MONOTROPIC PROGRAMMING: DESCENT ALGORITHMS AND DUALITY

R. T. Rockafellar[1]

ABSTRACT

A monotropic programming problem consists of minimizing, subject to linear constraints, a function of the form $F(x) = \sum_k f_k(\ell_k(x))$, where each ℓ_k is a linear function on R^n and each f_k is a convex function on R, not necessarily differentiable (e.g., piecewise linear or quadratic). In such problems, there are special ways of generating directions of descent, and duality can play a very strong role. More attention paid to these features may make it possible to solve problems of larger scale than otherwise.

[1]Research sponsored by the Air Force Office of Scientific Research, Air Force Systems Command, USAF, under grant no. 77-3204 at the University of Washington, Seattle.

ISBN 0-12-468662-1

1. INTRODUCTION

At one end of the spectrum of finite-dimensional optimization problems are the general nonlinear programming problems, involving constraint and objective functions that may or may not be differentiable. At the other end are problems of linear programming and network programming, which have a strongly combinatorial character. Convex programming problems lie somewhere in between. They benefit from duality phenomena, but not in so powerful a manner as do linear programming problems. They are often approached by the same techniques as are general problems in nonlinear programming; the main distinction is perceived in the fact that for one reason or another, these techniques work better when convexity is present.

There is a natural subclass of convex programming problems that, to our thinking, has not yet received adequate attention as a whole. This is roughly the largest class which exhibits combinatorial properties and duality to the same degree as in linear and network programming. The problems in question are those that can be formulated as separable convex programming problems with linear constraints. We speak of the subject as monotropic programming, for short. The word "monotropic", which means "turning or varying in one direction only", is used here to convey both the unidirectional curvature of the graphs of convex functions of a single variable and the monotonicity of their derivative relations.

Thus, it is intended as a synonym for one-dimensional convexity and proposed as a term to be used whenever one-dimensional convexity is paramount.

Just as a linear programming problem can be described qualitatively as one where a linear function is minimized over a convex polyhedron, so can a monotropic programming problem be described as one of the form

$$\begin{aligned} &\text{minimize } \Phi(w) \text{ over all } w \in K, \\ &\text{where } K \text{ is a convex polyhedron in } R^N \\ &\text{and } \Phi \text{ is a } \underline{\text{preseparable}} \text{ convex function on } R^N, \end{aligned} \tag{1.1}$$

which is to say Φ can be expressed as

$$\Phi(w) = \sum_{j=1}^{n} f_j(\ell_j(w)), \tag{1.2}$$

where each ℓ_j is a linear function on R^N and each f_j is a convex function on R. As a special case, if $f_j(x_j) = x_j^2$ for all j one has in (1.1) the general <u>quadratic</u> convex programming problem with linear constraints. It is not always to be supposed, however, that each f_j is differentiable or even given as a finite function on all of R.

To introduce the exact technical assumptions that will be needed, and to put problem (1.1) in a convenient "normalized" form at the same time, we specify now that for each $j = 1,\ldots,n$ we have

$$\text{a nonempty real interval } C_j, \tag{1.3}$$

not necessarily closed, possibly all of $R = (-\infty, \infty)$, and

$$\text{a convex function } f_j : C_j \to R \text{ which is continuous relative to } C_j. \text{ (Regard } f_j \text{ as } +\infty \text{ outside of } C_j.) \tag{1.4}$$

We denote the left and right endpoints of C_j (not necessarily contained in C_j and possibly infinite) by c_j^- and c_j^+, respectively, and make the following closure assumption:

$$\begin{aligned} &\text{if } c_j^+ < \infty \text{ but } c_j^+ \notin C_j, \text{ then } f_j(x_j) \to \infty \\ &\quad \text{as } x_j \uparrow c_j^+, \\ &\text{if } c_j^- > -\infty \text{ but } c_j^- \notin C_j, \text{ then } f_j(x_j) \to \infty \\ &\quad \text{as } x_j \downarrow c_j^-. \end{aligned} \tag{1.5}$$

These conditions provide flexibility that is useful in itself but turns out to be indispensible in setting up an adequate duality theory. (The device of regarding f_j as $+\infty$ outside of C_j allows us to identify pairs C_j, f_j satisfying (1.3), (1.4), (1.5), with closed proper extended-real-valued convex functions defined on all of R, see [3, §24].)

In this framework, we are ultimately interested in $\Phi(w)$ only for those values of w such that

$$\ell_j(w) \in C_j \quad \text{for} \quad j = 1,\ldots,n,$$

as well as $w \in K$. Note, though, that since C_j is an interval and ℓ_j is linear, the set of points w satisfying

(1.6) is itself a convex polyhedron, or at least it would be if each C_j were closed. In general it is a "partial convex polyhedron". (Some faces might be missing, but Φ goes to $+\infty$ on these according to (1.5)). For purposes of normalization, then, it is superfluous to represent the constraint $w \in K$ apart from (1.6); the convex polyhedron K could always be expressed by additional conditions as in (1.6) associated with closed intervals C_j and functions $f_j \equiv 0$ on C_j (so that f_j makes no contribution in (1.2)).

From this discussion it is clear that the following precise model can be adopted for monotropic programming: given C_j and f_j as in (1.3), (1.4), (1.5) for $j = 1,\ldots,n$, minimize the function (1.2) over all w satisfying (1.6). Actually, still another simplification is helpful, this time for the sake of duality. Letting

$$x_j = \ell_j(x) , \quad x = (x_1,\ldots,x_n) ,$$

observe from the linearity of ℓ_j that x ranges over a certain subspace C of R^n as z ranges over R^N. In the formulation we arrived at a moment ago, only the values $\ell_j(w)$ really take part in the action, not w itself. Every monotropic programming problem can therefore be reduced to the more fundamental form:

$$\begin{aligned} &\text{minimize} && F(x) = \sum_{j=1}^{n} f_j(x_j) \\ &\text{subject to} && x_j \in C_j \quad \text{for} \quad j = 1,\ldots,n , \qquad (P) \\ & && x = (x_1,\ldots,x_n) \in C , \end{aligned}$$

where C_j and f_j are as in (1.3), (1.4), (1.5), for $j = 1,\ldots,n$, and C is some subspace of R^n. This is a separable convex programming problem with linear constraints, some of which are given abstractly by the condition $x \in C$ but can be represented in other ways as the situation warrants.

Having thrown the spotlight on the problems that can be put in the form (P), we must say what it is about them that merits careful attention. First there is the fact that non-differentiable functions f_j can readily be accommodated by special mechanisms for finding directions of descent in (P). This is more important than it might seem.

The case of piecewise linear functions f_j serves as an illustration; such functions are not smooth, due to jumps in their slopes at certain "breakpoints". Suppose each f_j in (P) is piecewise linear with a sizeable number of pieces. Then, as is well-known, (P) could be recast as a linear programming problem and solved by the simplex method, say, but this would be at a great expense in dimensionality. The coefficient matrix for the linear programming problem would be exceedingly sparse.

