A NEW CLASS OF PIECEWISE-LINEAR APPROXIMATION METHODS
FOR MATHEMATICAL PROGRAMMING

Thin-Yin Leong

November 1992
A NEW CLASS OF PIECEWISE-LINEAR APPROXIMATION METHODS
FOR MATHEMATICAL PROGRAMMING

Thin-Yin Leong

ABSTRACT

We propose a new approach for solving general constrained nonlinear maximization problems with concave objective functions. Neither the objective functions need to be differentiable nor the constraints need to be linear, convex, differentiable or separable. Stochastic programming problems have been shown to have deterministic nonlinear equivalents. The methods proposed are particularly useful for these problems as well because it requires few evaluations of the stochastic functions. Our inner-linearizing approximation always give feasible solutions. At the expense of solving problems of infinite size or infinite number of problems, approximations can be made exact. The approach is amenable to parallel computing or multi-pass adaptive procedures.

Department of Decision Sciences, National University of Singapore, 10 Kent Ridge Crescent, S 0511.
REPUBLIC OF SINGAPORE
Consider the nonlinear problem (P0) \( Z^*(\mathbf{X}) = \operatorname{Max} Z(\mathbf{X}) \) s.t. \( \mathbf{X} \in G \) where vector \( \mathbf{X} \equiv (x_1, \ldots, x_n) \) and \( Z \) is a concave function on the compact set \( G \subseteq (\mathbb{R}^n)^n \). It is not necessary for objective function \( Z \) to be differentiable or separable. The constraints that define feasible region \( G \) do not need to be linear, convex, differentiable or separable.

As described by Ermoliev and Wets [1988], any stochastic programming problem can be reduced to

\[
\begin{align*}
\text{(SP)} \quad Z^*_{SP} &= \operatorname{Min} E_\omega \{ f_0(\mathbf{X}, \omega) \} \\
\text{s.t.} \quad E_\omega \{ f_m(\mathbf{X}, \omega) \} &\geq 0, \quad m=1, \ldots, M \\
\mathbf{X} &\in G \subseteq \mathbb{R}^n.
\end{align*}
\]

\( f_m(\mathbf{X}, \omega) \geq 0, \ m=0, \ldots, M \), are functions where \( f_0 : \mathbb{R}^n \times \Omega \to \mathbb{R} \cup \{-\infty, +\infty\} \) and \( f_m : \mathbb{R}^n \times \Omega \to \mathbb{R} \). With appropriate selection of these functions, deterministic nonlinear problem (SP) can be distribution, recourse, or chance-constrained problems. This establishes the relationship between stochastic and nonlinear programming problems. Hence, without loss of generality, we can focus on nonlinear problems. In later sections, we explain why our nonlinear programming methods are better suited for solving stochastic programs than other nonlinear programming methods.

Studies on linear approximations to nonlinear programming problems have been done before. Recently, Feijoo and Meyer [1988] piecewise-linearize the convex objective function of their minimization problem by segmenting the function in the axis directions from a feasible solution. This make the
nonseparable objective function separable. This linear approximation of the objective function is a special case of the one used in our formulation. Their solution approach for the general case uses an iterative descent algorithm to search in the feasible set. To capitalize on linear programming methodology and its inherent benefits, the authors indicated that their problem is a linear program if the feasible set is given by a set of linear constraints. In this case, standard linear programming solution methods can be used. This reduces the computational burden associated with search routines. However, the requirement for the feasible set limits its use to other situations.

Chance-constrained problems with normal distributions and safety factors not less than zero have convex nonlinear equivalents. For this specialized structure, Hillier [1967] provided outer-linearizing approximations and Seppala, [1971] and [1972], provided inner-linearizing approximations. But the methodologies proposed are not directly applicable to other nonlinear programming problems in general. Earlier versions of the method proposed in this paper are in Bitran and Leong [1989] and Leong [1990]. There, the method was applied to chance-constrained problems with general probability distributions and linear objective functions. These problems include, as special cases, the problems considered by Hillier and by Seppala.

In this paper, we generalize Bitran-Leong's [1989] and Leong's [1990] approaches further and propose them as a class of methods for solving general nonlinear and stochastic programming problems. Our methods approximate the boundary of feasible set G with hyperplanes, each formed by connecting selected points on the boundary of a single constraint or a subset of constraints in G. The main principle behind the method is to make some 'guesses' about the relative magnitudes of the decision variables. For each guess, a multivariate constraint becomes a single variate constraint. Solving this gives a point on the boundary. We take selected rays (vectors from the origin), each corresponding to a 'guess', to give the set of selected boundary points. When a nonlinear constraint (or, collectively, a set of linear and/or nonlinear constraints) is convex, the approximated feasible region formed by the hyperplanes is a polyhedron whose extreme points are the
boundary points.

