Journal of the Korean Institute of Industrial Engineers
Vol. 17, No. 2, Dec., 1991

Development of Nonlinear Programming Approaches to Large Scale Linear Programming Problems†

Soo Y. Chang*

비선형계획법을 이용한 대규모 선형계획해법의 개발

장 수 영

Abstract

The concept of *criterion function* is proposed as a framework for comparing the geometric and computational characteristics of various nonlinear programming approaches to linear programming such as the method of centers, Karmakar's algorithm and the gravitational method. Also, we discuss various computational issues involved in obtaining an efficient parallel implemetation of these methods. Clearly, the most time consuming part in solving a linear programming problem is the direction finding procedure, where we obtain an improving direction. In most cases, finding an improving direction is equivalent to solving a simple optimization problem defined at the current feasible solution. Again, this simple optimization problem can be seen as a least squares problem, and the computational effort in solving the least squares problem is, in fact, same as the effort as in solving a system of linear equations. Hence, getting a solution to a system of linear equations fast is very important in solving a linear programming problem efficiently. For solving system of linear equations on parallel computing machines, an iterative method seems more adequate than direct methods. Therefore, we propose one possible strategy for getting an efficient parallel implementation of an iterative method for solving a system of equations and present the summary of computational experiment performed on transputer based parallel computing board installed on IBM PC.

---

† 이 논문은 1990년도 문교부지원 학술진흥재단의 신진연구 학술연구조성비에 의하여 연구되었음.

* Industrial engineering Department, Pohang Institute of Science and Technology
(포항공과대학, 산업공학과)

1. Introduction

In order to explain the geometric and computational aspects of existing nonlinear programming approaches to linear programming problems such as the method of centers, Karmarkar's algorithm and the gravitational method, we introduce the concept of "the criterion function" as a framework for comparing such methods. The process of solving a linear programming problem can be seen as a process of generating a piecewise linear path that consists of sequence of points and directional vectors linking these points. These directional vectors are obtained by solving a simple optimization problem, and can be viewed as the negative gradient of a certain function which we define as the "*criterion function*".

The most time consuming part in solving a linear programming problem is certainly the direction finding procedure, where we obtain an improving direction. The task of finding an improving direction always involves the process of solving a least squares problem which is equivalent to solving a system of linear equations. Therefore, it is vital to have an efficient algorithm for solving a system of equations in order to build an efficient implementation of linear programming algorithms.

On parallel computing machines, the iterative method seems more adequate than direct methods for solving system of linear equations, especially when the linear system is large and sparse. Hence, we propose one possible strategy for getting an efficient parallel implementation of an iterative method for solving a system of equations and we present the summary of computational experiment performed on transputer based parallel computing board installed on IBM PC.

2. Comparison of Nonlinear Programming Approaches to Linear Programming

In this section, we compare and contrast existing nonlinear programming approaches to linear programming through the concept of "*criterion function*". The discussion will be focused on the computational aspects of these methods.

A paradigm which is common to almost all LP methods is "*the path generation*", that is, starting from a given initial point, a piecewise linear path is generated until certain termination criteria are met. The computational effort involved in generating such a path involves four distinct procedures ; initialization, direction finding, step length determination and finalization. At initialization procedure, the problem is transformed into an appropriate format with respect to a given method, and a starting point of the path is selected. Then, points on the path are generated through an alternating sequence of direction finding and step length determination procedures. At termination, a certain finalization procedure may be required to yield a true optimum point, if the terminal point of the path is only a near optimum point.

In order to select a direction, each method has its own criteria for evaluating the "goodness" of a given direction. Often the goodness of a direction is measured by its steepeness with respect to a given direction, that is, by taking the inner product of a given direction and the vector that defines the linear objective function. In some cases, the chosen direction is simply the negative gradient

vector of a certain nonlinear function. These functions that LP methods use as criterion for selecting directions are defined to be the "criterion functions."

