EXPERIMENTS WITH THE REDUCED GRADIENT METHOD FOR LINEAR PROGRAMMING

Markku Kallio
William Orchard-Hays

September 1979
WP-79-84

INTERNATIONAL INSTITUTE FOR APPLIED SYSTEMS ANALYSIS
A-2361 Laxenburg, Austria

ABSTRACT

This article deals with some methods for linear programming which generate a monotonically improving sequence of feasible solutions. Examples of such methods are the simplex method and the reduced gradient method. A larger class of such methods as well as their convergence has been discussed in a recent article by Kallio and Porteus.

We have implemented a version of such methods in the SESAME system developed by Orchard-Hays. This version resembles the reduced gradient method except that only a subset of nonbasic variables to be changed is considered at each iteration. We shall try out several modifications of this basic version. These modifications are concerned with the choice of an initial basis and an initial solution, with strategies for finding a feasible solution, as well as with strategies for determining the direction of change for a feasible solution at each iteration.

We have experimented with moderate sized nonstructured as well as dynamic problems. Compared with the simplex method, the overall performance of such methods is about equal in the case of linear programs with no particular structure. For dynamic LP we have obtained some encouraging results. Although we have been able to experiment with only a few problems, so far it seems that using specially defined starting basis and initial nonbasic

solution allow a reduction by a factor of eight in the computing time of the reduced gradient method. This starting basis is chosen so that its columns are likely to appear also in an optimal basis. For the initial solution, available information, such as current level of activities in real life, may be employed. Of course, our starting basis for dynamic LP may be used also in the simplex method, and, indeed this results in a considerable improvement of efficiency.

EXPERIMENTS WITH THE REDUCED GRADIENT METHOD FOR LINEAR PROGRAMMING

M. Kallio and W. Orchard-Hays

1. Introduction

Consider the linear program (LP):

$$\text{find } x \in R^n \quad \text{to}$$

$$\text{(LP1)} \qquad \text{maximize} \quad cx$$

$$\text{(LP2)} \qquad \text{subject to } Ax = b$$

$$\text{(LP3)} \qquad 0 \leq x \leq u \quad ,$$

where $c, u \in R^n$, $b \in R^m$, and $A \in R^{m \times n}$ is of full row rank. For solving (LP) we shall consider methods, which can be characerized as follows: Like the simplex method [1], these methods move from one feasible solution to another at each iteration, thereby improving the objective function. Each feasible solution is also associated with a basis. However, this feasible solution need not be an extreme point and the basic solution corresponding to the associated basis need not be feasible. Nevertheless, as shown in [2], an optimal solution, if one exists, can be found in a finite number of iteration (under nondegeneracy).

In the following, we shall first review this class of methods as presented in [2]. Thereafter, we discuss an implementation of such methods in the SESAME system, an interactive mathematical programming system developed by Orchard-Hays [4] and written in assembler language for IBM 370 under VM/CMS. In the last two sections we shall report experiments which we carried out both for nonstructured and for dynamic linear programs (LP).

2. The Class of Methods

We shall now review the methods in consideration as presented in [2]. We call x a system solution if it satisfies (LP2), a homogeneous solution if it satisfies $Ax = 0$, and a feasible solution if it satisfies (LP2) and (LP3). If x is feasible and z is a homogeneous solution, then $x + \theta z$ is feasible as long as it is nonnegative, for all $\theta \in R$. As θ increases, the objective function increases if and only if $cz > 0$. The simplex method chooses as z one of the homogeneous solutions corresponding to increasing the value of a nonbasic variable such that cz, the reduced cost, is positive. The methods considered here may choose as z a linear combination of such vectors, rather than just one. In particular, the direction may be chosen according to the reduced gradient method (e.g. [7]). As in the simplex method, a new feasible solution is found by increasing θ (and the objective function) as much as possible without losing feasibility.

The Admissible Directions

Before stating the method, we shall discuss how an admissible direction is constructed. Let β denote the set of basic indices (indices for basic variables), and let α and γ be sets of variables

at their lower and upper bounds at x, respectively; i.e.

$$\alpha = \alpha(x) = \{i \mid x_i = 0\} \quad \text{and}$$
$$\gamma = \gamma(x) = \{i \mid x_i = u_i\} \quad . \tag{1}$$

In the simplex method, all nonbasic variables would be in $\alpha \cup \gamma$, but this is not necessarily the case here. For convenience, assume that the variables have been ordered so that $\beta = \{1,2,\ldots,m\}$. Let B be the corresponding basis matrix, and let a^j denote the j^{th} column of A. For each nonbasic variable $j \in \bar{\beta}$ (the complement of β) define a column vector $z^j \in R^n$ componentwise as follows:

$$z_i^j = \begin{cases} -(B^{-1}a^j)_i & \text{if } i \in \beta \quad , \\ 1 & \text{if } i \in \bar{\beta} \text{ and } i = j \quad , \\ 0 & \text{otherwise} \quad . \end{cases} \tag{2}$$

