
| (d)200×200地图中两种算法扩展节点数量比较 |

图16 两种地图中本文方法和对比算法规划结果

|  |
| --- |
|  |
| (a) 28×30地图下A\*路径规划 |
|  |
| (b) 28×30地图下Jps路径规划 |
|  |
| （c）28×30地图下GSLST路径规划 |
|  |
| (d) 28×30地图下本文方法路径规划 |

|  |
| --- |
|  |
| (e) 200×200地图下A\*路径规划 |
|  |
| (f) 200×200地图下Jps路径规划 |
|  |
| (g) 200×200地图下GSLST路径规划 |
|  |
| (h) 200×200地图下本文方法路径规划 |

四种方法在两组地图中的规划结果如表1、表2所示，其中表1展示平均规划时间和平均扩展节点结果，而表2展示规划路径长度和规划成功率结果。

由表1可知，本文方法的搜索速度、扩展节点数量均明显优于所有对比算法。随着栅格地图分辨率和复杂度的提升，对比算法的规划时间和扩展节点都成倍增加，尤其是第二组高分辨率地图对比算法的规划时间和扩展节点数量呈几十倍增加，本文算法相关指标却增加缓慢，最大增幅在分别在5倍和2倍左右。第二组地图(100×100~793×100)中本文方法的平均规划时间均低于17ms和平均扩展节点数量均低于26个。在最高分辨率1593×200栅格地图中，本文算法的规划时间只有A\*算法的1/8000，JPS算法的1/57，平均扩展节点数量只有A\*算法的1/3000，JPS算法的1/71，而GSLST算法规划失效因此没有参与对比。分析表一结果，主要是由于基于图搜索的A\*、JPS算法将状态空间离散化为图，并在离散空间中搜索最优路径，能在有限的时间内找到最优解，但在面对高分辨复杂栅格地图时，A\*、JPS算法搜索空间会随着地图分辨率的提高而爆炸性增长，而本文算法利用区域生成巧妙解耦了计算复杂度最高的全局规划部分并通过离线方式事先计算，显著降低了实际规划复杂度并提高了算法实时性，随着分辨率和复杂度的提高，本文算法的优势更为明显。

由表2内容可知，本文方法的规划路径长度、规划成功率与A\*算法和JPS算法完全一致，并优于GSLST算法，这说明本文方法在规划效率显著提升时并未牺牲规划路径有效性。而GSLST算法在其指定的规划时间内存在规划失败的现象，且在面对高分辨率地图情况下的失败概率明显提高，这是由于采样的GSLST算法难以避免局部最小值问题该。

除此之外，为了详细对比相关算法在不同复杂地图中的规划效率和复杂度，我们分别统计了28×30和200×20两种尺度地图中各算法的规划时间和扩展节点数量，具体结果如图？？所示，其中横坐标为实验组序数，纵坐标分别为规划时间和扩展节点数量。其次展示了四种算法在两种地图中规划实例对比。由于高分辨率地图中A\*算法规划结果与JPS、本文算法指标相差过大，因此28×30地图下展示本文方法和A\*、JPS算法规划的对比效果，200×200地图下只展示本文方法和JPS算法规划的对比效果。

可以出本文方法规划时间波动较小，在28×30地图中规划时间始终小于1ms以下，200×200地图中大多数情况下的规划时间在10ms以下，明显小于对比方法，规划效率优势显著。随着规划路径长度的增加，本文方法搜索节点数量明显小于对比算法，并且在28×30、200×200地图中的数量分别低于10、20个，且扩展节点数量随规划路径长度整体浮动最小。本文方法的搜索空间复杂度较低，并在高分辨率地图中效果更为显著。

如图17展示了在28×30、200×200地图中相同规划任务下四种算法规划效果，蓝色节点为已扩展节点，绿色节点为待扩展节点，蓝色线段为规划路线。

总体来说，实验结果表明本文提出的方法在复杂地图中的规划性能方面表现出色，具有显著的优势。在规划质量与对比算法相当条件下，1593×200最高分辨率的复杂栅格地图的路径规划时间只有JPS算法的1/57，速度和效率优势较为突出。