There is advantage in methods capable of handling (P) without such a reformulation. The advantage becomes even more apparent when it is realized that in many situations where piecewise linear functions are at hand, they are there as approximations of more general functions. What if we were involved in a scheme of successive approximations involving more and more linear pieces? Reformulation as a linear programming problem at each stage would require higher and

higher dimensionality, whereas a direct method for (P) could avoid this and might be able to work "locally", with approximations generated around a point only as needed. A "local" method is difficult to implement, when the very identity of the variables in the problem can be destroyed by a series of reformulations.

The direction-finding methods referred to will be described in §2 and §4. Their interesting feature is that they generate descent directions, if such exist at all for the convex function F at a given point x, from only a limited and essentially finite class of vectors in C, called elementary vectors. It is here that the combinatorial underpinnings of the subject are apparent.

Much motivation comes from the network case, where C is the space of all "circulations" in a given directed graph (flows, not necessarily nonnegative, which are conserved at every node: x_j is the flow in arc j, and $\sum_{j=1}^{n} e_{ij}x_j = 0$ for all nodes i, where $((e_{ij}))$ is the node-arc incidence matrix of the network). The elementary vectors of C then correspond to "elementary circuits" in the network, and to say that descent in (P) is possible in such cases, is to say that F can be minimized by a series of corrections to the initial circulation x that are obtained in terms of flows around elementary circuits.

Obviously, this is a type of result that has no counterpart in general convex programming. The fact that it has a universal analogue in monotropic programming underlines the distinctive nature of that subject. The analogue in the non-network case is related to concepts in linear programming such

as descent along special rays forming the edges of the feasible region, instead of steepest descent. Computationally, it is closely connected with pivoting techniques for handling linear systems of variables.

Still another feature of monotropic programming problems that sets them apart from general convex programming problems is the existence of a duality theory that is as sharp and almost as constructively applicable as the one in linear programming. This theory has been available for some time [4], but it does not seem to be widely known. It assigns to (P) a dual monotropic programming problem (D) of the same fundamental form. In contrast with general convex programming, but in resemblance to linear programming, the dual (D) can often be written down explicitly. This is true, for instance, whenever the functions f_j are piecewise linear or quadratic, in which event (D) too involves functions that are piecewise linear or quadratic. Thus in particular, duality can be put to work in piecewise linear problems without having to pass through a linear programming reformulation.

In this paper, we not only demonstrate special descent vector properties of the kind mentioned above, but show how they can be invoked to get a new, constructive proof of the main duality theorems in monotropic programming.

2. OPTIMALITY AND ELEMENTARY DIRECTIONS OF DESCENT

In problem (P), the minimand F is a convex function defined on all of R^n but (cf. (1.4)) finite only when $x_j \in C_j$ for all j. The problem can be viewed simply as that of minimizing F over the subspace C, since the constraints $x_j \in C_j$ will be taken care of automatically by infinite penalties. Thus x is a feasible solution to (P) if and only if it is a point of C where F is finite, and it is an optimal solution if and only if, in addition,

$$F'(x;z) \geq 0 \quad \text{for all} \quad z \in C, \tag{2.1}$$

where of course

$$F'(x;z) = \lim_{t \downarrow 0} \frac{F(x+tz) - F(t)}{t}. \tag{2.2}$$

On the other hand, any $z \in C$ with $F'(x;z) < 0$, if one exists, gives a direction of descent from x: for small enough $t > 0$, $x + tz$ is another feasible solution to (P) and $F(x+tz) < F(x)$. The question to be addressed is: what are the special consequences of the separability of F?

Each f_j has a right derivative $f'_{j+}(x_j)$ and a left derivative $f'_{j-}(x_j)$ at every $x_j \in C_j$. These are nondecreasing functions of x_j that satisfy

$$-\infty \leq f'_{j-}(x_j) \leq f'_{j+}(x_j) \leq \infty.$$

Obviously, $f'_{j+}(x_j) = +\infty$ if $x_j = c_j^+ \in C_j$ (the case of a finite right endpoint), but $f'_{j+}(x_j) < +\infty$ if $x_j < c_j^+$. Likewise $f'_{j-}(x_j) = -\infty$ if $x_j = c_j^- \in C_j$, but $f'_{j-}(x_j) > -\infty$ if $x_j > c_j^-$.

Proposition 1: For any x satisfying $x_j \in C_j$ for all j and any $z \in R^n$, one has

$$F'(x;z) = \sum_{j:z_j>0} f'_{j+}(x_j)z_j + \sum_{j:z_j<0} f'_{j-}(x_j)z_j ,$$

where the convention is used in the sum that $(+\infty) + (-\infty) = +\infty$.

Proof: Substituting the formula for F into (2.2), one sees that $F'(x;z)$ is

$$\lim_{t\downarrow 0}\left[\sum_{j:z_j>0} \frac{f_j(x_j + tz_j) - f_j(x_j)}{t} + \sum_{j:z_j<0} \frac{f_j(x_j + tz_j) - f_j(x_j)}{t}\right].$$

The result is then obvious, except perhaps for the claim about $\pm\infty$. However, since each difference quotient is monotone in t by convexity, it can only approach $-\infty$ in the limit, whereas it can approach $+\infty$ only if it is actually $+\infty$ for all $t > 0$. In the latter case the overall limit defining $F'(x;z)$ must itself be $+\infty$. This explains why the convention gives the right answers; if none of the limits $f'_{j+}(x_j)z_j$ or $f'_{j-}(x_j)z_j$ of the individual difference quotients is $+\infty$, there is no difficulty about the sum, regardless of any $-\infty$'s, but if any one of them is $+\infty$, then the sum must be interpreted as $+\infty$.

As a matter of fact, $-\infty$ cannot enter into the formula in Proposition 1 unless there is some j such that $f'_{j+}(x_j) = -\infty$ or $f'_{j-}(x_j) = +\infty$. This is possible only in certain cases where $x_j = c_j^- > -\infty$ or $x_j = c_j^+ < +\infty$, respectively. Let us introduce the intervals

$$\partial f_j(x_j) = \{v_j \in R \mid f'_{j-}(x_j) \leq v_j \leq f'_{j+}(x_j)\}, \qquad (2.3)$$

$$\tilde{C}_j = \{x_j \in C_j \mid f'_{j+}(x_j) > -\infty \text{ and } f'_{j-}(x_j) < +\infty\}$$
$$= \{x_j \in C_j \mid \partial f_j(x_j) \neq \phi\}, \tag{2.4}$$

and speak of x as regularly feasible for (P) if $x \in C$ and $x_j \in \tilde{C}_j$ for $j = 1,\ldots,n$. Note that regular feasibility can differ only slightly, if at all, from feasibility itself; one always has

$$(c_j^-, c_j^+) \subset \tilde{C}_j \subset C_j \subset \text{cl } \tilde{C}_j \subset [c_j^-, c_j^+]. \tag{2.5}$$

The following conclusions can then be drawn.

Proposition 2: If x is a regularly feasible solution to (P), then $F'(x;z) > -\infty$ for all z. On the other hand, if x is a feasible solution to (P) that is not regularly feasible, but a regularly feasible solution $\tilde{x}$ does exist, then $F'(x;z) = -\infty$ for $z = \tilde{x} - x \in C$, and in particular x cannot be optimal.

