城市路网的最短路径并行求解

卢 照,师 军,于海蛟,方 昕
(陕西师范大学 计算机科学学院,陕西 西安 710062)

摘 要:随着当前城市规模的不断扩大,交通网络变得越来越复杂,计算最短路径问题不仅要消耗大量的空间资源,同时也花费了更多的时间资源。为了提高最短路径求解的实时性,基于城市小区将复杂网络进行化简,在各个小区中寻找代表节点,将其他无关的节点看做透明的不参与计算,保持原有网络的特性。然后采用并行搜索算法及分层思想进行路径查询,并且在PC机群的并行环境下对其进行实现。实验结果表明,该方法在运行时间和内存空间分配都具有明显的优势,具有良好的实用性。

关键词:大规模网络;城市小区;网络划分;最短路径;MPI
中图分类号:TP393;O157.6 **文献标识码**:A **文章编号**:1673-629X(2010)01-0082-04

Solving the Shortest Path in Parallel of City Road Network

LU Zhao, SHI Jun, YU Hai-jiao, FANG Xin
(College of Computer Science, Shaanxi Normal University, Xi'an 710062, China)

Abstract: With the current scale of urban expansion and transport network more complex, the computation of the shortest path problem will consume more time and spatial resources. In order to have a more efficient real-time, simplify the complex network based on the city community. To maintain the original characteristics of network, in every district to find the representative node, and other nodes as transparent has nothing to do. Then do wayfinding used layered calculation of thought and parallel search algorithms. The experimentation has been implemented in parallel PC environment. The experimental results show that the running time and memory space allocation have a obvious advantage, and have good practicability.

Key words: large-scale networks; city community; network division; the shortest path; MPI

0 引 言

最短路径问题是图论中的一个经典问题,也一直是交通运输问题中应用的一个研究热点。从20世纪50年代末期至今,约有几千篇论文讨论最短路径计算。对最短路径算法的研究热点,一是针对实际网络特征优化数据和算法运行结构,在时间复杂度不变的情况下尽可能地提高算法的运行效率;二是对网络特征进行限制,如要求网络中的边具有整数权值等,以便采用基数堆等数据结构设计算法的运行结构;三是采用有损算法,像限制范围搜索、限定方向搜索及采用层次空间推理方法的限制几何层次递归搜索;四是采用拓扑层次编码路径视图,对最优路径进行部分实例化编码存储,将部分最优路径计算转变为子图间路径的直接查询;五是采用并行算法[1]。19世纪80年代以来,国外许多专家学者对最短路径并行算法的实现方法进行了探讨,建立了串行标号设定算法和标号修正算法的各种不同并行实现方法,并在此基础上对比各种算法的性能及效率。目前,针对各种不同的串行最短路径算法而提出的并行算法大体可以分为两类:一类是不依赖于计算机类型的算法,包括Tseng等[2]、Paige和Kruskal[3]、Chandy和Misra[4]提出的算法;另一类是针对特定计算机类型的算法,包括Habbal等以及Dey和Srimani提出的算法。在有关最短路径并行算法研究的文献中,大部分都是针对全源最短路径问题,大体分为以下3种:基于标号设定的并行算法、基于标号修正的并行算法、动态规划方法。

对于大规模网络来说,由于网络规模较大,采用上述五种方法并不能显著改善算法的效率。要从根本上解决算法的效率问题,就必须对大规模网络进行化简,由于化简后的网络规模可以大幅度减小,最短路径的计算速度就可以大幅度提高。文中在此思想上提出:在对大规模交通网络进行化简的基础上,在PC机群

收稿日期:2009-05-06;**修回日期**:2009-08-17
基金项目:国家自然科学基金资助项目(40471102)
作者简介:卢 照(1983-),男,山西运城人,硕士研究生,研究方向为并行算法、体系结构;师 军,副教授,研究方向为智能信息处理、并行计算。

环境下采用分层思想,利用并行 Dijkstra 算法求解最短路径。

1　大规模交通网络的化简

首先说明的是文中将交通网络中路段的交叉口作为节点,将路段作为连接节点的边。在大规模网络中,由于节点和边都很庞大,计算最短路径不论是从时间还是从所消耗的资源来说,都是很不乐观的。只有通过化简将网络中的节点数和边数都减少,从而提高效率。

