

**Limited Memory Space Dilation and Reduction Algorithms**

by  
Zafar A. Ansari

Thesis submitted to the Faculty of the  
Virginia Polytechnic Institute and State University  
in partial fulfillment of the requirements for the degree of  
Master of Science  
in  
Industrial and Systems Engineering

APPROVED:

---

Dr. Hanif D. Sherali, Chairman

---

Dr. S. C. Sarin

---

Dr. G. V. Loganathan

July 13, 1998  
Blacksburg, Virginia

# Limited Memory Space Dilation and Reduction Algorithms

by

Zafar A. Ansari

Dr. Hanif D. Sherali, Chairman

Industrial and Systems Engineering

(ABSTRACT)

In this thesis, we present variants of Shor and Zhurbenko's  $r$ -algorithm, motivated by the memoryless and limited memory updates for differentiable quasi-Newton methods. This well known  $r$ -algorithm, which employs a space dilation strategy in the direction of the difference between two successive subgradients, is recognized as being one of the most effective procedures for solving nondifferentiable optimization problems. However, the method needs to store the space dilation matrix and update it at every iteration, resulting in a substantial computational burden for large-sized problems. To circumvent this difficulty, we first develop a memoryless update scheme. In the space transformation sense, the new update scheme can be viewed as a combination of space dilation and reduction operations. We prove convergence of this new algorithm, and demonstrate how it can be used in conjunction with a variable target value method that allows a practical, convergent implementation of the method. For performance comparisons we examine other memoryless and limited memory variants, and also prove a modification of a related algorithm due to Polyak that employs a projection on a pair of Kelley's cutting planes. These variants are tested along with Shor's  $r$ -algorithm on a set of standard test problems from the literature as well as on randomly generated dual transportation and assignment problems. Our computational experiments reveal that the proposed memoryless space dilation and reduction algorithm (VT-MSDR) and the proposed modification of the Polyak-Kelly cutting plane method

(VT-PKC) provide an overall competitive performance relative to the other methods tested with respect to solution quality and computational effort. The  $r$ -Algorithm becomes increasingly more expensive with an increase in problem size, while not providing any gain in solution quality. The fixed dilation (with no reduction) strategy (VT-MSD) provides a comparable, though second-choice, alternative to VT-MSDR. Employing a two-step limited memory extension over VT-MSD sometimes helps in improving the solution quality, although it adds to computational effort, and is not as robust a procedure.

# Table of Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Literature Review</b>	<b>6</b>
2.1	Newton's Algorithm and Variants	6
2.1.1	Newton's Method	6
2.1.2	quasi-Newton Methods	8
2.1.3	Memoryless quasi-Newton Methods	13
2.1.4	Subgradient Algorithm	13
2.1.5	Step Length Selection Strategies	16
2.2	Variable Target Value Method	17
2.3	Polyak and Kelley's Methods	21
2.3.1	Polyak's Algorithm	21
2.3.2	Kelley's Cut Method	28
2.3.3	Polyak's 2-Kelley Cut Algorithm	29
2.4	Shor's Algorithm	30

<b>3</b>	<b>Proposed Modified Algorithms</b>	<b>32</b>
3.1	Memoryless Shor's $r$ -Algorithm	32
3.2	Modified Polyak-Kelley Algorithm	37
3.3	Memoryless Space Dilation and Reduction Algorithm and its Convergence	38
<b>4</b>	<b>Outline of Related Algorithmic Variants</b>	<b>53</b>
4.1	Variable Target Memoryless Space Dilation and Reduction Algorithm (VT-MSDR)	54
4.2	Shor and Zhurbenko's $r$ -Algorithm	54
4.3	Variable Target Memoryless Space Dilation Algorithm (VT-MSD)	55
4.4	Variable Target Limited Memory Space Dilation Algorithm VT-LMSD	55
4.5	Variable Target Polyak Kelley-Cut Algorithm (VT-PKC)	56
<b>5</b>	<b>Computational Experience</b>	<b>57</b>
<b>6</b>	<b>Conclusions and Recommendations for Future Research</b>	<b>62</b>
	<b>References</b>	<b>64</b>
	<b>Vita</b>	<b>68</b>

## List of Tables

Table 5.1. Computational Results Using Test Problems from the Literature . . . . .	60
Table 5.2. Computational Results for Dual Transportation and Assignment Problems . . . . .	61

# **Chapter 1**

## **Introduction**

Consider the nondifferentiable optimization (NDO) problem of minimizing a convex, though not necessarily differentiable, function  $f$  over  $E_n$ . (The case of  $x$  being restricted to lie in some convex subset  $X$  of  $E_n$  is treated subsequently.) This problem has received a great deal of attention over the past three decades, and several new methods have been proposed. Since nondifferentiability causes serious difficulties when the classical methods designed for differentiable problems are used, most algorithms for NDO use subgradients in some modified, related fashion for finding a direction of motion, and usually prescribe in closed-form a suitable step-size to take along this direction.

Depending on the particular strategy used for finding the direction of motion, algorithms for NDO can be categorized as follows. The *pure subgradient algorithm*, which is an analogue of the steepest descent method for differentiable problems, uses an anti-subgradient as the direction of motion. On the other hand, *deflected subgradient algorithms*, also called *conjugate subgradient algorithms*, imitate conjugate gradient methods. As with the steepest descent direction for the differentiable case, the anti-subgradient direction for the nondifferentiable case can result in a zigzagging phenomenon that might manifest itself at any stage of the subgradient algorithm, causing the procedure to crawl toward optimality. As a tool to overcome this

difficulty, a conjugate subgradient concept has been introduced in which the direction of motion is computed by combining the current anti-subgradient with the previous direction. Note that while some conjugacy requirement (see [6] and [12]) is enforced for the differentiable case, for the nondifferentiable case, this concept is used only as a strategy to deflect the anti-subgradient. The *Modified Gradient Technique (MGT)* of Camerini *et al.* [3] and the *Average Direction Strategy (ADS)* of Sherali and Ulular [38] are two well-known and extensively tested deflection strategies (see also Lemarechal [22], and Wolfe [46]).

As an alternate approach for NDO, *bundle type methods* have been proposed by Mifflin [28] and Lemarechal [23] in which the direction of motion is obtained via the convex hull of a set of previously generated subgradients, known as (subgradient) bundles. Unlike the choice of the step-length in subgradient based algorithms, these bundle methods involve an inexact line search that produces either an improved solution (serious step) or a trial solution that is rejected (null step). In either case, a new subgradient is computed and added to the existing bundle to find a modified direction of motion. Lemarechal [24] presents the concepts of this approach. Kiwiel [18, 19] and Lemarechal [26] introduce several different strategies to construct such bundles. Also, Kiwiel [20, 21] presents improved strategies for such methods under the title of *proximal bundle methods*. A major difference between bundle and ordinary subgradient methods is that, unlike the latter, the bundle type methods generate a sequence of iterates for which the objective function values are monotone decreasing. For this reason, the bundle type methods are classified as “descent methods.” However, one difficulty of the bundle methods is that they require the solution of a quadratic subproblem at each iteration for finding the direction of motion, and this can become quite expensive, particularly for larger sized problems.

In contrast with the foregoing methods, *variable metric methods* have different algorithmic and theoretical characteristics. For the differentiable case, we can obtain a deflected gradient by using a transformed metric based on the Hessian (or its approximation). Similarly, for NDO, a subgradient can be deflected by premultiplying it with a suitable matrix. Shor [40, 41] presents a *space dilation* (or “*dilatation*”) procedure that employs an analogous space transformation along the gradient direction. Goffin [9] presents convergence results for several different choices of transformation parameters. Of noteworthy value due to its computational performance, Shor and Zhurbenko [44] and Shor and Shabashova [43] propose the *r-algorithm*, which is a space dilation algorithm that dilates the space along the difference of two successive subgradients. Assuming that the selected anti-subgradient is almost perpendicular to the direction toward optimality, this *r-algorithm* is designed to reduce the orthogonal component of the subgradient with respect to the optimal direction, with the intent of alleviating the zigzagging phenomenon. In practice, the computational performance appears good, but the method is expensive because of the matrix storage and updating requirements. Moreover, its theory is complicated, and it loses the simplicity of subgradient based algorithms that have made the latter so popular.

To overcome these disadvantages, we propose the idea of adopting memoryless updates as in quasi-Newton methods (see [29]). This can be done by computing the space transformation operator at each iteration by updating the identity matrix instead of updating the previous approximation. Naturally, no matrix storage is required. Moreover, in this case, with a proper choice of parameters that admits convergence, we show that the memoryless update turns out to be a convex combination of two successive subgradients, leading to a combination of space dilation and space reduction operations.

From the viewpoint of computational interest, we also study three other variants of this scheme. In the first variant we adopt a simple fixed dilation strategy (as opposed to varying degrees of dilation or reduction) in the context of memoryless updates. In the second variant, we study a two-step *limited memory* update strategy. The third variant is motivated by the work of Polyak [32] where a projection onto a pair of Kelley's cutting planes, based on the current and the previous iterations, is adopted. This projection is shown by Polyak [32] to yield a direction that combines the subgradients generated at the present and the previous iterations in a specific manner. (Kim, Koh, and Ahn [15] propose a similar scheme in which a single projection is adopted onto a Kelley cutting plane that is generated based on previous iterates.) Since Polyak assumes a known optimal solution value, we embed this scheme in a variable target value method [36] where the target value replaces the (unknown) optimal solution value, leading to an implementable modification of Polyak's method.

The main purpose of this thesis is to present a theoretically convergent memoryless variant of Shor and Zhurbenko's  $r$ -algorithm, and from the viewpoint of computational interest, to test this against the  $r$ -algorithm along with other related memoryless and limited memory strategies that combine two successive subgradients in determining directions. The remainder of this thesis is organized as follows. In Chapter 2, pertinent background material is presented. In Chapter 3 the proposed memoryless and limited memory update schemes are presented, along with a modification of the Polyak-Kelly algorithm. Strategies for selecting values for the space transformation parameter based on relationships with deflected subgradient methods are presented, and related convergence properties are established. These properties enable us to embed this update scheme within a variable target value method [36] that guarantees overall convergence without any *a priori* information on the optimal objective function value. The

different aforementioned memoryless and limited memory variants that are tested herein, including Polyak's [32] scheme that leads to a direction that combines two successive subgradients, are briefly outlined in Chapter 4. Computational results and comparisons are presented in Chapter 5 using a set of standard test problems from the literature, as well as some randomly generated dual transportation and assignment problems. Finally, Chapter 6 summarizes the results and concludes the thesis.

## **Chapter 2**

### **Literature Review**

A general literature review is given for the subject matter pertinent to our research.

#### **2.1 Newton's Method and Variants**

The following section briefly describes Newton's Algorithm and some of its variants.

##### **2.1.1 Newton's Method**

The multidimensional Newton's method for minimizing a function deflects the steepest descent direction by premultiplying it by the inverse of the Hessian matrix. This method is motivated by finding a suitable direction for the quadratic approximation to the function rather than using a linear approximation to the function, as in the steepest descent method. To motivate the procedure, consider the following approximation  $q$  at a given point  $x_k$ :

$$q(x) = f(x_k) + \nabla f(x_k)^t (x - x_k) + \frac{1}{2} (x - x_k)^t H(x_k) (x - x_k)$$

where  $H(x_k)$  is the Hessian matrix of  $f$  at  $x_k$ . A necessary condition for a minimum of the quadratic approximation  $q$  is that  $\nabla q(x) = 0$ , or  $\nabla f(x_k) + H(x_k)(x - x_k) = 0$ . Assuming that the inverse of  $H(x_k)$  exists, the successor point  $x_{k+1}$  is given by

$$x_{k+1} = x_k - H(x_k)^{-1} \nabla f(x_k).$$

The above equation gives the recursive form of the points generated by Newton's method for the multidimensional case. Assuming that  $\nabla f(\bar{x}) = 0$ , that  $H(\bar{x})$  is positive definite at a local minimum  $\bar{x}$ , and that  $f$  is continuously twice-differentiable, it follows that  $H(x_k)$  is positive definite at points close to  $\bar{x}$ , and hence, the successor point  $x_{k+1}$  is well defined.

### Theorem 2.1.1

Let  $f: E_n \rightarrow E_1$  be continuously twice differentiable. Consider Newton's algorithm defined by the map  $A(x) = x - H(x)^{-1}\nabla f(x)$ . Let  $\bar{x}$  be such that  $\nabla f(\bar{x})$  and  $H(\bar{x})^{-1}$  exists. Let the starting point  $x_1$  be sufficiently close to  $\bar{x}$  so that this proximity implies that there exist  $k_1, k_2 > 0$  with

$k_1 k_2 = \|x_1 - \bar{x}\| < 1$  such that

1.  $\|H(x)^{-1}\| \leq k_1$

and by the Taylor series expansion of  $\nabla f$ ,

2.  $\|\nabla f(\bar{x}) - \nabla f(x) - H(x)(\bar{x} - x)\| \leq k_2 \|\bar{x} - x\|^2$

for each  $x$  satisfying  $\|x - \bar{x}\| \leq \|x_1 - \bar{x}\|$ . Then, the algorithm converges sublinearly to  $\bar{x}$  with, at least, an order-two convergence.

Proof: See Bazaraa, Sherali, and Shetty [2].

However, calculating the exact Hessian can be computationally intensive. To cut down on calculations, initially Davidon, then later Fletcher and Powell [7] proposed a method, where search directions are of the form  $d_j = -D_j \nabla f(y)$ , instead of  $-H^{-1}(y) \nabla f(y)$ , as in Newton's method. The Davidon-Fletcher-Powell (DFP) method belongs to the general class of *quasi-Newton methods*.

### **2.1.2 quasi-Newton Methods**

This method was originally proposed by Davidon and later developed by Fletcher and Powell [7]. The Davidon-Fletcher-Powell (DFP) falls under the general class of *quasi-Newton procedures*, where the search directions are of the form  $d_j = -D_j \nabla f(y)$ , in lieu of  $-H^{-1}(y) \nabla f(y)$ , as in Newton's method. The gradient direction is thus deflected by premultiplying it by  $-D_j$ , where  $D_j$  is an  $n \times n$  positive definite symmetric matrix. The positive definiteness property ensures that  $d_j$  is a descent direction whenever  $\nabla f(y) \neq 0$ , since then  $d_j^T \nabla f(y) < 0$ . For the purpose of the next step,  $D_{j+1}$  is formed by adding to  $D_j$  two symmetric matrices, each of rank one. For quadratic functions, this update scheme is shown later to produce the exact representation of the actual inverse Hessian within  $n$  steps.

#### **Theorem 2.1.2**

Let  $H$  be an  $n \times n$  symmetric positive definite matrix, and consider the problem to minimize  $f(x) = c^T x + \frac{1}{2} x^T H x$  subject to  $x \in E_n$ . Suppose that the problem is solved by the DFP method, starting with an initial point  $y_1$  and a symmetric positive definite matrix  $D_1$ . In particular, for  $j = 1, \dots, n$ , let  $\lambda_j$  be an optimal solution to the problem to minimize  $f(y_j + \lambda d_j)$  subject to  $\lambda \geq 0$ , and let  $y_{j+1} = y_j + \lambda_j d_j$ , where  $d_j = -D_j \nabla f(y_j)$  and  $D_j$  is determined by (2.1.1), (2.1.2), and (2.1.3). If  $\nabla f(y_j) \neq 0$  for each  $j$  then the directions  $d_1, \dots, d_n$  are  $H$ -conjugate and  $D_{n+1} = H^{-1}$ . Furthermore,  $y_{n+1}$  is an optimal solution to the problem.

Proof: The proof of this theorem, given in [2], is based on showing that for any  $j$  with  $1 \leq j \leq n$ , we must have the following conditions:

1.  $d_1, \dots, d_j$  are linearly independent.

2.  $d_i^t H d_k = 0$  for  $i \neq k; i, k \leq j$ .

