A New Interior Point Algorithm

Alastair McNaughton
Department of Mathematics
University of Auckland
New Zealand
a.mcnaughton@auckland.ac.nz

December 1999

Abstract

This algorithm solves linear programmes by an iterative process which may be likened to a moving point which rebounds off the interior faces of the polytope. The talk will include an analysis of the effectiveness of this algorithm in comparison with established methods.

1 Introduction

A linear programme consists of a linear objective function, $f(x)$, in n variables, which is to be minimised subject to m linear constraints. It has been found that an astonishing number of industrial problems can be formulated in this manner. Geometrically, a linear programme can be pictured as an attempt to minimize $f(x)$ over a set of feasible solutions consisting of a polytope lying in $n-$dimensional space. Each surface of this polytope represents a constraint. In 1947 George Dantzig [1] d! in that it is an exponential algorithm, and hence performs poorly for large pathological data structures.

In recent years many papers have been written proposing interior point methods. The most successful of these has been that of Narendra Karmarkar [2]. An intricate mathematical method is used to navigate the solution iteration by iteration across the interior of the polytope, keeping well clear of the outer surface. The laborious nature of Karmarkar's computations render his method inappropriate for moderately sized problems, even though it is theoretically a polynomial time algorithm. Those many wri

2 Motivation

Dr A. Philpott introduced me to interior point methods in 1993. At that time I formed an interest in developing a method in which the solution would bounce about inside the polytope, each iteration corresponding to a rebound off a certain constraint surface. It was found that such a method required surprisingly simple numerical calculations.

However, my early attempts were thwarted by a tendence for the solution to become trapped in a valley, or in some sub-optimal subspace. Later, when I began to deal w!olytope. Rather than bouncing off the actual constraint surface, which often leads to a subsequent impact on an adjacent surface, it is better to bounce off an imaginary parallel surface closer to the interior. A compass sensor is required to indicate the direction of the interior of the polytope each iteration. The locus of the solution should keep following this direction rebounding back through the centre. the very long distances between imp[acts that result may be used to produce large gains in objecI

> We present a parallel projection algorithm for general programming problems, ACELPAR, whose core is a parallel projection feasibility algorithm (which we speed up by means of a conjugate gradient). ACELPAR is independent from problem structure (unlike most parallel decomposition algorithms). It does not require exact projections, unlike the Boyle-Dijkstra method, and outperforms other recent projection possibilities.

Much of the appeal of this method lies in the creative use of geometrical cognitive structures and its close relationship with concepts from linear algebra. A major departure from traditional interior point methods is the use of the actual polytope surface detail as an aid to interior space navigation.

3 Formulating the linear programme

The linear programme is formulated in a vector form.

$$\text{LP: min} \quad z = -\mathbf{c^T}.\mathbf{x}, \text{ subject to} \qquad \mathbf{a_i^T}.\mathbf{x} \leq \pm \mathbf{a_i^T}.\mathbf{a_i}, \qquad \text{with } i = 1, ..., m,$$

where a_i^T is the i-th row of the matrix A used in the traditional LP formulation. In this paper we will consider the case where the right hand side of each constraint is positive, as this makes it easier to assimilate the geometric concepts to be presented. However, the case when some of these are negative is not difficult, as will be indicated later.

The role of the vector $\mathbf{c}$ in the algorithm will prove to be most significant. It indicates the direction in which objective improvement can be attained. However, it is seldom possible to move directly in this direction due to the many constraints. Instead the locus of the solution will move mostly in directions orthogonal to $\mathbf{c}$, with slight but significant objective gains being made rather analagous to the manner in which a yacht tacks against the wind.

4 Maintaining feasibility

Suppose at the t-th iteration the solution $\mathbf{x_t}$ is feasible, and the chosen direction of advance is given by vector $\mathbf{v}$. We need to know which constraint surface will be encountered first. This is done by solving

$$\mathbf{a_i^T}.(\mathbf{x_t} + k_i\mathbf{v}) = \mathbf{a_i^T}.\mathbf{a_i},$$

for each i, and then selecting the smallest non-negative k_i. Notice that any currently infeasible constraints would be transparent to this procedure. However, once such a constraint surface has been crossed the solution will be prevented from returning to the infeasible side. Consequently the number of infeasible constraints never increases. Once the closest constraint surface to $\mathbf{x_t}$ has been found, along with the distance k_t, it is necessary to decide how far to travel along this direction. Let

$$\mathbf{x_{t+1}} = \mathbf{x_t} + sk_t\mathbf{v}.$$

It is a good idea to start with s about 0.9, to take full advantage of the large steps available, and then to allow s to decrease to 0.5 in the latter stages when the risk of getting trapped against the polytope surface is much higher. Often the initial solution may be infeasible. In this case the vector $\mathbf{a_i}$ corresponding to each of the violated constraints may be used as a pseudo objective vector until the solution crosses this constraint surface.