1. PIECEWISE-LINEAR APPROXIMATIONS

Without loss of generality, we can re-write (P0) as

\[
\begin{align*}
\text{(P)} & \quad Z^* = \text{Max } Z(X) \\
\text{s. t.} & \quad A_0 X \leq b_0 \\
& \quad g_m(X) \leq b_m, \quad m = 1, \ldots, M \\
& \quad x_i \geq 0, \quad i = 1, \ldots, n .
\end{align*}
\]

G is now given by a set of linear inequalities \( A_0 X \leq b_0 \) and nonlinear inequalities \( g_m(X) \leq b_m, m=1,\ldots,M \).

For any ray \( S \equiv (s_1, \ldots, s_n) \geq 0 \), there exist at least one scalar \( \theta \geq 0 \) such that \( g_m(\theta S) = b_m \). We let \( \varphi_m(s_1, \ldots, s_n) = \varphi_m(S) \) be the point at which ray \( S \) intersects the boundary of constraint \( g_m(X) \leq b_m \). We take \( n \) linearly independent rays \( S_i, i=1,\ldots,n \), to form cone \( [S_1, \ldots, S_n] \). We call this cone the candidate cone (CC). Let \( \{V_{ki} : V_{kj} = (v_{k1}, \ldots, v_{kn}) \geq 0, \quad i=1,\ldots,n\} \) be a set of linearly independent vectors. Hence, \( \{\Sigma_{j=1}^n v_{kj} S_j : i=1,\ldots,n\} \) is a set of \( n \) independent rays. We construct hyperplane \( \Sigma_{j=1}^n w_{mki} y_i = b_m \) with \( X = \Sigma_{j=1}^n S_j y_i \) to contain the points of intersection between these rays and constraint (3). Vector \( Y \equiv (y_1, \ldots, y_n) \) is vector \( X \) in the transformed space that has \( \{S_i : i=1,\ldots,n\} \) as axes. The coefficients for this hyperplane, \( w_{mk}, m=1,\ldots,M \) and \( k=1,\ldots,K \), is obtained by solving the following system of equations.

\[
\begin{bmatrix}
V_{k1} & w_{n1} \\
\vdots & \vdots \\
V_{kn} & w_{nn}
\end{bmatrix}
\begin{bmatrix}
\varphi_m(\Sigma_{j=1}^n v_{k1j} S_j) \\
\vdots \\
\varphi_m(\Sigma_{j=1}^n v_{knj} S_j)
\end{bmatrix}
= 
\begin{bmatrix}
\varphi_m(\Sigma_{j=1}^n v_{k1j} S_j) \\
\vdots \\
\varphi_m(\Sigma_{j=1}^n v_{knj} S_j)
\end{bmatrix}
\]

\[\text{(5)}\]
Vector set \( \{V_i : i=1,...,n\} \) should be selected such that hyperplanes are formed from boundary points that are adjacent to each other; other boundary points should not be obtainable by taking convex combinations of the boundary points forming a hyperplane.

\( (P1) \) is the approximation of \( (P) \) with \( G \) approximated by linear constraints.

\( (P1) \quad Z_{p1}^* = \text{Max } Z_{p1}(X) \)

s. t. \( A_0 X \leq b_0 \) \hspace{1cm} (6)

\[ \sum_{i=1}^{n} w_{mki} y_i \leq b_m : \pi_{mk}, \quad m=1,...,M, k=1,...,K \] \hspace{1cm} (7)

\[ X = \sum_{i=1}^{n} S_i y_i \] \hspace{1cm} (8)

\( y_i \) unrestricted in sign, \quad i=1,...,n \hspace{1cm} (9)

\( x_i \geq 0 , \quad i=1,...,n \) \hspace{1cm} (10)

where \( Z_{p1}(X) \equiv Z(X) \).

\( K \) is the number of linear constraints used to approximate each nonlinear constraint (3) and we denote the feasible set of \( (P1) \) as \( G_{p1} \). In the special case where \( G \) is formed by linear constraints, \( G_{p1} = G \) if the rays are selected such that they each passes through an extreme point of the polyheron \( G \). For convex \( G \), the original constraints are reproduced as \( K \) approaches infinity. That is, \( G_{p1} \to G \) as \( K \to \infty \).