Clearly, z^j is a homogeneous solution, since $Az^j = B(-B^{-1}a^j) + a^j = 0$. As mentioned before, z^j serves as the direction of change in the simplex method, when changing the value of a nonbasic variable j. For the methods considered here, linear combinations of such vectors serve as such directions z; i.e., if $Z = z(z^j)$ is the $n \times (n-m)$ matrix having vectors z^j as its columns and w is an (n-m)-vector of weights, then

$$z = Zw \quad . \tag{3}$$

We shall index the components of w by nonbasic variables rather than the first n - m integers. Thus, reference to w_j always carries the convention that $j \in \bar{\beta}$. Taking (2) into account, the components w_j indicate the direction of change in the space of

nonbasic variables while z is the direction in the space R^n of all variables.

In general, certain conditions are to be met by an admissible direction in order for the method to converge: (i)For the direction to be feasible, we require (for a nonbasic variable j currently at its bound) that $w_j \geq 0$ for $j \in \alpha$ and $w_j \leq 0$ for $j \in \gamma$. (ii)In order to improve the objective function, we must have $cZw > 0$. (iii)Finally, in order to prevent zig-zagging, we require that $cz^j w_j > 0$ if $w_j \neq 0$. If no $w \in R^{n-m}$ satisfies conditions (i) - (iii), then the current solution is optimal for (LP). (For a proof, see reference [2].)

In the simplex method, an admissible direction w is a unit vector for which cZw is positive or negative depending on whether the particular nonbasic variable is currently on its lower or upper bound. For the reduced gradient method, w is given by

$$w_j = \begin{cases} 0 & \text{if } j \in \alpha \text{ , and } cz^j < 0 \text{ , or} \\ & j \in \gamma \text{ , and } cz^j > 0 \text{ ,} \\ cz^j & \text{otherwise .} \end{cases} \qquad (4)$$

That is, nonbasic variables are adjusted in proportion to their reduced costs unless they are currently at a bound and a feasible movement off from the bound will not increase the objective function.

The Basis Change

Initially, any basis can be chosen independently of the initial solution. At an iteration, if a nonbasic variable moves to its bound, then we simply leave the basis unchanged. Otherwise, at least one basic variable reaches its lower or upper bound.

We may arbitrarily[1] select one of these to be the leaving variable ℓ. For the entering variable, there may be many candidates: any variable e is a candidate if it is currently off from its bounds (i.e. $0 < x_e < u_e$) and $\beta' = \beta \cup \{e\} - \{\ell\}$ is a legitimate set of basic variables. It has been shown in [2], that if (LP) is nondegenerate, then such a variable e always exists. Implementation of the basis change rule will be dicussed in Section 3 in detail.

The Method

The steps of the methods in consideration can be stated as follows:

1° Initialization: Specify an initial basis (set of basic variables β), an initial feasible solution x and the corresponding sets $\alpha = \alpha(x)$ and $\gamma = \gamma(x)$.

2° Specify direction: Determine a vector w of weights satisfying conditions (i) - (iii) above. If none exists, then stop (the current solution x is optimal).

3° Determine step size: Let $\bar{\theta}$ be the largest θ for which $x + \theta Zw$ is feasible. If $\bar{\theta} = \infty$, then stop ((LP) is unbounded).

4° Update: Replace x by $x + \bar{\theta}Zw$. Thereafter,

4.1° if any of the nonbasic variables moved to its upper or lower bound, update α and γ, and return to 2° (without a basis change);

4.2° otherwise, update α and γ, and pick any $\ell \in \beta \cap (\alpha \cup \gamma)$

---

[1]Actually, standard pivot selection rules are used.

(a basic variable on its bound) as leaving variable. Pick $e \in \overline{\beta} \cap \overline{\alpha} \cap \overline{\gamma}$ (a nonbasic variable off from its bounds) such that $\beta' = \beta \cup \{e\} - \{\ell\}$ is a legitimate set of basic variables. Replace β by β' and return to 2°.

3. Implementation: The Basic Version

The SESAME system was modified for adopting the features of the method described above. We shall describe an implementation which later will be referred to as the basic version. In subsequent sections we report computational experience with the basic version as well as with several of its modifications.

Shortly stated, the basic version is just the reduced gradient method modified so that only a certain subset of nonbasic variables is considered for changing at each iteration. We shall first give a brief overview of the SESAME system. Thereafter, following the steps listed for the method in Section 2, we shall discuss details of our implementation. Such a discussion ought to be useful when we consider alternative implementations for these particular steps in subsequent sections.