3.  $D_{j+1} H p_k = p_k$ , or equivalently,  $D_{j+1} H d_k = d_k$  for  $1 \leq k \leq j$ , where  $p_k = \lambda_k d_k$ .

## Summary of the DFP Method

We now summarize the DFP method for minimizing a differential multi-variable function.

### Initialization Step

Let  $\varepsilon > 0$  be a termination tolerance. Choose an initial point  $x_1$  and an initial symmetric positive definite matrix  $D_1$ . Let  $y_1 = x_1$ , let  $k = j = 1$ , and go to the Main Step.

### Main Step

1. If  $\|\nabla f(y_j)\| < \varepsilon$ , stop; otherwise, let  $d_j = -D_j \nabla f(y_j)$  and let  $\lambda_j$  be an optimal solution to the problem to minimize  $f(y_j + \lambda d_j)$  subject to  $\lambda \geq 0$ . Let  $y_{j+1} = y_j + \lambda_j d_j$ . If  $j < n$ , go to Step 2. If  $j = n$ , let  $y_1 = x_{k+1} = y_{n+1}$ , replace  $k$  by  $k+1$ , let  $j = 1$ , and repeat Step 1.

2. Construct  $D_{j+1}$  as follows:

$$D_{j+1} = D_j + \frac{p_j p_j^t}{p_j^t q_j} - \frac{D_j q_j q_j^t D_j}{q_j^t D_j q_j} \quad (2.1.1)$$

where

$$p_j = \lambda_j d_j \equiv y_{j+1} - y_j \quad (2.1.2)$$

$$q_j = \nabla f(y_{j+1}) - \nabla f(y_j) \quad (2.1.3)$$

replace  $j$  by  $j+1$ , and repeat the Main Step.

The inner loop of the foregoing algorithm resets the procedure every  $n$  steps (whenever

$j = n$  at Step 1). Any variant that resets every  $n' < n$  inner iteration steps is called a *partial quasi-Newton method*. This strategy can be useful from the viewpoint of conserving storage when  $n' \ll n$ , since then the inverse Hessian approximation can be stored implicitly by instead storing only the generating vectors  $p_j$  and  $q_j$  themselves within the inner loop iterations.

At each step of the DFP method we have seen that, given some approximation  $D_j$  to the inverse Hessian matrix, we computed the search direction  $d_j \equiv -D_j \nabla f(y_j)$  by deflecting the negative gradient of  $f$  at the current solution  $y_j$ , using this approximation  $D_j$  in the spirit of Newton's method. We then performed a line search along this direction and, based on the resulting solution  $y_{j+1}$  and the gradient  $\nabla f(y_{j+1})$  at this point, we obtained an updated approximation  $D_{j+1}$  according to (2.1.1), (2.1.2) and (2.1.3). As seen in Theorem 2.1.2, if  $f$  is a quadratic function given by  $f(x) = c'(x) + \frac{1}{2}x'Hx$ ,  $x \in E_n$ , where  $H$  is symmetric and positive definite; and if  $\nabla f(y_j) \neq 0$ ,  $j = 1, \dots, n$ , then we indeed obtain  $D_{n+1} = H^{-1}$ . In fact, observe from parts 1 and 3 of Theorem 2.1.2 that, for each  $j \in \{1, \dots, n\}$ , the vectors  $p_1, \dots, p_j$  are linearly independent eigenvectors of  $D_{j+1}H$  with eigenvalues equal to 1. Hence at each step of the method, the revised approximation accumulates one additional linearly independent eigenvector, with a unit eigenvalue for the product  $D_{j+1}H$  until  $D_{n+1}H$  finally has its  $n$  eigenvalues equal to 1, giving  $D_{n+1}HP = P$ , where  $P$  is the nonsingular matrix of eigenvectors of  $D_{n+1}H$ . Hence,

$$D_{n+1}H = I, \text{ or } D_{n+1} = H^{-1}.$$

Based on this observation, let us derive the update scheme for the DFP method and other prominent quasi-Newton methods. Toward this end suppose that we have some symmetric, positive definite approximation  $D_j$  of the inverse Hessian matrix for which  $p_1, \dots, p_{j-1}$  are the eigenvectors of  $D_jH$  with unit eigenvalues. (For  $j = 1$  no such vectors exist.) Adopting the

inductive scheme of the theorem, assume that these eigenvectors are linearly independent and are H-conjugate. Now, given the current point  $y_j$ , we conduct a line search along the direction  $d_j = -D_j \nabla f(y_j)$  to obtain the new point  $y_{j+1}$  and, accordingly, we define

$$\begin{aligned} p_j &= y_{j+1} - y_j \text{ and} \\ q_j &= \nabla f(y_{j+1}) - \nabla f(y_j) \equiv H(y_{j+1} - y_j) = H p_j. \end{aligned} \quad (2.1.4)$$

Following the argument in the proof of Theorem 2.1.2, the vectors  $p_k \equiv \lambda_k d_k$ ,  $k = 1, \dots, j$ , are easily shown to be linearly independent and H-conjugate. We now want to construct a matrix

$$D_{j+1} = D_j + C_j$$

where  $C_j$  is some symmetric correction matrix that ensures that  $p_1, \dots, p_j$  are eigenvectors of  $D_{j+1}H$  with unit eigenvalues. Hence, we want  $D_{j+1}H p_k = p_k$  or, from (2.1.4) that  $D_{j+1} q_k = p_k$  for  $k = 1, \dots, j$ . For  $1 \leq k < j$ , this translates to requiring

$$\begin{aligned} p_k &= D_j q_k + C_j q_k = D_j H p_k + C_j q_k = p_k + C_j q_k, \text{ or that} \\ C_j q_k &= 0 \text{ for } k = 1, \dots, j-1. \end{aligned} \quad (2.1.5)$$

For  $k \equiv j$ , the aforementioned condition

$$D_{j+1} q_j = p_j \quad (2.1.6)$$

is called the *quasi-Newton condition*. This condition translates to the requirement that

$$C_j q_j = p_j + D_j q_j \quad (2.1.7)$$

Now, if  $C_j$  had a symmetric rank-one term  $p_j p_j^t / p_j^t q_j$ , then  $C_j q_j$  operating on this term would yield  $p_j$ , as required in (2.1.7). Similarly, if  $C_j$  had a symmetric rank-one term  $\frac{-(D_j q_j)(D_j q_j)^t}{(D_j q_j)^t q_j}$ ,

then  $C_j q_j$  operating on this term would yield  $-D_j q_j$ , as required in (2.1.7). This therefore leads to the *rank-two DFP update* (2.1.1) via the correction term

$$C_j = \frac{p_j p_j^t}{p_j^t q_j} - \frac{D_j q_j q_j^t D_j}{q_j^t D_j q_j} \equiv C_j^{DFP} \quad (2.1.8)$$

which satisfies the quasi-Newton condition (2.1.6) via (2.1.7). Moreover (2.1.5) also holds since, for any  $k \in \{1, \dots, j-1\}$ , we have from (2.1.4) and (2.1.8) that

$$C_j q_k = C_j H p_k = \frac{p_j p_j^t H p_k}{p_j^t q_j} - \frac{D_j q_j p_j^t H D_j H p_k}{q_j^t D_j q_j} = 0$$

since  $p_j^t H p_k = 0$  in the first term and  $p_j^t H D_j H p_k = p_j^t H p_k = 0$  in the second term as well.

Hence, following this sequence of corrections, we shall ultimately obtain  $D_{n+1} H = I$  or  $D_{n+1} = H^{-1}$ .

To further illustrate this convergence property, take the following example.

Consider the problem:

$$\text{Minimize: } -12x_2 + 4x_1^2 + 4x_2^2 + 4x_1x_2$$

Note that the Hessian matrix  $H$  is given by

$$H = \begin{bmatrix} 8 & -4 \\ -4 & 8 \end{bmatrix}$$

We generate two conjugate directions,  $d_1$  and  $d_2$ . Suppose we choose  $d_1^t = (1, 0)$ . Then,

$d_2^t = (a, b)$  must satisfy  $0 = d_1^t H d_2 = 8a - 4b$ . In particular, we may choose  $a = 1$  and  $b = 2$  so that

$d_2^t = (1, 2)$ . It may be noted that the conjugate directions are not unique.

If we minimize the objective function  $f$  starting from  $x_1^t = (-\frac{1}{2}, 1)$  along the direction  $d_1$ ,

we get the point  $x_2^t = (\frac{1}{2}, 1)$ . Now, starting from  $x_2$  and minimizing along  $d_2$ , we get  $x_3^t = (1, 2)$ .

Note that  $x_3$  is the minimizing point.

Therefore starting from any point if we minimize in directions  $d_1$  and  $d_2$ , we reach the optimal point in, at most, two steps.

### **2.1.3 Memoryless quasi-Newton Methods**

Suppose that at each iteration, instead of using the previous Hessian approximation to calculate the current Hessian approximation, an identity matrix is used. Therefore, we are “forgetting” the previous Hessian approximation, hence the name *memoryless quasi-Newton method*. This method is related to the conjugate gradient methods as discussed in Bazaraa et al. [2], and have the same n-step convergence property of quasi-Newton methods for minimizing quadratic functions. The storage requirements are similar to conjugate gradient methods, and inexact line searches can be performed. Also the loss of positive definiteness of Hessian approximations in the quasi-Newton method are no longer of concern.

### **2.1.4 Subgradient Algorithm**

The above discussion holds true only for differentiable problems. However, we now describe an algorithm which develops a direction finding procedure in the absence of differentiability. It is called the subgradient algorithm.

Consider the problem P defined as

$$P: \text{Minimize } \{f(x): x \in X\} \tag{2.1.9}$$

where  $f: E_n \rightarrow E_1$  is a convex but not necessarily differentiable function, and where  $X$  is a nonempty, closed, convex subset of  $E_n$ . We assume that an optimal solution exists as would be, for example, if  $X$  is bounded, or if  $f(x) \rightarrow \infty$  whenever  $\|x\| \rightarrow \infty$ .

For Problem P, we now describe a subgradient optimization algorithm that can be viewed as a direct generalization of the steepest descent algorithm in which the negative gradient direction is substituted by the negative subgradient-based direction. However the latter direction need not necessarily be a descent direction, although it does result in the new iterate being closer to an optimum solution for a sufficiently small step size. Because of this reason we do not perform a line search along the negative subgradient direction, but rather prescribe a step size at each iteration that guarantees that the sequence generated will eventually converge to an optimum solution. Also, given an iterate  $x_k \in X$  and adopting a step size  $\lambda_k$  along the direction  $d_k = -\xi_k / \|\xi_k\|$ , where  $\xi_k$  belongs to the subdifferential  $\partial f(x_k)$  of  $f$  at  $x_k$  ( $\xi_k \neq 0$ , say), the resulting point  $\bar{x}_{k+1} = x_k + \lambda_k d_k$  need not belong to  $X$ . Consequently the new iterate  $x_{k+1}$  is obtained by projecting  $\bar{x}_{k+1}$  onto  $X$ , that is, finding the (unique) closest point in  $X$  to  $\bar{x}_{k+1}$ .

We denote this operation as  $x_{k+1} = P_X(\bar{x}_{k+1})$ , where

$$P_X(\bar{x}_{k+1}) \equiv \operatorname{argmin} \{\|x - \bar{x}_{k+1}\| : x \in X\}.$$

## Summary of a Subgradient Algorithm

### Initialization Step

Select a starting solution  $x_1 \in X$ , let the current upper bound on the optimal objective value be  $UB_1 = f(x_1)$ , and let the current incumbent solution be  $x^* = x_1$ . Put  $k = 1$ , and go to the main step.

### Main Step

Given  $x_k$ , find the subgradient  $\xi_k \in \partial f(x_k)$ , of  $f$  at  $x_k$ . If  $\xi_k = 0$ , then stop;  $x_k$  (or  $x^*$ ) solves Problem P. Otherwise, let  $d_k = -\xi_k / \|\xi_k\|$ , select a step size  $\lambda_k > 0$ , and compute  $x_{k+1} = P_X(\bar{x}_{k+1})$ ,

where  $\bar{x}_{k+1} = x_k + \lambda_k d_k$ . If  $f(x_{k+1}) < \text{Ub}_k$ , put  $\text{Ub}_{k+1} = f(x_{k+1})$  and  $x^* = x_{k+1}$ . Otherwise, let  $\text{Ub}_{k+1} \equiv \text{Ub}_k$ . Increment  $k$  by 1 and repeat the Main Step.

Note that the stopping criteria  $\xi_k = 0$  may never be realized. Even if there exists an interior point optimum and we do find a solution  $x_k$  for which  $0 \in \partial f(x_k)$ , this sufficient condition for optimality may not be realized because of the arbitrarily selected subgradient  $\xi_k$  generated by the algorithm. Hence, a practical stopping criterion based on a maximum limit on the number of iterations performed is almost invariably used.

Note that the directional derivative of  $f$  at  $x$  in the direction  $d$  is given by

$$f'(x; d) = \lim_{t \rightarrow 0} \frac{f(x + td) - f(x)}{t}$$

for any  $x, d \in E^n$ . This can be shown to yield

$$f'(x; d) = \sup\{d \cdot \xi : \xi \in \partial f(x)\}.$$

A necessary and sufficient condition that  $x$  minimizes  $f$  is that  $f'(x; d) \geq 0$  for all  $d$ , or equivalently that  $0 \in \partial f(x)$ .

It is certainly possible to use  $\partial f$  to define a direction of steepest ascent for any point which does not solve the problem, and develop a close analogue of the method of steepest ascent for our problem. We take the view that finding the entire set of  $\partial f(x)$  in order to get a locally “best” direction is too heavy a computational requirement for the problems of interest to us, and we suppose that, for any  $x$ , we obtain only a unique element  $\xi$  of  $\partial f(x)$  and use this as noted above.

Another respect in which this algorithm differs from the usual steepest ascent prescription is in making no effort to minimize  $f$  in the chosen direction. First, because that would entail a heavy computational burden; secondly, because the method of choice of direction hardly makes that

advisable. Indeed, the direction might not even be a descent direction. Our choice of step length depends only very modestly on the behavior of the function.

### **2.1.5 Step Length Selection Strategies**

The following result prescribes a step size selection scheme that will guarantee convergence to an optimum.

#### **Theorem 2.1.3**

Let Problem P be as defined in (2.1.9) and assume that an optimum exists. Consider the foregoing subgradient optimization algorithm to solve Problem P, and suppose that the prescribed nonnegative step size sequence  $\{\lambda_k\}$  satisfies the conditions  $\{\lambda_k\} \rightarrow 0^+$  and  $\sum_{k=0}^{\infty} \lambda_k = \infty$ . Then, either the algorithm terminates finitely with an optimal solution, or else an infinite sequence is generated such that

$$\{\text{UB}_k\} \rightarrow f^* \equiv \min \{f(x) : x \in X\}$$

Proof: See Bazaraa, Sherali, and Shetty [2].

Theorem 2.1.3 guarantees that, so long as the step sizes  $\lambda_k$  satisfy the stated conditions, convergence to an optimal solution will be obtained. Although this is theoretically true, it is far from realized in practice. For example, choosing  $\lambda_k = 1/k$  according to the divergent harmonic series ( $\sum_{k=1}^{\infty} 1/k = \infty$ ), the algorithm can easily stall and be remote from optimality after thousands of iterations. A careful fine tuning of the choice of step sizes is usually required to obtain a satisfactory algorithmic performance.

