基于 OpenMP 求解 QAP 的并行粒子群优化算法

周洪斌

(沙洲职业工学院 电子信息工程系,江苏 张家港 215600)

摘 要:提出了一种基于 OpenMP 求解 QAP 的并行粒子群优化算法。该算法将遗传算法的交叉策略引入 PSO 算法中,同时采用禁忌搜索算法作为局部搜索算法。在 QAPLIB 实例上的测试结果表明,并行 PSO 算法在所有测试实例上都获得了超线性加速比,且运行结果优于串行算法。

关键词:粒子群优化算法;二次分配问题;OpenMP

中图分类号:TP301 **文献标识码**:A **文章编号**:1674-7720(2010)10-0084-04

OpenMP-based parallel particle swarm optimization algorithm to solve quadratic assignment problem

ZHOU Hong-Bin

(Electronic Information Engineering Department, Shazhou Professional Institute of Technology, Zhangjiagang 215600, China)

Abstract: This paper puts forward an OpenMP-based parallel particle swarm optimization algorithm to solve QAP. The new algorithm combinies with the crossover operation of the genetic algorith and uses tabu search algorithm as its local search. Experiments are performed on QAP instances from QAPLIB. The results show that parallel PSO obtains superlinear speed ratio and produces better results than serial algorithm.

Key words: particle swarm optimization; quadratic assignment problem; OpenMP

粒子群优化算法起源于鸟类群体智能,是一种基于群体的新型随机元启发搜索算法。自 1995 年提出以来,引起了广大研究者的关注,成为研究热点,并已在连续函数优化、人工神经网络训练、数据聚类等问题中得到了成功的应用。粒子群优化算法中初始解产生、目标函数适应值计算、粒子飞行、更新自身位置,是最为费时的部分,并且初始解产生、目标函数适应值计算、粒子飞行、更新自身位置是相互独立的,因此粒子群优化算法具有天然的并行性。OpenMP 是共享存储平台上的一种高效的并行程序实现手段,可以使用编译指导的方式把串行程序转化为并行程序。采用 OpenMP 共享存储的并行方式,既可以大大减少通信代价,又可以充分利用并行物理系统带来的高性能。

1 标准粒子群优化算法简介

粒子群优化算法 PSO(Particle Swarm Optimization)是由美国的 Kennedy 和 Eberhart 在 1995 年提出的[1]。在 PSO 中,每个优化问题的解可看作是搜索空间中的一只鸟,称之为粒子。粒子的位置代表优化问题在搜索空间中的潜在解,粒子的速度决定飞行的方向和距离,所有的粒子都有一个被优化的函数决定的适应值。假设在一个 D 维的目标搜索空间中,m 个粒子组成一个群落。第 i 个粒子的位置表示为矢量 $X(x_1, x_2, \cdots, x_D)$,飞行速度表示为 $V(v_1, v_2, \cdots, v_D)$。每个粒子通过跟踪两个"最好位置"来更新自己,一个是粒子本身目前所找到的最好位置(pbest),另一个是目前整个群体中所有粒子发现的最好位置(gbest),gbest 是在 pbest 中的最好值。对于第 k 次迭代,每个粒子是按照式(1)、式(2)进行变化的。

$$v_{id}^{k+1} = w \times v_{id}^{k} + c_1 \times rand() \times (p_{id} - x_{id}^{k}) + c_2 \times rand() \times (p_{gd} - x_{id}^{k}) \quad (1)$$

$$x_{id}^{k+1} = x_{id}^{k} + v_{id}^{k+1} \quad (2)$$

式(1)、式(2)中:$i=1, 2, \cdots, m$;v_{id}^{k} 为第 k 次迭代粒子 i 飞行速度矢量的第 d 维分量;x_{id}^{k} 为第 k 次迭代粒子 i 位置矢量的第 d 维分量;p_{id} 为粒子 i 个体最好位置 pbest 的第 d 维分量;p_{gd} 为群体最好位置 gbest 的第 d 维分量;c_1、c_2 为学习因子;w 为惯性权重,为非负数;$rand()$ 为随机函数,产生[0,1]的随机数。

2 OpenMP 概述

OpenMP 的应用程序接口(API)是在共享存储体系结构上的一个编程模型，它可以使应用程序在对称多处理器或多核系统上并行执行而获得性能的大幅度提升[2]。OpenMP 包含编译指导(Compiler Directive)、运行函数库(Runtime Library)和环境变量(Environment Variables)。OpenMP 是一个编译器指令和库函数的集合，这些编译器指令和库函数主要用于创建共享存储器计算机的并行程序。目前，支持 OpenMP 的语言主要有 Fortran、C/C++。

OpenMP 是基于线程的并行编程模型，使用 Fork-Join 并行执行模型。一个 OpenMP 程序从单个线程开始执行，在程序的某些点需要并行执行时，程序派生出额外的线程，组成一个线程组。这些线程在一个称为并行区域的代码区中并行执行。线程到达并行区域的末尾时等待，直到整个线程组都到达，然后它们连接在一起，只有初始或者主线程继续执行，直到下一个并行区域(或者程序结束)。