1.1　大规模交通网络的特征

大规模交通网络的最大特点是网络中的节点数达到成百上千,甚至几千几万个,并且网络中节点分布比较密集,节点与节点之间的边较短,特别是在小区中体现的更加明显;而各个小区之间边的连接相对来说较少。基于这个特点,可以将大规模交通网络以小区为单位进行划分。

1.2　大规模网络划分的依据

为了能够在多个 PC 机上进行并行处理,最短路径并行算法中的网络分割不仅会影响到各处理器间的通信量,而且还会影响算法的运行时间[5]。网络划分需要具备以下几个特点:

①各个子网之间具有较少的连接边。边界节点数量与最短路径并行算法中各处理器之间的通信量直接相关,通信量会随着边界节点数的增多而变大,从而会增加求解最短路径所需的运行时间。

②考虑到负载平衡问题,各子网包含的节点数趋于平均。最短路径并行算法中,各处理器上的负载主要取决于子网中需要更新的节点标号数。否则负载不平衡时,同步中会出现闲置等待状态,降低运行效率。

③在分界边上的节点分别包含到各个子网中,在计算过程中可以大大减少进程之间的通讯[6]。

1.3　交通网络的分割

为了实现并行处理,就必须经行网络分割。而网络分割的质量会极大地影响并行计算的效率,文中用 Metis 将图进行划分,把网络分割成 P 个子网,每个分配给一个处理器。多个处理器共享的节点叫做边界节点,剩下节点叫做内部节点。

Metis 的算法是基于多层图分割的算法,通过删除点和边把图分割成较小的图,再计算较小图的分割,然后把这个划分映射到原始图中,再进行不断调整。Metis 有两个程序 p－metis 和 k－metis 将图分割成多个部分。文中使用的是 k－metis 进行划分,k－metis 使用的算法是基于多层 k 路分割法。k－metis 的命令格式为:kmetis GraphFile Nparts。其中第一个参数 GraphFile 是存储的图的文件名,第二个参数 Nparts 表示预期划分的部分数。为了减少通讯量,采用最小化边分割,因为分割的边代表了边界点的个数。在用 Metis 分割完后还要将被分割的边节点进行补充,即将被分割后的节点分别添加到相邻的两个子网中。如图 1 所示。

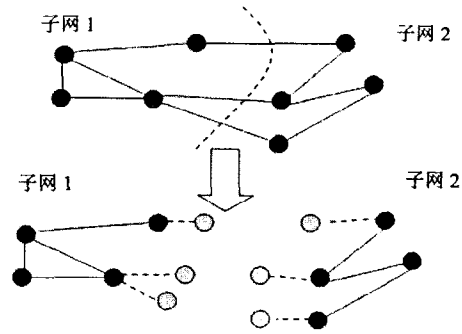

图 1　网络分割图

1.4　交通网络的简化

交通网络的简化主要分为两个方面:一是对交通网络中边的简化,在计算过程中减少边的数目;二是对节点的简化,缩小节点的数目。

在各个小区中,包含有很多节点,而且各个节点之间大都是被很短的路段相连接。为了减少节点数,在各自小区中只取有代表性的节点作为代表,参加最短路径的求解计算,其他节点相对来说是透明的,不在第一层参与计算。例如:行政部门、银行、邮局、广场等周围的路口作为代表节点。在选取这些节点时必须注意几个因素:①所选的代表点必须均匀分配在各自小区的各个地方,不能集中于某一处。②所选的节点必须保持一定的距离,不能离较短的距离。③所选的代表节点是人流量比较大地方,不能选取较偏僻的节点。

化简后的网络在宏观层上代表了原有大规模网络的某些特征,使用小区中与外界有边连接的节点作为小区的代表点,代表这个小区的所有节点,特别是在计算两点之间的最短路径时,由于节点数目大幅度减少,就降低问题求解的规模,提高计算效率。