Now, an ideal step size to adopt might be that which brings us closest to  $x^*$ . This step size  $\lambda_k^*$  can be found by requiring that the vector  $(x_k + \lambda_k^* d_k) - x^*$  be orthogonal to  $d_k$ , or that  $d_k' [x_k + \lambda_k^* d_k - x^*] = 0$ . This gives

$$\lambda_k^* = (x^* - x_k)' d_k = \frac{(x_k - x^*)' \xi_k}{\|\xi_k\|}. \quad (2.1.10)$$

The problem with trying to implement the step size  $\lambda_k^*$  is that  $x^*$  is unknown. However, by the convexity of  $f$ , we have  $f^* = f(x^*) \geq f(x_k) + (x^* - x_k)' \xi_k$ ; and, hence, from (2.1.10), we have that  $\lambda_k^* \geq [f(x_k) - f^*] / \|\xi_k\|$ . Since  $f^*$  is also usually unknown, we can recommend using an underestimate  $\bar{f}$  in lieu of  $f^*$ , noting that the foregoing relationship is a “greater than or equal to” type of inequality. This leads to a choice of step size

$$\lambda_k = \frac{\beta_k (f(x_k) - \bar{f})}{\|\xi_k\|} \quad (2.1.11)$$

where  $\beta_k > 0$ . In fact, by selecting  $\varepsilon_1 < \beta_k \leq 2 - \varepsilon_2$  for all  $k$ , for some positive  $\varepsilon_1$  and  $\varepsilon_2$ , and using  $f^*$ , itself, instead of  $\bar{f}$  in (2.1.11), it can be shown (see [2]) that the generated sequence  $\{x_k\}$  converges to an optimum  $x^*$ .

## **2.2 Variable Target Value Method**

Sherali, Choi and Tuncbilek [36] introduced a variable target value method, which assumes no *a priori* knowledge regarding bounds on the optimum value. This algorithm can be used in conjunction with pure or deflected subgradient strategies, and provides a great deal of flexibility by imposing only mild restrictions on the deflection parameter. The algorithm accepts any

subgradient deflection strategy as long as the deflection parameters  $\Psi_k \geq 0$  are chosen such that  $\|d_k\| < M$  for all  $k$ , for some sufficiently large number  $M$ .

## Summary of the Variable Target Value Method (VTVM)

### Notation:

counters:  $l \equiv$  outer loop iteration counter,  $\tau \equiv$  current loop iteration counter,  $k \equiv$  total inner loop iteration counter, and  $\gamma \equiv$  counter of ongoing consecutive nonimprovements.

For any iteration  $k$ :  $x_k =$  iterate,  $f_k = f(x_k)$ ,  $g_k =$  subgradient of at  $x_k$ ,  $d_k =$  direction,  $\lambda_k =$  step-length, and  $z_k =$  incumbent solution value. Also,  $\Delta =$  accumulated improvements within the current set of inner loop iterations.

For any outer loop  $l$ :  $w_l =$  target value, and  $\varepsilon_l =$  acceptance tolerance for declaring that the current incumbent value is close enough to the target value  $w_l$ .

Optimum:  $x^* \in \operatorname{argmin} \{f(x): x \in X\}$ , and  $f^* \equiv f(x^*)$ .

### Fixed Parameters for the Algorithm

$\beta \in (0,1) =$  step-length parameter = 0.95.

$\sigma \in (0, 1/3) =$  acceptance interval parameter = 0.15.

$\bar{\tau} =$  maximum allowable iterations in the inner loop without coming within the acceptance tolerance of the target value = 75.

$\bar{\gamma} =$  maximum consecutive nonimprovements permitted within the inner loop (initialized at 20, and incremented by 10 each time this limit is reached, up to value of 50).

$\eta \in (0,1) =$  fraction of cumulative inner loop improvement that is used to decrease the target value = 0.75.

### Detailed Steps of the Algorithm (VTVM)

**Initialization.** Select termination parameters  $\varepsilon_0 > 0$  for the tolerance on the subgradient norms,  $\varepsilon > 0$  for the overall convergence tolerance and  $k_{max} \leq \infty$  for the limit on the maximum number of iterations. (In our runs, we have used  $\varepsilon_0 = 10^{-6}$ ,  $\varepsilon = 0.1$ , and  $k_{max} = 2,000$ . Other parameters used in the algorithm are as defined above.)

Select a starting solution  $x_1 \in X$ , compute  $f_1 \equiv f(x_1)$  and let  $d_1 = -g_1$ . If  $\|g_1\| < \varepsilon_0$ , then stop with  $x_1$  as near as optimal solution. Otherwise, set  $x^* = x_1$  and  $g^* = g_1$ , and record  $z_1 = f_1$  as the best known objective function value. Initialize the target value  $w_1 = \max\{\text{LB}, f_1 - \|g_1\|^2 / 2\}$  and the acceptance tolerance  $\varepsilon_1 = \sigma(f_1 - w_1)$ , where LB is any known lower bound on  $f^*$ , being taken as  $-\infty$  if no such lower bound is available. (Note that any reasonable value  $< f_1$  would suffice for the second term in the maximand for  $w_1$ . The stated value corresponds to the minimum value if  $f$  is quadratic with an identity Hessian.)

**Step 1 (Inner Loop Main Iteration).** If  $k > k_{max}$ , stop. Else, determine  $d_k = -g_k + \Psi_k d_{k-1}$ , where  $\Psi_k \geq 0$  is selected via any suitable strategy so long as  $\|d_k\| < M$  for all  $k$ , for some sufficiently large number  $M$ , and where  $d_0 \equiv 0$ . If  $\|d_k\| < \varepsilon_0$ , then set  $d_k = -g_k$ . Also, compute the step-length

$$\lambda_k = \beta \frac{f_k - w_l}{\|d_k\|^2}.$$

Find the new iterative  $x_{k+1} = P_X [x_k + \lambda_k d_k]$ , where  $P_X[\cdot]$  denotes the projection operation onto  $X$ , and determine  $f_{k+1}$  and  $g_{k+1}$ . If  $\|g_{k+1}\| < \varepsilon_0$ , terminate the algorithm with  $x_{k+1}$  as a (near) optimal solution. Otherwise, increment  $\tau$  by 1. If  $f_{k+1} < z_k$ , update  $\Delta \leftarrow \Delta + (z_k - f_{k+1})$ , and go to Step 2(a). Otherwise, go to Step 2(b).

**Step 2(a) (Improvement in the inner Loop).** Put  $\gamma = 0$ ,  $z_{k+1} = f_{k+1}$ , and update  $x^* = x_{k+1}$  and  $g^* = g_{k+1}$ . If  $z_{k+1} \leq w_l + \varepsilon_l$ , then go to Step 3(a). Otherwise, if  $\tau \geq \bar{\tau}$ , go to Step 3(b), or else, increment  $k$  by one, and return to Step 1.

**Step 2(b) (Nonimprovement in the Inner Loop).** Put  $z_{k+1} = z_k$ , and increment  $\gamma$  by one. If  $\gamma \geq \bar{\gamma}$  or  $\tau \geq \bar{\tau}$ , go to Step 3(b). Otherwise, increment  $k$  by one, and return to Step 1.

**Step 3(a) (Outer Loop Success Iteration :  $z_{k+1} \leq w_l + \varepsilon_l$ ).** Compute

$$w_{l+1} = z_{k+1} - \varepsilon_l - \eta\Delta \text{ and } \varepsilon_{l+1} = \max\{(z_{k+1} - w_{l+1})\sigma, \varepsilon\}.$$

**Step 3(b) (Outer Loop Failure Iteration:  $z_{k+1} > w_l + \varepsilon_l$ ).** Compute

$$w_{l+1} = \frac{(z_{k+1} + \varepsilon_l) + w_l}{2} \text{ and } \varepsilon_{l+1} = \max\{(z_{k+1} - w_{l+1})\sigma, \varepsilon\}.$$

If  $\gamma \geq \bar{\gamma}$ , adjust  $\bar{\gamma}$  as recommended above. If  $w_{l+1} - w_l \leq 0.1$ , then replace  $\beta$  by  $\max\{\beta/2, 10^{-6}\}$ . Put  $\gamma = 0$ ,  $\tau = 0$ ,  $\Delta = 0$ , increment  $l$  and  $k$  by one, and return to Step 1.

From a practical efficiency point of view, to ensure adequate step-lengths during improving phases of the algorithm, we can replace the target value update in Step 3(a) by  $w_{l+1} = z_{k+1} - \max\{\varepsilon_l + \eta\Delta, r/z_{k+1}\}$ , where  $0 < r < 1$ , and where  $r$  is divided by some  $\bar{r} > 1$  whenever  $w_{l+1}$  is determined by the second term in this maximand. (We used  $r = 0.08$  and  $\bar{r} = 1.08$  in our computations.)

A restarting technique is often an important computational ingredient of subgradient procedures (see [2], [11], and [38]). In the same spirit, for VTVM, whenever the target needs to be increased at Step 3(b) due to  $\bar{\gamma}$  consecutive failures, we restart the algorithm by setting

$x_k = x^*$ ,  $f_k = z_k$ , and  $g_k = g^*$  at the end of Step 3, and then at the next visit to Step 1, we adopt  $d_k = -g_k$ .

Another consideration is concerned with the stopping criterion that is employed. Note that as stated, the algorithm is terminated whenever the iteration count exceeds  $k_{max}$  or when the norm of the current subgradient becomes sufficiently small. Additionally, we can terminate the algorithm based on its progress in improving the incumbent value. For example, we can terminate the algorithm whenever after  $k > 500$  and the algorithm has executed Step 3(a) (outer loop success iteration) at least once, we obtain that the average of the relative improvement  $\Delta/(z_{k+1} - w_l)$  over four consecutive visits of Step 3(b) is less than or equal to 0.05.

## **2.3 Polyak and Kelley's Methods**

The theoretical basis of the Polyak and Kelley Cut methods is given in this section. In Section 2.3.1 Polyak's Algorithm and Kim and Um's version of Polyak's Algorithm are presented. In Section 2.3.2 Kelley's Cut Method is described and in Section 2.3.3 a Polyak's 2-Kelley Cut algorithm is discussed.

### **2.3.1 Polyak's Algorithm**

In [32] Polyak establishes the convergence of two gradient based minimization methods as follows.

Consider the minimization of a functional  $f(x)$  in a set  $Q$  of Hilbert space  $H$ . We shall assume that  $f(x)$  has a support functional at every point (we recall that a linear functional  $c$  is a support of  $f(x)$  at  $x$  if  $f(x+y) \geq f(x) + c'y$  for all  $y$ ). Call this support functional  $f'(x)$ . Also, the set

$Q$  is assumed to be “simple” in the sense that the projection  $P_Q(x)$  onto it can be found easily. We minimize  $f(x)$  by an iterative method in which starting with some  $x_0$ , we construct a sequence of  $\{x_n\}$  in accordance with the relation

$$x_{n+1} = P_Q(x_n - \alpha_n f'(x_n)) \quad (2.3.1)$$

for some suitable step size  $\alpha_n$ .

If  $Q = H$  (i.e. the problem has no constraints) while  $f(x)$  is differentiable (so that  $f'(x)$  is unique and is the same as the gradient of  $f(x)$ ), (2.3.1) amounts to an ordinary gradient method and its convergence has been proved in several papers (see [2]). The following step length selection method, allows fairly fast convergence if we know the minimum value  $f^*$  of the functional.

$$\alpha_n = \lambda_n \frac{f(x_n) - f^*}{\|f'(x_n)\|^2}, \quad 0 < 1 \leq \lambda_n \leq 2 - \varepsilon_2, \quad \varepsilon_2 > 0. \quad (2.3.2)$$

In [32] Polyak provides the necessary theorems (Theorems 2.3.1 to 2.3.6) and discussion.

### **Theorem 2.3.1**

Let  $f(x)$  be convex and continuous, let  $Q$  be convex and closed, and let the minimum point  $x^* \in Q$ ,  $f(x^*) = f^*$  exist and  $\|f'(x)\| \leq c$  on  $S = \{x \in Q: \|x - x_0\| \leq \|x^* - x_0\|\}$ . Then,  $f(x_n) \rightarrow f^*$  in the method (2.3.1), (2.3.2), while  $x_n$  is weakly convergent to some minimum point.

Proof: See Polyak [32].

### Theorem 2.3.2

Let  $Q = H$  and  $f(x)$  be a strongly convex, with constant  $m$ , differentiable functional, while  $f'(x)$  satisfies a Lipschitz condition with constant  $M$  in  $S = \{x: \|x - x_0\| \leq \|x^* - x_0\|\}$ . Then,

$\|x_n - x^*\| \leq q^n \|x_0 - x^*\|$ , and

$$q = \left(1 - \varepsilon_1 \varepsilon_2 \frac{m^2}{M^2}\right)^{1/2} < 1.$$

Proof: See Polyak [32].

Hence, it follows from Theorem 2.3.2 that the method (2.3.1), (2.3.2) is convergent for the smooth case at the rate of a geometric progression (when  $\lambda_n \equiv 1$ ),  $q = (1 - m^2/M^2)^{1/2}$ , i.e. at roughly the same rate as an ordinary gradient method.

Now consider the unsmooth case. We impose an extra condition on  $f(x)$ :  $f(x) - f^* \geq m\rho(x, S)$ ,  $m > 0$ , for all  $x$  such that

$$\|x - P_S(x_0)\| \leq \rho(x_0, S), \tag{2.3.3}$$

where  $S = \{x \in Q, f(x) = f^*\}$ . Here and below,  $\rho(x, S)$  denotes the distance from  $x$  to the set  $s$ , i.e.,

$$\rho(x, S) = \|x - P_S(x)\| = \inf_{y \in S} \|x - y\|.$$

### Theorem 2.3.3

Let  $f(x)$  satisfy (2.3.3), in addition to the conditions of Theorem 2.3.1. Then, the sequence  $\{x_n\}$  in the method (2.3.1), (2.3.2) is convergent to a minimum point  $x^*$  at the rate of a geometric progression:

$$\|x_n - x^*\| \leq Kq^n, \quad q = \left(1 - \varepsilon_1 \varepsilon_2 \frac{m^2}{M^2}\right)^{1/2} < 1.$$

Proof: See Polyak [32].

Now take a more general definition of the step length than (2.3.2):

$$\alpha_n = \lambda_n \frac{f(x_n) - \bar{f}}{\|f'(x_n)\|^2}, \quad 0 < 1 \leq \lambda_n \leq 2 - \varepsilon_2, \quad \varepsilon_2 > 0 \quad (2.3.4)$$

where  $\bar{f}$  is not necessarily the same as  $f^*$ . This method is of interest in the case when we seek the minimum of  $f(x)$  without knowing  $f^*$ .

#### **Theorem 2.3.4**

Let the conditions of Theorem 2.3.1 be satisfied and let  $\bar{f} > f^*$ . Then one of the following two cases is possible in the method (2.3.1), (2.3.4): a)  $f(x_n) < \bar{f}$  for some  $n$ , or b)  $f(x_n) \geq \bar{f}$  for all  $n$ ,  $f(x_n) \rightarrow \bar{f}$ , at the rate of convergence of a geometric progression. If  $Q = H$ ,  $\lambda_n \leq 1$ ,  $f(x_0) \geq \bar{f}$ , only case b) is possible. If  $Q = H$ ,  $\lambda_n \equiv 1$ ,  $f(x_0) \geq \bar{f}$ , and  $f(x)$  is smooth ( $f'(x)$  satisfies a Lipschitz condition), then  $x_n$  is more rapidly convergent to  $\bar{x}$ :  $\|x_n - \bar{x}\| \leq Kq^{(3/2)^n}$ ,  $q < 1$ . Finally if  $\bar{f} < f^*$ , there exists in the method (2.3.1), (2.3.4), with  $\lambda_n \equiv 1$  and  $\varepsilon > 0$  arbitrary, an  $n$  such that  $f(x_n) \leq 2f^* - \bar{f} + \varepsilon$ .

Proof: See Polyak [32].

We now describe a method that is at least as fast as (2.3.1), (2.3.2), and gives the exact minimum for a piecewise linear functional.