In many situations and to quote Ermoliev and Wets [1988] ".. for well-formulated stochastic optimization problem, we may expect a lot of regularity, such as convexity of the feasible region, convexity and/or Lipschitz properties of the objective function, and so on. This is well documented in the literature."

Similarly, a lot of practical nonlinear problems also have convex feasible regions. We continue the discussion assuming \( G \) is convex. Later sections will examine what can be done when \( G \) is concave or contains concave subregions. Where analytical evidence for convexity is difficult to find, a rough check on the convexity of constraint \( g_m(X) \leq b_m \) within \( CC \) is
\[ |\varphi_m(S^*)| \geq \frac{1}{n} \sum_{i=1}^{n} |\varphi_m(S_i)| \] (11)

with the centroid ray \( S^* \equiv \sum_{i=1}^{n} S_i/|S_i| \) and \( S^* \in \text{CC} \). When inequality (11) is satisfied, the constraint \( g_m(X) \leq b_m \) is expected to be convex within \( \text{CC} \).

In principle, coefficient \( w_{mki} \) in (P1) is obtained by solving a system of equations, one for each \( k \). This requires a lot of computational effort, especially when \( K \) is large. Simpler methods are available. One simplifying trick is to select the vector set \( \{ V_{ki} : i=1,\ldots,n \} \) such that the matrix it forms has an underlying triangular structure. For illustration, we demonstrate one such method. Taking \( \varphi_m(S^*) \) and \( n-1 \) of the points in set \( \{ \varphi_m(S_i) : i=1,\ldots,n \} \), we form a hyperplane. The \( n \) linear inequalities resulting from the \( n \) hyperplanes thus formed approximate \( g_m(X) \leq b_m \). This is the same as letting \( V_{ki} \) be equal to the \( i \)-th canonical unit vector for \( i \neq k \) and \( V_{kk} = (1,\ldots,1) \). In this way, the system of equations (5) yields its solution easily. For \( m=1,\ldots,M, k=1,\ldots,K, \) and \( K=n \),

\[
\begin{cases}
\varphi_m(S_i), & \forall \ i \neq k \\
\varphi_m(S^*) - \sum_{j \neq k} \varphi_m(S_j), & i = k
\end{cases}
\] (12)

If the objective function \( Z(X) \) is piecewise-linear concave and separable, (P1) is a linear program. Otherwise, using a technique similar to one used by Feijoo and Meyer, we linearize \( Z(X) \). In linearizing, we temporarily restrict decision vector \( X \) to take only values along ray \( S_i \). The concave objective function \( Z(X) \) under this condition becomes single variable concave function \( Z(S_i y_i) \) in \( y_i \). For the \( n \) linearly independent rays in set \( \{ S_1,\ldots,S_n \} \), \( Z(X) \) becomes \( Z(\sum_i S_i y_i) \), a function in \( Y \). We now apply a standard approach in separable convex programming to piecewise-linearize \( Z(S_i y_i) \). Let
\[
Z(S, y_i) \approx \Sigma_{q=1}^{Q} c_{iq} y_{iq}, \quad \text{i=1,...,n} \tag{13}
\]

where \[ y_i = -\delta + \Sigma_{q=1}^{Q} y_{iq}, \quad \text{i=1,...,n} \tag{14} \]

and \[ 0 \leq y_{iq} \leq y_{u_{iq}}, \quad \text{i=1,...,n, q=1,...,Q}. \tag{15} \]

Decision variable \( y_i \) is divided into \( Q \) segments \( y_{iq} \), \( 0 \leq y_{iq} \leq y_{u_{iq}} \), and \( y_i \in [-\delta, -\delta + \Sigma_{q=1}^{Q} y_{u_{iq}}] \). Parameter \( y_{u_{iq}} \) is the maximum size of each segment. Parameter \( \delta > 0 \) is predefined to permit \( y_i \) to take on negative values. Our methods need \( \delta \) to be only a small real number. Parameter

\[
c_{iq} = \frac{Z(S, (-\delta + \Sigma_{j=1}^{q} y_{u_{ij}})) - Z(S, (-\delta + \Sigma_{j=1}^{q-1} y_{u_{ij}}))}{y_{u_{iq}}} \tag{16}
\]

is the slope of a piecewise-linear segment between \( x = -\delta + \Sigma_{j=1}^{q} y_{u_{ij}} \) and \( x = -\delta + \Sigma_{j=1}^{q-1} y_{u_{ij}} \).

