Dijktra 改进算法及其在地理信息系统中的应用①

Updated Dijkstra Algorithm and Its Application to GIS

郭建科　张仁平　（解放军后勤工程学院　重庆 400016）
邹孙楷　张新建　（重庆警备区　重庆 400000）

摘要：最短路径问题是地理信息系统的关键问题，Dijkstra 改进算法是解决有附加条件的最短路问题的有效算法。本文在结合例子分析 Dijkstra 算法的基础上，编程实现了 Dijkstra 改进算法。最后对 Dijkstra 改进算法进行应用与分析。

关键词：Dijkstra 改进算法　地理信息系统　最短路径　算法

1　引言

在交通运输、市政规划、通讯、选址等诸多方面，往往会遇到网络最优化问题，其核心是路径问题，即求最短路径，包括含权路（指道路质量、类别等所含不同的权重系数，下同）的最短路径。在许多地理信息系统开发平台中，都提供了相应的函数或方法，比如说 MapEngin、Surpermap 等，但是，这些函数或方法都进行了封装，并且是解决不含权的最短路。如果要解决含权道路的最短路，或者在排除某一条或几条道路（可能被炸毁、冲毁）之后的最短路，就需要在最短路中增加一些附加条件，地理信息系统开发平台提供的函数或方法就显得无能为力了。因此，自己动手去实现最短路算法，并且在算法中就可以增加前面提到的附加条件，从而可以解决用户需求。特别重要的是，在以后根据不同的功能需求，仅对附加条件进行修改就 OK 了，其中的重要性和必要性我就不在赘述。

2　Dijkstra 算法介绍

在求两固定点之间的最短路径问题，Dijkstra 算法是核心的算法，其它多数改进算法都基于该算法基础之上。Dijkstra 算法实际上是一种标记作业法，每次在计算完当前最短路径后就产生一个永久标记。算法的基本思想、标记原理和算法步骤虽然在相关教材都有各自不同的表述，但由于教材的严谨性和专业性，表述有些抽象、晦涩，不便于理解，更不便于计算机编程。下面，结合例子，将 Dijkstra 算法的基本思想、标记原理和算法步骤描述如下：

Dijkstra 算法的基本思想：以始发点 U_0 为树根，按照距 U_0 的最短距离以及节点的相邻关系，逐次放入树中，由近至远，直到所有节点（包括目标点）全部放入树中，最后形成最短路树，树上每一个节点与 U_0 的路径都是最短路径。这个描述仍然有些抽象，下面我们看看算法的关键所在——节点标记和算法步骤。

标记原理及步骤：第一步，在 U_0 的邻近节点中找出一个最近的节点 U_1，记下它的最短距离和最短路径，作上标记，放入最短路树中；第二步，在 U_0 的邻近节点（与 U_0 直接相连通的节点，以下类推）和 U_1 的邻近节点中找出一个离 U_0 最近的顶点 U_2，记下它的最短距离和最短路径，作上标记，放入最短路树中；第三步，在 U_0、U_1、U_2 的所有邻近节点（不含已作标记的节点，如 U_1、U_2，下同）中，找出一个离 U_0 最近的顶点 U_3，记下它的最短距离和最短路径，作上标记，放入最短路树中；同样原理，找出并标记 U_k，放入最短路树中；第 K+1 步，在所有已经标记的顶点 U_i（i=1，2，…，k）的邻近节点（一般不会太多，大多节点已作标记）中，找出一个离 U0 最近的顶点 U_{k+1}，记下它的最短距离和最短路径，作上标记，放入最短路树中。

① ＊本课题得到成都军区项目资助

最后直到所有节点被标记。为了便于理解、操作，下面举个简单例子加以说明。

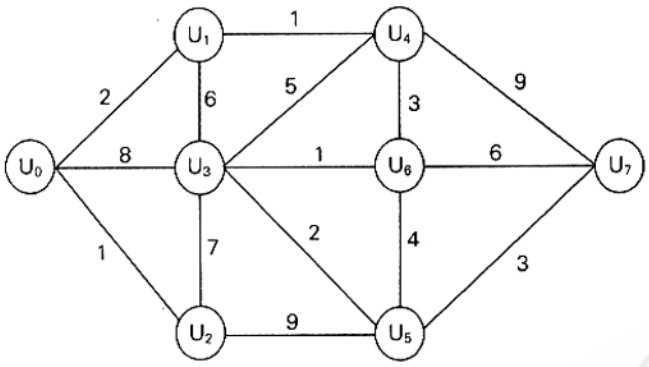

图 1

例：在图 1 中，用 Dijkstra 算法求从 U_0 到所有节点的最短距离及路径。

根据前面介绍的标记原理及步骤，计算如下。

第一步，在 U_0 的邻近节点（U_1、U_2、U_3）中找出一个最近的节点 U_2，并作标记，其最短距离：1，最短路径：$U_0 \rightarrow U_2$