The SESAME System

The SESAME mathematical programming system is a large MPS with simplex algorithms and supporting procedures in traditional style. Its grandparentage is partly IBM's MPS/360 (the second-generation antecedent of MPSX/370) and its parentage partly Management Science System's (now Ketron) MPS-III [5]. In other words, with respect to algorithms, it is on the main branch of development of large commercial MPS's. SESAME includes an

elaborate data management extension, called DATAMAT, which has very similar external (but not internal) specifications to MPS-III's DATAFORM. Both these extensions are the outgrowth of several lines of development going back as far as 1959 [3].

In most other attributes, however, SESAME is unique. This is particularly true in two respects. First, it was developed at the National Bureau of Economic Research's Computing Research Center (now part of MIT), not as a commercial product but to be available as both a production and research tool to universities, research centers and other nonprofit organizations. Of perhaps more importance here, it was designed from the beginning for use only on an interactive host, namely an IBM/370 operating under VM/CMS. While this restricts its portability, specialization to one type of computer enhances efficiency as with all other large MPS's. Availability has been made broad through its access on standard networks. Both SESAME and, particularly, DATAMAT have been enhanced and extended at IIASA, utilizing the IBM 370/168 at the CNUCE center in Pisa, Italy, via remote terminals and high-speed file transmission facilities. Indeed, the entire development of SESAME since 1972 has been done remotely. At no time did the development team have "hands-on" access to the computer on which the work was being done.

SESAME is controlled by the user through and only through a remote terminal. There is no such thing as "submitting a job." In fact, however, the user creates standard sequences of instructions--at various levels--in the form of files which are then invoked by a command at the terminal. The creation, modification and invocation of these "run" and "program" files are all done

interactively as well as ad hoc use of various system facilities. The whole arrangement is very versatile and system modifications and extensions are carried out in the same style (but restricted to knowledgable professionals). A number of difficult models have been handled at IIASA which would have been virtually impossible with batch methods.

The main simplex algorithm in SESAME combines the primal, dual, generalized upper bounding (GUB) and separable programming all in one procedure. It also includes bounds and ranges of all types, multiple and partial pricing, and a number of algorithm control switches. (Multiple pricing and suboptimization is permanently limited to seven columns, which becomes important below). Both standard MPS input and MPS-III extensions as well as another better but little-used format are accepted. Most models, however, are created with DATAMAT which enfiles them directly without an intermediate card-image form. Standard output of the various usual kinds is provided and, additionally, LP results may be enfiled directly for subsequent use with DATAMAT functioning as a report generator or master algorithm control. The system includes a number of other features which are of no particular pertinence here.

Initialization of the Method

We shall now turn our discussion to the implementation of our basic version of the reduced gradient method in the SESAME system. For the basic version, either an all logical starting basis (i.e. a basis consisting of slacks and artificials only) can be constructed or an advanced basis is loaded. The latter

alternative is available if a basis from previous runs has been saved or if such a basis has been generated by other means. However, no crash algorithm has been employed.

The initial solution of the basic version is the basic solution corresponding to the initial basis. If this solution is not feasible, we start Phase I in the usual way for minimizing the sum of infeasibilities. Thus in this case, the objective function coefficient is set to -1 for all variables above their upper bound (including artificial variable at a positive value), 1 for all negative variables and to 0 in other cases.

Specifying Direction

At each iteration we consider at most k = 7 nonbasic variables to be changed simultaneously. In the following, this set is called the k-set. The maximum number of elements in the k-set was due to an implementation similar to one employed for a multiple pricing procedure in the SESAME system. In such a case, the alpha columns (the columns a^j premultiplied by the basis inverse) for nonbasic variables j to be moved are stored explicitly, and core limitation soon becomes prohibitive for larger k.

While choosing the k-set we cycle the nonbasic variables similarly to what is normally done in an implementation for the simplex method. We need to find, if possible, a set of t (standard value of t = 12) nonbasic variables, called the t-set, for which formula (4) of the reduced gradient method yields a nonzero weight w_j. Among the t-set we choose, when possible, k variables with the largest weights in absolute value. The optimum for (LP) has been obtained if the t-set is empty.

After choosing in this way the k-set from the set of all nonbasic variables, we set the weights according to (4) and move in this direction. If a nonbasic variable (one or more) becomes binding, we redefine its weight according to (4). Otherwise, a basic variable ℓ having moved to its bound is replaced by a variable e of the k-set. Thereby the size of the k-set is reduced by one element. We repeat such iterations until either the k-set becomes empty or the weights for all variables in the k-set are equal to zero. Thereafter, a new k-set (of at most 7 variables) is chosen among the nonbasic variables as described above.