Again, consider minimization of  $f(x)$  on  $Q$  and assume that  $f^* = \inf_{x \in Q} f(x)$  is known. We

construct as follows a sequence of points  $x_n$  and a sequence of sets  $Q_n$ :

$$Q_n = \{x: f(x_k) + (x-x_k)'f'(x_k) \leq f^*, k \in I_n\}, x_{n+1} = P_Q P_{Q_n}(x_n). \quad (2.3.5)$$

### Theorem 2.3.5

Whatever the choice of  $I_n$ , method (2.3.5) is convergent, and we obtain the same bound as for method (2.3.1), (2.3.2) with  $\lambda_n \equiv 1$ , via

$$\|x_{n+1} - x^*\| \leq \|x_n - x^*\|^2 - \frac{(f(x_n) - f^*)^2}{\|f'(x_n)\|^2}.$$

If  $Q = H$ , and  $f(x)$  is piecewise linear in a neighborhood close to its minimum, i.e. representable as  $f(x) = f^* + \max_{1 \leq i \leq m} (x - x^*)' c_i$ , method (2.3.5) with  $I_n = \{n - m + 1, n - m + 2, \dots, n\}$  is convergent after a finite number of steps.

Proof: See Polyak [32].

Now consider a set of convex inequalities  $f_i(x) \leq 0, i = 1, \dots, m$ , in finite dimensional space  $E_n$ . As before we introduce  $f(x) = \max_{1 \leq i \leq m} f_i(x)$  and adopt the method (2.3.1), (2.3.2), which here runs as

$$x_{n+1} = x_n - \lambda_n \frac{f_i(x_n)}{\|f'_{i(n)}(x_n)\|^2} f'_{i(n)}(x_n), \quad f_{i(n)}(x_n) = \max_{1 \leq i \leq m} f_i(x_n). \quad (2.3.6)$$

### Theorem 2.3.6

Let  $x^*$  be the unique solution of the set of inequalities, for which the  $(n+1)$ th inequality becomes an equality (for clarity, let  $f_1(x^*) = \dots = f_{n+1}(x^*) = 0, f_{n+2}(x^*) < 0, \dots, f_m(x^*) < 0$ ). If any  $n$  vectors  $f'_1(x^*), \dots, f'_{n+1}(x^*)$  are linearly independent, the sequence  $\{x_n\}$  in method (2.3.6) converges to  $x^*$  at the rate of a geometric progression.

Proof: See Polyak [32].

We can now describe how method (2.3.1), (2.3.2) can be applied when  $f^*$  is unknown. We take an initial approximation for  $f^*$ , say  $\bar{f}$ , and perform the computations in accordance with (2.3.1) and (2.3.4). If we get  $f(x_n) < \bar{f}$ , or if  $f(x_n) \rightarrow \bar{f}$ , it follows from Theorem 2.3.4 that  $\bar{f} > f^*$  and we have to reduce  $\bar{f}$ . If  $f(x_n) \geq \bar{f} + \varepsilon$ ,  $\varepsilon > 0$ , for all  $n$ , then  $\bar{f}$  must be increased. In the latter case, we can see from Theorem 2.3.4 that  $\frac{1}{2}(\bar{f} + \inf_n f(x_n))$  should be taken as the new value of  $\bar{f}$ .

Kim and Um [17] modified Polyak's method [32] to improve its computational efficiency. They did this by combining two consecutive projection operations in a single projection operation. The resulting algorithm has a convergence property which is strictly stronger than the original.

To understand this method, consider the problem of minimizing a convex, not necessarily differentiable, function  $f: E_n \in E_1$  on a closed convex set. The problem can be described as follows:

$$f^* = \text{minimize } f(x)$$

subject to  $x \in S$ ,

where  $S$  is a closed convex set. We assume that this problem has a finite optimal solution.

For a convex function  $f$ ,  $g$  is a subgradient of  $f$  at  $x$  if

$$f(x') \geq f(x) + (x' - x)'g \quad \text{for all } x' \in E_n.$$

The subdifferential  $\partial f(x)$  is the set of subgradients of  $f$  at  $x$ . Since  $f$  is finite and convex,  $\partial f(x)$  is nonempty for all  $x$ .

Let  $X \neq \emptyset$  be a closed convex subset of  $E_n$ . For each  $x \in E_n$ , define the projection of  $x$  on  $X$ , denoted by  $P_X(x)$ , to be the unique point in  $X$  such that for all  $y \in X$ ,

$$\|P_X(x) - x\| \leq \|y - x\|.$$

Let  $\rho$  be a given parameter which is called the target value, an estimation of  $f^*$ . For given  $x_k \in S$ ,  $g_k \in \partial f(x_k)$  and  $\lambda_k > 0$ , let

$$H_k^{\lambda_k} = \{x: f(x_k) + (1/\lambda_k)(x - x_k)'g_k \leq \rho\}.$$

For  $0 < \lambda_k \leq 1$ ,  $H_k^{\lambda_k}$  is a cut which does not eliminate any feasible solution with objective value less than or equal to  $\rho$ . Then the subgradient method suggested by Polyak [32] can be described as follows:

$$x_{k+1} = P_S(P_{H_k^{\lambda_k}}(x_k)), \quad 0 < \varepsilon_1 \leq \lambda_k \leq 2 - \varepsilon_2 < 2.$$

When  $\rho \geq f^*$ , Polyak [32] established the following convergence property if  $f(x_k) > \rho$ .

$$\|x^* - x_{k+1}\|^2 \leq \|x^* - x_k\|^2 - \lambda_k(2 - \lambda_k) \frac{(f(x_k) - \rho)^2}{\|g_k\|^2},$$

where  $x^*$  is a point in  $S$  with  $f(x^*) \leq \rho$ . This relation guarantees that this algorithm finds  $x_k$  with  $f(x_k) \leq \rho$  for some finite  $k$  or  $\{x_k\}$  converges to a point  $x^* \in S$  with  $f(x^*) \leq \rho$ . When  $\rho < f^*$ , the convergence of this algorithm is not guaranteed, but it is possible to obtain a good sub-optimal solution. A target value updating scheme in [37] is developed for strongly convex functions (i.e., for which  $f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2) - \varepsilon \forall x_1 \neq x_2, 0 < \lambda < 1$ , and some  $\varepsilon > 0$ ), such that the algorithm finds the optimal value without any prior knowledge of  $f^*$ .

### **Summary of Polyak's Algorithm**

#### **Initial Step**

Choose any  $x_1 \in S$  and  $\varepsilon_1 \in (0,1]$ . Let  $k = 1$ .

#### **Main Step**

If  $f(x_k) \leq \rho$  then terminate the algorithm with  $x_k$ .

Otherwise choose  $g_k \in \partial f(x_k)$ .

Let  $x_{k+1} = P_S(x_k - \lambda_k \partial f(x_k))$ ,  $0 < \varepsilon_1 \leq \lambda_k \leq 1$ .

Let  $k = k+1$  and repeat the Main Step.

### **2.3.2 Kelley's Cut Method**

Let  $G(x)$  be a continuous convex function defined on the  $n$ -dimensional compact polyhedral convex set  $S = \{x: Ax \geq b\}$ , where  $A$  is an  $m \times n$  matrix,  $x$  is an  $n \times 1$  column vector of variables and  $b$  is a  $m \times 1$  column vector. We assume at every point  $t$  in  $S$  that there exists an extreme support,  $y = p(x, t)$ , to the graph of  $G(x)$  with the property that, for some finite constant  $K$ ,

$\|\nabla p(x, t)\| \leq K$  for all  $x \in S$ . We will assume that the gradient vector is a row vector. Let  $cx$  be a linear form, where  $c$  is an  $n \times 1$  row vector and  $\|c\| < \infty$ . Let  $R = \{x: G(x) \leq 0\}$  be nonempty and  $R \subset S$ .

We may state the convex programming problem formally as follows:

*Find a vector  $\tau$  such that  $f = c\tau = \min \{cx: x \in R\}$ .*

Since  $R$  is compact, a minimum exists and is finite. Kelley suggested the following algorithm for solving such a problem.

### **Theorem 2.3.7**

Let  $G(x)$  be a continuous convex function defined on the  $n$ -dimensional compact convex set  $S$  such that at every point  $t \in S$  there exists an extreme support  $y = p(x, t)$ , to the graph of  $G(x)$  with the property that, for some finite constant  $K$ , point  $\|\nabla p(x; t)\| \leq K$  for all  $x \in S$ . Further, let  $cx$  be a linear form such that  $\|c\| < \infty$  and let  $R = \{x: G(x) \leq 0\} \subset S$ , with  $R$  nonempty. If  $t_k \in S_k$  is such that

$$ct_k = \min \{cx: x \in S_k\} \quad (k = 0, 1, \dots)$$

where  $S_0 = S$  and

$$S_k = S_{k-1} \cap \{x: p(x; t_{k-1}) \leq 0\},$$

then the sequence  $\{t_k\}$  contains a subsequence that converges to a point  $\tau$ , in  $R$  with  $c\tau \leq cx$

for all  $x \in R$ .

Proof: see Kelley [13].

### 2.3.3 Polyak's 2-Kelley Cut Algorithm

Recall from (2.3.5) that,  $I_n$  is a subset of indices taken from  $\{0, 1, \dots, n\}$ , containing  $n$ , and that  $P_Q, P_{Q_n}$  are projection operators onto  $Q$  and  $Q_n$  respectively. Notice that the method (2.3.1), (2.3.2), with  $\lambda_n \equiv 1$ , is a particular case of (2.3.5) in which  $I_n = \{n\}$ , i.e., we take account only of this last restriction. This follows since, the point  $x_n - \frac{f(x_n) - f^*}{\|f'(x_n)\|^2} f'(x_n)$  is simply the projection of  $x_n$  on to the subspace  $\{x: f(x_n) + f'(x_n)(x - x_n) \leq f^*\}$ . This method has a great deal in common with [13], though it is not identical. The method (2.3.5) is convergent at the rate of a geometric progression [32], whereas this is unproven for [13]. A variant of (2.3.5) which is similar to using two Kelley cuts with known  $f^*$ , and at the same time is more rapidly convergent than method (2.3.1), (2.3.2) is obtained if we take  $I_n = \{n-1, n\}$ . In this case the projection on the set  $Q_n$  can be written explicitly

$$x_{n+1} = P_Q(x_n - \alpha_n f'(x_n) - \beta_n f'(x_{n-1})),$$

$$\text{if } (x_{n-1}) + (f'(x_{n-1}))^t x_n - \frac{f(x_n) - f^*}{\|f'(x_n)\|^2} f'(x_n) - (x_{n-1}) \leq f^*,$$

$$\alpha_n = \frac{f(x_n) - f^*}{\|f'(x_n)\|^2} \quad \text{and } \beta_n = 0,$$

otherwise,

$$\alpha_n = \frac{(f(x_n) - f^*)\|f'(x_{n-1})\|^2 - (f'(x_n))^t f'(x_{n-1}))(f(x_{n-1}) - f^*) + f'(x_{n-1})^t (x_n - x_{n-1})}{\|f'(x_n)\|^2 \|f'(x_{n-1})\|^2 - (f'(x_n))^t f'(x_{n-1}))^2}$$

$$\beta_n = \frac{\|f'(x_n)\|^2 (f(x_{n-1}) - f^* + (f'(x_{n-1}))^t (x_n - x_{n-1})) - (f(x_n) - f^*)(f'(x_n))^t f'(x_{n-1}))}{\|f'(x_n)\|^2 \|f'(x_{n-1})\|^2 - (f'(x_n))^t f'(x_{n-1}))^2}.$$

## 2.4 Shor's $r$ -Algorithms

Shor [40,41] introduced the space dilation operator which can be represented as

$$R_\alpha(r) = I + (\alpha - 1)rr^t$$

for some parameter  $0 < \alpha < 1$  and some dilation vector  $r$  that satisfies  $r^t r = 1$ . Note that under the transformation  $x = R_\alpha(r)y$ , we get

$$y = R_\alpha(r)^{-1}x = \left[ I + \left( \frac{1}{\alpha} - 1 \right) rr^t \right] x,$$

so that the vector  $y$  is obtained by stretching the vector  $x$  along the direction  $r$ , depending on the value of  $\alpha$ . The original algorithm uses the (sub)gradient as a dilation vector. Subsequently, in [43] and [44], a dilation scheme along the direction of the difference between two consecutive subgradients has been proposed, and this is widely recognized as being among the most efficient methods for nondifferentiable optimization problems. In this procedure, at each iteration  $k$ , given a subgradient  $g_k$ , the previous subgradient  $g_{k-1}$ , and the previous transformation operator  $B_{k-1}$ , a new transformation operator  $B_k$  is obtained via the following update scheme:

$$B_k = B_{k-1}R_{\alpha_k}(r'_k) \quad \text{where } r'_k = \frac{B_{k-1}^t(g_k - g_{k-1})}{\|B_{k-1}^t(g_k - g_{k-1})\|}.$$

Using the space transformation  $x = B_k y$ , so that  $B_k^t g_k$  is a subgradient of  $F(y) \equiv f(B_k y)$  under this transformation, a step-length of  $\lambda_k$  along the anti-subgradient in the transformed space yields

$y_{k+1} = y_k - \lambda_k B_k^t g_k$ . In the original  $x$  space, this yields the new iterate

$$x_{k+1} = x_k - \lambda_k B_k B_k^t g_k.$$

Skokov [45] showed that one can reduce the number of arithmetic operations in the computations by using the symmetric matrix  $H_k = B_k B_k^t$ , and directly updating this equivalently via the following rank-one update scheme, where  $q_k \equiv g_k - g_{k-1}$ , and  $p_k = H_{k-1} q_k$ .

$$H_k = H_{k-1} - (1 - \alpha_k^2) \frac{p_k p_k^t}{p_k^t q_k}. \quad (2.4.1)$$

Accordingly, we would then have,

$$x_{k+1} = x_k - \lambda_k H_k g_k.$$

This *r-algorithm*, as it is called, was shown to be quite promising for various practical problems (see [43 and 44]). However, for large sized problems, the matrix updating process becomes very expensive. By imitating the memoryless quasi-Newton algorithm for the differentiable case, we now proceed to alleviate this computational difficulty associated with the *r-algorithm*. (See [2, 27] for a discussion on memoryless quasi-Newton algorithms.)

## **Chapter 3**

### **Proposed Modified Algorithms**

We are now in a position to present more advanced variants of the algorithms discussed in Chapter 2. We present a memoryless variant of Shor and Zhurbenko's  $r$ -algorithm, motivated by the memoryless and limited memory updates for differentiable quasi-Newton methods. Using this concept, we build a new update scheme that, in the space transformation sense, can be viewed as a combination of space dilation and reduction operations. We prove convergence of this new method, and demonstrate how it can be used in conjunction with a variable target value method that allows a practical, convergent implementation of the method.

In this chapter, we also present a modification of a related algorithm due to Polyak that employs a projection on a pair of modified Kelley's cutting planes.

#### **3.1 Memoryless Shor's $r$ -Algorithm**

From the discussion in Section 2.4, we see a need to develop variants of Shor's  $r$ -Algorithm that yield comparable, if not better, performance with lesser memory requirements. Toward this end, assuming that  $\|g_k - g_{k-1}\| \neq 0$ , let us define

$$r_k = \frac{q_k}{\|q_k\|} \equiv \frac{g_k - g_{k-1}}{\|g_k - g_{k-1}\|} \quad (3.1.1)$$

to be the normalized difference vector of two successive subgradients. For each  $k$ , by replacing  $H_{k-1}$  (or the previous transformation matrix  $B_{k-1}$ ) with the identity matrix in (2.4.1), we have

$$H_k = I - (1 - \alpha_k^2) \frac{q_k q_k^t}{q_k^t q_k} = I - (1 - \alpha_k^2) r_k r_k^t. \quad (3.1.2)$$

Accordingly, the direction of motion is computed as

$$d_k = H_k(-g_k) = -g_k + (1 - \alpha_k^2)(g_k^t r_k) r_k \quad (3.1.3a)$$

and the new iterate is given by

$$x_{k+1} = x_k + \lambda_k d_k \quad (3.1.3b)$$

for some step-length  $\lambda_k$ . To gain some insight into the resulting prescribed direction, let us define

$$s_k = (1 - \alpha_k^2) \frac{g_k^t r_k}{\|g_k - g_{k-1}\|}. \quad (3.1.4)$$

Then, from (3.1.1), (3.1.3a), and (3.1.4), the direction of motion can be written as

$$d_k = -g_k + s_k (g_k - g_{k-1}) = -[(1 - s_k)g_k + s_k g_{k-1}] \quad (3.1.5)$$

which is an affine combination of the present and previous anti-subgradients. Note that in Shor's original algorithm,  $\alpha_k$  is chosen to be fixed and to lie in  $(0, 1)$  so that it dilates the space as mentioned above. But here, we relax this requirement, permitting  $\alpha_k$  to vary as well as exceed 1, if necessary, so that at each iteration  $k$ , the transformation can either dilate or shrink the space in varying degrees along the direction  $r_k$ . However, we would like to maintain  $0 \leq s_k < 1$  so that the direction (3.1.5) is similar to that obtained via a conjugate subgradient deflection strategy in which the previous direction is substituted by the previous anti-subgradient. As an aside, note in this connection that if  $\alpha_k = 1$  for any iteration  $k$ , then no space transformation is performed and we obtain  $s_k = 0$ , so that this corresponds to simply adopting the pure anti-subgradient direction

at this iteration. The following is a step-by-step description of the proposed memoryless  $r$ -algorithm.

