蚁群算法解决TSP问题的并行化研究与实现

张 军,刘 羽,卢奉良
(桂林理工大学 信息科学与工程学院,广西 桂林 541004)

摘 要: 蚁群算法在处理大规模TSP(Traveling Salesman Problem)问题时耗时较长,为了解决这一不足,给出一种基于多核环境下的并行优化算法。采用OpenMp并行优化技术对蚁群算法中最为耗时的循环迭代和循环赋值部分进行改进,减少其运算时间,同时利用粗粒度并行策略和PC机多核的优势将具有一定规模的小蚁群分配到对应的处理器上,使其并行执行,并且在适当时机让各处理器上的蚁群进行相互间的通信。通过实验证明,改进后的并行蚁群算法程序执行时间明显缩短,执行效率显著提高。由此可见,改进后的并行蚁群算法是可行有效的。

关键词: 蚁群算法; TSP问题; 多核; OpenMp; 并行优化
中图分类号: TP39 **文献标识码:** A **文章编号:** 1673-629X(2011)05-0072-03

Parallel Research and Implementation of Ant Colony Algorithm to Solve Problem of TSP

ZHANG Jun,LIU Yu,LU Feng-liang
(School of Information Science and Engineering,Guilin University of Technology,Guilin 541004,China)

Abstract: In order to resolve the disadvantage that ant colony algorithm solves large scale TSP(Traveling Salesman Problem) consuming a large amount of time,give a parallel optimization algorithm at multi-core environment. Applying the technology of parallel optimization about OpenMp improves the part of iteration and cyclic assignment in ant colony algorithm,because this part consumes the most of time. At the same time using Coarse-grained parallel strategy and multi-core's advantage assign a certain amount of small-colony to the corresponding processor, then makes it executed parallelly and communicated with each other at the appropriate time. The experiment proves that the improved method makes the time of program execution shorter significantly and the efficiency higher observably when solve large scale TSP. This shows that the improved ant colony algorithm in parallel is feasible and effective.

Key words: ant colony algorithm; the problem of TSP; multi-core; OpenMp; parallel optimization

0 引 言

随着互联网技术的发展,网络上信息量越来越大,这对计算机的处理能力和处理速度也提出了更高的要求,在多核PC机下采用OpenMp[1]并行编程模式进行并行运算可以充分发挥多核的优势,将计算任务划分然后分配给多个核并行处理,大大节省了计算时间,提高效率;而且在所需资源方面不像MPP需要大型计算机或是像COW需要大量PC机组成机群进行实验,节省成本。所以,在多核处理器下使用OpenMp编译指令对计算量较大的一些算法进行优化很有意义。

旅行商问题(Traveling Salesman Problem,TSP)是一个NP完全问题,TSP问题是组合优化领域中的一个典型问题。目前蚁群算法[2]求解TSP问题的并行化主要是在机群环境下基于MPI[3]来实现,而文中将在多核环境下采用基于OpenMp的多线程编程技术,对传统蚁群算法进行一些改进使其并行运行,在求解TSP问题时能最大程度上利用CPU资源,提高工作效率。

1 解决TSP问题的蚁群算法

1.1 算法原理

蚁群算法的基本原理吸纳了自然界实际蚁群行为的一些重要特点。TSP问题是指对于给定的n个城市的集合$(1,2,\cdots,n)$,找到一条经过每一个城市一次且最后回到起点的最短封闭环路[4]。TSP问题的目标函数是:

$$\text{Min } A = \sum_{i=1}^{n-1} d(i,i+1) + d(n,1) \quad (1)$$

其中$d(i,j)$ $(i,j=1,2,\cdots,n)$表示城市i和j之间的距离。

收稿日期:2010-09-26;修回日期:2010-12-25
基金项目:广西自然科学基金(桂科自0832249)
作者简介:张 军(1984-),男,湖北监利人,硕士研究生,研究方向为并行计算;刘 羽,教授,博士,研究方向为并行计算、计算机网络。

基本蚁群算法可以描述如下:

(1) 起初将 m 只蚂蚁随机放在 n 个城市上,$\tau_{ij}(0)$ 为城市间的每一条边上的初始化信息素,蚂蚁 k 当前所经过的城市信息保存在禁忌表 $\text{tabu}_k(k=1,2,\cdots,m)$ 中,每只蚂蚁的禁忌表 $\text{tabu}_k(s)$ 的第一个元素设置为其初始城市。

