universität wien

MASTERARBEIT

Titel der Masterarbeit

“Quasi Newton Methods for Bound Constrained Problems”

Verfasser

Behzad Azmi

angestrebter akademischer Grad

Master of Science (MSc.)

Wien, im Juni 2012

Studienkennzahl lt. Studienblatt:	A 066 821
Studienrichtung lt. Studienblatt:	Masterstudium Mathematik
Betreuer:	o.Univ.-Prof. Dr. Arnold Neumaier

Acknowledgement

First and foremost, I would like to express my sincere gratitude to my advisor Prof. Arnold Neumaier for the continuous support of my Master's study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. Special thanks go to Prof. Hermann Schichl for his thoughtful guidance and for giving me the opportunity to work with them. I would also like to thank my beloved parents for their supporting and endless love, through the duration of my studies.

Abstract

In this thesis, we are concerned with methods for solving large scale bound constrained optimization problems. This kind of problems appears in a wide range of applications and plays a crucial role in some methods for solving general constrained optimization problems, variational inequalities and complementarity problems. In the first part, we provide the general mathematical background of optimization theory for bound constrained problems. Then the most useful methods for solving these problems based on the active set strategy are discussed. In second part of this thesis, we introduce a new limited memory quasi Newton method for bound constrained problems. The new algorithm uses a combination of the steepest decent directions and quasi Newton directions to identify the optimal active bound constraints. The quasi Newton directions are computed using limited memory SR1 matrices and, if needed, by applying regularization. At the end, we present results of numerical experiments showing the relative performance of our algorithm in different parameter settings and in comparison with another algorithm.

Zusammenfassung

Diese Arbeit befasst sich mit Methoden zur Lösung von hochdimensionalen Optimierungsproblemen mit einfachen Schranken. Probleme dieser Art tauchen in einer grossen Anzahl von unterschiedlichen Anwendungen auf und spielen eine entscheidende Rolle in einigen Methoden zur Lösung von Optimierungsproblemen mit allgemeinen Nebenbedingungen, Variationsungleichungen und Komplementaritätsproblemen. Im ersten Teil dieser Arbeit beschreiben wir den allgemeinen mathematischen Hintergrund der Optimierungstheorie für Optimierungsprobleme mit Schrankenbedingungen. Danach werden die nützlichsten auf aktiven Mengen beruhenden Verfahren zur Lösung dieser Probleme diskutiert. Im zweiten Teil dieser Arbeit stellen wir ein neues Limited Memory Quasi Newton-Verfahren für hochdimensionale und einfach eingeschränkte Probleme vor. Der neue Algorithmus verwendet eine Kombination aus den steilsten Abstiegsrichtungen und Quasi Newton Richtungen, um die Menge der optimalen aktiven Variablen zu identifizieren. Die Quasi Newton-Richtungen werden mit Hilfe der Limited Memory SR1 Matrizen und, falls erforderlich, durch die Anwendung einer Regularisierung berechnet. Zum Schluss präsentieren wir die Ergebnisse von numerischen Experimenten, die die relative Performance unseres Algorithmuses bezüglich unterschiedlicher Parametereinstellungen und im Vergleich mit einem anderen Algorithmus darstellen.

Contents

1 Introduction **1**

1.1 Formulation and Motivation . . . 1

1.2 Preliminaries . . . 2

2 Proposed Algorithms for Bound Constrained Problems **4**

2.1 Introduction . . . 4

2.2 Standard Active-Set Method . . . 4

2.3 The Gradient Projection Methods . . . 7

2.3.1 Armijo Rule Along the Projection Arc . . . 8

2.3.2 Projected Search . . . 9

2.4 Approaches for Quadratic Bound Constrained problems . . . 11

2.4.1 BCQP . . . 11

2.4.2 GPCG . . . 14

2.5 A Trust Region Algorithm for Bound Constrained Problems . . . 17

2.5.1 Outline of the Algorithm . . . 18

2.5.2 The Generalized Cauchy Point . . . 20

2.5.3 A Conjugate Gradient Type Method . . . 24

2.5.4 The Algorithm . . . 27

2.6 L-BFGS-B . . . 29

2.6.1 Outline of the Algorithm . . . 30

2.6.2 Limited Memory BFGS Update . . . 31

2.6.3 The Generalized Cauchy Point . . . 34

2.6.4 Subspace Minimization . . . 38

2.6.5 The Algorithm for L-BFGS-B . . . 45

3 A New Quasi Newton Method for Bound Constrained problems **47**

3.1 Introduction . . . 47

3.2 Bent line Searches . . . 47

3.3 Overview of the Algorithm . . . 51

3.4 Convergence Analysis . . . 53

3.5 Local Step . . . 56

3.5.1 Limited Memory SR1 Update 57
3.5.2 Computing the Search Direction 59

4 Numerical Experiments 64
4.1 Introduction 64
4.2 Numerical Results 64

5 Conclusion and Future Work 71

Bibliography 72

Chapter 1

Introduction

1.1 Formulation and Motivation

In this thesis we are concerned with method for solving the bound constrained optimization problem. Such a problem appears quite naturally in a wide range of applications including optimal design problem [5], contact and friction in rigid body mechanics [19], the obstacle problem [22], journal bearing lubrication and flow through a porous medium [18]. Some approaches for solving variational inequalities and complementarity problems have been proposed [1], in which these problems are reduced to bound constrained problems. Moreover, the bound constrained optimization problem arises as a subproblem of the algorithms for solving general constrained optimization problems based on the augmented Lagrangian and penalty schemes [17, 16, 20]. In fact, some authors claim that each variable for optimization problems can only be considered meaningful within a particular interval [13]. These facts have motivated significant research dealing with development of efficient numerical algorithms for solving bound constrained optimization problems, especially when the dimension of the problem is large.

