ON THE USE OF MODELS IN THE SYNTHESIS OF OPTIMIZATION ALGORITHMS

by

E. Polak

Memorandum No. ERL-M274

16 July 1970

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

ON THE USE OF MODELS IN THE SYNTHESIS OF OPTIMIZATION ALGORITHMS

by

E. Polak
Department of Electrical Engineering and Computer Sciences
Electronics Research Laboratories
University of California, Berkeley, California 94720

1. INTRODUCTION

Two of the pivotal problems in the construction of optimization (as well as other) algorithms are the problem of ensuring that an algorithm is convergent and the problem of ensuring that it is implementable. We shall present methods for coping with both. In particular, we shall show that the solution of both of these problems is considerably facilitated by the use of algorithm models together with corresponding convergence theorems. In this paper we shall consider algorithms for solving problems of the following kind.

1.1. Abstract Problem: Given a closed subset T of a Banach space B (with norm $\|\cdot\|_B$), construct points in T that have property P. □

1.2. Definition: We shall denote by S the set of all points in

Research sponsored by the National Aeronautics and Space Administration, Grant NGL-05-003-016(Sup 8) and the Joint Services Electronics Program Grant AFOSR-68-1488.

T having property P, and we shall call S the solution set. (We shall assume that S is non empty.)

The concept of an algorithm for solving the problem (1.1) being convergent is simple to define; that of it being implementable is not. Let us begin with convergence.

1.3. Definition: We shall say that an algorithm for finding a $\hat{z} \in S$ is convergent if given a $z_0 \in T$, the algorithm constructs a sequence $z_0, z_1, z_2, \ldots$, in T such that if $\{z_i\}$ is finite, its last element is in S, and if it is infinite then all of its accumulation points are in S. □

The somewhat fuzzy concept of an algorithm being implementable rests on the assumption that, for practical purposes, certain functions can be evaluated on a digital computer in finite time, whereas other functions cannot. For example, we may be willing to assume that we can evaluate f(z) in finite time, but we cannot assume that we can compute $\min_{\lambda \geq 0} f(z + \lambda h)$ in finite time. Essentially, an implementable algorithm is one in which each iteration can be assumed to be executable in finite (preferably short) time. Let us elaborate this further. It is quite common for an algorithm to be stated in the following form, where $A: T \to 2^T$ (the set of all subsets of T).

1.4. Algorithm Model: (Solves problem (1.1), $A: T \to 2^T$.)

0. Compute a $z_0 \in T$ and set $i = 0$.
1. Compute a $z_{i+1} \in A(z_i)$.
2. If $z_{i+1} \in S$, stop; otherwise, replace i by i+1 and return to 1.

When stated constructively, we may find that algorithm 1.4 expands as follows.

1.5. Expanded Algorithm Model:

0. Compute a $z_0 \in T$ and set $i = 0$.

1. Select a subalgorithm Σ for computing a point in $A(z_i)$.

2. Set $y_0 = z_i$ and use the subalgorithm Σ to construct an infinite sequence $y_0, y_1, y_2, \cdots$

3. Set $z_{i+1} = \lim_{j\to\infty} y_j$, replace i by i+1 and return to 2.

Obviously, the construction of z_{i+1} according to 3. in (1.5) cannot be carried out in finite time on a digital computer. In practice, the construction of the sequence $\{y_j\}$ is stopped after a finite number of terms are constructed and the last of these terms is used as an approximation to z_{i+1}. If the construction of the sequence $\{y_j\}$ is truncated too early, the resulting "implementation" of algorithm (1.4) may not be convergent. If this truncation occurs too late, then one may be using considerably more computer time than is really necessary. The problem of constructing an implementation for an algorithm can be stated as follows. Given an algorithm of the form (1.5), derive from it a convergent algorithm which, when stated constructively, does not use infinite subprocedures. In addition, we would expect the implementation to be efficient. In the next section we shall give conditions for algorithms of the form (1.4) to be convergent and we shall illustrate by means of two examples how one verifies the various assumptions needed in specific cases. In the remaining sections of this paper we shall give a few specific methods

for algorithm implementation and, again, we shall illustrate these by examples.

2. CONVERGENCE OF CONCEPTUAL ALGORITHMS

Probably, the most useful convergence theorem for algorithms of the form (1.4) is the following one.

2.1. Theorem [1], [2]: Consider algorithm (1.4). Suppose that there exists a function $c:T \to R^1$ which is either continuous or bounded from below, and that for every $z \in T$ such that $z \notin S$, there exist an $\varepsilon(z) > 0$ and a $\delta(z) < 0$ such that

2.2
$$c(z'') - c(z') \leq \delta(z) < 0$$
$$\text{for all } z' \in T, \ \|z' - z\|_B \leq \varepsilon(z),$$
$$\text{and for all } z'' \in A(z')$$

Then algorithm (1.4) is convergent.

