DOI:10.13374/j.issn1001-053x.2007.03.022
第29卷 第3期 北京科技大学学报 Vol.29 No.3
2007年 3月 Journal of University of Science and Technology Beijing Mar. 2007

一种基于Dijkstra算法的启发式最优路径搜索算法

王景存[1,2)] 张晓彤[1)] 陈 彬[2)] 陈和平[2)]

1) 北京科技大学信息工程学院，北京 100083 2) 武汉科技大学信息科学与工程学院，武汉 430081

摘 要 为了建立一个高效的路径搜索引擎，针对大型应用系统中寻径算法的平衡最优性、时间复杂度以及空间复杂度问题，从经典Dijkstra算法出发，将AI领域的决策机制引入到路径搜索中来，提出了一个启发式最优路径搜索算法．该算法在寻径过程中引入代价函数，由代价函数来决定寻径策略(即优先搜索哪些中间节点)，以期望减少搜索节点数．给出了该算法得到最佳解的条件及其证明过程，并且以实例数据对两种算法进行了对比测试．

关键词 路径搜索；导航；启发式；最优

分类号 TP301.6

路径搜索是地理信息系统、导航系统以及计算机网络等高级系统中的一个典型应用，通常需要考虑以下几个方面：第一，网络规模庞大，节点数量较多；第二，路径搜索频繁；第三，需要近乎实时的相应速度．如果系统是应用在移动设备上，还需要考虑设备的硬件资源(运算、存储等)有限等因素．

在众多的路径搜索算法中，Dijkstra算法提供了从图的一个节点到另一个节点的最短路径．经过一次Dijkstra算法计算，可以得到从起点到图内被其搜索到的所有节点的最短路径，其时间复杂度为$o(n^2)$(其中n为图的节点数)．但是在实际应用中，所关心的是某两个特定节点之间的最短路径而非起点到其他点的情况，且在大规模网络中Dijkstra算法的时间复杂度较高，使其应用受到限制．文献[1]提出了利用堆数据结构来改善Dijkstra算法的寻径速度，但对Dijkstra算法的时间复杂度性能提高有限．文献[2]提出了预先计算图中每对节点之间的最短路径并存储在数据库中，在实际应用时，根据起点和终点直接在数据库中查询的方法．该方法的时间复杂度得到了很大程度的降低，但是额外存储空间开支较大．文献[3]根据文献[2]的思路，将数据库视图进行分层、编码以便于查询且减少存储空间耗费．这种方法只适合静态图，如果图的拓扑结构发生变化，则必须更新数据库和编码；而且对于移动设备来说，这种额外的存储空间开支使得本就稀缺的存储资源显得更加捉襟见肘．文献[4]结合地理信息系统的特点，将起点和终点的地理信息考虑到寻径算法的计算中，在寻找两点之间路径的最优而非最佳解时有较佳的表现；但为了尽量逼近最佳解，必须对图搜索两次．

为了解决大型应用系统中寻径问题，且使算法能较好地平衡最优性、时间复杂度以及空间复杂度，本文提出了一种启发式最优路径搜索算法(heuristic optimization pathfinding algorithm，HOPA)，即在寻径过程中引入代价函数，由代价函数来决定寻径策略(即优先搜索哪些中间节点)，以期望减少搜索节点数，从而降低其时间复杂度．

1 HOPA算法

1.1 启发式路径搜索(heuristic pathfinding，HP)

定义1 寻径算法可应用的网络可以用定义来$G=(V,E)$描述，其中$V=\{v\mid v_i=\langle x_i,y_i\rangle\}$，$E=\{\mathrm{VR}\}$，$\mathrm{VR}=\{\langle u,v\rangle\mid P(u,v)\wedge(u,v\in V)\}$．$V$为网络中的节点的有穷非空集合，$\langle x_i,y_i\rangle$为节点$v_i$的地理坐标，VR为网络中两个顶点之间的关系集合，$P(u,v)$为从顶点$u$到顶点$v$一条边的权．

