Simplex Method Implementation

CSE:501, Parallel Computing

Submitted by

Tulshi Chandra Das, Roll:BSSE0811

Maloy Kanti Sarkar, Roll:BSSE0834

Submitted to

Dr. B M Mainul Hossain
Assistant Professor,IIT,DU

15 March, 2018

Abstract- **The simplex method is frequently the most efficient method of solving linear programming (LP) problems. We review previous attempts to parallelise the simplex method in relation to efficient serial simplex techniques and the nature of practical LP problems. For the major challenge of solving general large sparse LP problems, there has been no parallelisation of the simplex method that offers significantly improved performance over a good serial implementation. However, there has been some success in developing parallel solvers for LPs that are dense or have particular structural properties.**
Keywords: linear programming, simplex method, parallel computing

1 Introduction

Simplex method is an algorithm which is used to solve the linear programming problems effeciently.

We consider the computation of econometric estimators given by the optimization problem: minx z = 8x1 +9x2, where x1,x2 are the variables.

$2x1 + 3x2 \leq 50$

$2x1 + 6x2 \leq 80$

$3x1 + 3x2 \leq 70$

Simplex method works on standard form of the equations. So, we have to transform the equations to standard form: z = = 8x1 +9x2 + s1 + s2 + s3 $2x1 + 3x2 + s1 = 50$

$2x1 + 6x2 + s2 = 80$

$3x1 + 3x2 + s3 = 70$

Here s1,s2 and s3 are slack variable.

The Starting Tableau

Basic	z	x1	x2	s1	s2	s3	solution
z	1	-8	-9	0	0	0	0
s1	0	2	3	1	0	0	50
s2	0	2	6	0	1	0	80
s3	0	3	3	0	0	1	70

For optimization problems without closed form solutions, an iterative procedure is typically used. These algorithms successively evaluate the objective function z at different trial parameter vectors until a parameter vector achieves a convergence criterion.

Let bi be the RHS of the ith row. Let aij be the coefficient of the entering variable xj in the ith row. The following "minimum ratio test" decides the leaving variable

The optimization steps are:
1. Determining the pivot row and pivot element using aij and xj
2. Make a non basic variable to basic variable
3. change the pivot row
4. Then change all other element in the table

The step 1 and 2 and 3 are serialized process and not costly. The step 4 is more costly and time consuming. We will parallelise the step 4.

There are two main sources of computation time in these numerical procedures. First, for each trial vector of parameters, there is the computational cost of evaluating the objective function at this vector of parameters. Where the objective function involves simulation or the solution of a complex behavioral model, such as a dynamic programming model, the objective function level computation costs can be large. A second source of computational costs is that for a high dimensional and continuous parameter space , a large number of vectors of parameters may need to be tried before the convergence criterion is achieved. This parameter level computation cost is increasing in the number of parameters, which is generally related to the complexity of the underlying behavioral model.

2 Related work

2.1 Concept of MPI

Most popular high-performance parallel architectures used in the parallel programming can be divided into two classes: message passing and shared storage. The cost of message passing parallel processing is larger, suitable for large-grain process-level parallel computing. Compared with other parallel programming environment, message passing has good portability, supported by almost all parallel environments. Meanwhile, it has good scalability and complete asynchronous communication function, which can well decompose tasks according to the requirements of users, organize data exchange between dierent processes and is applied to scalable parallel algorithms. MPI is an interface mode widely used in various parallel clusters and network environ- ments based on a variety of reliable message passing libraries. MPI is a message passing parallel programming standard used to build highly reliable, scalable and exible distributed applications, such as work ow, network management, communication services. MPI is a language-independent communications pro- tocol. FORTRAN, C and C++ can directly call the API library. The goals of MPI are high performance, scalability and portability. MPI is a library spec- ication for message passing, not a language. Message Passing Interface is a standard developed by the Message Passing Interface Forum (MPIF). MPI is a standard library specication designed to support parallel computing in a

distributed memory environment. The rst version (MPI-1) was published in 1994 and the second version (MPI-2) was published in 1997 [1]. Both point- to-point and collective communication are supported. MPICH is an available and portable implementation of MPI, a standard for message passing used in parallel computing. MPI has become the most popular message passing standard for parallel programming. There are several MPI implementations among which MPICH is the most popular one. MPICH1 is the original im- plementation of MPICH that implements the MPI- 1standard. MPICH2 is a high-performance and widely portable implementation of the MPI standard (both MPI-1 and MPI-2). The goals of MPICH2 are to provide an implementa- tion of MPI that eciently supports dierent computation and communication platforms including commodity clusters, high-speed networks and proprietary high-end computing systems. The standards of MPI are as follows [2]: -Point-to-point communication
-Collective operations
-Communication contexts
-Process groups.
-Process topologies
-Bindings for FORTRAN 77 and C
-Environmental management and inquiry
-Proling interface