The intervals $\partial f_j(x_j)$ are the subgradient sets of the functions f_j in the sense of convex analysis, as the notation suggests. For F itself, the subgradient set is by definition

$$\partial F(x) = \{v \in R^n \mid F(x') \geq F(x) + v \cdot (x'-x) \text{ for all } x' \in R^n\}, \tag{2.6}$$

and because of separability this reduces to

$$\partial F(x) = \partial f_1(x_1) \times \ldots \times \partial f_n(x_n). \tag{2.7}$$

Proposition 3: Suppose (P) has at least one regularly feasible solution. Then x is an optimal solution if and only if $x \in C$ and $\partial F(x) \cap \mathcal{D} \neq \phi$, where $\mathcal{D} = C^\perp$.

Proof: We know from Proposition 2 that any $x \in C$ which is optimal must be regularly feasible, i.e., have $\partial F(x) \neq \phi$. But since $\partial F(x)$ is a polyhedral convex set by (2.7), a regularly feasible solution x has $\partial F(x) \cap D = \phi$ if and only if $\partial F(x)$ and D can be separated strongly, i.e. there exists $z \in D^{\perp} = C$ such that

$$0 > \sup \{v \cdot z \mid v \in \partial F(x)\} . \tag{2.8}$$

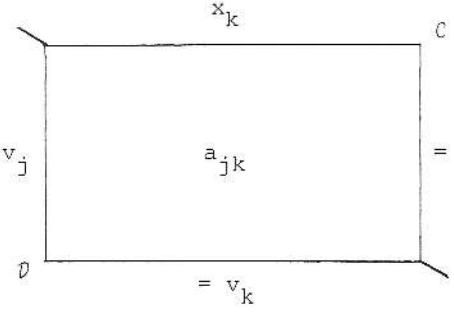
Recalling (2.3) and (2.7), we see that the right side of (2.8) coincides with the expression for $F'(x;z)$ in Proposition 1. Thus a regularly feasible x fails to have $\partial F(x) \cap D = \phi$ if and only if there exists $z \in C$ with $F'(x;z) < 0$, i.e. a direction of descent from x. This establishes the optimality criterion as stated.

The proof of Proposition 3 uses the fact that $\partial F(x)$ is a convex polyhedron, but there is more that can be said on the basis of $\partial F(x)$ actually being a product of intervals. Another concept is needed.

An elementary vector of the subspace C is a vector $z \in C$ such that $z \neq 0$, yet there does not exist another vector $z' \in C$ for which $\{j \mid z'_j \neq 0\}$ is a nonempty, proper subset of $\{j \mid z_j \neq 0\}$. It can readily be demonstrated (see [5], [3, §22]) that if z and z' are any two elementary vectors of C having $\{j \mid z_j \neq 0\} = \{j \mid z'_j \neq 0\}$, then z and z' are scalar multiples of each other. Thus, up to scalar multiples, C has only finitely many elementary vectors. Another way of saying this is that there are only finitely many "elementary directions" in C.

The combinatorial nature of this concept is well illustrated by the network case, where C is the space of circulations in a directed graph. Then the elementary vectors of C can be identified with the special flows which involve a fixed quantity flowing around a single elementary circuit in the graph and nothing flowing in any arcs that do not belong to the circuit. The elementary directions in C thus correspond one-to-one with the elementary circuits. (See [5], [6].)

In other cases too, it may be expected that in focusing on the elementary directions in C one is bringing to the fore whatever combinatorial structure is inherent in the constraint $x \in C$. Be that as it may, one can always fall back on a certain "tableau" representation of elementary vectors and deal with them computationally by means of pivoting operations. To see why this is so, recall that the condition $x \in C$ is equivalent to a system of homogeneous linear relations in the variables x_j. Such a system can be solved for a maximal set of variables in terms of the others so as to reduce it to a set of equations of the special form given by the rows in the following tableau:



(2.9)

This is a Tucker representation of C; a general vector of C is obtained by giving arbitrary values to the variables x_k at the top of the tableau and letting the variables x_j have the corresponding values given by the row equations in the tableau. Incidentally, the columns of such a tableau furnish at the same time a Tucker representation of the complementary subspace $D = C^\perp$.

There are only finitely many tableaus giving Tucker representations of C and D, and it is possible to pass from any one to any other by a sequence of pivoting transformations. We shall not go into this matter more deeply here but merely wish to point out the fundamental connection with elementary vectors (see [5], [6]): each column of a Tucker tableau as above yields in a certain way an elementary vector of C, and conversely every elementary vector of C can be obtained from some column of some Tucker tableau. (In similar fashion, rows of Tucker tableaus correspond to elementary vectors of the subspace D.) Thus the language of "elementary vectors" provides a means of describing the properties of homogeneous linear systems of equations that can be uncovered or manipulated in terms of pivotal algebra, but are not strictly tied to any one tableau. There is a close relationship between "elementary vectors" and the "basic solutions" (not necessarily feasible) that play a role in linear programming theory.

Elementary vectors are important in the present context because of the following result in the theory of linear inequalities.

Combinatorial Separation Theorem [5], [3, §22]: Given subspaces C and $D = C^{\perp}$ in R^n and any nonempty real intervals I_j, $j = 1,\ldots,n$ (not necessarily closed or bounded), one has

$$[I_1 \times \ldots \times I_n] \cap D = \phi$$

if and only if there is an elementary (!) vector z of C such that

$$v \cdot z < 0 \quad \text{for all} \quad v = (v_1,\ldots,v_n) \in I_1 \times \ldots \times I_n .$$

Applying this to the closed intervals $I_j = \partial f_j(x)$, we can draw a conclusion from Proposition 3 that reveals very clearly the combinatorial substructure that is a special feature of descent theory in monotropic programming.

Theorem 1: A regularly feasible solution x to (P) fails to be optimal if and only if there is a descent direction from x which is one of the finitely many elementary directions in C.

Summarizing the above results in operational terms, we have a special descent procedure that can be implemented in monotropic programming. Given any regularly feasible solution x to (P), we may form the nonempty closed intervals $\partial f_j(x_j)$ in (2.3) and test for the existence of

$$v = (v_1,\ldots,v_n) \in D \text{ with } v_j \in \partial f_j(x_j),\ j = 1,\ldots,n . \quad (2.10)$$

If such a v is found, then x is optimal (Proposition 3). If not, then there will be instead an elementary vector z of C giving a direction of descent from x.

Since $\mathcal{D}$ can be described by a system of linear equations and each $\partial f_j(x_j)$ is a closed interval, it is evident that the existence of v satisfying (2.10) could be settled as a problem in linear programming, by means of the simplex method, say. If v did not exist, this would be detected in such a way that a descent vector z would somehow be furnished instead. The trouble with a conventional linear programming approach, however, is that it would require a substantial reformulation of (2.10) to bring it in line with the "standard form" used for problems in linear programming. The fact that the exact character of the intervals $\partial f_j(x_j)$ cannot be told in advance makes matters worse. Any reformulation that involves supplementary variables, for instance, would in effect replace $\mathcal{D}$ by some other subspace and thereby alter the very combinatorial structure we wish to exploit.