第二步，在 U_0、U_2 未被标识的邻近节点（U_1、U_3、U_5）中找出距离 U_0 最近的节点 U_1，并作标记，其最短距离：2，最短路径：$U_0 \rightarrow U_1$

第三步，在 U_0、U_2、U_1 未被标识的邻近节点（U_3、U_4、U_5）中找出距离 U_0 最近的节点 U4，并作标记，其最短距离：3，最短路径：$U_0 \rightarrow U_1 \rightarrow U_4$

第四步，在 U_0、U_2、U_1、U_4 未被标识的邻近节点（U_3、U_5、U_6、U_7）中找出距离 U_0 最近的节点 U_6，并作标记，其最短距离：6，最短路径：$U_0 \rightarrow U_1 \rightarrow U_4 \rightarrow U_6$

第五步，在 U_0、U_2、U_1、U_4、U_6 未被标识的邻近节点（U_3、U_5、U_7）中找出距离 U_0 最近的节点 U_3，并作标记，其最短距离：7，最短路径：$U_0 \rightarrow U_1 \rightarrow U_4 \rightarrow U_6 \rightarrow U_3$

第六步，在 U_0、U_2、U_1、U_4、U_6、U_3 未被标识的邻近节点（U_5、U_7）中找出距离 U_0 最近的节点 U_5，并作标记，其最短距离：9，最短路径：$U_0 \rightarrow U_1 \rightarrow U_4 \rightarrow U_6 \rightarrow U_3 \rightarrow U_5$

第七步，在 U_0、U_2、U_1、U_4、U_6、U_3、U_5 未被标识的邻近节点（U_7）中找出距离 U_0 最近的节点 U_7，并作标记，其最短距离：12，最短路径：$U_0 \rightarrow U_1 \rightarrow U_4 \rightarrow U_6 \rightarrow U_3 \rightarrow U_5 \rightarrow U_7$ 或者 $U_0 \rightarrow U_1 \rightarrow U_4 \rightarrow U_6 \rightarrow U_7$ 或者 $U_0 \rightarrow U_1 \rightarrow U_4 \rightarrow U_7$。如果出现几条最短路径，在通常情况下，取第一条，即 $U_0 \rightarrow U_1 \rightarrow U_4 \rightarrow U_7$

由此，我们计算出 U_0 到所有节点的最短距离及路径，并可以画出其最短路树，树上每一个节点与 U_0 的路径都是最短路径，如图 2 所示，图中方框内的值是对应节点到 U_0 的最短距离。

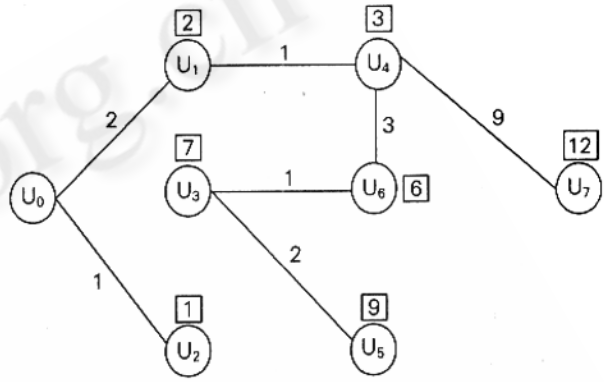

图 2

3 Dijkstra 改进算法及编程实现

在弄清 Dijkstra 算法的基本思想、标记原理和算法步骤后，编程实现就显得比较容易。在实际工程应用中，需要对 Dijkstra 算法进行改进并能自己编程实现，其主要原因有：第一，我们不需要计算始发点 U_0（树根）到所有节点的最短距离及路径，可能只关心到某一节点（即目标点）之间的最短距离及路径。因此，直接将目标点是否进行了永久标记，即是否进入最短路树作为运算停止的判断条件，这样大大减少了 Dijkstra 算法的标记步数，提高运算效率；第二，我们不需要把所有的道路节点都加入网络节点中进行计算，只是把当前节点的邻近节点，动态加入道路网络节点中参与计算。这样可以避免由于电子地图巨大的道路节点数而带来的超额运算量，从而大大降低算法的运算强度，提高系统运行效率；第三，根据道路质量、类型，可以对道路的长度乘以相应的权重，之后参与计算，另外，如果某一条或几条道路（可能被炸毁、冲毁），那么就需要将其道路的长度设为非常大的一个常数（即无穷大）或者去掉不能通行的公路（在邻近线中去除，不参与计算，见图 3）；第四，可以解决运

