A*算法改进算法及其应用①

Updated A* Algorithm and Its Application

张仁平[1]　周庆忠[1]　熊　伟[2]　王红旗[1]　(1.后勤工程学院 油料应用与管理工程系 重庆 400016;
2.沈阳军区空军后勤部 军需物资油料处 辽宁 沈阳 110015)

摘　要： 路径优化问题是现代生活和工作中的一个重要而复杂的问题，路径优化算法则是解决路径优化问题并推广应用路径优化问题的关键。在回顾 Dijkstra 算法和 A*算法的基础上，提出了 A*改进算法，并结合例子对算法求解过程进行说明。最后编程实现了 Dijkstra 算法、A*算法和 A*改进算法，并对运行结果进行比较分析。

关键词： Dijkstra 算法 A*算法 A*改进算法 路径优化

1　问题提出

在现代生活和工作中，路径优化问题往往是一个重要而复杂的问题。随着经济、生活水平提高，车辆越来越多，产生交通大量拥挤、阻塞，同时由于拥挤、阻塞容易导致交通事故上升、噪声和空气污染加大、时间延误增多、经济损失加大等，特别是大城市交通。最直接解决的办法就是修建更多道路、扩大路网规模。但是，一方面需要随着投入大量人力、物力和财力，且修建周期较长，另一方面，道路设施的增加速度较快，始终无法跟上机动车数量的增长速度且，现实情况使人们逐渐认识到：单纯依赖修建更多道路去试图满足日益增长的交通需求是不现实的，于是人们转变了思路，开始考虑采用高新技术特别是信息技术来改造升级现有道路运输系统，从而提高路网通行能力、服务质量以及交通安全水平。路径优化问题就成为解决道路拥挤、阻塞的重要问题，随着该技术的发展及应用范围推广，已应用于交通运输、市政规划、现代物流、作战指挥、后勤保障、最优选址、旅游观光等方面或领域，发挥着越来越重要的军事经济和社会效益。由于在不同部门、不同领域，路径优化中关心的问题可能不尽相同，甚至考虑的因素是多方面的。如在交通运输、现代物流、旅游观光等方面，人们可能更关心时间最短、出行距离最短、出行费用最省等因素；而在市政规划方面，规划者更关心各种公众社会保障设施设备的最优选址问题，以数量最少、分布最佳来满足和保障人们社会生活需要；在作战指挥方面，指挥员更关心道路最安全的问题，在道路方面有很多军事因素方面的考虑，如考虑各种道路的安全性，某一条或几条道路(可能被炸毁、冲毁)不能通行，或者由于某种军事需要，某一条或几条道路必需经过等情况的最优路径问题；在后勤保障方面，后勤人员除了关心道路安全性问题之外，也关心在现有交通道路情况下，如何实现众多保障资源与众多部队需求的最佳结合问题。因此，路径优化问题可以分为两个大问题来解决：一是众多因素的无量纲处理问题，即将主要因素进行统一处理或无量纲处理，转化为类似最短路问题；二是有效路径探索算法。关于第一个问题，可以引入层次分析法、模糊层次分析法或代价系数法等解决，笔者已进行相应研究，撰写了 2 篇相关论文；关于第二个问题，就是本文研究的主要内容。笔者之前进行了一些研究，如在《计算机系统应用》2007 年第四期发表了“A＊算法及其在地理信息系统中的应用”一文 1，经过 1 年多的教学和科研实践，认为 A＊算法是路径探索有效算法，在后勤保障优化方面发挥了重要作用，同时在应用中也对该算法进行改进，改进后的 A*算法探索路径结点减少，运算效率更高。

① 基金项目:后勤工程学院博士创新基金(HGB200605)
收稿时间:2008-12-30

2 常用路径探索算法回顾

在介绍 A*改进算法以前，有必要回顾一下常用路径探索算法，一是“温故而知新”，二是便于进行算法比较。

2.1 Dijkstra 算法

2.1.1 算法概述

1959 年，E.W.Dijkstra 提出了 Dijkstra 算法，它适合所有非负弧的最短路算法，是目前公认的求解最短路问题的最经典算法。它给出从某定点到图中其它所有顶点的最短路，其时间复杂度为 $O(n^2)$，n 为结点个数。

2.1.2 算法基本思想[1]

以始发点 U_0 为树根，按照距 U0 的最短距离以及结点的相邻关系，逐次放入树中，由近至远，直到所有结点(包括目标点)全部放入树中，最后形成最短路树，树上每一个结点与 U0 的路径都是最短路径。算法将路络结点分为未标记、临时标记和永久标记三种类型。网络中所有结点初始时全部为未标记结点，在搜索过程中若某个结点和最短路径结点相连通，则该结点为临时标记结点，每一次循环都是从临时标记结点中搜索距探索起点路径最短的结点作为永久标记结点，直至找到目标结点或者所有结点都成为永久标记结点才结束算法。