The bound constrained optimization problems have the form

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & l \leq x \leq u \end{array} \tag{1.1}$$

where the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is a sufficiently smooth function and the number of variables n is assumed to be large. Moreover, the fixed vectors l and u are lower and upper bounds on the variables respectively and the inequalities are componentwise.

The organization of this thesis goes as follows: In the rest of this chapter we review some fundamental results of optimization theory in case of bound constrained problems. Chapter 2 describes some effective methods used for solving large scale bound constrained problems. In chapter 3 we introduce a new quasi Newton method for solving bound constrained problems. In chapter 4 we present numerical results demonstrating the relative performance of our algorithm in different cases and in comparison to other algorithms. Finally, chapter 5 concludes the thesis with directions for future work.

Throughout this thesis, we use the following notation. With the superscript T we denote the transpose of a matrix(or vector), g and B represent the gradient and the Hessian matrix of the objective function respectively.

1.2 Preliminaries

For solving problem (1.1), first, we need to characterize its solutions and their properties. In addition, once the solutions of the problem are mathematically characterized, questions arise about the solution for example, whether it is unique or under which conditions it is the global minimizer of the problem. The following theorems and definitions answer these questions.

Theorem 1.2.1. *Let $\mathscr{F} = \{x \in \mathbf{R}^n \colon l \leq x \leq u\}$ be feasible region of* (1.1) *and suppose that f is continuously differentiable on $\mathscr{F}$ with the gradient g.*

1. *If x^* is the solution of* (1.1) *then*

$$g(x^*)^T(x - x^*) \geq 0, \qquad x \in \mathscr{F}. \tag{1.2}$$

2. *If f is convex on $\mathscr{F}$ then, conversely, every $x^* \in \mathscr{F}$ satisfying* (1.2) *is global solution of the* (1.1).

3. *The problem* (1.1) *has an unique solution x^* whenever the objective function f is strictly convex on feasible region $\mathscr{F}$.*

Proof. The proof of Abstract first order optimality conditions for general constrained optimization problem [24, p, 97]. $\square$

Theorem 1.2.2 (General first order optimality conditions for bound constrained problem)**.** *Let x^* be the solution of* (1.1)*, then there exist vectors $\lambda, \mu \in \mathbb{R}^n$ such that*

$$\begin{cases} g(x^*) - \lambda + \mu = 0, \\ (x^* - l)^T\lambda = 0, \\ (u - x^*)^T\mu = 0, \\ \lambda, \mu \geq 0, \end{cases}$$

*where above conditions are called **Karush-Kuhn-Tucker** or **KKT** conditions and they are equivalent to*

$$\begin{cases} g_i(x^*) \geq 0, & \textit{if } x_i^* = l_i, \\ g_i(x^*) \leq 0, & \textit{if } x_i^* = u_i, \\ g_i(x^*) = 0, & \textit{if } l_i < x_i^* < u_i. \end{cases} \tag{1.3}$$

Proof. The proof of General first order optimality conditions [24, p, 105]. □

Note that the KKT conditions are necessary conditions for every solution of (1.1) but converse of above theorem does not hold unless, for example, the objective function of (1.1) is convex in a neighbourhood of the feasible point satisfying KKT conditions.

Definition 1.2.1. *A point $\bar{x} \in \mathscr{F}$ is said to be a **stationary point** for problem* (1.1) *if it satisfies KKT conditions, or equivalently, the conditions* (1.3) *hold for $\bar{x}$. Moreover, **strict complementarity** is said to hold at $\bar{x}$ if the strict inequality hold in the first and second implications of* (1.3)*.*

In general, convergence to a stationary point is all that we can expect of an algorithm for solving problem (1.1). In order to construct an algorithm for solving (1.1) we need to characterize the stationary point in a more convenient way. Hence, if we define the **reduced gradient** of a feasible point x as a vector $g_{red} = g_{red}(x)$ with components

$$(g_{red})_i : = \begin{cases} \min(0, g_i(x)) & \text{if } x_i = l_i, \\ \max(0, g_i(x)) & \text{if } x_i = u_i, \\ g_i(x) & \text{if } l_i < x_i < u_i, \end{cases} \tag{1.4}$$

a stationary point $\bar{x}$ can be characterized as

$$g_{red}(\bar{x}) = 0.$$

Definition 1.2.2. *We call a stationary point $\bar{x}$ of* (1.1) ***degenerate** whenever the strict complementarity fails to hold at $\bar{x}$. In other words, there are some variables $\bar{x}_i \in \{l_i, u_i\}$ with $g_i(\bar{x}) = 0$.*

Degeneracy is a property of stationary points that causes difficulties for some algorithms. When the strict complementarity does not hold at a solution of the problem, there are some constraints of the problem that are weakly active at the solution. In other words, all optimal Lagrange multipliers corresponding to these constraints are equal to zero. This fact makes it hard for an algorithm to find out whether these constraints are active at the solution. Particularly, in the case of active-set methods and gradient projection methods this indecisiveness may cause zigzagging, which means that the iterates of an algorithm move on and off these weakly active constraints successively.

