Author	: P.A. Bruijs
Address	: Gershwinstraat 1, 2807 SG, Gouda, The Netherlands
E-mail	: p.a.bruijs@wxs.nl
Affiliation	: None

A strong polynomial gradient algorithm in Linear Programming

Abstract.
It has been an open question whether the Linear Programming (LP) problem can be solved in strong polynomial time. The simplex algorithm does not offer a polynomial bound, and polynomial algorithms by Khachiyan and Karmarkar don't have the strong characteristic. The curious fact that non-linear algorithms would be needed to deliver the strongest complexity result for LP served as the motivation to look at feasible gradient methods. Experiments and theory show that a linear algorithm has the desired strong characteristic.

Keywords. linear programming - simplex algorithm - ellipsoid method - interior-point methods - gradient methods - computational complexity

1. Introduction

The Linear Programming problem (LP) was presented some 75 years ago. It may be defined in primal standard form as follows : max { $c^T x \mid A x \leq b$, $x \geq 0$ } where A is an m x n matrix, x and c are row vectors in R^n , b belongs to R^m, and rows of A are denoted as $a(i)$, i=1,…,m. Since Dantzig 1950 [2], Khachiyan 1979 [4], and Karmarkar 1982 [5] an impressive body of literature on the problem has been generated. However directly after the publications of Khachiyan and Karmarkar one might have been astonished about the fact, that this linear problem derived its best complexity result from non-linear algorithms. Think for instance of Ockham's famous razor. So it seems natural to look for a linear algorithm offering at least the same complexity result. Looking at the extremely simple definition of the problem we see a couple of purely linear constraints, in combination with the linear gradient of the objective, all elements that are absolutely constant throughout the problem space.
And it is not that hard to devise a simple algorithm that operates on the basis of the gradient. But what would be the profit ? M.J. Todd's paper "The many facets of linear programming" [6] gives an historic overview of various algorithmic developments, but pays almost no attention to gradient-like methods. His comment: 'These try to follow projected gradient directions from one feasible point to another. Since this seems like such a natural strategy, and apparently so obviously superior to edge-following methods, such methods have been proposed numerous times in slightly varying guises.' But the approach never became mainstream, and interestingly, no complexity result for these gradient-like methods is given. Diagnosing the (linear) simplex approach we see that it looks for improvement by scanning basic solutions only, so excluding as tentative candidates many feasible solutions that can be reached directly by a feasible path based on directions close to the gradient. In the ellipsoid approach there is an approximation initially overestimating the relevant feasible space which is divided in half in each iteration by means of a separating hyperplane that has no direct relation to the constraints of LP. In the interior point method there is an approximation underestimating the relevant feasible space in favour of the construction of a direction weakly related to the gradient that may lead far through the interior of the feasible region. So the concepts used in mainstream approaches are: approximated or partial use of the feasible space, separating hyperplanes approximately related to constraints, and feasible directions weakly related to the gradient. Ockham suggests : in your problem you have just a fixed direction to follow, but stay within the constraints that are imposed on your movements ! Probably he had no notion of the concept of a convex space, so that is what will have to be added. Following Ockham's suggestion the first action is to see which move can be made in the direction of the original gradient. If LP is not unbounded a constraint is found that hinders further movement in this direction. Going further would mean entering the non-feasible half-space associated with this constraint. But the exclusion from the gradient of directions leading into this half-space generates a local feasible part of the gradient that is as close as possible to the original gradient. And this 'feasible gradient' fg is a proper basis to move on. In this way it is possible to reach n binding constraints giving fg = 0 because the directions associated with these n constraints span the problem space completely.

The question then : is the optimum solution to LP found ? This can be decided on the basis of Farkas' Lemma [1] saying that in the optimum the following condition must be met : c is in the cone of the normal vectors to the binding constraints, or $c = \sum \mu(i)\, a(i)$ where $\mu(i) \geq 0$, where $a(i)$ is a binding constraint. The multipliers $\mu(i)$ are computed most easily on the basis of an orthogonal representation of the set of binding constraints, that can be found by the Gram-Schmidt orthogonalization procedure. If there is no negative multiplier the optimum has been found, but constraints with a negative multiplier should locally not be considered binding. But then, again locally, a new search space can be opened, and computing *fg* on the basis of a new orthogonal representation of the binding constraints that are left over will result in $fg > 0$. So having reached a basic solution it can be assessed whether the optimum has been reached, and if not so, there will be a new direction *fg* to follow to find a new and better basic solution to LP. The next and probably most crucial question is : how many basic solution will be reached before reaching the solution, the optimum or the conclusion that LP is unbounded. To answer the question the convexity of the feasible region has to be brought on stage. The convexity guarantees that the sets of binding constraints of successive basic solutions differ in at least one constraint, and moreover, as will be shown in the sequel, this at least one constraint is discarded permanently as a candidate for the binding set in the optimum. This means that at most $m+1$ basic solutions are visited. Thereby strong polynomial complexity is guaranteed. To reach this point no other concept is used than available in the definition of LP, so the approach complies completely with Ockham's razor. There is no need to solve a quadratic optimization problem to find 'the best direction' to follow in basic solutions, following *fg* in a kind of 'simultaneous coordinate descent' suffices. There is also no need to construct separating hyperplanes, the original constraints of LP can be used as such.