### Summary of Memoryless Variant of Shor's $r$ -Algorithm

#### Initialization Step

The first step is specified by the initial value of  $x_0 \in E_n$ ; we calculate  $g_f(x_0)$ , choose

$\lambda_1 > 0$ , find

$$x_1 = x_0 - \lambda_1 g_f(x_0),$$

and proceed to the Main Step.

#### Main Step

Let,

$$q_k \equiv g_k - g_{k-1}$$

$$r_k = \frac{q_k}{\|q_k\|}$$

$$H_k = I - (1 - \alpha_k^2) \frac{q_k q_k^t}{q_k^t q_k}$$

$$d_k = H_k(-g_k)$$

$$\text{and } x_{k+1} = x_k + \lambda_k d_k$$

for some step-length  $\lambda_k$ .

If a specified termination criterion is not met, repeat the Main Step.

This is a description of a class of algorithms for which particular variants can be designed by choosing the sequences  $\{\lambda_{k+1}\}$  and  $\{\alpha_k\}$  and specifying a termination criterion. We shall call

such algorithms as memoryless  $r$ -algorithms. There is a further possibility for modifying  $r$ -algorithms - the use of the so-called restoration operator, for which periodically, after a given number of iterations, we restore the matrices  $B_k$ , i.e.,  $\{B_k\}$  is replaced by a single matrix. The problem of the convergence of the algorithms when the values of the function being minimized vary monotonically, and when there is restoration, is solved comparatively simply, since in fact, it reduces to the problem of the usual gradient descent method without a change of metric. This has been studied in detail by several authors [3]. In particular, if  $f(x)$  is continuously differentiable and  $\alpha_k$  in (3.1.3) is chosen to yield the minimum along the generated direction, when we use  $r$ -algorithms with restoration, the set of limit points of the sequence  $\{x_k\}$  consist of stationary points of  $f(x)$ . The proof of this is virtually the same as the proof of the analogous result for the method of the steepest descent.

### **3.2 Modified Polyak-Kelley Algorithm**

We now propose a version of Polyak's Algorithm, to be used within the VTVM framework, which uses the intersection of two Kelley cuts for finding the next iterate. By the convexity of  $f$ , we have

$$f(x) \geq f(x_k) + (x - x_k)' g_k.$$

Imposing  $f(x) \leq z^*$ , the optimal value, and denoting  $f_k \equiv f(x_k) \forall k$ , we obtain the Kelley cut

$$(x - x_k)' g_k \leq (z^* - f_k). \tag{3.2.1}$$

Note that  $x_k$  violates this since  $z^* < f_k$ . The projection of  $x_k$  onto (3.2.1) is given by

$$x_{k+1} = x_k - \left[ \frac{f_k - z^*}{\|g_k\|^2} \right] g_k \quad (3.2.2)$$

which is a negative subgradient step of the type we take with  $\beta \equiv 1$  in (2.1.11), assuming that  $z^*$  is known. Note that at the previous iteration, except when  $\text{RESET} = 1$ , i.e., when the inner VTVM loop restarts, we would have had a cut similar to (3.2.1) given by

$$(x - x_{k-1})^t g_{k-1} \leq (z^* - f_{k-1}). \quad (3.2.3)$$

Therefore, via (3.2.9) or via what follows, we would have  $x_k$  satisfying (3.2.3) so that

$$(x_k - x_{k-1})^t g_{k-1} + f_{k-1} - z^* \leq 0. \quad (3.2.4)$$

Now,  $z^*$  is not typically known, and we have been using a target value to estimate it. Noting (3.2.2) and the usual choice of a step length given by

$$\lambda_k = \frac{\beta(f_k - \omega_l)}{\|g_k\|^2} \quad (3.2.5)$$

when  $d_k = -g_k$ , we can in effect equate (as an estimate) by virtue of (2.3.4)

$$\beta(f_k - \omega_l) \cong (f_k - z^*). \quad (3.2.6)$$

Using (3.2.6) in (3.2.1) and (3.2.3), we get the pair of Kelley cuts

$$(x - x_k)^t g_k \leq \beta(\omega_l - f_k) \quad (3.2.7)$$

$$(x - x_{k-1})^t g_{k-1} \leq (f_k - f_{k-1}) + \beta(\omega_l - f_k). \quad (3.2.8)$$

Note that unlike (3.2.4) we may not have  $x_k$  satisfying (3.2.8) as assumed by Polyak [32].

Hence, we determine  $x_{k+1}$  as the solution to the following problem that seeks the projection of  $x_k$  onto the intersection of the pair of half spaces (3.2.7) and (3.2.8).

$$\text{Minimize } \frac{1}{2} \|x - x_k\|^2$$

subject to (3.2.7) and (3.2.8).

For ease in notation let us define the following:

$$\bar{x} = x_k, g_0 = g_{k-1}, g_1 = g_k, \theta_0 = x_{k-1}^t g_{k-1} + (f_k - f_{k-1}) + \beta(\omega_l - f_k), \text{ and}$$

$$\theta_1 = x_k^t g_k + \beta(\omega_l - f_k)$$

Hence, the problem becomes

$$\text{Minimize } \frac{1}{2} \|x - x_k\|^2 \tag{3.2.9}$$

$$\text{subject to } g_1^t x \leq \theta_1 \tag{3.2.10}$$

$$g_0^t x \leq \theta_0. \tag{3.2.11}$$

Denoting  $\alpha$  and  $\delta$  as the Lagrangian multipliers associated with (3.2.10) and (3.2.11), respectively, the KKT conditions are given as follows, in addition to (3.2.10) and (3.2.11).

$$x = x_k - \alpha g_1 - \delta g_0 \tag{3.2.12}$$

$$(g_1^t x - \theta_1) \alpha = 0, (g_0^t x - \theta_0) \delta = 0, (\alpha, \delta) \geq 0.$$

Note that  $\alpha = \delta = 0$  does not yield a KKT solution since  $x_k$  is known to violate (3.2.10). Moreover, noting (3.2.13), we would like  $\alpha > 0$  so that the direction  $d_k$  is of the type that effectively deflects  $-g_k$ . Hence, we first test if  $\alpha > 0, \delta = 0$  yields a KKT point. If not, we test if  $\alpha > 0, \delta > 0$  yields a KKT solution, and if the latter does not, then we discard (3.2.12) and simply select the former solution.

### Summary of the Modified Polyak-Kelley Algorithm

This uses the procedure VTVM with parameters chosen as described therein, except that we also store  $g_{k-1}, f_{k-1}$  and  $x_{k-1}^t g_{k-1}$  (when RESET = 0).

**Step 1** Compute  $\alpha = \frac{\beta(f_k - \omega_l)}{\|g_k\|^2}$ , and let  $\bar{x}_{k+1} = x_k - \alpha g_k$ .

If  $\text{RESET} = 1$  or  $g_{k-1}^t \bar{x}_{k+1} \leq \theta_0$  (i.e. 2.3.18 is satisfied), go to Step 3.

Else, go to Step 2.

**Step 2** Compute  $\phi_0 = \|g_0\|^2 \|g_1\|^2 - (g_0^t g_1)^2$ .

If  $\phi_0 \leq 10^{-6}$ , go to Step 3.

Else, compute  $\phi_1 = \beta(f_k - w_l)$ ,  $\phi_2 = (x_k - \theta_0)$ , and

$$\alpha = \frac{\phi_1 \|g_0\|^2 - \phi_2 (g_0^t g_1)}{\phi_0}, \quad \delta = \frac{\phi_2 - \alpha (g_0^t g_1)}{\|g_0\|^2}.$$

If  $\alpha \leq 0$  or  $\delta \leq 0$ , go to Step 3.

Else, compute  $\bar{x}_{k+1} = x_k - \alpha g_k - \delta g_{k-1}$

and proceed to Step 3.

**Step 3** Compute  $x_{k+1} = P_X[\bar{x}_{k+1}]$  and repeat these steps.

### **3.3 Memoryless Space Dilation and Reduction Algorithm and its**

#### **Convergence**

Before we present the proposed algorithm and examine its convergence, let us consider the issue of selecting a step-length rule. Note that for generalized subgradient methods, we can write

$$x_{k+1} = x_k - \sum_{j \in I_k} t_{kj} g_j, \quad (3.3.1)$$

where  $J_k \subseteq \{1, 2, \dots, k\}$  and  $t_{kj} \geq 0$  for all  $j$  and  $k$ ; that is, the direction of motion is some nonnegative linear combination of the previously obtained anti-subgradients. The convergence of this procedure has been proven by Kim and Ahn [14] using the  $\varepsilon$ -subgradient concept of Nurminski and Zhelikhovski [31] and Demyanov and Vasilev [5], when the step-length is given by the usual rule satisfying  $\lambda_k \geq 0$ ,  $\lim_{k \rightarrow \infty} \lambda_k = 0$ , and  $\sum_{k=1}^{\infty} \lambda_k = \infty$ . Since the negative direction of motion  $\sum_{j \in J_k} t_{kj} g_j$  is an  $\varepsilon_k$ -subgradient of  $f$  at  $x_k$  for some  $\varepsilon_k$  for each  $k$  (see [14] for the derivation of  $\varepsilon_k$ ), Kim and Ahn require  $\lim_{k \rightarrow \infty} \varepsilon_k = 0$  for their convergence theorem to hold. Using this same divergent series step-length rule, we can establish the convergence of the procedure (3.1.3) under a suitable assumption. To see this, let us define

$$v_k = f_k - f_{k-1} - g_{k-1}^t(x_k - x_{k-1}) \geq 0 \quad (3.3.2)$$

where  $f_t \equiv f(x_t)$  for all  $t$ , and consider the following lemma.

**Lemma 3.3.1.** Suppose that the direction of motion  $d_k$  is given by (3.1.3a) or (3.1.5). If  $0 \leq s_k < 1$  in (3.1.4) for all  $k$ , then

$$f(x) - f_k \geq -d_k^t(x - x_k) - s_k v_k$$

for all  $x \in R^n$ , that is,  $-d_k$  is an  $(s_k v_k)$ -subgradient of  $f$  at  $x_k$ .

Proof: Using the fact that  $g_k$  is a subgradient of  $f$  at  $x_k$ , and that  $g_{k-1}$  is a  $v_k$ -subgradient of  $f$  at  $x_k$ , we have for all  $x \in E_n$ ,

$$\begin{aligned}
-d_k^t(x - x_k) - s_k v_k &= (1 - s_k)g_k^t(x - x_k) + s_k g_{k-1}^t(x - x_k) - s_k v_k \\
&\leq (1 - s_k) [f(x) - f_k] + s_k [f(x) - f_k + v_k] - s_k v_k = f(x) - f_k.
\end{aligned}$$

This completes the proof.

Thus, according to the results in [14], if  $\lim_{k \rightarrow \infty} s_k v_k = 0$ , then the procedure (3.1.3), used along with a divergent series step-length rule, will generate a sequence of solutions that converges to an optimal solution. Note that the assumptions  $0 \leq s_k < 1$  and  $\lim_{k \rightarrow \infty} s_k v_k = 0$  can be established by choosing  $\{\alpha_k\} \rightarrow 1$  suitably, so that  $0 \leq s_k < 1$  for all  $k$  and  $\lim_{k \rightarrow \infty} s_k = 0$ , provided that  $\{v_k\}$  is bounded. However, this strategy loses the merit of using a variable metric method, since  $\{\alpha_k\} \rightarrow 1$  eventually leads the algorithm to a pure subgradient method. Likewise, for generalized subgradient methods characterized by (3.3.1), since  $t_{kj}$  implicitly involves both the affine combination weights associated with the subgradients and the step-lengths, we cannot control the affine combination weights independently, and thus, we might lose the advantage of freely selecting a desired direction. Considering these weaknesses, that are further aggravated by the poor computational behavior of a divergent series step-length rule, we devise a new algorithm that employs (3.1.3) along with the more popular step-length rule

$$\lambda_k = \beta_k \frac{f_k - w}{\|d_k\|^2} \tag{3.3.3}$$

where  $w$  is a suitable target value and  $\beta_k > 0$  is an appropriately controlled step-size parameter. Note that among the various step-length rules for subgradient algorithms (see [1] and [10]), this rule is known to be quite promising in practice.

From now on, we consider a constrained problem of minimizing  $f(x)$  over some convex subset  $X$  of  $E_n$ , so that the proposed algorithm can be used for solving a wider class of problems.

Hence, a new iterate is computed by

$$x_{k+1} = P_X(x_k + \lambda_k d_k) \quad (3.3.4)$$

instead of (3.1.3b), where  $P_X(\cdot)$  denotes a projection operation onto the set  $X$ . The following result establishes the principal convergence property of the proposed algorithm.

**Theorem 3.3.1.** Suppose that  $w \geq f^* \equiv f(x^*)$ , where  $x^*$  is an optimal solution to the problem of minimizing  $f(x)$  over  $x \in X$ , and that the procedure (3.1.3a) and (3.3.4) using the step-length rule (3.3.3) is run while  $f_k$  remains greater than the target value  $w$ . Moreover, suppose that  $\alpha_k$  and  $\beta_k$  used in (3.1.4) and (3.3.3), respectively, are chosen such that

$$0 \leq s_k < 1 \quad \text{and} \quad 0 < \varepsilon_1 \leq \beta_k \leq \varepsilon_{2k} \equiv 1 - \frac{s_k v_k}{f_k - w} \leq 1 \quad (3.3.5)$$

holds for all  $k$  for some  $\varepsilon_1 > 0$ , where  $v_k$  is given by (3.3.2). Assume that  $\|g_k\| < M$  for all  $k$ , for some  $M > 0$ . Then, either (i) for some  $k$  we will have  $f_{k+1} \leq w$ , or else, we will have (ii)  $f_k > w$  for all  $k$  and  $\lim_{k \rightarrow \infty} f_k = w$ .

