第44卷第10期 中国科学技术大学学报 Vol. 44, No. 10
2014年10月 JOURNAL OF UNIVERSITY OF SCIENCE AND TECHNOLOGY OF CHINA Oct. 2014

文章编号:0253-2778(2014)10-0867-07

路网上异步并行加权 A* 最短路径算法

冷勋泰,孙广中

(中国科学技术大学计算机科学与技术学院,安徽合肥 230027)

摘要:图上最短路径问题是一个经典问题,应用广泛.对于路网路径的计算,要求程序能够在有限的时间内找到一条尽量短的路径,且允许运行的时间越长,找到的路径越短.由于传统的最短路径算法在设计时未考虑这一约束条件,故不能满足应用需求.为此提一种 APWA* (asynchronous parallelism weighted A*)算法,该算法能够响应用户的中断信号并返回当前找到的最短的路径.在多个地图数据上的实验表明,APWA* 能够很好地满足实际需求.

关键词:路网;最短路径;异步并行

中图分类号:TP301 **文献标识码**:A doi:10.3969/j.issn.0253-2778.2014.10.011

引用格式:Leng Xuntai, Sun Guangzhong. Asynchronous parallelism weighted A* algorithm for the finding shortest path on road networks[J]. Journal of University of Science and Technology of China, 2014, 44(10):867-873.

冷勋泰,孙广中. 路网上异步并行加权 A* 最短路径算法[J]. 中国科学技术大学学报,2014,44(10):867-873.

Asynchronous parallelism weighted A* algorithm for the finding shortest path on road networks

LENG Xuntai, SUN Guangzhong

(*School of Computer Science and Technology, University of Science and Technology of China, Hefei* 230027, *China*)

Abstract: Finding the shortest path is a classic problem with numerous applications. For road networks, it is desirable to find a sufficient short path within a limited period of time, and find a shorter path if there is more time. Since the traditional shortest path algorithms did not consider this constraint when designed, they can not meet the application requirement. To address this issue, an algorithm called APWA*, asynchronous parallelism weighted A*, was proposed, which can respond to users' interrupt signal and return to the currently shortest path. Experiments on multiple maps show APWA* can meet the application requirement.

Key words: road network; shortest path; asynchronous parallelism

收稿日期:2014-01-17;**修回日期**:2014-05-19

基金项目:国家自然科学基金(61033009,61303047)资助.

作者简介:冷勋泰,男,1988年生,硕士生. 研究方向:路由计算. E-mail: xuntai@mail.ustc.edu.cn

通讯作者:孙广中,博士/副教授. E-mail: gzsun@ustc.edu.cn

0 引言

图上点到点的最短路径计算在诸多领域有着广泛的应用.随着信息化的高速发展,路网趋于精细,数据量大.在路网路径计算的实际应用中,如果最短路径的计算需要大量时间,那么用户更倾向于能够在短时间内找到一条与最短路径相差不大的路径.若一个路径计算算法找到的路径随着运行时间的增加而更加接近最短路径,则称这个算法是一个任意时间算法(anytime algorithm).

由于路网图数据过于庞大,使得传统算法在计算最短路径时较为耗时.近年来,学术界提出了一系列基于预处理的最短路径计算算法[1].预处理算法通常假定图数据是静态的,且预处理时间较长,不能很好地适应动态变化的路网.同时,云计算平台(如微软 Windows Azure 平台[2])和多核系统的出现与普及,使得计算时有冗余资源可以选择.

本文提出的 APWA* (asynchronous parallelism weighted A*)算法能够随时间给出不同长度的路径,且允许计算的时间越长找到的路径的长度就越接近最短路径长度,很好地满足了实际需求.APWA*算法异步运行多个权值不同的 WA*算法实例,并在用户给出中断信号后终止各个 WA*算法实例的执行,然后从已完成路径计算的 WA*算法实例中选取最短的路径返回.实验在 Windows Azure 云计算平台上进行,选取了真实地图数据作为实验数据,取得了良好的实验结果.