3 QAP 介绍

二次分配问题 QAP(Quadratic Assignment Problem)是一个经典的组合优化问题，目的是寻找 n 个设备到 n 个位置的最优布局。该问题具有重要的理论与实用价值，许多实际问题，如大学校园中建筑物的布局、医院中科室的安排、键盘布局问题等都可以转化为 QAP 来解决。其数学描述为：给定 n 个设备和 n 个位置，两个 $n\times n$ 的矩阵 $\boldsymbol{A}=[a_{ij}]$ 和 $\boldsymbol{B}=[b_{rs}]$，其中 a_{ij} 是设备 i 与 j 之间的信息流，b_{rs} 是位置 r 与 s 之间的距离。QAP 问题就是要寻找一个 $(1,2,\cdots,n)$ 的一个排列，使得总费用 f 最小。f 的定义为：

$$f=\sum_{i=1}^{n}\sum_{j=1}^{n}a_{ij}b_{M(i)M(j)} \tag{3}$$

其中，$M(i)$ 指在当前解中，设备 i 所在的位置；$a_{ij}\times b_{M(i)M(j)}$ 表示同时将设备 i 放到位置 $M(i)$ 和设备 j 放到位置 $M(j)$ 后所需要的费用。

QAP 是一个 NP-Hard 问题[3]，无法寻找求解 QAP 的多项式时间算法。一般，当问题规模 $n>20$ 时就很难找到最优解。为了实际可行地解决二次分配问题，人们不得不采用元启发式算法，以便在较短的时间内求得较优解。

4 基于 OpenMP 的并行粒子群优化算法

由于目前对 PSO 算法的研究主要集中在连续型的 PSO 算法，即描述粒子状态及运动规律的量都是连续的，要将其应用于解决组合优化问题，存在一定的困难。为此，将遗传算法的交叉操作及禁忌搜索算法引入粒子群优化算法中，采用混合算法求解二次分配问题。将式(1)中的 $c_1\times rand()\times(p_{id}-x_{id}^{k})+c_2\times rand()\times(p_{gd}-x_{id}^{k})$ 看做是遗传算法的交叉操作，让当前解与个体极值和全局极值分别作交叉操作产生新的解[4]。而由于是离散优化问题，式(1)中的 $w\times v_{id}^{k}$ 不再需要；将式(2)用禁忌搜索算法替代，对交叉操作产生的新解用禁忌搜索算法进行局部寻优，产生的解为粒子新的位置。

对于 QAP 问题而言，是一个解为 $(1,2,\cdots,n)$ 的排列。为了避免交叉后产生重复的基因码，采用如下的交叉方法：

(1)从第二个串 old2 中随机选择一个交叉区域；

(2)将 old2 的交叉区域加到 old1 前面，删除 old1 中已在 old2 的交叉区域中出现过的数字。

例如两父串为：

old1=12 345 678，odl2=87 654 321

假设交叉区域为 654，则交叉后的串为 65 412 378。

基于 OpenMP 的并行粒子群优化算法描述如下：

①设定粒子数 iPSONum，规定迭代次数 iStep

②#pragma omp parallel for

For each particle

初始化，随机产生 iPSONum 个初始解 dpos。根据 QAP 问题的解的特点，粒子位置 dpos 采用顺序编码方式，即 $(1,2,\cdots,n)$ 的一个排列；

End

③Do

#pragma omp parallel for

For each particle

根据式(3)计算当前位置的适应值 m_dFitness，设置当前适应值为个体极值 m_dBestfitness，当前位置为个体极值位置 dpbest；

End

根据各个粒子的个体极值 m_dBestfitnesst，找出全局极值 gBestFitnesst 和全局极值位置 gbest#pragma omp parallel for；

For each particle

第 j 个粒子位置 dpos 分别与 gbest、dpbest 交叉得到新的 dpos；

用禁忌搜索算法对 dpos 进行局部寻优；

End

根据各个粒子的个体极值 m_dBestfitness，找出全局极值 gBestFitnesst 和全局极值位置 gbest；

While 迭代次数 < 规定迭代次数 iStep

④输出全局极值 gBestFitnesst 和全局极值位置 gbest。

算法对初始解产生、目标函数适应值计算、粒子飞行、更新自身位置采用了 OpenMP 并行化技术。

5 实验结果

实验中采用 QAPLIB(http://www.seas.upenn.edu/qaplib/)数据作为测试实例。实验环境为 Intel Centrino Duo T2300E(双核)、内存 1 GB、操作系统为 Windows XP SP2、开发软件为 Visual Studio .Net 2005 SP1。粒子群优化算法的最大迭代步数为 10，群体大小为 10，每个粒子调用