Applying (13) to (16), (P1) becomes linear program (LP).

\[
(LP) \quad Z_{LP}^* = \text{Max} \Sigma_{i=1}^{n} \Sigma_{q=1}^{Q} c_{iq} y_{iq} \tag{17}
\]

s. t. constraints (6) to (10), (14) and (15).

We denote the feasible set of (LP) as \( G_{LP} \). The broad strategy in our methods is to look for solutions within \( CC \), supported by a mechanism that checks if there is an optimal solution in \( CC \). It is not necessary to know what \( X^* \) optimal to (LP) is as long as we know its location relative to \( CC \). For this reason, the reader should note that \( G_{LP} \subseteq G_{P1} \). Detail discussion about how \( CC \) can be selected is presented later. As with \( K \), the larger the \( Q \) the more accurate the approximation of \( Z(X) \). When the size of \( CC \) approaches zero and \( Q \) approaches infinity, the approximation of \( Z(X) \) becomes exact.

For simplicity of presentation, we have shown how each nonlinear constraint (3) is
piecewise-linearized. The shadow (dual) price of constraint (3) for each m, \( \pi_m \), is approximately equal to Max \( \{ \pi_{mk} \mid k=1,\ldots,K \} \). Because individual constraints are linearized, the number of linear constraints in the approximated \( G \) will be large; the M nonlinear constraints are replaced by \( M \times K \) linear constraints. In practice, the number of approximating linear constraints can be reduced by linearizing collectively all the constraints (linear and nonlinear) in \( G \). However, using this approach, the shadow prices of the individual constraints are lost. Hence, the post-analysis of the solution of will not be as complete. One compromise is to selectively choose and group constraints to linearize. Individual constraint or subsets of constraints (including a mix of linear and nonlinear constraints) can be linearized.

Problem (P) can be solved approximately in a single pass of solving (LP). Here we let \( S_i \) be the \( i \)th canonical unit vector. This method is applicable when the nonlinear constraints are almost linear. In this way, the methods proposed provided an answer to the perennial criticism that linear programs ignores the nonlinear reality of the actual world. Bitran and Leong, [1989a] and [1989b], applied the approach on production planning problems with random yields and service constraints. In these problems, the linear approximations to the first period constraints are exact. Planning on a rolling horizon basis, only the first period solution is implemented. The single-pass approach was shown to give good results. Another application is in hotel reservations planning (see [Bitran and Leong 1989c]).

When more accurate results is desired, an iterative improvement approach can be adopted. As opposed to having large \( K \), the iterative approach solves a series of smaller linear programs successively. This approach can be further adapted for parallel computing. The iterative approach may be the only way when the feasible set either is concave or has concave subregions. We present three iterative methods: one for convex \( G \), one for concave \( G \) and a heuristic for the general situation.

2. REDUCING CONE METHOD

The essence of this approach is to reduce CC while keeping the optimal solution to (P) in CC. We know that
a solution to (LP) is in CC when \( y_i \geq 0 \), for all \( i \). The following theorem makes use of this information to give a sufficient condition for when the optimal solution to the original problem is in CC.

**Theorem 1.** If \( G \) is convex and \( \mathbf{XA}^* \) optimal to (P1) with \( y_i \geq 0 \) for all \( i \), then there exists a \( \mathbf{X}^* \in G \cap CC \) optimal to (P).

**Sketch of proof.** All variables \( y_i \geq 0 \), \( i=1,...,n \), implies \( \mathbf{XA}^* \in CC \). For \( G \) convex, \( G_{p1} \cap CC \subseteq G \cap CC \) and \( G \cap CC' \subseteq G_{p1} \cap CC' \). That is, the linear approximation in (P1) is uniformly-tighter inside CC and uniformly-loosier outside CC. Hence, for \( \mathbf{XA}^* \in CC \) optimal to (P1),

\[
\text{Max}\{Z(\mathbf{X}) : \mathbf{X} \in G \cap CC\} \geq Z(\mathbf{XA}^*) = Z_{p1}(\mathbf{XA}^*) = Z_{p1}^*.
\]

But \( Z_{p1}^* \geq \text{Max}\{Z_{p1}(\mathbf{X}) = Z(\mathbf{X}) : \mathbf{X} \in G_{p1} \cap CC'\} \geq Z(\mathbf{X}) \), for all \( \mathbf{X} \in G \cap CC' \).

Hence, \( Z^* = \text{Max}\{Z(\mathbf{X}) : \mathbf{X} \in G \cap CC\} \).

