C-PROGRAMMING: A NONLINEAR PARAMETRIC OPTIMIZATION METHOD

M. SNIEDOVICH
National Research Institute for Mathematical Sciences, CSIR, P.O. Box 395,
Pretoria 0001, South Africa

Abstract—This paper provides an overview of the methodological, theoretical and computational aspects of c-programming.

1. INTRODUCTION

In this paper we outline the main points of the nonlinear parametric optimization method that we entitled c-programming [1–9]. This method is designed for the solution of optimization problems that admit the following representation:

\[ \text{Problem } P \]
\[ p := \max_{x \in X} f(x) = \phi(u(x)), \]

where \( u \) is a function on some set \( X \) with values in the \( k \)-dimensional Euclidean space \( \mathbb{R}^k \), and \( \phi \) is a real-valued function on \( u(X) = \{ u(x) : x \in X \} \). Let \( X^* \) denote the set of optimal solutions to this problem.

We shall describe the stance that c-programming takes with regard to such problems, explain the motive prompting this stance, and identify the grounds that legitimize it. Then we shall sketch the algorithms that c-programming provides. And finally, by illustrating in broad terms the manner in which c-programming operates, we shall make clear what its capabilities and merits are.

2. STANCE

It will be remembered that problems of the type defined by Problem \( P \) often prove unyielding to the solution techniques of standard optimization methods. Taking note of the difficulties involved in their solution, c-programming proposes that these problems be tackled indirectly. In other words, c-programming argues the following.

Suppose that the function \( \phi \) is linearized and the objective function \( f \) is expanded as follows:

\[ f_\lambda(x) := \sum_{n=1}^{k} \lambda_n u_n(x), \quad x \in X, \lambda \in \mathbb{R}^k, \]

where \( \lambda_n \) and \( u_n(x) \) denote the \( n \)th component of the vectors \( \lambda \) and \( u(x) \) respectively. Then, what we would have in hand is the following parametric problem:

\[ \text{Problem } P(\lambda) \]
\[ p(\lambda) := \max_{x \in X} f_\lambda(x) = \sum_{n=1}^{N} \lambda_n u_n(x), \quad \lambda \in \mathbb{R}^k. \]

Let \( X^*(\lambda) \) denote the set of optimal solutions to Problem \( P(\lambda) \).

As Problem \( P(\lambda) \) is in many cases tractable, the establishment of a link between Problem \( P \) and Problem \( P(\lambda) \), such that a solution of the latter will be tantamount to a solution of the former
for some \( \lambda \in \mathbb{R}^k \), will enable one to circumvent handling of Problem \( P \) directly. Obviously, to prove such a link it must be shown that a \( \lambda \in \mathbb{R}^k \) such that \( X^*(\lambda) \subseteq X^* \) exists. But what is more, to make the consequent proposition—that a solution to Problem \( P \) is obtainable via the solution of Problem \( P(\lambda) \)—workable, a mechanism for the recovery of such a \( \lambda \) must be made available.

In this section we therefore describe the general drift of the argument that we shall advance to show under what conditions \( X^*(\lambda) \subseteq X^* \) for some \( \lambda \in \mathbb{R}^k \). Assume that the set \( u(X) \) is bounded, and let \( U_c \) denote the closure of its convex hull. Also, assume that there exists an \( x^* \in X^* \) such that \( u(x^*) \) lies on the boundary of \( U_c \). Next, let \( \lambda \) be any element of \( \mathbb{R}^k \) such that

\[
\lambda'v \leq \lambda'u(x^*), \quad \forall v \in U_c,
\]

where \( \lambda' \) denotes the transpose of \( \lambda \) and \( \lambda'v \) denotes the inner product of \( \lambda \) and \( v \), that is,

\[
\lambda'v := \sum_{n=1}^{k} \lambda_n v_n.
\]

Then, by construction, \( \{v \in \mathbb{R}^k : \lambda'v \leq \lambda'u(x^*)\} \) is the negative closed half space induced by the hyperplane supporting \( U_c \) at \( u(x^*) \). Thus, since \( U_c \) is convex and \( u(x^*) \) is on the boundary of \( U_c \) such a \( \lambda \) exists, and since \( u(X) \subseteq U_c \), it follows from condition (4) that \( x^* \in X^*(\lambda) \). In short, the above conditions provide for the existence of a pair \( (\lambda, x) \in \mathbb{R}^k \times X \) such that \( x \in X^*(\lambda) \) and \( x \in X^* \).