For a given criterion function value, we can define a surface where all points have the same function value. Understanding this surface is very helpful to the analysis of the geometric and computational aspects of a given method. The following figures ; Figure 1, 2, and 3 illustrate such surfaces of various LP methods.

We consider an LP in the form

$$\text{Minimize} \quad z(x)=c\ x \qquad \cdots\cdots (1)$$
$$\text{subject to} \quad Ax \geq b$$

In all of these examples, the feasible region is shown as a box, and the objective is to minimize a linear function cx.

In the Simplex method, the criterion function is the same as the original linear objective function, i. e, cx, and the choice of direction is restriced to those of edges incident at the current extreme point(See figure 1.). In the case of minimization, the rate of descent of the objective function, is used to evaluate the goodness of a given direction. Note that the gradient of the criterion function is c at all points in the solution space, in this case.

In the method of Centers, the criterion function is a logarithmic barrier penalty function[12]. (See figure 2) With respect to an LP in the form of(1), an example of such function can be constructed as ;

$$Bp(x^r) = -k \log(cx^r - cx) - \sum_{i=1}^{m} \log (A_i x - b_i),$$

where x^r is the current interior point, A_i. is the i-th row of the matrix A, and k is a positive integer. The value of this criterion function blows up to $+\infty$, either when the point gets close to the boundary of the polytope that defines the feasible region, or when it gets close to the hyperplane defined as ;

$$\{x \mid cx = cx^r\}$$

This hyperplane gets "pushed" down as the point x^r gets close to the optimum point, and the criterion function changes. In this case, note that the gradient of the criterion function depends on all the constraints in the problem, as well as the current point.

In Karmarkar's method, the criterion function is the potential function, which is quite similar to the barrier function in the method of Centers[1, 6, 15] (See figure 3). With respect to the LP (1), an example of such function can be constructed as ;

$$Po = n \log(cx - z^*) - \sum_{i=1}^{m} \log (A_i x - b_i),$$

where z^* is the optimum objective value, which is assumed to be known, and n is the number of variables. (This particular potential function is constructed, on the basis of the dual variant of Karmarkar's method). In this potential function, the term $-\sum_{i=1}^{m} \log (A_i x - b_i)$, blows up to $+\infty$, as the point gets close to the boundary of the polytope that defines the feasible region, prohibiting the point to get close to the boundary. However, as the point gets close to the actual optimum solution, the term $n\log(cx - z^*)$ blows up to $-\infty$ so that it allows the point to get close to the boundary. In this method, also note that the negative gradient of the criterion function depends on the current point and all the constraints in the problem.

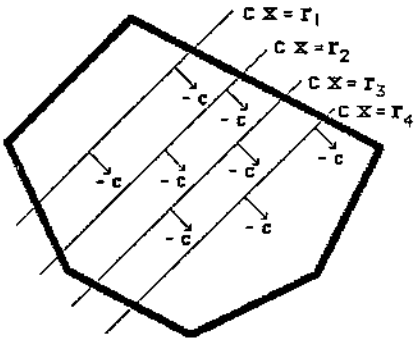

Fig 1. Equi-objective lines in the Simplex method in R^2.

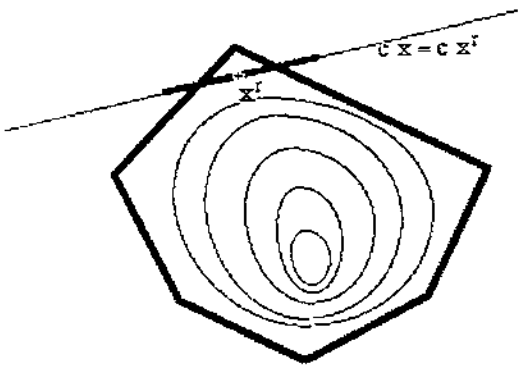

Fig 2. Equi-barrier-penalty curves in the Method of centers in R^2.