1 相关工作

1.1 Dijkstra、A*、WA*

Dijkstra、A*[3-4]、WA*[5-6]算法均可通过维护 Opened 表及 Closed 表来实现.初始时,Opened 表中仅包含源点,Closed 表为空.算法开始后反复地从 Opened 表中移除节点,并将移除的节点放入 Closed 表中,更新移除节点的邻居点信息,若其邻居在 Closed 表中,则不做任何操作,否则将其放入 Opened 表中.我们称这个过程为 Opened 表的扩展.当 Opened 表为空或移除的节点是目标节点时,算法终止.Dijkstra、A*、WA*算法主要的不同,在于 A*及 WA*利用了点到点的最短路径长度的估值,加快了 Opened 表向目标点扩展,从而能够更快地找到路径.其中,WA*算法不能保证找到的是最短路径.

1.2 基于 WA* 的 Anytime 算法

基于 WA*的 Anytime 算法采用多轮迭代的方法在每一轮返回一条路径,为加速路径计算,它们均使用了上一轮计算得到的节点信息,并在新一轮计算时减小权值.ARA*算法[6]在新一轮计算开始前,会将上一轮得到的 Opened 表及 InCons 表合并更新得到新的 Opened 表,然后继续扩展 Opened 表,在计算过程中,每个节点最多被扩展一次.与 ARA*算法不同,RWA*算法[7]每一轮起始时,Opened 表仅包含源点.Richter 等认为,保留上一轮 Opened 表将会先扩展靠近目标节点的节点,不利于找到新的更短的路径[7].在 AHS 算法[8]中,将利用上一轮找到的路径的长度作为剪枝标准,避免将不可能在新的路径上的节点加入 Opened 表,提高了 Opened 表操作效率.ANA*算法[9]与上述算法最大的不同,在于该算法修改了 Opened 表节点移除规则,因此不需要设置或调整权值.ANA*算法一方面在运行过程中对节点进行剪枝,另一方面又在开始新一轮计算时避免从目标节点开始扩展,有利于找到新的更短路径.

2 APWA* 算法

2.1 问题描述

假设 p 是图上从源点到目标点的一条路径,p^* 是一条最短路径.并记路径 p 的长度为 $|p|$,则路径 p 的路径质量为 $q(p)=|p|/|p^*|$,路径质量越接近 1,表明路径 p 的长度越接近最短路径长度.与传统最短路径问题不同,APWA*算法要解决的是路径计算时间和路径质量之间的平衡问题.即 APWA*倾向于快速找到一条尽量短的路径,且允许运行的时间越长,找到的路径就越短.

2.2 WA* 算法

WA*算法在 A*算法的基础上引入了权值参数 $\varepsilon(\varepsilon>1.0)$,其算法伪代码如下:

```
Procedure update(n, pre, dist, ε)
  n.pre=pre; n.dist=dist;
  n.full=dist + ε · h(n);

Procedure expand(n, Closed, Opened)
  for each n′ in neighbors(n)
    cur_g=n.dist + cost(n, n′);
    if (cur_g > n′.dist or n′ in Closed)
      continue;
```

```
    update(n′, n, cur_g, ε);
    if (n′ in Opened)
      remove n′ from Opened;
    insert n′ into Opened;

Procedure WA*(s, t, ε)
  update(s, s, 0, ε);
  Closed={}; Opened={s};
  while (Opened not empty)
    remove n with minimum n.full from Opened;
    insert n into Closed;
    if (n is t)
      break;
    expand(n, Closed, Opened);
```

其中，n.dist 是从源点到点 n 的已知的最短的路径的长度，n.pre 则记录了使 n.dist 更新的邻接点. 函数 $h(n)$将返回节点 n 到目标点距离的估值，称其为启发函数；函数 neighbors(n)返回点 n 的邻居点集合；函数 cost(u, v)返回点 u 到点 v 的边长. 若对图上任意两点 u、v，均有 $h(u)-h(v)\leqslant d^*(u, v)$. 其中，$d^*(u, v)$是从 u 点到 v 点的最短路径的长度，则称 $h(n)$满足一致性(consistent). 此时，A* 算法($\varepsilon=1.0$)能够找到最短路径[3]，WA*($\varepsilon>1.0$)算法找到的路径长度与最短路径长度的比值不超过 ε[5-6].

当 ε 取值为 1.0 时，WA* 算法退化为 A* 算法；当 ε 取值为 0 时，WA* 算法退化为 Dijkstra 算法.