Proof: Obviously, we only need to consider the case when the sequence $\{z_i\}$ constructed by (1.4) is infinite. Suppose that $\hat{z}$ is an accumulation point of $\{z_i\}$ and that $\hat{z} \notin S$. Then there exist an $\varepsilon(\hat{z}) > 0$ and a $\delta(\hat{z}) < 0$ for which (2.2) holds. Since $\hat{z}$ is an accumulation point of $\{z_i\}$, there exist a subset $K \subset \{0, 1, 2, \ldots\}$ and an integer $k > 0$ such that $z_i \to \hat{z}$ as $i \to \infty$ for $i \in K$, and $\|z_i - \hat{z}\|_B \leq \varepsilon(\hat{z})$ for $i \geq k$ and

$i \in K$. Suppose that $i \geq k$, $j > 0$ are such that $i \in K$, $(i+j) \in K$. Then

$$c(z_{i+j}) - c(z_i) = [c(z_{i+j}) - c(z_{i+j-1})] + \ldots$$

$$\ldots + [c(z_{i+1}) - c(z_i)]$$

Since none of $z_i \in S$, $i = 0, 1, 2, \ldots$, we must have $c(z_{i+1}) - c(z_i) < 0$ for $i = 0, 1, 2, \ldots$, by (2.2), and since in addition $\|z_i - z\|_B \leq \varepsilon(\hat{z})$, we find that

2.3 $$c(z_{i+j}) - c(z_i) \leq \delta(\hat{z}) < 0$$

$$\text{for all } i \in K,\ i \geq k$$

Relation (2.3) shows that the sequence $\{c(z_i)\}_{i \in K}$ is not Cauchy. However, $\{c(z_i)\}_{i \in K}$ must converge either because $z_i \overset{K}{\to} \hat{z}$ and because $c(\cdot)$ is continuous, or else because $c(z_{i+1}) - c(z_i) < 0$, $i = 0, 1, 2, \ldots$, and $c(\cdot)$ is bounded from below. Hence we have obtained a contradiction and we therefore conclude that $\hat{z} \in S$. □

We shall now show what is involved in using Theorem (2.1) in the case of two well known algorithms. The first is a modified method of centers due to Huard [5] and the second is the Frank and Wolfe algorithm [6]. In order to shorten our exposition, we shall make use of a few

simplifying assumptions.

The Huard algorithm can be used for solving the following problem.

$$2.4 \qquad \min \quad \{f^0(z) \mid f^i(z) \leq 0,\ i = 1, 2, \ldots, m\}$$

where the $f^i: R^n \to R^1$, $i = 0, 1, \ldots, m$, are strictly convex, continuously differentiable functions, such that for some feasible z_0 (i.e. $f^i(z_0) \leq 0$ for $i = 1, 2, \ldots, m$), the set

$$2.5 \quad T = \{z \in R^n \mid f^0(z) - f^0(z_0) \leq 0,\ f^i(z) \leq 0,\ i = 1, 2, \ldots, m$$

is compact and has an interior.

2.6 Modified Method of Centers (Huard [5]):

0. Compute a feasible point z_0 such that the set T in (2.5) is compact and has an interior and set $i = 0$.

1. Set $z = z_i$.

2. Solve the linear programming problem

$$\min \{h^0 \mid -h^0 + \langle \nabla f^0(z), h \rangle \leq 0;\ -h^0 + f^j(z)$$

$$+ \langle \nabla f^i(z), h \rangle \leq 0,\ j = 1, 2, \ldots, m;\ |h^\ell| \leq 1,$$

(cont.)

$\ell = 1, 2, \ldots, n\}$,

and denote its solution by $(h^0(z), h(z))$, where $h^0(z) \in R^1$, $h(z) \in R^n$.

3. If $h^0(z) = 0$, set $z_{i+1} = z$ and stop; otherwise go to 4.

4. Compute the step size $\mu(z, h(z))$ so that

$$2.8 \qquad d(z + \mu(z,h(z))h(z),\ z) = \min_{\mu \geq 0} d(z + \mu h(z),\ z),$$

where the distance function $d:R^n \times R^n \to R^1$ is defined by

$$2.9 \qquad d(z',z'') = \max\{f^0(z') - f^0(z'');\ f^i(z'),\ i = 1, 2, \ldots, m\}$$

5. Set $z_{i+1} = z + \mu(z,h(z))h(z)$, replace i by i+1, and return to 1. □

Comment: Note that $h^0(z)$, as computed in 2. of (2.6), satisfies

$$2.10 \qquad h^0(z) = \min_{\substack{|h^\ell| \leq 1 \\ \ell = 1, 2, \ldots, m;}} \max\{\langle \nabla f^0(z), h \rangle\ ;\ f^j(z) + \langle \nabla f^j(z), h \rangle\ , \quad j = 1, 2, \ldots, m\}$$

that $h^0:R^n \to R^1$ defined by (2.10) is continuous, that $h^0(z) \leq 0$ for all

feasible vectors z, and that $h^0(\hat{z}) = 0$ if and only if $\hat{z}$ is optimal for (2.4), i.e. if and only if $\hat{z}$ is the solution of 2.4 (see [2], [3]). In this case, because of the strict convexity of the $f^i(\cdot)$, the solution set S consists of only one point.