送费用的费用最优或部队机动时间最优等类似问题，因为对于不同类型的道路，如高速公路、省国道、县乡公路，相同里程的费用或时间是不同的，需要进行修正。

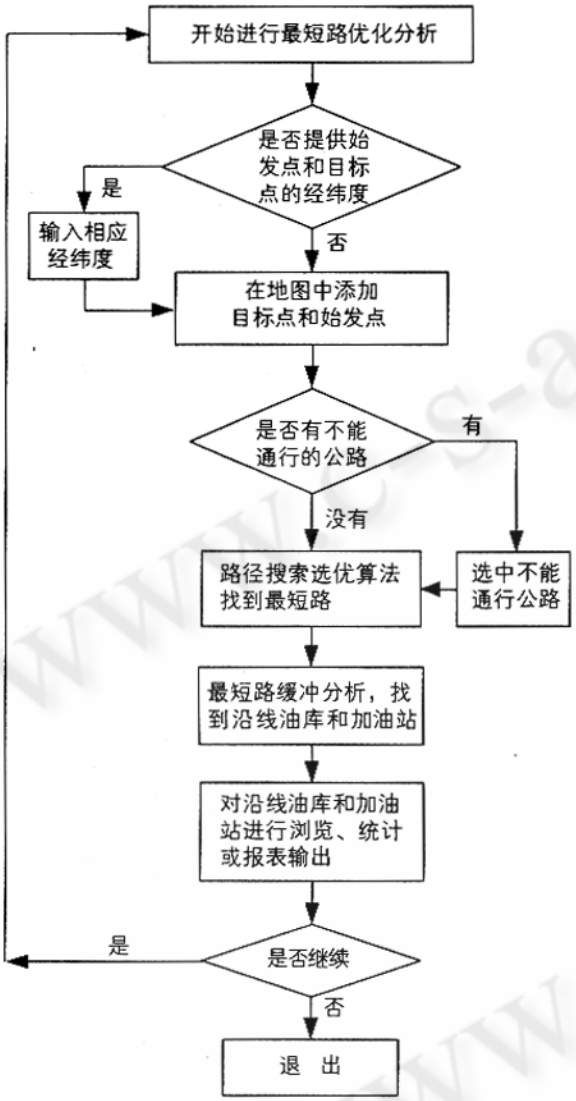

图 4

综上所述，Dijkstra 改进算法的基本思想：以始发点 U0 为树根，按照距 U0 的最短距离以及节点的相邻关系，逐次放入树中，由近至远，直到目标点放入树中，形成最短路树，树上每一个节点与 U0 的路径都是最短路径。其节点标记与 Dijkstra 算法完全相同，算法步骤的方法、原理也相同，只是算法的停止条件是目标点是否进入最短路树，而不是把所有节点是否都标记作为运算停止的判断条件，因此计算量则大大减少。另外，由于有时需要考虑时间最优，费用最优，最安全或者某些道路必须通行或者不能通行等有附加条件的最优路，而不仅仅是距离最短，因此在公路网格中，需要对道路赋于不同的权重，之后参与计算，这些在 Dijkstra 改进算法中，也要充分考虑。

笔者在最近开发的几个地理信息系统中，Dijkstra 改进算法都得到成功应用，解决了部队的实际问题。系统的 GIS 软件开发平台是优秀的国产软件 MapEngine，。系统的前台开发工具是非常优秀的面向对象开发工具 Delphi6.0 C/S 版。其程序执行流程见图 3（见下页），主要变量说明见表 1。

表 1　图 3 的主要变量说明

变量名称	变量类型	变 量 含 义
PathPoint	五维整数型动态数组（其长度由所有参与计算的节点数量来决定）	PathPoint[i,0]：当前点的 ID PathPoint[i,1]：当前点的父节点在 Pathpoint 数组中的位置 PathPoint[i,2]：当前点的标记，有标记表示该点是最优化路径中的节点，否则不是（0：无标记；1：有标记） PathPoint[i,3]：存放当前点与其父节点间的路径的 ID PathPoint[i,4]：存放当前点与循环起点间的路径的 ID
PathLength	二维浮点型动态数组	PathLength[i,0]：当前点到计算起点之里程 PathLength[i,1]：当前点到父节点之里程

4　Dijkstra 改进算法运行结果与分析

笔者用在 MapEngine2.5 版和 Delphi6.0 C/S 实现了 Dijkstra 改进算法，并在部队机动最短路优化分析功能中得到应用。确定始发点和目标点有两种方式：一是在地图中选择；二是输入相应的经纬度。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，并给出最短路长度等信息。如果最短路中有某条公路不能通行时，注上标记，系统继