Next, recall that a problem involving the maximization of a convex function over a compact set is characterized by the property that at least one of its optimal solutions is an extreme point. Thus, for \( u(X^*) \) to intersect the boundary of \( U_c \), it is sufficient that \( \phi \) satisfy certain convexity conditions. And so, as we shall shortly see, granting these conditions will ensure the existence of a \( \lambda \in \mathbb{R}^k \) such that \( X^*(\lambda) \subseteq X^* \).

3. ANALYSIS

In preparation for the ensuing arguments, let us assume the following to be the case.

Assumption

The set \( X^* \) is not empty, and the function \( \phi \) is differentiable on some open convex set \( U \subseteq \mathbb{R}^k \) such that \( u(X) \subseteq U \).

It is important to note that this assumptions in no way requires \( f \) to be differentiable on \( X \), nor that the solution set \( X \) be convex. Consequently, \( X \) can be a discrete set. Next, consider the following.

Theorem 1 [5,7]

If \( \phi \) is pseudoconvex on \( U \), then

\[
X^*(\nabla \phi(u(x))) \subseteq X^*, \quad \forall x \in X^*
\]

and

\[
x \in X^* \Rightarrow x \in X^*(\nabla \phi(u(x))), \quad \forall x \in X^*,
\]

where \( \nabla \phi(z) \) denotes the gradient of \( \phi \) with respect to \( u \) at \( z = u(x) \).

Theorem 2 [5, 7]

If \( \phi \) is pseudoconcave on \( u \), then

\[
x \in X^*(\nabla \phi(u(x))) \Rightarrow x \in X^*
\]

\[
\Rightarrow X^*(\nabla \phi(u(x))) \subseteq X^*. \quad \star
\]

Given that a function is said to be pseudolinear if it is both pseudoconvex and pseudoconcave, these results yield the following.
Theorem 3 [5, 7]
If \( \phi \) is pseudolinear on \( U \), then
\[
\begin{align*}
  x \in X^* & \iff x \in X^*(\nabla \phi (u(x))) \\
\end{align*}
\]  
and
\[
\begin{align*}
  X^*(\nabla \phi (u(x))) & = X^*, \quad \forall x \in X^*. \quad \blacklozenge
\end{align*}
\]
Details concerning the congenital properties of generalized convex and concave functions can be found in Ref. [10].

Having thus established that the simple generalized convexity conditions on \( \phi \) would guarantee the link between Problem \( P \) and Problem \( P(\lambda) \), we can now proceed to examine the general profile of the algorithms that c-programming proposes for the recovery of the desired \( \lambda \)'s.

Case 1: \( \phi \) is Pseudolinear on \( U \)

The first-order necessary and sufficient optimality condition stipulated by condition (9) suggests the following.

Algorithm 1

Step 1. Select an element \( x^{(0)} \) from \( X \) and set \( m = 0 \) and \( \lambda^{(0)} = \nabla \phi (u(x^{(0)})) \).

Step 2. Solve Problem \( P(\lambda^{(m)}) \) and select an element \( y \) from \( X^*(\lambda^{(m)}) \).

Step 3. If \( x^{(m)} \in X^*(\lambda^{(m)}) \), set \( x^* = x^{(m)} \) and stop. Otherwise, go to Step 4.

Step 4. Set \( x^{(m+1)} = y \) and \( \lambda^{(m+1)} = \nabla \phi (u(y)) \).

Step 5. Set \( m = m + 1 \) and go to Step 2. \quad \blacklozenge

Some of this algorithm's properties are spelled out by the following theorem.