表1 不同地图中四种算法规划结果I

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 地图  规格 | | 平均规划时间/ms | | | | 平均扩展节点数量 | | | |
| A\* | JPS | GSLST | 本文  方法 | A\* | JPS | GSLST | 本文  方法 |
| 第一组 | 23×30 | 18.25 | 3.81 | 295.79 | **1.46** | 148.48 | 16.13 | 595.08 | **7.75** |
| 28×30 | 14.56 | 3.30 | 2816.18 | **0.98** | 127.32 | 25.55 | 2746.27 | **5.46** |
| 30×30 | 18.38 | 6.16 | 124.84 | **1.56** | 145.63 | 12.46 | 393.55 | **5.35** |
| 30×40 | 30.05 | 5.64 | 113.18 | **2.57** | 140.27 | 21.59 | 673.50 | **9.94** |
| 第二组 | 100×100 | 174.68 | 14.19 | 139.14 | **8.69** | 1288.17 | 37.34 | 346.13 | **21.93** |
| 199×100 | 954.27 | 34.44 | 693.16 | **10.00** | 3277.68 | 88.03 | 1331.72 | **19.78** |
| 397×100 | 5638.31 | 121.64 | 2038.03 | **14.10** | 7938.41 | 195.64 | 2544.04 | **22.05** |
| 793×100 | 32347.90 | 327.12 | 3576.71 | **16.19** | 21080.84 | 502.9 | 2787.93 | **25.54** |
| 140×151 | 656.10 | 29.38 | 3929.26 | **3.80** | 2890.68 | 125.83 | 5021.55 | **13.47** |
| 191×151 | 1270.91 | 40.48 | - | **3.96** | 3966.09 | 166.40 | - | **13.85** |
| 381×151 | 9903.25 | 169.16 | - | **6.18** | 11458.18 | 464.24 | - | **13.70** |
| 761×151 | 77896.67 | 491.06 | - | **7.97** | 28933.25 | 1141.33 | - | **13.80** |
| 200×200 | 1955.43 | 53.39 | 2396.85 | **7.15** | 4615.09 | 109.81 | 3517.17 | **13.74** |
| 399×200 | 6609.14 | 138.67 | 4010.37 | **10.76** | 8866.51 | 207.68 | 2969.75 | **13.82** |
| 797×200 | 83397.35 | 451.19 | 6023.26 | **11.91** | 25410.67 | 581.62 | 1178.63 | **15.02** |
| 1593×200 | 133158.25 | 932.12 | - | **16.27** | 65146.15 | 1532.26 | - | **21.50** |

表2 不同地图中四种算法规划结果II

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 地图  规格 | | 平均路径长度 | | | | 规划成功率 | | | |
| A\* | JPS | GSLST | 本文  方法 | A\* | JPS | GSLST | 本文  方法 |
| 1 | 23×30 | 24.39 | 24.39 | 35.19 | 24.39 | 100% | 100% | 100.00% | 100% |
| 28×30 | 25.13 | 25.13 | 30.90 | 25.13 | 100% | 100% | 94.00% | 100% |
| 30×30 | 19.40 | 19.40 | 29.33 | 19.40 | 100% | 100% | 100.00% | 100% |
| 30×40 | 21.98 | 21.98 | 35.80 | 21.98 | 100% | 100% | 100.00% | 100% |
| 2 | 100×100 | 65.42 | 65.42 | 145.66 | 65.42 | 100% | 100% | 100.00% | 100% |
| 199×100 | 109.49 | 109.49 | 233.45 | 109.49 | 100% | 100% | 100.00% | 100% |
| 397×100 | 177.55 | 177.55 | 445.19 | 177.55 | 100% | 100% | 98.33% | 100% |
| 793×100 | 348.24 | 348.24 | 963.88 | 348.24 | 100% | 100% | 96.67% | 100% |
| 140×151 | 112.43 | 112.43 | 163.76 | 112.43 | 100% | 100% | 56.34% | 100% |
| 191×151 | 128.04 | 128.04 | - | 128.04 | 100% | 100% | 34% | 100% |
| 381×151 | 226.48 | 226.48 | - | 226.48 | 100% | 100% | 18% | 100% |
| 761×151 | 389.72 | 389.72 | - | 389.72 | 100% | 100% | 8% | 100% |
| 200×200 | 132.33 | 132.33 | 325.01 | 132.33 | 100% | 100% | 95.00% | 100% |
| 399×200 | 192.47 | 192.47 | 581.05 | 192.47 | 100% | 100% | 86.67% | 100% |
| 797×200 | 324.39 | 324.39 | 1422.23 | 324.39 | 100% | 100% | 50.00% | 100% |
| 1593×200 | 664.56 | 664.56 | - | 664.56 | 100% | 100% | 19.34% | 100% |

## 结论

面对高分辨率、大规模搜索的复杂栅格地图场景，特别是存在密集障碍物时，使用传统算法会面临搜索空间的爆炸性增长和高计算复杂度的挑战，针对这些问题，本文提出了一种基于区域分层的全局路径规划算法，将复杂度最高的全局搜索解耦为离线区域连接先验计算，有效降低了复杂栅格地图的规划计算复杂度。在保持规划路径有效性前提下在平均规划时间和平均扩展节点两项关键指标上获得了明显优势，伴随着栅格地图分辨率和复杂度的提高，本文算法的实时性优势越发明显，具有良好工程应用前景。

下一步我们将基于本文方法结合目标运动规律深入研究基于物理约束的复杂场景实时路径规划和基于强化学习的开放场景中高效规划方法。

参考文献:

