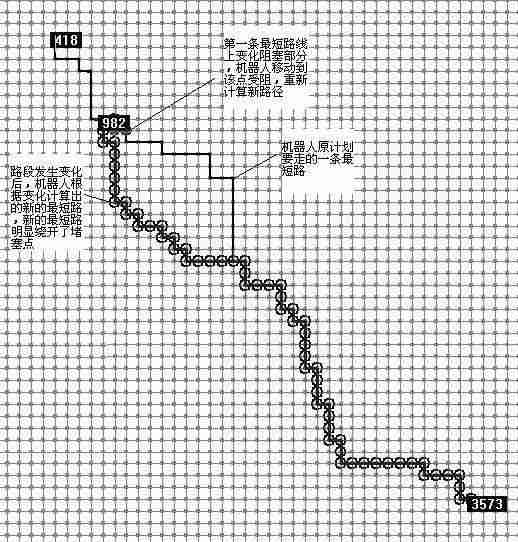
A\* 在静态路网中非常有效（very efficient for static worlds），但不适于在动态路网，环境如权重等不断变化的动态环境下。   
  
D\*是动态A\*（D-Star,Dynamic A Star） 卡内及梅隆机器人中心的Stentz在1994和1995年两篇文章提出，主要用于机器人探路。是火星探测器采用的寻路算法。

[Optimal and Efficient Path Planning for Partially-Known Environments](http://web.mit.edu/16.412j/www/html/papers/original_dstar_icra94.pdf)

[The Focussed D\* Algorithm for Real-Time Replanning](http://www.frc.ri.cmu.edu/projects/mars/publications/focussed_dstar_ijcai95.pdf)

主要方法（这些完全是Drew在读了上述资料和编制程序中的个人理解，不能保证完全正确，仅供参考）：  
  
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：  
  
while()  
{  
从OPEN表中取k值最小的节点Y;  
遍历Y的子节点a,计算a的h值 h(a)=h(Y)+Y到子节点a的权重C(Y,a)  
{  
    if(a in OPEN)     比较两个a的h值   
    if( a的h值小于OPEN表a的h值 )  
    {  
　　　　　更新OPEN表中a的h值;k值取最小的h值  
          有未受影响的最短路经存在  
          break;   
    }  
    if(a in CLOSE) 比较两个a的h值 //注意是同一个节点的两个不同路径的估价值  
    if( a的h值小于CLOSE表的h值 )  
    {  
　　　　　更新CLOSE表中a的h值; k值取最小的h值;将a节点放入OPEN表  
          有未受影响的最短路经存在  
          break;  
    }  
    if(a not in both)  
        将a插入OPEN表中;　//还没有排序  
}  
放Y到CLOSE表；  
OPEN表比较k值大小进行排序；  
}  
机器人利用第一步Dijstra计算出的最短路信息从a点到目标点的最短路经进行。

D\*算法在动态环境中寻路非常有效，向目标点移动中，只检查最短路径上下一节点或临近节点的变化情况，如机器人寻路等情况。对于距离远的最短路径上发生的变化，则感觉不太适用。



上图是Drew在4000个节点的随机路网上做的分析演示，细黑线为第一次计算出的最短路，红点部分为路径上发生变化的堵塞点，当机器人位于982点时，检测到前面发生路段堵塞，在该点重新根据新的信息计算路径，可以看到圆圈点为重新计算遍历过的点，仅仅计算了很少得点就找到了最短路，说明计算非常有效，迅速。绿线为计算出的绕开堵塞部分的新的最短路径。