5 The hyperplane

As noted above, most of the movement of the solution will be orthogonal to the vector $\mathbf{c}$ of objective improvement. Hence it is helpful to consider at each iteration the hyperplane $\mathbf{H_t}$ which contains $\mathbf{x_y}$ and is orthogonal to $\mathbf{c}$. For clarity of thought $\mathbf{H_t}$ should be restricted to its intersection with the polytope. In this way it is seen as a huge subspace of dimension $n-1$, with $\mathbf{x_t}$ an interior point. Ranged around the boundary there will be at least $n!$next rebound. This is achieved by choosing $\mathbf{v_h}$ as the direction of uniform retreat from the projections onto the hyperplane of the $n-1$ most recent rebound surfaces. This is achieved by constructing an n by n matrix, M. Every entry in the first row of M is 1. The second row is the same as $\mathbf{c^T}$. The remaining rows are formed in the following way. Let the orthogonal projections on to H of the rows in the A matrix corresponding to the n-1 most recent rebounds be $\mathbf{a_{hk}^T}$, with$v_h = \mathbf{e_1} -$

$$Noticethatthisimplies$$

v_h is non-zero (from row 1), within $H_t(fromrow2), andmakinganequalanglewiththenornaltoeachof$ to ensure it is leaving rather than approaching the constraint surfaces. This is done by requiring that $\mathbf{a_{hi}}.\mathbf{v_h} \leq 0$. Notice that $\mathbf{v_h}$ is not the final direction, which will be called $\mathbf{v}$, but the projection of $\mathbf{v}$ on to the hyperplane H.

6 Making objective gains

It is desirable that the direction vector $\mathbf{v}$ have a component in the direction $\mathbf{c}$, so that an objective gain is achieved each iteration. However, we still must ensure that it cannot rebound off one of the most recent n-1 surfaces. To do this we calculate the angle of dip, θ_i, for each of the n-1 most recent constraint rebound surfaces,

$$\theta_i = \arccos\left(\frac{\mathbf{a_i}.\mathbf{c}}{|a_i||c|}\right).$$

We then find the smallest of these, with $\theta = \min\{\theta_i\}$. Finally

$$v = v_h + \tan\theta |v_h| \hat{c}.$$

7 Achieving optimality

The sequence of points ¡¡¿ will eventually spiral into rebounds off a closed loop of n surfaces. Once we suspect this stage has been reached, say by the same n rebound surfaces thrice in succession, we merely test the intersection point of these surfaces. If this point is feasible, then it is optimal. If it is not feasible, then the algorithm continues by taking the next direction vector v in the direction of this intersection point. A slight modification is needed for the possible, but unlike

8 Results from numerical trials

The algorithm has been tested by a MATLAB programme. This comfortably solves LPs with up to 100 variables and 500 constraints in around 150 seconds. A much faster computational time is anticipated once the code is translated into fortran or C. It is encouraging that the actual optimal values are attained without recourse to a phase of traditional simplex method at the end.

9 Conclusions

It is not claimed that this is a polynomial algorithm. Nor is it anticipated that computational times will ever seriously threaten those of the revised simplex method. However, this method does have considerable merit from a conceptual point of view. It is possible it may stimulate creative optimisation thinking on the interface between OR and linear algebra.

References

[1] Dantzig, G.B. *Linear Programming and Extensiions*, 1963. Princetown University Press, New Jersey.

[2] Karmarkar, N. *A New Polynomial-time Algorithm for Linear Programming*,1984. Combinatoria 4, pp 373-395.

A.J. McNaughton
Division of Science and Technology

Tamaki Campus
University of Auckland
Private Bag 92019 New Zealand