Thus, to associate algorithm (2.6), with the model (1.4), we define the set T as in (2.5), we define the set $S \subset T$ to be the set consisting of $\hat{z}$, the unique optimal solution to (2.4), and we define $A:T \to 2^T$ as follows:

$$A(z) = \{z' \mid z' = z + \mu(z,h(z))h(z)\} \tag{2.11}$$

where $h(z) \in R^n$ is any vector such that $(h^0(z),h(z))$ is optimal for (2.7). (Note that $A(\hat{z}) = \{\hat{z}\}$ where $\hat{z}$ is the unique optimal solution to (2.4). Also note that the solution of (2.7) need not be unique.)

To apply Theorem (2.1) to the above algorithm, we have to produce a function $c(\cdot)$. We set $c(\cdot) = f^0(\cdot)$. Since $f^0(\cdot)$ is continuous, we see that algorithm (2.6) is convergent if for every non-optimal $z^* \in T$, there exist an $\varepsilon(z^*) > 0$ and a $\delta(z^*) < 0$ such that

$$f^0(z') - f^0(z) \leq \delta(z^*) < 0 \tag{2.12}$$

for all $z \in T$ satisfying $\|z - z^*\| \leq \varepsilon(z^*)$ and for all $z' \in A(z)$, with A(z) defined by (2.11).

We shall now show this to be the case. Suppose that the vector $z^* \in T$ is not optimal for the problem (2.4). Then we must have $h^0(z^*) = \gamma^* < 0$. Making use of the continuous differentiability of the $f^j(\cdot)$, $j = 0, 1, \ldots, m$, and of the compactness of the sets T and $H \triangleq \{h \mid |h^j| \leq 1, j = 1, 2, \ldots, n\}$, we can show that there must exist an $\varepsilon^* > 0$ and a $\lambda^* > 0$ such that the following three relations hold:

$$h^0(z) \leq \frac{\gamma^*}{2} \tag{2.13}$$

$$|\langle \nabla f^i(z + \lambda h), h\rangle - \langle \nabla f^i(z), h\rangle| \leq \frac{-\gamma^*}{8} \tag{2.14}$$

$$|f^i(z + \lambda h) - f^i(z)| \leq \frac{-\gamma^*}{8} \tag{2.15}$$

for all $z \in T$ satisfying $\|z - z^*\| \leq \varepsilon^*$, for all $\lambda \in [0,\lambda^*]$, and for all $h \in H$.

Now, let z be any point in T satisfying $\|z - z^*\| \leq \varepsilon^*$, and let h(z) be any vector such that $(h^0(z),h(z))$, is optimal for (2.7). Then, because of (2.13), in view of (2.10), we get

$$\langle \nabla f^0(z), h(z)\rangle \leq h^0(z) \leq \frac{\gamma^*}{2} \tag{2.16}$$

$$f^j(z) + \langle \nabla f^j(z), h(z)\rangle \leq h^0(z) \leq \frac{\gamma^*}{2} \tag{2.17}$$

$$j = 1, 2, \ldots, m$$

By the mean value theorem,

2.18 $$f^j\big(z + \lambda^* h(z)\big) - f^j(z) = \lambda^* \langle \nabla f^j\big(z + \lambda^j h(z)\big), h(z) \rangle$$

$$\lambda^j \in (0,\lambda^*), \quad j = 0, 1, 2, \ldots, m$$

Consequently, (2.14) implies that

2.19 $$f^0\big(z + \lambda^* h(z)\big) - f^0(z) \leq \frac{3\lambda^*\gamma^*}{8} < \frac{\lambda^*\gamma^*}{8}$$

We now make use of the dichotomy that either $f^j(z) > \gamma^*/4$ or else that $f^j(z) \leq \gamma^*/4$. If for some $j \in \{1, 2, \ldots, m\}$, $f^j(z) > \gamma^*/4$, then from (2.17), (2.18) and (2.14) we obtain that