Chapter 2

Proposed Algorithms for Bound Constrained Problems

2.1 Introduction

In this chapter we discuss several proposed algorithms for solving bound constrained optimization problems. We begin this chapter by giving an overview of the standard active-set method. This method consists of two main steps. First, identifying the set of optimal active bound constraints or reaching the face containing a stationary point of the problem. Second, exploring the face of feasible region by solving an unconstrained subproblem. Then, we study the gradient projection methods and their applications in the active-set methods. Indeed, all of the algorithms described in this chapter deal with the bound constraints by explicitly distinguishing between steps that search through the faces of the feasible region and steps that explore the current face of the feasible region. The last two sections of this chapter are concerned with algorithms that, at each iteration, approximate the objective function by a quadratic model whose Hessian matrix is computed by a limited memory method.

2.2 Standard Active-Set Method

In this section we describe the standard active-set method for solving quadratic bound constrained optimization problems. A quadratic bound constrained problem has the form

$$\begin{array}{ll} \text{minimize} & f(x) = x^T Bx + a^T x, \\ \text{subject to} & l \leq x \leq u, \end{array} \tag{2.1}$$

where B is symmetric and a belongs to $\mathbb{R}^n$. Problem (2.1) is a special form of the (1.1), In addition, we will see later that some effective algorithms for solving

(1.1) approximate the objective function of (1.1) with a quadratic model at each iteration. Therefore, they rely on the solution of (2.1) at each iteration.

The standard active-set method for solving (2.1) finds step from one iterate to next by solving an unconstrained quadratic subproblem in which some of variables are fixed in one of their bounds. We call this subset of variables *working set* and denote it at the kth iteration x_k with $W_k \subset \mathscr{A}(x_k)$, where $\mathscr{A}(x)$ is the set of active variables and defined by

$$\mathscr{A}(x) = \{i : x_i = l_i \vee x_i = u_i\}. \tag{2.2}$$

We call variables with indices in W_k *bound* variables and other variables are referred to as *free* variables. Now it is also helpful to define the *binding* set at x by

$$\mathscr{B}(x) = \{i : x_i = l_i \wedge g(x)_i \geq 0, \quad \vee \quad x_i = u_i \wedge g(x)_i \leq 0,\}, \tag{2.3}$$

where $g(x)$ is the gradient of the objective function.

We assume that the initial point x_0 is feasible and $W_0 \subset \mathscr{A}(x_0)$. Now suppose that we are at the kth iteration with working set W_k, in order to compute x_{k+1} we solve the following subproblem

$$\begin{array}{ll} \text{minimize} & f(x_k + p) \\ \text{subject to} & p_i = 0, \quad i \in W_k. \end{array} \tag{2.4}$$

The above problem is an unconstrained quadratic problem defined over the subspace of free variables corresponding to W_k. Once the (2.4) is solved, we have two cases:

1. Problem (2.4) has a global solution p^k. In this case if p^k is zero, then x_k is a global solution of the (2.1) over the working set W_k. Now assume that the vector p^k is nonzero. For computing x_{k+1} we need to know how far we can go along p_k while staying in the feasible region. If $x_k + p^k$ is feasible with respect to all bound constraints, we set $x_{k+1} = x_k + p^k$. Otherwise, we set

   $$x_{k+1} = x_k + \alpha_k p^k,$$

   where the step length α_k is defined by

   $$\alpha_k := \min\left(1, \min_{i \notin w_k \wedge p_i^k > 0} \frac{u_i - (x_k)_i}{p_i^k}, \min_{i \notin w_k \wedge p_i^k < 0} \frac{l_i - (x_k)_i}{p_i^k}\right).$$

   If $\alpha_k < 1$, that is, the step along direction p^k is blocked by at least one of the bound constraints whose indices are not in W_k. Consequently, the new working set W_{k+1} is updated by adding atleast one of the constraints which becomes active at x_{k+1} to W_k.

2. Problem (2.4) has no global solution. In this case it is possible to find the direction p^k such that the objective function of (2.4) is strictly decreasing and unbounded below along the ray $x_k + \alpha p^k$ for $\alpha > 0$. In this case either $f(x)$ is unbounded in the feasible region, or we set $x_{k+1} = x_k + \alpha_k p^k$, where the finite step length α_k is defined by

$$\alpha_k := \max\{\alpha \geq 0 : l \leq x_k + \alpha p^k \leq u\}.$$

This means at least one bound constraints becomes active at x_{k+1}.

In both of the cases once x_{k+1} is computed, W_{k+1} is updated by adding at least one of the blocking bound constraints to the W_k. Usually just one constraint is added to the working set at each iteration.

We continue to iterate in the manner that outlined above, adding constraints to the working set until we reach a point x_t for $t \geq 1$, which minimize the quadratic objective function of (2.1) over the working set W_t. Equivalently, we keep iterating until $p^t = 0$ is obtained as the global solution of the subproblem (2.4) with respect to W_t.

