

地标导向的启发式路径规划算法

孟珂*, 张春艳

(中国矿业大学 计算机科学与技术学院, 江苏 徐州 221008)

(*通信作者电子邮箱 kekee000@126.com)

摘要:为提高大规模交通网络路径规划算法的查询效率,以A*算法为基础,提出一种地标导向的启发式算法。在预处理中将重要的顶点和边选为地标,在点对点寻径时使用地标作为启发式函数的启发参数,并进行分段计算。实验结果表明,此算法在处理长距离的路径规划问题时有较高的查询效率和更合理的计算结果。

关键词:路径规划;地标;预处理;层次缩减算法;三角启发算法

中图分类号: TP312.8 **文献标志码:** A

Landmark-oriented heuristic routing algorithm in traffic network

MENG Ke*, ZHANG Chun-yan

(School of Computer Science and Technology, China University of Mining and Technology, Xuzhou Jiangsu 221008, China)

Abstract: To improve the query efficiency of road routing algorithm in large-scale traffic network, a landmark-oriented algorithm based on A* algorithm was proposed. Select the most important vertexes and edges as landmarks during preprocessing, choose appropriate landmarks as the reference parameters and calculate in sections in point-to-point routing. The experimental results indicate that it has higher query efficiency and more reasonable results in long-distance road routing.

Key words: path-planning; landmark; preprocessing; Contraction Hierarchies (CH) algorithm; A* Landmarks Triangle (ALT) algorithm

0 引言

对于大规模交通网络,Dijkstra算法^[1]需要花费长时间进行计算,不符合实时性的要求。目前相关的优化算法有启发式算法和预处理算法两种。启发式算法(A*)^[2]使用合适的启发函数减少搜索空间以获得较高的查询效率,启发函数会直接影响最后的计算结果;预处理算法使用点或边标记法、快捷路径法、区域分割法等对网络中的边进行合并和标记以迅速求出最短路径,但需要大量的辅助存储空间。

根据实际交通网络的特点:在主干道上通过的最短路径最多,存在重要的边和点;对于长距离的路径规划,出发点和目标点的中间节点有可能成为最短路径上的点。本文以A*算法为基础,将地图中关键的节点选为地标,并将地标作为启发函数的启发参数来求得路径规划的合理解。为提高长距离路径规划的查询效率,使用分段处理的思想将查询分割为若干子查询,并给出相关的优化方法。

1 相关研究

对于静态网络图 $G=(V,E)$,大多数优化算法需要经过充分的预处理。三角启发算法(A* Landmarks Triangle, ATL)^[3]算法将图 G 按中心点划分为若干区域,每个区域选取一个标志点(LandMark),根据三角不等式使搜索路径趋向于目标节点,大幅度减少搜索空间,从而提高查询效率。

文献[4]提出一种分层合并的预处理算法CH,对原始图 G 的边进行迭代合并,产生一组生成图 $\{G^1, G^2, \dots, G^h\}$,生成图和原始图的边使用标号对应,以便求解后还原原始路径。这种预处理算法非常消耗存储空间,不适合于大规模网络,但是

可以快速求出最短路径。经过预处理的CH算法时间复杂度可以达到 $O(N \log H)$, N 为合并图的平均边数, H 为合并图的层数。

Arc-Flags^[5]基于区域划分的思想对图进行预处理。将图划分为 K 个区域,每一条边 (v,w) 存储一个 K 比特的参数,第 i 位代表从点 v 到区域 i 的最短路径中包含此边。ARC-Flags可以精确求出最短路径,但预处理时间较长。Chase算法^[6]综合CH和ARC-Flags的特点,对ARC-Flags划分的区域使用CH算法进行合并处理,加快预处理时间。

Bauer等^[7]提出一种混合算法SHARC,对CH和ARC-Flags进行了多项改进,提出分层标记的思想,可以缩短预处理时间和减少额外空间。分层的ARC-Flags提供搜索方向,CH加速区域内的路径搜索,在单向搜索环境中SHARC可以提供非常高效的精确最短路径。经过修改的SHARC可以进行时变最短路径问题的搜索,文献[8]对此有详细描述。

2 地标导向的启发式算法

2.1 地标的选取

交通网络图一般拥有层次关系,乡镇与城市之间有干道相连,城市与城市之间有高速相连,在路径规划中这些连接线路被通过的次数最多。对于具有这一特点的图 G ,地标集的定义如下:

设 r 为一个搜索半径,点 u 为中心, $2r$ 为半径的 G 的子图记为 $B_{u,2r}$,选取满足以下条件的最短路径 $P, P \subseteq B_{u,2r}$ 并且 $Len(P) > r(Len(P)$ 为 P 的欧拉长度);如果存在点集 C_u 对于所有的 P 满足 $C_u \subseteq P$,则 C_u 为 $B_{u,2r}$ 的地标集,并且设 $h = \max(|C_u|)$ 为 G 的地标度数($|C_u|$ 为 C_u 的节点个数)。显然

