DOI:10.16661/j.cnki.1672-3791.2016.11.100

线性规划问题的分块并行求解及应用①

黄丽嫦 林结
(佛山职业技术学院 基础教学部 广东佛山 528137)

摘 要:在线性规划问题的众多求解算法中,单纯形法仍然是最有效和最常用的算法。分析了单纯形法的计算原理及过程,并对换基迭代过程中的相关运算进行了分块处理,在此基础上,设计实现了一种具有并行处理机制的线性规划问题的求解算法。实际应用表明,新算法具有良好的加速比,且在具有多核架构的微机中易于实现。

关键词:线性规划问题 单纯形法 分块 并行求解
中图分类号:O15 文献标识码:A 文章编号:1672-3791(2016)04(b)-0100-03

Block Parallel Solution of Linear Programming Problem and its Application

Huang Lichang Lin Jie
(Department of basic teaching,Foshan Polytechnic,Foshan Guangdong,528137,China)

Abstract:Simplex method is still the most effective and most commonly used algorithm for solving linear programming problems. Analysis of the calculation principle and process of the simplex method and the correlation operation and swapping based iterative process were divided into blocks, on this basis, design and implementation of the a kind of parallel processing algorithm for solving the mechanism of the linear programming problem. The practical application shows that the new algorithm has a good speedup, and is easy to be implemented in a computer with multi core architecture.

Key Words:Linear programming problem;Simplex method;Block;Parallel solution

规划问题所涉及的是,对有限的资源进行合理地利用或调配,从而达到所期望的目的。这些问题的特点是,有大量的方案(解)满足每个问题的基本条件,究竟把哪一方案(解)选为最优,则与问题中某一个实际要求或目标有关[1]。而线性规划(Linear Programming)问题则是规划问题中特例,该类问题的数学模型可用线性的关系式进行描述。通常,线性规划所研究的问题有两类,一类为资源(人力、物力、财力)是给定的,要求充分利用这些资

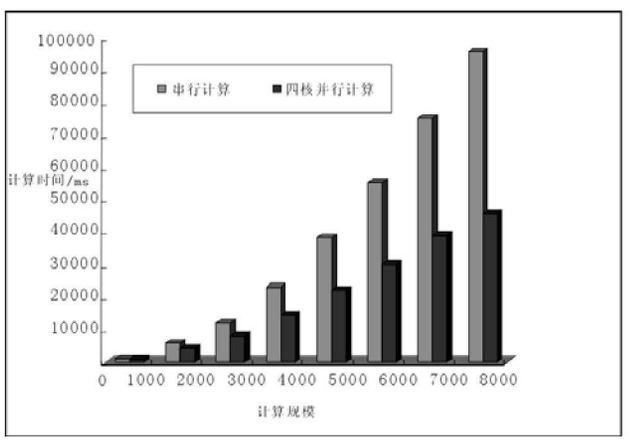

图1 不同计算规模下串行计算和四核并行计算所对应的计算时间

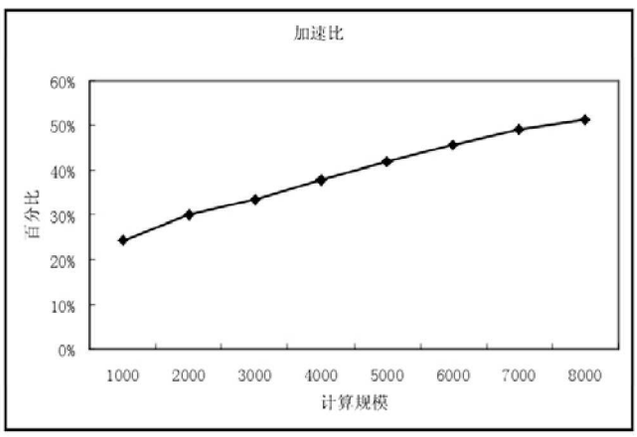

图2 不同计算规模下四核并行计算的加速比

①基金项目:佛山职业技术学院校级科研基金资助项目:2014KY017。
作者简介:黄丽嫦(1962—),女,汉,广东佛山人,学士,讲师,研究方向:计算数学及运筹学。