1. Lin S, Liu A, Wang J, et al. A review of path-planning approaches for multiple mobile robots[J]. Machines, 2022, 10(9): 773.
2. Nokata M, Kato S, Feng L K, et al. Measurement of mechanical characteristics for soft materials by using medical robot with piezoelectric tactile sensors[C]. Nagoya, Japan: International Symposium on Micro-NanoMechatronics and Human Science (MHS). IEEE, 2016: 1-4.
3. Teng S, Deng P, Li Y, et al. Path planning for autonomous driving: The state of the art and perspectives[J]. arXiv preprint arXiv:2303.09824, 2023.
4. Karur K, Sharma N, Dharmatti C, et al. A survey of path planning algorithms for mobile robots[J]. Vehicles, 2021, 3(3): 448-468.
5. Oommen B, Iyengar S, Rao N, et al. Robot navigation in unknown terrains using learned visibility graphs. Part I: The disjoint convex obstacle case[J]. IEEE Journal on Robotics and Automation, 1987, 3(6): 672-681.
6. Raj R, Kos A. A comprehensive study of mobile robot: history, developments, applications, and future research perspectives[J]. Applied Sciences, 2022, 12(14): 6951.
7. Wang H, Yu Y, Yuan Q. Application of Dijkstra algorithm in robot path-planning[C]//2011 second international conference on mechanic automation and control engineering. IEEE, 2011: 1067-1069.
8. Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics, 4(2), 100-107.
9. Koenig S, Likhachev M. D^\* lite[J]. Aaai/iaai, 2002, 15: 476-483.
10. Dang V H, Thang N D, Viet H H, et al. Batch-Theta\* for path planning to the best goal in a goal set[J]. Advanced Robotics, 2015, 29(23): 1537-1550.
11. Shu, W.; Zhao, J.; Xie, Z.; Zhang, X.; Ma, X. Path planning for unmanned surface vessels based on improved A-star algorithm. J. Shanghai Marit. Univ. 2022, 43, 1–6.
12. Rubio F, Valero F, Llopis-Albert C. A review of mobile robots: Concepts, methods, theoretical framework, and applications[J]. International Journal of Advanced Robotic Systems, 2019, 16(2): 1729881419839596.
13. Bremermann H J. The evolution of intelligence: The nervous system as a model of its environment[M]. Washington, DC, USA: University of Washington, Department of Mathematics, 1958.
14. Sanchez-Ibanez J R, Perez-del-Pulgar C J, García-Cerezo A. Path planning for autonomous mobile robots: A review[J]. Sensors, 2021, 21(23): 7898.
15. Zhang Y, Gong D, Zhang J. Robot path planning in uncertain environment using multi-objective particle swarm optimization[J]. Neurocomputing, 2013, 103: 172-185.
16. Brand M, Masuda M, Wehner N, et al. Ant colony optimization algorithm for robot path planning[C]. Qinhuangdao, China: international conference on computer design and applications. IEEE, 2010, 3: V3-436-V3-440.
17. Tu J, Yang S X. Genetic algorithm based path planning for a mobile robot[C]//2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422). IEEE, 2003, 1: 1221-1226.
18. Lamini C, Benhlima S, Elbekri A. Genetic algorithm based approach for autonomous mobile robot path planning[J]. Procedia Computer Science, 2018, 127: 180-189.
19. Yu X, Chen W N, Gu T, et al. ACO-A\*: Ant colony optimization plus A\* for 3-D traveling in environments with dense obstacles[J]. IEEE Transactions on Evolutionary Computation, 2018, 23(4): 617-631.
20. Noreen I, Khan A, Habib Z. Optimal path planning using RRT\* based approaches: a survey and future directions[J]. International Journal of Advanced Computer Science and Applications, 2016, 7(11).
21. LaValle S M. Rapidly-exploring random trees: A new tool for path planning[J]. 1998.
22. Kavraki L E, Svestka P, Latombe J C, et al. Probabilistic roadmaps for path planning in high-dimensional configuration spaces[J]. IEEE transactions on Robotics and Automation, 1996, 12(4): 566-580.
23. Sandakalum T, Ang Jr M H. Motion planning for mobile manipulators—a systematic review[J]. Machines, 2022, 10(2): 97.
24. Ghosh D, Nandakumar G, Narayanan K, et al. Kinematic constraints based Bidirectional RRT (KB-RRT) with parameterized trajectories for robot path planning in cluttered environment[C]. Montreal, QC, Canada: International Conference on Robotics and Automation (ICRA). IEEE, 2019: 8627-8633.
25. Yuan C, Zhang W, Liu G, et al. A heuristic rapidly-exploring random trees method for manipulator motion planning[J]. IEEE Access, 2019, 8: 900-910.
26. Tang Z, Ma H. An overview of path planning algorithms[C]//IOP Conference Series: Earth and Environmental Science. IOP Publishing, 2021, 804(2): 022024.
27. Preparata F P, Shamos M I. Computational geometry: an introduction[M]. Springer Science & Business Media, 2012.
28. V. K. Balakrishnan, "Graph Theory," 1st ed., McGraw-Hill, 1997, ISBN 978-0-07-005489-9.
29. Y. Liu, C. Mo, B. Wang, and G. Zhao, "Combined UGV Path Planning Based on Improved D\* Lite and Jump Point Search Algorithms," in 2023 42nd Chinese Control Conference (CCC), Tianjin, China, 2023, pp.