Dijkstra算法首先建立了一个集合$V'=\{v\mid v_i\in V\}$，该集合初始化时只有一个元素，即路径的起始节点v_s，$\overline{V}'$为V'的补集．其计算步骤为：在$\overline{V}'$中找到所有与V'直接相连的节点，然后将这些节点中与V'之间具有最小权值的节点添加到V'中，如此循环直到将终点v_e添加到V'中．Dijkstra算法实际上是宽度优先的搜索算法，它对所有与V'直接相连节点采取同等对待策略，因此导致了Dijkstra算法为了找到最短路径需要搜索较多的节点，时间复杂度也就相应地较高，当然这种平等对待的策略也是该算法取得最佳解的保证．

收稿日期：2006-04-05 **修回日期**：2006-10-20

基金项目：中国科学院计算所知识创新工程"HPC—OG模拟系统及相关技术"研究项目(No.20036040)

作者简介：王景存(1963—)，男，副教授，博士研究生

在 Dijkstra 算法中引入代价函数来描述新的策略机制，即：在搜索前，先判断从哪些节点查找下去最有可能找到最佳路径.

定义 2　设 $f(v_j)=g(v_s,v_j)+h(v_j,v_e)$，其中 $g(v_s,v_j)$为从起始节点 v_s 经过寻径算法到达 v_j 的权值之和，且 $g(v_s,v_s)=0$，则称 $h(v_j,v_e)$为节点 v_j 的代价函数，$h(v_e,v_e)=0$，它预测从 v_j 到终点v_e 的代价，$f(v_j)$的值决定该点是否能被优先查找.

引入代价函数，就可以将均等式的宽度优先搜索机制改进成有方向性的深度优先搜索机制，即启发式路径搜索(HP)，描述如下：

集合 $V^*=\{v^*\mid v_i^*\in V,\forall(i\neq j)v_i^*\neq v_j^*\}$，$\overline{V}^*$为 V^*的补集，V^*初始状态只有一个起始节点 v_s；$V_c^*=\{v_c^*\mid v_{ci}^*\in V,\forall(i\neq j)v_{ci}^*\neq v_{cj}^*\}$表示 $\overline{V}^*$中所有与 V^*直接连接的节点集合. 算法的步骤大体为：首先在 V_c^* 中查找 v_{cj}^*，使

$$f(v_{cj}^*)=\min_i\{f(v_{ci}^*),v_{ci}^*\in V_c^*\},$$

然后将 v_{cj}^*从 V_c^* 中删除并添加到 V^*中，最后再在集合 $\overline{V}^*+\overline{V_c}^*$ 中查找与 v_{cj}^*直接相连的节点，并添加到 V_c^* 中去，重复以上步骤直到将终点 v_e 被添加到集合 V_c^* 中.

定理 1　设有序序列 $V'=\{v_s,v_1,v_2,\cdots,v_j,\cdots,v_n,v_e\}$为通过 HP 找到的从起点 v_s 到终点 v_e 的路径，$\mathrm{dist}_{\min}(v_k,v_l)$表示 v_k 到 v_l 最小权值之和，若满足

$$(\forall v_j)((h(v_j,v_e)\leqslant\mathrm{dist}_{\min}(v_j,v_e))\wedge(v_j\in V')),$$

则 V'为最短路径.

证明　设 $V''=\{v_s,v_1',v_2',\cdots,v_{i-1}'\}$为最短路径的前 i 个节点组成的集合，v_i' 为该最短路径的下一个节点. 从 v_{i-1}'到终点 v_e 有 m 条路径，其中一条路径为

$$P_q=\{v_{i-1}',v_{q1}',v_{q2}',\cdots,v_{qr}',v_e\},q\leqslant m;$$

令 $v_{p1}'\neq v_i'$，P_q 满足

$$\forall v_{pi}'(f(v_{pi}')<f(v_i')\wedge(v_{pi}'\in V'')),$$

则按照 HP 算法，必然会优先按照 P_q 的方向按深度搜索下去. 当算法搜索到终点 v_e 时，

$$f_p(v_e)=g_p(v_e,v_s)+h_p(v_e,v_e)=g_p(v_e,v_s),$$

引入符号 $d(v_k,v_l)$表示边 $\langle v_k,v_l\rangle$ 的权，则

$$g(v_e,v_s)=d(v_s,v_1')+\sum_{g=1}^{i-2}d(v_g',v_{g+1}')+d(v_{i-1}',v_{q1}')+\sum_{h}^{r-1}d(v_{qh}',v_{q(h+1)}')+d(v_{qr}',v_e).$$