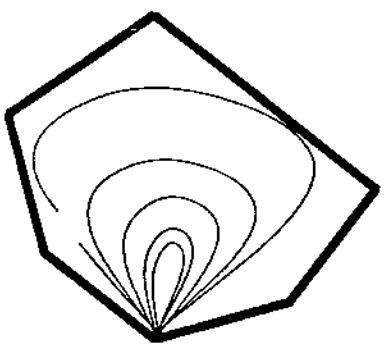

Fig 3. Equipotential curves in Karmarkar's method in R^2.

Certainly, the major part of the computational effort involved in generating a path goes into the direction finding procedure. We will now investigate the nature of the directions chosen by various LP methods, and computational effort involved in selecting directions by considering their criterion functions.

The Simplex method is based on the principle of using only "edge directions" which are descent directions for the original objective function. Karmarkar's method used the negative gradient of a logarithmic "potential function". [6] The Method of Centers uses the negative gradient of a logarithmic "barrier penalty function". [1, 12]

In the Simplex method, the current point is always an extreme point of the polytope defining the feasible region. With respect to a given extreme point, the variables of the problem are partitioned into two groups ; basic and nonbasic variables. Under the nondegeneracy assumption, an edge incident at the current extreme point is defined, for each nonbasic variable. Then, the Simplex method selects one among these edges, and takes a step along this edge to generate the next extreme point. Various selection rules have been proposed for choosing one among these edges. However, all such rules are based on the measure called "the updated cost." This measure is based on the inner product of the directional vector corresponding to a given edge and the cost vector c, which is fixed throughout the execution of the method. Therefore, the choice of direction is done on the basis of only the local information, namely the current extreme point and the edges incident at it.

Opposed to this, the directions chosen is Karmarkar's method, and method of Centers are the negative gradients of their criterion functions which depend on the current point and all the constraints in the problem. Hence, the computational effort

involved in direction finding procedure is quite substantial in those methods.

After a direction is selected, the step length is to be determined in order to generate the next point on the path, from which the whole process repeats. In this step length determination process, maintaining the feasibility of points on the path is the key computational issue.

the Simplex method, and all of its variants use the usual minimum ratio test for the step length determination, where the next point on the path is determined by moving all the way to the first blocking face of the feasible region. Interior methods such as Karmakar's method and method of Centers use different mechanism for determining the step length. In Karmarkars's method as originally presented in [6], the step length is a fixed fraction of the radius of the sphere inscribed in the simplex of the projected space. (.25 is the fraction recommended in [6]) However, other variants of Karmarkar's method determines the step length by taking a fraction of the step length obtained from the usual minimum ratio test. In the method of Centers as presented in [12], the step length is determined to be small enough to ensure the strict interior feasibility of the next point, without explicitly performing the minimum ratio test.

An advantage of performing the minimum ratio test is that we can guarantee the maximum possible improvement in the objective value at each iteration. On the other hand, using a fixed step length is computationally attractive, since it does not require the computation of the minimum ratio. However, this does not guarantee the maximum possible improvement at each iteration.

In Figures 4, 5 and 6, we illustrate the paths taken by the Simplex method, Method of Centers, and Karmarkar's method.

The Simplex method and all its variants generate a "connected edge path", All the points on the path where the direction changes are extreme points. See Figure 4 for an illustration.

The edges are the one dimensional faces of the polyhedron that defines the feasible region. We can represent each edge by a thin "pipe" that a small drop can travel through. This way, we get a "pipeline network" which is a "web" of pipes connecting vertices. The path taken by the Simplex method can be interpreted as that of a small drop released into this pipeline network at an extreme point of the polyhedron, as it is pulled down by a powerful gravitational force pulling everything in the direction of $-c^T$. The interior methods such as Karmarkar's method and method of Centers, always use directions only through the interior, which is the full dimensional face of the feasible region.

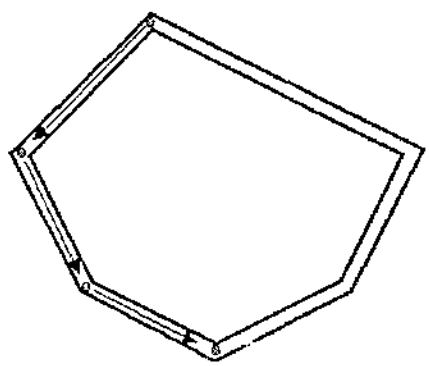

Fig 4. The path taken by the Simplex method.