2. Feasible direction and step-size in the gradient approach

Critical operations in every iteration of a gradient algorithm are first the computation of the available feasible direction, and second the computation of the step size. This section specifies the methods to be used.
The first question to be asked is how far a start solution $x(0)$ can be improved into $x(1)$ simply, by proceeding for a step-size $s(1)$ in the direction of the gradient, the problems object-vector c. So

$$x(1) = x(0) + s(1)\, c$$

where $x(1)$ too has to be a feasible solution, so,

$$a(i)^T x(1) <= b(i) \qquad \text{all } i$$
$$a(i)^T x(0) + s(1)\, a(i)^T c <= b(i) \qquad \text{all } i$$
$$s(1) <= (\, b(i) - a(i)^T x(0)\,) / a(i)^T c \qquad \text{all } i$$

so the maximum step without violating any constraint is

$$\min_i \; (\, b(i) - a(i)^T x(0)\,) / a(i)^T c$$

But by then there is at least one constraint that has to be considered binding, and assume the minimum is found for $i = bi1$, the index of the binding constraint found in step 1, meaning $a(bi1)^T x(1) = b(bi1)$, so there is no more slack that can be used in the direction c of the original gradient. In the meantime, it is clear that for the computation of an improving step one only has to consider those constraints for which one has in this first step $a(i)^T c > 0$ because a feasible solution is already available, so only these are candidates to become binding. In order not to violate the constraint $a(bi1)$ in further steps of our computation, one has to exclude every solution in the non-feasible half-space of this binding constraint. For the best direction that is left now, one has to exclude from the gradient c exactly and exclusively the component directing into this half-space.
The resulting feasible gradient $fg(1)$ in $x(1)$ may then be written as

$$fg(1) = c - \{\, c^T a(bi1) / a(bi1)^T a(bi1) \,\}\, a(bi1)$$

and in $x(1)$ this is truly the best feasible direction. In the next iteration one looks for the best value of

$$x(2) = x(1) + s(2)\, fg(1)$$

thereby determining a new binding constraint $a(bi2)$. In further iterations one will have to exclude the non-admissible half-spaces related to $a(bi1)$ and $a(bi2)$. It is convenient to determine an orthogonal basis consisting of vectors $aog(bi1)$, $aog(bi2)$, etc to span the combined non-admissible space related to binding constraints $a(bi1)$, $a(bi2)$, etc.
This may be done with the Gram-Schmidt procedure

$$aog(bi) = a(bi) - \sum \{ bj < bi \} \{ a(bi)^T aog(bj) / aog(bj)^T aog(bj) \} aog(bj)$$

In this way one has

$$fg(1) = c - \{ c^T aog(bi1) / aog(bi1)^T aog(bi1) \} aog(bi1)$$

$$fg(2) = c - \{ c^T aog(bi1) / aog(bi1)^T aog(bi1) \} aog(bi1) - \{ c^T aog(bi2) / aog(bi2)^T aog(bi2) \} aog(bi2)$$

while in step $k+1$ of the algorithm this will result in

$$fg(k) = c - \sum_{1 \le j \le k} \{ c^T aog(bij) / aog(bij)^T aog(bij) \} aog(bij)$$

Clearly the number of operations involved in these computations is only mildly polynomial related to m and n, while the operations used are just the simple arithmetic operations addition, subtraction, multiplication and division.