Determining the step size

As indicated above, the alpha-columns for all nonbasic variables in the k-set are stored explicitly. When a new k-set is chosen, an FTRAN pass is needed to compute these alpha-columns. Otherwise, the existing alpha-columns are just updated in the usual way utilizing the alpha-column of the entering variable. Given the alpha-columns, a composite column is computed as a weighted sum of these vectors, the weights being those given by the direction w.

For Phase II, the minimum ratio test is carried out using the composite vector as usual to determine the step size. For Phase I, however, there are several alternatives. The rule adopted in our basic version is to move as far as (i) a currently feasible variable reaches its bound, or (ii) an infeasible variable, moving towards feasibility, reaches its farthest finite bound, whichever occurs first.

Updating the basis inverse

The basis inverse is stored in a product form and, given a leaving and an entering variable, updated exactly as in the simplex algorithm of the SESAME system. In our case, however, there is some freedom in choosing the entering variable. As shown by the following result,we may exclude from consideration all non-basic variables which are not in the k-set.

Lemma. Let $\ell \in \beta$ be a basic variable becoming binding at the current iteration. Then there exists in the current k-set a variable e such that $\beta' = \beta \cup \{e\} - \{\ell\}$ is a legitimate set of basic variables, and such that the updated price vector corresponding to β' is (dual) feasible for column ℓ.

Proof: Let d_j be the reduced cost and α_ℓ^j the element of the alpha-column j in pivot row ℓ, for each j in the k-set. If basic variable ℓ is forced to its lower bound, then there must be a variable j in the k-set for which either $d_j > 0$ and $\alpha_\ell^j > 0$ or $d_j < 0$ and $\alpha_\ell^j < 0$. On the other hand, if ℓ is forced to its upper bound, there exists variable j, for which either $d_j > 0$ and $\alpha_\ell^j < 0$ or $d_j < 0$ and $\alpha_\ell^j > 0$. In each case one can readily check that the result follows. ||

Among all candidates e implied by this Lemma, we choose as the entering variable the one off bound with the largest pivot element. If this element is within the range of a pivot tolerance (standard threshhold is 10^{-8}) the variable with the largest pivot element among all columns suggested by our Lemma is chosen. If both fail, this can only be due to digital difficulties, and no provision has been implemented to avoid this, except the possibility to change the tolerance.

4. Computational Experience: Nonstructured LP

4.1 Test Problems

The following test problems were considered: a tiny oil refinery model (A), agricultural planning models (B), (C) and (D), an energy supply model (E), and dynamic forest sector models (F) and (G). Statistics concerning these test problems is given in Table 1 below.

Table 1. Summary of test problems.

Problem	Rows	Columns	Density (%)
A	23	23	18.1
B	451	507	1.0
C	476	532	1.0
D	152	218	6.1
E	50	165	12.1
F	521	612	0.6
G	2321	3188	0.14

4.2 Results with the basic version

Table 2 below shows computational results of our basic version compared with the simplex method (as implemented in the SESAME system).

Table 2. Experience with the basic version of the reduced gradient method compared with the simplex method of SESAME.

	Reduced gradient method					
Problem	A	B	C	D	E	F
Initialization:						
Infeasibilities	4	58	0	32	13	81
Basic variables equal to zero	13	266	48	93	21	362
Feasible solution:						
At iteration	26	-	1700+	288	47	976
Optimal solution:						
At iteration	28	400*		444	106	1462
Basic variables equal to zero	0	3		16	10	20
Nonbasic variables not on bound	1	15		4	1	25

	Simplex method					
Problem	A	B	C	D	E	F
Feasible solution:						
At iteration	23	-	1175	171	40	818
Optimal solution:						
At iteration	25	360*	1688	293	105	1085

*the problem was found to be infeasible.
+run was interrupted without finding a feasible solution.

In each case, we have started with an all logical basis and the initial solution is the corresponding basic solution. The initial number of infeasibilities is shown, and the number of iterations required for reaching a feasible solution as well as an optimal solution is given. Furthermore, a measure for primal degeneracy is given for the initial and optimal solution in terms of the number of basic variables equal to zero. We shall refer to this measure in subsequent sections.

As a measure for computational efficiency, the number of iterations, or rather the number of basis changes, may be used. For the reduced gradient method we did not count the minor iterations when a nonbasic variable moves to its lower or upper bound (the case without a basis change). On the other hand, an iteration is counted for the simplex method, when a nonbasic variable is moved from one bound to another. An experiment was carried out on Problem F, which shows that the average CPU time per iteration for the reduced gradient method is .8 times that for the simplex method. Thus, to make the number of iterations comparable measures for computational efficiency, the iteration numbers in Table 2 for the reduced gradient method should be multiplied by a factor of .8.

According to Table 2, the overall performance of the basic version of the reduced gradient method is about equal compared with the simplex method of the SESAME system. (The difficulty in finding a feasible solution to problem C is unexplained. The source of the model is obscure and no investigation was possible).