In the method of centers, the center is defined to be a point that minimizes the value of a logarithmic barrier penalty function. As the new points on the path are generated, the form of the barrier function changes parametrically, and the new cen-

ter asymptotically gets closer to the optimum point of the problem. In this method, the direction is the negative gradient of the barrier penalty function. [4, 12]

In Figure 5 (a) and (b), two steps taken by the method of Centers are illustrated. The "O" mark in the figure shows the current point. The straight line passing through the point shows the objective hyperplane that is "pushed down" as a new point on the path is generated[12].

Karmarkar's method is quite similar to the method of Centers. Instead of the barrier penalty function, a potential function is used to monitor the performance of the method. The direction chosen for the next move is the same as the negative gradient of this potential function. In Figure 6 (a) and (b), two steps taken by Karmarkar's method are illustrated. The "+" mark in the figure shows the current point.

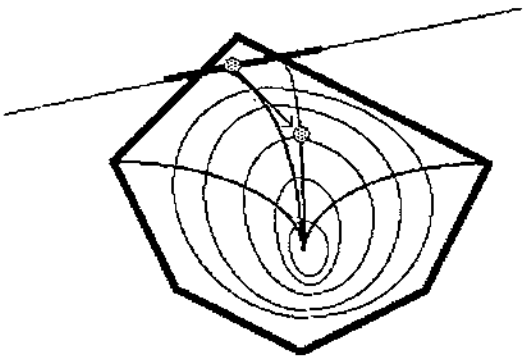

Fig 5. (a) The path taken by the method of Centers.

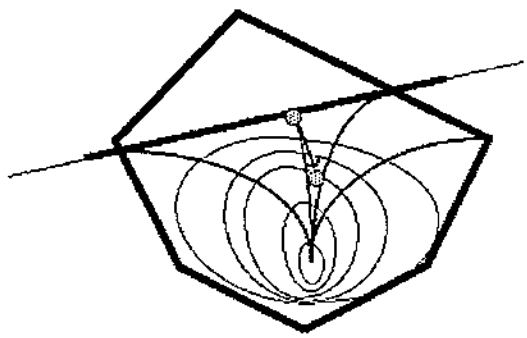

Fig 5. (b) The path taken by the method of Centers.

The main advantage of Karmarkar's method and the method of Centers is that all the combinatorial features of the polytope are "abstracted" into a single measure ; i. e. the barrier penalty function, and potential function. Then, the polynomial convergence of these method is directly derived from the fact that this measure is strictly decreased by a certain fraction at each step.

The main disadvantage of these interior methods is that the direction finding procedure is computationally expensive. This is due to the fact that the chosen direction is the negative gradient of the criterion function which must include the information about all the constraints and the current point. In other words, the contour of the equi-criterion function values in Figure 5, and 6 is formed by taking all the constraints and the current point into account. Futhermore, the redundant constraints, (the i-th constraint is said to be a redundant constraint in (1), if its removal from (1) does not change the set of feasible solutions) play the same role as nonredundant constraints of the problem in forming the contour, since redundant constraints, if any, are also included in the defintion of the criterion functions.

The path of the Gravitatinal method is shown

in Figure 7 below. The criterion function of the method is same as the objective function. The arrow in the figure is the negative gradient, and the small drop in the figure shows the point on the path, x^1, x^2, x^3 and x^4. The path taken by the method is the same as the path that an actual drop may have taken if the gravitational force in the negative gradient of the objective functin was present.