3. Computation of the multipliers

The computation of the multipliers in the test for optimality in basic solutions is derived from the fact that, according to Farkas' Lemma, in the optimum c may be written as a positive linear combination of the normal vectors to the constraints in set B of binding constraints with indexset BI

$$c = \sum \{ i \text{ in } BI \} \mu(i)\, a(i) \qquad \mu(i) > 0,\ i \text{ in } BI$$

This at least is the general formulation of the optimality condition found in the literature. The meaning is that, by Farkas' Lemma, a theorem of the alternative, either c is in the cone of the $a(i)$ in B, or there exists a hyperplane separating c from this cone, guaranteeing, that there is a direction away from the cone with a positive projection on c, and this must be a feasible direction. This becomes more clear if Farkas' Lemma is not related to the $a(i)$ but instead to the orthogonal set $aog(i)$, derived directly from the $a(i)$ in B in the present basic solution. But the $a(i)$ and the $aog(i)$ in B equally span the space excluded as infeasible for LP. So applying Farkas' Lemma to the $aog(i)$

$$c = \sum \{ i \text{ in } B \} \mu og(i)\, aog(i) \qquad \mu og(i) > 0,\ i \text{ in } BI$$

will generate the same conclusion, as when applied to the $a(i)$, and a negative multiplier proves the existence of a feasible direction with a positive projection on c. For let $a(ineg)$ be a constraint with a negative multiplier then

$$c^T aog(ineg) = \mu(ineg)\, aog(ineg)^T aog(ineg) < 0$$

meaning that $aog(ineg)$ is locally not binding.
However it should be clear that the $aog(i)$ are constructed in a sequential manner, which implies that $aog(1)$ equals $a(1)$, $aog(2)$ is orthogonal relative to $aog(1)$, $aog(i)$ is orthogonal relative to all $aog(j)$ with $j<i$! With this in mind the computation of the multipliers can be organized in the following way.
To start with we have

$$c = \sum \{ i \text{ in } BI \} \mu og(i)\, aog(i)$$

with il being the last index entering BI we get

$$c^T aog(il) = \sum \{ i \text{ in } BI \} \mu og(i)\, aog(i)^T aog(il) = \mu og(il)\, aog(il)^T aog(il)$$

This is correct because the orthogonality of the vectors aog guarantees that

$$aog(i)^T aog(i) > 0 \qquad \text{all } i$$

$$aog(i)^T aog(j) = 0 \qquad i \text{ not equal } j$$

So

$$\mu og(il) = c^T aog(il) / aog(il)^T aog(il)$$

Now that the multiplier for il is known we may write Farkas'relation as follows

$$c - \mu og(il)\, aog(il) = \sum \{ i \neq il \text{ in } BI(\text{step } 4) \} \mu og(i)\, aog(i)$$

But this relation can be used in a quasi-recursive way to compute the other multipliers, because the left hand side is completely known.

4. A feasible gradient algorithm

The computations of a feasible direction, and a feasible step-size, and the computation of the multipliers resulting from Farkas' Lemma, are relevant in the proposed feasible gradient algorithm.

4.1 Specification of the feasible gradient algorithm

In case a feasible solution $x(0)$ is available one proceeds as follows :

```
(step 1)   initially set k=0, a feasible solution is $x(0)$, the feasible gradient is $fg(0) = c$ and
           identify index-sets $BI(k)$ of binding, and $NBI(k)$ of non-binding constraints,
           initially $BI(k)$ empty, $NBI(k) = \{ i$ not in $BI(k) \}$
(step 2)   compute the orthogonal set $B(k)$ for the binding constraints
           by means of the Gram-Schmidt procedure, so
           $B(k) = \{ aog(i) \mid i,j \text{ in } BI(k) ;\ aog(i)^T aog(j) = 0,\ j \neq i;\ aog(i)^T aog(i) > 0 \}$
(step 3)   compute the actual feasible gradient
           $fg(k) = c - \sum \{ aog(j) \text{ in } B(k) \} \{ c^T aog(j) / aog(j)^T aog(j) \} aog(j)$
(step 4)   if $fg(k)^T fg(k) > 0$ : go to step 5, else
           if $fg(k)^T fg(k) = 0$ : an optimality check must be performed,
           so compute the multipliers on the basis of Farkas' Lemma applied to the $aog(i)$
           - if all multipliers > 0 : stop : the optimal LP solution has been reached
           - if not, remove constraints with negative multiplier from $B(k)$, add them to $NBI(k)$,
           and go to step 2
(step 5)   compute the maximum feasible step away from $x(k)$ along $fg(k)$
           $s(k) = \min_{\{ i \text{ in } NBI(k),\ a(i)^T fg(k) > 0 \}} \{ b(i) - a(i)^T x(k) \} / a(i)^T fg(k)$
           if $s(k)$ is infinite : stop : LP is unbounded
           if the minimum is not unique, select constraint $i$ with $a(i)^T fg(k)$ maximal
           add the proper index in $BI(k)$ and remove it from $NBI(k)$
(step 6)   $x(k+1) = x(k) + s(k)\, fg(k)$
(step 7)   set k = k+1 and go to step2
```