What is better for purposes of monotropic programming is a subroutine that can determine the existence or nonexistence of v satisfying (2.10) simply by starting from any Tucker tableau (2.9) for $\mathcal{C}$ and $\mathcal{D}$ and performing a series of pivot steps dictated by the intervals $\partial f_j(x_j)$. In the case of nonexistence, it should be possible to obtain the desired elementary descent vector z from some column of the terminal tableau. Such methods are indeed possible; one is given in detail in [6, 10K-10L]. This is a topic that certainly deserves a great deal more investigation than it has received. Much of the advantage that might be gained in monotropic programming in terms of combinatorics and the ability to handle cost functions with "corners" seems to hinge on the development of a truly efficient subroutine along such lines.

Of course in special situations such as network programming, it may be possible to execute the test in a purely combinatorial manner. This too deserves more exploration. One could begin with cases where there is underlying structure similar to what is found in ordinary network problems, for example certain problems involving networks with gains or multicommodity flows.

3. CONJUGATE FUNCTIONS AND THE DUAL PROBLEM

We have already seen that the analysis of problem (P) leads to the consideration of certain vectors v in the subspace $\mathcal{D} = \mathcal{C}^{\perp}$. Now in general, any problem where an extended-real-valued convex function F (closed and proper) is minimized over C can be dualized to a problem where the negative of the conjugate convex function

$$G(v) = \sup_{x \in R^n} \{v \cdot x - F(x)\} \tag{3.1}$$

is maximized over $\mathcal{D}$. The difficulty one often runs into, though, is that the conjugate cannot be written down in "closed form" on the basis of the defining formula (3.1), although this is indeed possible in a number of highly significant cases.

We want to emphasize here that in monotropic programming the circumstances are much more favorable. Not only are there effective, alternative ways of constructing G, but because of separability, the duality theory that is obtained is unusually complete and exhibits several properties not enjoyed by other classes of convex programming problems.

To start with, the fact that $F(x) = \sum_j f_j(x_j)$, where f_j is a closed proper convex function on R (regarded as having the value $+\infty$ outside of C_j as explained in §1), yields at once via (3.1) that $G(v) = \sum_j g_j(v_j)$, where g_j is the closed proper convex function on R conjugate to f_j:

$$\begin{aligned} g_j(v_j) &= \sup_{x_j \in R} \{v_j x_j - f_j(x_j)\}, \\ f_j(x_j) &= \sup_{v_j \in R} \{v_j x_j - g_j(v_j)\}. \end{aligned} \tag{3.2}$$

All questions about conjugacy can therefore be reduced to one dimension. We shall see in a moment the special manner in which such questions can then be answered. The important thing to note right away is that the set

$$D_j = \{v_j \in R \mid g_j(v_j) < \infty\} \tag{3.3}$$

is a nonempty real interval; the pair D_j and g_j again satisfy not only (1.3) and (1.4), but also (1.5). Thus the information embodied in C_j, f_j is dualized to information of the same kind embodied in D_j, g_j, and the dualization is completely reversible.

Obviously $G(v) < \infty$ if and only if v belongs to $D_1 \times \ldots \times D_n$. The problem dual to (P) can therefore be stated as follows.

$$\begin{aligned} &\text{maximize} && -G(v) = -\sum_{j=1}^{n} g_j(v_j) \\ &\text{subject to} && v_j \in D_j \quad \text{for} \quad j = 1,\ldots,n, \\ & && v = (v_1,\ldots,v_n) \in \mathcal{D}. \end{aligned} \tag{D}$$

This is another monotropic programming problem in fundamental form, and its dual is in turn (P).

It may help the reader's appreciation of the scope of this form of duality to think not in terms of subspaces $\mathcal{C}$ and $\mathcal{D}$ but two homogeneous linear systems of variables as expressed mutually by some Tucker tableau of the type described in §2. (Any such tableau can be interpreted as giving Tucker representations for a complementarity pair of subspaces in R^n.) In the primal problem, we are concerned with the row system; each variable has an associated feasibility interval

(of arbitrary type), and a sum of costs for each variable must be minimized. Linear programming is the case where the costs are all linear on the intervals in question (maybe identically zero for certain variables). In this very general situation, without reducing to anything more uniform in its description even in the linear programming case, we can pass directly to a dual problem. In the dual we are concerned with the column system in the tableau. Again each variable has an associated feasibility interval and cost expression, and the sum of the costs must be minimized (its negative maximized).

Let us now tackle the issue of how g_j (and D_j) can be constructed from f_j (and C_j) without explicit calculation of the supremum in (3.1), even though such a calculation is sometimes quite easy. The main idea is to rely on the correspondence between closed proper convex functions and their derivative relations.

From f_j, we presumably know how to get the left and right derivative functions f_{j+}' and f_{j-}' on C_j and therefore the multifunction ∂f_j in (2.3). Denote by Γ_j the graph of ∂f_j; thus

$$\Gamma_j = \{(x_j, v_j) \mid v_j \in \partial f_j(x_j)\}, \tag{3.4}$$

$$\partial f_j(x_j) = \{v_j \mid (x_j, v_j) \in \Gamma_j\}. \tag{3.5}$$

Although we shall not go into the details here (see [3, §24]), the geometric appearance of Γ_j is very close to that of the graph of a nondecreasing function, except that it can include vertical as well as horizontal segments. The sets Γ_j that arise in this way can be characterized as the "maximal totally

ordered subsets of R^n." The crucial fact is that, just as Γ_j can readily be determined from f_j by differentiation, so can f_j be determined from Γ_j by integration. Specifically, one can begin with the interval

$$\tilde{C}_j = x_j\text{-projection of } \Gamma_j \tag{3.6}$$

and any point $\tilde{x}_j \in \tilde{C}_j$. The next step is to select any function $\gamma_j: \tilde{C}_j \to R$ such that $(x_j, \gamma_j(x_j)) \in \Gamma_j$ for all $x_j \in \tilde{C}_j$. (According to (3.5) and (2.3), the latter condition is equivalent to $f'_{j-} \leq \gamma_j \leq f'_{j+}$.) Then

$$f_j(x_j) = \int_{\tilde{x}_j}^{x_j} \gamma_j(t)dt + \text{const.} \quad \text{for } x_j \in \tilde{C}_j . \tag{3.7}$$

Leaving aside the choice of the constant of integration temporarily, we observe that this formula suffices to determine f_j not only on $\tilde{C}_j$ but everywhere else as well. There is a unique way of extending f_j to be continuous on the closure of $\tilde{C}_j$, and in view of (2.5), f_j has the value $+\infty$ at all points outside this closure. It is possible too for the extended f_j to have the value $+\infty$ at a finite endpoint of $\tilde{C}_j$ which does not belong to $\tilde{C}_j$. This the source of whatever discrepancy there may be between $\tilde{C}_j$ and the interval

$$C_j = \{x_j \in R \mid f_j(x_j) < \infty\} . \tag{3.8}$$

These facts are all explained more fully in [3, §24].

