《智能机器人设计》

机器人路径规划

机器人路径规划概述

路径规划技术作为机器人实现自主定位和路径导航的关键技术之一，是机器人领域的一个重要研究方向。

路径规划的应用领域：机器人导航避障、智能交通、无人机自主飞行、物流运输、GPS卫星导航等。

路径规划的算法：人工势场法、栅格法和全覆盖路径规划等。

路径规划的历史：路径规划问题的研究最早是由18世纪著名的哥尼斯堡“七桥问题”引出的。这个问题在世界范围得到了极大的关注，吸引了大量研究者进行探究。而后，由“七桥问题”衍生出“旅行商”问题以及其他相关问题。这些问题都可以抽象成一个由点和线构成的无向图，然后在图中以某点为起点不重复地遍历图中的其他节点并最终回到起点。这类问题均被证明是不能在多项式时间内进行求解的，即NP-hard problem。由于这些问题的成功解决，以及将问题抽象为图进行解决的方法得到不断应用，一系列与之类似的问题被不断提出，也引发了人们对图论领域中路径规划问题的不断关注和深入研究。

机器人路径规划概述

★路径规划定义

路径规划是机器人研究领域的一个重要分支。机器人的最优路径规划问题，就是依据某个或某些最优准则（如工作代价最小、行走路线最短、行走时间最短等），在其工作空间中找到一条从起点到目标点的、能避开障碍物的最优路径，其本质是在几个约束条件下得到最优可行解的问题。

路径规划结果的优劣，影响机器人执行任务的效率。机器人路径规划要实现3个任务：

- **任务1：**
  机器人规划出一条从起点到目标点的路径.
- **任务2：**
  使机器人的路径绕开空间中的障碍物.
- **任务3：**
  优化机器人路径，使其尽可能达到更短、更平滑的要求

机器人路径规划概述

★路径规划分类

按照机器人对周围环境信息的识别与对信息的掌握程度以及对不同种类障碍物的识别进行分类，可以将机器人路径规划分成4类：

第1类

在已知的比较熟悉的环境中，根据静态障碍物的位置对机器人的路径进行规划；

第2类

在未知的比较陌生的环境中，根据静态障碍物的位置对机器人的路径进行规划；

第3类

在已知的比较熟悉的环境中，根据动态障碍物的运行状态对机器人的路径进行规划；

第4类

在未知的比较陌生的环境中，根据动态障碍物的运行状态对机器人的路径进行规划。

机器人路径规划概述

★路径规划分类

根据路径应用目的的不同，如时间最优、路径最优、路径全遍历，将路径分为点对点的路径规划和全覆盖路径规划：

点对点的路径规划

点对点的路径规划指规划两点之间的路径。

全覆盖路径规划

是要找到一条能够遍历区域内的所有点，同时避开障碍物的路径，该方法要求较强的遍历性和低重复率，相关算法可以分为3类：
随机遍历策略、**沿边规划策略**和**漫步式探测路径规划**。

机器人路径规划概述

常见的路径规划算法主要包括：

对固体在退火过程中的各状态进行控制和模拟的模拟退火算法、采用禁忌表提高搜索效率的禁忌搜索算法、引入力学知识的虚拟人工势场法等；

图形学衍生算法，如引入构形空间，将问题放入高维空间的C-Space法、将空间进行分解并在得到的一系列小单元中进行问题求解的栅格法等；

与仿生学相结合形成的**智能仿生学算法**，如对蚂蚁群体觅食活动规律进行仿真的蚁群算法、与生物神经系统相结合模拟生物神经反馈系统的神经网络算法、借用生物遗传学观点模拟适者生存的遗传算法等；

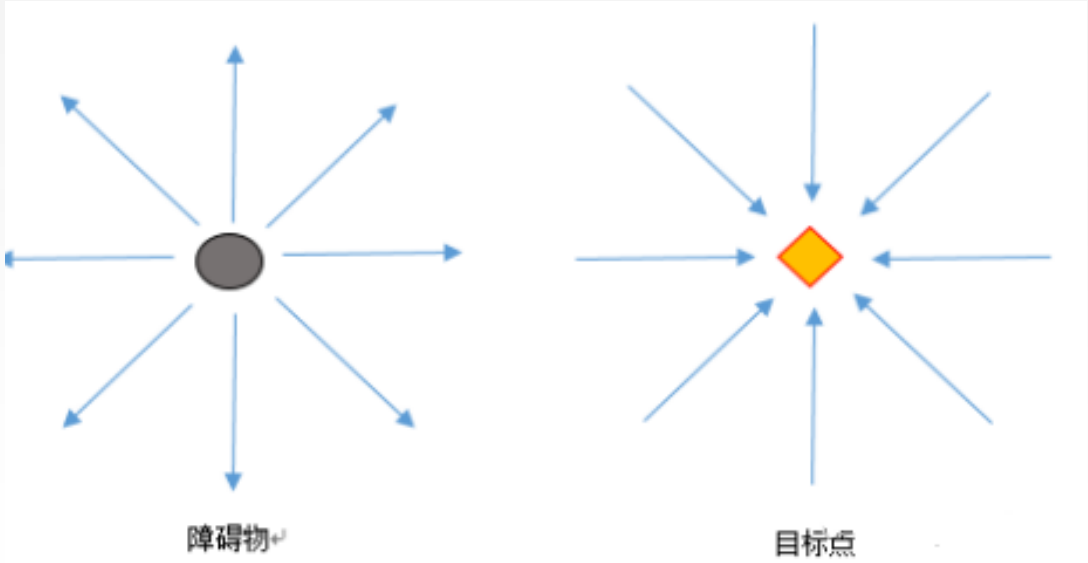
一些具有很强搜索路径能力，并具有很好的搜索效率的**路径规划算法**，如寻找单源最短路径的著名算法——迪杰斯特拉算法、引入评价函数的启发式路径扩展式算法——A*(A-Star)等。

每个算法都有自身设计的优点和缺点，适用于不同的范围和领域。因此，针对不同的应用环境和使用条件及环境约束等，应该选择适当的算法进行求解。随着对各种环境的不断研究和对应用的拓展，基于各类算法的改进算法不断被提出，各种改进算法也应用在具体的领域中，并取得较好的效果

人工势场法路径规划

★人工势场法的基本概念

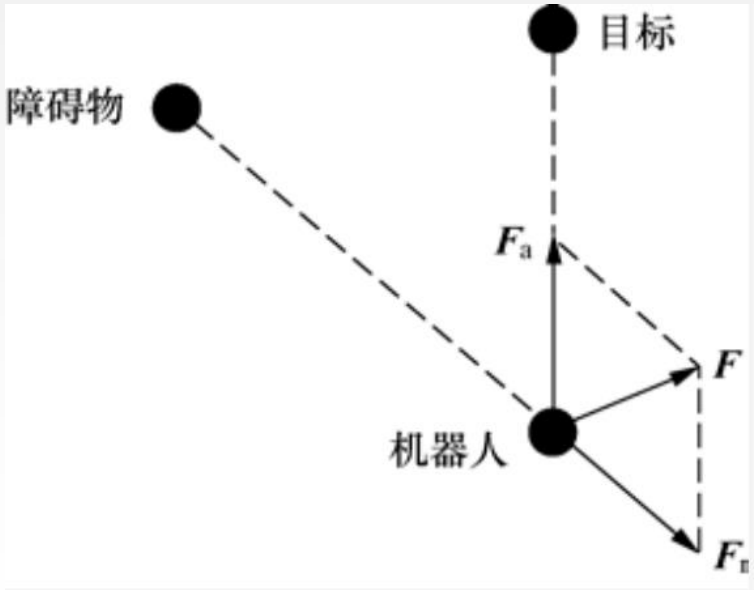
人工势场法的基本概念是将机器人或者机械手在周围环境中的运动设计成一种在人造引力场中的运动，目标位置对于机器人相当于一个吸引极点，而障碍物对机器人相当于一个排斥面，机器人在目标位置和障碍物两者“作用力”的合力效果下移动，该方法的难点是如何设计“引力场”。人工势场法进行避障具有较好的实时性和实用性，该方法可以较好地适应环境的变化，但存在类似于陷入局部最优解等许多难题。



人工势场法路径规划

★ 势场法概述

人工势场法的具体实现方法为：首先，在机器人运行环境空间中构建一个人工虚拟势场，该势场由两部分组成，其中一部分是由目标点对机器人产生的引力场，方向为由机器人指向目标点；另一部分是由障碍物对机器人产生的斥力场，方向为由障碍物指向机器人。运行空间的总势场为引力场和斥力场一起叠加作用，从而通过引力和斥力的合力控制机器人的移动。