而

$$f(v_i')=d(v_s,v_1')+\sum_{g=1}^{i-2}d(v_g',v_{g+1}')+d(v_{i-1}',v_i')+\mathrm{dist}_{\min}(v_i',v_e),$$

又因为 v_i' 为最短路径的下一个节点，则有

$$d(v_{i-1}',v_i')+\mathrm{dist}_{\min}(v_i',v_e)<d(v_{i-1}',v_{q1}')+\sum_{h}^{r-1}d(v_{qh}',v_{q(h+1)}')+d(v_{qr}',v_e),$$

即

$$f(v_i')<f_p(v_e).$$

即表明 HP 算法此时会放弃路径 P_q，并最终会将 v_i'添加到 V''中去. 证明完毕.

不难看出，当 $f(v_{ci}^*)=g(v_s,v_{ci}^*)$，即 $h(v_{ci}^*,v_e)=0$ 时，HP 算法即为 Dijkstra 算法. 当 $h(v_{ci}^*,v_e)\neq0$ 时，$h(v_{ci}^*,v_s)$越小，在决策中 $g(v_{ci}^*,v_s)$所起的作用越大；相反 $h(v_{ci}^*,v_s)$所起的作用越大. $h(v_{ci}^*,v_s)$的引入使得寻径算法带有方向性，提高了寻径效率. 但 $h(v_{ci}^*,v_s)$只是估计值，不一定能保证找到的路径就是最佳的；但可以选择恰当的代价函数，使找到的路径尽量逼近最佳解，同时又能降低整个算法的复杂度.

1.2　代价函数选定

代价函数用以估计节点到终点的代价. 代价函数的值越大，搜索所花费的时间越少，算法的解也就越偏离最佳解；代价函数的值越小，算法的解也就越逼近最优解，同时时间耗费也就越大. 可见选取好的代价函数是问题解决的关键. 本文以城市交通网为例，说明选取代价函数的方法.

在 GIS 系统中，城市道路构成的网络具有的最大特点是：节点间具有几何特性，两个相互连接的节点之间的权值几乎可以用两点间的几何距离来表征. 另外，还需注意到城市交通网的规划都比较规整，道路分布多近似于垂直和水平分布.

鉴于上述的网络结构，可选取代价函数：

$$h(v_i,v_s)=D\sqrt{(v_{ix}-v_{ex})^2+(v_{iy}-v_{ey})^2}\quad(1)$$

式中，v_{ix}和 v_{iy}分别为节点 v_i 的横坐标和纵坐标，v_{ex}和 v_ey 分别为终点 v_e 的横坐标和纵坐标，D 为放缩系数且 $D\geqslant1$. 由定理 1 可知：当 $D=1$ 时，能确保在城市交通网络中找到最短路径.

为了降低算法的复杂度也可以将式(1)近似为：

$$h(v_i,v_e)=\sqrt{2}D\max\{(|v_{ix}-v_{ex}|),(|v_{iy}-v_{ey}|)\}\quad(2)$$

如此将不能保证得到的解一定为最佳解.

图 1 形象地展现了一个典型的城市交通网的网络结构. 从 v_i 到 v_e 的最佳路径最有可能是沿着箭

头方向行走，将代价函数设为：

$$h(v_i, v_e) = D(|v_{ix} - v_{ex}| + |v_{iy} - v_{ey}|) \quad (3)$$

又 $|v_{ix} - v_{ex}| + |v_{iy} - v_{ey}| > \sqrt{(v_{ix} - v_{ex})^2 + (v_{iy} - v_{ey})^2}$，所以亦不能保证解的最佳.