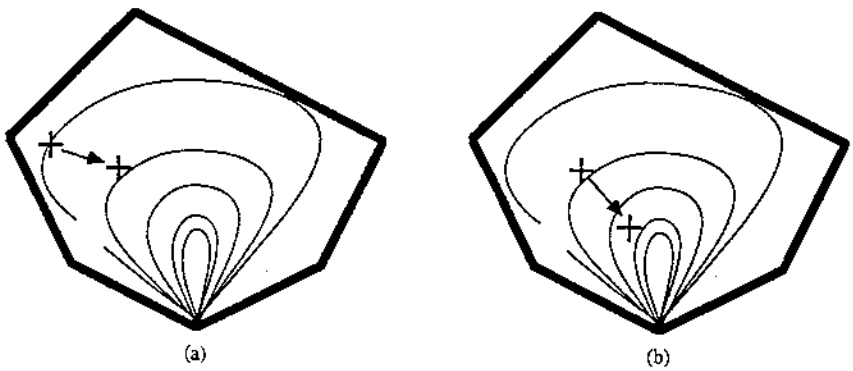

Fig 6. The path taken by Karmarkar's method

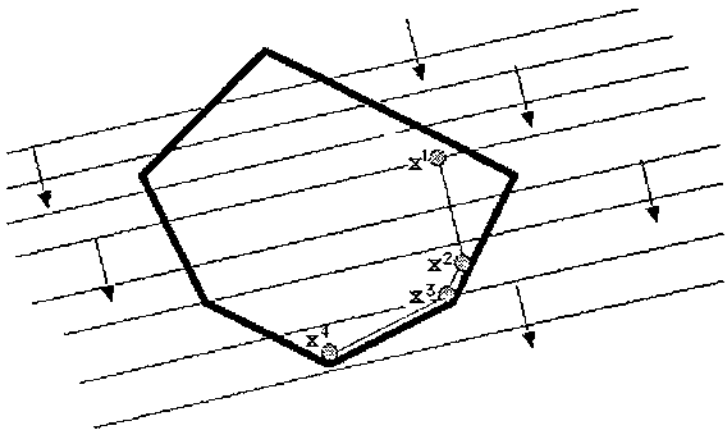

Fig 7. The path taken by the Gravitational method.

3. Solving Least Squares Problem on parallel computer

In general, the problem with combinatorial nature is much harder to solve on parallel computer than the problem without it. Hence, in this section, we take a close look at the direction finding problem and its computatinal characteristics, especially the

combinatorial nature of the problem in order to select the adequate method for parallel implementation.

At each step of the interior point methods such as the method of centers, and Karmarkar's method, a vector which can be written in the following form must be computed ;

$$u = cD^2B^T(BD^2B^T)^{-1} \quad \cdots\cdots (2)$$

where the matrix **D** is diag(a_1, a_2, ···, a_n), and (a_1, a_2, ···, a_n) is the coordinate of the current point, and the matrix **B** is a constant coefficeint matrix given in the problem. (there are slight differences among different methods, but the computational task involved in getting an improving direction is essentially same as computing the vector that can be written in the above form in all those methods).

Clearly, the **u** vector defined in (2) solves the following least squares probelm ;

$$\textbf{Minimize} \ \| cD - u\,BD\| \quad \cdots\cdots (3)$$

In the method of centers, Karmarkar's method and its variants, the direction is obtained by solving a least squares problem and in the gravitational method, the problem of finding an improving direction is a least squares problem with non-negativity restriction on variables. Hence, the direction finding problem is essentially same except that the matrix **D** changes as the points on the path are generated. In the gravitatinal method, however, the direction finding problem is in the following form ;

$$\textbf{Minimize} \ \|c - uF\| \quad \cdots\cdots (4)$$
$$\textbf{subject to} \ u > 0$$

where the matrix **F** is defined to be a matrix which consists of a set row vectors corresponding to the constraints that are "touched" by the spherical ball modelled in the gravitational method[2].

Even though (3) and (4) are both least squares problems, there is a striking difference in difficulty of solving each problem. That is, the solution to (3) can be written as an explicit closed form, but the solution to (4) can never be written in a closed form. This is due to the fact that the problem (4) has a very difficult combinatiorial nature.