图为人工势场法受力分析。目标点对机器人施加引力，机器人越靠近目标点，引力值F越大，引力方向指向目标点位置。障碍物对机器人施加斥力，机器人越靠近障碍物，斥力值F越大，斥力方向由障碍物指向机器人。

机器人的最终运动方向为引力和斥力的合力方向，如式：

$$\boldsymbol{F} = \boldsymbol{F}_{\mathrm{a}} + \boldsymbol{F}_{\mathrm{r}}$$

人工势场法路径规划

★ 势场函数的建立

这里介绍一种经典的人工势场法势场函数。设给定二维坐标空间$X_{max}*Y_{max}$，则点q可表示为：$q=[x\ y]^T$，其中$0\leqslant x\leqslant X_{max}$，$0\leqslant y\leqslant Y_{max}$。

在虚拟势场中，

目标点对机器人产生的引力势场函数被定义为Uatt(q)，目标点与机器人距离越远，势能越大，反之势能越小。

障碍物对机器人产生的斥力势场函数被定义为Urep(q)，产生的势场大小与障碍物和机器人的距离有关，距离越远，则势能越小，反之势能越大。

人工势场法路径规划

★ 势场函数的建立

引力函数

引力函数受机器人与目标点的距离影响，目标点与机器人的距离越远，其所受的势能越大；目标点与机器人的距离越近，其所受的势能越小。当机器人势能为零时，则表明机器人到达目标点位置。引力势函数表示为：

$$U_{att}(q)=\frac{1}{2}\eta\left\|X-X_g\right\|^2$$

其中，η为正比例位置增益系数；X为机器人现在的位置，X_g为目标点的位置；

$|X-X_g|^2$为机器人与目标点之间的相对距离的二范数，则由引力场产生的引力为：

$$F_{att}(q)=-\eta\left\|X-X_g\right\|\frac{\partial\left\|X-X_g\right\|}{\partial X}$$

机器人的引力的方向为机器人指向目标点。

人工势场法路径规划

★ 势场函数的建立

斥力函数

斥力函数受机器人与障碍物的距离影响，当障碍物与机器人的距离越远时，其所受的排斥力越小；当障碍物与机器人的距离越近时，其所受的排斥力越大。当机器人势能为零时，则表明机器人已经脱离障碍物的影响范围，最大影响距离设定为 ρ_0。斥力势函数表示为：

$$U_{rep}(q)=\begin{cases}\frac{1}{2}k\left(\frac{1}{(X-X_0)}-\frac{1}{\rho_0}\right)^2, & X-X_0\leqslant\rho_0\\ 0, & X-X_0>\rho_0\end{cases}$$

其中，k为正比例位置增益系数；ρ_0为正常数，表示在机器人周围障碍物区域对机器人产生作用的最大距离；为路障的位置，则由斥力场产生的斥力的大小为：

$$F_{rep}(q)=\begin{cases}k\left(\frac{1}{X-X_0}-\frac{1}{\rho_0}\right)\frac{1}{(X-X_0)^2}, & X\leqslant X_g\quad\rho_0\\ 0, & X-X_g>\rho_0\end{cases}$$

机器人的斥力的方向为障碍物指向机器人。

人工势场法路径规划

★ 势场函数的建立

全局势场函数

根据上述定义的引力场函数和斥力场函数，可以得到整个运行空间的全局势场函数，机器人的全局势场大小为机器人所受的斥力势场和引力势场之和，故全局势场总函数为：

$$U(\boldsymbol{q}) = U_{\text{att}}(\boldsymbol{q}) + U_{\text{req}}(\boldsymbol{q})$$

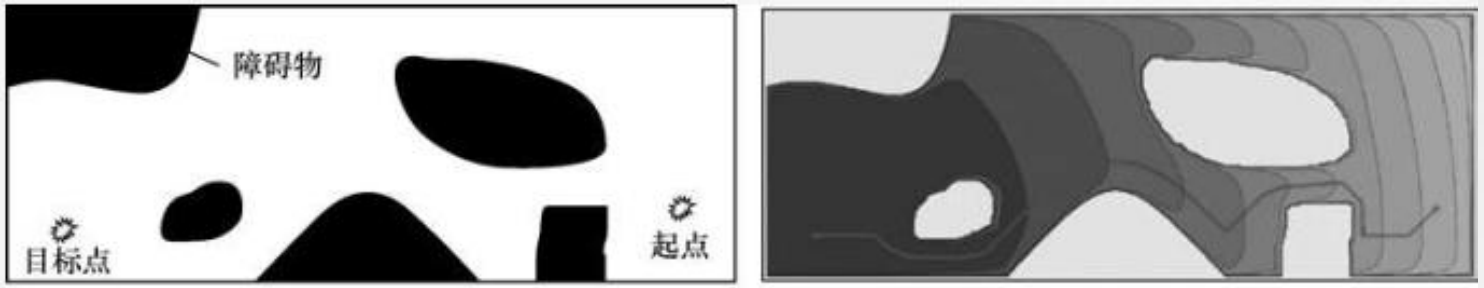
机器人所受的合力为：

$$\begin{aligned} \boldsymbol{F}(\boldsymbol{q}) &= -\nabla U(\boldsymbol{q}) \\ &= \boldsymbol{F}_{\text{att}}(\boldsymbol{q}) + \boldsymbol{F}_{\text{req}}(\boldsymbol{q}) \end{aligned}$$

人工势场法路径规划

★ 势场函数的建立

图为人工势场法路径规划的结果示意图。已知起点、目标点和环境信息，规划的一条路径。



（a）地图　　（b）路径规划结果

人工势场法实际上是试图使障碍物的分布情况及其形状等信息反映在环境的每一点的势场值中，机器人的运动由机器人当前的位置所承受的势场（即其梯度方向）决定。所以，它与全局规划相比具有计算量小、实时性好的特点。人工势场法在较为简单的环境中具有不错的路径规划能力，从起点到目标点的无碰撞路径可以简单快捷地生成。但是，一旦环境变得复杂，人工势场法就会出现相应的问题，如极限值点造成的局部最优解问题，即局部障碍物过多，从而引力与斥力抵消导致机器人判定此时合力为0，生成局部最优解。

栅格法路径规划

栅格法在进行路径规划时采用栅格（grid）表示地图。环境地图Map由n个栅格map构成：

$$\mathrm{Map}=\{\mathrm{map}_1,\mathrm{map}_2,\mathrm{map}_3\cdots,\mathrm{map}_n\}$$

其中，map的值为-1、0或1，map=-1表示栅格为未知状态，map=0表示栅格为空闲状态，map=1表示栅格为占据状态。栅格法将机器人规划空间分解成一系列的栅格单元，栅格的一致性和规范性使得栅格空间中的邻接关系简单化。赋予每个栅格一个通行因子后，路径规划问题就变成在栅格地图中寻求两个栅格节点间的最优路径问题。

栅格法路径规划

★ 状态空间搜索

状态空间搜索就是从初始状态到目标状态寻找路径的过程。

由于过程中分支很多，主要是由求解条件的不确定性、不完备性造成的，这使得求解的路径很多，从而构成了一个图，也就是状态空间。问题的求解就是在这个图中找到一条从起点到目标点的路径。这个寻找的过程就是状态空间搜索，一般分为两类：**深度优先搜索**（depthfirstsearch，DFS）和**广度优先搜索**（breadth first search，BFS）。