局部搜索算法的次数为 $10\times n$。

本文选取了 QAPLIB 4 种类型中典型的实例进行测试，每个实例独立运行 10 次后取与最优解的平均偏离程度 D_{avg}。D_{avg} 定义为：

$$D_{avg}=[(Z_{avg}-Z_{bks})/Z_{bks}]\times 100\% \quad (4)$$

其中，Z_{avg} 是独立运行 10 次后的平均最优值，Z_{bks} 是来自 QAPLIB 的最优值。测试结果如表 1 所示。表 1 中第 4 列、第 5 列分别为串行、并行算法独立运行 10 次后按照式(4)求得的平均偏离程度 D_{avg}。

并行算法的加速比和效率如表2 所示。

结果分析表明，采用 OpenMP 共享存储的并行方式，大大减少了通信代价，充分利用了并行物理系统带来的高性能，对所有测试实例都获得了超线性加速比，且取得了比串行算法更好的运行结果。实验结果证明了基于 OpenMP 的并行算法的有效性。

本文提出一种用于求解 QAP 的

表 1 串、并行算法在 QAPLIB 上的运行结果对比

QAP 实例	问题规模	目前最优解	串行算法 D_{avg}	并行算法 D_{avg}
Unstructed, randomly generated instances(ⅰ)				
tai20a	20	703 482	0.202	0.061
tai25a	25	1 167 256	0.736	0.726
tai30a	30	1 818 146	0.602	0.411
tai40a	40	3 139 370	0.909	0.901
tai50a	50	4 938 796	1.300	1.305
tai60a	60	7 205 962	1.319	1.312
tai80a	80	13 515 450	1.355	1.338
Grid-based distance matrix(ⅱ)				
nug20	20	2 570	0	0
nug30	30	6 124	0	0
sko42	42	15 812	0.019	0.014
sko49	49	23 386	0.115	0.081
sko56	56	34 458	0.121	0.101
sko64	64	48 498	0.054	0.072
Real-life instances(ⅲ)				
bur26a	26	5 426 670	0	0
kra30a	30	88 900	0	0
kra30b	30	91 420	0.008	0.006
ste36a	36	9 526	0.239	0.296
Real-life like instances (ⅳ)				
tai20b	20	122 455 319	0	0
tai25b	25	344 355 646	0	0
tai40b	40	637 250 948	0.082	0.007
tai50b	50	458 821 517	0.457	0.378
tai60b	60	608 215 054	0.672	0.663
tai80b	80	81 845 043	1.845	1.634

表 2 并行算法的加速比和效率

QAP 实例	串行独立运行 10 次的时间/s	并行独立运行 10 次的时间/s	加速比 speedup	效率 efficiency
Unstructed, randomly generated instances(ⅰ)				
tai20a	18.438	8.219	2.243	1.122
tai25a	33.36	16.11	2.071	1.035
tai30a	56.234	27.812	2.022	1.011
tai40a	172.453	67.578	2.552	1.276
tai50a	279.703	132.891	2.105	1.052
tai60a	509.833	230.485	2.212	1.106
tai80a	1 242.935	559.125	2.223	1.112
Grid-based distance matrix(ⅱ)				
nug20	18.440	8.329	2.214	1.107
nug30	65.668	28.281	2.322	1.161
sko42	159.906	77.656	2.059	1.030
sko49	254.651	124.89	2.039	1.020
sko56	393.032	186.625	2.106	1.053
sko64	577.319	280.797	2.056	1.028
Real-life instances(ⅲ)				
bur26a	36.876	18.328	2.012	1.006
kra30a	57.715	28.25	2.043	1.022
kra30b	59.311	28.203	2.103	1.052
ste36a	99.274	49	2.026	1.013
Real-life like instances (ⅳ)				
tai20b	17.164	8.312	2.065	1.033
tai25b	33.0382	16.235	2.035	1.018
tai40b	140.406	67.016	2.095	1.048
tai50b	270.156	131.969	2.047	1.024
tai60b	480.391	231.172	2.078	1.039
tai80b	1145.772	557.282	2.056	1.028

基于 OpenMP 的并行粒子群优化算法，对 QAPLIB 数据进行了测试。并行算法在所有测试实例上都获得了超线性加速比，且运行结果优于串行算法。当前，随着多核技术的不断发展，探讨在多核微机上进行并行计算的实现技术，在合理的计算时间内解决更大规模的问题，有着极其重要的现实意义。

参考文献

[1] EBERHART R C, KENNEDY J. A new optimizer using particle swarm theory[C]. Proc. on 6th International Symposium on Micromachine and Human Science. Piscataway: IEEE Service Center, 1995: 39-43.

[2] 多核系列教材编写组. 多核程序设计[M]. 北京：清华大学出版社, 2007.

[3] SAHNI S, GONZALEZ T. P-complete approximation problems[J]. Journal of the ACM, 1976, 23(3): 555-565.

[4] 高尚. 群智能算法及其应用[M]. 北京：中国水利水电出版社, 2006: 89-90.

(收稿日期：2009-12-18)

作者简介：

周洪斌，男，1981 年生，硕士，讲师，主要研究方向：智能信息处理。