2　分层并行求解的思想及步骤

在并行 PC 集群下,为了更好地提高效率,这里将划分后的每个子网即小区分配个不同的 PC 机上,各个小区的代表节点也被分配给不同的处理器。

2.1　并行 Dijkstra 算法描述

(1) 首先将节点进行划分,每个处理器划分 $n = S/p$ 个节点(其中 S 表示总的代表节点数,p 表示处理器的数目)。

(2) 每个处理器并行进行求解离起始点距离的局

部最小值,然后由其中一个处理器收集这些分布在其他处理器的局部最小值及相对应的节点编号,再由该处理器求出这些局部最小值中的较小值,即为全局最小值 min。记下最小值的位置,然后再将该最小值 min 及所对应节点的位置 index 广播给其他各个处理器。

(3) 各处理器根据这个全局最小值独自更新,更新如下:min + $W(v,\text{index}) < \text{dist}(v)$,则修改 $\text{dist}(v)$ 为:$\text{dist}(v) = \text{min} + W(v,\text{index})$。在更新过程中,由 path 数组存放离当前节点最近的前一节点的下标值(path 数组用于最后路径输出,$W(v,\text{index})$ 表示邻接矩阵的第 v 行第 index 列元素的值)。

(4) 重复(2)、(3) 步,直到所有的标记位都被标记为止。

与传统 Dijkstra 算法对比很容易发现,传统算法的时间复杂度为 $O(N^2)$,并行算法使用了 p 个处理器,其中每求出一个全局最小值要进行 p 个局部最小值的一次比较,所以一共要经过 S 次比较,因此该并行最短路径搜索算法的时间复杂度为 $O(S^2/p + S*(p-1))$。

2.2 分层思想

对划分好的交通网络进行最短路径求解时可以分为两层进行,其一:对化简后的小区及各自的代表节点之间求最短路径,此步是运用并行算法在 MPI 集群上实现的;其二:求各个小区内部的所有节点到代表节点的最短路径距离[7,8]。此步是在不同处理器中同时求解。经过两层的计算后再由 0 号 PC 机进行汇总,得出结果。

2.3 交通网络求解最短路径流程

在上述算法中由于分层后还要进行结合求出最终结果,此时可能会出现求解的距离起始或终止节点最短的代表节点在所有代表节点之间并不是最小值,因此在设计数据结构时加入了方向标志位,可以解决该问题。即起始和终止节点都朝着它们相互夹角为锐角的方向求解。图 2 为求解最短路径流程图。

2.4 算法效率分析

以上分析可知,从时间复杂度来看,网络划分计算的时间复杂度为 $O(T*E*N)$,其中 T 为在网络划分过程中所选的代表边的数目,E 为整个网络中的边数,N 为整个网络中的节点数。第二步利用并行 Dijkstra 算法时间复杂度为 $O((S^2/p + S*(p-1)) * S)$。其中 S 为代表节点的总数,$S = \sum_{i=1}^{k} s_i$(s_i 为第 i 个处理器上所分小区代表节点数)。第三步找起始和终止节点所在的子网时,时间复杂度为:$O(2*\max(n_i))$,其中 n_i 表示每个处理器上所分节点的数目,有 $N = \sum_{i=1}^{k} n_i$,在第四步由两台处理器同时用串行 Dijkstra 算法求解,时间复杂度为:$O((\max(n_i))^2)$,n_i 同上。所以可以得出总时间复杂度为 $O(T*E*N + (S^2/p + S*(p-1)) * S + 2*\max(n_i) + (\max(n_i))^2)$,式中,$E$ 和 N 的值较大,特别是 E 值,并且在并行求解多源点最短路径时所花费的时间较多,但在实际交通网络中,当网络固定时,只需要划分一次然后就可以一直计算。

从上面的分析中可以看出,并行计算代表节点之间最短路径也只需要计算一次,在以后的计算中,只需要计算两步:一是在网络中查找起始和终止节点所在的子网;二是计算起始和终止节点所在子网中到其他节点的最短路径。从而以后的时间复杂度降低为:$O(2*\max(n_i) + (\max(n_i))^2)$,相对于传统的 Dijkstra 算法时间复杂度 $O(N^2)$ 有了很大的提高。