Consider first the case where \( G \) is convex. After solving the (LP), we checked if the optimal solution to (LP) is in CC. If \( Z(\mathbf{X}) \) is linear, this solution is also the optimal solution to (P1). If \( Z(\mathbf{X}) \) is concave and the optimal solution to (LP) is well inside CC, we assume that optimal solution to (P1) is also in CC. Therefore, by theorem 1, the optimal solution to (P) is in CC. If stronger evidence is desired, \( Q \) can be increased to make (LP) closer to (P1).

For the next iteration, if the optimal solution is in CC, we reduce CC taking into account of the shadow prices \( r_{mk} \). The new CC should be in the subregion where the shadow prices are higher. Various ways can be devised to do this. As an example, suppose equation (12) is used to determine \( w_{mk} \). In this case, we can let \( S_i := S_i \), for \( i \neq r \) and \( S_r := (1 - \alpha) S_r / |S_r| + \alpha S*/|S*| \) where \((.,r) = \text{Arg Max}_{m,k} \{r_{mk}\} \) and \( \alpha \), \( 0 < \alpha \leq 1 \), is the predetermined reduction parameter. The CC is reduced such that \( CC^{\text{new}} \subset CC \). The centroid ray of CC, \( S_* \), is moved at each iteration and the boundary point associated with \( S_* \), \( \text{Min}_m(\varphi_m(S*)) \), gravitates towards the optimal solution to (P). If parameter \( \alpha \) is close to 0, convergence will be slow; if \( \alpha \) is close to 1, some \( y_i \) may go negative indicating that the optimal solution may have gone outside CC. In this case, CC needs to be moved
to 'recapture' the optimal solution. Again, shadow price information may be used to suggest the direction of move.

Alternatively, CC may be repositioned with the Max\[\{S_i, S_i^+\}\] as its new centroid. The size of CC is reduced to a fraction of its size in the previous iteration. We terminate the routine when the size of CC is small. Therefore, the algorithm converges in polynomial number of steps and each step solves a linear program (LP).

3. HONEY CONE METHOD

Suppose now G is concave. Then theorem 2 applies. For these cases, we devise another approach known as the honey cone method.

**Theorem 2.** If G is concave and \(X_A^*\) optimal to (P1) with \(y_i < 0\) for at least one i, then there exist a \(X_i^* \in G \cap CC'\) optimal to (P).

**Sketch of proof.** At least one \(y_i < 0\), implies \(X_A^* \in CC'\). For G concave, \(G \cap CC \subseteq G_{P1} \cap CC\) and \(G_{P1} \cap CC' \subseteq G \cap CC'\). That is, the linear approximation in (P1) is uniformly-looser inside CC and uniformly-tighter outside CC. Hence, for \(X_A^* \in CC'\) optimal to (P1),

\[
\max\{Z(X) : X \in G \cap CC'\} \geq Z(X_A^*) = Z_{P1}(X_A^*) = Z_{P1}^*.
\]

But \(Z_{P1}^* \geq \max\{Z_{P1}(X) = Z(X) : X \in G_{P1} \cap CC\} \geq Z(X)\), for all \(X \in G \cap CC\).

Hence, \(Z^* = \max\{Z(X) : X \in G \cap CC'\}\). □

The honey cone method moves CC when optimal solution to (LP) is not in CC. Obviously, CC should be moved towards where the shadow prices are higher and should not overlap the previous CC. As a result of shifting CC, part of the new CC may be outside \((R^+)^n\). We drop the part of CC outside \((R^+)^n\) by taking the projection of rays \(S_i, i=1,..,n\), on the boundary surface of \((R^+)^n\) to form the new CC. Each CC should also be deliberately made smaller than its predecessor but the rate of reduction must not be so fast that the procedure terminates before an optimal solution is found. The procedure described can be used as the recapture step in
the reducing cone method. Again, the algorithm should terminate in polynomial number of iterations.

In parallel computing environments, we can adapt the honey cone method as follows: CC is divided into mutually exclusive collectively exhaustive sub-CCs. Linearize each sub-CC and solve their corresponding (LP)s. Applying theorem 2, eliminate sub-CCs that has at least one $y_i > 0$. Sub-CCs remaining are further sub-divided and the procedure is repeated.

4. A HEURISTIC

When $G$ is generally convex with concave subregions, the reducing cone and the honey comb methods can be used together: the reducing cone approach is applied to convex subregions and the honey comb approach is applied to concave subregions. The difficulty lies in identifying the convexity property of subregions. Equation (11) was suggested earlier as a rough check. But still each step in both methods require a linear program to be solved which is quite a bit of work. We propose a heuristic that mimics the reducing cone methods and exploits the use of the honey cone method as the recapture step. The heuristic, based on theorems 1 and 2, do not need to solve linear programs.