The key to the alternative construction of g_j, knowing Γ_j, is an elementary consequence of the definitions (3.2) and (3.4):

$$f_j(x_j) + g_j(v_j) - v_j v_j \geq 0 \quad \text{for all} \quad (x_j, v_j) \in R^2 ,$$
$$\text{and equality holds} \iff (x_j, v_j) \in \Gamma_j . \tag{3.9}$$

Since the relationship between f_j and g_j is entirely symmetric, it is clear from (3.9) that Γ_j describes the differential properties of g_j as well as it does those of f_j. The roles of x_j and v_j need only be reversed. Thus one has

$$\Gamma_j = \{(x_j, v_j) \in R^2 \mid x_j \in \partial g_j(v_j)\} , \tag{3.10}$$

$$\partial g_j(v_j) = \{x_j \mid (x_j, v_j) \in \Gamma_j\} , \tag{3.11}$$

where

$$\partial g_j(v_j) = \{x_j \in R \mid g'_{j-}(v_j) \leq x_j \leq g'_{j+}(v_j)\} . \tag{3.12}$$

In parallel with the construction of f_j from Γ_j, the interval

$$\begin{aligned} \tilde{D}_j &= \{v_j \in D_j \mid g'_+(v_j) > \infty \quad \text{and} \quad g'_{j-}(v_j) < +\infty\} \\ &= \{v_j \in D_j \mid \partial g_j(v_j) \neq \phi\} \end{aligned} \tag{3.13}$$

can be determined by

$$\tilde{D}_j = v_j\text{-projection of } \Gamma_j . \tag{3.14}$$

Taking any $\tilde{v}_j \in \tilde{D}_j$ and any function $\psi_j : \tilde{D}_j \to R$ such that $(\psi_j(v_j), v_j) \in \Gamma_j$, one obtains

$$g_j(v_j) = \int_{\tilde{v}_j}^{v_j} \psi_j(t)dt + \text{const.} \quad \text{for} \quad v_j \in \tilde{D}_j . \tag{3.15}$$

The next step is to extend g_j continuously to the closure of $\tilde{D}_j$ and give it the value $+\infty$ everywhere else; then D_j is recovered from (3.3). As for the constant of integration in (3.15), this is adjusted to the one in (3.7) by the equation

$$f_j(\bar{x}_j) + g_j(\bar{v}_j) - \bar{x}_j\bar{v}_j = 0 ,$$

which according to (3.9) must hold for any convenient choice of a point $(\bar{x}_j,\bar{v}_j)$ lying on Γ_j.

Two examples worth bearing in mind are listed in the next propositions. ("Piecewise" refers here to finitely many pieces.)

Proposition 4: The following are equivalent:

(a) f_j is piecewise linear $(\tilde{C}_j = C_j$, closed),

(b) g_j is piecewise linear $(\tilde{D}_j = D_j$, closed),

(c) Γ_j is a staircase relation (comprised of finitely many line segments, alternately vertical and horizontal).

Proposition 5: The following are equivalent:

(a) f_j is piecewise quadratic $(\tilde{C}_j = C_j$, closed),

(b) g_j is piecewise quadratic $(\tilde{D}_j = D_j$, closed),

(c) Γ_j is a polygonal relation (comprised of finitely many line segments, which may be vertical, horizontal, or of positive slope).

The dual of a piecewise linear monotropic programming problem is therefore piecewise linear, and the same for piecewise quadratic. Moreover, in these cases no distinction is necessary between "feasibility" and "regular feasibility".

Obviously there is no difficulty in passing from f_j to Γ_j to g_j by the method above when Γ_j is of such type.

To drive this point home and make apparent the directness and flexibility of this duality scheme in monotropic programming, we turn to the example of (P) as a general linear programming problem with both upper and lower bounds for each variable. Linear programming theory is incapable of producing a dual without first subjecting the problem to a transformation into one of the canonical forms where no single variable is bounded in both directions. In the monotropic programming context, however, we can regard a problem of this sort as arising from the row system of some Tucker tableau as in §2 together with the specification of an interval

$$C_j = [c_j^-, c_j^+] , \quad -\infty < c_j^- < c_j^+ < \infty ,$$

and cost term

$$f_j(x_j) = d_j x_j \quad \text{for} \quad x_j \in C_j$$

for every variable. The corresponding Γ_j then consists of three segments: the horizontal line segment joining the points (c_j^-, d_j) and (c_j^+, d_j) in R^2, the infinite vertical ray extending upward from (c_j^+, d_j) and the infinite vertical ray extending downward from (c_j^-, d_j). It is easy to see by either the integration technique or the definition (16) of g_j that

$$g_j(v_j) = \begin{cases} c_j^+(v_j - d_j) & \text{if} \quad v_j \geq d_j , \\ c_j^-(v_j - d_j) & \text{if} \quad v_j \leq d_j , \end{cases} \tag{3.16}$$

$$D_j = (-\infty, \infty) .$$

The dual consists of maximizing the negative sum of these expressions over the column system of variables in the tableau, there being no inequality constraints at all.

Note that in this example the dual of a linear programming problem turns out, in general, to be merely piecewise linear. It is no wonder, then, that linear programming theory cannot fully capture such duality. Other forms of linear programming problems can be handled similarly. In essence, one need only modify the formula in (3.16) in the obvious way to cover the cases where $c_j^- = c_j^+$ or $c_j^+ = \infty$ or $c_j^- = -\infty$; the feasibility interval D_j shrinks to $(-\infty,d_j]$ if $c_j^+ = \infty$, to $[d_j,\infty)$ if $c_j^- = -\infty$, and to $[d_j,d_j]$ in the case of both.

The monotone relations Γ_j have an importance far beyond their possible use in constructing g_j from f_j. The condition $(x_j,v_j) \in \Gamma_j$ turns out to be the correct generalization in monotropic programming of the familiar complementary slackness conditions in linear programming. To understand this better, let us look at the following equilibrium problem:

$$\text{find } x \in C \text{ and } v \in \mathcal{D} \text{ such that } (x_j,v_j) \in \Gamma_j \text{ for } j = 1,\ldots,n. \qquad (E)$$

The role of this problem in characterizing optimality is as follows.

Theorem 2: A pair (x,v) solves (E) if and only if x is an optimal solution to (P) and v is an optimal solution to (D).

Theorem 3: Except in the case where $\inf(P) = +\infty$ and $\sup(D) = -\infty$ (i.e., neither (P) nor (D) is feasible), one has

$$\inf(P) = \sup(D) .$$

These two theorems, which are closely related, have already been established elsewhere [5] by inductive arguments. One of the contributions of this paper will be to supply in §4 a constructive proof based on descent properties in monotropic programming that augment the ones already discussed in §2.