Theorem 4 [5, 7, 8]
Assume that \( \phi \) is pseudolinear on \( U \). Then,

1. The sequence \( \{f(x^{(m)})\} \) generated by Algorithm 1 is strictly increasing.
2. If \( u(X) \) is finite, then Algorithm 1 terminates and \( X^*(\nabla \phi (u(x^*))) = X^* \).
3. Assume that \( \nabla \phi \) is continuous on \( U \) and that the sequence \( \{u(x^{(m)})\} \) generated by Algorithm 1 converges to some \( u' \in U \). Then, the sequence \( \{f(x^{(m)})\} \) converges to \( p \) and \( X^*(\nabla \phi (u')) = X^* \). \quad \blacklozenge

What is interesting about this case is that it includes the class of fractional programming problems. Recall that fractional programming problems are nonlinear optimization problems of the following form [11, 12]:

\[
q = \max_{x \in X} \frac{N(x)}{D(x)},
\]

where \( N \) and \( D \) are real-valued functions on some set \( X \) and \( D(x) > 0, \forall x \in X \). However, setting \( k = 2 \),
\[
u(x) = (N(x), D(x))
\]
and
\[
\phi(z) = \frac{z_1}{z_2}, \quad z \in \mathbb{R} \times \mathbb{R}^+ := \{a \in \mathbb{R}, a > 0\}
\]
brings fractional programming problems under the c-programming format. Furthermore, \( \phi \) being differentiable on \( U = \mathbb{R} \times \mathbb{R}^+ \), that is,
\[
\nabla \phi (z) = (1/z_2, -z_1/z^2_2), \quad z \in U,
\]
and pseudolinear on \( U \) [8], entail that fractional programming problems are in fact pseudolinear c-programming problems. The inference therefore is that fractional programming should be regarded a classical method par excellence. More details about the relation between fractional programming and c-programming problems can be found in Ref. [8].
Case 2: \( \phi \) is Convex on \( U \)

Since convexity, in conjunction with differentiability, entails pseudoconvexity, it follows that Theorem 1 holds in this case as well. Under these terms Algorithm 1 would possess the following properties.

**Theorem 5** [7]

Assume that 4) is convex on \( U \). Then,

1. The sequence \( \{f(x^{(m)})\} \) generated by Algorithm 1 is strictly increasing.
2. If \( u(X) \) is finite, then Algorithm 1 terminates. ♦

However, for want of an obvious first-order sufficient optimality condition, this case lacks a criterion for determining whether the solution yielded by Algorithm 1 is optimal.

**Example 1**

\[
p := \max_{0 \leq x \leq 6} (x - 4)^2.
\]  

Set \( k = 1, X = [0, 6], u(x) = x - 4, U = \mathbb{R} \) and \( \phi(z) = z^2 \). Clearly, \( \phi \) is convex and differentiable on \( U \). Note that the parametric problem would be of the form

\[
p(\lambda) := \max_{0 \leq x \leq 6} \lambda(x - 4), \quad \lambda \in \mathbb{R},
\]  

so that by inspection

\[
X^*(\lambda) = \begin{cases} 
\{6\}, & \lambda > 0, \\
X, & \lambda = 0, \\
\{0\}, & \lambda < 0.
\end{cases}
\]  

Consider now the feasible solution \( x^{(i)} = 6 \). Since \( \nabla \phi(u(x)) = 2(x - 4) \), we have \( \nabla \phi(u(x^{(i)})) = 4 \), so that \( x^{(i)} \in X^* \nabla \phi(u(x^{(i)})) \). It then follows that Algorithm 1 terminates after one iteration, to yield \( x^* = x^{(i)} = 6 \). However, since \( X^* = \{0\} \), it follows that \( x^* = 6 \notin X^* \). ♦

To obviate this difficulty we propose that auxiliary measures, or what we term *exclusionary rules*, be introduced in the search procedure. The definition of an exclusionary rule is as follows.