Proof: Clearly, either  $f_{k+1} \leq w$  for some  $k$ , or else  $f_k > w$  for all  $k$ . Assuming the latter, we have

$$\begin{aligned} \|x_{k+1} - x^*\|^2 &= \|P_X(x_k - \lambda_k d_k) - x^*\|^2 \leq \|x_k + \lambda_k d_k - x^*\|^2 \\ &= \|x_k - x^*\|^2 + \lambda_k^2 \|d_k\|^2 + 2\lambda_k d_k'(x_k - x^*) = \|x_k - x^*\|^2 + \lambda_k^2 \|d_k\|^2 - 2\lambda_k R_k \end{aligned}$$

$$= \|x_k - x^*\|^2 + \lambda_k^2 d_k^2 + 2\lambda_k d_k^t (x_k - x^*) = \|x_k - x^*\|^2 + \lambda_k^2 d_k^2 - 2\lambda_k R_k \quad (3.3.6)$$

where

$$R_k = (1 - s_k)g_k^t(x_k - x^*) + s_k g_{k-1}^t(x_k - x^*).$$

Since  $0 \leq s_k < 1$ , we have

$$\begin{aligned} R_k &= (1 - s_k)g_k^t(x_k - x^*) + s_k g_{k-1}^t(x_{k-1} - x^*) + s_k g_{k-1}^t(x_k - x_{k-1}) \\ &\geq (1 - s_k)(f_k - f^*) + s_k(f_{k-1} - f^*) + s_k g_{k-1}^t(x_k - x_{k-1}) \\ &= (1 - s_k)(f_k - f^*) + s_k(f_{k-1} - f^*) + s_k(f_k - f_{k-1}) - s_k v_k \\ &= f_k - f^* - s_k v_k \geq f_k - w - s_k v_k = (f_k - w)\left(1 - \frac{s_k v_k}{f_k - w}\right) \geq \beta_k(f_k - w). \end{aligned}$$

Hence,

$$\lambda_k^2 \|d_k\|^2 - 2\lambda_k R_k \leq \lambda_k^2 \|d_k\|^2 - 2\lambda_k \beta_k (f_k - w) = -\beta_k^2 \frac{(f_k - w)^2}{\|d_k\|^2} < 0.$$

Therefore, from (3.3.6), we have that  $\|x_{k+1} - x^*\|^2 < \|x_k - x^*\|^2$  for all  $k$ , that is,  $\{\|x_k - x^*\|\}$  is a nonnegative, monotone decreasing sequence, and so, we must have,

$$\beta_k^2 \frac{(f_k - w)^2}{\|d_k\|^2} \rightarrow 0 \quad \text{as } k \rightarrow \infty.$$

Since  $\|d_k\| \leq (1-s_k)\|g_k\| + s_k\|g_{k-1}\| < M$  by the boundedness of the subgradient norms, and  $\beta_k$  is bounded away from 0, we have that  $\lim_{k \rightarrow \infty} f_k = w$ . This completes the proof.

The above theorem establishes convergence of the procedure (3.1.3a) and (3.3.4) used in conjunction with the step-length rule (3.3.3), given an upper bound on the optimal objective function value  $w$  that is used as a *fixed* target value. On the other hand, the following corollary addresses the convergence behavior when the upper bounding target values vary.

**Corollary 3.3.1.** Suppose that for each  $k$ , the target value  $w$  is permitted to vary, and is taken as some  $w_k$  satisfying  $f_k > w_k \geq f^*$  for all  $k$ . If  $\alpha_k$  and  $\beta_k$  are selected to satisfy (3.3.5) with  $w$  replaced by  $w_k$ , and if  $\|g_k\| < M$  for all  $k$ , for some  $M$ , then, for any given  $\delta_1 > 0$ , there exists a  $K_1$  such that

$$f_k - w_k \leq \delta_1 \quad \text{for all } k \geq K_1. \quad (3.3.7)$$

Proof: Following the proof of Theorem 3.3.1 for the case  $f_k > w_k$  for all  $k$ , we have

$$\lim_{k \rightarrow \infty} \beta_k^2 \frac{(f_k - w_k)^2}{\|d_k\|^2} = 0$$

which gives  $\lim_{k \rightarrow \infty} (f_k - w_k) = 0$ . The assertion therefore holds true, and the proof is complete.

For convergence to hold in Theorem 3.3.1 and Corollary 3.3.1, we require the condition (3.3.5). To avoid the case of simply taking an anti-subgradient as the direction of motion (i.e.,  $d_k = -g_k$ ), we may wish to restrict the choice of  $s_k$  in (3.3.5) to  $s_k > 0$  as far as possible.

Now, to construct a procedure for selecting  $\alpha_k$  that defines  $s_k$  via (3.1.4), let us first assume that for all  $k$ ,

$$\|g_k - g_{k-1}\| \neq 0 \quad \text{and} \quad g_k^t r_k \neq 0. \quad (3.3.8)$$

Accordingly, define

$$a_k = 1 - \frac{\|g_k - g_{k-1}\|}{g_k^t r_k} \quad \text{and} \quad \bar{a}_k = \max\{0, a_k\} \quad (3.3.9)$$

and consider the following results.

**Lemma 3.3.2.** For each  $k$  such that (3.3.8) holds, if  $\alpha_k$  is chosen to satisfy

$$\begin{cases} \bar{a}_k < \alpha_k^2 < 1 & \text{if } g_k^t r_k > 0 \\ 1 < \alpha_k^2 < \bar{a}_k & \text{if } g_k^t r_k < 0 \end{cases} \quad (3.3.10)$$

then,  $0 < s_k < 1$ .

Proof: If  $g_k^t r_k > 0$ , then  $\alpha_k^2 < 1$  implies that  $s_k > 0$  from (3.1.4). Since  $\bar{a}_k < 1$ , we have

$$1 > \alpha_k^2 > \bar{a}_k \geq 1 - \frac{\|g_k - g_{k-1}\|}{g_k^t r_k}$$

which implies that

$$(1 - \alpha_k^2) < \frac{\|g_k - g_{k-1}\|}{g_k^t r_k}$$

and hence that  $0 < s_k < 1$ . For the case of  $g_k^t r_k < 0$ , we again have  $s_k > 0$  in (3.1.4) since

$\alpha_k^2 > 1$ . Since  $\bar{a}_k = a_k > 1$ , we have

$$\alpha_k^2 < a_k = 1 - \frac{\|g_k - g_{k-1}\|}{g_k^t r_k}.$$

Hence,

$$(1 - \alpha_k^2) > \frac{\|g_k - g_{k-1}\|}{g_k^t r_k}.$$

Therefore, we have  $s_k < 1$  since  $g_k^t r_k < 0$ . This completes the proof.

Now, in addition to the condition (3.3.8), let us assume that for each  $k$ ,  $v_k \neq 0$ , and define

$$b_k = 1 - \frac{1}{v_k} (f_k - w_k) (1 - \varepsilon_1) \frac{\|g_k - g_{k-1}\|}{g_k^t r_k} \quad \text{and} \quad \bar{b}_k = \max\{0, b_k\} \quad (3.3.11)$$

for any given  $0 < \varepsilon_1 < 1$  and for some target value  $w_k$  satisfying  $f_k > w_k \geq f^*$ . Note that if  $g_k^t r_k > 0$ , then  $\bar{b}_k < 1$ , and if  $g_k^t r_k < 0$ , then  $\bar{b}_k = b_k > 1$ .

**Lemma 3.3.3.** For each  $k$  such that  $v_k \neq 0$  and (3.3.8) holds, if  $f_k > w_k$  and if  $\alpha_k$  is chosen to

$$\text{satisfy } \begin{cases} \bar{b}_k < \alpha_k^2 < 1 & \text{if } g_k^t r_k > 0 \\ 1 < \alpha_k^2 < \bar{b}_k & \text{if } g_k^t r_k < 0 \end{cases} \quad (3.3.12)$$

then, we have

$$\varepsilon_{2k} \equiv 1 - \frac{s_k v_k}{f_k - w_k} > \varepsilon_1.$$

Proof: Observe that when  $g_k^t r_k$  is of either sign, we have

$$\begin{aligned} s_k &= (1 - \alpha_k^2) \frac{g_k^t r_k}{\|g_k - g_{k-1}\|} < (1 - \bar{b}_k) \frac{g_k^t r_k}{\|g_k - g_{k-1}\|} \\ &\leq (1 - b_k) \frac{g_k^t r_k}{\|g_k - g_{k-1}\|} = \frac{1}{v_k} (f_k - w_k) (1 - \varepsilon_1). \end{aligned}$$

This implies that

$$\frac{s_k v_k}{f_k - w_k} < 1 - \varepsilon_1$$

and hence the assertion holds true.

From Lemmas 3.3.2 and 3.3.3, combining (3.3.10) and (3.3.12), we see that for each  $k$ , whenever  $v_k \neq 0$  and (3.3.8) holds true, if  $\alpha_k$  is chosen to satisfy

$$\max\{\sqrt{\bar{a}_k}, \sqrt{\bar{b}_k}\} < \alpha_k < 1, \quad \text{if } g_k^t r_k > 0$$

$$1 < \alpha_k < \min\{\sqrt{\bar{a}_k}, \sqrt{\bar{b}_k}\}, \quad \text{if } g_k^t r_k < 0$$

then, noting (3.3.5), we will have that

$$0 < s_k < 1 \quad \text{and} \quad \varepsilon_1 < \varepsilon_{2k}. \quad (3.3.13)$$

On the other hand, if (3.3.8) is violated, we can simply select  $s_k = 0$  so that (3.3.5) holds true, thereby supporting Theorem 3.3.1 and Corollary 3.3.1 in either case. Moreover, note that while defining  $b_k$ , we require that  $v_k \neq 0$ . However, this requirement is not necessary for the assertions of Theorem 3.3.1 and Corollary 3.3.1 to hold true, since  $\varepsilon_{2k} = 1$  if  $v_k = 0$ . Hence, to avoid the case of an undefined  $b_k$  in (3.3.11) when  $v_k = 0$ , we can set  $\bar{b}_k = 0$  if  $g_k^t r_k > 0$ , and set  $\bar{b}_k = \bar{M}$  for some large number  $\bar{M} \geq \bar{a}_k$  if  $g_k^t r_k < 0$ . (Note that we can simply take  $\bar{M} = \bar{a}_k$  in this latter case.) Then, under assumption (3.3.8), the above choice of  $\alpha_k$  still guarantees (3.3.13). To summarize, the routine for finding the direction of motion, along with the prescribed parameters for determining the step-length, is formally stated below.

**Memoryless Space Dilation and Reduction (MSDR) Strategy: Direction and Step-Length Parameter Subroutine**

Let  $\varepsilon_3 > 0$  and  $\varepsilon_4 > 0$  be some (small) tolerances, and let  $0 < \varepsilon_1 < 1$  be a chosen step-length parameter. (We chose  $\varepsilon_1 = 0.1$  and  $\varepsilon_3 = \varepsilon_4 = 10^{-6}$ .)

**Step 1.** If  $\|g_k - g_{k-1}\| < \varepsilon_3$ , exit the subroutine with  $d_k = -g_k$ , and let  $s_k = 0$  and  $\varepsilon_{2k} = 1$ .

Otherwise, compute  $r_k$  as in (3.1.1). If  $g_k^t r_k = 0$  (practically,  $\leq 10^{-6}$ ), exit the subroutine with

$d_k = -g_k$ , and let  $s_k = 0$  and  $\varepsilon_{2k} = 1$ . Otherwise, find  $a_k$  and  $\bar{a}_k$  as in (3.3.9), and compute  $v_k$  as in (3.3.2). Proceed to Step 2.

**Step 2.** If  $v_k > \varepsilon_4$ , compute  $b_k$  and  $\bar{b}_k$  as in (3.3.11) and proceed to Step 3. Otherwise, set

$$\bar{b}_k = \begin{cases} 0, & \text{if } g_k^t r_k > 0 \\ \bar{a}_k, & \text{if } g_k^t r_k < 0 \end{cases} \quad (3.3.14)$$

and proceed to Step 3.

**Step 3.** Select a suitable combination weight  $\phi_k$  satisfying  $0 < \phi_k < 1$  (we selected  $\phi_k = 0.5$ ),

and compute  $\bar{\alpha}_k$  as

$$\bar{\alpha}_k = \begin{cases} \max\{\sqrt{\bar{a}_k}, \sqrt{\bar{b}_k}\}, & \text{if } g_k^t r_k > 0 \\ \min\{\sqrt{\bar{a}_k}, \sqrt{\bar{b}_k}\}, & \text{if } g_k^t r_k < 0. \end{cases} \quad (3.3.15)$$

Determine the transformation parameter  $\alpha_k$  as

$$\alpha_k = \phi_k + (1 - \phi_k)\bar{\alpha}_k,$$

(3.3.16) and proceed to Step 4.

**Step 4.** Compute a direction of motion  $d_k$  as  $d_k = -g_k + (1 - \alpha_k^2)(g_k^t r_k)r_k$ , or equivalently, as

$$d_k = -[(1 - s_k)g_k + s_k g_{k-1}],$$

where  $s_k$  is defined as

$$s_k = (1 - \alpha_k^2) \frac{g_k^t r_k}{\|g_k - g_{k-1}\|}, \quad 0 < s_k < 1.$$

Compute

$$\varepsilon_{2k} = 1 - \frac{s_k v_k}{f_k - w},$$

and terminate the subroutine.

The overall algorithm for which Theorem 3.3.1 and Corollary 3.3.1 hold true can then be stated as follows. At the  $k$ -th iteration of the prescribed procedure, assume that we have a current subgradient  $g_k$  and a previous subgradient  $g_{k-1}$  (when  $k \geq 2$ ). Then the direction of motion  $d_k$  can be computed by the foregoing subroutine MSDR when  $k \geq 2$ , with  $d_1$  taken simply as  $-g_1$ . Computing the step-length by (3.3.3), where the parameter  $\beta_k$  is chosen to satisfy (3.3.5), and where  $s_k$  is given by the subroutine MSDR, the new iterate is determined via (3.3.4), and the process is repeated.

We remark here that the result in Theorem 3.3.1 requires the target value to be chosen as an upper bound on the optimal objective value. On the other hand, when the target value is a lower bound on the optimal value, an analogous result to that of Allen *et al.* [1] for the pure subgradient strategy is elusive. However, by imbedding the foregoing scheme within the framework of a variable target value method as described in [36], we can construct a convergent algorithm. As shown in [36], because the proposed scheme satisfies the property of Theorem 3.3.1 and Corollary 3.3.1, this procedure will generate a monotone, non-increasing sequence of incumbent objective values  $\{z_k\}$  that converges to an  $\varepsilon$ -neighborhood of the optimal objective function value, for any chosen  $\varepsilon > 0$ , without any *a priori* knowledge about the optimal objective function value. Accompanying this scheme, is a sequence of target values  $\{w_l\}$  updated via an outer loop of the procedure, which guides the step-length selection process defined by (3.3.3) during the inner loop iterations. (Note that these target values might lie above or below the

optimal value.) The inner loop of the procedure attempts to approach the current target value within a variable tolerance, and depending on whether or not it is successful, it readjusts the target value along with the tolerance in an outer loop update. For the sake of completeness, the specific details of this algorithm are provided below, with the main inner loop being suitably modified to accommodate the MSDR strategy. (Note that all parameter values are fixed as prescribed.)

### **Variable Target Memoryless Space Dilation/Reduction Algorithm (VT-MSDR)**

**Initialization.** Select termination parameters  $\varepsilon_0$  for the tolerance on subgradient norms,  $\varepsilon$  for the overall convergence tolerance, and  $k_{\max}$  for the limit on the maximum number of iterations. (In our runs, we have used  $\varepsilon_0 = 10^{-6}$ ,  $\varepsilon = 0.1$ , and  $k_{\max} = 2000$ .) Other parameters used in the algorithm are defined as follows (recommended values are specified):  $\beta \in (0, 1)$  = step-length parameter = 0.95;  $\sigma \in (0, \frac{1}{3})$  = acceptance interval parameter = 0.15;  $\bar{\tau}$  = maximum allowable iterations in the inner loop without coming within the acceptance tolerance of the target value = 75;  $\bar{\gamma}$  = maximum consecutive nonimprovements permitted within the inner loop (initialized at 20 and incremented by 10 each time this limit is reached, up to a value of 50);  $\eta \in (0, 1)$  = fraction of cumulative inner loop improvement that is used to decrease the target value = 0.75;  $r \in (0, 1)$  = factor for ensuring adequate step sizes during initial, improving iterations = 0.1. (Related parameter  $\bar{r} = 1 + r$ .)