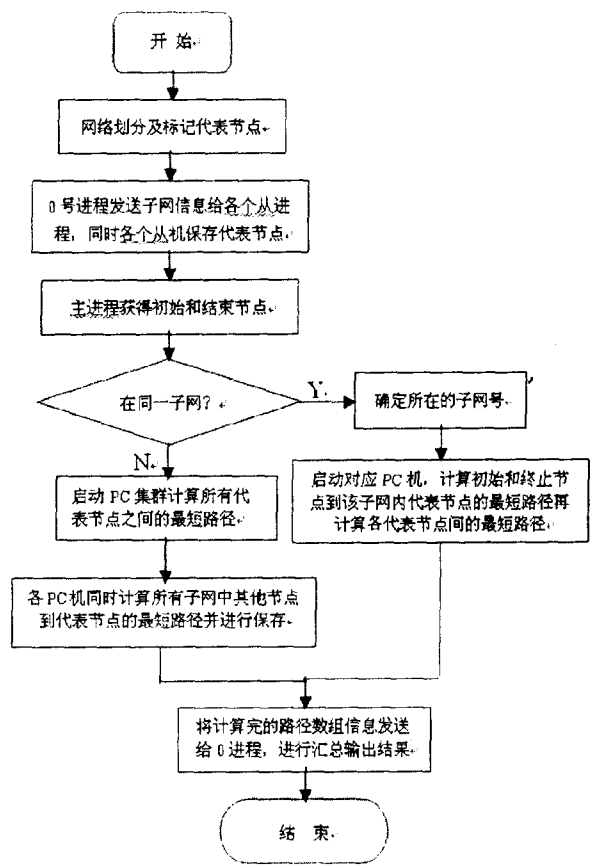

图 2 求解最短路径步骤流程图

从空间复杂度来看,并行环境的分布式存储有它自身的存储优势,这里不再赘述。

3 实际测试与结果

3.1 试验环境

MPI(Message Passing Interface)是目前很流行的

一个基于消息传递的并行编程工具。它具有移植性好、功能强大、效率高等优点,几乎所有的并行计算机厂商都提供对它的支持。它支持高效的通讯函数,支持 Fortran 和 C/C++语言编程[9,10]。鉴于 MPI 的强大通信功能,本试验是通过 5 台 PC 机(联想 P4 /2.0GHzCPU/256 MB)在 Windows 系统下构建一个集群来实现的。通过应用 Visual C++ 6.0 编译器,要将 MPICH 提供的 include 文件和 lib 文件分别添加到 Visual C++ 6.0 编译器相应的库中。其中,0 号处理器是作为主进程,用于存储各个代表节点的全源点最短路径,并且最后由 0 号处理器来进行最短路径的汇总输出。

3.2 试验说明及结果

对于交通网络的分割要消耗大多时间,但对于固定的网络只需要进行一次划分,也只需要计算一次全源点最短路径,之后当每次查询时只需要在各个小区中用 Dijkstra 算法求解出发和终止节点到代表节点的最短路径。这里的时间不包含网络划分的时间。在进行并行求解各源点的最短路径时,与局域网的性能和各个 PC 机的状态等因素密切相关,为了减少误差,这里对于每组数据都进行了 15 次测试,然后求它们的平均值。并且采用 MPI 自带的时间函数 MPI_Wtime 做差来进行计时的。

本实验是根据西安市道路路网电子地图数据进行测试的,最多有 6581 个节点,将地图分割成不同的节点数目进行不同的测试。每次划分后将数据从 orcal 数据库中读出存放到邻接表结构中。所用的应用软件是用 MapInfo MapXtreme 2005 6.6,当输入起点和终点编号时所求路径在地图回显出来,在小区内路线比较曲折,是采用 Dijkstra 算法在小区内进行搜索,而在小区与小区之间回显的路段级别比较高,相对比较直,也比较快捷。对于小区间的路径搜索算法是在将网络化简后的地图上运用并行的 Dijkstra 算法来求解的。