2.3　APWA* 算法描述

APWA* 算法伪代码描述如下：

```
class Route
  Route(s, t, ε, pathLength=∞,
    interrupted=false, reachable=false, finished=false);
  update(n, pre, dist, ε)...;
  expand(n, Closed, Opened)...;
  weightedAstar()
    update(s, s, 0, ε);
    Closed={}; Opened={s};
    while (Opened not empty)
      if (interrupted)
        break;
      remove n with minimum n.full from Opened;
      insert n into Closed;
      if (n is t)
        reachable=true;
        pathLength=n.dist;
        break;
      expand(n, Closed, Opened);
  run()
    weightedAstar();
    finished=true;

class APWA
  APWA*(s, t, time=null, interrupted=false, routes=null);
  getEpsilons()...;
  moniter()...;
  isReachable()...;
  isFinishedAll()...;
  getShortestRoute()...;
  main()
    start another thread to run moniter();
    declare Route objects routes by getEpsilons();
    for each route in routes
      start a thread to run route.run();
    while (not interrupted)
      if (not isReachable() or isFinishedAll())
        break;
    if (not isFinishedAll())
      interrupte all routes;
      wait for all Route thread finished;
    solution=getShortestRoute();
    return solution
```

APWA 类通过 Route 类异步运行多个参数不同的 WA* 算法实例，通过 APWA.getShortestRoute()方法返回当前已找到的最短的路径。其中，APWA.main() 是主方法入口；APWA.moniter()方法异步监测用户中断信号及算法运行时间，并通过 APWA.interrupted 标记是否需要终止运行；APWA.getEpsilons()方法用于获取要异步并行的 WA* 算法实例的权值集合；APWA.isReachable 方法确定源点与目标点是否可达；APWA.isFinishedAll()方法检查所有的 WA* 算法实例是否已完成计算。为方便 APWA 类对 Route 类进行监测，Route.weightedAsta()方法在 WA* 算法的基础上增加了对一些变量的检测和操作，它们是 Route.interrupted、Route.reachable 和 Route.finished。此外，APWA.update() 方法及 APWA.expand()方法与 WA* 算法伪代码同名方法相同.

2.4　APWA* 权值集合选择策略

影响权值集合选择的因素有：平台可异步并行

的实例数、用户设定的运行时间、源点与目标点的距离等.简单起见,本文仅考虑可异步并行的实例数.为保证一定能够找到最短路径,权值集合应包含权值1.0,对应A*算法;为了能够尽快找到一条路径,权值集合还应包含值为无穷大的权值.这样,当可异步并行的实例数为num时,我们还需要确定其他num−2个WA*算法实例的权值.实验表明,当我们从序列1.01, 1.03, 1.07, 1.15, 1.31, 1.63…中顺序选取num−2个权值时,APWA*算法能够有效地在路径计算时间和路径质量之间取得平衡.这个序列从1.01开始,间隔从0.02逐倍递增.

2.5 APWA*算法与其他算法

用于求解最短路径的传统anytime算法在运行时只有一个线程,且将路径计算过程划分为多个轮次依次进行;APWA*则异步运行多个权值不同的WA*算法实例.由于利用了更多的计算资源,在相同的运行时间内,APWA*算法能够比传统anytime算法更快地找到新的路径;且避免了上一轮计算信息的干扰[7],APWA*算法在相同权值参数下找到的路径更短.相对于传统的anytime算法,APWA*算法实现简单,求解质量好.

3 实验结果

3.1 实验说明

实验在Windows Azure云计算平台进行.该计算平台采用Windows Server 2008 R2 Datacenter 64位操作系统,处理器为AMD Opteron(TM) Processor 4171和20.9 GHz (2 processors),共有8核CPU,允许访问的最大内存空间为56 G.实验选取了欧洲尼德兰(Netherlands)、美国西部(USA-road-d.W)、新墨西哥州(New Mexico)、纽约州(USA-road-d.NY)四个国家或地区的地图数据作为实验数据.其中,New Mexico及Netherlands地图数据来自互联网,格式为ShapeFile[10],USA-road-d.NY及USA-road-d.W地图数据来自第9届DIMACS实现挑战赛[11].地图数据特征如表1所示.