4.2 Computational experience

The performance of the algorithm has been tested on a large number of random LP test problems, in ranges as indicated in Table 1. For n and m <= 100 we tested with 10.000 problems in a series, for 150<=n and m<=300 with 1000 problems, and for n or m=500 with 100 problems.
We compare the number of basic solutions in phase1 plus the number in phase 2 needed to reach the optimum with the number of constraints. The worst case performance in a series for an (n,m) pair is defined in the following indicator:

(maximum number of basic solutions reached in a series in phase1 + idem phase2) / (n+m)

and the results are in Table 1. The interesting characteristic of the results is that the number of basic solutions needed to reach the optimal solution is limited by the number of constraints in LP. This in sharp contrast with the findings communicated by Klee and Minty [3] in relation to the simplex algorithm. Testing the gradient algorithm on Klee-Minty-type problems has been successful too.

		m								
		5	**25**	**50**	**75**	**100**	**150**	**200**	**300**	**500**
n	**5**	1,20	0,90	0,73	0,68	0,64	0,53	0,52	0,480	0,438
	25	0,07	0,56	0,65	0,56	0,51				
	50	0,04	0,21	0,42	0,38	0,45			0,254	
	75	0,03	0,13	0,22	0,27	0,29				
	100	0,02	0,08	0,23	0,17	0,28			0,200	
	150	0,013					0,20			
	200	0,010						0,16		
	300	0,007		0,006		0,005			0,100	
	500	0,004								

Table 1 Sub-linear growth in numbers of basic solutions needed to reach the optimum

But having reached this point it must be concluded that still no definitive result as to the complexity of the proposed approach is available. After the guidance of Ockham, we meet the warning from Popper !

5. Complexity analysis

The complexity analysis of the proposed gradient algorithm has to consider three cases :
5.1 Consider first the situation that the conclusion that LP is unbounded is reached while in every iteration $fg(\mathrm{k}) > 0$. The maximum number of iterations is n, as the dimension of the problem space is n.
5.2 Consider next the situation that all multipliers are positive the first time a solution is reached where $fg(\mathrm{k}) = 0$. As the dimension of the problem space is n, this number will be exactly equal to n.
5.3 Consider now the case that $fg(\mathrm{k}) = 0$ and there is at least one negative multiplier. The conclusion must be, that the optimum has not yet been reached, cf. Farkas' Lemma, and according to the algorithm we will now remove from $B(\mathrm{k})$ all constraints with a negative multiplier.

Now what should happen next ? And the answer is based on the convexity of the feasible region.

Lemma 1 If the algorithm finds basic solution $x(\mathrm{k})$ with binding set $B(\mathrm{k})$ and basic solution $x(\mathrm{l})$ with binding set $B(\mathrm{l})$ where $c^{\mathrm{T}} x(\mathrm{k}) < c^{\mathrm{T}} x(\mathrm{l})$ then $B(\mathrm{k}) \neq B(\mathrm{l})$.
Proof Suppose an improved basic solution $x(\mathrm{l})$ would exist while $B(\mathrm{k}) = B(\mathrm{l})$, then this would mean that $x(\mathrm{l})$ and $x(\mathrm{k})$ are on the same binding constraints in the convex hull of the feasible region of LP. Because of the convexity of the feasible region we must conclude that every solution on the line connecting $x(\mathrm{k})$ and $x(\mathrm{l})$ is a feasible solution, and because the algorithm does not exclude any feasible solution from the set of possibly optimal solutions, there was no reason in $x(\mathrm{k})$ not to use this line to improve $x(\mathrm{k})$ into $x(\mathrm{l})$. But this is in contradiction with the fact that in $x(\mathrm{k})$ the direction $x(\mathrm{l}) - x(\mathrm{k})$ with a positive projection on c has not been identified as a feasible direction of improvement.□

Lemma 2 If at least one of the constraints in $B(\mathrm{k})$ has a negative multiplier then at least one of the constraints in $B(\mathrm{k})$ does not appear in the binding set $B(\mathrm{opt})$ in the optimum.
Proof From the fact that one or more negative multipliers are found for constraints in $B(\mathrm{k})$ it must be concluded, on the basis of Farkas' Lemma, that $x(\mathrm{k})$ is not the optimum. So $c^{\mathrm{T}} x(\mathrm{k}) < c^{\mathrm{T}} x(\mathrm{opt})$ and by Lemma 1 $B(\mathrm{k}) \neq B(\mathrm{opt})$.□