收稿日期:2011-09-07;修回日期:2011-11-20。

作者简介:孟珂(1987-),男,江苏徐州人,硕士研究生,CCF会员,主要研究方向:GIS、路径规划;张春艳(1985-),女,江苏徐州人,硕士研究生,主要研究方向:云计算、蚁群算法。

h 越大地标对最短路径的贡献越大,在路径规划时可利用的地标节点越多。

计算大规模交通网络的地标集,采用以下几个步骤。

1) 选择节点密集型的区域,将图分割为搜索半径为 r 的不同区域。在实验中会讨论 r 在不同取值时地标集获取情况以及启发式算法的查询速度。

2) 对于区域 $B_{u,2r}$,使用 CH 算法进行预处理以便于快速计算最短路径。

3) 为 $B_{u,4r}$ 中的每一个点对计算最短路径,获取最短路径集合 $P = \{P_{v,w} \mid v, w \in B_{u,4r}, |P_{v,w}| > r\}$ 。

4) 对 P 中所有的路径取交集获得地标集 C_u 。

2.2 启发式算法设计

本文将地标节点作为启发式搜索的启发节点,求解思想如下。

对于点对 (s, t) 如果属于同一分割区域,由于使用了 CH 算法进行预处理,可以快速求得精确的最短路径。如果 (s, t) 属于不同区域则使用以下启发式规则。

1) 从 s 所在区域的地标集中选取距离 t 最近的地标 c 作为下一跳的启发节点, s 到 c 的最短路径使用 CH 求得。如果 s 所在区域没有地标集,则设 $c = s$,转向第 2) 步。

2) 从邻近区域的地标集中选取距离 t 最近的地标 c' 作为启发点,使用 ALT 算法求出 (c, c') 的最短路径。

3) 重复以上步骤直到 c' 与 t 在同一分割区域,使用 CH 算法计算 (c', t) 最短路径。

4) 对最短路径进行合并输出。

CH 算法在区域内进行快速搜索,同时对于不同区域采用 ALT 算法控制搜索方向,使搜索始终沿着目标进行,这是一种分段搜索的思想,对于长距离的最短路径求解由于有地标集提供搜索参考,搜索线路比 ALT 更加精确,耗时更短。

根据以上算法思想,对于跨区域的分段启发函数定义如下:

$$f(i) = g(i) + \begin{cases} CH(i, c'), c' = \min_c \{dist(c, t) \mid c \in C(i)\} \\ ALT(i, c''), c'' = \min_c \{dist(c, t) \mid c \in C_{nextto}(i)\} \end{cases}$$

其中, $f(i)$ 为节点 i 的启发函数, $g(i)$ 为起始点 s 到当前节点 i 的最短距离, c' 为 i 所在区域地标集 $C(i)$ 距离目标点 t 最近的地标, $CH(i, c')$ 使用 CH 计算同一区域内 (i, c') 的最短路径, c'' 为 i 邻近区域地标集 $C_{nextto}(i)$ 距离目标点 t 最近的地标, $ALT(i, c'')$ 使用 ALT 算法计算 (i, c'') 的最短路径, $dist(c, t)$ 指 (c, t) 之间的距离。

设 $P_{s,t}$ 为点 s 和 t 之间的最短路径, $g(t)$ 为 s 到 t 的最短距离,根据以上算法思想,对于已经获取地标集的网络 G ,地标导向的启发式算法 (Contraction Hierarchies A* Landmarks Triangle Algorithm, CHALT) 执行过程描述如下:

1) 如果 $(s, t) \subseteq B_{s,2r}$, 计算 $CH(s, t)$ 输出 $P_{s,t}, g(t)$, 结束算法; 否则转向第 2) 步;

2) 将 t 加入 $C(t)$, 如果 $C(t) = \emptyset$, 则 $C(t) = \{t\}$; 并设 $i = s, P_{s,t} = \emptyset$;

3) while $i \neq t$ 并且 $(c, t) \not\subseteq B_{i,2r}$

如果 $C(i) \neq \emptyset$, 选取 $c = \min_c \{dist(c, t) \mid c \in C(i)\}$, 计算 $CH(i, c)$, 得到 $P_{i,c}$;

否则, 如果 $C_{nextto}(i) = \emptyset$, 计算 $ALT(i, t)$, 获取 $P_{i,t}$;
设 $c = t$

否则, 选取 $c = \min_c \{dist(c, t) \mid c \in C_{nextto}(i)\}$, 计

算 $ALT(i, c)$, 得到 $P_{i,c}$;