2.2   MPI Functions

MPI is a library with hundreds of function-calling interfaces, and FORTRAN, C language and C++ can directly call these functions. Many parallel programs

Table 1: MPI Basic Functions

Function	Functionalities
MPI Init()	Initialization
MPI Finalize()	Termination
MPI Send()	Send
MPI Recv()	Receive
MPI Comm size()	Access to the number of processes
MPI Comm rank()	Access to the identication number of a process

can be written with just six basic functions, almost complete all of the communication functions. Table 1 illustrates the basic functions. MPI Init() initializes the MPI environment and assigns all spawned processes; MPI Finalize() terminates the MPI environment; MPI Comm size() nds the number of processes in a communication group; MPI Comm rank() gives the identication num- ber of a process in a communication group; MPI Send() sends message to the destination

process of rank dest and MPI Recv() receives message from the specied process of rank source.

2.3 The Messing Passing Process of MPI

MPI is a parallel programming standard based on message passing, whose function is to exchange information, coordinate and control the implementa- tion steps with the denition of program grammar and semantics in the core library by sending messages between the concurrent execution parts. First all of the MPI programs contain mpi:h header le, and then complete the initial- ization of the program by MPI Init(), after that, establish process topology structure and new communicator and call the functions and applications to be used for each process, nally use MPI Finalize() to terminate each process.

2.4 Simplex algorithm

Suppose, Objective function,maximise z = ax1 + bx2
Subject to constraints,
$ax1 + bx2 \leq c1$
$ax1 + bx2 \leq c2$
$ax1 + bx2 \leq c3$
$x1, x2 \geq 0$
Introducing slack variables, the LPP is same as, Objective function,maximise z =
ax1 + bx2 + s1 +s2 + s3
ax1 + bx2 + s1 = c1
ax1 + bx2 + s2 = c2
ax1 + bx2 + s3 = c3
$x1, x2, s1, s2, s3 \geq 0$

<div align="center">Basic Feasible Solution</div>

Let,

- -x1,x2 as non basic variable

- -Then BFS can be got immediately as ,(0,0,c1,c2,c3)

This is possible because in the LHS of the constraint equations (*) the coefficients of the basic variables s1, s2, s3 form an Identity matrix (of size 3 x 3).

<div align="center">Closer Look (1)</div>

- We note that the z-row is the objective function row.

- The remaining 3 rows are the basic variable rows.

- Each row corresponds to a basic variable;

  - the leftmost variable denotes the basic variable corresponding to that row.

- In the objective function row,

  - the coefficients of the basic variables are zero.

- In each column corresponding to a basic variable,

  - basic variable has a non-zero coefficient, namely 1,
  - all the other basic variables have zero coefficients.

- We see at present, the objective function, z, has value zero.

  - the leftmost variable denotes the basic variable corresponding to that row.

- We now seek to make

  - one of the nonbasic variables as basic (and so) one of the basic variables will become nonbasic (that is will drop down to zero)
  - the nonbasic variable that will become basic is chosen such that the objective function will "improve":

- The nonbasic variable that will become basic is referred to as "entering" variable and the basic variable that will become nonbasic is referred to as "leaving" variable.

<div align="center">Criterion for "leaving" variable (Feasibilty Condition)</div>

Let while b_i be the RHS of the ith row. Let $a_i j$ be the coefficient of the entering variable x_j in the ith row. The following "minimum ratio test" decides the leaving variable: Choose x_k as the leaving variable where k is given as that row index i for which the ratio
$\{b_i \div a_i j : a_i j \geq 0\}$ is least (break the ties arbitrary)

Pivot element:

- The entering variable column is called the pivot column.

- The leaving variable row is called the pivot row.

- The coefficient in the intersection of the two is referred to as the pivot element.

Operations: We apply elementary row operations to modify the simplex tableau so that the pivot column has 1 at the pivot element and zero in all other places.

The elementary Row operations are as follows:

- New pivot row = old pivot row pivot element

- New z row = old z row − (coefficient of the entering variable in old z row * New pivot row)

- Any other new row = corresponding old row − (old coefficient of the entering variable in that row * New pivot row)

- We shall also change the legend of the new pivot row only as the entering variable.

The last tableau is the optimal tableau as all entries in the objective function row are ≥ 0 and the LPP is a maximization problem.

References

1. W. Gropp, E. Lusk, "Implementing MPI: the 1994 MPI Implementors Workshop", Pro- ceedings of Scalable Parallel Libraries Conference, pp. 55-59, 2002.

2. Y. Aoyama, J. Nakano, "RS/6000 SP: Practical MPI Programming", International Tech- nical Support Organization,August, 1999.