Lemma 3 Finding an improved basic solution $x(\mathrm{l})$ asks for disappearance of at least one and possibly more constraints from $B(\mathrm{k})$, thereby forcing the same number of non-binding constraints relative to $B(\mathrm{k})$ to enter $B(\mathrm{k}+1)$.
Proof By Lemma 2 at least one constraint in $B(\mathrm{k})$ gets discarded as a candidate for $B(\mathrm{opt})$, and for every basic solution the binding set necessarily consists of n binding constraints.□

Lemma 4 The maximum number of basic solutions the feasible gradient algorithm can find is m+1.
Proof Lemma 2 asks for the disappearance of at least one constraint as a candidate for $B(\mathrm{opt})$, while Lemma 3 states the necessity of the entrance of at least one formerly non-binding constraint to replace this at least one constraint. In the first basic solution we have n constraints in $B(1)$, leaving m constraints that may replace constraints that are discarded as candidate for $B(\mathrm{opt})$.□

At this point the answer to the question what should happen next is available : follow the receipt of the feasible gradient algorithm, and if this is done the conclusion is, that at most m+1 basic solutions will be reached. Thereby a strong fundament for a strong complexity result has been identified.

Lemma 5 (Iterations) The feasible gradient algorithm has an absolute bound of $n + m\,n$ iterations.
Proof To reach the first basic solution n iterations are needed, and to reach the at most m next basic solutions no more than $m\,n$ iterations will be necessary.□

Lemma 6 (Arithmetic Operations) The number of arithmetic operations per iteration of the feasible gradient algorithm is polynomially related to m and n.

Proof From the definition of the algorithm we derive a maximum to the number of arithmetic operations for every step of the algorithm in every single iteration: whereby the inner product of n-vectors involves n multiplications, and whereby simple addition, subtraction, and comparison are of the same order. So the effort per iteration is polynomial related to m and n, being $O((m+n)n)$. □

The results presented in the preceding Lemma's logically lead to the main theorem :

Theorem (Strong Polynomiality) The feasible gradient algorithm offers a strong polynomial bound for the solution of Linear Programming problems.

Proof In the solution process every set of binding constraints $B(k)$ is generated by the process of successive selection of the strongest constraining constraint until the binding set contains n constraints and a basic solution is reached, or the problem is found to be unbounded, in step 4 resp. step 5 of the algorithm. According to Lemma 6 (Arithmetic Operations) and the fact that at most n iterations are needed to reach this situation the effort so far is clearly polynomial in m and n .
On reaching the first or any subsequent basic solution in step 4 with $fg = 0$, the computation of the multipliers corresponding with the constraints in the current binding set produces the information needed to decide whether a final solution has been reached.
The multipliers may turn out to be all positive or some may prove to be negative, however, Lemma 4 guarantees that no more than $m+1$ basic solutions will be reached. Furthermore Lemma 5 (Iterations) guarantees that to reach these basic solutions no more than $n + n\,m$ iterations have to be performed.
Taken together with the resulting complexity in Lemma 6 (Arithmetic Operations) this results in a total complexity for arithmetic operations and iterations of the feasible gradient algorithm of $O((n + n\,m)(m + n)n)$, and the complexity of the algorithm in a Turing model of computation is

$$O(\ n^3m + n^2m^2\ ) \text{ on } O(L) \text{ numbers } □$$

The complexity measure derived for the algorithm gives a pessimistic bound. An initial guess has led to the idea that only $m+n$ iterations would be sufficient to reach the optimum. However, this idea has been refuted, although the results in many random problems have numbers of iterations close to $m+n$.

6. Conclusion

A feasible gradient algorithm answers a longstanding question concerning the complexity of LP. The approach may generate new results for linear problems, in theory as well as in computational practice.

7. Acknowledgement

Discussions with M.J. Todd, J.K. Lenstra, several referees and especially my fruitful discussions with Kees Roos and his indispensable opposition against my intuition, were very helpful.

References

1 J. Farkas. Ueber die Theorie der einfachen Ungleichungen.*J. Math*, 124:1-24, 1902
2 G.B. Dantzig. *Linear Programming and extensions*. Princeton University Press, 1963
3 V. Klee and G.J. Minty. How good is the simplex algorithm ? In O. Shisha, editor, *In equalities III*, 159-175, Academic Press, 1972
4 L.G. Khachiyan. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady*, 20:191-194, 1979
5 N.K. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373-395, 1984
6 M.J. Todd. The many facets of linear programming. *Mathematical Programming*, 91:417-436, 2002