(2) 然后每只蚂蚁开始选择下一步要访问的城市。在搜索过程中蚂蚁依据概率函数:

$$p_{ij}^{k}=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}\times[\eta_{ik}(t)]^{\beta}}{\sum\limits_{s\subset \text{allowed}_k}[\tau_{is}(t)]^{\alpha}\times[\eta_{is}(t)]^{\beta}} & \text{若 } j\subset \text{allowed}_k\\ 0 & \text{否则}\end{cases} \tag{2}$$

选择将要到达的城市,该概率由城市间的距离和信息素的强度决定。其中 $\tau_{ij}(t)$ 表示 t 时刻城市 i 到城市 j 边上信息素的强度(i 为出发城市,j 为到达城市);η_{ij} 表示城市间距离因子,通常取值为城市 i 到城市 j 之间距离的倒数 $1/d_{ij}$;α 表示信息素在选择概率上的作用;β 是指路径长度在选择概率上的作用[5]。

(3) 经过 n 次循环后,蚂蚁完成对所有城市的一次遍历,所有蚂蚁的禁忌表都已填满,此时计算每只蚂蚁所经过的路径长度,通过比较后找到最短的那条路径并且保存下来,同时对本次遍历所经过的路径进行信息素的更新。重复这一过程直至达到最大周游值时结束。

信息素更新公式为:

$$\tau_{ij}(t+n)=(1-\rho)\cdot\tau_{ij}(t)+\Delta\tau_{ij}(t) \tag{3}$$

$$\Delta\tau_{ij}(t)=\sum_{k=1}^{m}\Delta\tau_{ij}^{k}(t) \tag{4}$$

其中 $\Delta\tau_{ij}$ 表示城市 i 到城市 j 边上的累加新增信息素之和,ρ 表示信息素挥发系数,为了防止信息的无限积累,ρ 的取值范围设置为 $0\leqslant\rho<1$,而 $1-\rho$ 则表示信息素残留因子。$\Delta\tau_{ij}^{k}$ 表示第 k 个蚂蚁本次循环中在 t 和 $t+n$ 之间城市 i 到城市 j 边上遗留下的信息量。常用到的计算模型为:

$$\Delta\tau_{ij}^{(k)}(t)=\begin{cases}\dfrac{Q}{L_k} & \text{若第 } k \text{ 只蚂蚁在本次循环中经过城市 } i \text{ 和 } j \text{ 之间}\\ 0 & \text{否则}\end{cases} \tag{5}$$

其中,Q 为常数,算法的收敛速度在一定程度上受到 Q 的影响; 第 k 只蚂蚁在本次循环中所经过的路径长度用 L_k 来表示[6]。

1.2　蚁群算法实现步骤

具体的实现步骤如下:

(1) 参数初始化。首先设时间 t 和循环次数 Nc 都为0,$Nc_{\max}$ 表示最大循环次数,在 n 个城市上随机放置 m 只蚂蚁,城市间每条边(i,j) 的初始信息量 $\tau_{ij}(t)$ 设置为常数 const,并且令 $\Delta\tau_{ij}(0)=0$;

(2) 循环次数迭加 $Nc\leftarrow Nc+1$;

(3) 用 k 表示蚂蚁的禁忌表索引号且设初值为 1;

(4) 蚂蚁数目迭加 $k\leftarrow k+1$;

(5) 根据状态转移概率公式计算出的概率(式 2),蚂蚁个体选择下一个将访问的顶点(城市)j,$j\in\{0,1,\cdots,n-1\}-\text{tabu}_k$;

(6) 选择好下一个顶点后蚂蚁移动到新的顶点(城市),并把该顶点(城市)移动到蚂蚁个体的禁忌表中,修改禁忌表指针;

(7) 若 $k<m$,即未遍历完所有顶点(城市),则跳转到步骤(4),否则执行步骤(8);

(8) 根据公式(3)、(4)更新每条路径上的信息量;