栅格法路径规划

★ 状态空间搜索

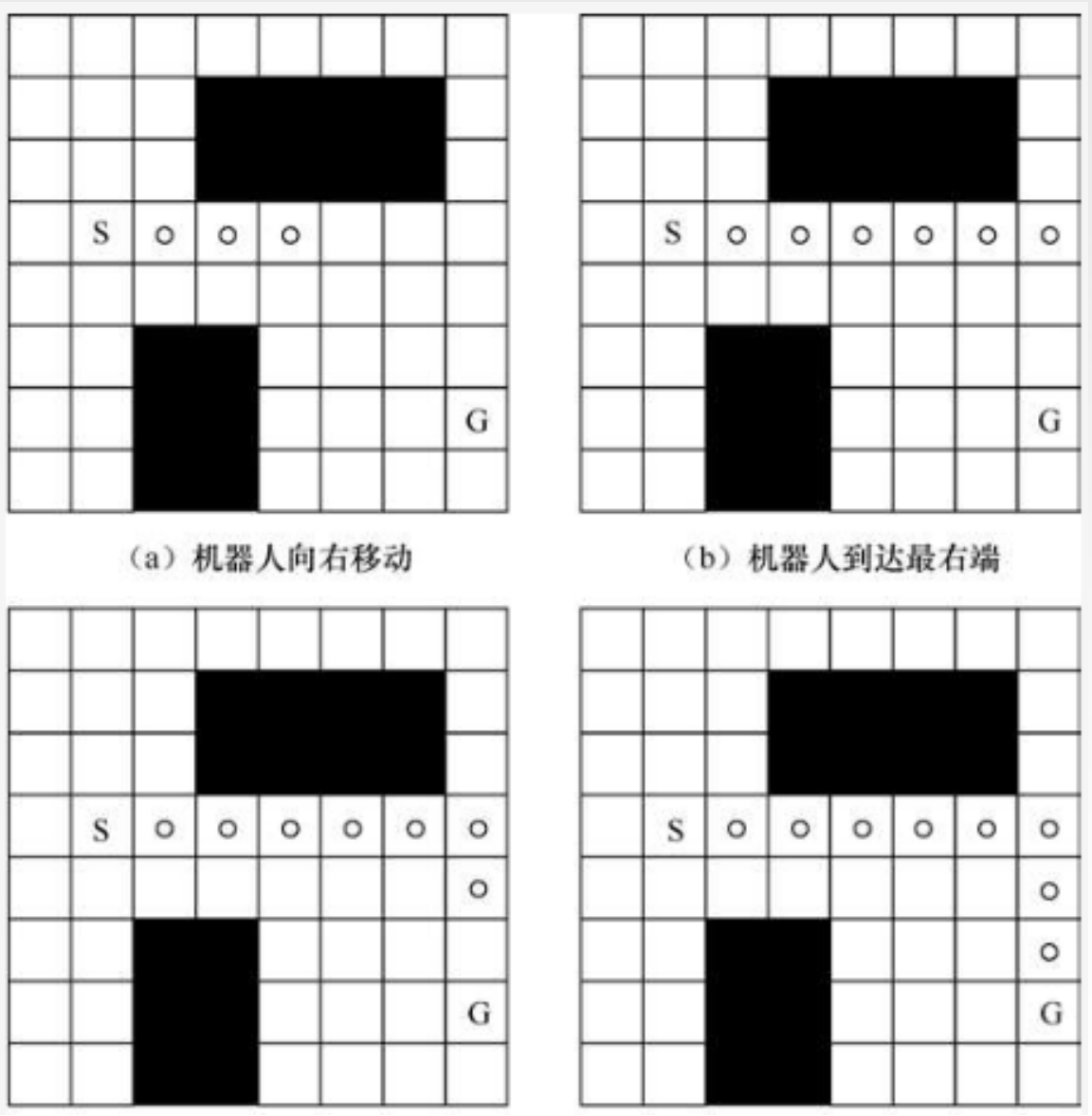
深度优先搜索

深度优先搜索从起点出发，依次从它周围的各个邻近节点中先选取一个节点向下搜索，沿着一个顺序一直搜索，直至到达目标点。如果中间遇到障碍点，则返回上一节点选取另外一个邻近节点向下搜索。这个方法依据每条路径搜索完一个树的分支后，再搜索另一个分支，以搜索更接近目标点位置的路线，因此称为深度优先。其搜索过程是一个递归的过程，有一定的顺序。

栅格法路径规划

★ 状态空间搜索

如图所示，S代表机器人的起点，G代表机器人的目标点，采用四邻域扩展法（即只有四个可移动方向），并且按照正右、正下、正左、正上的顺序搜索。图(a)表示机器人先向右移动，图(b)表示中机器人到达最右端，图(c)表示机器人开始向下移动，图(d)表示机器人到达目标点G。

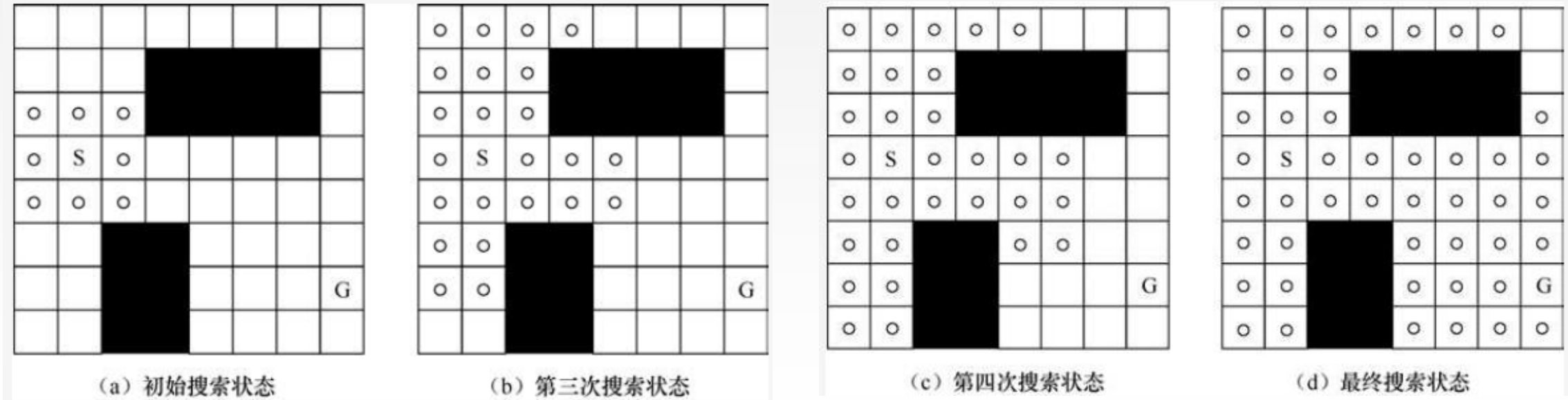


（a）机器人向右移动　（b）机器人到达最右端

栅格法路径规划

★ 状态空间搜索

广度优先搜索从起点出发，先将最靠近起点周围的方向都走一遍，之后再进一步扩散搜索范围，直到找到目标点。一般用队列数据结构辅助实现广度优先搜索算法。如图所示，S代表机器人的起点，G代表机器人的目标点。图(a)表示先将起点周边的栅格搜索一遍，(b)和图(c)展示了当最近的一层搜索完之后再扩展到下一层的过程，直到找到目标点，如图(d)所示



(a) 初始搜索状态　(b) 第三次搜索状态　(c) 第四次搜索状态　(d) 最终搜索状态

栅格法路径规划

★ A*算法路径规划

A*算法属于具有鲜明启发式特征的搜索算法，因为其强大的灵活性，以及对不同路况超强的适应能力，所以在路径规划搜索中广受欢迎。A*算法通过全面评价各个节点的代价值，并把这些代价值进行比较，选取代价值最小的节点作为扩展节点，接着以此节点继续扩展下一个节点，直至目标点被选为扩展节点，从而产生从起点到目标点代价值最小的路径。

A*算法的评价函数为：

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

其中，f(n)表示当前节点与初始节点，以及目标节点路径的估价函数；g(n)表示初始节点与当前节点n之间的真实代价；h(n)表示从当前节点n至目标节点路径的估算代价。

栅格法路径规划

★ A*算法路径规划

A*算法中，h(n)在评价函数中起关键性作用，决定了A*算法效率的高低。若h(n)的预算代价小于节点到目标点的真实代价，那么此时A*算法同样可以达到搜索出最优路径的目的。h(n)越小，A*算法经过扩展得到的节点就会增加，此时的运行速率就会降低。若h(n)的估算距离与真实代价相等，此时A*算法就可以更快地寻找到最佳路径，此时的速率将最快。若h(n)付出的代价高于某一节点与目标点的代价，此时可能无法寻找到最佳路径，但是速率却提升了。另一种情况是，若h(n)比g(n)大很多，此时g(n)的作用基本被忽略，算法就变成了广度优先搜索算法。

栅格法路径规划

★ A*算法路径规划

A*算法结合具体路径规划要达到的目的，设计与之相适应的启发函数，从而使搜索方向更智能化地与目标状态越来越接近。A*算法的执行步骤如下所示。

(1)创建两个线性表，分别为OPEN表和CLOSE表，OPEN表的作用是保存搜索过程中遇到的扩展节点，同时将这些节点按代价值的大小进行排序。CLOSE表的作用是保存OPEN表中已扩展的代价值最小的节点。