4.3. Choosing a Nonbasic Starting Solution

Because the right hand side vector b normally is a relatively sparse vector, the initial solution is highly degenerate, when an all logical starting basis is chosen. This in turn results in a large number of iterations with a step size equal to zero. The ratio of such iterations for problems B and D, for instance, was more than 50 percent, most of which occured during the early iterations for both of the methods. In the following we report a little study, where we consider an approach for avoiding this phenomenon and investigate whether something can be gained in doing so.

Basically, our approach is to start the reduced gradient method with a nonbasic solution. We try to provide some motivation for this approach through the following example, which has been illustrated in Figure 1.

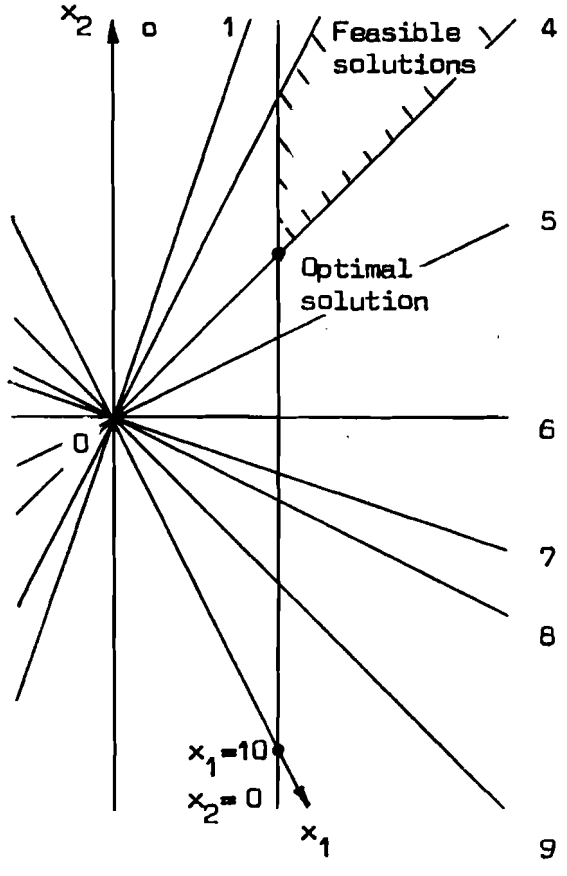

Figure 1. A degenerate, all logical starting basis.

$$
\begin{aligned}
\text{minimize} \quad & -x_1 + s_1 = -10 \\
\text{subject to} \quad & -5x_1 + x_2 + s_2 = 0 \\
& -4x_1 + x_2 + s_3 = 0 \\
& 3x_1 - x_2 + s_4 = 0 \\
& 5x_2 - 2x_2 + s_5 = 0 \\
& 2x_1 - x_2 + s_6 = 0 \\
& 5x_1 - 3x_2 + s_7 = 0 \\
& 3x_1 - 2x_2 + s_8 = 0 \\
& x_1 - x_2 + s_9 = 0 \\
& 2x_1 - x_2 + s_{10} = 0 \\
& x_1, x_2 \geq 0 \quad , s_i \geq 0 \text{ for all } i.
\end{aligned}
$$

The origin $(x_1, x_2) = (0, 0)$ in the picture corresponds to the basic solution for an all logical starting basis which is comprised by the (columns of the) slacks s_i. This solution is highly degenerate as nine out of ten of the basic variables are equal to zero. There is only one infeasibility ($s_1 = -10$). When the standard simplex method or our basic version is used, either 2,3,4,5,6, or 7 iterations are required, depending on the choice of alternative pivot paths, to reach the optimal solution $(x_1, x_2) = (10, 10)$. For all the iterations, except the last one, the step size is equal to zero and the resulting solution is the same as the starting solution.

For the reduced gradient method, we may choose a nonbasic starting solution. For instance, we may choose the starting basis as above, set the nonbasic variables to any nonnegative value, and solve (LP2) for the basic variables to obtain a nonbasic system solution to start with. In particular choosing any such point, other than the origin, the number of iterations to reach the

optimum is either 2 or 3, depending on the choice. Thus, it seems likely that starting with a nonbasic solution results in a decrease in the number of iterations in this example. Notice, that the number of infeasibilities at such a starting solution ranges between 0 and 7. (For brevity, we shall not discuss the possible pivot paths here).

We shall now add to our basic version the possibility of setting nonzero values to the nonbasic variables at the starting solution (given that the initial basis has already been chosen). Because, in general, no indication may be available as to which values should be used, we have implemented the possibility of setting the same arbitrarily chosen nonnegative value for all nonbasic variables.