表1 实验数据集概况

Tab.1 Experimental datasets

数据名	节点数	边数	类型
New Mexico	3 532 776	3 663 670	ShapeFile
Netherlands	4 601 037	5 208 135	ShapeFile
USA-road-d.NY	264 346	733 846	DIMACS
USA-road-d.W	6 262 104	15 248 146	DIMACS

实验采用Java 1.7进行开发,通过diewald_shapeFile读图包读取ShapeFile格式文件,使用Thread创建并运行线程.在构造地图时,选取两点间的直线距离作为相邻两点的边长.使用点到目标点的直线距离作为点到目标点的最短路径长度的估值.显然,这个估值满足一致性.实验分为两部分:第一部分给出了WA*算法权值参数与路径计算所需时间及路径质量之间的关系;第二部分给出了APWA*算法运行时间与路径质量之间的关系.

3.2 WA*算法

为了讨论WA*算法参数与路径计算时间和路径质量关系,我们通过一系列实验观察了WA*算法参数对路径计算时间和路径质量的影响.在图1和图2中,WA*算法使用的参数ε有1.0、1.01、1.03、1.07、1.15、1.31、1.63、2.27、3.55.实验中,对每一个地图数据,均随机选取了500个点对分别运行上述不同参数的WA*算法求解点到点的路径.

图1显示了参数ε与平均路径计算时间之间的关系.实验结果以参数为1.0时的WA*(即A*)路径计算时间进行归一化.从图1可以看出,总的来说,随着参数增大,计算路径所需时间变小;且在参数值超过一定值之后,路径计算时间不再随参数的增大而减小.图2显示了参数与平均路径质量之间的关系.从图2可以看到,总的来说,随着参数增大,所找到的路径的路径质量也在增大,即路径长度在增大.此外,从图2还可看到,找到的路径的路径质量远小于对应的参数ε,与1.0非常接近.

3.3 APWA*算法

为了模拟用户中断路径计算这一行为,我们以30ms为初值,成倍增加允许程序运行的时间.最终选取了30 ms, 60 ms, 120 ms, 240 ms, 480 ms, 960 ms, 1 920 ms, 3 840 ms共8个时间间隔.图3和图4显示了APWA*允许运行的时间与取得的路径质量之间的关系.其中,图3为500个随机点对在不同时间约束下得到的平均路径质量;图4则是5组随机点对在不同时间约束下得到的路径质量.

从图3,4可以看到,随着允许运行的时间逐渐增加,APWA*算法返回的路径的长度逐渐接近最短路径的长度,且在较短时间内找到的路径的长度依然非常接近最短路径长度.比较图3和图4可以看到,APWA*算法的这一性质不仅在平均情况下表现良好,在一般情况下也是成立的.

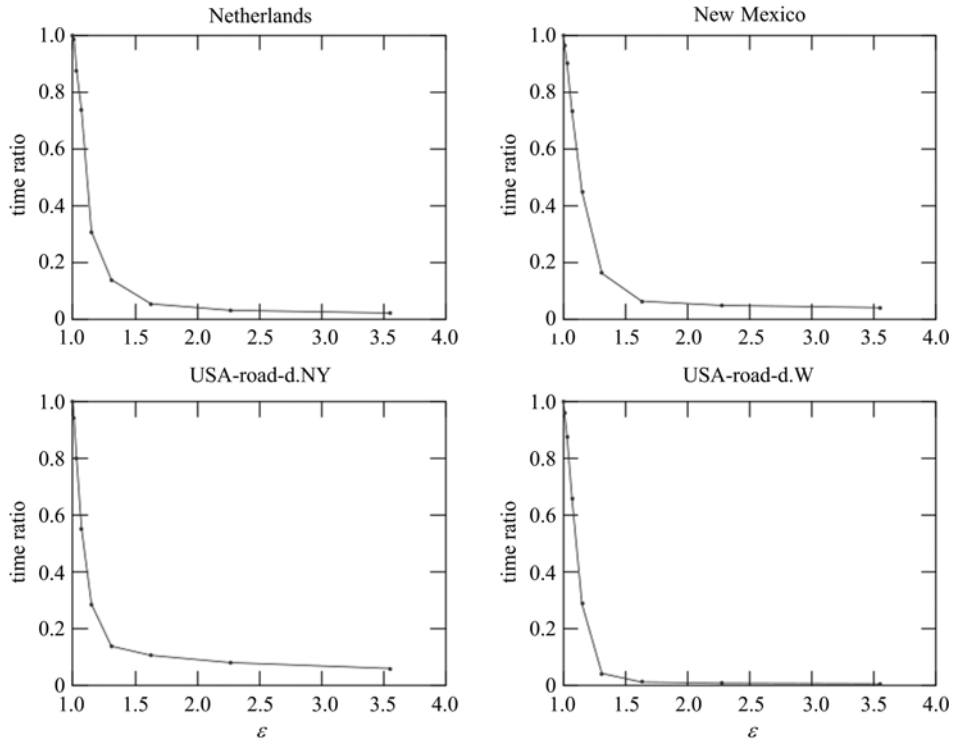

图 1　参数 ε 与路径计算所需时间(均值)