2.20 $$f^j\big(z + \lambda^* h(z)\big) - f^j(z) \leq \frac{\lambda^*\gamma^*}{8}$$

and hence, since $\gamma^* < 0$ and $f^j(z) \leq 0$, we conclude that $f^j\big(z + \lambda^* h(z)\big) \leq \lambda^*\gamma^*/8$. Now, if $f^j(z) \leq \gamma^*/4$ for some $j \in \{1, 2, \ldots, m\}$, then from (15), we conclude that $f^j\big(z + \lambda^* h(z)\big) < \gamma^*/8$. Now, let $z' = z + \mu\big(z,h(z)\big)h(z)$, with $\mu\big(z,h(z)\big)$ as in (2.8). Then we must have

2.21 $$f^0(z') - f^0(z) \leq d(z',z) \leq \max \{f^0\big(z + \lambda^* h(z)\big) - f^0(z);$$

$$f^j\big(z + \lambda^* h(z)\big),\ j = 1, 2, \ldots, m\} \leq \max \{\frac{\gamma^*}{8}, \frac{\lambda^*\gamma^*}{8}\}$$

$$\triangleq \delta^* < 0$$

Since (2.21) is true for all $z \in T$ satisfying $\|z - z^*\| \leq \varepsilon^*$, we see that we are done. □

The Frank and Wolfe algorithm solves the following problem.

2.22 $$\min \{\|t - z\| \mid f^j(z) \leq 0,\ j = 1, 2, \ldots, m\}$$

where the $f^j: R^n \to R^1$ are continuously differentiable, strictly convex functions, and

$$t \notin T \triangleq \{z \in R^n \mid f^j(z) \leq 0,\ j = 1, 2, \ldots, m\}$$

We shall assume that the set T is compact and that it has an interior.

2.23. The Frank and Wolfe Algorithm [6]

0. Compute a $z_0 \in T$ and set $i = 0$.

1. Compute $v(z_i) \in T$ such that

2.24 $$\langle t - z_i, v(z_i) \rangle = \max_{v \in T} \langle t - z_i, v \rangle$$

2. Set

$$[z_i, v(z_i)] = \{z \mid z = \lambda z_i + (1 - \lambda)v(z_i), \lambda \in [0,1]\}$$

and compute $z_{i+1} \in [z_i, v(z_i)]$ such that

2.25 $$\|t - z_{i+1}\| = \min_{z \in [z_i, v(z_i)]} \|t - z\|$$

3. If $\|t - z_{i+1}\| = \|t - z_i\|$, set $z_{i+1} = z_i$ and stop; otherwise replace i by i+1 and return to 1. □

To identify algorithm (2.23) with the model (1.4), we define $S = \{\hat{z}\}$, where $\hat{z} \in T$ is the unique solution to (2.22). It is not difficult to show that under the assumption stated $\|t - z_{i+1}\| = \|t - z_i\|$ (and $z_{i+1} = z_i$) if and only if $z_i = \hat{z}$. Hence we see that the stop command in 3. of (2.23) corresponds to the stop command in 2. of (1.4). The map $A: T \to 2^T$ is now defined as follows.

2.26 $$A(z) = \{z' \mid \|t - z'\| = \min_{z'' \in [z, v(z)]} \|t - z''\|\}$$

Since $A(z)$ consists of exactly one point for all $z \in T$, we see that $A:T \to T$. Note that $A(\hat{z}) = \hat{z}$ $(\hat{z} \in S)$.

To apply theorem (2.1) to algorithm (2.23), we define $c:T \to R^1$ by

$$c(z) = \|t - z\| . \tag{2.27}$$

Since $c(\cdot)$, as defined by (2.27), is obviously continuous, to show that (2.23) is convergent, we only need to show that for every non-optimal $z^* \in T$ there exist an $\varepsilon(z^*) > 0$ and a $\delta(z^*) < 0$ such that (recall that $A:T \to T$ in this case)

$$c\big(A(z)\big) - c(z) \leq \delta(z^*) \tag{2.28}$$

for all $z \in T$ such that $\|z - z^*\| \leq \varepsilon(z^*)$. Now the function $v:T \to T$, defined by (2.24), is continuous since T is strictly convex. In addition, the function $\bar{c}:T \times T \to R^1$ defined by

$$\bar{c}(z',z'') = \min_{t \in [z',z'']} \|t - z\| \tag{2.29}$$

is also continuous. Therefore both $c(\cdot)$ and $c\big(A(\cdot)\big)$ are continuous on T and hence the existence of an $\varepsilon(z^*) > 0$ and a $\delta(z^*) < 0$ satisfying (2.28) follows directly. □

Algorithm (2.6) is non-implementable because of the instruction in 4., while algorithm (2.23) is non-implementable because of the instruction 1. We shall now introduce a few methods for making such algorithms implementable.

3. ADAPTIVE IMPLEMENTATION

Since we may have difficulty in computing points in $A(z_i)$, as we have just seen, we shall construct an algorithm model which uses a map $\bar{A}:R^+\times T\to 2^T$ such that

$$\bar{A}(0,z) = A(0) \quad \text{and} \quad \bar{A}(\varepsilon,z) \to A(z) \quad \text{as} \quad \varepsilon \to 0.$$

In addition, it must be relatively easy to compute points in $\bar{A}(z)$.