Table 3 below shows the effect of starting with such nonbasic solutions. As a general observation, we may conclude that setting all nonbasic variables initially to a given nonzero value indeed yields a slight improvement (but not in that degree which might be suggested by our example). The number of iterations with a stepsize equal to zero was decreased dramatically, and thereby the functional value both in Phase I and in Phase II improved smoothly.

4.4 Improving the Functional Value in Phase I

The fact that the feasible solution generated in Phase I is often a relatively poor solution, led us to try to take into account also the functional when choosing the direction in Phase I. We shall report such an experiment as well as another attempt aimed at improving Phase I in the following.

Table 3. Starting with a nonbasic solution and an all logical basis.

	Initial solution			Feasible solution		Optimal solution	
Problem	N.b. value	Degeneracy	Infeasibilities	Iteration	Functional	Iteration	Functional
D	0	93	32	288	-7.3×10^{7}	444	6.0×10^{7}
D	1	2	115	233	-7.6×10^{7}	366	6.0×10^{7}
D	10	2	115	217	-7.4×10^{7}	367	6.0×10^{7}
D	100	3	115	220	-7.2×10^{7}	411	6.0×10^{7}
F	0	362	81	976	-2.1×10^{9}	1462	-5.9×10^{7}
F	1	9	322	986	-2.1×10^{9}	1475	-5.9×10^{7}
F	10	9	354	1101	-5.1×10^{8}	1264	-5.9×10^{7}
F	100	7	338	961	-6.0×10^{8}	1064	-5.9×10^{7}

N.b. value = initial value for nonbasic variables; Degeneracy = initial number of basic variables equal to zero; Feasible solution = number of iterations for feasibility and the functional value; Optimal solution = number of iterations for optimality and the functional value.

Our intention now is to specify the vector of weights w for the direction z = Zw in such a way that, in Phase I, improvement is made for the functional value cx as well as for the sum of infeasibilities.

Let c^1x denote the objective function of an ordinary Phase I. We shall now replace this objective by $(c^1 + \lambda c)x$, where λ is a positive parameter. Each time, when optimality has been reached with this objective function, and there are still infeasibilities left, we switch back to the ordinary Phase I routine and stay there as long as the solution remains optimal subject to the modified objective.

The results of our experiments were negative: our general observation was that the total number of iterations for reaching optimality increased considerably; e.g., by fifty percent for Problem F when the standard version was used. Typically, the primal objective function improved well along the Phase I iterations, even reaching the neighborhood of the optimal value, but then a switch to the ordinary Phase I resulted in a large degradation in the functional value.

As another attempt to improve Phase I we implemented a procedure for choosing the step size at each iteration in such a way that the sum of the values for infeasible variables is minimized. We denote this sum as a function of step size θ by $\zeta(\theta)$.

A typical picture of such a function is shown in Figure 2. It is a convex, piece-wise linear function whose derivative is discontinuous at points θ_0, θ_1, θ_2, etc. At each of these points one or more variables become either feasible or infeasible. The minimization of this function, subject to the requirement that

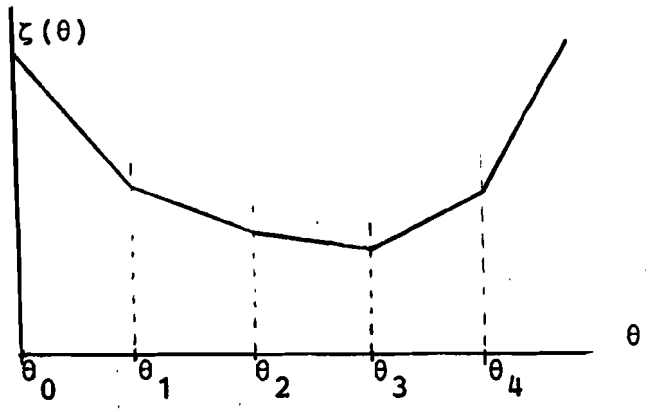

Figure 2. Sum of infeasible variables as a function of step size.

the nonbasic variables are not allowed to become infeasible, can be done easily because the information needed to compute the slope changes at each of the points θ_i, is readily available in the composite vector $z = Zw$.

Somewhat surprisingly, the approach was also a setback compared with the basic version. Again, the suboptimization caused an increase in the number of iterations for reaching feasibility.

5. Specialization for Dynamic Linear Programming

In this section, further elaboration is made on choosing an initial nonbasic solution as well as an initial basis in the case of dynamic linear programming.