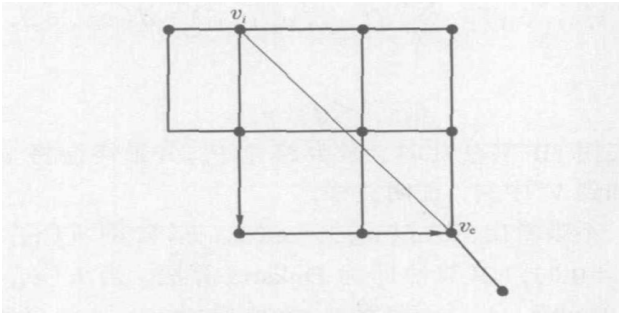

图 1 城市交通网示意图

Fig. 1 Conventional diagram of urban traffic network

代价函数也可以根据具体情况设置成其他形式，但只要能保证解的最优，就能将 HP 算法优化成 HOPA 算法.

1.3 HOPA 算法改进

对 HOPA 算法的分析可知，HOPA 算法将均等的宽度搜索改进成有策略的深度搜索，可有效地减少被搜索的节点数(当然，在最坏情况下，还是有遍历图的所有节点的可能)，从而节省搜索时间. 但是如果网络的规模庞大，在 V_c^* 中找到能满足 $f(v_{cj}^*) = \min_i\{f(v_{ci}^*), v_{ci}^* \in V_c^*\}$ 的节点的时间耗费也是不容忽视的. 为了缩短 HOPA 算法的搜索时间，本文将二叉堆[6]引入到对集合 V_c^* 的查找过程中.

定义 3 对于集合 $K = \{k_1, k_2, \cdots, k_n\}$，若其所有元素满足

$$\begin{cases} k_i \leqslant k_{2i} \\ k_i \leqslant k_{2i+1} \end{cases}, i = 1, 2, \cdots, \left\lfloor \frac{n}{2} \right\rfloor,$$

则称该集合为二叉堆.

如果集合

$$F = \{f | f_i = f(v_{ci}^*, v_e) \wedge \forall v_{ci}^* (v_{ci}^* \in V_c^*)\}$$

满足定义 3，则

$$f(v_{c1}^*) = \min_i\{f(v_{ci}^*), v_{ci}^* \in V_c^*\}$$

成立. 此时查找最小 f 时不用在 V_c^* 中查找，而只需将其第一个元素取出即可.

要保证一个集合满足二叉堆，主要体现在对这个集合添加元素和删除元素的操作上. 设集合 $K = \{k_1, k_2, \cdots, k_n\}$ 是一个二叉堆，现有一元素 k 需要插入到 K 中，其具体步骤为：首先将 k 插入到 K 的尾端，即 k 在 K 中的下标为 $n+1$；然后比较 k 与 $k_{\lfloor (n+1)/2 \rfloor}$，如果 k 较小，则交换这两个元素，此时 k 的新下标为 $i = \lfloor (n+1)/2 \rfloor$；接下来再比较 k 与 $k_{\lfloor i/2 \rfloor}$. 如此循环直到比较过程中 k 较大，或 k 已经移到 K 的首部. 如果要将 k_1 从 K 中删除，其具体步骤为：首先将 k_1 从 K 中删除，然后将 k_n 移到 K 的首部，则 k_n 的新下标为 1；接下来比较 k_n 与 $k_{2\times1}, k_{2\times1+1}$，如果 $k_n < \min(k_{2\times1}, k_{2\times1+1})$ 则交换 $k_n, \min(k_{2\times1}, k_{2\times1+1})$，设 k_n 的新下标为 i；如果此时 $k_n < \min(k_{2\times i}, k_{2\times i+1})$ 则交换 $k_n, \min(k_{2\times i}, k_{2\times i+1})$. 如此循环直到比较过程中 k_n 为三者中最大，或 k_n 无元素可比较(即 $2i > n-1$).

通过对堆的插入和删除过程的分析可知：在一个二叉堆序列插入一个元素或删除第一个元素需比较的次数 $\leqslant \lfloor \mathrm{lb}\, n \rfloor$ 次，当 n 足够大时，采用二叉堆比其他数据结构能大大加快查找速度；但在元素个数较小的情况下效果不太明显. 对于网络规模较大，节点数量较多的场合(节点数 $> 10^3$)，使用二叉堆对加快寻径算法的速度有举足轻重的作用.