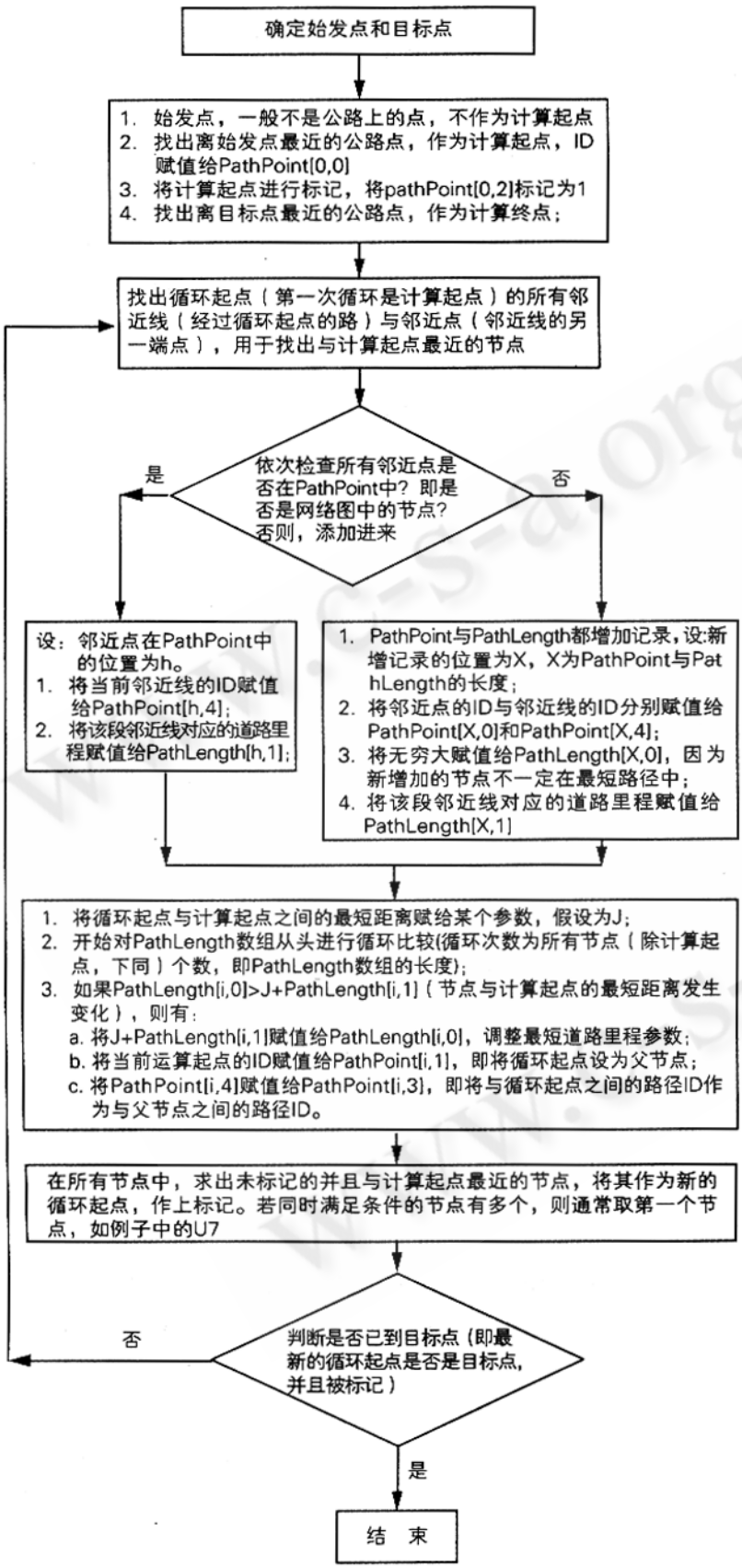

图 3

续寻找去除该条路之后的另一条最短路。如图 4 所示(见上页)。

如果考虑县乡公路，计算北京到重庆的最短路径，只须 5 秒钟，计算成都到重庆的最短路径，只须不到 1 秒钟；如果不考虑县乡公路，只考虑高速公路和省国道，北京到重庆大约都在 1 秒钟，成都到重庆则几乎没有延迟。系统运行稳定、效率高。另外，有时由于电子地图拓扑关系的原因，计算起点与计算终点不能通行，容易出现死循环，可以在获得新的循环起点之后加上判断，看看循环次数是否大于所有满足条件的公路条数，如例子中是否大于 15，如果大于，则退出循环，给出“不能通行”等相应提示信息。

参考文献

1 龚劬，图论与网络最优化算法，重庆大学出版社，1998.

2 Steve Teixeira & Xavier Pacheco，Delphi5 开发大全，人民邮电出版社，1999 年 8 月.

3 Steve Teixeira & Xavier Pacheco，Delphi5 开发人员指南，机械工业出版社，2001 年 10 月.