After the global solution x_t of subproblem (2.1) has been found we have two cases, either the working set W_t is a subset of $\mathscr{B}(x_t)$, or there exist some indices of W_t that do not belong to $\mathscr{B}(x_t)$. If W_t is a subset of $\mathscr{B}(x_t)$, then x_t is a stationary point of (2.1) and algorithm terminates with x_t. On the other hand, if there is a index in W_t that does not belong to $\mathscr{B}(x_t)$, the KKT conditions are not satisfied for x_t, in particular, a Lagrange multiplier corresponding to an index in W_t is negative. Furthermore, by dropping this index from the working set the objective function of (2.1) may be decreased. Therefore, we remove this index from the working set and solve the subproblem (2.4) with a new working set. It can be shown by a short computation that it is possible to find x_{t+1} with respect to the new working set such that

$$f(x_{t+1}) < f(x_t).$$

If the quadratic function $f(x)$ is bounded below on the feasible region, it is not difficult to show that the active-set method converges to a stationary point in a finite number of iterations. Our argument for verifying this result is quite instructive. At each iteration either we reach a global minimizer of the subproblem or we increase the size of the working set. Because the cardinality of working set is always bounded by n, the global solution of (2.4) will be obtained after at most n iterations. On the other hand, if the global solution x_t of the subproblem is determined with respect to some working set W_k, then x_t is a global solution of the $f(x)$ subject to the *face* of the feasible region which is defined by

$$\{x \in \mathbb{R}^n : l \leq x \leq u, x_i \in \{l_i, u_i\}, i \in W_t\}. \tag{2.5}$$

Since $f(x_{t+1}) < f(x_t)$, it is impossible for the algorithm to arrive in this face again in an iteration x_r with $r > t$. Moreover, the number of possible faces is finite, thus the active set method terminates in a finite number of iterations.

Finally, we summarize the active-set method in two main steps:

Global Step: If x_t is a global solution of the $f(x)$ subject to the face (2.5), then find a feasible point x_{t+1} such that $f(x_{t+1}) < f(x_t)$. Indeed, in this step we move from a face to another until we land on the face that contains a global solution of (2.1).

Local Step: Otherwise find a feasible point x_{t+1} such that $f(x_{t+1}) \leq f(x_t)$ and $W_t \subset W_{t+1}$. If $W_t = W_{t+1}$ then x_{t+1} must be a global solution of $f(x)$ subject to the face (2.5). In this step we explore the face of the feasible region which is defined by the current working set.

As it has been mentioned, the standard active-set method drops only one constraint from the current working set in the global step and adds only one constraint to the current working set in the local step. That is, at each step of the standard active-set method, the dimension of the subspace of bound variables is changed only by one. This fact implies that if there are n_1 active constraints on x_0 and n_2 active constraints on the solution of (2.1), we need at least $|n_2 - n_1|$ iterations to reach the solution of (2.1). This may be serious drawback in the case of large scale problems. This observation motivated many authors, specially BERTSEKAS [2], to use gradient projection method for solving bound constrained problems. He has established that the set of active constraints at the nondegenerate stationary point of bound constrained problem (1.1) is identified in a finite number of iterations by using gradient projection method. We will see later how the gradient projection method and active-set method are combined with each others in order to construct an efficient algorithm for solving bound constrained problems.

2.3 The Gradient Projection Methods

In this section we begin with an overview of the gradient projection methods for solving general optimization problems with the form

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in \mathscr{F} \end{array} \tag{2.6}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and $\mathscr{F}$ is a convex feasible region.

Given an iterate x_k, the next iterate x_{k+1} is computed by the gradient projection method as follows

$$x_{k+1} = P_{\mathscr{F}}[x_k - \alpha_k g(x_k)].$$

Where $P_{\mathscr{F}}[x]$ is the orthogonal projection on the feasible region $\mathscr{F}$, $g(x)$ is the gradient of objective function, and the step length $\alpha_k > 0$ is chosen so that $f(x_{k+1}) < f(x)$. The projection part plays important rule in the gradient projection method and it must be computed in a fairly simple way. Otherwise, it does not make practical sense to use the gradient projection method. In general the orthogonal projection on the feasible region has the form

$$P_{\mathscr{F}}[x] = arg \min_{y \in \mathscr{F}} \|x - y\|. \tag{2.7}$$

The computation of (2.7) for general constrained optimization problems can be expensive but in the case of bound constrained problems with $\mathscr{F} = \{x \in \mathbb{R}^n : l \leq x \leq u\}$, it requires only order of n operations and it is defined by

$$(P_{\mathscr{F}}[x])_i = (P[x,l,u])_i = \begin{cases} l_i & \text{if } x_i \leq l_i, \\ u_i & \text{if } x_i \geq u_i, \\ x_i & \text{if } l_i < x_i < u_i. \end{cases} \tag{2.8}$$

It can be shown that by choosing an appropriate step length α_k, the gradient projection method is able to identify the set of active constraints at the nondegenerate stationary point in finite number of iterations. This is a consequence of the fact that if $\bar{x}$ is a nondegenerate stationary point of the problem, then there exists a neighbourhood $\mathscr{N}(\bar{x})$ of $\bar{x}$ such that $\mathscr{A}(x_{k+1}) = \mathscr{A}(P_{\mathscr{F}}[x_k - \alpha_k g(x_k)])$ provided $x_k \in \mathscr{N}(\bar{x})$. This identification ability of the gradient projection method is valid according to [12, 2, 8]. Now we describe some methods for computing the step length α_k, in which the gradient projection method has this identification ability.