First, using equation (12) to obtain the coefficients for the linear inequalities, we linearize $G$ as a whole. In a problem with $X \in (R^*)^n$, $G_{P1}$ now consists of $n$ linear constraints and $n$ nonnegativity constraints. The extreme ray and centroid boundary points are now denoted as $\phi(S_i)$, $i=1,..,n$, and $\phi(S*)$ respectively. In the reducing cone and honey comb methods, the dual variables are used to influence the next iteration. In the heuristic, we take the solution values of the boundary points $\{Z(\phi(S_i)) : 1=1,..,n\} \cup \{Z(\phi(S*))\}$ to modify CC.

We present below the details of the heuristic algorithm described.

**Algorithm.**

0. **Initialise.** $S_i =$ canonical $i$th unit vector, $i=1,...,n$. Set values for error tolerance parameter $\varepsilon$ (small) and parameters $\alpha$, $0 < \alpha \leq 1$; $\beta$, $0 < \beta \leq 1$; and $\gamma$, $0 < \gamma < 1$. 
1. **Check.** Let \( r = \text{Arg Max} \{ Z(\varphi(S_i)) : 1=1,...,n \} \cup \{ Z(\varphi(S_*)) \} \). If cone size \( \text{Min}_i( \frac{\| S_i \|}{\| S \|} \cdot \frac{\| S \|}{\| S_* \|} ) < \varepsilon \), terminate with \( \varphi(S_r) \) as the solution.

2. **Modify CC.** If \( r = * \), let \( S_j := (1 - \alpha) \frac{\| S_j \|}{\| S \|} + \alpha \frac{\| S \|}{\| S_* \|} \) where \( j = \text{Arg Min}_i(\{ Z(\varphi(S_i)) \}) \) and \( S_i \) unchanged for \( i \neq j \). Otherwise, let \( S_i := \beta \frac{\varphi(S_i)}{\| \varphi(S_i) \|} + (1 - \beta) \frac{\varphi(S_*)}{\| \varphi(S_*) \|} + \gamma \left( \frac{\| S_i \|}{\| S \|} - \frac{\| S \|}{\| S_* \|} \right) \) for \( i=1,...,n \). Repeat step 1.

In step 2 when the centroid solution is largest and hence equation (11) is satisfied, the ray associated with the smallest solution value is shifted towards the centroid. This reduces the size of CC and moves \( S_* \) away from where \( Z(.) \) is low. When a boundary point solution is largest, equation (11) may not be satisfied. In this case, the 'center' of CC is shifted towards \( S_r \) and the 'radial' vectors from the 'center' is reduced by a factor of \( \gamma \).

For illustration and computational experience, we test it on two simple problems: (EP1) and (EP2).

(EP1) \[ Z^* = \text{Max } cx_1 + x_2 \]
\[
\begin{align*}
\text{s.t.} & \quad x_1^2 + x_2^2 \leq 36 \\
& \quad -x_1 + x_2 \leq 5 \\
& \quad 3x_1 - x_2 \leq 9 \\
& \quad x_1, x_2 \geq 0
\end{align*}
\]

(EP2) \[ Z^* = \text{Max } cx_1 + x_2 \]
\[
\begin{align*}
\text{s.t.} & \quad (x_1 - 6)^2 + (x_2 - 8)^2 \leq 36 \\
& \quad -x_1 + x_2 \leq 5 \\
& \quad 3x_1 - x_2 \leq 9 \\
& \quad x_1, x_2 \geq 0
\end{align*}
\]

G in (EP1) is convex while G in (EP2) has a concave subregion. The heuristic is applied for values of c in
ranges where the same set of constraints are binding in the optimal solution. Parameters for the algorithm are arbitrarily chosen as follow: $\alpha = 1.0$, $\beta = 1.0$ and $\gamma = 0.9$. The results are in tables I and II.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|}
\hline
\toprule

\midrule

\bottomrule
\end{tabular}
\end{table}

The heuristic finds the optimal solution in all the cases tested and converges fairly quickly. For $\varepsilon = 0.05$, the algorithm terminates in about 5 iterations. Each iteration is completed very fast. Though the test problems are in $(\mathbb{R}^*)^2$, we expect the heuristic to work well in high dimensions.