2.2 A*算法

2.2.1 算法概述

A*算法是目前最流行的启发式搜索算法(Heuristic Search)，该算法由 Hart、Nilsson、Raphael 等人首先提出，算法的创新之处在于探索下一个结点时引入了已知的路网信息，特别是目标点信息，增加当前结点有效评估，即增加约束条件，作为评价该结点处于最优路线上可能性的量度，因此首先搜索可能性较大的结点，从而减少探索结点个数，提高了算法效率。

2.2.2 算法思想[2]

A*算法是建立在典型的 Dijkstra 算法基础之上的启发式搜索算法，在 Dijkstra 算法基础之上引入了当前结点的估计函数 f*(i)，则当前结点的评价函数可定义为：

$$f^*(i)=g(i)+h^*(i)$$

式中 g(i)是从起点到当前结点的最短距离，在实际应用中常常用当前结点 i 之父结点 j 的距离 g(i)加上它们之间距离 D(i,j)来替代，因为直接获取当前结点与起始点最短距离往往有些困难，而 g(i)与 g(i)+D(i,j)差距不大，且 g(i)已获得，D(i,j)属于已知条件。h*(i)是从当前结点到终点最短距离的估计值，可取结点到终点的直线或球面距离。若 h*(i)=0，即没有利用任何路网信息，这时 A*算法就变成了 Dijkstra 算法，这说明经典的 Dijkstra 算法可看作 A*算法的特例，即 A*算法是 Dijkstra 算法的“改进算法”。对于 h*(i)的具体形式，除了当前结点到终点的最短距离的估计值之外，也可以是其它估计值，如方向，算法使用者可以依据实际情况选择。

3 A*改进算法

3.1 算法思想

A*改进算法利用 A*算法对当前结点进行评估的思想，增加了最短路径中当前结点的父结点，并对该结点与终点的距离进行评估。算法核心思想是：将当前临时标记结点到探索起点的最短距离、当前临时标记结点到目标结点距离的评估值和当前临时标记结点的父结点到目标结点距离的评估值三者之和作为评价该结点处于最优路线上可能性的量度，这种优化方法称为 A*改进算法，假设当前临时标记结点为 i，则其评估函数可定义为：

$$f^*(i)=g(i)+h^*(i)+h^*(j)$$

其中，g(i)与 h*(i)与 A*算法含义相同，g(i)是从起点到当前临时标记结点 i 的最短距离。同样，在实际应用中由于 g(i)暂时难以获得，常常用当前结点 i 之父结点 j 的最短距离 g(j)加上它们之间距离 D(i,j)来替代，即用 g(j)+D(i,j)替代 g(i)，因为 g(j)已获得，而 D(i,j)属于已知条件，h*(i)是从当前结点到终点最短距离的估计值，h*(j)是从当前结点的父结点 j 到终点最短距离的估计值。

经过改进后的估计函数在算法搜索时，使 A*算法的搜索方向更快趋向终点，极大地减少了算法中遍历的结点个数，从而提高了搜索速度。当然，如果需要可以增

加当前结点之父结点的父结点的父结点……，同样，将该结点与终点的距离进行评估。需要说明的是，并不是加入评估点越多，算法效率超高。虽然加入评估点越多，算法遍历的结点个数会越少，但是一会增加存储容量，二会增加评估值数量，从而增加算法负担。

3.2 算法步骤

算法步骤与原 A*算法相似，只是在评价函数增加了当前结点之父结点与目标点距离评估值。大家可参见《计算机系统应用》2007 年第四期发表了“A＊算法及其在地理信息系统中的应用”一文，在此就不重复了。

3.3 算法求解过程示例

例：在图 1 中，用 A*改进算法求从 U_0 到 U_7 最短距离及路径，其中假定 $U_i(i=0\cdots6)$到终点 U_7 的距离估值 h*(i)分别为 7、6、5、4、3、2、1。

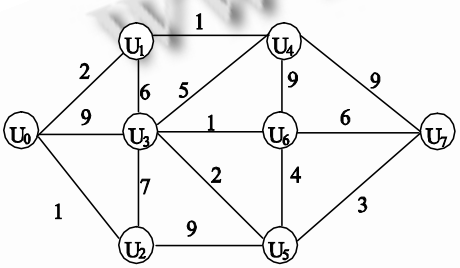

图 1　路径示例

求解之前，需要对一些参数进行说明。数组 P 表示已经被探索到但尚未进入最短路径的结点集合，数组 S 为已经进入最短路径的结点集合，算法执行完毕之后，S 即为已找到从 U_0 出发的最短路径的结点集合。f*值是当前结点的评价值，是当前结点 Ui 到起始点的最短距离 gui、U_i 到目标点距离的估计值 $h^*(U_i)$以及 U_i 之父结点 U_j 到目标点距离的估计值 $h^*(U_j)$三者之和；式中的 gui 为当前结点 U_i 到起始点的最短距离；$F(U_i)$表示当前结点 U_i 的父结点。$D(U_i, U_j)$表示 U_i 与 U_j 之间的距离。