2.3.1 Armijo Rule Along the Projection Arc

This procedure was proposed by BERTSEKAS [3, 4] and its concept is really simple. First we chose the fixed scalers $\bar{\alpha}$, β, and $\sigma > 0$, with $\bar{\alpha} > 0$, $\beta \in (0,1)$, and $\sigma \in (0,1)$. Then we define the following piecewise linear path

$$x_k(\alpha) := P_{\mathscr{F}}[x_k - \alpha g(x_k)], \quad \alpha > 0, \tag{2.9}$$

where P is the projection into the feasible region $\mathscr{F}$ and g represents the gradient of the objective function. Finally the step length α_k is determined by proportionally reduction of the $\bar{\alpha}$ with a factor $\beta \in (0,1)$ until the Armijo condition is satisfied. In other words, we set $\alpha_k = \beta^{r_k}\bar{\alpha}$, where r_k is the first nonnegative integer r for which the following conditions holds.

$$f(x_k) - f(x_k(\beta^r\bar{\alpha})) \geq \sigma g(x_k)^T(x_k - x_k(\beta^r\bar{\alpha})). \tag{2.10}$$

It can be shown that the above step-size rules is well defined, That is, after a finite number of trials based on the (2.10) the steplength α_k will be found. Furthermore, the gradient projection method with the above linesearch procedure is able to identify the set of active constraints at the nondegenerate solution of the (2.6) in finite iterations.

2.3.2 Projected Search

The projected search is a procedure for finding α_k for the case in which $f(x) = x^T Bx + a^T x$ and the feasible region $\mathscr{F}$ consisting of bound constraints. This procedure was proposed by MORÉ and TORALDO [21, 22]. In this procedure the computation of α_k relies on the function defined by

$$\phi_k(\alpha) := f(P_{\mathscr{F}}[x_k + \alpha p_k]), \tag{2.11}$$

where P is the projection (2.8) into the bound constraints and p_k is a search direction with $\phi_k'(0) < 0$. We determine a step-length $\alpha_k > 0$ so that the sufficient decrease condition holds for $\phi_k(\alpha)$. The sufficient decrease condition is defined by

$$\phi_k(\alpha_k) \leq \phi_k(0) + \sigma g(x_k)^T(P_{\mathscr{F}}[x_k + \alpha p_k] - x_k), \tag{2.12}$$

where $\sigma \in (0, \frac{1}{2})$ is fixed. Since the path $P_{\mathscr{F}}[x_k + \alpha p_k]$ is a linear function on any interval on which the set of active constraints at $P_{\mathscr{F}}[x_k + \alpha p_k]$ is unchanged, $\phi_k(\alpha)$ is a continuous piecewise quadratic function. For defining Breakpoints for $\phi_k(\alpha)$ we compute

$$0 = t_1 < t_2 < \ldots < t_m < t_{m+1} = +\infty$$

so that $\mathscr{A}(P_{\mathscr{F}}[x_k + \alpha p_k])$ is unchanged on the intervals (t_i, t_{i+1}) for $i = 0, \ldots, m$. If we allow infinite value for l_i or u_i, it is possible to have $m = 0$ in the computations of breakpoints. Moreover, the first breakpoint has a interesting feature, namely it can be shown that there is $\alpha \in [0, t_1]$ that satisfies the sufficient decrease condition (2.12) and for all $\alpha \in [0, t_1]$ we have also

$$\phi_k(\alpha) = f(P_{\mathscr{F}}[x_k + \alpha p_k]) = f(x_k + \alpha \hat{p}_k), \tag{2.13}$$

where $\hat{p}_k$ is given from p_k by taking the components whose indices correspond to the free variables at x_k. Similarly, we define matrix $\hat{B}_k$ which is obtained from

B by choosing the rows and columns whose indices are corresponding to the free variables. To put it another way, if Z_k denotes the matrix whose columns span the subspace of free variables at x_k, we have

$$\hat{p}_k = Z_k^T p_k, \quad \hat{B}_k = Z_k^T B Z_k.$$

The projected search method generate a positive decreasing sequence $\{\alpha_k^r\}_r$ of trial values until a trial value is found which satisfies (2.12). Note that this process will terminate after a finite number of steps, since there exists a trial value in the interval $[0,t_1]$ which fulfils (2.12).

Now for choosing the initial trial value α_k^0 we have two cases depending on the behaviour of ϕ_k on the interval $[0,t_1]$:

1. ϕ_k is strictly convex on the $[0,t_1]$. In this case the initial trial value is given by the first minimizer of the quadratic function $\phi_k(\alpha)$ in $[0,t_1]$ and computed by
$$\alpha_k^0 = \frac{-\phi_k'(0)}{\phi_k''(0)} = \frac{\|\hat{p}_k\|^2}{\hat{p}_k^T \hat{B}_k \hat{p}_k},$$
where $\hat{p}_k^T \hat{B}_k \hat{p}_k > 0$. Note that if $\alpha_k^0 < t_1$, the procedure terminates with the acceptable step length $\alpha_k = \alpha_k^0$.