Actually, much of the content of the theorems can quickly be deduced right from the basic conjugacy relation in (3.9): adding over $j = 1,\dots n$, one sees that

$$\begin{aligned} &F(x) + G(v) - x\cdot v \geq 0 \quad \text{for all } x \in R^n, \\ &v \in R^n, \text{ with equality if and only if} \\ &(x_j, v_j) \in \Gamma_j \quad \text{for } j = 1,\dots,n . \end{aligned} \tag{3.17}$$

Since $x\cdot v = 0$ for $x \in C$, $v \in D$, it follows that

$$\begin{aligned} &F(x) \geq -G(v) \quad \text{for all } x \in C, \ v \in D, \\ &\text{with equality if and only if also} \\ &(x_j, v_j) \in \Gamma_j^*, \ j = 1,\dots,n . \end{aligned} \tag{3.18}$$

Thus $\inf(P) \geq \sup(D)$, and the solutions to (E) are the pairs (x,v) such that x is feasible in (P), v is feasible in (D), and $F(x) = -G(v)$. The only thing missing, in order for us to draw all the conclusions in Theorems 2 and 3, is the assurance that a "duality gap", where $\inf(P) > \sup(D)$, is impossible when either problem is feasible. This is what will be supplied in §4.

Incidentally, Theorem 3 is unusual in the annals of duality theory in not containing at the same time some assertion about the existence of an optimal solution to one problem or the other. Most duality theorems involve a primal or dual constraint qualification and assert either that $\inf(P) = \max(D)$ or $\min(P) = \sup(D)$. The results for geometric programming problems and the like are a partial exception (cf. [7], [8]) but they lack the symmetry of Theorem 3. This is not to say, however, that existence theorems are absent from monotropic programming. Indeed, existence criteria are quite simple and complete.

Theorem 4 [6]:

(a) An optimal solution to (P) exists if and only if (P) is feasible and (D) is regularly feasible.

(b) An optimal solution to (D) exists if and only if (D) is feasible and (P) is regularly feasible.

(c) (E) is solvable if and only if (P) and (D) are both regularly feasible.

This result will not be treated further in this paper, although some of it is already evident from the foregoing. A special case worth recording is the one where $\tilde{C}_j = C_j$ and $\tilde{D}_j = D_j$ for all j, as is true for instance in piecewise linear or piecewise quadratic monotropic programming (cf. Propositions 4 and 5). There, since "regular feasibility" is no different from "feasibility", the conclusion is that none of the problems (P), (D), (E) is solvable unless all three are solvable, and this holds if and only if (P) and (D) are both feasible.

In the network case particularly, the equilibrium problem (E) is interesting to interpret. Then C is the space of all "circulations" in a certain directed graph, as discussed earlier. The vectors v in the complementary space $\mathcal{D}$ are "tensions" arising from potential functions defined on the nodes of the graph; v_j is the rise in potential as the arc j is traversed in the direction of its orientation. The condition $(x_j;v_j) \in \Gamma_j$ express a certain relation that must be satisfied by the flow and tension in the arc j, a sort of generalization of the kind of resistance relation represented by Ohm's law in electrical networks. A problem of the form (E) can well arise on its own; corresponding problems (P) and (D), unique up to an additive constraint which makes no difference in the optimization, can be constructed from (E) by the integration techniques explained earlier in this section. Viewed in this light, Theorem 2 is a variational principle that characterizes the solutions to (E).

Minty in 1960 [2] was the first to present equilibrium problems in quite so broad and refined a framework, although he dealt only with the network case and did not capture details of the kind that depend on possible distinctions between C_j, D_j and $\tilde{C}_j$, $\tilde{D}_j$.

Another observation about problem (E) is this. Just as

$$\partial F(x) = \partial f_1(x_1) \times \ldots \times \partial f_n(x_n) ,$$

one also has

$$\partial G(v) = \partial g_1(v_1) \times \ldots \times \partial g_n(v_n) , \tag{3.19}$$

and hence by (3.5) and (3.11);

$$v \in \partial F(x) \iff x \in \partial G(v) \iff (x_j, v_j) \in \Gamma_j \quad \text{for} \quad j = 1, \ldots, n . \tag{3.20}$$

The conditions in (E) are therefore simply a symmetric version of the optimality conditions derived in more direct fashion for (P) in Proposition 3. Conclusion: in the kind of descent algorithm for (P) outlined in §2, when the test procedure produces a vector $v \in \partial F(x) \cap \mathcal{D}$ rather than a descent vector z, not only is x an optimal solution to (P) but v is an optimal solution to (D). Such an algorithm then, if it terminates at all, cannot help but solve both (P) and (D) simultaneously.

In §4, we will look at a modified version of the descent algorithm for (P) that produces an "approximately" optimal solution to (P) in predictably many iterations. Again it will turn out that the method produces such a solution not only for (P), but for (D) as well.

All this leads us to one of the most intriguing applications of the duality theory above, the dual approach to solving a given monotropic programming problem (P). This consists in executing a descent algorithm on (D) (perhaps we should speak of "ascent", since (D) has been stated as a maximization problem, but computationally (D) is equivalent to minimizing G over $\mathcal{D}$, and this symmetric interpretation is more convenient for now). The procedure is possible whenever (D) can be written down explicitly, as for instance in the piecewise linear or quadratic cases.

It goes as follows. Given any regularly feasible solution v to (D), i.e. $v \in \mathcal{D}$ satisfying $v_j \in \tilde{D}_j$ for $j = 1,\ldots,n$, we test for the existence of

$$x = (x_1,\ldots,x_n) \in C \text{ with } x_j \in \partial g_j(v_j),\ j = 1,\ldots,n. \quad (3.21)$$

If such an x is found, then (x,v) solves (E), so that x solves (P) and v solves (D) (Theorem 2). If not, there will instead be an elementary vector w of $\mathcal{D}$ giving a direction of descent for G from v. Replacing v by $v' = v + tw$ for some $t > 0$ (determined perhaps by a line search), one will have another regularly feasible solution to (D) with $G(v') < G(v)$.

Of course, (P) cannot actually be solved by such an approach unless the algorithm terminates at some state with an x satisfying (3.21). Otherwise one never even sees a candidate for x that might "approximately" be a solution to (P). For this reason the modified algorithm to be described in the next section, with its property of certain termination, is of particular relevance to the dual approach in monotropic programming.

4. FORTIFIED ALGORITHMS WITH GUARANTEED DESCENT

A vector z is said to give a direction of ε-descent in (P) from a feasible solution x, where $\varepsilon > 0$, if $z \in C$ and

$$F(x+tz) < F(x) - \varepsilon \quad \text{for some} \quad t > 0. \tag{4.1}$$

Then $x+tz$ is feasible solution to (P) which is better than x by more than ε. Clearly, such a z exists if and only if x is not an ε-optimal solution to (P) in the sense of having $F(x) \leq \inf(P) + \varepsilon$. Since F is convex, the difference quotient $[F(x+tz) - F(x)]/t$ is monotone in t, so any z which satisfies (4.1) in particular has $F'(x;z) < 0$, i.e., gives a direction of descent as defined in §2.