2 实验验证

2.1 搜索区域比较

建立一个理想网络，即图中所有节点在二维平面内均匀分布，图中的任意一条边的权值相同，所有的边都按水平方向或垂直方向分布，来模拟一个城市交通网. 虽然这种棋盘分布的理想网络并不能完全客观地反映实际情况，但可直观地、简略地说明 HOPA 算法(代价函数取式(1)，(2)或(3)结果大致相同，只将选取式(3)的情况列举了出来)与 Dijkstra 算法相比其降低被搜索节点数的程度，以及这两种算法的搜索空间分布(图 2 中实心点表示被搜索过的节点，粗线表示最终路径).

通过图 2 可以看出，Dijkstra 算法的均等宽度优先的搜索策略导致其搜索区域是以起点为中心，呈放射状蔓延到终点，而 HOPA 算法的搜索区域基本上分布在最终路径的两侧，从而使搜索的节点数大幅度减少(在实际情况下，HOPA 算法的效果虽不会如此显著，但还是在很大程度上减少搜索的节点数). 虽然两种算法得到的解不同，但 HOPA 算法得到的路径的权值和与 Dijkstra 算法相同，也就是说 HOPA 算法的解也是最佳解.

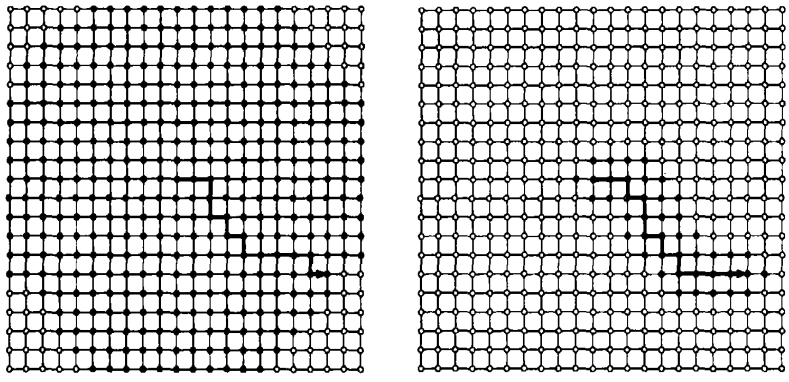

图 2　Dijkstra 算法(a)和 HOPA 算法(b)搜索区域分布

Fig. 2　Searching area distribution of Dijkstra (a) and HOPA algorithm (b)

2.2　实际情况实验数据对比

采用某城市 5 km×5 km 的一块区域内的城市交通网作为验证网络，其网络规模为：2 008 个节点，3 167 条边；实验平台：P4 2.6G，1G DDR. 表 1 中的每组数据是随机抽取 2 008 对点进行运算后得出的.

表 1　实验数据

Table 1　Test data

算法	代价函数	使用二叉堆	放缩系数	总耗费时间/ms	平均每次寻径耗费时间/ms	与最短路径切合度
HOPA	式(3)	否	1	13 532	6.74	94.28%
			2	12 703	6.33	78.70%
			3	12 375	6.16	40.13%
		是	1	9 969	4.96	94.28%
			2	9 609	4.79	78.70%
			3	9 228	4.60	40.13%
	式(1)	否	1	14 359	7.15	100%
			2	13 063	6.51	89.68%
			3	12 735	6.34	81.17%
		是	1	11 312	5.63	100%
			2	10 516	5.24	89.68%
			3	10 250	5.10	81.17%
Dijkstra	—	—	—	68 921	34.32	100%

由表 1 可以看出：Dijkstra 算法每次寻径耗费的时间为 35 ms 左右；而 HOPA 算法在能确保所有解为最佳的情况下，耗费的时间也不超过 8 ms，效率提高近 4 倍多. 随着 D 值的增加，时间耗费能进一步减少，但不能保证所有解都为最佳，其中代价函数采取式(2)其准确率随 D 值增加降低较快. 通过表 1 亦可看出，使用二叉堆能进一步提高算法效率.