Fig. 1　The average computing time ratio with different ε on WA*

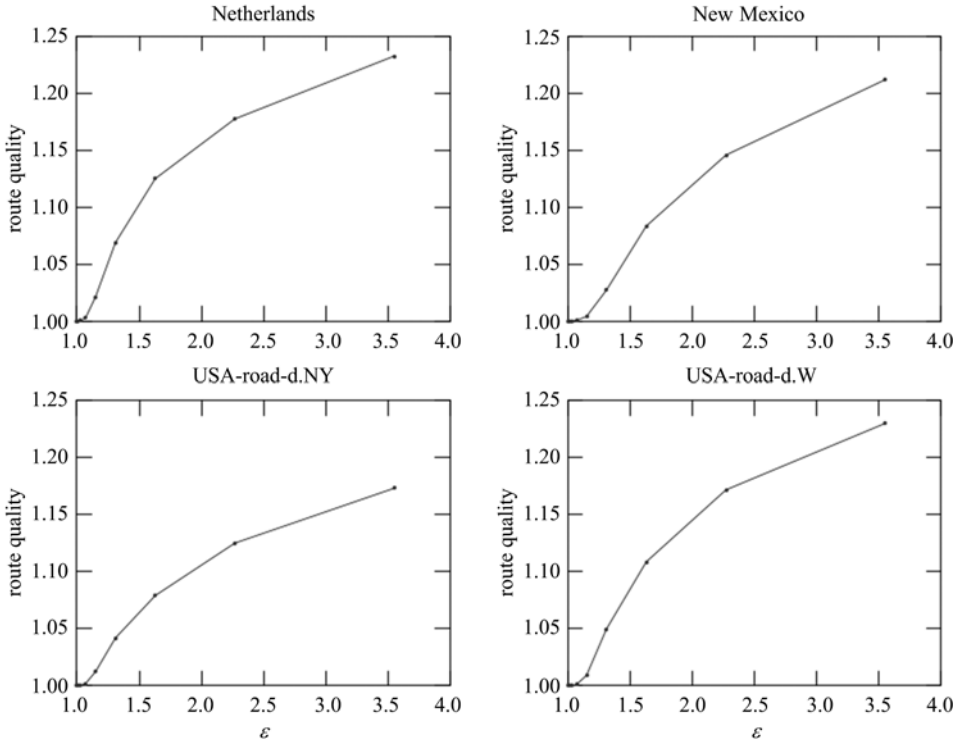

图 2　参数 ε 与路径质量(均值)