对网络进行分割后,对不同节点数和边数的时间测试结果如表 1 所示。

表 1　试验结果

时间(s)	节点数/边数					
	200	400	600	800	1000	2000
本方法第一次	0.546	0.759	1.089	1.654	2.174	3.776
本方法第二次	0.597	0.750	1.197	1.669	2.055	3.918
本方法第三次	0.584	0.824	1.096	1.409	2.119	3.659
本方法均值	0.576	0.778	1.128	1.580	2.116	3.784
串行 Dijkstra	0.145	0.599	1.514	1.947	2.947	4.421

与串行 Dijkstra 运行时间的对比曲线图如图 3 所示。

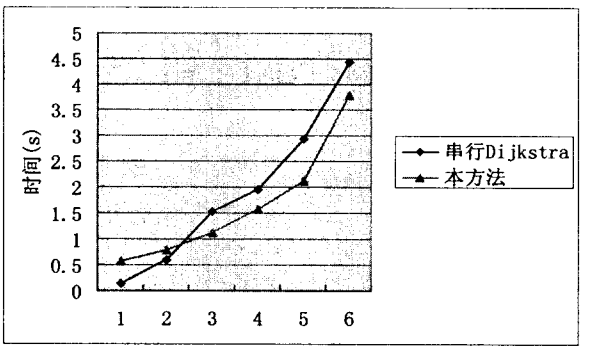

图 3　运行时间对比曲线图

从试验结果可以看出,当节点数不够多时,由于在不同 PC 机间进行计算,最后要将结果返回到 0 号 PC 机要消耗通讯时间,所以本方法所花费的时间要多,但随着节点数目的不断增大,该方法明显花费的时间变短,体现出优越性。

4 结束语

在大规模交通网络中,运用简化网络的方法和分层求解的思想,在试验中获得更加高效的实时性。并且该思想方法可以应用于所有的具有小区结构大规模网络中,具有一定的应用领域。

负载平衡是影响算法的一个很重要的因素,既要保证性能差的服务器不成为系统的瓶颈,也要保证性能高的服务器资源得到充分利用。下一步工作采用适当的平衡算法进行任务划分和迁移,进一步提高各个节点的效率和提高系统实时性。

参考文献:

[1] 马明全,周明全,耿国华,等.基于社区分析的最短路径计算[J].计算机应用与软件,2008,25(4):177-180.

[2] Kmchandy, Jmisra. Distributed computation on graphs: shortest path algorithms[J]. Communications of the ACM, 1982, 25(11):45-47.

[3] Paige R C, Cpkruskal. Parallel algorithms for shortest path problems[C]//International Conference on Parallel Processing (ICPP'85). University Park, PA, USA: IEEE Computer Society Press, 1985:442-448.

[4] Mhabbal, Hkoutsopoulos, Slerman. A composition algorithm for the all pairs shortest path problem on massively parallel computer architectures[J]. Transportation Science, 1994, 28(3):26-33.

[5] 隽志才,高林杰,倪安宁.面向对象的交通网络分布式仿真并行数据结构[J].交通与计算机,2006,24(1):36-39.

[6] 倪安宁.交通网络分析中的最短路径并行算法研究与实现[D].长春:吉林大学交通学院,2004.

[7] 沈项军.基于子网络结构属性的网络分割研究[J].计算机

(下转第 89 页)

运行时间均少于 PeerTrust。而且随着总运行步数的增加,BTrust 的运行时间仅有略微上升,而 PeerTrust 的运行时间由于其本身管理机制的复杂性而迅速上升。因此,BTrust 相对于 PeerTrust 有计算量小,能满足普适计算环境下小型设备计算能力受限的优点。

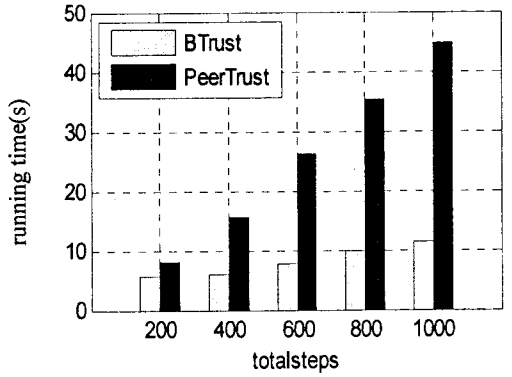

图 3　运行时间的比较

节点存储节点的直接服务信任值、推荐服务信任值、推荐信任值,以及维持邻居节点列表和查询节点列表需要一定的存储空间。可以从存储开销上比较 BTrust 和 PeerTrust 两种信任管理模型。仿真参数保持不表,仅改变总运行步数。