(2)判断起点与目标点的输入有效性并验证两点是否重合。若输入无效或两点一致，则退出路径规划。

(3)计算起点启发函数f值，并将起点存入OPEN表中。

(4)将OPEN表中f值最小的节点取出，该节点作为规划路径的节点，并将该节点存入CLOSE表。

栅格法路径规划

★ A*算法路径规划

(5)若插入CLOSE表中的节点就是目标节点，则路径规划成功。逆向循环取出该节点的父节点（步骤(4)），形成路径，退出算法，否则，继续执行。

(6)获得该节点的所有可通行的节点，针对每一个可通行的节点做以下考察：

①若CLOSE表存在该节点，则跳过该节点，不再考察。

②若OPEN表不存在该节点，则通过启发函数计算该节点的f、g、h值，将该节点的父节点（步骤①）设置为规划路径的节点，并将该节点添加到OPEN表中。

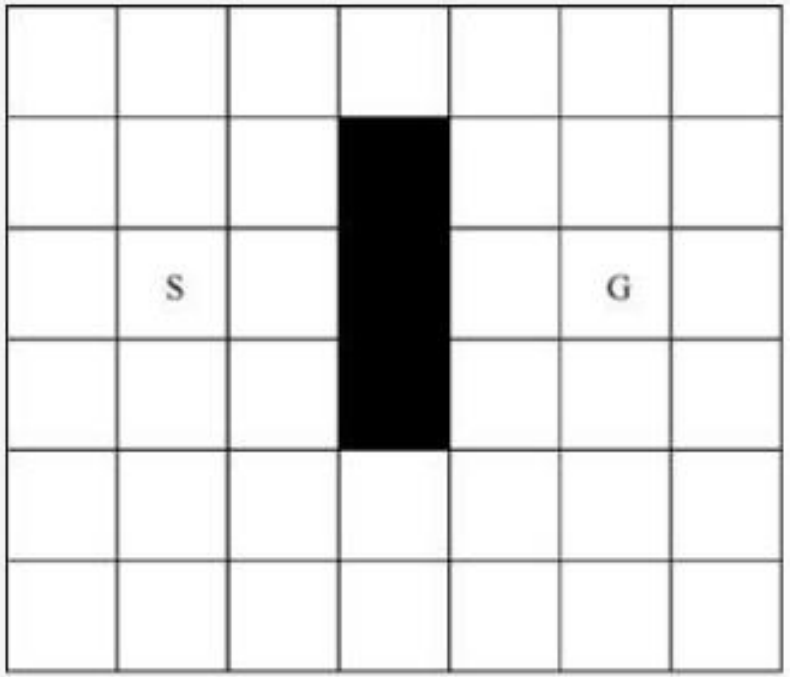
③若OPEN表已经存在该节点，则重新计算f、g、h值，比较新旧f值的大小，若新计算的f值更小，则使用新计算的f、g、h值替换旧的一组值，并将父节点设置为规划路径的节点。

(7)若Open表不为空，则回到步骤(3)，否则，寻径失败，起点与目标点之间没有可通行的路径。

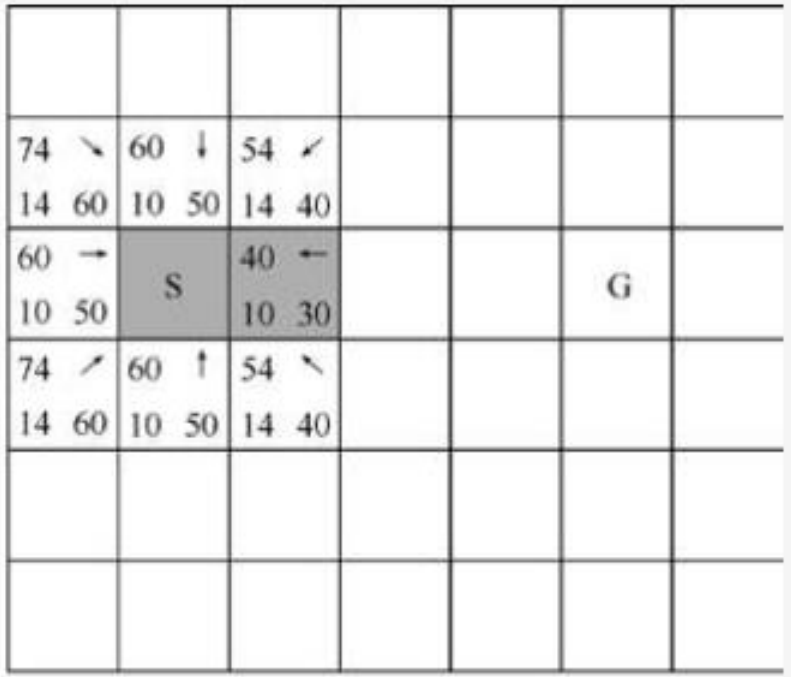
栅格法路径规划

★ A*算法路径规划 - 例

如图(a)所示，先创建一个地图，地图中黑色的区域表示障碍物，已知起点S和目标点G，规划一条从起点S到目标点G的路径。在本例中，横向和纵向移动一个栅格的代价为10，对角方向移动一个栅格的代价为14。在一个栅格中，左下角的值表示g值，右下角的值表示h值，左上角的值表示f值。箭头的指向为该栅格节点的父节点。



（a）初始搜索状态

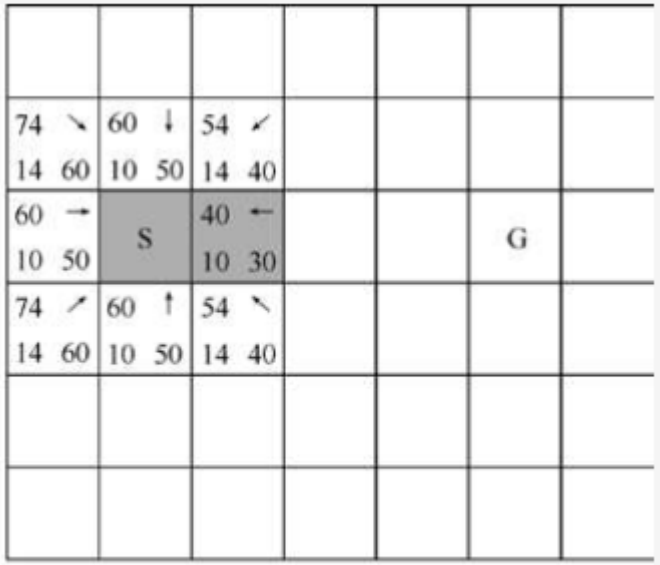


（b）第一次搜索状态

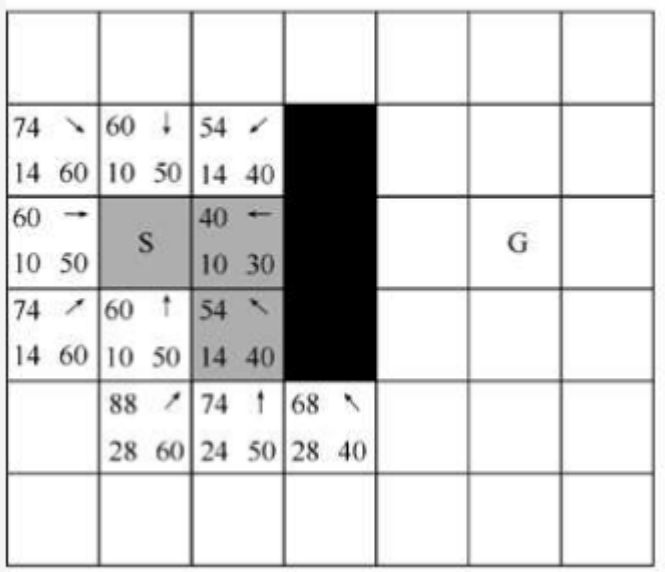
栅格法路径规划

★ A*算法路径规划 - 例

初始时，OPEN表中有1个节点，CLOSE表中有0个节点。(b)为搜索与起点相邻的栅格，选出值最小的节点并搜索该节点相邻的栅格，此时OPEN表中有7个节点，CLOSE表中有2个节点。图(c)为继续选择f值最小的节点并搜索该节点相邻的栅格，此时OPEN表中有9个节点，CLOSE表中有3个节点。图4.5(d)~(g)为继续搜索得到的结果。最后，规划得到的路径如图4.5(h)所示，从目标节点开始寻找父节点，直到到达起点节点。