Fig. 2　The average route quality with different ε on WA*

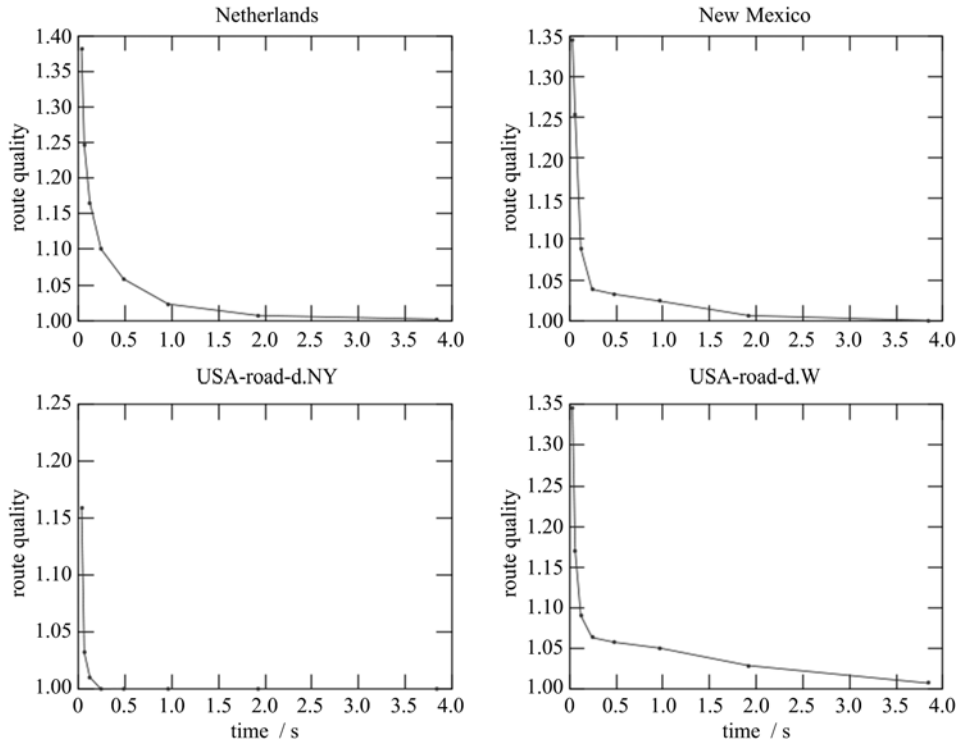

图 3 APWA* 允许运行的时间与取得的路径质量(均值)

Fig. 3 The average route quality under different time constraint on APWA*

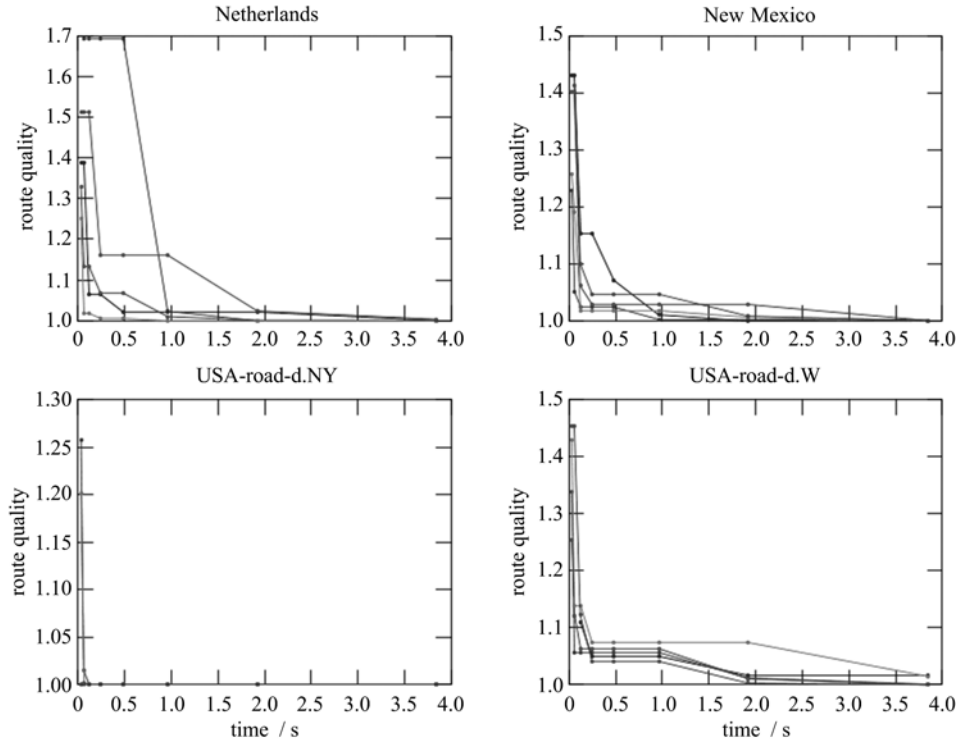

图 4 APWA* 允许运行的时间与取得的路径质量(抽样)

Fig. 4 The sample route qualities under different time constraint on APWA*

4 结论

本文提出了一个异步并行的路网最短路径计算算法，APWA* 算法. 该算法异步运行多个权值不同的 WA* 算法实例，并从中返回已找到的最短的一条路径，从而使返回的路径的长度随着允许计算的时间的增大而减小，很好地满足了路网路径查询的实际需求.

参考文献(References)

[1] Wagner D, Willhalm T. Speed-up techniques for shortest-path computations[C]// Proceedings of the 24th International Symposium on Theoretical Aspects of Computer Science. Aachen, Germany: Springer, 2007: 23-36.