**Definition 1**

Let \( E \) be a map from \( \mathbb{R}^k \times X \) to the power set of \( \mathbb{R}^k \), that is, assume that to each pair \( (\lambda, x) \in \mathbb{R}^k \times X \) the map \( E \) assigns a subset of \( \mathbb{R}^k \). Then, \( E \) is said to be an exclusionary rule for Problem \( P \) if

\[
(x \in X^*(\lambda), y \in X, \nabla \phi(u(y)) \in E(\lambda, x)) \Rightarrow f(x) \geq f(y). \quad (18)
\]

An exclusionary rule is intended then to regulate the elimination of subsets of \( \mathbb{R}^k \) in the search for the elements of the set \( \{\nabla \phi(u(x)) : x \in X^*\} \). The following result is an immediate consequence of this definition.

**Theorem 6** [5, 7]

Assume that \( E \) is an exclusionary rule, and let \( \{(\lambda^{(m)}, x^{(m)}): 1 \leq m \leq M\} \) be any finite sequence such that \( \lambda^{(m)} \in \mathbb{R}^k \) and \( x^{(m)} \in X^*(\lambda^{(m)}) \) for all \( 1 \leq m \leq M \), and

\[
V = \bigcup_{m=1}^{M} E(\lambda^{(m)}, x^{(m)}),
\]  

where \( V \) is any subset of \( \mathbb{R}^k \) such that \( \nabla \phi(u(x)) \in V, \forall x \in X \).

Then,

\[
p = \max_{0 \leq m \leq M} f(x^{(m)}). \quad (20)
\]  

In [14], [17], [18].
Also, with function (2) providing that \( x \in X^*(\lambda) \) entails that \( x \in X^*(\beta \lambda), \forall \beta \geq 0 \), the implication is that an exclusionary rule would possess the following congenital property.

**Theorem 7** [5, 7]

Assume that \( E \) is an exclusionary rule for Problem \( P \). Then the map \( E' \) specified by

\[
E'(\lambda, x) := \bigcup_{\beta \geq 0} E(\beta \lambda, x), \quad (\lambda, x) \in \mathbb{R} \times X
\]

is also an exclusionary rule. ♦

With the above results in mind, consider the following.

**Algorithm 2**

1. Find a set \( V \) such that \( \nabla \phi(u(x)) \in V, \forall x \in X \).
2. Select an element \( \lambda^{(1)} \) from \( V \), solve Problem \( P(\lambda^{(1)}) \), select an element \( x^{(1)} \) from \( X^*(\lambda^{(1)}) \), and set \( V^{(1)} = E(\lambda^{(1)}, x^{(1)}), x^* = x^{(1)} \), and \( m = 1 \).
3. If \( V \subseteq V^{(m)} \), stop. Otherwise, go to Step 4.
4. Select an element \( \lambda^{(m+1)} \) from \( V \).
5. Solve Problem \( P(\lambda^{(m+1)}) \) and select an element \( x^{(m+1)} \) from \( X^*(\lambda^{(m+1)}) \).
6. If \( f(x^{(m+1)}) > f(x^{(m)}) \), set \( x^* = x^{(m+1)} \).
7. Set \( V^{(m+1)} = V^{(m)} \cup E(x^{(m+1)}, x^{(m)} \).
8. Set \( m = m + 1 \) and go to Step 3.

Note that Theorem 6 ensures that, on termination, Algorithm 2 provides that \( x^* \in X^* \). Going back to the case under consideration, consider the following.

**Theorem 8**

If \( \phi \) is convex on \( U \), then the map \( E_i \) defined by

\[
E_i(\lambda, x) := \{\alpha \lambda + (1 - \alpha) \nabla \phi(u(x)) : 0 < \alpha \leq 1\}, \quad (\lambda, x) \in \mathbb{R}^k \times X,
\]

is an exclusionary rule, and so is

\[
E'_i(\lambda, x) := \bigcup_{\beta > 0} E_i(\beta \lambda, x). \quad ♦
\]

Admittedly, \( E_i \) will not be effective in cases where \( k > 2 \), because, by construction, \( E_i(\lambda, x) \) is the line segment connecting the two points \( \lambda \) and \( \nabla \phi(u(x)) \) in \( \mathbb{R}^k \). This, however, is a direct consequence of the relative indeterminateness of the structure of \( \phi \). Indeed, as borne out by the next case, far more powerful exclusionary rules can be formulated under more favorable conditions.

