1.先用Dijstra算法从目标节点G向起始节点搜索。储存路网中目标点到各个节点的最短路和该位置到目标点的实际值h,k（k为所有变化h之中最小的值,当前为k=h）。每个节点包含上一节点到目标点的最短路信息1(2),2(5),5(4)，4（7）。则1到4的最短路为1-2-5-4。

原OPEN和CLOSE中节点信息保存

2.机器人沿最短路开始移动，在移动的下一节点没有变化时，无需计算，利用上一步Dijstra计算出的最短路信息从出发点向后追述即可，当在Y点探测到下一节点X状态发生改变，如堵塞。机器人首先调整自己在当前位置Y到目标点G的实际值h(Y)，h(Y)=X到Y的新权值c(X,Y)+X的原实际值h(X).X为下一节点(到目标点方向Y->X->G），Y是当前点。k值取h值变化前后的最小。

3.用A\*或其它算法计算，这里假设用A\*算法,遍历Y的子节点，点放入CLOSE,调整Y的子节点a的h值，h(a)=h(Y)+Y到子节点a的权重C(Y,a),比较a点是否存在于OPEN和CLOSE中，方法如下：

上文中提及的A^\*A∗ 算法实际上只适用于已经给定了从起始点到终点范围的明确地图或规划空间的情况。然而在很多实际应用场景中，智能体并不完全清楚自己周边的环境信息，或者只拥有一张不完整的地图。那么如果我们基于起始状态的地图进行规划得到的路径大概率是不正确的或是次优的。因此要求智能体具备根据最新状态来实时更新地图以及重规划的能力。一个简单的方法是每次收集到新的信息后就根据更新的地图进行一次上文中的A^\*A∗。但是这样对计算资源的损耗与浪费无疑是极大的，尤其是当更新信息不会实际影响到当前求解出的最优路径时。所以更优的方法应该是基于上一个周期的规划路径结合新的更新信息做一定的修复改变。这类算法就叫做增量式的重规划算法，D^\*D∗(Dynamic A^\*A∗) 则是其中的代表之作。

（可以理解成，其实A\*算法也可以用来做动态规划，只不过A\*算法的特性决定了其在动态规划的时候需要重新运算（问题在于为什么A\*算法需要重新运算？不是有Path表么？）增大了运算量，）