仿真结果如图 4 所示。

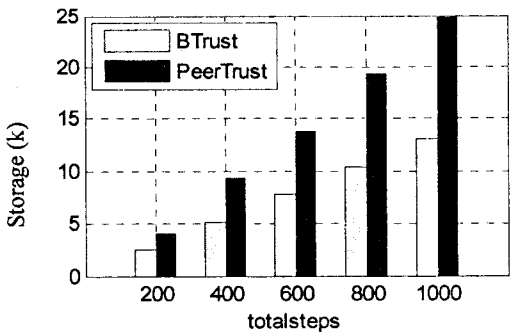

图 4　存储开销的比较

从图 4 中可看出,随着总运行步数的增加,两种信任管理模型的存储开销均在增加,但 BTrust 所需的存储空间明显少于 PeerTrust,能适应普适计算环境下小型设备存储空间有限的特点。

5　结束语

普适计算环境中存在多个小型嵌入式设备,是一种资源受限的环境。在该环境中建立信任管理模型,需考虑该环境中小型设备计算能力和存储空间有限的特点。BTrust 通过直接服务信任和推荐信任的计算建起了该环境下轻量级的信任管理模型。仿真结果表明,BTrust 能有效地扼制恶意节点的攻击,提高服务成功率,并在运行时间、计算量和存储开销上有明显的优势,能有效地运行在普适计算环境中。文中的模型仅在理论上分析了抵御信任攻击的能力,并未对此进行全面的仿真,而且未加入对信任值的风险评估机制,这将是下一步研究任务。

参考文献:

[1] Weiser M. The computer for the twenty - first century[J]. Scientific American, 1991, 265(3): 94 - 104.

[2] 徐光佑,史元春,谢伟凯. 普适计算[J]. 计算机学报,2003, 26(9): 1042 - 1052.

[3] 徐文拴,辛运帏,卢桂章. 普适计算环境下信任机制的研究进展[J]. 计算机科学,2008,35(2): 52 - 57.

[4] 荆　琦,唐礼勇,陈　钟. 无线传感器网络中的信任管理[J]. 软件学报,2008,19(7): 1716 - 1730.

[5] Kamvar S D, Schlosser M T, Molina H G. The Eigentrust algorithm for reputation management in P2P networks[C]// In: Proceedings of the 12th International World Wide Web Conference. Budapest: ACM Press, 2003: 640 - 651.

[6] Jøsang A, Ismail R. The beta reputation system[C]//In: Proceedings of the 15th Bled Electronic Commerce Conference. Bled, Slovenia: [s. n.], 2002.

[7] Xiong Li, Liu Ling. PeerTrust: Supporting Reputation - Based Trust for Peer - to - Peer Electronic Communities[J]. IEEE Transaction on Knowledge and Data Engineering, 2004, 16 (7): 843 - 857.

[8] Liang Zhengqiang, Shi Weisong. Analysis of Recommendations on Trust Inference in Open Environment[J]. Journal of Performance Evaluation, 2008, 65(2): 99 - 128.

[9] Florina A, Andres M, Daniel D, et al. Developing a model for trust management in pervasive devices[C]//Proceedings of Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops 2006. Pisa, Italy: Institute of Electrical and Electronics Engineers Computer Society, 2006: 267 - 271.

(上接第 85 页)

工程与应用,2007,43(23): 64 - 68.

[8] 章昭辉,闫春钢,丁志军. 交通信息网络中的最短出行路径并行算法[J]. 同济大学学报:自然科学版,2006,34(12): 1606 - 1611.

[9] 陈国良. 并行算法实践[M]. 北京:高等教育出版社,2004: 389 - 400.

[10] Gtama A, Gupta A, Karypis G, et al. Introduction to Parallel Computer[M]. 张　武,等译. 北京:机械工业出版社,2005: 321 - 323.