According to theorem (2.1), we need to have a function $c:T\to R^1$ with certain properties, so as to ensure that algorithm (1.4) is convergent. Since from the examples given we saw that such a $c(\cdot)$ is usually available, we can include it in the adaptive model below, which solves problem (1.1).

3.1. Algorithm Model:

$$\left(A:R^+\times T\to 2^T,\ c:T\to R^1,\ \varepsilon_0 > 0,\ \varepsilon' \in (0,\varepsilon_0),\ \alpha > 0,\ \beta \in (0,1)\right)$$

0. Compute a $z_0 \in T$, set $i = 0$.

1. Set $\varepsilon = \varepsilon_0$.

2. Compute a $y \in \bar{A}(\varepsilon, z_i)$.

3. If $c(y) - c(z_i) \leq \alpha\varepsilon$, set $z_{i+1} = y$, replace i by i+1 and return to 1; otherwise go to 4.

4. If $\varepsilon > \varepsilon'$, set $\varepsilon = \beta\varepsilon$ and return to 2; otherwise go to 5.

5. If $z_i \in S$, set $z_{i+1} = z_i$ and stop; otherwise set $\varepsilon = \beta\varepsilon$ and return to 2. □

When it is difficult to determine if $z_i \in S$, we may prefer the following approach.

3.2. Algorithm Model:

$$\left(\bar{A}: R^+ \times T \to 2^T,\ c: T \to R^1,\ \varepsilon_0 > 0,\ \alpha > 0,\ \beta \in (0,1)\right)$$

0. Compute a $z \in T$, set $i = 0$.

1. Set $\varepsilon = \varepsilon_0$.

2. Compute a $y \in A(\varepsilon, z_i)$.

3. If $c(y) - c(z_i) \leq -\alpha\varepsilon$, set $z_{i+1} = y$, replace i by i+1 and return to 1; otherwise set $\varepsilon = \beta\varepsilon$, set $z_{i+1} = z_i$, replace i by i+1 and return to 2. □

When returning from 3. to 1., in the algorithms (3.1) or (3.2) results in excessive computing times, due to the process of reducing ε to an acceptably low value, we may return from 3. to 2. without loss of convergence, provided the assumptions of the theorem below are satisfied.

3.3. Theorem: Suppose (i) that $c(\cdot)$ is either continuous or bounded from below on T, and (ii) that for every $z \in T$ which is not in S, there exist an $\varepsilon(z) > 0$, a $\delta(z) < 0$ and a $\gamma(z) > 0$ such that

$$3.4 \qquad c(z'') - c(z') \leq \delta(z) < 0$$

for all $z' \in T$, $\|z' - z\|_B \leq \varepsilon(z)$, for all $z'' \in A(\gamma,z')$, for all $\gamma \in [0,\gamma(z)]$.

Then the algorithms (3.1), (3.2) and the time varying versions of these algorithms obtained by returning from 3. to 2., instead of to 1., are convergent.

Proof: We shall only give a proof for algorithm (3.2) (as stated), since the convergence of the other algorithms can be established in a similar manner.

First, suppose that for some i, $z_i = z_{i+1} = z_{i+2} = \ldots$, i.e., that the algorithm jams up. Then it must be constructing a sequence of vectors $y_{i+j} \in A(\beta^j\varepsilon_0, z_i)$, $j = 0, 1, 2, \ldots$, such that $c(y_{i+j}) - c(z_i) > -\alpha\beta^j\varepsilon_0$, $j = 0, 1, 2, \ldots$, and therefore we conclude from assumption (ii) in (3.3) that z_i is in S.

Now suppose that there is no i such that $z_i = z_{i+1} = z_{i+2} = \ldots$. Suppose that $z^* \in S$ is an accumulation point of the sequence $\{z_i\}$ (constructed by (3.2)), i.e. $z_i \to z^*$ as $i \to \infty$ for $i \in K \subset \{0, 1, 2, \ldots\}$. Then, by assumption (ii) in (3.3), there exist $\varepsilon(z^*) > 0$, $\delta(z^*) < 0$ and $\gamma(z^*) > 0$ for which (3.4) is satisfied. Since $z_i \to z^*$, $i \in K$, there

must exist an integer $k \geq 0$ such that $\|z_i - \hat{z}\|_B \leq \varepsilon(z^*)$ for all $i \in K$, $i \geq k$ and $\max \{\beta^k \varepsilon_0, \alpha\beta^k \varepsilon_0\} \leq \min \{\gamma(z^*), -\delta(z^*)\}$. Hence, for any two consecutive points z_i, z_{i+j}, $z_i \neq z_{i+j}$, $i \geq k$ of the subsequence $\{z_i\}_{i \in K}$, we must have

$$3.5 \quad c(z_{i+j}) - c(z_i) = [c(z_{i+j}) - c(z_{i+j+1})] + \ldots + [c(z_{i+1}) - c(z_i)] \leq -\alpha\beta^k \varepsilon_o \dagger$$

which shows that the monotonically decreasing sequence $\{c(z_i)\}_{i \in K}$ does not converge. But the sequence $\{c(z_i)\}_{i \in K}$ must converge because of assumption (i) and hence we have a contradiction; i.e. the accumulation point z^* must be in S. □