5. COMMENTS AND SUMMARY

The methods have been applied to linear programming problems with individual chance-constraints. Other chance-constrained problems where the chance-constraints are nonlinear functions of decision vector $\mathbf{X}$ and joint chance-constrained problems can be solved too. We foresee the potential of applying the methods on all stochastic programs. As opposed to the new partitioning and scenario aggregation methods where constraints ('rows') are combined, our methods transform the decision variables ('columns'). The nonlinear optimization methods proposed in this paper are good for solving stochastic programs because they do not require too many evaluations of stochastic functions.

The methods may be used for large linear programming problems. The proposed approach reduces the size of problems and can give the $\varepsilon$-optimal solutions when solved iteratively. It seems ironical that whereas the interior point methods for solving linear programs makes linear programs nonlinear, our methods perform the converse. But we believe both methods can be used to complement each other. In general, the proposed approach has potential in problems where the feasible region boundaries are difficult to define or are implicitly defined, or when it is expensive to obtain information about the whole feasible region boundary. Applications include identifying products to load on FMS and capacity planning in complex job shops.
The proposed approach aggregates the decision variables by taking linear combinations of the variables. Linearly combining decision variables is like transforming the axes so that the ray corresponding the weights for combining variables becomes a new axis. Each aggregated variable is a variable along the transformed axes. Points where the rays intersect the boundary of G are used to construct hyperplanes. The half-space created by the intersection of hyperplanes forms a linear approximation of G. This suggests a new decision-making paradigm. Instead of determining values of decision variables, we determine the values of 'proposal' sets or alternatives. Each alternative comprises a collection of the decision variables of some relative proportions. This corresponds to an extreme CC ray $S_i$. In the iterative improvement methods, new alternatives are offered at each iteration. These alternatives sets are 'better' because they are modified from the old alternatives making use of past experience information. The best proposal is found when all alternatives are almost the same.

This new decision-making paradigm, with a genetic algorithm flavor, bridges the gap between mathematical programming and traditional decision-making. On one extreme, traditional decision-making evaluates an explicitly given set of alternatives which have been externally generated. Limiting choices to this set only, the 'best' solution is selected. On the other extreme, mathematical programming implicitly lists an infinitely large set of alternatives in the constraints. In fact in most instances, it is easier to specify constraints than to list alternatives. But ground experience can be used to suggest where certain regions in the defined feasible region may give better solutions.

The proposed methodology straddles between these extremes. It permits the specification of constraints and at the same time makes use of given alternatives. Further, it enhances decision-making by assisting in the generation of more and better alternatives by combining old ones. Further research is being pursued to extend this methodology to unconstrained optimization. It is conceivable that the approach would be to first create a "boundary" which contains the optimal solution and further reducing this. This is then
coupled with the methodology described in this paper. Again, only minimal requirements such as convexity and continuity is needed; differentiability and separability of functions are not necessary.
REFERENCES


Working paper #3097-89-MS.

paper #3108-89-MSA.


Addison-Wesley Publ., Reading, Mass.

Springer-Verlag, New York.


HILLIER, F S 1967. "Chance-Constrained Programming with 0-1 or Bounded Continuous Decision  


Management Science 17:736-749.

<table>
<thead>
<tr>
<th>Range</th>
<th>c</th>
<th>Optimum</th>
<th># Zₜ/Z* size</th>
<th># Zₜ/Z* size</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-∞,-1)</td>
<td>-2.0</td>
<td>(0.5)</td>
<td>1 1.000 0.765</td>
<td>5 1.000 0.028</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.0000</td>
<td>2 1.000 0.312</td>
<td>6 1.000 0.013</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3 1.000 0.140</td>
<td>7 1.000 0.006</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4 1.000 0.063</td>
<td></td>
</tr>
<tr>
<td>[-1,0.1565)</td>
<td>0.0</td>
<td>(0.9278,5.9278)</td>
<td>1 0.843 0.765</td>
<td>6 1.000 0.035</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.9278</td>
<td>2 0.963 0.312</td>
<td>7 1.000 0.032</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3 1.000 0.157</td>
<td>8 1.000 0.016</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4 1.000 0.078</td>
<td>9 1.000 0.014</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5 1.000 0.070</td>
<td>10 1.000 0.007</td>
</tr>
<tr>
<td>[0.1565,1.0631)</td>
<td>0.6</td>
<td>(3.0870,5.1450)</td>
<td>1 0.970 0.765</td>
<td>5 1.000 0.049</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.9972</td>
<td>2 0.989 0.390</td>
<td>6 1.000 0.025</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3 0.999 0.196</td>
<td>7 1.000 0.012</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4 0.999 0.098</td>
<td>8 1.000 0.006</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.8 (3.7482,4.6854)</td>
<td>1 0.994 0.765</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7.6840</td>
<td>2 0.994 0.390</td>
<td>7 1.000 0.037</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3 0.996 0.328</td>
<td>8 1.000 0.019</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4 0.997 0.165</td>
<td>9 1.000 0.009</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5 1.000 0.082</td>
<td></td>
</tr>
<tr>
<td>[1.0631,∞)</td>
<td>2.0</td>
<td>(4.3703,4.1109)</td>
<td>1 0.990 0.765</td>
<td>8 0.996 0.033</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12.8515</td>
<td>2 0.990 0.390</td>
<td>9 0.996 0.030</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3 0.996 0.328</td>
<td>10 0.996 0.015</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4 0.996 0.165</td>
<td>11 1.000 0.013</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5 0.996 0.146</td>
<td>12 1.000 0.012</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6 0.996 0.073</td>
<td>13 1.000 0.006</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7 0.996 0.066</td>
<td></td>
</tr>
</tbody>
</table>