**Case 3: \( \phi \) is Convex Additive and Separable**

Under this heading we bring problems whose functions \( \phi \) admit the following representation:

\[
\phi(z) = \sum_{n=1}^{k} \phi_n(z_n), \quad z \in U,
\]

where for each \( 1 \leq n \leq k, \phi_n \) is a convex differentiable function on some open convex set \( U_n \) such that \( u_n(x) \in U_n, \forall x \in X \).

**Theorem 9** [5, 7]

If \( \phi \) is convex additive and separable, then the function \( E_2 \) defined by

\[
E_2(\lambda, x) := \{z \in \mathbb{R}^k : z_n = \alpha_n \lambda_n + (1 - \alpha_n) \nabla \phi_n(u(x)), 0 < \alpha_n \leq 1, 1 \leq n \leq k\}, \quad (\lambda, x) \in \mathbb{R}^k \times X
\]

is an exclusionary rule. ♦
Note that, by construction, $E_2(\lambda, x)$ is the $k$-dimensional rectangular defined by the two points $\lambda$ and $\nabla \phi(u(x)))$, excluding the surface connected to $\nabla \phi(u(x)))$. This rule, and consequently, its extension,

$$E^*_2(\lambda, x) = \bigcup_{\beta \neq 0} R_\beta(\beta \lambda, x), \quad (\lambda, x) \in \mathbb{R}^k 	imes X$$

will therefore perform effectively even in cases where $k > 2$.

Details concerning the technical issues bearing on the working of Algorithm 2 can be found in Ref. [8].

4. MODE OF OPERATION

C-Programming does not provide a technique for solving Problem $P(\lambda)$. Indeed, it proceeds on the assumption that the machinery for solving Problem $P(\lambda)$ will be supplied by other optimization methods. What c-programming does do, however, is to latch onto any optimization method capable of solving Problem $P(\lambda)$ and in doing assists any such method in the following manner. Through the medium of its algorithms, c-programming generates values of $\lambda$ that it relays to the method engaged in the solution of Problem $P(\lambda)$. In view of the already established link between Problem $P$ and Problem $P(\lambda)$, this collaboration has the effect of procuring a solution to Problem $P$. Or to state this differently, by teaming up with a method capable of solving Problem $P(\lambda)$, c-programming makes the latter's capabilities applicable to cases that would otherwise fall outside the method's original scope.

Example 2

$$P = \max_{x \in \mathcal{X}} \phi_1(c'x) + \phi_2(d'x),$$

where

$$X = \{x \in \mathbb{R}^N: Ax \leq b, x \geq 0\},$$

$A$ is an $M \times N$ matrix and $b$ is vector in $\mathbb{R}^M$. Bringing c-programming into play will mean setting $k = 2$, $u(x) = (c'x, d'x)$ and $\phi(z) = \phi_1(z_1) + \phi_2(z_2)$, in which case Problem $P(\lambda)$ will take the following form:

$$p(\lambda) = \max_{x \in \mathcal{X}} \lambda_1 c'x + \lambda_2 d'x, \quad \lambda \in \mathbb{R}^2$$

$$= \max_{x \in \mathcal{X}} (\lambda_1 c + \lambda_2 d)'x.$$  \hfill (28)

$$p(\lambda) = \max_{x \in \mathcal{X}} (\lambda_1 c + \lambda_2 d)'x.$$  \hfill (29)

Since this is a standard linear programming problem, it follows that c-programming's cooperation with linear programming techniques will enable one to treat the nonlinear problem specified by equations (26) and (27). A solution procedure for problems of the type is discussed in Ref. [7].

Example 3

$$P = \min_{x \in \mathcal{X}} \sum_{n=1}^N \left[ g_n(x_n) - \frac{1}{N} \sum_{m=1}^N g_m(x_m) \right]^2,$$  \hfill (30)

where

$$X = \left\{ x \in \mathbb{R}^N: \sum_{n=1}^N x_n \leq r, x_n \in \{0, 1, 2, 3, \ldots, r\} \right\}.$$  \hfill (31)

and $r$ is a positive integer.