We now ask whether it might be computationally possible, by a modification of the basic algorithm §2, to generate directions of ε-descent for prescribed ε, perhaps even elementary directions of such type. This could provide a means of circumventing some of the convergence difficulties that might be encountered in naive descent, such as the production of a sequence of feasible solutions $\{x^i\}$ with $\lim_i F(x^i) > \inf(P)$.

A useful concept in the study of this matter is that of the ε-subgradient set of the convex function F at x:

$$\partial_\varepsilon F(x) = \{v \in R^n \mid F(x') \geq F(x) + v\cdot(x' - x) - \varepsilon, \ \forall x' \in R^n\}. \tag{4.2}$$

This may be compared with the ordinary subgradient set $\partial F(x)$ in (2.6), where ε is replaced by 0. Expressed in terms of the conjugate function G (think of (3.1) with x' in place of x) we have

$$\partial_\varepsilon F(x) = \{v \in R^n \mid F(x) + G(v) - x \cdot v \leq \varepsilon\}, \tag{4.3}$$

and since the inequality in this characterization is symmetric in x and v, we deduce that

$$v \in \partial_\varepsilon F(x) \iff x \in \partial_\varepsilon G(v), \tag{4.4}$$

cf. (3.17). If x is any point where F is finite, for instance any feasible solution to (P), then for all $\varepsilon > 0$ the set $\partial_\varepsilon F(x)$ is nonempty, closed and convex with

$$\sup_{v \in \partial_\varepsilon F(x)} v \cdot z = \inf_{t>0} \frac{F(x+tz) - F(x) + \varepsilon}{t} \quad \text{for all } z \in R^n. \tag{4.5}$$

(cf. [3, §23]).

Proposition 6: For $x \in C$ and $v \in \mathcal{D}$, one has $v \in \partial_\varepsilon F(x)$ if and only if $F(x) \leq -G(v) + \varepsilon$, in which event x is an ε-optimal solution to (P) and v is an ε-optimal solution to (D).

Proof: The first assertion is based on (4.3) and the complementarity of C and $\mathcal{D}$, while the second is a consequence of the fundamental inequality (3.18), which implies $\inf(P) \geq \sup(D)$.

Proposition 7: A vector z gives an ε-descent direction in (P) from a feasible solution x if and only if z strongly separates $\partial_\varepsilon F(x)$ from $\mathcal{D}$ in the sense that $z \perp \mathcal{D}$ and

$$\sup_{v \in \partial_\varepsilon F(x)} v \cdot z < 0 .$$

Proof: This is apparent from (4.5) and the relation $C = \mathcal{D}^{\perp}$.

Bertsekas and Mitter [1] discovered the fact in Proposition 7 in the case of a general convex function F and were the first to propose its use in descent algorithms. In principle, the procedure would be to take any feasible solution x to (P) and test for the existence of $v \in \partial_{\varepsilon}F(x) \cap \mathcal{D}$. If such a v is found, the conclusion is that x and v are ε-optimal solutions to (P) and (D) (Proposition 6). If not, then since $\partial_{\varepsilon}F(x)$ is convex one may hope to determine instead a vector z providing strong separation of $\partial_{\varepsilon}F(x)$ from $\mathcal{D}$ as in Proposition 7. In that event z gives a direction of ε-descent and one can pass by line search to a new feasible solution x' to (P) with $F(x') < F(x) - \varepsilon$.

There are serious difficulties in implementing such a procedure directly. Even with F a separable convex function as here, $\partial_{\varepsilon}F(x)$ is typically not a product of intervals (in contrast to $\partial F(x)$) but some nonpolyhedral convex set. The description of $\partial_{\varepsilon}F(x)$ by (4.2) or (4.3) may not lend itself to computation. Testing for the existence of $v \in \partial_{\varepsilon}F(x) \cap \mathcal{D}$ may be as hard a problem as (P) itself. Furthermore, while the nonexistence of such a v implies that $\partial_{\varepsilon}F(x) \cap \mathcal{D}$ can be separated, it does not necessarily (in cases where $\partial_{\varepsilon}F(x)$ is unbounded, as is true when x is not an interior point of $C_1 \times \ldots \times C_n$) ensure the possibility of strong separation.

Our aim is to demonstrate how to capitalize on the separability of F in monotropic programming by working instead with the product of the sets

$$\partial_\varepsilon f_j(x_j) = \{v_j \mid f_j(x_j') \geq f_j(x_j) + v_j(x_j' - x_j) - \varepsilon, \ \forall x_j'\}$$
$$= \{v_j \mid f_j(x_j) + g_j(x_j) - x_j v_j \leq \varepsilon\} . \tag{4.6}$$

Note as a one-dimensional specialization of the properties of ε-subgradients cited above that for any $x_j \in C_j$ and $\varepsilon > 0$, $\partial_\varepsilon f_j(x_j)$ is a nonempty closed interval:

$$\partial_\varepsilon f_j(x_j) = \{v_j \in R \mid \lambda_j^- \leq v_j \leq \lambda_j^+\} , \tag{4.7}$$

where λ_j^+ and λ_j^- are the possibly infinite values given by

$$\lambda_j^+ = \inf_{t>0} \frac{f_j(x_j + t) - f_j(x_j) + \varepsilon}{t} . \tag{4.8}$$

$$\lambda_j^- = \sup_{t<0} \frac{f_j(x_j + t) - f_j(x_j) + \varepsilon}{t} \tag{4.9}$$

In terms of practicality, much will depend on the ease with which λ_j^+ and λ_j^- can be determined, given x_j and ε. We shall return to this matter after a look at what it is we can accomplish if such values are available.

Proposition 8: (For any x satisfying $x_j \in C_j$ for $j = 1,\ldots,n$ (i.e. $F(x)$ finite) and any $\varepsilon > 0$, one has

$$\partial_\varepsilon F(x) \subset \partial_\varepsilon f_1(x_1) \times \ldots \times \partial_\varepsilon f_n(x_n) \subset \partial_{n\varepsilon} F(x) . \tag{4.10}$$

Proof: If $v \in \partial_\varepsilon F(x)$, we have

$$\sum_{j=1}^{n} [f_j(x_j) + g_j(x_j) - x_j v_j] \leq \varepsilon$$

by (4.3), and this implies via (3.9) and (4.6) that

$v_j \in \partial_\varepsilon f_j(x_j)$ for all j. The latter, on the other hand, implies itself via (4.6) that

$$\sum_{j=1}^{n} f_j(x_j) + \sum_{j=1}^{n} g_j(v_j) - \sum_{j=1}^{n} x_j v_j \leq n\varepsilon$$

and therefore by (4.3) that $v \in \partial_{n\varepsilon} F(x)$.