Select a starting solution  $x_1 \in X$ , compute its objective function value  $f_1$ , and let the initial direction of motion be  $d_1 = -g_1$ , where  $g_1$  is a subgradient of  $f$  at  $x_1$ . If  $\|g_1\| < \varepsilon_0$ , then stop with

$x_1$  as a near-optimal solution. Otherwise, set  $x^* = x_1$  and  $g^* = g_1$ , and record  $z_1 = f_1$  as the best known objective function value. Initialize the outer loop counter  $L = 1$ , and compute the initial target value  $w_1 = \max\{LB, f_1 - \|g_1\|^2/2\}$ , where  $LB$  is any known lower bound on  $f^*$ , being taken as  $-\infty$  if no such lower bound is available. Also, select the initial *acceptance tolerance* that defines an acceptable degree of proximity to the current target value as  $\varepsilon = \sigma(f_1 - w_1)$ . Set the counter  $\tau$  of current inner loop iterations to zero, the counter  $\gamma$  of consecutive nonimprovements to zero, and the total inner loop iteration counter  $k$  to one. Put  $\Delta = 0$ , where  $\Delta$  measures accumulated improvements within the current inner loop iterations.

### Step 1 (Inner Loop Main Iteration)

**Step 1(a).** If  $k > k_{\max}$ , stop. Else, use subroutine MSDR to compute the search direction  $d_k$  along with the step-length parameter  $\varepsilon_{2k}$ . If  $\|d_k\| < \varepsilon_0$ , then set  $d_k = -g_k$  and  $\varepsilon_{2k} = 1$ . Also, compute the step-length

$$\lambda_k = \beta_k \left[ \frac{f_k - w_L}{\|d_k\|^2} \right],$$

where  $\beta_k = \beta$  if  $k \leq 1000$ , and  $\beta_k = \text{minimum}\{\varepsilon_{2k}, \beta\}$  otherwise. (3.3.17)

**Step 1(b).** Find the new iterate  $x_{k+1} = P_X[x_k + \lambda_k d_k]$ , evaluate its objective function value  $f_{k+1}$ , and find a subgradient  $g_{k+1}$  of  $f$  at  $x_{k+1}$ . If  $\|g_{k+1}\| < \varepsilon_0$ , terminate the algorithm with  $x_{k+1}$  as a (near) optimal solution. Otherwise, increment  $\tau$  by 1. If  $f_{k+1} < z_k$ , update  $\Delta \leftarrow \Delta (z_k - f_{k+1})$ , and go to Step 2(a). Otherwise, go to Step 2(b).

**Step 2(a) (Improvement in the Inner Loop).** Put  $\gamma = 0$ ,  $z_{k+1} = f_{k+1}$ , and update  $x_k^* = x_{k+1}$  and  $g_k^* = g_{k+1}$ . If  $z_{k+1} \leq w_L + \varepsilon_L$ , then go to Step 3(a). Otherwise, if  $\tau \geq \bar{\tau}$ , go to Step 3(b), or else, increment  $k$  by one, and return to Step 1.

**Step 2(b) (Nonimprovement in the Inner Loop).** Put  $z_{k+1} = z_k$ , and increment  $\gamma$  by one. If  $\gamma \geq \bar{\gamma}$  or  $\tau \geq \bar{\tau}$ , go to Step 3(b). Otherwise, increment  $k$  by one, and return to Step 1.

**Step 3(a) (Outer Loop Success Iteration:  $z_{k+1} \leq w_L + \varepsilon_L$ . Decrease Target Value and Adjust Acceptance Tolerance.)** Compute a new target value as

$$w_{L+1} = z_{k+1} - \max\{\varepsilon_L + \eta\Delta, r|z_{k+1}|\},$$

and if  $w_{L+1}$  is determined by the second term in the above maximand, divide  $r$  by  $\bar{r}$ . Let

$$\varepsilon_{L+1} = \max\{(z_{k+1} - w_{L+1})\sigma, \varepsilon\}.$$

Put  $\tau = 0$  and  $\Delta = 0$ , increment  $L$  and  $k$  by one, and return to Step 1.

**Step 3(b) (Outer Loop Failure Iteration:  $z_{k+1} > w_L + \varepsilon_L$ . Increase Target Value and Adjust Acceptance Tolerance.)** Compute a new target value as

$$w_{L+1} = \frac{(z_{k+1} - \varepsilon_L) + w_L}{2} \text{ and let } \varepsilon_{L+1} = \max\{(z_{k+1} - w_{L+1})\sigma, \varepsilon\}.$$

If  $\gamma \geq \bar{\gamma}$ , adjust  $\bar{\gamma}$  as recommended at the Initialization step. If  $w_{L+1} - w_L \leq 0.1$ , then replace  $\beta$  by  $\max\{\beta / 2, 10^{-6}\}$ . Put  $\gamma = 0$ ,  $\tau = 0$ ,  $\Delta = 0$ , and increment  $L$  and  $k$  by one. Reset  $x_k = x^*$ ,

$f_k = z_k$ , and  $g_k = g^*$ , and let  $d_k = -g_k$  and  $\varepsilon_{2k} = 1$ . Compute the step-length  $\lambda_k$  as in (3.3.17), and return to Step 1(b).

## **Chapter 4**

### **Outline of Related Algorithmic Variants**

In the foregoing chapter, we have described a theoretically convergent memoryless space dilation/reduction algorithm. For the sake of computational interest, we also consider now some other memoryless as well as limited memory variants of this algorithm, along with Shor and Zhurbenko's  $r$ -algorithm [44]. In addition, within the framework of the variable target value method, we test our variant of Polyak's [32] strategy that uses projections onto a pair of modified Kelley's cutting planes to arrive at a search direction. This direction yields a different combination of the current and the previous subgradients than does the MSDR strategy.

Originally the LMSD and MSD algorithm variants were run separately for Shor's  $r$ -algorithm and VTVM search strategies using the same direction finding routines. It was observed that Shor's  $r$ -algorithm gives rapid improvement during the first few iterations but is slow to converge to the actual optimum. Therefore, it was decided to merge the two search techniques, using Shor's step-length procedure for initial iterations and the VTVM step-lengths for later iterations for the LMSD and MSD strategies. The ideal number of Shor's step-lengths before switching to the VTVM strategy was experimentally determined to be 750 iterations.

It should be noted that only the VT-MSDR Algorithm and the  $r$ -algorithm are theoretically convergent; the remaining procedures must be viewed as heuristic methods. Also, in all algorithms using VTVM,  $\beta$  is initialized at 0.95 and adjusted as in the VTVM [36]. The tested algorithms are summarized below.

## **4.1 Variable Target Memoryless Space Dilation and Reduction Algorithm**

### **(VT-MSDR):**

This is the algorithm described in Chapter 3. In Step 1, if the normalized difference between the current and previous subgradients ( $q_k$ ) is  $< 0.1$ , the previous  $\alpha$  value ( $\alpha_{k-1}$ ) is retained, and  $s_k$  is calculated directly. Using (3.3.14) in Step 2,  $\bar{b}_k$  is calculated only if  $v_k < 10^{-6}$ .

### **4.2 Shor and Khurbenko's $r$ -Algorithm:**

This is the algorithm due to Shor and Zhurbenko [44]. (We used the space dilation parameter  $\alpha = 1/3$  as recommended.) Shor and Shabashova [43] and Lemarechal [25] suggest two step-length strategies for this algorithm. After several trial runs, we found the following choice of step-lengths to yield the best results:

$$\lambda_k = h_k / \|d_k\|, \text{ where } h_k = h_0 \delta^{k-1}, \text{ and where } h_0 = 7 \text{ and } \delta = 0.95.$$

This algorithm was also run for a maximum limit of 2000 iterations and the maximum consecutive non-improvement iteration limit ( $\bar{\gamma}$ ) was set to 20, i.e., if 20 iterations do not result in an improved solution, the search procedure proceeds with the Hessian matrix reinitialized to the identity matrix.

### **4.3 Variable Target Memoryless Space Dilation Algorithm (VT-MSD):**

This is the same memoryless algorithm as VT-MSDR except that instead of the direction finding routine MSDR, we employ a fixed dilation parameter  $\alpha_k = 1/3$  along with a combination of step-length strategies used for the two foregoing procedures. Accordingly, Step 1(a) of the algorithm VT-MSDR described in Chapter 3 gets replaced by the following.

**Step 1(a).** If  $k > k_{\max}$ , stop. Else, compute the search direction

$$d_k = \begin{cases} -g_k & \text{if } \|q_k\| < 10^{-6} \\ -g_k + \left[ \frac{(8/9)q_k' g_k}{\|q_k\|^2} \right] q_k & \text{otherwise} \end{cases} \quad (4.3.1)$$

where  $q_k = g_k - g_{k-1}$ . If  $\|d_k\| < 10^{-6}$ , set  $d_k = -g_k$ . Also, compute the step-length

$$\lambda_k = \frac{7(0.95)^{k-1}}{\|d_k\|} \text{ if } k < 750, \text{ and } \lambda_k = \beta \left[ \frac{f_k - w_\ell}{\|d_k\|^2} \right], \text{ otherwise.} \quad (4.3.2)$$

### **4.4 Variable Target Limited Memory Space Dilation Algorithm (VT-LMSD):**

This is a two-step limited memory variant of the foregoing procedure. Specifically, starting with the negative subgradient direction, we use at the next iteration a one-step update as given by (4.3.1), except that we use  $10^{-1}$  in (4.3.1) in lieu of  $10^{-6}$ , and then use a two-step limited memory update at the next iteration (in case  $\|q_k\| < 10^{-1}$  and  $\|d_k\| > 10^{-6}$  in (4.3.1)). Thereafter, we stay with the limited memory two-step lag direction so long as the denominators in the formulae remain greater than  $10^{-1}$ , falling back to the one-step update direction or the anti-subgradient direction

otherwise, and then building up again to the two-step update direction as before. Intuitively, we can see that this 2-step upgrade strategy is intended to perform closer to the  $r$  - algorithm by adopting an additional update step beyond the memoryless scheme, but involving a limited increase in storage requirements. Also, in this scheme, the step-length is selected according to (4.3.2).

#### **4.5 Variable Target Polyak Kelley-Cut Algorithm (VT-PKC):**

In Polyak's [32] scheme, given an iterate  $x_k$  and having determined a subgradient  $g_k$  of  $f$  at  $x_k$ , the next iterate is computed by first projecting  $x_k$  onto the intersection of the pair of Kelley's cutting planes

$$(x - x_k)^t g_k \leq (z^* - f_k) \quad \text{and} \quad (x - x_{k-1})^t g_{k-1} \leq (z^* - f_{k-1}), \quad (4.5.1)$$

where  $z^*$  is the optimal objective function value. However,  $z^*$  is (typically) unknown, and by our step-length strategy (3.3.17) in the variable target algorithm, we are estimating  $(z^* - f_k)$  by  $\beta(w_L - f_k)$ . Hence, in lieu of (4.5.1), we project  $x_k$  onto the pair of Kelley cuts

$$(x - x_k)^t g_k \leq \beta(w_L - f_k) \quad \text{and} \quad (x - x_{k-1})^t g_{k-1} \leq [f_k + \beta(w_L - f_k)] - f_{k-1}. \quad (4.5.2)$$

This yields a solution of the form  $x_k - \psi_1 g_k - \psi_2 g_{k-1}$ , where  $\psi_1$  and  $\psi_2$  are easily determined coefficients (see Polyak [32]). Using  $x_k - \psi_1 g_k - \psi_2 g_{k-1}$  effectively as " $x_k + \lambda_k d_k$ " at Step 1(a) of the algorithm of Chapter 3, we accordingly obtain the required variant of Polyak's algorithm. Note that the search direction here is composed of some combination of the current and the previous subgradients, as in the proposed MSDR procedure.

## Chapter 5

### Computational Experience

In this Chapter, we provide computational results for the five procedures outlined in Chapter 4. For this purpose, we used a set of 15 test problems from the literature, as well as some randomly generated dual transportation and assignment problems. The 15 test problems were taken from various sources from the literature and are described in [36] along with their prescribed starting solutions (also, see Kiwiel [20]). The dual transportation problems are of the form

$$\mathbf{TR}(\mathbf{n}): \text{Minimize } \left[ -\sum_{i=1}^n S_i x_i + \sum_{j=1}^n D_j \max_{1 \leq i \leq n} \{x_i - c_{ij}\} \right]$$

where there are assumed to be  $n$  sources and  $n$  destinations having respective supplies  $S_i$ ,  $i = 1, \dots, n$ , and demands  $D_j$ ,  $j = 1, \dots, n$ , and where  $c_{ij}$  is the cost of shipping a unit of the product from source  $i$  to destination  $j$ . (Here,  $x$  represents the dual variable vector with respect to the standard transportation supply constraints and was initialized at zero in the test runs.) The corresponding primal transportation problems were generated using the technique described in [34] and [37]. The dual assignment problems, denoted by  $\mathbf{AS}(\mathbf{n})$ , were generated similarly, having  $S_i = 1$  and  $D_j = 1 \forall i, j$  in [14].

All the algorithms were coded in C and were run on an IBM 3090, Model 300E computer. The parameter choices were fixed at the stated prescribed values for all the runs. Table 1 gives the results obtained for the aforementioned 15 standard test problems, and Table 2 gives the results obtained for some 15 dual transportation and assignment problems of various specified sizes. In each of these tables,  $f(x^*)$  denotes the known optimal objective value,  $f(x_{best})$  denotes the objective value of the best solution found by each of the corresponding methods, and “cpu secs” denotes the cpu time in seconds. All methods were run for a maximum of 2000 iterations.

Examining the results in Table 1, we observe that the methods VT-MSDR and the  $r$ -Algorithm yield a comparable performance with respect to the quality of the solution produced and the effort required, except for Problem 3 for which the  $r$ -Algorithm stalled relatively further from optimality. The fixed dilation and the limited memory variants VT-MSD and VT-LMSD improve over VT-MSDR for a few problems such as Problems 2, 5, 6, and 7, but similar to the  $r$ -Algorithm, they experienced convergence difficulties with Problem 3. By far, our modified version of Polyak’s method (VT-PKC), where we have employed the modified Kelley cuts (4.5.2) and have embedded the resulting direction and step-length strategy in the variable target value method of Chapter 3, provided the best overall performance in terms of the quality of solutions produced for this set of test problems.

Subgradient-based methods usually find dual transportation problems very challenging to solve, and this is evident from the % gap from optimality at termination for the test cases reported in Table 2. Here, VT-MSDR exhibits a superior performance over the  $r$ -Algorithm as problem size increases, both with respect to solution quality, and particularly, with respect to computational effort. The variant VT-MSD exhibits a comparable performance to VT-MSDR, with the latter again dominating the performance as problem size increases. The limited memory

variant VT-LMSD performs identically as the algorithm VT-MSDR with respect to solution quality except that it stalled far from optimality for Problem 2. However, because of the two-step update process, it is computationally somewhat more expensive. The method VT-PKC exhibits a closely competitive performance in comparison with VT-MSDR on this class of dual transportation problems. In contrast, the dual assignment problems are relatively easier to solve, although still presenting a useful set of test problems. As seen in Table 2, the class of procedures VT-MSDR, VT-MSD, VT-LMSD, and VT-PKC all yield competitive results, with VT-LMSD producing the best quality solutions, although not significantly better than VT-MSDR. Again, in comparison with VT-MSDR, the  $r$ -Algorithm requires a substantially greater effort and does not produce solutions of as good a quality as the other methods over the same number of iterations. Overall, with respect to solution quality and effort, VT-MSDR and VT-PKC appear to yield the best performance.