（b）第一次搜索状态

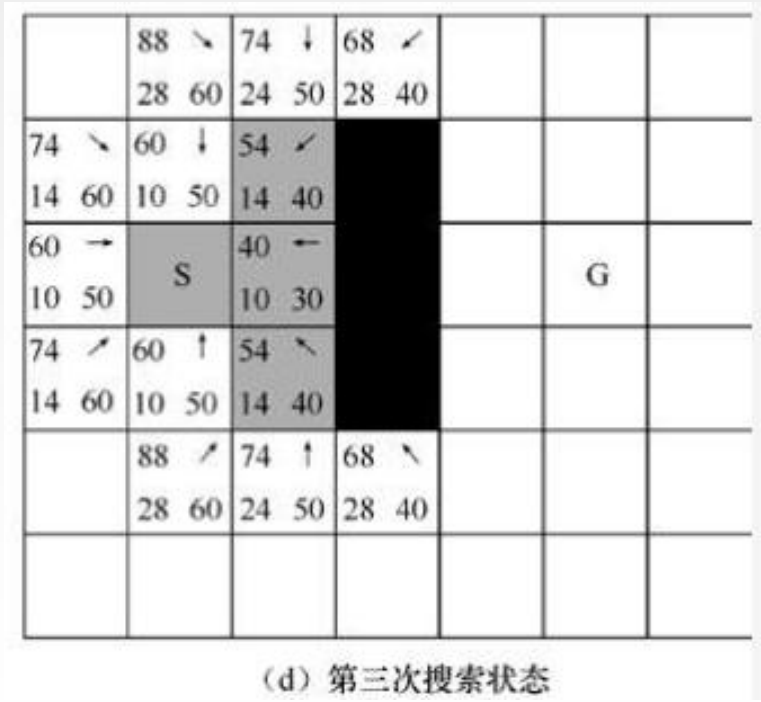


（c）第二次搜索状态

栅格法路径规划

★ A*算法路径规划 - 例

图(d)为继续搜索得到的结果。最后，搜索到目标点G，并规划得到的路径，从目标节点开始寻找父节点，直到到达起点节点。



（d）第三次搜索状态

栅格法路径规划

★ D*算法路径规划

D*算法是一种高效的动态路径规划方法，是在A*算法的基础上对二次路径规划进行改进而得到的启发式搜索算法。D*算法利用启发函数计算节点的代价估计值，每次选择具有最小代价估计值的节点作为最佳的扩展方向，并迭代循环计算周围节点的代价估计值。如果失败，则选择其他路径进行搜索，直到找到目标节点为止。当机器人在前进的过程中遇到动态障碍物时，D*算法利用路径规划已有的节点信息进行二次规划，重新找到从当前位置到目标位置的优化路径。

A*算法是从起始节点开始扇形展开搜索，计算的是当前节点距离起始节点的路径长度，当进行二次路径规划时，前一次路径规划计算的节点信息作废。而D*算法从目标节点开始扇形展开搜索，计算的节点信息为距离目标节点的路径长度，当进行二次路径规划时，此信息可以再次利用，减少了对相同数据的重复计算，提高了二次路径规划的效率。

栅格法路径规划

★ D*算法路径规划

D*算法的评价函数为：

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

其中，f(n)表示初始节点与当前节点及目标节点三者的估价函数，g(n)表示从目标节点到当前扩展节点的路径代价，h(n)表示从节点n到初始节点的最短路径代价的估计值，一般选取曼哈顿距离。假设地图中有点(X_n, Y_n)和点(X_g, Y_g)，则这两点的曼哈顿距离可表示为：

$$h(n)=|x_n-x_g|+|y_n-y_g|$$

栅格法路径规划

★ D*算法路径规划

D*算法的实现步骤为：从目标节点开始遍历搜索，直到找到初始节点。分别创建OPEN表和CLOSE表两个空链表。D*算法的具体实施步骤如下：

(1)创建两个空链表OPEN表和CLOSE表。

(2)将目标节点加入OPEN表中。

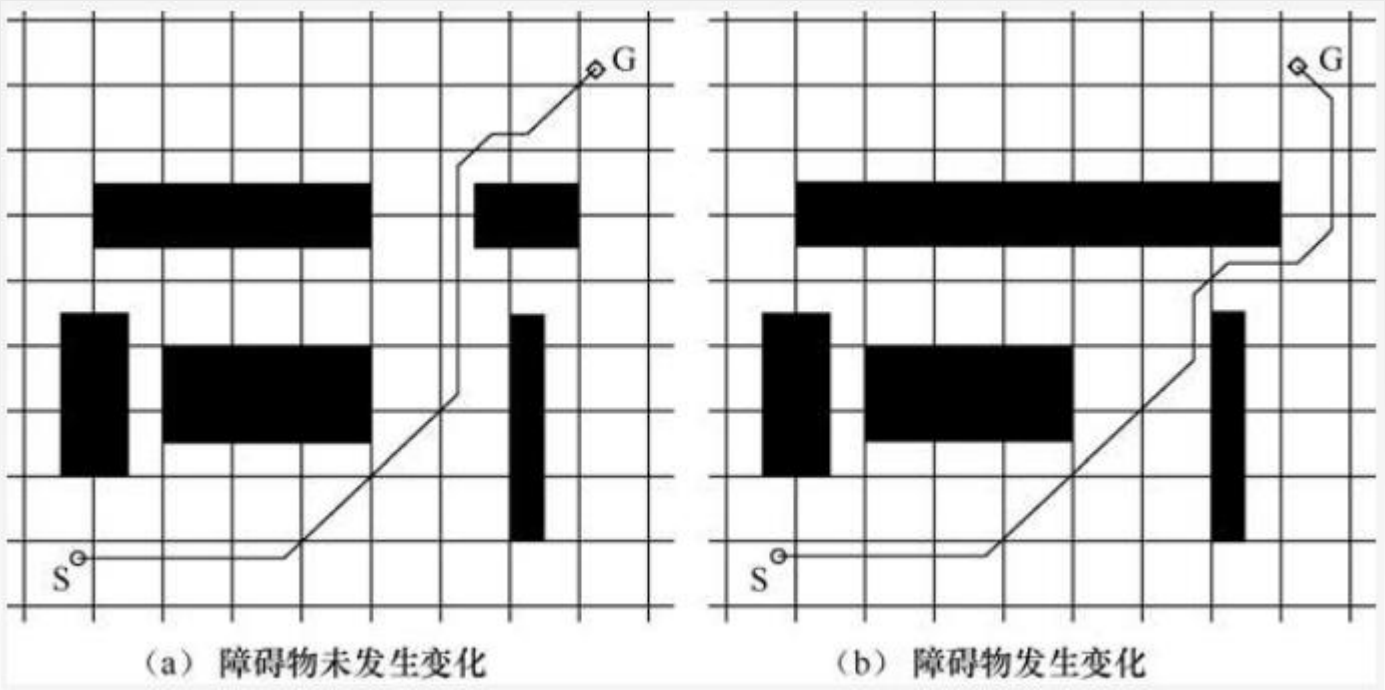
(3)检测OPEN表是否为空，若空，则路径规划失败；若非空，

则选择OPEN表中代价估计值最小的节点作为待扩展节点，并将待扩展节点加入CLOSE表中。若待扩展节点是初始节点，则路径规划结束，从初始节点开始沿着后向指针找到最优路径，算法结束；若待扩展节点不是初始节点，则以该节点为父节点，扩展该节点周围所有的子节点。

栅格法路径规划

★ D*算法路径规划-例

如图所示，地图中黑色的区域表示障碍物，已知起点S和目标点G，规划一条从起点S到目标点G的路径。图(a)为障碍物未发生变化的路径规划示意图，图(b)为障碍物发生变化后的路径规划示意图。由此可见，D*算法在发现路径中存在新的障碍物时，对于目标位置到新障碍物范围内的路径节点，新的障碍物是不会影响其到目标点的路径的，新障碍物只会影响物体所在位置到障碍物之间范围的节点的路径。这时通过新的障碍物周围的节点加入OPEN表中进行处理，然后向物体所在位置进行传播，能够最大程度地减少计算开销。