In order to understand this combinatorial nature of the problem (4) more clearly, we present figure 8 and 9 which shows the geometric nature of two different least squares problems.

In figure 8 and 9, the vector c_p represents the solution vector of the least squares problem for (3) and (4). The plane in the figure 8 represents the subspace spanned by the row vactors of the matrix **BD**, and the cone in the figure 9 represents the cone generated by the row vectors of the matrix **F**. (vector a_1, a_2, and a_3 can be thought of as some row vectors of the matrix **F**).

As one can see from the figure 8, the solution to (3) is simply an orthogonal projection of the vector **c** on the subspace. In figure 9, however, even if we consider only two dimensional subspaces, we have 3 different subspaces spanned by arbitray combinations of two out of three vectors a_1, a_2, and a_3. Hence, in this case, we have three different projections of the vector **c**, among which one yields the optimum solution to (4). (in this example, the subspace spanned by the vector a_1 and a_2 is the subspace that defines the optimum solution). In fact, if the row dimension of the matrix **F** in (4) is **k**, there could be 2^k different subspaces with various dimension, and their corresponding othogonal projections among which we must

choose the "right" one in order to find the optimum solution. This very fact adds the combinatorial flavour to the problem (4). Therefore, it is much harder to handle the problem (4) than (3).

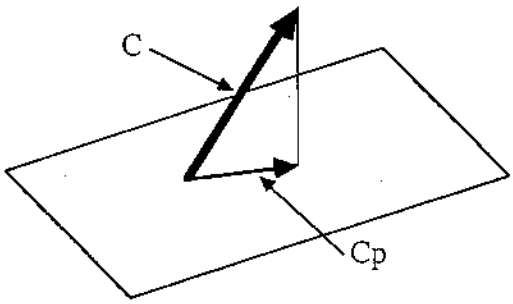

Fig 8. Least Squares Problem without nonnegativity

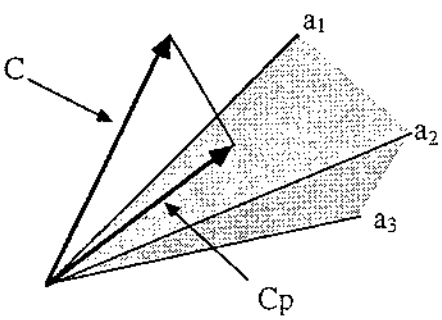

Fig 9. Least Squares Problem with nonnegativity

4. Computational experiment

As we have seen in the previous section, the case of least squares problem without nonnegativity constraints seems to have simpler combinatorial structure. Hence, we attempt to obtain a parallelization of an algorithm for (3) instead of (4). Namely, we try to solve the following system of equation.

$$\alpha BD^2B^T = cD^2B^T \quad \cdots\cdots (5)$$

among many existing algorithms for (5), the preconditioned conjugate gradient method is selected for our preliminary computational experiment. The most time consuming part of the conjugate gradient method is the matrix vector multiplication which takes $O(n^2)$ operations, where n is the dimension of the matrix BD^2B^T in (5). The number of steps of the conjugate gradient method could be upto n, if the matrix BD^2B^T is ill-conditioned. However, if the matrix is preconditioned properly, the number of the conjugate gradient step could be as small as 1. (when we have perfect preconditioner, such as the exact cholesky factor of the matrix BD^2B^T.)

We have to solve (5) at each step of the interior point methods such as the method of centers and Karmarkar's method, but only part that changes in the problem of (5) is the diagonal matrix D, which is nothing but a diagonal matrix with coordinates of the current solution on its diagonal. Hence, if the solution does not change significantly, we don not have to calculate the preconditioner which is the cholesky factor of the matrix BD^2B^T. Instead, we could use the preconditioner that was obtained in the previous step. Only when there is a big chagnge in the solution we calculate the preconditioner.

The target hardware system is the Quatputer® from Micro Way Inc. which has four T800 computing elements(CE's). The figure 10 shows the hardware confinguration of the Quatputer®.