5.1 The Dynamic Linear Programming Problem

The dynamic linear programming problem (DLP) is an important special case of (LP). At the same time, it is known as a particularly difficult class of LP problems. The problem can be stated as follows [6]:

find a control $u = \{u(0), u(1), \ldots, u(T-1)\}$, and

a trajectory $x = \{x(0), x(1), \ldots, x(T)\}$ to

(DLP1) maximize $\sum_{t=0}^{T-1} (a(t)x(t) + b(t)u(t)) + a(T)x(T)$

subject to

(DLP2) $x(t+1) = A(t)x(t) + B(t)u(t) + s(t)$,

for $t = 0, 1, \ldots, T-1$

(DPL3) $G(t)x(t) + D(t)u(t) = f(t)$

for $t = 0, 1, \ldots, T-1$

(DLP4) $u(t) \geq 0$, $x(t) \geq 0$ for all t ,

and with the initial state

(DLP5) $x(0) = x^0$.

Here $x(t) \in R^{nt}$ is the vector of state variables at the beginning of period t, for $t = 0, 1, \ldots, T$, and $u(t) \in R^{rt}$ is the vector of control activities during period t, for $t = 0, 1, \ldots, T-1$. For each t, $a(t) \in R^{nt}$, $b(t) \in R^{rt}$, $s(t) \in R^{mt}$ and $f(t) \in R^{kt}$ are externally given vectors, and A(t), B(t), G(t) and D(t) are externally given matrices of appropriate dimension. The initial state of the system is described by the vector $x^0 \in R^{n0}$. The objective function in (DLP1) is a linear function of state variables x(t) and control variables u(t). Constraints (DLP2) may be called the state equations, as they determine the state x(t+1) at the end of a period t (beginning of the subsequent period t+1) given the initial state x(t) and the control action u(t) for that period.

Clearly, (DLP) is a special case of (LP). The constraint matrix A for (DLP) has been illustrated in Figure 3 for T = 3.

$$A = \begin{array}{|c|c|c|c|c|c|c|}
\hline
G(0) & D(0) & & & & & \\
\hline
-A(0) & -B(0) & I & & & & \\
\hline
 & & G(1) & D(1) & & & \\
\hline
 & & -A(0) & -B(1) & I & & \\
\hline
 & & & & G(2) & D(2) & \\
\hline
 & & & & -A(0) & -B(2) & I \\
\hline
\end{array}$$

Figure 3. A dynumic LP with three time periods.

In the following, we shall experiment with ideas of choosing an initial basis and an initial solution, when the reduced gradient method is applied to (DLP).

5.2. An Advanced Basis for Dynamic LP

For dynamic linear programs, it may seem intuitively appealing that most of the state variables appear in the optimal basis. In fact, for various versions of DLP Problems F and G, over 90% of the state variables appear in the optimal basis. Furthermore, we believe that in a typical dynamic LP formulation, besides the state equations (DLP2), there are only a relatively small number of constraints of equality type; i.e., most of the constraints (DLP3) are just inequalities which have been converted to equalities through adding the slack variables. For Problem F, 95% of constraints (DLP3) are converted inequalities. For problem G this ratio is 80%.

These remarks led us to construct an advanced triangular basis which consists of (i) columns of all state variables, (ii) columns of slacks for inequality type constraints in (DLP3), and (iii)

artificial columns for equality type constraints in (DLP3). An example of such a basis corresponding to our example in Figure 3 is given in Figure 4.

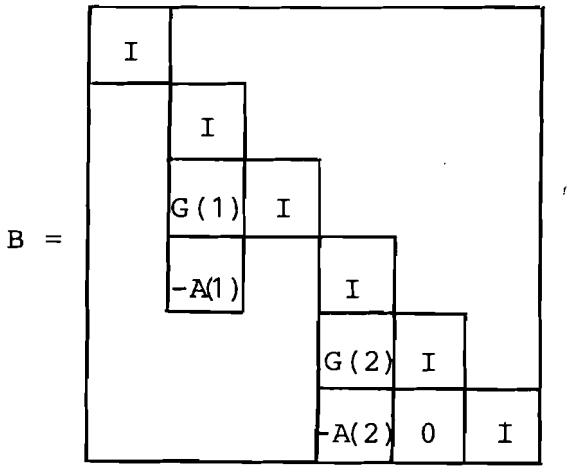

Figure 4. An advanced basis for dynamic LP.

When the basic version was used for Problem F and the above constructed basis was used as a starting basis, the number of iterations was reduced from 1462 corresponding to an all logical starting basis to 583. When the same basis was used for the simplex method, only 363 iterations were needed. However, when the constructed initial basis was combined with an initial nonbasic solution where all the nonbasic variables were set to one, the number of iterations was reduced to 260. For the nonbasic variables equal to 10 and 100, the respective numbers of iterations were 313 and 399. This may support our earlier conjecture in Section 4.3 concerning possible advantages in starting with a nonbasic solution. In any case, the result seems promising as the total number of iterations was reduced by a factor of four to five.