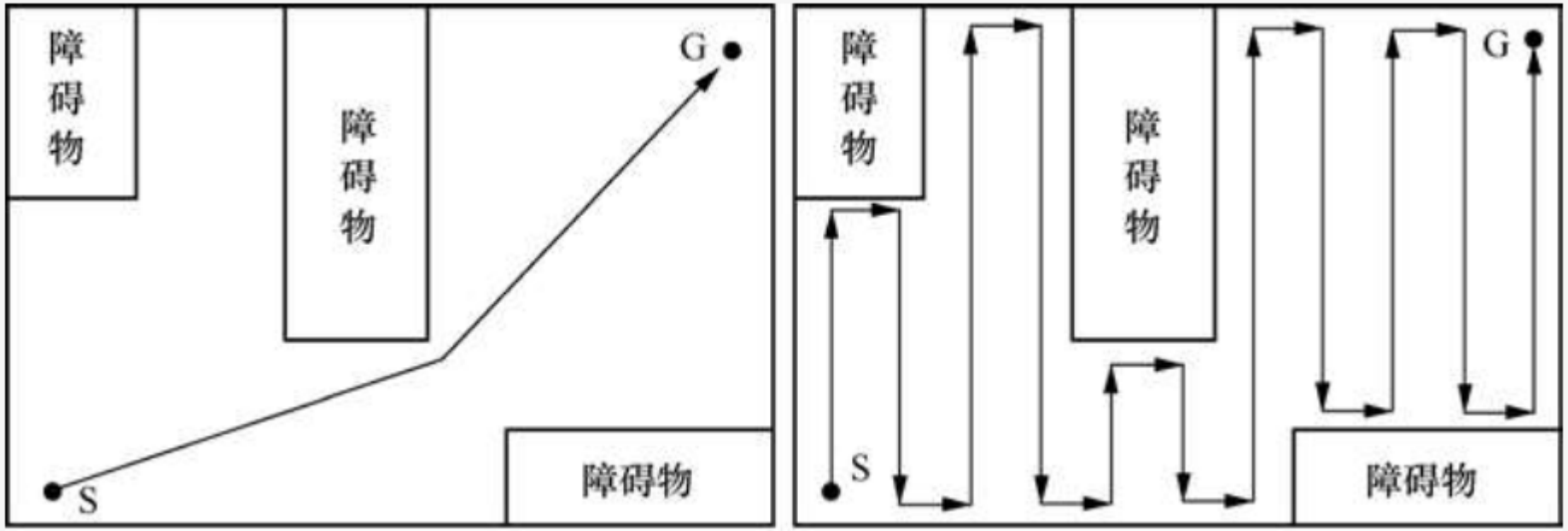


（a） 障碍物未发生变化　　（b） 障碍物发生变化

全覆盖路径规划

★ 全覆盖路径规划问题

根据规划方式不同，路径规划分为点到点的路径规划和全覆盖路径规划（Complete Coverage Path Planning，CCPP）。点到点的路径规划要求从一个给定的起点出发，规划出一条到任意目标点无碰撞的最优路径；全覆盖路径规划则要求规划出一条遍历给定环境中空闲区域的全覆盖路径。两种路径规划的对比如图所示。



（a） 点到点的路径规划　　（b） 全覆盖路径规划

全覆盖路径规划

★ 全覆盖路径规划问题

一般来说，全覆盖路径规划的结果可以由区域覆盖率、路径重复率及总行程这3类技术指标衡量。目前，比较成熟的全覆盖路径规划算法有**单元分解法**和**栅格地图法**等。

现实生活中，全覆盖路径规划技术有广泛的应用，从生活服务到农业生产，从资源勘探到军事侦察都有广泛的应用前景，具体包含以下4个方面。

(1) 服务方面：室内外清洁、草坪修剪、铲除积雪、农林业的耕作等。

(2) 军事方面：战区扫雷布雷、战场环境监测和巡逻等。

(3) 安全方面：对受污染区域的安全监控、灾难现场救援等。

(4) 资源方面：陆地、海底的地形测绘，外星探索，地理数据采集等。

全覆盖路径规划

★ 单元分解法

单元分解法是一种基于空间分解地图的方法，首先需要将全部地图空间分解为一系列不相交的子区域，再按照一定的顺序依次对各个子区域进行遍历，进而完成对整个区域的遍历。这种化整为零，逐个攻克的思想有效降低了复杂环境下机器人遍历的难度。

单元分解法的具体实现有多种方式，但是都需要经历以下几个步骤：

- **区域分解**
- **子区域衔接**
- **子区域覆盖**

全覆盖路径规划

★ 单元分解法

区域分解

区域分解法是将计算域分解为若干子域，分别求解再进行综合的一种数值计算方法。这种方法便于在各子域中运用适应其特点的数学模型、计算方法和格式，使总体解更符合实际，并有利于采用并行算法加快运算速度。

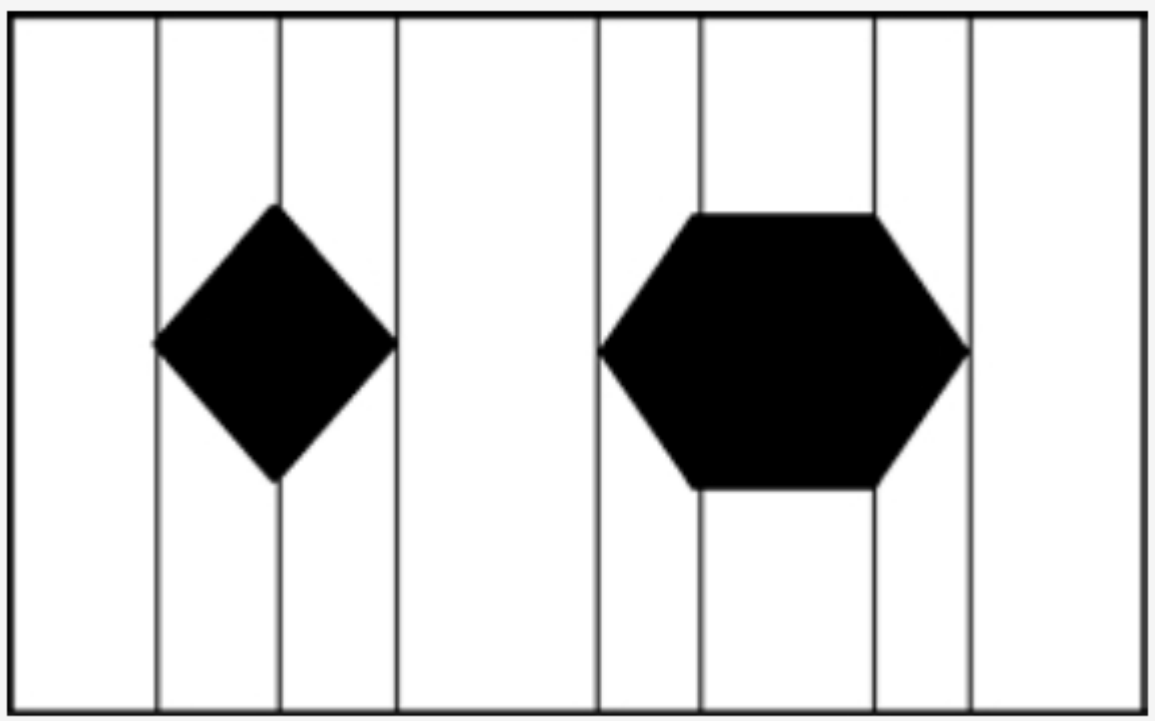
在单元分解法中，主要的区域分解方法有**梯形分解法**、**牛耕式分解法**和**Morse分解法**等。

全覆盖路径规划

★ 单元分解法-区域分解

(1) 梯形分解法

梯形分解法是一种简单的单元分解法，通常用在二维环境中且环境中的障碍物一般为多边形。用一条竖直的垂线自左向右（自右向左）扫过目标区域，每当直线经过目标区域中多边形障碍物的一个顶点，便会产生一个子区域，通常被分解的每个子区域都是呈梯形，如图所示。在每个子区域中，机器人仅需要通过简单的往返运动便可以完成覆盖，按照事先规划好的顺序依次遍历每个子区域，便可以完成对整个空间的全覆盖。