源，最大限度地实现预期的目标(产量、产值最大、利润最高等)；另一类为任务是给定的，要求以消耗最少的资源(原料、工时、成本)来完成它。前一类问题称为极大值问题，后一类问题称为极小值问题[2-4]。

在线性规划的解法中，单纯形法是一个最著名的方法。它在理论上是完善和严格的，在实践上是方便和有效的。注意到当前的微机普遍具有多核计算架构，为更好地发挥这一特性，笔者对线性规划问题中的单纯形求解法进行了分块并行计算的改进。

1 线性规划问题的数学模型及其标准形式

1.1 线性规划问题的数学模型

现实生活中的线性规划问题是各式各样的，但经过抽象处理后，它们普遍具有如下的共同特点：表示问题的最优化的目标指标是线性函数，表示约束条件的数学式子是一组变量x_1、x_2、Λ 、x_n的线性等式或线性不等式组，为此，可以得到线性规划问题其数学模型的一般形式为[5]：

求一组决策变量x_1、x_2、…、x_n的值，使之满足下列约束条件：

$$\begin{cases} a_{11}x_1+a_{12}x_2+\cdots+a_{1n}x_n=b_1(\text{或}\geqslant b_1\text{、或}\leqslant b_1) \\ a_{21}x_1+a_{22}x_2+\cdots+a_{2n}x_n=b_2(\text{或}\geqslant b_2\text{、或}\leqslant b_2) \\ \cdots \\ a_{m1}x_1+a_{m2}x_2+\cdots+a_{mn}x_n=b_m(\text{或}\geqslant b_m\text{、或}\leqslant b_m) \end{cases} \tag{1}$$

并使目标函数

$$f=c_1x_1+c_2x_2+\cdots+c_nx_n \tag{2}$$

取最大值(或最小值)。

为简便起见，可用矩阵形式来表示上述模型：

$$\max(\min)Z=CX \tag{3}$$

$$s.t\begin{cases} AX\leqslant=\geqslant b \\ X\geqslant 0 \end{cases} \tag{4}$$

对应的，式(3)为目标函数，可能实现最大化，也可能实现最小化；s.t是英文subject to的缩写，表示“满足……条件”之意；式(4)中的第1个式子为约束条件，它可以是“≥”或“≤”的不等式，也可以是“=”的等式，而第2个式子则为非负约束条件。

式(3)～式(4)中，$C=(c_1,c_2,\cdots,c_n)$，$X=(x_1,x_2,\cdots,x_n)^T$，

$$A=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ & \cdots & \cdots & \\ a_{m1} & a_{m2} & \cdots & a_{mm} \end{bmatrix},\ b=\begin{bmatrix} b_1 \\ b_2 \\ \mathrm{M} \\ b_n \end{bmatrix},\ 0=\begin{bmatrix} 0 \\ 0 \\ \mathrm{M} \\ 0 \end{bmatrix}$$

1.2 线性规划问题数学模型的标准形式

为了得到一种通用的线性规划问题的求解方法，还应把上述数学模型的一般形式转换成统一的标准形式，具体转换的内容如下。

(1)目标函数统一为求最大化：若某一实际问题的目标函数为求最小化，则可做一个$Z'=-CX$，从而把问题转换为求最大化，即有：

$$\min Z=CX\Rightarrow\max Z'=-CX \tag{5}$$

(2)引入松弛变量将小于等于型约束条件统一转化为等于型约束条件：若约束条件是小于等于型，则可在不等式的左边加上一个松弛变量，从而把小于等于型约束条件转化为等于型约束条件，即有：

$$a_{j1}x_2+a_{j2}x_2+\cdots+a_{jn}x_n\leqslant b_j\Rightarrow a_{j1}x_2+a_{j2}x_2+\cdots+a_{jn}x_n+x_{n+k}=b_j \tag{6}$$