$P_{s,t} = P_{s,t} \cup P_{i,c}$;

$g(i) = g(i) + Len(P_{i,c})$;

$i = c$;

4) 从 $C(t)$ 中去除节点 t ;

5) 输出 $P_{s,t}$ 和 $g(t)$;

6) 算法结束。

2.3 算法分析与优化

CHALT 算法的地标集预处理比较耗时,但可以在多项式时间之内完成计算。对于 G 的一个稠密子图,从空集开始,使用 CH 算法从所有待处理的路径中选取一个覆盖所有路径的点,然后将此点从图中移除,对剩余路径迭代计算,直到不存在满足条件的点为止,在有限次迭代后算法会终止。对于图预处理的时间复杂度为 $O(n \log nO(CH))$, 其中 n 为子图的节点个数, $O(CH)$ 为 CH 算法的时间复杂度。

对于 ALT 算法,双向搜索的收敛速度一般比单向搜索快^[9],因此使用双向 CHALT 查询可以获得更好的时间效率,具体执行步骤如下:

1) 使用前向搜索计算 (s, t) 的最短路径获取一个启发点 c_f ;

2) 使用后向搜索计算 (t, s) 的最短路径获取一个启发点 c_b ;

3) 设 $s = c_f, t = c_b$ 重复 1), 2) 两步,最终搜索会在同一个节点相遇;

4) 合并前向搜索和后向搜索的最短路径后输出。

对于地标集,可以使用 TNR^[10] 的思想进行最短路径索引, TNR 计算并存储所有地标之间的最短路径并存储在一张 $|C| \times |C|$ 的表格中, 其中 $|C|$ 为图 G 中地标节点的个数。如果 s 和 t 分别在不同的分割区域,并且存在地标,则根据索引表查询地标之间的最短路径,否则执行启发式搜索。对地标的查询可以在常数时间之内完成。大规模网络图使用 CHALT + TNR 算法可以在牺牲少量存储空间的前提下提供最优的性能。

3 实验

使用 Intel Pentium CPU 2.5 GHz, 2 GB RAM 完成本算法和其他算法的比较实验,算法采用 C++ 编写。实验数据选用北京市交通路网(包含 81 534 个路段和 34 219 个节点)。最短路径的度量标准为距离最短,在实验中使用欧拉距离完成路径计算。

表 1 为不同的最短路径算法在 1 000 组随机查询中的平均时间比较。预处理的时间使用分钟计算,预处理每节点所占用的额外空间单位为字节,额外空间为负说明预处理后的搜索图比原图规模小。从表 1 中可以看出 ARC-Flags 和 SHARC 虽然执行效率比较高,但需要长时间的预处理,并且节点变更对算法的影响大,不适用于大规模网络; CHALT 算法执行时间属于中上等,但预处理时间短,在经过 TNR 优化后的执行时间接近 SHARC 算法的查询时间;双向 CHALT 算法在时间上比单向快一些。由于 CHALT 使用地标节点作为启发参数,地标节点仅占所有节点的小部分,不容易受到节点变更的影响。

在 CHALT 算法中,划分区域的大小将影响地标集的选取

和路径规划结果。表2表示不同搜索半径 r 对查询速度的影响, r 的单位为km。从表2中可以看出在 $r=3\text{ km}$ 和 $r=4\text{ km}$ 时候在预处理时间少的情况下依然可以获得不错的查询效率,极端情况下 $r=0$ 时算法变为ALT算法; $r=\infty$ 时算法将仅使用CH算法,地标节点个数接近于0,启发函数不可用,也就失去了地标的参考价值。在实际应用中需要根据实验来确定合适的搜索半径,来达到效率与合理性的权衡。CHALT算法获取的解为近似解,但接近最优解,如图1(图1中黑色路径为CHALT算法,白色路径为Dijkstra算法)。CHALT算法优先选择重要的节点和边,在地图上表现为主要的街道和路口;Dijkstra算法对所有与 (s,t) 相关的路径计算以获得最优解,而不会考虑节点的重要性,在实际应用中存在不合理性。CHALT算法获取的路径比Dijkstra更平滑并且更合理。

表1 各种算法的预处理时间和1000次查询的平均时间

算法	预处理		查询结果	
	预处理时间/min	每节点占用额外空间/字节	每次查询扫描节点数	平均时间/ms
Dijkstra	/	/	14754	1023.00
ALT(16) ^{[3]41}	2	70	6354	47.10
REACH ^{[3]39}	5	17	4371	4.23
CH	10	-3	311	0.18
ARC-Flags	42	25	593	0.75
SHARC	19	20	145	0.09
CHALT	4	-3	283	0.34
CHALT(BI)	4	-3	172	0.27
CHALT+TNR	6	86	84	0.07

表2 CHALT不同搜索半径对预处理和查询的影响(1000次查询)

