第34卷 第20期 计 算 机 工 程 2008年10月
Vol.34 No.20 Computer Engineering October 2008
·人工智能及识别技术· 文章编号：1000—3428(2008)20—0203—02 文献标识码：A 中图分类号：TP393

基于 Agent 的动态路网行车最短路径求解

付天成，莫松海，王 晖，郑黎明
(国防科技大学信息系统与管理学院，长沙 410073)

摘 要：针对动态路网中最短路径求解算法复杂度高、计算量大、响应不及时等问题，提出基于 Agent 的分布式求解方法。用 kd-tree 将整个路网分区，每个区域由一个 RMA Agent 进行管理，利用多个 Agent 协作求解最短路径。实验表明，在路网节点较多且变化频繁时，该方法具备优势。

关键词：最短路径；多 Agent；Dijkstra 算法；运输问题

Solution of Shortest Path in Dynamic Traffic Network Based on Agent

FU Tian-cheng, MO Song-hai, WANG Hui, ZHENG Li-ming
(School of Information Systems and Management, National University of Defense Technology, Changsha 410073)

【Abstract】 The shortest path algorithm in dynamic traffic network is complex and has no timely response. The problem raises Agent-Based distributed solving method. This paper divides the entire network into sub-regions with kd-tree and each region is managed by RMA Agent. The problem is solved by collaboration. Experiments show that when the nodes are numerous and have more frequent changes, this method performs better.

【Key words】 shortest path; multi Agent; Dijkstra algorithm; transporting problem

Dijkstra算法是求解无向赋权图内单源最短路径的经典算法。然而，随着研究的深入进行，人们发现针对串行计算机的最短路径算法已经几乎到达理论上的时间复杂度极限[1]。现有的研究大多都集中在静态网络中最短路径的优化求解，对路网状态动态变化情况下的求解方法研究较少[2]。最短路径求解的实时性和并行化已经成为研究的发展方向[2]。本文在文献[3]的基于图分割加速的Dijkstra算法基础上，利用多Agent技术求解动态路网最短路径，使求解过程具有一定的并行性，并能更好地响应路网的动态变化。

1 问题的描述

本文将路网抽象为一个简单的无向赋权稀疏图，设图$D=(V, E, W)$，其中，V是图中所有的n 个顶点的集合$V=\{v_1, v_2,\cdots,v_n\}$，根据求解的精确不同，可以将顶点视为路网上的路口或是某一地域；E是图中所有的m条边的集合$E=\{e_1, e_2,\cdots, e_m\}$；$W$是边的权值集合$W=\{l_1, l_2,\cdots,l_m\}$。权重可以是路径的长度，也可以是运输成本、运输时间或其综合值，设Source和Destination分别代表起始节点和目的节点，为了方便问题求解，假设路网的权值都为正值并且任意一对节点的最短路径只有一条。

2 基于图分割的最短路径算法

文献[3]提出一种基于图分割加速的Dijkstra求解方法，大大降低了问题的复杂度，具体方法如下：在一个静态的无向赋权图中，将节点V所在的平面划分为p个子区域，并将每个节点映射到其中的一个区域，使用一个p bit的*Vector*: *ba*:{1,2,⋯,*p*}->{true,false}，每一个区域用*Vector*中的一个元素标记。当边e_i是节点v_i到区域$region_j$内的某一节点的最短路径上的起始边时，则将这条边的*bit-vector*={1, 2,⋯, *j*,⋯, *p*}内的*j*赋值为true，当用户再次提交请求求解两点间最短路径时，只须搜索到目的节点所在区域的标识为true的边所组成的子图G_t，就能求得最优解，这样大大降低了搜索空间，加快了求解速度。

当 $p=n$ 时，即每个节点都映射到一个 region 中时，则从源节点出发，连接所有到目标节点即目标节点所在区域的边为 true 的边组成的路径就是最短路径；而即使当 $p<<n$ 时，也可以使搜索空间平均缩小 4 倍。文献[4]证明了采用 kd-tree 的方法进行区域的划分可以取得较快的计算速度。

3 算法的不足

文献[3-4]在求解静态网络两点间最短路径问题上有较好的性能，大大降低了问题求解的复杂度，但当网络的某个弧的权值发生变化后，会导致对全网内两点间的最短路径进行重新计算，同时要对每一条边的 *bit-vector* 进行标记，当网络变化频繁时，这种在求解静态网络最短路径的优势反而增加了计算的负担。为了加快问题求解速度和对路网状态的实时响应，本文提出基于分区管理的两点间最短路径问题的 MAS 求解方法，每个分割的区域由一个 Agent 进行管理，随时监视路网的变化，并及时作出响应，为车辆运行路径提供建议。

4 基于图分割的两点间最短路径问题的 MAS 求解

本文将某一时刻的路网设为静止状态，按文献[4]指出 KD 树方法将路网划分为 P 个区域。对于起始节点都在同一区域内的边，称为内含边，起始节点分别在 2 个不同的区域内的边，称为边界边(boundary)。每个 Agent 负责管理某一区

作者简介：付天成(1982－)，男，硕士研究生，主研方向：信息系统工程，多 Agent 系统；莫松海，副教授；王 晖，教授、博士生导师；郑黎明，硕士研究生

收稿日期：2007-11-15 **E-mail：**tcfu.nmg@163.com

域的内含边及边界边，将这个 Agent 称为 Region Manage Agent(RMA)。

4.1 RMA 功能描述

(1)感知路网的变化，并能对本区域内动态变化的路网求解任意一对节点的最短路径；

(2)多个 RMA 可通过通信和协作求解分布在不同区域内节点间的最短路径，能存储一定的全路网优化信息，以防止在进行路径优化时“短视”。

4.2 RMA 设计描述

本文借鉴Cougaar[5]的体系结构对RMA的结构进行描述。Cougaar最初是美国国防高级计划局(DARPA)为建设美军现代化后勤而开发的多Agent系统开发平台。RMA主要由以下几类功能单元构成：