Note that this minimum-variance type problem is rendered nonseparable when given a standard dynamic programming formulation [4]. However, an expansion of the objective function in equation (30) yields

$$\sum_{n=1}^N \left[ g_n(x_n) - \frac{1}{N} \sum_{m=1}^N g_m(x_m) \right]^2 = \left[ \sum_{n=1}^N g_n^2(x_n) \right] - \frac{1}{N} \left[ \sum_{m=1}^N g_m(x_m) \right]^2,$$  \hfill (32)
whereupon the following can be set: $k = 2$, 
\[
  u(x) = \left( \sum_{n=1}^{N} g_2^n(x_n), \sum_{n=1}^{N} g_4(x_n) \right), \quad x \in X
\]  
and 
\[
  \phi(z_1, z_2) = z_1 - \frac{1}{N} z_2^2, \quad z \in U = \mathbb{R}^2.
\]

In this case Problem $P(\lambda)$ would take the following form:
\[
  \begin{align*}
    p(\lambda) &= \min_{x \in X} \lambda_1 \sum_{n=1}^{N} g_2^n(x_n) + \lambda_2 \sum_{n=1}^{N} g_4(x_n), \quad \lambda \in \mathbb{R}^2, \\
    &= \min_{x \in X} \left[ \lambda_1 g_2(x_n) + \lambda_2 g_4(x_n) \right], \\
    &= \min_{x \in X} \sum_{n=1}^{N} g_2'(x_n, \lambda), \quad g_2'(x_n, \lambda) = \lambda_2 g_2^2(x_n) + \lambda_2 g_4(x_n).
  \end{align*}
\]

Clearly, for each $\lambda \in \mathbb{R}^2$, this is a standard additive dynamic programming problem. Furthermore, since the partial derivative of $q_1$ with respect to $u_1$ is equal to one everywhere, we can set $\lambda_1 = 1$. Also, given that $\phi$ is differentiable and concave on $U$ and admits the presentation stipulated by equation (23), the implication is that the problem under consideration falls within the class of additive separable c-programming problems. A solution procedure for the problem is described in detail in Ref. [4].

Example 4
\[
  q^* = \max_{x \in X} \frac{N(x)}{\varphi(D(x))},
\]
where $N$ and $D$ are real-valued functions on some set $X$ and $\varphi$ is a real-valued function on $\mathbb{R}$. Recall that the precondition for using the standard parametric fractional programming method [11, 12] in this case is that the following parametric problem be tractable:
\[
  q(t) = \max_{x \in X} N(x) - t\varphi(D(x)), \quad t \in \mathbb{R}.
\]

The trouble is, however, that the latter is hard to solve if $\phi$ is not linear. Bringing c-programming into play will entail setting $k = 2$, $u(x) = (N(x), D(x))$ and $\phi(z) = z_1/z_2$, in which case Problem $P(\lambda)$ will take the following form:
\[
  \begin{align*}
    p(\lambda) &= \max_{x \in X} \lambda_1 N(x) + \lambda_2 D(x), \quad \lambda \in \mathbb{R}^2.
  \end{align*}
\]

Again, the partial derivative of $\phi$ with respect to $u_1$ is equal to one everywhere, so that we can set $\lambda_1 = 1$. The conclusion to be drawn is that by enabling the solution of equations (38)-(40) rather than equation (39), c-programming extends the capabilities of fractional programming techniques. A solution strategy for this problem, integrating c-programming and fractional programming techniques, is discussed in Ref. [9].

5. CONCLUSIONS

Practically speaking, c-programming's merit is in the technique that it makes available for the solution of some challenging optimization problems—such as those cited above—that otherwise would have gone unsolved. Of no lesser importance, however, is the format that c-programming proposes for linking Problem $P$ with its linearized version Problem $P(\lambda)$. Although our discussion of problems falling under the c-programming format was confined to cases where $\phi$ satisfies certain convexity and differentiability conditions, it is clear that this format encompasses other cases
as well. For example, Geoffrion's [13] and Henig's [14] treatments of bicriterion optimization problems would fall within the c-programming format.

We may conclude then that the conceptual framework that c-programming provides for the analysis and solution of Problem $P$ opens new avenues for addressing it.

REFERENCES