搜索半径/km	预处理时间/min	查询结果	
		每次查询扫描节点数	平均时间/ms
1	1.2	625	0.86
2	1.8	587	0.62
3	3.1	322	0.39
4	4.3	283	0.34
5	4.8	266	0.31
6	7.3	231	0.21
7	11.0	220	0.16

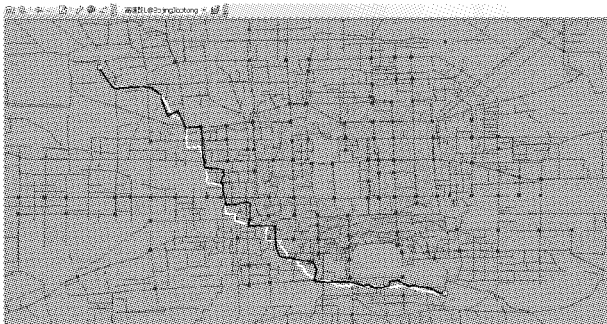


图1 使用Dijkstra算法与CHALT算法获取的最短路径

4 结语

为解决大规模长距离的最短路径规划问题,本文根据分段计算的思想,使用地标集将启发式搜索限制在靠近最短路径的方向。实验证明CHALT算法在保证预处理和查询效率的基础上,得出更合理的计算结果,优化后的算法查询效率更高,可以应用在大型交通网络中。下一步研究方向为以地标为导向的启发式算法在离散变权网络中的应用。

参考文献:

- [1] DIJKSTRA E W. A note on two problems in connexion with graphs [J]. *Numerische Mathematik*, 1959(1): 269–271.
- [2] GOLDBERG A V, KAPLAN H, WERNECK R F. Reach for A*: Efficient point-to-point shortest path algorithms [C]// *Proceedings of 7th International Workshop on Algorithm Engineering and Experiments*. Miami: SIAM, 2006: 129–143.
- [3] GOLDBERG A V, KAPLAN H, WERNECK R F. Better landmarks within reach [C]// *WEA'07: Proceedings of the 6th International Conference on Experimental Algorithms*. Berlin: Springer-Verlag, 2007: 38–51.
- [4] GEISBERGER R, SANDERS P, SCHULTES D, *et al.* Contraction hierarchies: faster and simpler hierarchical routing in road networks [C]// *WEA'08: Proceedings of the 7th International Conference on Experimental Algorithms*. Berlin: Springer-Verlag, 2008: 319–333.
- [5] KÖHLER E, MÖHRING R H, SCHILLING H. Fast point-to-point shortest path computations with arc-flags [C]// *The Shortest Path Problem: Ninth DIMACS Implementation Challenge*. Piscataway: IEEE, 2009: 41–72.
- [6] LAUTHER U. An extremely fast, exact algorithm for finding shortest paths in static networks with geographical background [EB/OL]. [2011-07-02]. <http://gi-days.de/archive/2004/downloads/gi-tage2004/vortraege/lauther.pdf>.
- [7] BAUER R, DELLING D. SHARC: Fast and robust unidirectional routing [J]. *ACM Journal of Experimental Algorithmics*, 2009, 14(12): 12–26.
- [8] DELLING D. Time-dependent SHARC-routing [J]. *Algorithmica*, 2009, 60(7): 60–94.
- [9] GOLDBERG A V, HARRELSON C. Computing the shortest path: A* search meets graph theory, #MSR-TR-2004-24 [R]. USA: Microsoft Research, 2004.
- [10] BAST H, FUNKE S, SANDERS P, *et al.* Fast routing in road networks with transit nodes [J]. *Science*, 2007, 316(5824): 566–593.
- [11] HILGER M. Accelerating point-to-point shortest path computations in large scale networks [R]. Berlin: Technische University, 2007.
- [12] SANDERS P, SCHULTES D. Highway hierarchies hasten exact shortest path queries [EB/OL]. [2011-06-06]. <http://algo2.iti.kit.edu/schultes/hwy/esaHwyHierarchies.pdf>.
- [10] DEZERT J, SMARANDACHE F. On the generation of hyper-power-sets for the DSMT [EB/OL]. [2011-05-01]. <http://mmsip.bas.bg/mmosi/partners/s47.pdf>.
- [11] 曲圣杰,程咏梅,潘泉,等.冲突再分配DSMT及解决证据间矛盾的新方法[J].*控制与决策*,2009,24(12):1856–1860.

(上接第1052页)

- [8] 张浩,蔡晋辉,周泽魁. DS证据理论在SAR图像边缘检测中的应用[J]. *武汉大学学报:信息科学版*, 2008, 33(1): 105–108.
- [9] SMETS P. The combination of evidence in transferable belief model [J]. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1990, 12(5): 447–458.