(1)黑板(blackboard)：是一个 Agent 本地的内存存储结构，它接受 Agent 内部各 plugin 在黑板上的预定(subscription)，以获得 plugin 的功能和偏好。通过 Agent 之间的消息传递插件，RMA 之间可以互相进行通信，并将收到的信息(包括路网的变化、路径求解子任务等)以对象的形式发布到黑板上，供 RMA 内的 plugin 计算时使用。

(2)插件(plugin)：完成 Agent 的某一种业务逻辑的软件实体，它随时监视黑板的变化，并根据需要动态加载到 Agent 中对变化的类进行处理，随后将处理结果发布到黑板上。本文中 RMA 主要包括以下几种 plugin：

1)Dijkstra Algorithm(DA) plugin：计算已知路网每一对节点间的最短路径；

2)Road Net Maintenance(RNM) plugin：路网维护插件，负责对路网的修改、对某一子路网的提取等；

3)LDM plugin：逻辑数据结构，主要提供各种数据类型的标准化，使数据信息转化为 RMA 能够识别的结构(类)；

4)Message Transport(MT) plugin：负责 Agent 间消息的发送和接收；

5)Min plugin：将 2 个链表内的各个边的权值分别相加并得到最小值，是多个 RMA 协同求解最短路径时另一个重要的 plugin；

6)Dss plugin：对路网的变化进行评价，是否需要重新进行路径优化；

7)Allocate plugin：负责任务的拆分和分配。

4.3 多 RMA 协同求解最短路径

已知车辆在出发前已经生成了一条最短路径，并向最短路径上各边所在的 RMA 发出预定信息(Taskid)，请求 RMA 对这些边进行监视，并及时通知变化情况。

假设节点 *s* 和节点 *d* 之间的弧段权值发生变化，RMA 首先调用 Dss plugin 对路网的变化进行评价，如果影响为显著的则要进行 *s* 到 *d* 节点路径的重新计算，设 *s* 为 RMA *s* 的一个节点，*d* 为 RMA *d* 的一个节点，下面就几种不同的情况阐述问题求解步骤。

4.3.1 Region *s* 与 Region *t* 是相邻的区域

```
Road Net Maintenance plugin :
Get boundary between RMA s and RMA d
Get the boundary -node in the RMA d
Dijkstra Algorithm plugin(in RMA d):
(List) resultweightd=Dijkstra(d, boundary-node, region d)
//取得节点 d 到边界节点的权值
Message Transport plugin:
Send message containing the result above to RMA s
Dijkstra Algorithm plugin(in RMA s):
(List) resultweights =Dijkstra(s, boundary-node, region s)
//取得节点 s 到边界节点的权值
Min plugin:
shortest path is: Min(resultweightd + resultweights)
//得到最短路径
Message Transport plugin:
Send message containing the result above to the vehicle
//将结果发送给车辆 Agent
```

相邻 RMA 最短路径求解如图 1 所示。

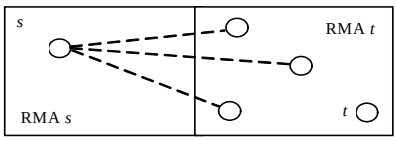

图 1 相邻 RMA 最短路径求解

4.3.2 Region *s* 与 Region *t* 不是相邻的区域

如果通过上述的求解方法不能求得满意解，则需要扩大搜索范围，在最短路径上沿 *s* 点向车辆当前位置回溯一个节点，设该节点为 *b*，位于 Region *b* 内，由 RMA *b* 管理。

当 2 个节点不在同一个 RMA 或相邻的 RMA 内，则从源节点和目的节点进行双向搜索，当双向搜索的末端节点在同一个 RMA 内时，可考虑结束计算，并由该 RMA 分别向起始节点所在 RMA 求解最短路径，最后由 RMA 的 Min plugin 求解 *b*, *t* 之间的最短路径。

```
正向(b-t)：
While receivedmessage.getstaskid is different in RMA i
//如果 RMA i 接收到的任务 i
Get the boundary of RMA i which bit-vectory is true
Record (receivedmessage.getstaskid, taskstep, RMA rma)
//RMA i 要记录任务 ID，任务求解的步骤及“上级”RMA
逆向(t-b)//同正向求解方法
While RMA i receivedmessage.getstaskid form different RMA is same
Trackback from RMA i to RMA b and RMA t
```

至此，可以由求解相邻 RMA 间节点的最短路径算法进行迭代求解。

非相邻 RMA 最短路径求解如图 2 所示。

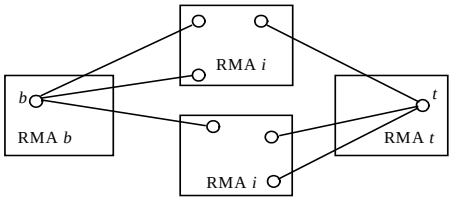

图 2 非相邻 RMA 最短路径求解

5 实验及结果分析

本文利用Cougaar 12.0 version 进行了系统开发，实验计算机的配置：CPU为PⅢ 2.2 GHz，内存为 1 GB。本实验使用 Challenge benchmarks[6]随机生成 3 组路网数据，并计算在不同分区数量情况下求解的速度和精确度，在每一种分区方法下随机请求 50 次任意两点间的最短路径，分别用DIMACS提供的方法和本文的方法进行求解，并计算运算结果的权值及其运算时间，得到的结果见表 1~表 4。

(下转第 207 页)