(3)引入剩余变量将大于等于型约束条件统一转化为等于型约束条件：若约束条件是大于等于型，则可在不等式的左边减去一个剩余变量，从而把大于等于型约束条件转化为等于型约束条件，即有，

$$a_{j1}x_2+a_{j2}x_2+\cdots+a_{jn}x_n\geqslant b_j\Rightarrow a_{j1}x_2+a_{j2}x_2+\cdots+a_{jn}x_n-x_{n+l}=b_j \tag{7}$$

2 单纯形法分块并行求解的改进设计

2.1 单纯形法的计算原理及步骤

求解线性规划问题的单纯形法，是美国学者G.B.Dantzig在20世纪40年代提出来的，这是一种有效实用的算法。单纯形法的计算原理为：从可行域中的任一基本可行解出发，通过转基运算切换至另一个基本可行解，新的基本可行解能使目标函数的数值上升(求最大化时)，基本可行解之间的切换过程称为迭代，经过有限次迭代后，最后能找到最优解。

单纯形法的具体计算步骤为[6]：

步骤(1)：依照式(5)(6)(7)将线性规划问题的数学模型转化为标准形式。

步骤(2)：寻找基本可行解，并令：

$$A'=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & 1 & & 0 \\ a_{21} & a_{22} & \cdots & a_{2n} & & 1 & \\ & \cdots & \cdots & & 0 & & \ddots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & & & \underbrace{\quad 1\quad}_{m} \end{bmatrix} \tag{8}$$

$$C'=\begin{bmatrix} c_1 & c_2 & \cdots & c_n & \underbrace{0\ 0\ \cdots\ 0}_{m} \end{bmatrix} \tag{9}$$

$$b'=[b_1\quad b_2\quad \cdots\quad b_m] \tag{10}$$

步骤(3)：进行最优解的判别：若$c_j'<0$成立，则跳转步骤(4)；否则，结束计算，并以当前基本可行解作为最优解。

步骤(4)：识别主元素，分两步进行，首先找出主元素所处的列数s，然后再找出主元素所处的行数r：

①计算$s\in\min\{c_j'|\ c_j'<0\},j=1,2,\cdots,n+m$，若$a_{is}'>0$成立，跳转步骤②；否则，结束计算，并判定该线性规划问题具有一个无界的解。

②计算$r\in\min\left\{\dfrac{b_i'}{a_{is}'}\middle|\ a_{is}'>0\right\},i=1,2,\cdots,m$

步骤(5)：进行转基迭代运算：执行①和②后，跳转步骤3。

①对主元素行进行变换，其变换表达式为：

$$a_{rj}'=\frac{a_{rj}'}{a_{rs}'},\ j=1,2,\cdots,n+m \tag{11}$$

$$b_i'=\frac{b_i'}{a_{is}'} \tag{12}$$

②对其他元素行进行变换，其变换表达式为：

$$a_{ij}'=a_{ij}'-\frac{a_{is}'}{a_{rs}'}\times a_{rj}',\ i=1,2,\cdots m;\ j=1,2,\cdots,n+m;\ i\neq r \tag{13}$$

$$c_j^{'}=c_j^{'}-\frac{c_s^{'}}{a_{rs}^{'}}\times a_{rj}^{'},j=1,2,\cdots,n+m \tag{14}$$

$$b_i^{'}=b_i^{'}-\frac{a_{is}^{'}}{a_{rs}^{'}}\times b_r^{'},i\neq r \tag{15}$$

2.2 分块并行计算的改进

从上述的单纯形计算步骤可以发现，最优解的迭代计算工作主要集中在式(13)中，且$a_{ij}^{'}$之间的求解并没有前后的依赖关系，为此，可进行有关的分块并行计算改造，进而有效地提升单纯形法的计算速度。