We shall now show how algorithm model (3.2) leads to an implementation of the modified method of centers (2.6). Referring to (2.8), we begin by noting that, because the functions $f^i(\cdot)$, $i = 0, 1, \ldots, m$, are convex, the function $\theta: R^1 \to R^1$ defined by

---

† Let $\ell \in \{i, i+1, \ldots, i+j-1\}$ be such that $z_i = z_\ell$ but $z_{\ell+1} \neq z_\ell$. Then we must have $c(z_{\ell+1}) - c(z_\ell) \leq -\alpha\varepsilon$ with $\varepsilon \geq \beta^k \varepsilon_0$ because of (3.4), since $\beta^k \varepsilon_0 \leq \gamma(z^*)$ and $\alpha\beta^k \varepsilon_0 \leq -\delta(z^*)$. In addition, $c(z_{p+1}) - c(z_p) \leq 0$ for all $p \in \{i, i+1, \ldots, i+j-1\}$. Hence (3.5) follows directly.

3.6 $$\theta(\mu) = d\big(z + \mu h(z),\ z\big)$$

is also convex. Hence we can compute its minimum on $[0,\infty)$ by means of the golden section rule, using the Fibonacci fractions $F_1 = (3 - \sqrt{5})/2$, $F_2 = 1 - F_1$, as follows. Note that $\theta(0) = 0$.

3.7. Golden Section Subprocedure:

0. Find the smallest integer p such that

3.8 $$\theta(p - 1) < \theta(p) \quad \text{and} \quad \theta(p - 1) < \theta(p - 2)$$

and set $a_0 = p - 2$, $b_0 = p$, and $j = 0$.

1. Set $v_j = a_j + F_1(b_j - a_j)$, $w_j = a_j + F_2(b_j - a_j)$

2. If $\theta(v_j) < \theta(w_j)$, set $a_{j+1} = a_j$, set $b_{j+1} = w_j$, replace j by j+1 and return to 1; otherwise set $a_{j+1} = v_j$, set $b_{j+1} = b_j$, replace j by j+1 and return to 1. □

The intervals $[a_j,b_j]$, $j = 0, 1, 2, \ldots$, contain the minimizer $m\big(z,h(z)\big)$ required by algorithm (2.6). Their lengths decrease exponentially, with $b_j - a_j \doteq 0.62^j(b_0 - a_0)$, $j = 0, 1, 2, \ldots$. Quite obviously, this procedure will not find $\mu\big(z,h(z)\big)$ in finite time.

Now, in the context of problem (2.4) and algorithm (2.6), suppose that we define the map $\bar{A}:R^+ \times T \to 2^T$ as follows:

$$3.9 \quad \bar{A}(\varepsilon,z) = \{z' = (z + \mu h(z)) \in T \mid |\mu - \mu(z,h(z))| \leq \varepsilon\}$$

where $h(z)$ and $\mu(z,h(z))$ are as in (2.11). Then we see that $\bar{A}(0,z) = A(z)$, and, if we define $c(\cdot) = f^0(\cdot)$, then the assumptions of theorem (3.3) will be satisfied by $f^0(\cdot)$ and $\bar{A}(\cdot,\cdot)$ as defined by (3.9). A convergent implementation for algorithm (2.6) is therefore obtained as follows.

3.10. Implementation of Algorithm (2.6):

0. Compute a feasible point z_0 such that the set T in (2.5) is compact; select an $\varepsilon_0 > 0$, an $\alpha > 0$, and set $i = 0$.

1. Set $\varepsilon = \varepsilon_0$.

2. Set $z = z_i$.

3. Compute $(h^0(z), h(z))$ by solving (2.7).

4. If $h^0(z) = 0$, set $z_{i+1} = z$ and stop; otherwise go to 5.

5. Use the Golden Section Subprocedure (3.7) to find an interval $[a_j,b_j]$ such that $b_j - a_j \leq \varepsilon$, containing $\mu(z,h(z))$ as defined by (2.8), and set $\mu' = (a_j + b_j)/2$.

Comment: The search for the interval $[a_j,b_j]$ is a finite process.