2. ϕ_k is not strictly convex on $[0,t_1]$ and also $\hat{p}_k^T \hat{B}_k \hat{p}_k \leq 0$. In this case the quadratic function $\phi_k(\alpha)$ is on $[0,t_1]$ strictly decreasing and unbounded below. Consequently, it is better to choose α_k^0 desirable large. Hence, we set
$$\alpha_k^0 = t_m.$$

Now suppose that the initial trial value α_k^0 is given. If it satisfies (2.12), we terminate with $\alpha_k = \alpha_k^0$. Otherwise, given the current trial value α_k^r with $r > 1$, for which the (2.12) is not satisfied. For computing the new trial value, we determine the minimizer of the quadratic approximation that interpolates the pieces of information $\phi_k(0)$, $\phi_k'(0)$, and $\phi_k(\alpha_k^r)$. Then the new trial value α_k^{r+1} is given by

$$\max(t_1, \mathrm{mid}(0.01\alpha_k^r, \hat{\alpha}, 0.5\alpha_k^r)), \tag{2.14}$$

where $\mathrm{mid}(0.01\alpha_k^r, \hat{\alpha}, 0.5\alpha_k^r)$ is the middle element of the set $\{0.01\alpha_k^r, \hat{\alpha}, 0.5\alpha_k^r\}$. The above process is continued until a trial value satisfying (2.12) is found. The finite termination is also guaranteed. Since, if either $\hat{p}_k^T \hat{B}_k \hat{p}_k \leq 0$ or $\alpha_k^0 > t_1$, t_1 satisfies the sufficient condition with $\sigma \leq 0.5$.

Note that for the projection gradient method, we use $-g(x_k)$ as the search direction p_k at each iteration.

2.4 Approaches for Quadratic Bound Constrained problems

In this section we will deal with the algorithms for solving bound constrained quadratic problems (2.1). As we have mentioned, the standard active-set method has the major disadvantage that at each iteration the cardinality of the working set is changed by dropping or adding only one constraint. On the other hand, as we have discussed in the previous section, the gradient projection method is able to identify the set of optimal active set in finite number of iterations. This facts motivated many authors, for example, MORÉ and TORALDO [21, 22] to propose algorithms that use the gradient projection method in an efficiently manner in the active-set method schema. Particularly, these algorithms take advantage of the identification property of the gradient projection method to reach the face containing the solution quickly, and to explore each face of the feasible region, they solve an unconstrained optimization subproblem (2.4).

2.4.1 BCQP

This algorithm is proposed by MORÉ and TORALDO [21] to solve the quadratic bound constrained problem (2.1) and it consists of two main steps:

Global Step: Set $x_k = u_0$ and generate the iterations $u_0, u_1, ..., u_s$ by the gradient projection method. If for some $j \in \{1, ..., s\}$ the condition $\mathscr{A}(u_j) = \mathscr{A}(u_{j-1})$ is satisfied then we set $x_{k+1} = u_j$. Otherwise, we set $x_{k+1} = u_s$.

Local Step: Otherwise find a feasible x_{k+1} such that $f(x_{k+1}) \leq f(x_k)$ and $W_k \subset W_{k+1}$. If $W_k = W_{k+1}$ then x_{k+1} must be a global solution of $f(x)$ subject to the face $\{x \in \mathbb{R}^n : l \leq x \leq u, x_i \in \{l_i, u_i\}, i \in W_k\}$.

As we can see, the only deference of the above algorithm and the standard active-method is in the global step. Indeed, the gradient projection method is used in the global step to modify the standard active-set method in two aspects. Firstly, the cardinality of the working set is changed by dropping and adding many constraints. Secondly, the algorithm tends to find the optimal active constraints more quickly.

It should be noted that, in the global step the gradient projection method is performed with the help of projected search method 2.3.2. That is, a sequence $\{u_j\}$ is computed by

$$u_{j+1} = P_{\mathscr{F}}[u_j - \alpha_j g(u_j)],$$

where P is the projection (2.8) in to the bound constrained feasible region $\mathscr{F}$, the step length α_j is determined by the projected search strategy 2.3.2, and g represent the gradient of quadratic objective function. Moreover, Since in the case of degenerate solutions, the optimal active set may not settle down, in addition, in the case of nondegenerate solutions, it is only assured that the set of active constraints settles down in a neighbourhood of the solution, it is better to impose a bound s on the number of gradient projection iterations.

The local step of the above method is exactly the same as the standard active set method. That is, in this step we determine the solution p^k of the following unconstrained quadratic subproblem

$$\begin{array}{ll} \text{minimize} & f(x_k+p) \\ \text{subject to} & p_i=0, \quad i\in W_k. \end{array} \tag{2.15}$$

and we set $x_{k+1}=x_k+\alpha_k p^k$, where $f(x)=x^TBx+a^Tx$ and α_k is computed by

$$\alpha_k := \min\left(1, \min_{i\notin w_k \wedge p_i^k>0} \frac{u_i-(x_k)_i}{p_i^k}, \min_{i\notin w_k \wedge p_i^k<0} \frac{l_i-(x_k)_i}{p_i^k}\right).$$

Now suppose that the working set W_k is given and $j_1, j_2, \ldots, j_{m_k}$ represent the indices of free variables with respect to W_k, i.e. the variables that do not belong to W_k. And let $Z_k\in\mathbb{R}^{n\times m_k}$ be the matrix whose columns span the subspace of free variables. Then we can transform the above subproblem (2.15) to the following subproblem.

$$\min_{v\in\mathbb{R}^{m_k}} f_k(v), \tag{2.16}$$

where

$$f_k(v)=f(x_k+Z_kv)-f(x_k)=\frac{1}{2}v^TB_kv+\hat{g}_k^Tv,$$
$$\hat{B}_k=Z_k^TBZ_k, \quad \hat{g}_k=Z_k^Tg(x_k).$$

Now we determine the solution p^k of (2.4) with help of the unconstrained subproblem (2.16), In other words, we choose a vector $v_k\in\mathbb{R}^{m_k}$ and set $p^k=Z_kv_k$. For choosing v_k we have the following cases:

1. $\hat{B}_k$ is positive definite. In this case we set $v_k=-\hat{B}_k^{-1}\hat{g}$, since it is the global solution of $f_k(v)$.