设CPU的核数为P，则可把式(13)的计算任务并行分配给CPU的各个计算核。

(1)核1分配的计算任务为：

$$\begin{bmatrix} a_{11}^{(k+1)'} & a_{12}^{(k+1)'} & \cdots & a_{1(n+m)}^{(k+1)'} \\ a_{21}^{(k+1)'} & a_{22}^{(k+1)'} & \cdots & a_{2(n+m)}^{(k+1)'} \\ \cdots & \cdots & \cdots & \cdots \\ a_{\mathrm{int}(m/p)1}^{(k+1)'} & a_{\mathrm{int}(m/p)2}^{(k+1)'} & \cdots & a_{\mathrm{int}(m/p)(n+m)}^{(k+1)'} \end{bmatrix}=\begin{bmatrix} a_{11}^{(k)'}-\frac{a_{1s}^{(k)'}}{a_{rs}^{(k)'}}a_{r1}^{(k)'} \\ a_{21}^{(k)'}-\frac{a_{2s}^{(k)'}}{a_{rs}^{(k)'}}a_{r1}^{(k)'} \\ \cdots \\ a_{\mathrm{int}(m/p)1}^{(k)'}-\frac{a_{\mathrm{int}(m/p)s}^{(k)'}}{a_{rs}^{(k)'}}a_{r1}^{(k)'} \end{bmatrix}$$

$$\begin{bmatrix} a_{12}^{(k)'}-\frac{a_{1s}^{(k)'}}{a_{rs}^{(k)'}}a_{r2}^{(k)'} & \cdots & a_{1(n+m)}^{(k)'}-\frac{a_{1s}^{(k)'}}{a_{rs}^{(k)'}}a_{r(n+m)}^{(k)'} \\ a_{22}^{(k)'}-\frac{a_{2s}^{(k)'}}{a_{rs}^{(k)'}}a_{r2}^{(k)'} & \cdots & a_{2(n+m)}^{(k)'}-\frac{a_{2s}^{(k)'}}{a_{rs}^{(k)'}}a_{r(n+m)}^{(k)'} \\ \cdots & \cdots & \cdots \\ a_{\mathrm{int}(m/p)2}^{(k)'}-\frac{a_{\mathrm{int}(m/p)s}^{(k)'}}{a_{rs}^{(k)'}}a_{r2}^{(k)'} & \cdots & a_{\mathrm{int}(m/p)(n+m)}^{(k)'}-\frac{a_{\mathrm{int}(m/p)s}^{(k)'}}{a_{rs}^{(k)'}}a_{r(n+m)}^{(k)'} \end{bmatrix} \tag{16}$$

(2)步骤省略。

(3)核p分配的计算任务为：

$$\begin{bmatrix} a_{(\mathrm{int}(m/p)*(p-1)+1)1}^{(k+1)'} & a_{(\mathrm{int}(m/p)*(p-1)+1)2}^{(k+1)'} & \cdots & a_{(\mathrm{int}(m/p)*(p-1)+1)(n+m)}^{(k+1)'} \\ a_{(\mathrm{int}(m/p)*(p-1)+2)1}^{(k+1)'} & a_{(\mathrm{int}(m/p)*(p-1)+2)2}^{(k+1)'} & \cdots & a_{(\mathrm{int}(m/p)*(p-1)+2)(n+m)}^{(k+1)'} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1}^{(k+1)'} & a_{m2}^{(k+1)'} & \cdots & a_{m(n+m)}^{(k+1)'} \end{bmatrix}=$$