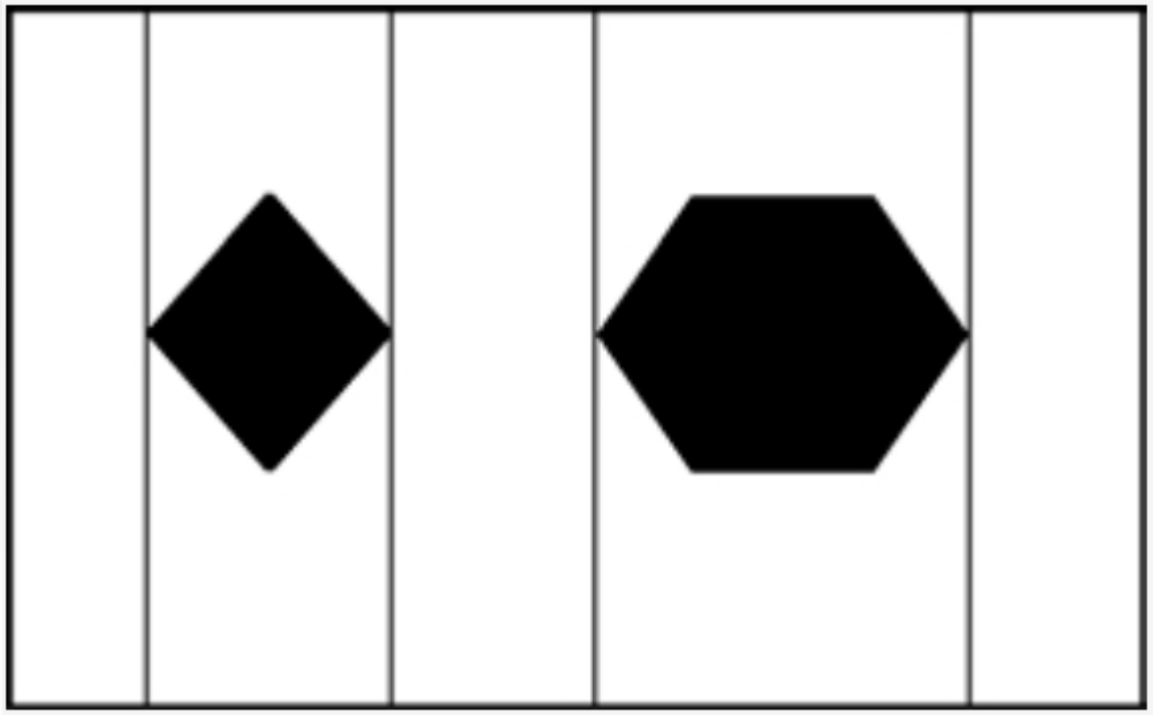
梯形分解法

全覆盖路径规划

★ 单元分解法-区域分解

(2) 牛耕式分解法

梯形分解的不足之处是会产生过多的子区域，从而造成许多不必要的往返运动。牛耕式分解法是对梯形分解法的改进。这种方法的思路是用一条垂线自左向右地扫过目标区域，当垂线的连通性发生改变时，生成新的区域，当连通性增加时，旧区域结束，两个或多个新的子区域生成；相反，连通性减少时，多个旧的子区域结束，新的子区域生成，如图所示。牛耕式分解法有效减少了分解后子区域的数目，从而减少了覆盖过程中的重复率。但是，该算法仅适用于环境中的障碍物均为多边形的情况。



牛耕式分解法

全覆盖路径规划

★ 单元分解法-区域分解

(3)Morse分解法

为了解决场景内存在不规则障碍物的情况，在牛耕式分解法的基础上提出了Morse分解法。该方法通过使用Morse函数计算与障碍物边界的切点，从而形成新的子单元，当选择的计算函数不同时，产生的子单元的形状也将不同。图(a)和图(b)分别为基于Morse函数h(x, y)=x和基于Morse函数h(x, y)=y的分解结果。

（a）基于函数$h(x, y)=x$的分解结构　（b）基于函数$h(x, y)=y$的分解结构

该算法适用于基于传感器的全覆盖路径规划，机器人利用传感器信息在线地寻找障碍物关键点，可通过构造不同的Morse方程寻求最合理的区域分解方式。但是，该算法要求待覆盖区域中障碍物的边界为光滑曲线，否则将造成关键点的退化，给传感器的搜索增加难度。

全覆盖路径规划

★ 单元分解法

子区域衔接

对目标区域进行划分后，便到了邻接图的建立阶段，即对各个子区域衔接的阶段。该阶段关系到机器人在整个遍历过程中覆盖的重复率和遗漏率的问题，因此，对衔接路径的有效规划能够明显地提高机器人全区域覆盖的效率。当前，全覆盖路径规划中邻接图衔接有基于改进旅行商问题的方法和蚁群优化方法等。

全覆盖路径规划

★ 单元分解法

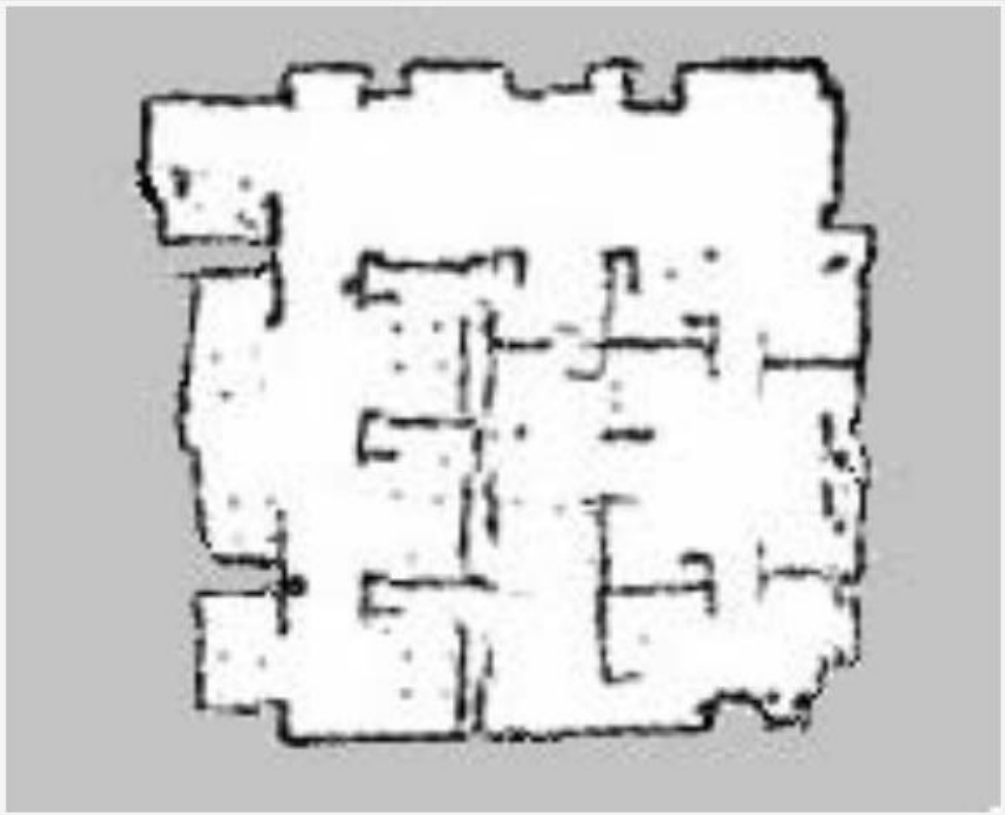
子区域覆盖

子区域覆盖是对整个目标区域进行全覆盖的缩小和简化，不同的是，对于静态环境而言，子区域内无障碍物。因此，对于子区域的覆盖，无须考虑避障的问题，重点是能够提高覆盖效率，即在最短的时间内以最短的路径覆盖各个子区域。

全覆盖路径规划

★ 栅格地图法

栅格地图法全覆盖路径规划方法是指将整个环境分解为栅格地图并进行对应的全覆盖路径规划。栅格地图具有直观、简明、易于创建的优点，而且主要位置信息由数组维护，便于与传感器信息融合。如图所示，栅格地图将环境空间分解为一系列互不重叠的大小相同的栅格，每个栅格都会分配一个概率值，表示其被障碍物占据的概率大小。栅格地图中的栅格有3种状态：空闲、占据和未知。栅格地图的优势在于创建和维护容易，尽量保留了全局环境的各种信息。



栅格地图

目前，栅格地图的全覆盖路径规划有**基于波前算法、基于生成树算法，**以及**基于神经网络**进行研究的方法。

全覆盖路径规划

★ 栅格地图法--基于波前算法的全覆盖路径规划

该方法首先需要在栅格地图上标记一个开始单元和一个目标单元,如图中的S和G所示，并像波浪一样从目标单元开始扩散，给每个空闲单元格赋一个特殊的权值。具体做法是：将目标单元的权值设为0,0周边空闲单元的权值设为1,1周边空闲单元的权值设为2，以此类推，直到开始单元也被赋上权值才停止。图是一张基于栅格地图的波阵图。