3　结论

本文提出的 HOPA 算法，其主要思想是借助 AI 的决策机制，将每个节点到达终点的代价估计引入到算法中来，将传统的均等式宽度优先搜索机制改进成带有方向性的深度优先搜索机制，以减少搜索的节点数. 为了进一步减少搜索时间，在搜索过程中采用二叉堆这一数据结构. 随后的实验证明了本文中的定义和结论的正确性，以及该算法在实际应用中的优越性.

参　考　文　献

[1] 唐文武，施晓东，朱大奎. GIS 中使用改进的 Dijkstra 算法实现最短路径的计算. **中国图像图形学报**，2000，5A (12)：1019

[2] Agrawal R，Jagadish H V. Materialization and incremental update of path information // Proceedings of the 5th International Conference on Data Engineering. Los Angeles，1989：374

[3] 吴京，景宁，陈宏盛. 最佳路径的层次编码及查询算法. **计算机学报**，2000，23(2)：184

[4] 严寒冰，刘迎春. 基于 GIS 的城市道路网最短路径算法探讨. **计算机学报**，2000，23(2)：210

[5] Aleksandrow J，Matheshwari A，Sack J R. Approximation algorithms for geometric shortest path problems // Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing 2000. Portland，2000：286

[6] Elkin M. Computing almost shortest paths // Annual ACM Symposium on Principles of Distributed Computing. Newport，2001：53

[7] Jacob R，Marathe M，Nagel K. A computational study of routing algorithms for realistic transportation networks // 2nd Workshop on Algorithm Engineering. Saarbrucken，1998：168

[8] Imielinski T，Julio C N. GPS-based Geographic Addressing，Routing and Resource Discovery. New York：ACM Press，1999

[9] 毕军，付梦印，周培德，等. 基于城市道路网的快速路径寻优算法. **计算机工程**，2002，28(12)：36

[10] 谭国真，高文. 时间依赖的网络中最小时间路径算法. **计算机学报**，2002，25(2)：165

A heuristic optimization path-finding algorithm based on Dijkstra algorithm

WANG Jingcun[1) 2)], *ZHANG Xiaotong*[1)], *CHEN Bin*[2)], *CHEN Heping*[2)]

1) Information Engineering School, University of Science and Technology Beijing, Beijing 100083, China

2) Information Engineering School, Wuhan University of Science and Technology, Wuhan 430081, China

ABSTRACT To make an efficient path-finding engine, a heuristic optimization path-finding algorithm was proposed for resolving the time and space complexity problems of a searching algorithm in a large application system. The algorithm was based on the classical Dijkstra algorithm and introduced the decision mechanism in AI into path-finding. To decrease the number of nodes to search, cost-function was incorporated into this algorithm and used to decide the path-finding policy, that was, which nodes were searched firstly. The condition of getting the optimal solution from this algorithm was put forward and proved. These two algorithms were tested comparatively.

KEY WORDS path-finding; navigation; heuristic; optimization

(上接第 324 页)

Basic study of modern blast furnace using natural lump ores rationally

WU Shengli[1)], *XU Haifa*[1)], *WANG Guojun*[2)], *LI Zhaoyi*[2)], *TIAN Yunqin*[1)], *CHEN Hui*[1)]

1) Metallurgical and Ecological Engineering School, University of Science and Technology Beijing, Beijing 100083, China

2) Iron-making Plant, Shanghai Baosteel Co. (Group), Shanghai 200941, China

ABSTRACT The fact that the metallurgical properties of natural lump ores were not insufficiently comprehensive and thorough grasped is obstructing the technology development of using lump ores in a modern blast furnace. Through analyzing and inspecting each metallurgical property of natural lump ores from main habitats in the world by experiment, the article indicates that the lump ores' metallurgical properties can meet a modern blast-furnace except the softening and melting properties. And the interaction between the sinter and the lump ores was found in the high temperature zone of the blast furnace, which can obviously improve the softening and melting properties of the lump ores. The blast burden structure also can be optimized by the interaction.

KEY WORDS blast furnace; natural lump ore; metallurgical property; high-temperature interaction