2. $\hat{B}_k$ is not positive definite. In this case v_k is determined such that the following conditions hold

$$v_k^T\hat{B}_kv_k\le 0, \quad \hat{g}_k^Tv_k\le 0, \quad \min\{v_k^T\hat{B}_kv_k, \hat{g}_k^Tv_k\}<0 \tag{2.17}$$

This is possible whenever $\hat{B}_k$ is either not positive semidefinite or regular, or $\hat{g}_k \notin Im(\hat{B}_k)$

The computation of v_k is based on the Cholesky factorization of $\hat{B}_k$ and we need almost one factorization for all matrices. If $\hat{B}_k$ is positive definite then the Cholesky factorization of the matrix exists and is used for computing v_k. Otherwise, while computing of Cholesky factorization we try to find the largest positive definite principal minor of the matrix. Let this principal minor has the order l and be represented by C_k, then the principal minor of order $l+1$ of $\hat{B}_k$ has the following form

$$\begin{pmatrix} C_k & w_k \\ w_k^T & \theta_k \end{pmatrix} \tag{2.18}$$

Due to positive definiteness, C_k has a Cholesky factorization $R_k^T R_k$. Now we claim that

$$\theta_k \leq w_k^T C_k^{-1} w_k. \tag{2.19}$$

Suppose on contrary that (2.19) does not hold, then it is easy to verify that the Cholesky factorization of (2.18) is

$$\begin{pmatrix} R_k^T & 0 \\ q_k^T & \eta_k \end{pmatrix} \begin{pmatrix} R_k & q_k \\ 0 & \eta_k \end{pmatrix},$$

where

$$R_k^T w_k = q_k, \quad \eta_k = (\theta_k - \|q_k\|^2)^{\frac{1}{2}} = (\theta_k - w_k^T C_k^{-1} w_k)^{\frac{1}{2}}.$$

But this is contradiction with the fact that (2.18) is not positive definite. Therefore (2.19) holds. Consequently, if we define the vector $v_k \in \mathbb{R}^{m_k}$ by

$$v_k = \pm(C_k^{-1} w_k, -1, 0, \ldots, 0),^T$$

where the sign of v_k is chosen such that $v^T \hat{g}_k \leq 0$, we have

$$v_k^T \hat{B}_k v_k = \theta_k - w_k^T C_k^{-1} w_k \leq 0.$$

This shows that (2.17) is satisfied, provided either $v_k^T \hat{g}_k \neq 0$ or $v_k^T \hat{B}_k v_k \neq 0$.

It should be noted that if for v_k the condition (2.17) is satisfied, then the $p^k = Z_k v_k$ defines a direction such that $f(x_k + \alpha p^k)$ is strictly decreasing and unbounded below for $\alpha > 0$. Hence, similar to the standard active-set method, either f is unbounded below on the feasible region or a finite α_k corresponding to a blocking constraint is computed.

The BCQP algorithm terminates at the iterate x_k if one of the following cases occurs:

1. The matrix $\hat{B}_k$ is positive semidefinite, $\hat{g}_k = 0$, and $W_k \subset \mathscr{B}(x_k)$. In this case the vector $p^k = 0$ is the global solution of the unconstrained quadratic subproblem (2.15) with respect to the working set W_k. In addition, because W_k is a subset of $\mathscr{B}(x_k)$, iterate x_k is a stationary point of the problem.

2. For all $\alpha > 0$, the ray $x_k + \alpha p^k$ is feasible and the function $f(x_k + \alpha p^k)$ is decreasing and unbounded below.

3. For all $\alpha > 0$, the ray $x_k + \alpha p^k$ is feasible and the function $f(x_k + \alpha p^k)$ is constant and $p^k = 0$. In this case we have two possible situations, either $f(x)$ is unbounded below on the feasible region, or there exists a arbitrary small perturbation on the $\hat{g}_k$ or $\hat{B}_k$ which leads to the fact that f is unbounded on the feasible region.

Moreover, in [21, p. 388] it has been shown that if the quadratic objective function $f : \mathbb{R}^n \to \mathbb{R}$ is bounded from the below on the feasible region, then the BCQP algorithm terminate at a stationary point in finite iteration steps.

It should be mentioned that in the global step of the BCQP, only one constraint is added to the working set. This can be serious disadvantage for large scale problems, hence we aim to construct an algorithm that is able to add many variable to the working set in the global step.

2.4.2 GPCG

In this section we present an algorithm to find the solution of (2.1), where the quadratic function f is strictly convex, i.e. the matrix B in (2.1) is positive definite, and the number of variable is assumed to be large. This algorithm was developed by MORÉ and TORALDO [22]. Similar to the standard active-method and BCQP, this method generates a sequence of iterations x_k that terminates at a solution of (2.1) in finite number of iterations. Moreover, the termination typically is obtain by solving a sequence of unconstrained quadratic subproblems of the form

$$\begin{array}{ll} \text{minimize} & f(x_k + p) \\ \text{subject to} & p_i = 0, \quad i \in W_k \end{array} \tag{2.20}$$

with a working set W_k, and changing the size of working set by dropping and adding constraints at each iteration. In particular, this algorithm performs the gradient projection method until either an adequate working set W_k is identified or the gradient projection method is no longer able to deliver an acceptable improvement. Furthermore, the conjugate gradient method is applied to find approximate solution of the above unconstrained subproblem with respect to the current W_k.