6. If $f^0(z + \mu' h(z), z) - f^0(z) \leq -\alpha\varepsilon$, set $z_{i+1} = z + \mu' h(z)$, replace i by i+1 and return to 1; otherwise set $\varepsilon = \varepsilon/2$ and return to 5. □

4. OPEN LOOP IMPLEMENTATION

The models for implementing algorithm (1.4) to be given in this section bear a strong resemblance to the ones introduced in the preceding one, with the notable exception that they do not make use of an ε test in constructing an approximation to the set A(z). The approximation is continuously improved as the computation proceeds. The advantage of "open loop" implementation lies in the fact that it somewhat reduces the need for tests which may involve lengthy calculations. The open loop approach is particularly recommended for use when the same problem has to be resolved over and over again, with relatively minor changes in its parameters.

As in the preceding section, we make use of a map $c:T\to R^1$ and, in addition, we shall use a map $\tilde{A}:N\times T\to 2^T$, where N is the set of all positive integers ($0 \in N$). Also, we shall require truncation functions which we define as follows: $t:N\to N$ is a <u>truncation function</u> if for every $k \in N$ there exists a $k' \in N$ such that $t(i) > k$ for all $i \geq k'$, and if $t(i) > i$ for all $i \in N$.

The following algorithm solves problem (1.4).

4.1. <u>Algorithm Model</u>:

$$(\tilde{A}:N\times T\to 2^T,\ c:T\to R^1,\ t_1:N\to N,\ t_2:N\to N,$$

$$t_1,\ t_2 \text{ are truncation functions})$$

0. Compute $z_0 \in T$, set $i = 0$, set $j = 0$.

1. Set $z = z_i$, set $j = t_1)i)$.

2. Compute a $y \in \tilde{A}(j,z)$.

3. If $c(y) < c(z)$, set $z_{i+1} = y$, replace i by i+1 and return to 1; otherwise replace i by i+1, j by $t_2(j)$ and return to 2. □

When the utility of the test $c(y) < c(z)$ in 3 of algorithm (4.1) is outweighed by the effort in computing c(y), we may prefer to use a direct test for determining whether $z_i \in S$, as follows.

4.2. Algorithm Model:

$(\tilde{A}: N \times T \to 2^T,\ t: N \to N$ is a truncation function)

0. Compute $z_0 \in T$, set $i = 0$.

1. Set $j = \ell(i)$.

2. Compute a $y \in \tilde{A}(j,z)$.

3. If $y \in S$, stop; otherwise, set $z_{i+1} = y$, replace i by i+1 and return to 1. □

4.3. Theorem: Consider algorithm (4.1). Suppose (i) that $c(\cdot)$ is either continuous or else bounded from below on T, and (ii) that for every $z \in T$ which is not in S, there exist an $\varepsilon(z) > 0$, a $\delta(z) < 0$, and an integer $k(z) \geq 0$ such that

4.4 $$c(z'') - c(z') \leq \delta(z) < 0$$

for all $z' \in T$, $\|z' - z\|_B \leq \varepsilon(z)$, for all $z'' \in A(j,z')$, for all $j \geq k(z)$.

Then algorithm (4.1) is convergent.

Proof: First suppose that there is an integer i such that $z_i = z_{i+1} = z_{i+2} = \ldots$, i.e., that the algorithm jams up at z_i cycling between 2 and 3. Then, because of (ii), $z_i \in S$.

Now, suppose that there is no j such that $z_j = z_{j+1} = z_{j+2} = \ldots$, and that $z_i \to \hat{z}$ as $i \to \infty$ for $i \in K \subset \{0, 1, 2, \ldots\}$, with $\hat{z} \notin S$. Then there exist an $\varepsilon(z) > 0$, a $\delta(z) < 0$ and a $k(\hat{z}) \in N$ such that

4.5 $$c(z'') - c(z') \leq \delta(\hat{z})$$

for all $z' \in T$, $\|z' - z\|_B \leq \varepsilon(\hat{z})$, for all $z' \in A(j,z')$, $j \geq k(\hat{z})$. Since $z_i \to \hat{z}$ for $i \in K$, and $t_1(\cdot)$, $t_2(\cdot)$ are truncation functions, there exists an integer $k \geq 0$ such that for all $i \in K$, $i \geq k$, $\|z_i - z\|_B \leq \varepsilon(\hat{z})$ and $t(i) \geq k(\hat{z})$. Hence, if i, i+j are consecutive elements in K, with $i \geq k$, then by the same argument as the one used in the proof of theorem (3.3), we conclude that

4.6 $$c(z_{i+j}) - c(z_i) \leq \delta(z)$$

Since (4.6) contradicts the convergence of the sequence $\{c(z_i)\}_{i\in K}$, we are done. □

Since algorithm (4.2) does not use a function $c(\cdot)$, we must stipulate its existence in our assumptions, as is done below.