The Qaudputer® is a board that can be installed on IBM PC. Each CE has computing power, of 20MIPS, and 1 megabytes of local memory. Only one of the CE's (which is called the "root transpu-

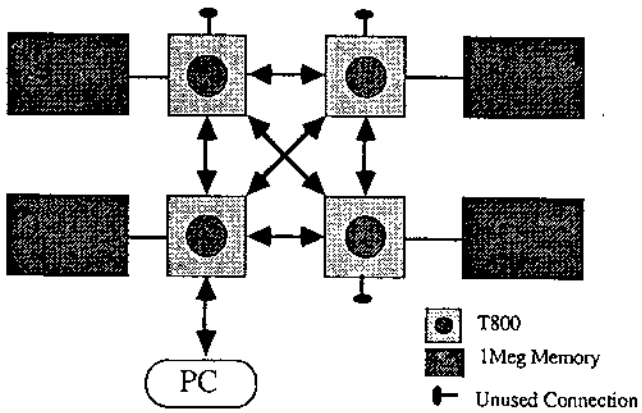

Fig 10.Hardware configuration of Qaudputer®

ter") is directly connected to the host PC. The Qaudputer® can be thought of as a 2-dimensional hypercube MIMD computer.

We have performed two basic experiments ; one is testing effect of preconditioning in the preconditioned conjugate method, and the other is testing a parallel version of the conjugate gradient method. Table 1 shows the effect of the preconditioning, and table 2 shows the effect of parallelizatin in the conjugate gradient method.

Table 1 is a case of an LP problem where the row dimension of the matrix $\mathbf{BD^2B^T}$ is 20. On Table 1, the first column is the iteration number of the Karmakar's algorthm, and the second through fourth columns contain the number of conjugate gradient steps in each iteration respectively for the cases of getting preconditioner every iteration, getting no preconditioner, and getting preconditioner every iteration, getting no preconditioner, and getting preconditioner whenever the matrix **D** changes significantly (in this particular case, we used 0.021 for judging the significance of change in the L_2 norm of the matrix **D**). As one can see from

Table 1. Number of conjugate gradient steps in each iteration of Karmarkar's algorithm

Iteration	Precond	No Precond	Skip Precond
1	1	20	1
2	1	19	2
3	1	20	2
4	1	19	2
5	1	20	2
6	1	20	8*
7	1	18	14*
8	1	20	16*
9	1	20	20*
10	1	19	20*
11	1	20	1
12	1	20	1
13	1	19	7
14	1	20	8

the first column of Table 1, getting preconditioner at every iteration takes the least number of conjugate gradient steps, but one must pay for the computing time of getting the preconditioner which is significant amount of time. The third column shows the fact tht one must go through full stpes (20 in this case) of the conjugate gradient method at every iteration, since the matrix is not conditioned

at all. On the last column, the numbers with "*" marks are for the cases when the preconditioner was not calculated since the matrix D did not change significandy. From the result of our experiment, it seems best to take the strategy of getting preconditioner whenever the change in the norm of D is greater than a certain threshold.

Table 2. Speed up for different matrix sizes

Matrix size	CPU time for 4 CE'S / CPU time for Single CE
5	1.48
20	1.34
30	1.05
40	0.98
50	0.87
60	0.85
70	0.82
80	0.81

Table 2 shows the speed up ratio, i. e. the time taken for 4 processors divided by the time taken for a single processor for various size of the problem. It shows the result of computational result where the matrix is decomposed into 4 CE'S, so that the part of matrix vector multiplication in the conjugate gradient step could be done in 4 CE'S concurrently. For smaller porblems, the case of multi-processors took more time than the case of a single processor due to the communication overhead (i. e. the extra time taken for communicating the computation results). Around 35 is the break-even point, at which the savings in computing time due to the parallelization eventually overcomes its communication overhead penalty.

5. Conclusion and further research issues

We proposed the concept of "criterion function", as a frame work for comparing computatinal and geometric characteristics of various LP algorithms. Also, we have looked into the combinatorial nature of least squares problems, and investigate the possible strategies of parallelizing algorithms for least squares problems.