**Table I.** Results for Problem (EP1).
<table>
<thead>
<tr>
<th>Range</th>
<th>c</th>
<th>Optimum</th>
<th>#</th>
<th>Z_d/Z^*</th>
<th>size</th>
<th>#</th>
<th>Z_d/Z^*</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-∞, -1)</td>
<td>-2.0</td>
<td>(0, 5)</td>
<td>1</td>
<td>1.000</td>
<td>0.765</td>
<td>5</td>
<td>1.000</td>
<td>0.028</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.0000</td>
<td>2</td>
<td>1.000</td>
<td>0.312</td>
<td>6</td>
<td>1.000</td>
<td>0.013</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>1.000</td>
<td>0.140</td>
<td>7</td>
<td>1.000</td>
<td>0.006</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>1.000</td>
<td>0.063</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[-1, 1.1684)</td>
<td>0.0</td>
<td>(0.5314, 5.5314)</td>
<td>1</td>
<td>0.904</td>
<td>0.765</td>
<td>5</td>
<td>0.984</td>
<td>0.035</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.5314</td>
<td>2</td>
<td>0.904</td>
<td>0.312</td>
<td>6</td>
<td>0.991</td>
<td>0.017</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>0.930</td>
<td>0.140</td>
<td>7</td>
<td>0.999</td>
<td>0.009</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>0.972</td>
<td>0.070</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[1.1684, ∞)</td>
<td>2.0</td>
<td>(3.8053, 2.4158)</td>
<td>1</td>
<td>0.861</td>
<td>0.765</td>
<td>8</td>
<td>0.980</td>
<td>0.054</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10.0264</td>
<td>2</td>
<td>0.861</td>
<td>0.390</td>
<td>9</td>
<td>0.980</td>
<td>0.048</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>0.875</td>
<td>0.328</td>
<td>10</td>
<td>0.999</td>
<td>0.024</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>0.889</td>
<td>0.282</td>
<td>11</td>
<td>0.999</td>
<td>0.012</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>0.911</td>
<td>0.244</td>
<td>12</td>
<td>0.999</td>
<td>0.011</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td>0.916</td>
<td>0.214</td>
<td>13</td>
<td>0.999</td>
<td>0.005</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7</td>
<td>0.980</td>
<td>0.107</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table II.** Results for Problem (EP2).
STATEMENT OF CONTRIBUTION

Traditional decision-making on one extreme evaluates a set of externally-generated, explicitly-given alternatives. Limiting choices to the alternatives given, the 'best' solution is selected. On the other extreme, mathematical programming implicitly lists an infinitely large set of alternatives using constraints. By evaluating a series of single alternatives sequentially, usual solution methods seek better and better solutions, and finally stopping at the optimum. In "A New Class of Piecewise-Linear Approximation Methods for Mathematical Programming", T-Y Leong introduces a new decision-making paradigm. This paradigm, with a genetic algorithm flavor, bridges the gap between mathematical programming and traditional decision-making. It makes use of constraints and given alternatives together. Decision-making is enhanced by cleverly combining alternatives to give better ones. This is particularly useful when solving the mathematical programming problem or defining the whole constraint space is hard. The new concept came about as a result of the development of a new class approximation methods for difficult mathematical programming problems. Potentially, further research may yield from this paradigm better methods or address more complex problems.