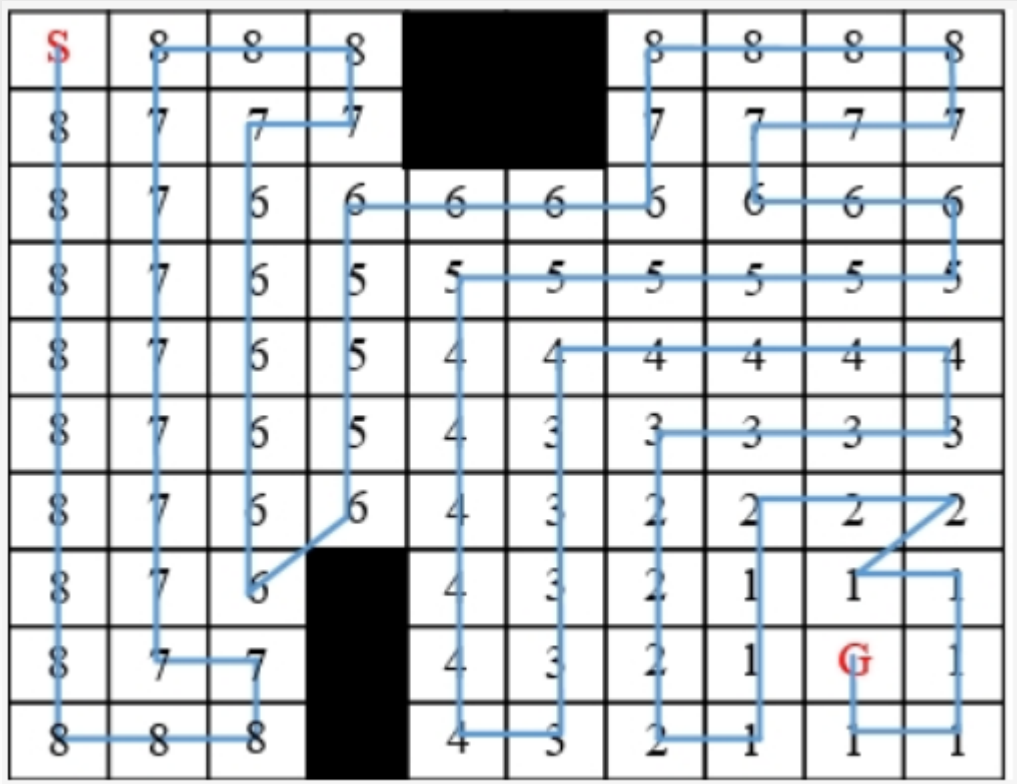
S	8	8	8	■	■	8	8	8	8
8	7	7	7	■	■	7	7	7	7
8	7	6	6	6	6	6	6	6	6
8	7	6	5	5	5	5	5	5	5
8	7	6	5	4	4	4	4	4	4
8	7	6	5	4	3	3	3	3	3
8	7	6	5	4	3	2	2	2	2
8	7	6	■	4	3	2	1	1	1
8	7	7	■	4	3	2	1	G	1
8	8	8	■	4	3	2	1	1	1

基于栅格地图的波阵图

全覆盖路径规划

★ 栅格地图法--基于波前算法的全覆盖路径规划

生成波阵图后，便可以开始栅格为起点规划覆盖路径，首先选择开始栅格周围权值最高的栅格，如果此时恰好有两个及两个以上权值相同且未被访问的栅格，则随机选取一个栅格；接着，以上一时刻选择的栅格为新的起点，再次寻找该栅格周围权值最大的栅格；不断重复以上步骤，直至到达目标点G为止。这个寻找覆盖路径的过程类似于使用伪梯度的方式，从起点逐渐递减至目标点。图中展示了生成的一条覆盖路径，如图蓝色线段所示。该覆盖算法最大的特点是起点和目标点可以根据实际需要任意设定，设定好之后按照生成的覆盖路径进行遍历，机器人可以高效地完成该区域的覆盖任务。

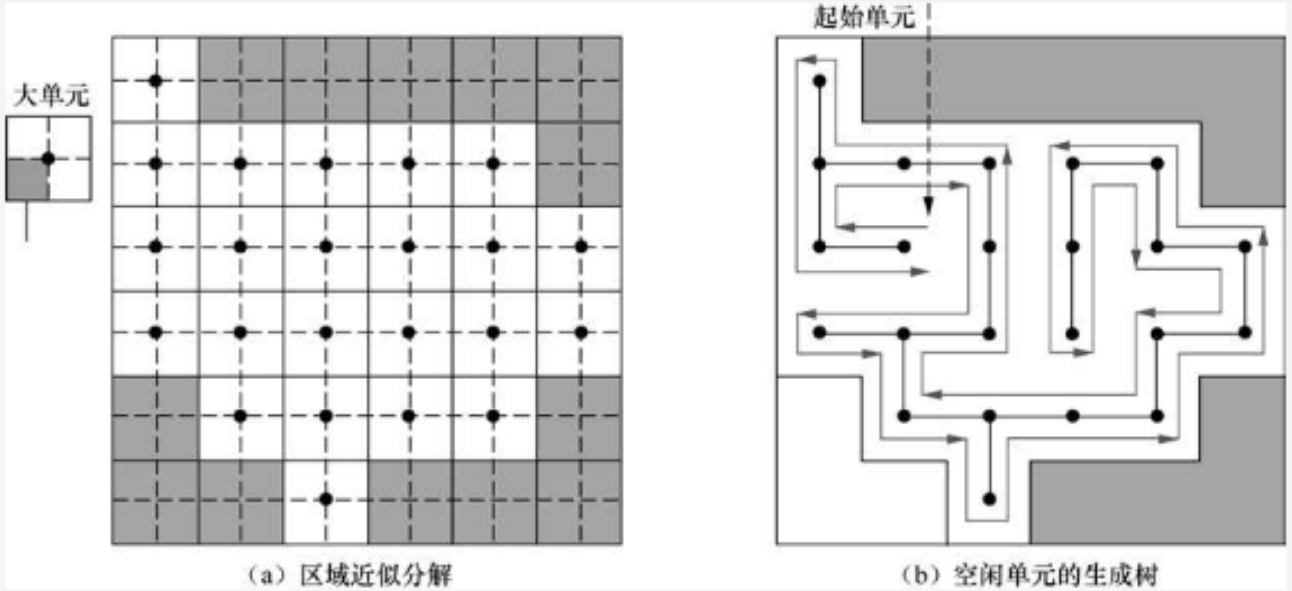


基于波阵图的覆盖路径

全覆盖路径规划

★ 栅格地图法--基于生成树算法的全覆盖路径规划

生成树覆盖算法将待覆盖区域分成一系列大小相同的单元，舍掉被障碍物占据的单元，再将其中的空闲单元递归地分解成4个大小相同的子单元，子单元的尺寸与机器人机身或者机器人上的传感器的探测范围相当，在此基础上构造连接所有空闲单元的生成树，通过对生成树的访问实现对整个区域的全覆盖，如图所示。



生成树覆盖算法

目前，生成树覆盖算法已经发展出离线、在线的形式。无论机器人是否掌握目标区域的先验知识，都有适合的方法应对。尽管生成树覆盖算法可以有效降低机器人覆盖过程中的重复率，但是会产生较多的转弯次数，降低机器人的工作效率。

全覆盖路径规划

★ 栅格地图法--基于神经网络的全覆盖路径规划

基于神经网络的全覆盖路径规划算法，是将神经网络模型与栅格地图路径规划相结合，其原理为：在栅格地图上，每个方格对角线的长度等于机器人的清扫半径，一个神经元对应一个栅格，每个神经元都与周围紧邻的8个神经元相连，如图所示。在路径规划开始时，将每个神经元的活性设为E，覆盖过程中，使用机器人上的传感器探测局部环境，在被探测到的环境中，将被障碍物占据的神经元的活性设为-E，被清扫过的神经元的活性设为0。每一时刻，机器人总是朝着离其最近的活性单元移动，直到所有的空闲区域被覆盖完。尽管该算法可以在不需要环境的先验信息的情况下完成对目标区域的覆盖，也可以很好地应对环境中有动态障碍物的情况，但是，该算法缺乏全局规划性，常常导致重复率较高。