[2] Zhang Q, Cheng L, Boutaba R. Cloud computing: State-of-the-art and research challenges[J]. Journal of Internet Services and Applications, 2010, 1(1): 7-18.

[3] Hart P E, Nilsson N J, Raphael B. A formal basis for the heuristic determination of minimum cost paths[J]. IEEE Transactions on Systems Science and Cybernetics, 1968, 4(2): 100-107.

[4] Dechter R, Pearl J. Generalized best-first search strategies and the optimality of A* [J]. Journal of the ACM, 1985, 32(3): 505-536.

[5] Pearl J. Heuristic: Intelligent Search Strategies for Computer Problem Solving[M]. Michigan: Addison-Wesley, 1984.

[6] Likhachev M, Gordon G, Thrun S. ARA*: Anytime A* with provable bounds on sub-optimality[C]// Proceedings of the Conference on Neural Information Processing Systems. Cambridge, Canada: MIT Press, 2003: 102-120.

[7] Richter S, Thayer J T, Ruml W. The joy of forgetting: Faster anytime search via restarting[C]// Proceedings of the 20th International Conference on Automated Planning and Scheduling. Toronto, Canada: 2010: 137-144.

[8] Hansen E A, Zhou R. Anytime heuristic search[J]. Journal of Artificial Intelligence Research, 2007, 28(1): 267-297.

[9] van den Berg J, Shah R, Huang A, et al. ANA*: Anytime nonparametric A* [3]// 25th Association for the Advancement of Artificial Intelligence Conference. San Francisco, USA: AAAI Press, 2011: 105-111.

[10] 9th DIMACS Implementation Challenge [EB/OL]. http://www.dis.uniroma1.it/challenge9/download.shtml.

[11] STAT Silk. ShapeFile data[EB/OL]. http://www.statsilk.com/maps/download-free-shapefile-maps.

(上接第 866 页)

动速度进行了测试. Matlab 仿真结果和 Wi-Fi 应用实验结果均表明，在快速移动环境下，若能合理设置扫描间隔，基于多无线模块的 AP 间快速切换方法可以保证较低的 AP 间切换误判率，满足车载 Wi-Fi 应用的切换判决. 下一步计划综合考虑高速移动环境下多径传输等因素对无线信号传输的影响，继续完善本文的理论模型.

参考文献(References)

[1] Morgan Stanley Global Technology & Telecom Reseach. The Mobile Internet Report[M]. America: Morgan Stanley Research, 2009.

[2] 中国互联网络信息中心. 第 28 次中国互联网络发展状况统计报告[R]. 北京：中国互联网络信息中心，2011.

[3] Federal Railroad Administration. Study of high-speed wireless data transmissions for railroad operation[R]. Department Of Transportation, US, 2007.

[4] Eriksson J, Balakrishnan H, Madden S. Cabernet: Vehicular content delivery using wifi [C]// Proceedings of the 14th Annual International Conference on Mobile Computing and Networking. San Francisco, USA: ACM Press, 2008: 199-210.

[5] 朱铨，蒋新华，邹复民. 交通干线无线宽带覆盖网络传输性能研究[J]. 计算机工程与应用，2012，48(9)：15-17，50.

[6] 邹复民，蒋新华，林漳希，等. 一种基于榕树型拓扑的铁路无线 Mesh 网络结构[J]. 铁道学报，2010，32(2)：47-54.

[7] 康玉文，邹复民，康兴斌，等. 几种典型车载移动环境的 WiFi 扫描机制研究[J]. 计算机工程与应用，2012，49(19)：89-96.

[8] 邹复民，蒋新华，王桐森，等. 一种支持车-地宽带互联的广义 AP 间切换模型[J]. 铁道学报，2011，33(2)：45-51.

[9] 蒋新华，邹复民，王桐森，等. 一种多无线模块快速切换方法，中国，200810071237[P]. 2009-12-23.

[10] 鄢丹，蒋新华，邹复民，等. 采用多无线模块的 AP 间硬切换的实验研究[J]. 计算机工程与应用，2010，46(29)：124-126.

[11] 鄢丹. 车载 WiFi 应用快速 AP 间切换关键技术研究[D]. 长沙：中南大学，2011.