4.7. Theorem: Consider algorithm (4.2). If there exists a function $c: T \to R^1$ which together with the map A satisfies the theorem (4.3), then every limit point of a sequence $\{z_i\}$ constructed by algorithm (4.2), and satisfying $c(z_{i+1}) < c(z_i)$ for $i = 0, 1, 2, \ldots$, must be in S. □

We shall now implement the Frank and Wolfe method (2.23) by modifying it to correspond to the form (4.2).

First, we note that we can solve the subproblem (2.24) of algorithm (2.23) by the method of feasible directions, below.

4.8. Algorithm for Computing $\bar{v}(z,y,j)$, given $y,z \in T$, given $j \in N$, $\beta \in (0,1)$

0. Set $k = 0$, set $x = y$.

1. Compute $(h^0(x), h(x))$ $(h^0 \in R^1, h \in R^n)$ by solving the linear programming problem $\min \{h^0 \mid -h^0 + \langle (z - t), h \rangle \leq 0; -h^0 + f^p(x) + \langle \nabla f^p(x), h \rangle \leq 0, p = 1, 2, \ldots, m; |h^q| \leq 1, q = 1, 2, \ldots, n\}$.

2. Find the smallest positive integer s such that $f^p(x + \beta^s h(x)) \leq 0$ for $p = 1, 2, \ldots, m$.

3. If $k = j$, set $\bar{v}(z,y,j) = x + \beta^q h(x)$ and stop; otherwise set $x = x + \beta^q h(x)$, replace k by k+1, and return to 1. □

Note that $v(z) = \lim_{j\to\infty} \bar{v}(z,y,j)$.

If we define the map $\tilde{A}:N\times T\to T$ by

$$4.9 \qquad \tilde{A}(j,z) = \{z' \mid \|t - z'\| = \min_{z''\in[z,\bar{v}(z,z,j)]} \|t - z''\|\}$$

with $\bar{v}(z,z,j)$ as computed by (4.8), then we see that $\tilde{A}(\infty,z) = A(z)$, with $A(z)$ as in (2.26). It is not too difficult to show that the maps $\tilde{A}(\cdot,\cdot)$, defined by (4.9), and $c(\cdot) = f^0(\cdot)$, satisfy the assumptions of theorem (4.3). Hence, we obtain the following, convergent implementation of the Frank and Wolfe algorithm. We only use one truncation function in this implementation, i.e. we set $t_1(\cdot) = t_2(\cdot)$.

4.10. Implementation of Algorithm (2.23):

0. Compute a $z_0 \in T$; select a truncation function $t(\cdot)$, and set $i = 0$.

1. Set $j = t(i)$.

2. Compute $\bar{v}(z_i,z_i,j)$ by means of (4.8).

Comment: Note that this process is finite.

3. Compute $z' \in [z_i,\bar{v}(z_i,z_i,j)]$ such that

$$4.11 \qquad \|t - z_{i+1}\| = \min \{\|t - z\| \mid z \in [z_i,\bar{v}(z_i,z_i,j)]\}$$

Comment: Note that (4.11) can be solved trivially and hence the

computation of z_{i+1} is a finite process.

4. If $\|t - z'\| < \|t - z_i\|$, set $z_{i+1} = z'$, replace i by i+1 and return to 1; otherwise set $z_{i+1} = z_i$, replace i by i+1 and return to 1. □

5. CONCLUSION

In this paper we have explored a few methods for algorithm implementation. Obviously, these methods represent only a small fraction of possibilities. A number of other implementation schemes can be found in [3] and [4]. When presented with the need of implementing an algorithm, the reader may find that the models which we have discussed in this paper do not fit his needs exactly. However, he will find that the general philosophy of adaptive and open loop implementation, which we have sketched out, can be utilized in a very wide variety of situations.

REFERENCES

1. E. Polak, "Computational Methods in Discrete Optimal Control and Nonlinear Programming: A Unified Approach," University of California, Berkeley, Electronics Research Laboratories, Memo No. ERL-M261, February 1969.

2. E. Polak, "On the Convergence of Optimization Algorithms," RIRO, No. R1, pp. 17-34, 1969.

3. E. Polak, "Computational Methods in Optimization: A Unified Approach," Academic Press, 1971 (in press).

4. G. Meyer and E. Polak, "Abstract Models for the Synthesis of Optimization Algorithms," University of California, Berkeley, Electronics Research Laboratories, ERL Memo No. ERL-M269, October 1969.

5. P. Huard, "Programmation Mathematique Convex," RIRO, R7, pp. 43-59, 1968.

6. M. Frank and P. Wolfe, "An Algorithm for Quadratic Programming," U. S. Naval Log. Quart., Vol. 3, pp. 95-110, 1956.