$$\begin{bmatrix} a_{(\mathrm{int}(m/p)*(p-1)+1)1}^{(k)'}-\frac{a_{(\mathrm{int}(m/p)*(p-1)+1)s}^{(k)'}}{a_{rs}^{(k)'}}a_{r1}^{(k)'} & a_{(\mathrm{int}(m/p)*(p-1)+1)2}^{(k)'}-\frac{a_{(\mathrm{int}(m/p)*(p-1)+1)s}^{(k)'}}{a_{rs}^{(k)'}}a_{r2}^{(k)'} \\ a_{(\mathrm{int}(m/p)*(p-1)+2)1}^{(k)'}-\frac{a_{(\mathrm{int}(m/p)*(p-1)+2)s}^{(k)'}}{a_{rs}^{(k)'}}a_{r1}^{(k)'} & a_{(\mathrm{int}(m/p)*(p-1)+2)2}^{(k)'}-\frac{a_{(\mathrm{int}(m/p)*(p-1)+2)s}^{(k)'}}{a_{rs}^{(k)'}}a_{r2}^{(k)'} \\ \cdots & \cdots \\ a_{m1}^{(k)'}-\frac{a_{ms}^{(k)'}}{a_{rs}^{(k)'}}a_{r1}^{(k)'} & a_{m2}^{(k)'}-\frac{a_{ms}^{(k)'}}{a_{rs}^{(k)'}}a_{r2}^{(k)'} \end{bmatrix}$$

$$\begin{bmatrix} \cdots & a_{(\mathrm{int}(m/p)*(p-1)+1)(n+m)}^{(k)'}-\frac{a_{(\mathrm{int}(m/p)*(p-1)+1)s}^{(k)'}}{a_{rs}^{(k)'}}a_{r(n+m)}^{(k)'} \\ \cdots & a_{(\mathrm{int}(m/p)*(p-1)+2)(n+m)}^{(k)'}-\frac{a_{(\mathrm{int}(m/p)*(p-1)+2)s}^{(k)'}}{a_{rs}^{(k)'}}a_{r(n+m)}^{(k)'} \\ \cdots & \cdots \\ \cdots & a_{m(n+m)}^{(k)'}-\frac{a_{ms}^{(k)'}}{a_{rs}^{(k)'}}a_{r(n+m)}^{(k)'} \end{bmatrix} \tag{17}$$

3 算法的性能测试

在Intel Xeon E5450四核3.0GHz CPU(每个核心的一级缓存各由32KB数据缓存和32KB指令缓存组成，二级缓存容量为12MB，)、KingSton DDR3 1333MHZ 4GB内存及Red Hat Enterprise Linux 6.1操作系统的环境中对上述的单纯形并行算法进行了模拟，程序采用OpenMP和C++语言进行编写[7-8]。

不是一般性，(4)式中矩阵A的行数$m=n$，并按如下规则产生[9]：

$$A=(a_{ij})_{n\cdot n}=\begin{cases} n,i=j; \\ \dfrac{i+j}{2},i\neq j; \end{cases},i,j=0,1,\cdots n-1$$

矩阵A的规模从1000阶逐渐增至8 000阶，串行计算和四核并行计算所对应的计算时间如图1所示；而图2则是不同计算规模下四核并行计算的加速比。

从图2可知，单纯形的分块并行计算的加速比随着计算规模的增加而增长，在矩阵的阶数为8 000阶时，其加速比达到51.2%。

4 结语

在单纯形法的基础上，提出了一种线性规划问题的分块并行求解算法，新算法具有良好的加速比和易于实现的特点，理论分析及相关实验均表明它是有效的。

参考文献

[1] 范玉妹，徐尔，赵金玲，等.数学规划及其应用[M].3版.北京：冶金工业出版社，2009：1-7.
[2] 张香云.线性规划[M].杭州：浙江大学出版社，2009：1-173.
[3] 杜红.应用运筹学[M].杭州：浙江大学出版社，2010：19-72.
[4] 张惠恩.管理线性规划[M].大连：东北财经大学出版社，2001：1-91.
[5] 胡运权.运筹学教程[M].北京：清华大学出版社，2007：11-14.
[6] 庞碧君.线性规划与随机线性规划[M].郑州：郑州大学出版社，2007：17-55.
[7] 周伟明.多核计算与程序设计[M].武汉：华中科技大学出版社，2009：75-124.
[8] 武汉大学多核架构与编程课程组.多核架构与编程技术[M].武汉：武汉大学出版社，2010：23-96.
[9] 尚月强.局域网上求解线性方程组的一种并行Gauss-Seidel迭代算法研究[J].计算机应用与软件，2008，25(9)：245-247.