第 1 步，将 U_0 的邻近节点{U_1 、U_2 、U_3}放入数组 P 中，则 P={U_1、U_2、U_3}，因为 $f^*(U_1)=2+6+7=15$，因为 $f^*(U_2)=1+5+7=13$，因为 $f^*(U_3)=9+4+7=20$，则估算值最小的节点是 U_2，放入数组 S 中。此时 S={U_0、U_2}，P={U_1、U_3}，$F(U_2)=U_0$

第 2 步，U_2 的邻近节点{U_3 、U_5}，对于 U_5 且，计算 $g_{u5}=1+9=10$，对于已经在 P 里的 U_3 ，由于 $g_{u3}>g_{u2}+D(U_2,U_3)$，(因为 $g_{u3}=9$，$g_{u2}=1$，$D(U_2,U_3)=7$)，则 g_{u3} 修改为 1+7=8，而不是原来的 9；将{U_3 、U_5}放入数组 P 中，则 P={U_1 、U_3 、U_5}。因为 $f^*(U_1)=2+6+7=15$，$f^*(U_3)=9+4+7=20$，$f^*(U_5)=10+2+5=17$，则估算值最小的节点是 U_1，放入数组 S 中。此时 S={U_0、U_2、U_1}，P={U_3、U_5}，$F(U_1)=U_0$

第 3 步，U_1 的邻近节点{U_4、U_3}，对于 U_4 且，则 $g_{u4}=g_{u1}+D(U_1,U_4)=2+1=3$，对于 U_3，由于 $g_{u3}=g_{u1}+D(U_0,U_1)$，则无须改变；将{U_4、U_3}放入数组 P 中，则 P={U_3 、U_4 、U_5}，因为 $f^*(U_3)=20$，$f^*(U_4)=12$，$f^*(U_5)=17$，则估算值最小的节点是 U_4，放入数组 S 中。此时 S={U_0、U_2、U_1、U_4}，P={U_3、U_5}，$F(U_4)=U_1$

第 4 步，U_4 的邻近节点{U_1 、U_3 、U_6、U_7}，对于已存在于 P 中的 U_3 和 S 中的 U_1，不满足算法描述中的修改条件，而无须修改，将 U_6、U_7 放入数组 P 中，则 P={U_3 、U_5、U_6、U_7}，因为 $f^*(U_3)=20$，$f^*(U_5)=17$，$f^*(U_6)=16$，$f^*(U_7)=15$，则估算值最小的节点是 U_7，放入数组 S 中。此时 S={U_0、U_2、U_1、U_4、U_6}，P={U_3、U_5、U_7}}，$F(U_7)=U_4$

因为 U_7 是目标点，则用回溯的方法找到其父节点 U_4，依次类推，最终找到起始点 U_0，从而得到从起始点到目标点的最短路径{U_0、U_1、U_4、U_7}

4　算法运行结果与分析

笔者用在 MapEngine2.5 和 Delphi7.0 C/S 实现了 Dijkstra 算法、A*算法和 A*改进算法，并在部队机动保障最短路优化分析功能中得到应用。确定部队始发点和目标点有两种方式：一是在地图中直接选择；二是输入始发点和目标点的经纬度。在确定部队机动的始发点和目标点后，系统能在地图上寻找到一条最短路，并给出最短路长度、沿途地名信息、并通过缓冲分析对沿途保障能力和动员能力进行评估。如果最短路中有某条公路不能通行时，注上标记，系统继续寻找去除该条路之后的另一条最短路。下面，列出部分 Dijkstra 算法、A*算法和 A*改进算法探索结点个数比较。

(下转第 107 页)

(上接第 100 页)

表 1　3 种算法比较表

起始点、目标点	Dijkstra 算法	A*算法	A*改进算法
重庆、北京	12419	6857	5836
重庆、成都	1505	640	491
重庆、沈阳	12616	8319	7029
北京、广州	16723	10142	8364
乌鲁木齐、上海	19029	11523	9263
拉萨、兰州	4478	1904	1653
兰州、北京	10149	4179	3626
重庆、兰州	7429	4280	3857
兰州、广州	19395	10790	8642

从表中可以看出，A*算法比 Dijkstra 算法的探索结点个数要少得多，探索效率提高从 34%到 57%。如果考虑起点、终点选择上的可能存在微小误差，再考虑不同道路结点个数及其位置不同，A*算法的效率比 Dijkstra 算法提高 40%左右。A*改进算法与 A*算法相比较，探索结点减少，算法提高效率从 10%到 23%。考虑误差因素和道路等地理信息不同因素，A*改进算法与 A*算法提高约 15%左右。

参考文献

1 熊伟,张仁平,等.A*算法及其在地理信息系统中的应用.计算机系统应用, 2007,16(4):14－17.

2 郭建科,张仁平,等.Dijktra 改进算法及其在地理信息系统中的应用.计算机系统应用, 2007,16(1):59－62.