(9) 若满足循环次数 $Nc\geq Nc_{\max}$ 的结束条件则结束循环且输出计算结果,否则清空禁忌表并跳转到步骤(2)[7]。

2　蚁群算法的并行化

蚁群算法本质上是一种并行的算法,具有极高的并行性。每只蚂蚁遍历各个城市的过程也就是构建问题可行解的过程,所有蚂蚁的遍历过程彼此独立,在遍历过程中并行构建问题的可行解。每只蚂蚁构建解的过程只与当前的信息激素和启发函数有关,只有当所有蚂蚁均完成了可行解的构建后,才开始蚂蚁间的通信。因此可以将各个蚂蚁遍历城市的过程分配到不同的线程中并行运行,这样可以充分利用计算机多核的资源优势,在基本不改变串行蚁群算法的基础上实现了算法的并行化,保持了原有的设计结构,并且采用串行计算和并行计算相结合的方法对蚁群算法中的信息素进行更新。

并行蚁群分为细粒度策略和粗粒度策略,细粒度策略是指将每只蚂蚁都分配到一个处理器上,然后使其并行运行; 粗粒度策略是指一个小蚁群分配到一个处理器上让其并行,适当时机彼此之间相互通信。虽然细粒度策略更接近自然蚁群,但只适用于大规模处理器阵列等并行机,且通信量较大,代价较高,研究表明粗粒度的并行蚁群更有潜力[8~12]。

蚁群算法主要的耗时部分在于循环迭代和循环赋值,特别是当蚁群算法在解决大规模 TSP 问题时这部分所消耗的时间明显增加,并且这些循环迭代和循环赋值部分相对独立,不存在循环依赖或循环依赖较弱,所以文中主要优化蚁群算法中最为耗时的循环迭代和循环赋值。

并行化蚁群算法的实现步骤如下(以蚁群算法在双核机上实现并行化为例加以说明):

1) 初始化部分:

(1) 初始化各项参数,输入各个城市的坐标;

(2) 设置每条路径上的信息素强度;

(3) 根据 PC 机核的个数设置线程数为 P($P = 2$) ,用于将一个蚁群分为 2 个小蚁群在多核机上并行求解;

(4) 各个蚁群中的每只蚂蚁被随机地放置在任一城市;

(5) 初始化信息素矩阵 τ_{ij} 且初始时刻每条边上的信息素增量设置为 0;

(6) 初始化用于存放最优解的 BestSolution 矩阵。

2) 循环开始:

(1) 进入并行区,用#pragma parallel omp for 语句对 for(int $k = 0$; $k < M$; $k + +$) 循环进行并行化,将原始蚁群分为 2 个小蚁群(记为 P_1 和 P_2) ,分别交给 2 个处理器处理使其在多核机上并行求解。

(2) 根据状态转移概率公式计算的概率(式 2) ,各蚁群中蚂蚁个体选择下一个将访问的顶点(城市) j 并前进, $j \in \{0,1,\cdots,n-1\} - \text{tabu}_k$,选择好之后将该蚂蚁移动到新的顶点(城市) ,并把该顶点(城市) 移动到该蚂蚁个体的禁忌表中;

(3) 在一次循环中各个蚁群中的每只蚂蚁完成一次周游,它们的禁忌表被填满,计算出蚁群 P_1 和 P_2 中的每只蚂蚁所走过的路径长度 L_k 并且比较它们的大小求出 P_1 和 P_2 中的最短路径 $\min L_1$ 和 $\min L_2$,将 $\min L_1$ 和 $\min L_2$ 保存到 BestSolution 矩阵中;

(4) 并行区结束,由公式(5) 对每条路径上的信息素增量 $\Delta\tau_{ij}$ 进行更新;

(5) 根据公式(3) 、(4) 更新每条路径上的信息量;

(6) 将所有蚂蚁的禁忌表清空;

(7) 一次循环结束,循环次数 Nc 加一,进入下次循环;

(8) 上述过程不断循环重复,最优路径上的信息素得到加强,直到达到循环结束条件,若蚁群 P_1 和 P_2 中的所有蚂蚁沿同一路线运动则认为蚁群 P_1 和 P_2 找到了各自的最优路径 $\min L_1$ 和 $\min L_2$;

(9) 比较 $\min L_1$ 和 $\min L_2$ 的大小选出其中较小的一个作为全局最优路径 W 。

3) 算法结束条件:

只要当算法满足下面其中任意条件时算法结束:

(1) 搜索出最优解;

(2) 循环次数达到最大;

(3) 搜索时间达到最长;

(4) 经过预设循环次数后全局最优解仍然没有改变。

3 实验结果及分析

试验硬件环境为 AMD Athlon(tm) 64 位酷睿双核的微机,内存为 2G; 软件环境为支持 SMP(对称多处理器系统) 的 Window XP; 编译环境为 VC + + 2008。本实验以 1000 只蚂蚁遍历 50 个城市为例,试验结果为程序多次运行结果的平均值。

对各参数分别设如下值进行实验: $\alpha = 1$,$\beta = 5$,$\rho = 0.5$,$Nc_{\max} = 300$,$Q = 100$ 。取有代表性的结果如表 1 所示。

表 1 改进前后测试结果对照表

核个数	最优路径和	执行时间(s)	加速比
1	196.652	1692	…
2	190.340	945	1.79
4	194.471	486	3.48
8	193.527	292	5.79

通过大量的实验发现各参数影响如下:

α 决定信息量对路径选择的影响程度,β 决定路径的长度对选择的影响程度。经过试验发现当 $\alpha = 1$,$\beta \in \{1,2,5\}$ 时可以取得较好解; Q 的值对结果并没有多大的影响; 循环次数越多,最优化结果越好,但是达到 1000 次以后,再提高循环次数就没有太多的影响。

从表 1 和图 1 可以看出经改进后的并行蚁群算法在双核 PC 机上计算时,计算公式 S = 单处理器下的算法运行时间/多核系统中算法运行时间可得其加速比 S 为 1.79,运算时间比在单核 PC 机上计算所用时间少了将近一半; 在四核 PC 机上计算时加速比为 3.48,程序运行时间又比在双核上运行的时间少了将近一半; 在八核 PC 机上的计算所需时间又将近是四核 PC 机上运行时间的一半,接近线性加速比。在实验过程中还可以发现,在单核 PC 机上串行运算时 CPU 的利用率只有 51%,而在多核 PC 机上并行运算时 CPU 的利用率达到了 100%,表明计算机的两个核都被充分利用了。

由上可知经改进后的蚁群算法在解决 TSP 问题时充分发挥多核的优势将计算任务分配到多个核中并行运行,节约了计算时间,提高了效率。

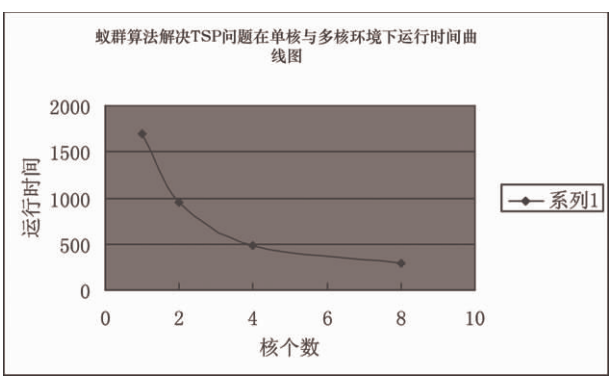

图 1 蚁群算法解决 TSP 问题在单核与多核环境下运行时间曲线图

(下转第 78 页)

语句搜索模块: 当形成词组时,需要对形成的词组进行输出以供用户选择,为了提高文字输入效率,需将常用词组排列在前面,此时可运用动态规划中的 Viterbi 算法,统计系统内的各个词组频率值,按照该值进行排列输出。

学习模块: 当用户输入修正后的词组或语句时,该模块会自动查询该语句是否已在字典中,若不在,可进行记忆学习,将该语句添加到自定义词典中,提高输入法输入的效率。

5 结束语

文中首先分析了 Windows Mobile 平台下智能手机输入法系统的构成和输入法用户接口,并设计了输入法的编码方案和字词库的创建与匹配,然后对实现输入法的接口原理进行了详细分析,最后给出了笔画输入法实现原理,从此可看出,本输入法可使用户连续进行笔画输入汉字,减少了停留时间和重码率等,提高了输入效率[12]。

随着 3G 时代的来临和智能手机用户数量的增多,它提供的强大娱乐功能逐渐丰富了人们的生活,而目前在 Windows Mobile 平台下笔画输入效率低下,且不具有词组以及语句输入。因此文中提出的笔画输入法设计模式对于智能手机输入技术的研究具有重要的意义。

参考文献:

[1] 欣 闻. 手机文字输入技术及其发展趋势[J]. 现代通信,2005(2):34 - 35.