The preconditioned conjugate gradient method as selected as a possible candidate for parallelization, and perfomed some preliminary experiments. From this computatinal experiment, we were able to draw some guidelines for effective parallel implementatin.

Different vairants of Karmarkar's algorithm such as the affine scaling method, Newton's method, and power series expansion are currently under extensive investigation as a possible candidate for efficient parallelization.

6. Bibliography

[1] K. M. Anstreicher, "Linear Programming and the Newton Barrier Flow," Yale University, (1987).

[2] S. Y. Chang and K. G. Murty, "The Steepest Descent Gravitational Method for Linear Programming," Descrete applied Mathmatics, Vol. 25, pp. 211-239(1989).

[3] S. S. Chiu and Y. Ye, "Simplex Method and Karmarkar's Algorithm : A Unifying Structure," EES Department, Standford University, (1985).

[4] P. E. Gill, W. Murrary, M. A. Saunders, J. a. Tomlin and M. Wright "On Projected Newton barrier Mthods for Linear Programming and an Equivalence to Karmarkar's Projective Method." Technical Report SOL 85-11R, Stanford University, (1986).

[5] P. E. Gill, W. Murrary, M. A. Saunders, J. A. Tomlin and M. Wright "A Note on Nonlinear Approaches To Linear Programming," Technical Report SOL 86-7, Standford University, (1986).

[6] N. Karmarkar, "A New Polynomial-Time Algorithm for Linear Programming," *Combinatorica* 4(1984) 373-395.

[7] M. Kojima, S. Mizuno, and A. Yoshise, "A Primal-Dual Interior Point Algorithm for Linear Programming," Technical Report B-188, Department of Information Sciences, Tokyo Institute of Technology, (tokyo, 1987).

[8] C. L. Monma, "Recent Breakthroughs in Linear Programming Methods," Bell Communications Research, (1987).

[9] K. G. Murty, *Linear Programming*, (Wiley, New York, 1983).

[10] K. G. Murty, "The Gravitatinal Method for Linear Programming," *OpSearch* 23 (1986), 206-214.

[11] K. G. Murty, *Linear Complementarity, Linear and Nonlinear Programming*, (Heldermann Verlag, West Berlin, 1988), to appear.

[12] J. Renegar, "A Polynomial-Time Algorithm, Based on Newton's Method, for Linear Programming," Technical Report MSRI 07118-86, Mathematical Sciences Research Institute(Berkeley, CA, 1986).

[13] A. Tamura, H. takehara, K. Fukuda, S. Fujishige and M. Kojima, "A Dual Interior Primal Simplex Method for Linear Programming," Technical Report, Department of Information Sciences, Tokyo Institute of Technolgy, (1987).

[14] M. J. Todd, "Polynomial expected behaviour of a pivoting algorithm for linear complementarity and linear programming problems," *Mathematica Programming* Vol. 35, Ed. I. Adler and R. Saigal (1986) 173-192.

[15] M. J. Todd, and B. P. Burrell, "An Extension of Karmarkar's Algorithm for Linear Programming Using Dual Variables," *Algorithmical* 1 (1986) 409-424.

[16] Y. Ye, "Barrier-Projection in Projective Algorithm for Linear Programming," manuscript, Engineering Economic Systems Department, Stanford University(Stanford, CA, 1985).

[17] Y. Ye, "Cutting-Objective and Scaling Method-'Simple' Polynomial Algorithm for Linear Programming," manuscript, Engineering Economic System Department, Standford University (Standford, CA, 1985).

[18] Y. Ye, and E. Tse, "A Polynomial-Time Algorithm for Convex Quadratic Programming," manuscript, Engineering Economic Systems Department, Standford University(Stanford, CA, 1986).

[19] Y. Ye, "Karmarkar's algorithm and the ellipsoid mthod," *Operations Research Lettes*, Vol. 6, no. 4, (1987) 177-182.