**Table 5.1.** Computational Results Using Test Problems from the Literature.

Problem	$n$	$f(x^*)$	VT-MSDR		$r$ -Algorithm		VT-MSD		VT-LMSD		VT-PKC	
			$f(x_{best})$	Cpu secs	$f(x_{best})$	cpu secs	$f(x_{best})$	cpu secs	$f(x_{best})$	cpu secs	$f(x_{best})$	cpu secs
1	5	22.600	22.600	0.16	22.610	0.28	22.600	0.18	22.602	0.19	22.602	0.15
2	10	-0.0841	-0.809	1.07	-0.834	0.77	-0.838	1.07	-0.838	1.48	-0.818	0.97
3	50	0.0	29.694	0.5	187.736	7.36	185.408	0.55	1067.857	1.04	0.0	0.49
4	48	-638565	-540062.3	4.86	-522383	4.11	-533958.3	4.97	-540062.3	6.49	-632919.4	4.6
5	48	-9870	-8774.3	4.97	-9675.06	3.98	-9851.205	4.98	-8114.299	6.54	-9869.999	4.6
6	50	0.0	6.165	6.6	0.132	9.48	0.0235	6.52	6.165	8.6	0.007	6.13
7	30	0.0	5.891	4.46	0.094	3.04	0.0513	1.71	0.3	1.97	0.026	1.58
8	4	0.707	0.707	0.11	0.707	0.12	0.707	0.13	0.707	0.17	0.707	0.1
9	4	1.014	1.014	0.09	1.015	0.19	1.014	0.13	1.014	0.15	1.014	0.1
10	6	1.015	0.115	1.07	0.062	1.21	0.069	1.1	0.115	1.54	0.11	1.04
11	5	-32.349	-32.348	0.21	-32.195	0.14	-32.322	0.23	-32.345	0.32	-32.346	0.21
12	4	-44.0	-44.0	0.09	-44.0	0.1	-44.0	0.12	-44.0	0.14	-44.0	0.08
13	4	23.887	29.792	0.53	23.887	0.58	23.887	0.2	23.887	0.71	23.887	0.5
14	6	68.829	68.83	0.81	68.847	0.35	68.83	0.86	68.83	1.19	68.83	0.79
15	10	-0.368	-0.362	1.1	-0.345	0.85	-0.362	1.17	-0.362	1.53	-0.362	1.07

**Table 5.2.** Computational Results for Dual Transportation and Assignment Problems.

Problem	Type ( $n$ )	$f(x^*)$	VT-MSDR		$r$ -Algorithm		VT-MSD		VT-LMSD		VT-PKC	
			% gap	Cpu secs	% gap	cpu secs	% gap	cpu secs	% gap	cpu secs	% gap	cpu secs
1	TR(20)	-4339	4.19	1.33	2.83	2.75	6.22	1.37	4.19	1.87	4.066	1.29
2	TR(30)	-6406	4.07	2.73	8.48	6.58	6.37	2.79	42.29	3.75	4.45	2.65
3	TR(50)	-11622	3.35	7.17	2.01	16.86	2.16	7.25	3.35	9.83	3.15	6.65
4	TR(80)	-18444	5.03	17.9	3.10	43.61	3.70	17.35	5.03	23.78	5.13	16.62
5	TR(100)	-23394	7.57	27.25	6.08	68.7	3.83	26.79	7.57	36.75	7.34	25.43
6	TR(120)	-27119	3.57	38.47	0.89	97.74	3.25	38.11	3.57	52.15	3.68	36.11
7	TR(150)	-34061	4.79	60.03	4.04	152.61	5.21	59.09	4.79	80.37	4.83	56.18
8	TR(180)	-40440	3.65	87.16	8.91	241.29	4.38	84.07	3.65	114.88	3.99	80.31
9	TR(200)	-45170	4.01	107.22	9.26	305.22	5.55	105.24	4.00	143.25	4.54	100.3
10	TR(250)	-59044	11.69	165.44	19.54	436.95	14.84	160.75	11.69	218.72	11.65	153.05
11	TR(300)	-72222	8.41	241.13	13.72	639.99	11.86	231.86	8.41	317.63	8.37	221.13
12	AS(180)	-1915	0.01	27.13	0.00	17.2	0.00	2.82	0.00	52.29	0.00	18.75
13	AS(200)	-2157	0.02	28.94	1.99	73.53	0.00	4.67	0.00	67.52	0.00	34.63
14	AS(250)	-2697	0.00	50.77	4.16	164.07	0.00	14.97	0.00	110.85	0.01	55.32
15	AS(300)	-3246	0.00	84.61	7.14	242.42	1.61	235.95	0.00	161.83	0.00	71.7

## **Chapter 6**

### **Conclusions and Recommendations for Future Research**

Shor and Zhurbenko's [44]  $r$ -Algorithm is a well-known and popular method for solving nondifferentiable optimization problems that employs a space dilation strategy in the direction of the difference between two successive subgradients. In this thesis, we have investigated the design of a memoryless variant of this algorithm. The variant developed permits a combination of space dilation as well as reduction strategies. Using a popular and computationally effective step-length rule (in contrast with the divergent series step-length rule), along with variable upper bounding target values, we have established the convergence of this proposed procedure. This convergence property permitted us to embed the proposed scheme into a variable target value method developed by Sherali *et al.* [36], which assumes no *a priori* knowledge of the optimal objective function value, and that employs target values that might lie above or below the optimal value, and yet ensure overall convergence.

The resulting procedure, called the VT-MSDR Algorithm, has been tested against the  $r$ -Algorithm using several standard problems from the literature as well as a set of randomly generated dual transportation and assignment problems. (Although these test problems are not too large, they are all quite challenging to solve and provide an adequate test-bed for our methods.) In addition to these algorithms, we have developed and tested three additional variants

from the viewpoint of computational interest, all embedded within the same variable target value method. The first variant (VT-MSD) uses a fixed dilation parameter in a memoryless space dilation scheme. The second variant (VT-LMSD) also uses a fixed dilation scheme, but employs a two-step limited memory update procedure. The third variant (VT-PKC) adopts a different combination of the two successive subgradients using Polyak's [32] scheme of projecting onto a pair of modified Kelley cutting planes. These cutting planes employ an estimate of the unknown optimal value as prescribed by the variable target value method within which this procedure is embedded.

The computational results exhibit that VT-MSDR and VT-PKC provide an overall competitive performance relative to the other methods tested with respect to solution quality and computational effort. The  $r$ -Algorithm becomes increasingly more expensive with an increase in problem size, while not providing any gain in solution quality. The fixed dilation (with no reduction) strategy VT-MSD provides a comparable, though second-choice, alternative to VT-MSDR. Employing a two-step limited memory extension over VT-MSD sometimes helps in improving the solution quality, although it adds to computational effort and is not as robust a procedure.

While our focus here has been on investigating space dilation (and reduction) methods and their memoryless and limited memory variants, comparisons with other methods and further tests on solving Lagrangian relaxations of various problems, for example, is of interest. This is recommended for future research.

## References

- [1] Allen, E., Helgason, R., Kennington, J., and Shetty, B., A Generalization of Polyak's Convergence Result for Subgradient Optimization, *Mathematical Programming*, 37, pp. 309-317, 1987.
- [2] Bazaraa, M. S., Sherali, H. D., and Shetty, C. M., *Nonlinear Programming: Theory and Algorithms*, Second Edition, John Wiley & Sons, New York, 1993.
- [3] Camerini, P. M., Fratta, L., and Maffioli, F., On Improving Relaxation Methods by Modified Gradient Techniques, *Mathematical Programming Study*, 3, pp. 26-34, 1975.
- [4] Clarke, F. H., Generalized Gradients and Applications, *Transaction of American Mathematical Society*, 205, 247-262, 1975.
- [5] Demyanov, V. F., and Vasilev, L. V., *Nondifferentiable Optimization*, Springer-Verlag, 1985.
- [6] Fletcher, R., and Reeves, C. M., Function Minimization by Conjugate Gradients, *Computer Journal*, 7, pp. 149-154, 1959.
- [7] Fletcher, R., and Powell, M. J. D., A Rapidly Convergent Descent Method for Minimization, *The Computer Journal*, 6, pp. 163, 1963.
- [8] Hestenes, M. R., and Stiefel, E., Methods of Conjugate Gradients for Solving Linear Systems, *Journal of Research, National Bureau of Standards*, 29, pp. 403-439, 1952.
- [9] Goffin, J. L., On the Convergence Rate of the Subgradient Methods, *Mathematical Programming*, 13, pp. 329-347, 1977.
- [10] Goffin, J. L., Convergence Results in a Class of Variable Metric Subgradient Methods, *Nonlinear Programming 4*, Mangasarian, Meyer, and Robinson, eds., pp. 283-326, 1980.
- [11] Held, M., Wolfe, P., and Crowder, H. D., Validation of Subgradient Optimization, *Mathematical Programming*, 6(1), 62-88, 1974.

- [12] Hestenes, M. R., And Stiefel, E., Methods of Conjugate Gradients for Solving Linear Systems, *Journal of Research, National Bureau of Standards*, 29, pp. 403-439, 1952.
- [13] Kelley, J. E., The Cutting Plane Method for Solving Complex Programs, *SIAM Journal on Applied Mathematics*, 8, 703-712, 1960.
- [14] Kim, S., and Ahn, H., Convergence of a Generalized Subgradient Method for Nondifferentiable Convex Optimization, *Mathematical Programming*, 50, pp. 75-80, 1991.
- [15] Kim, S., Koh, S., and Ahn, H., Two-Direction Subgradient Method for Nondifferentiable Optimization Problems, *Operations Research Letters*, 6, pp. 43-46, 1987.
- [16] Kim, S., Ahn, H., and Cho, Variable Value Target Subgradient Method, *Mathematical Programming*, 49, pp. 359-369, 1991.
- [17] Kim, S., and Um, B., An Improved Subgradient Method for Constrained Nondifferentiable Optimization, *Operations Research Letters*, 14, pp. 61-64, 1993.
- [18] Kiwiel, K. C., Methods of Descent of Nondifferentiable Optimization, *Lecture notes in Mathematics*, No. 1133, Springer-Verlag, 1985.
- [19] Kiwiel, K. C., A Survey of Bundle Methods for Nondifferentiable Optimization, *Mathematical Programming*, M. Iri, and K. Tanabe, eds., KTK Scientific Publishers, Tokyo, Japan, pp. 263-282, 1989.
- [20] Kiwiel, K. C., Proximity Control in Bundle Methods for Convex Nondifferentiable Minimization, *Mathematical Programming*, 46, pp. 105-122, 1990.
- [21] Kiwiel, K. C., A Tilted Cutting Plane Proximal Bundle Method for Convex Nondifferentiable Optimization, *OR Letters*, 10, pp. 75-81, 1991.
- [22] Lemarechal, C., An Extension of Davidon Methods to Nondifferentiable Problems, *Mathematical Programming Study*, 3, pp. 95-109, 1975.
- [23] Lemarechal, C., Bundle Method in Nonsmooth Optimization, *Nonsmooth Optimization: Proceedings of IASA Workshop*, C. Lemarechal and R. Mifflin, eds., pp. 79-109, 1978.
- [24] Lemarechal, C., A View of Line-Search, *Lecture Notes in Control and*

- Information Science*, A. Auslender, W. Oettli, J. Stoer, eds., Optimization and Optimal Control; Proceedings of a Conference Held at Oberwolfach, March 16-22, pp. 59-78, 1980.
- [25] Lemarechal, C., Numerical Experiments in Nonsmooth Optimization, *Progress in Nondifferentiable Optimization*, E. A. Nurminski, ed., IIASA, pp. 61-84, 1982.
- [26] Lemarechal, C., Constructing Bundle Methods for Convex Optimization, *Fermat Days 85: Mathematics for Optimization*, Hiriart-Urruty, ed., Elsevier Science Publisher B. V., North-Holland, pp. 201-241, 1986.
- [27] Luenberger, D. G., *Introduction to Linear and Nonlinear Programming*, second ed., Addison-Wesley, Reading, Mass., 1984.
- [28] Mifflin, R., An Algorithm for Constrained Optimization with Semismooth Functions, *Mathematics of Operations Research*, 2, pp. 191-207, 1977.
- [29] Nazareth, J. L., A Relationship Between the BFGS and Conjugate-Gradient Algorithms and its Implications for New Algorithms, *SIAM Journal on Numerical Analysis*, 26, pp. 794-800, 1979.
- [30] Norkin, V. I., A Method of Minimizing a Nondifferential Function with Gradient Averaging, *Kibernetika*, 6, 88-89, 1980.
- [31] Numinski, E. A., and Zhelikhovskii, A. A.,  $\epsilon$ -Quasigradient Method for Solving Nonsmooth Extremal Problems, *Kibernetika*, No. 1, pp. 109-113, 1977.
- [32] Polyak, B. T., Minimization of Unsmooth Functionals, *Zh. Vychisl. Mat. I Mat. Fiz.*, 9, pp. 509-521, 1969 (Russian). English transl. in U.S.S.R. *Comput. Math. and Math. Phys.*, 9, pp. 14-29, 1969.
- [33] Rockafellar, R. T., A Dual Approach to Solving non-LP Programming Problems by Unconstrained Optimization, *Math Programming* 5, pp. 354-373, 1973.
- [34] Rosen, J., and Suzuki, S., Construction of Nonlinear Programming Test Problems, *Communications of ACM*, 8, pp. 113, 1965.
- [35] Sherali, H. D., Choi, G., and Ansari, Z. A., Limited Memory Space Dilation and Reduction Algorithms, *Working Paper, Department of Industrial and Systems Engineering, Virginia Polytechnic Institute and State University*, Blacksburg, Virginia, 24061-0118, 1998.
- [36] Sherali, H. D., Choi, G., and Tuncbilek, C. H., Subgradient Deflection Strategies and a Variable Target Value Method, *Working Paper, Department of Industrial and Systems Engineering, Virginia Polytechnic Institute and State University*,

Blacksburg, Virginia, 24061-0118, 1993.

- [37] Sherali, H. D., and Myers, D. C., Dual Formulations and Subgradient Optimization Strategies for Linear Programming Relaxations of Mixed Integer Programs, *Discrete Applied Mathematics*, 20, pp. 51-68, 1988.
- [38] Sherali, H. D., and Ulular, O., A Primal-Dual Conjugate Subgradient Algorithm for Specially Structured Linear and Convex Programming Problems, *Applied Mathematics and Optimization*, 20, pp. 193-221, 1989.
- [39] Shor, N. Z., The Rate of Convergence of the Generalized Gradient Descent Method, *Kibernetika*, 4(3), 79-80, 1968.
- [40] Shor, N. Z., Utilization of the Operation of Space Dilatation in the Minimization of Convex Functions, *Kibernetika*, No. 1, pp. 6-12, 1970.
- [41] Shor, N. Z., Convergence Rate of the Gradient Descent Method with Dilatation of the Space, *Kibernetika*, No. 2, pp. 80-85, 1970.
- [42] Shor, N.Z., Convergence of a Gradient Method with Space Dilation in the Direction of the Difference Between Two Successive Gradients, *Kibernetika*, No. 4, pp. 48-53, 1975.
- [43] Shor, N. Z., and Shabashova, L. P., Solution to Minmax Problems by the Method of Generalized Gradient Descent with Dilation of the Space, *Kibernetika*, No. 1, pp. 82-88, 1972.
- [44] Shor, N. Z., and Zhurbenko, N. G., A Minimization Method Using the Operation of Space Dilatation in the Direction of the Difference of Two Successive Gradients, *Kibernetika*, No. 3, pp. 51-59, 1971.
- [45] Skokov, V. A., Note on Minimization Methods Employing Space Stretching, *Kibernetika*, No. 4, pp. 115-117, 1974.
- [46] Wolfe, P., A Method of Conjugate Subgradients for Minimizing Nondifferentiable Functions, *Mathematical Programming Study*, 3, pp. 145-173, 1975.

## **Vita**

Zafar Aqeel Ansari was born in Karachi, Pakistan, on January 14, 1971. He obtained Bachelor of Science degree in Electrical Engineering from West Virginia Institute of Technology, Montgomery, West Virginia, in 1993.

He was employed as a Graduate Research Assistant from February to August, 1995, and January to May, 1996, by Dr. Hanif D. Sherali, and from January to October, 1997, by Dr. Sheldon H. Jacobson. He was also employed from August to December, 1996, as a Graduate Teaching Assistant by the Department of Industrial and Systems Engineering, Virginia Polytechnic Institute and State University.

He joined the Department of Industrial and Systems Engineering of Virginia Polytechnic Institute and State University in August, 1994, and completed all requirements for Master of Science in July, 1998.

After gaining a few years work experience, he intends to continue his studies toward a Ph.D. in the development and application of Artificial Intelligence algorithms to operations research problems.