[2] 傅 曦,齐 宇,徐 俊. Windows Mobile 手机应用开发[M]. 北京: 人民邮电出版社,2005.

[3] 李培峰,朱巧明,钱培德. 一个应用于手持设备的汉字通用输入模型[J]. 计算机工程,2006,32(18):258 - 260.

[4] 乔建良,赵增建. 谈谈手机的中文输入法[J]. 现代通讯,2004,12(3):41 - 42.

[5] 张 晋. 汉字信息处理研究[M]. 北京: 北京语言学院出版社,1992:4 - 21.

[6] Po Lai - Man, Wong Chi - Kwan. Six - Digit Stroke - based Chinese Input Method[C]// Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics. San Antonio, TX, USA: [s. n.],2009:818 - 823.

[7] 吴海辉. 笔画码汉字输入法[D]. 合肥: 安徽大学,2004.

[8] 高 升,王晓龙. 语句级汉字输入系统中语义规则研究[J]. 计算机工程与应用,2003(4):80 - 82.

[9] Microsoft. Win32 Multilingual IME Application Programming Interface[M]. [s. l.]: [s. n.], 2003.

[10] Microsoft Corp. Win32 Multilingual IME Application Programming Interface[M]. [s. l.]: [s. n.],1998.

[11] 徐志明,王晓龙,姜守旭. 一种语句级汉字输入技术的研究[J]. 高技术通讯,2000(1):51 - 55.

[12] 李 政. 计算机汉字输入技术的现状和发展趋势[J]. 松辽学刊,1999,12(2):33 - 36.

(上接第 74 页)

4 结束语

文中给出一种基于多核环境下蚁群的并行优化算法,利用多核的优势并行处理蚁群算法中耗时较长的循环迭代和循环赋值部分。实验结果表明经过改进后的蚁群算法在多核环境下解决大规模 TSP 问题时所需运行时间明显少于单核机上的运行时间,程序执行效率明显提高。

参考文献:

[1] OpenMp C and C + + Application Program Interface (Version 2.0) [EB/OL]. 2002. http://www.openmp.org.

[2] 段海滨. 蚁群算法原理及其应用[M]. 北京: 科学出版社,2005:34 - 37.

[3] 付延友. PC 机群环境下蚁群算法的并行化研究[D]. 天津: 河北工业大学,2007:29 - 43.

[4] Stutzle T, Hhoos H. The MAX - MIN ant system and local search for the traveling salesman problem[C]// Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC'97). USA: Indianapolis,1997:309 - 314.

[5] 王 娟,王 建. 一种求解 TSP 问题的改进蚁群算法[J]. 计算机技术与发展,2008,18(12):50 - 52.

[6] 崔明义,张新祥,苏白云,等. 用蚁群算法实现地理信息系统空间曲线的描述[J]. 计算机工程与应用,2008,44(30):160 - 162.

[7] 赖金富. GIS 和蚁群算法及其在城市交通分配中的应用研究[D]. 昆明: 昆明理工大学,2008:22 - 23.

[8] Stützle T. Parallelization strategies for ant colony optimization[J]. Lecture Notes in Computer Science,1998,1498:722 - 741.

[9] Marcus, Randall, Andrew, et al. A parallel implementation of ant colony optimization[J]. Journal of Parallel and Distributed Computing,2002,62:1421 - 1432.

[10] Ellabib I, Calamai P, Basir O. Exchange Strategies for Multiple Ant Colony System[J]. Information Sciences: An International Journal,2007,177(5):1248 - 1264.

[11] Middendorf M, Reischle F, Schmech H. Multi Colony Ant Algorithms[J]. Journal of Heuristics: Special Issue on Parallel Metaheuristics,2002,8(3):305 - 320.

[12] 何丽莉,王克淼,白洪涛,等. 基于 CMP 的多种并行蚁群算法及比较[J]. 吉林大学学报(理学版),2010,48(5):787 - 792.