A variant of the Bertsekas-Mitter approach that we shall call the fortified descent algorithm for (P) can now be stated. Given any feasible solution x to (P) (which does not have to be regularly feasible, as was required by the descent algorithm in §2), test for the existence of

$$v = (v_1,\ldots,v_n) \in \mathcal{D} \quad \text{with} \quad v_j \in \partial_\varepsilon f_j(x_j)\ ,\ j = 1,\ldots,n\ . \tag{4.11}$$

(This test has the same character as the one in §2, since each $\partial_\varepsilon f_j(x_j)$ is a closed interval; it could be implemented in terms of the simplex method, if nothing else.) If such a v is found, then in particular $v \in \partial_{n\varepsilon} F(x)$ by virtue of Proposition 8, and we may conclude from Proposition 6 that x is an $n\varepsilon$-optimal solution to (P) and v is an $n\varepsilon$-optimal solution to (D), with

$$F(x) \leq -G(v) + n\varepsilon\ . \tag{4.12}$$

If not, there will be a vector z that strongly separates $\partial_\varepsilon f_1(x_1)\times\ldots\times\partial_\varepsilon f_n(x_n)$ from $\mathcal{D}$, indeed such a z which happens also to be an elementary vector of C (cf. the Combinatorial Separation Theorem §2). This z will be particular separate $\partial_\varepsilon F(x)$ strongly from $\mathcal{D}$ by the first inclusion

in Proposition 8, and hence it will provide a direction of ε-descent.

Before discussing the implementation of this method in finer detail, we draw some theoretical conclusions.

Remainder of proof of Theorems 2 and 3: As explained in §3, only the inequality $\inf(P) \leq \sup(D)$ is still needed in order to establish the validity of both theorems, and this only under the assumption that either $\inf(P) < \infty$ or $\sup(D) > -\infty$. If $\inf(P) < \infty$, the fortified descent algorithm is applicable starting from any feasible solution and for any $\varepsilon > 0$. If it continues for infinitely many iterations, then since the objective function decreases each time by more than ε, we know $\inf(P) = -\infty \leq \sup(D)$. If it terminates after finitely many iterations, it does so with feasible solutions x and v that satisfy (4.12). This inequality tells us that $\inf(P) \leq \sup(D) + n\varepsilon$. Since ε can be chosen arbitrarily, we can be sure therefore that $\inf(P) \leq \sup(D)$ in all cases where $\inf(P) < \infty$. To see that $\inf(P) \leq \sup(D)$ also in all cases where $\sup(D) > -\infty$, we need only invoke the symmetry of the relationship between (P) and (D).

Theorem 6: For any $\varepsilon > 0$ and any feasible solution x to (P) which is not $n\varepsilon$-optimal, there is an elementary direction of C which is an ε-descent direction from x.

This says roughly that a monotropic programming problem can be solved to any degree of accuracy, starting from any feasible solution (which does not have to be regularly feasible) and performing a series of line searches in elementary

directions of C alone. The algorithm actually enables us to give in advance an upper bound on the number of iterations that will be needed.

Let us state our computational goal as follows: given some $\delta > 0$ and $\alpha \in R$, as well as an initial feasible solution x^0 to (P), we want to determine a feasible solution x which is either δ-optimal or satisfies $F(x) \leq \alpha$. (In practical terms, α might, for instance, be a very large negative number that furnishes a stopping criterion for deciding "approximately" whether $\inf(P) = -\infty$.) The algorithm lets us accomplish this in no more than m iterations with $\varepsilon = \delta/n$, as long as

$$m \geq n[f(x^0) - \max\{\alpha, \inf(P)\}]/\delta .$$

If the algorithm terminates with a ε-optimal solution to (P), it provides also a ε-optimal solution to (D). If it terminates with $F(x) \leq \alpha$, the practical conclusion to be made about (D) (assuming α is a large negative number) is that (D) is "approximately" infeasible ($\sup(D) \approx -\infty$). Note that if a feasible solution v^0 to (D) is known at the start, one has $-G(v^0) \leq \inf(P)$, and it suffices to take

$$m \geq n[F(x^0) + G(v^0)]/\delta .$$

Further discussion is now in order about how the interval bounds (4.7) can be calculated, since all these are needed in a crucial way in every iteration. There are $2n$ of the values λ_j^+, λ_j^-, and formulas (4.8) and (4.9) appear to require a line search to determine each one. Things are not so

bad as this, however. Even if true line searches were required, they would at least be simplified by the fact that the difference quotients are convex as functions of $1/t$ (see [3, §23]). Also, the only λ_j's that need to be computed in any iteration are the ones which have changed from the preceding iteration, namely in the case of having arrived via a descent vector z, the ones for indices j such that $z_j \neq 0$. Interestingly enough, the effect of restricting attention to elementary descent vectors is to insist on having only a minimal set of indices that require updating in any iteration.

Nevertheless, the prime targets for success in implementing the fortified descent algorithm must be monotropic programming problems in which the formulas for λ_j^+ and λ_j^- can be replaced by expressions in closed form such as one might hope to have for the true derivations $f'_{j+}(x_j)$ and $(f'_{j-}(x_j)$, or if not that, at least by very simple subroutines requiring a relatively small, number of steps. Piecewise linear or quadratic problems fit this prescription, for example. When f_j is a piecewise linear convex function (with finitely many pieces), the minimization in (4.8) and maximization in (4.9) can be carried out discretely: only values of t which are breakpoints of f_j (points x_j where the slope jumps, $f'_{j-}(x_j) < f'_{j+}(x_j)$) need be inspected. The piecewise quadratic case can be worked out similarly. In that case, besides the breakpoints where the quadratic pieces are linked together, one must check for each interval whether the minimum (or maximum) of the difference expression in question is attained at an interior point of the interval. This is easy

to do, because the minimizing point is given by a simple formula in the quadratic case.

Quite a few possibilities can be explored here, not only in terms of special classes of functions f_j, but also various ways that the values of t which yield λ_j^+ and λ_j^- in (4.8) and (4.9) can be used in determining an appropriate step size for descent later in the direction of the vector z.

We conclude by underlining the fact that the fortified descent algorithm can be applied to (D) as well as to (P), and that this furnishes a second method of generating, in finitely many iterations, approximately optimal solutions to both (P) and (D).

REFERENCES

[1] D. P. Bertsekas and S. K. Mitter, "Descent numerical methods for optimization problem with nondifferentiable cost functions", SIAM Journal on Control 11, 1973, 637-652.

[2] G. J. Minty, "Monotone networks", Proc. Roy. Soc. London (Ser. A) 257, 1960, 194-212.

[3] R. T. Rockafellar, "Convex analysis", Princeton University Press, Princeton, New Jersey, 1970.

[4] R. T. Rockafellar, "Convex programming and systems of elementary monotonic relations", Journal of Mathematical Analysis and its Applications 19, 1967, 167-187.

[5] R. T. Rockafellar, "The elementary vectors of a subspace of R^n", in Combinatorial Mathematics and its Applications, Editors R. C. Bose and T. A. Dowling, University of North Carolina Press, Chapel Hill, North Carolina, 1969, 104-127.

[6] R. T. Rockafellar, "Optimization in networks and monotropic systems", book to appear.

[7] R. T. Rockafellar, "Some convex programs whose dual are linearly constrained", in Nonlinear Programming, Editors J. B. Rosen, O. L. Mangasarian and K. Ritter, Academic Press, New York, 1970, 293-322.

[8] R. T. Rockafellar, "Ordinary convex programs without a duality gap", Journal of Optimization Theory and Applications 7, 1971, 143-148.