5.3. Initial Solutions for Dynamic LP

We already obtained a relatively encouraging result while using initially the constructed basis and setting the nonbasic variables to a constant value. We shall now experiment further with some straightforward ideas for setting initial values to the controls.

Setting Controls to the Same Level at Each Period

Typically in a DLP the same or almost the same set of control variables (as well as state variables) repeat from one period to another. Let us concentrate on those controls which are common to all periods. Initially, we may set these controls to the same level at each period and the rest of the controls to zero.

At least the following two approaches may be used to determine an initial value for the joint set of controls: (i) We adopt the real current levels for those controls (provided that the system described by DLP already exists), or (ii) we solve first a one-period problem (perhaps with appropriate bounds for the final state variables) and adopt the values for the joint set of controls from this optimal solution.

For two dynamic problems F and G, exactly the same set of controls appear at each time period. As both of the models describe a real forest sector, the current rates for controls were easily available. When initially the constructed basis was used and all the controls were set to their current values it took 240 iterations to solve Problem F representing a reduction by a factor of about 6 compared with the basic version. We should note that the initial solution constructed this way was not feasible: there were 34 infeasibilities for Problem F initially.

The other approach (ii) for constructing initial values for controls was applied as well. For the first period model we require the final state to be at least as good as the initial state; i.e., for each state variable for which a large value is desireable (e.g. wood in the forest, production capacity, etc.) the initial value sets a lower bound for the final value, and for other state variables (e.g. amount of long term external financing) the initial value sets an upper bound for the final value. Starting with the constructed basis for DLP and the controls set to the optimal level of the one period model resulted in 213 iterations for Problem F, thus yielding a slight improvement over the previous approach. Neither in this case was the initial solution for DLP feasible. This approach was also applied to the larger DLP model G. The optimal solution was found in 3050 iterations.

Constructing a Feasible Solution

A relative drawback was notable in both of the previous attempts in trying to construct an initial nonbasic solution. As the initial solution was not feasible, it appeared that the relatively good initial functional value got substantially worse during the Phase I procedure. Thus we concluded that it would be desirable to construct an initial solution which is also feasible. Indeed, as described below, we were easily able to carry out this task for the two test problems F and G. Of course, the generality of such an approach may be doubtful. However, it is the authors' belief that a similar approach is applicable to most dynamic linear programs.

We shall now turn to a case of constructing a feasible starting solution. For Problem F, we first set the controls of all periods

to the optimal level of the one period model. The printout of this solution indicated only two types of infeasibilities: one state variable, cash, became negative for most time periods, and the only equality type of constraint--other than the state equations--was violated for all except the first time period, i.e., the corresponding artificial variable appeared at a non-zero level. This equality constraint defines the profit (for each time period). Taking into account the objective function it became clear that a profit as large as possible was desired for an optimal solution. This allowed us to replace the equality by an inequality, and consequently the artificial variable in the constructed basis was replaced by a slack variable. For bringing the negative cash to a feasible range we simply adjusted a control variable determining the level of external financing. After these changes, the cash was brought to a feasible range, all the new slacks, corresponding to the row defining profit were nonnegative, and no new infeasibilities appeared; i.e., the initial solution was feasible.

Starting with this feasible (nonbasic) solution for Problem F, and with the advanced basis, it took 161 iterations for finding an optimal solution. A similar process was carried out for Problem F to construct a feasible initial solution based on the current levels of controls. The resulting number of iterations for finding an optimal solution was 180.

Thus, when the advanced starting basis was used together with a feasible initial solution, the number of iterations for finding an optimal solution by the reduced gradient method is reduced approximately by a factor of eight to nine compared with starting with an all logical basis and the corresponding basic solution.

REFERENCES

[1] Dantzig, G.B. (1963) Linear Programming and Extensions. Princeton, N.J.: Princeton University Press.

[2] Kallio, M., and E. Porteus (1978) A class of methods for linear programming. Mathematical Programming 14:161-169.

[3] Orchard-Hays, W. (1968) Advanced Linear-Programming Computing Techniques. New York: McGraw-Hill.

[4] Orchard-Hays, W. (1978) Anatomy of a mathematical programming system. Design and Implementation of Optimization Software, edited by H. Greenberg. Netherlands: Sijthoff and Noordhoff.

[5] Orchard-Hays, W. (1978) Scope of mathematical programming software. Design and Implementation of Optimization Software, edited by H. Greenberg. Netherlands: Sijthoff and Noordhoff.

[6] Propoi A., and V. Krivonozhko (1978) The Simplex Method for Dynamic Linear Programming. RR-78-14. Laxenburg, Austria: International Institute for Applied Systems Analysis.

[7] Wolfe, P. (1967) Methods of nonlinear programming. Nonlinear Programming, edited by J. Abadie. New York: Wiley.