As we know from the previous section, the above subproblem can be transformed to the following quadratic unconstrained problem

$$\min_{v\in\mathbb{R}^{m_k}} f_k(v) = \frac{1}{2}v^T\hat{B}_k v + \hat{g}_k^T v, \tag{2.21}$$

where the matrix $\hat{B}$ consists of the columns and rows of B which correspond to free variables at x_k, similarly, $\hat{g}_k$ is defined by taking the elements of the gradient of f at x_k which correspond to free variables, and m_k denotes the number of free variable at x_k.

Since the matrix $\hat{B}$ is positive definite, the conjugate gradient method can be used to solve (2.21). Suppose that the initial point v_0 is given, the conjugate gradient method generates a sequence of iterations $v_0, v_1, \dots$ which terminates at the solution of (2.21) in at most m_k iterations. In GPCG algorithm the conjugate gradient method is performed until an iterate v_j is obtained that satisfies the following condition

$$f_k(v_{j-1}) - f_k(v_j) \le \eta_1 \max_{1\le s<j}(f_k(v_{s-1}) - f_k(v_s)) \tag{2.22}$$

for a fixed constant $\eta_1 > 0$. Then the approximate solution of (2.20) with respect to W_k is obtained by $p^k = Z_k v_{j_k}$, where Z_k is a matrix whose columns span the subspace of free variable at x_k, and j_k is a first index for which the condition (2.22) holds. In fact, the condition (2.22) reveals the situation in which the conjugate gradient method does not make reasonable progress.

As we have seen in previous sections, in the standard active-set method and BCQP the new iterate x_{k+1} is defined by

$$x_{k+1} = x_k + \alpha'_k p^k,$$

where

$$\alpha'_k := \min\left(1, \min_{i\notin w_k \wedge p_i^k>0} \frac{u_i - (x_k)_i}{p_i^k}, \min_{i\notin w_k \wedge p_i^k<0} \frac{l_i - (x_k)_i}{p_i^k}\right),$$

and p^k is the minimizer of (2.20). Indeed, in this strategy only one constraint is added to the working set, and it can make algorithm inefficient. Therefore, it is desirable to use a strategy for finding α_k, in which more than one constraint can be added to the working set at each iteration. For this reason, in CGQP the new iterate x_{k+1} is computed by

$$x_{k+1} = P_{\mathscr{F}}[x_k + \alpha_k p^k], \tag{2.23}$$

where $P_{\mathscr{F}}$ is the orthogonal projection in to the bound constrained feasible region $\mathscr{F}$, and α_k is determined by the **projected search** strategy, which has been described in the subsection 2.3.2. Note that if $\alpha_k > \alpha'_k$, then more than one constraint might be added to the working set.

Now assume that a new iteration x_{k+1} has been computed by the conjugate gradient method. If x_{k+1} is in the face containing the solution of problem, this face will be explored further by the conjugate gradient method. This decision is made in terms of verifying the following condition

$$\mathscr{A}(x_{k+1}) = \mathscr{B}(x_{k+1}), \tag{2.24}$$

where $\mathscr{A}(x)$, $\mathscr{B}(x)$ denote, respectively, the set of active and binding variables at x. Note that, if x_{k+1} is in the face containing the solution, the condition (2.24) holds.

Remark 2.4.1. *Note that if* (2.24) *holds, it does not necessarily mean that x_{k+1} is in the face that contains the solution of the problem. Nevertheless, if x_{k+1} is not in this face, because of the finite termination property of the conjugate gradient method, an iterate x_r with $r > k+1$ is eventually generated which violates the condition $\mathscr{A}(x_r) = \mathscr{B}(x_r)$.*

Once a face has been explored by conjugate gradient method, the gradient projection method is used to search through the different faces by generating a sequence $\{u_j\}$ which is defined by

$$u_{j+1} = P_{\mathscr{F}}[u_j - \alpha_j g(u_j)],$$

where P is the projection (2.8) in to the bound constrained feasible region $\mathscr{F}$, the step length α_j is determined by the projected search strategy 2.3.2, and g denotes the gradient of the quadratic objective function. The gradient projection method is used to choose a new face as follows: assume that we are at the kth iteration with x_k, we set $u_0 = x_k$ and generate iterations $u_0, u_1, u_3, \ldots$ until for some fixed η_2 one of the following conditions

$$\mathscr{A}(u_j) = \mathscr{A}(u_{j-1}), \tag{2.25}$$

$$f(u_{j-1}) - f(u_j) \le \eta_2 \max_{1 \le s < j} (f_k(u_{s-1}) - f_k(u_s)) \tag{2.26}$$

is satisfied. The justification of the condition (2.25) is based on the result which says, in the case of nondegenerate problems there exists a neighbourhood of the solution such that the condition (2.25) is satisfied provided that x_k belongs to this neighbourhood. Furthermore, the condition (2.26) prevents the gradient projection method from making deficient progress.

The above description can be summarized in to the following steps:

Global step: Generate a sequence $u_0, u_1, u_2, \ldots$ with $u_0 = x_k$ by the gradient projection method. Set $x_{k+1} = u_{j_k}$, where j_k is the first index j which fulfils either (2.25) or (2.26).

Local step: Generate a sequence $v_0, v_1, v_2, \ldots$ by the conjugate gradient method with $v_0 = 0$ in order to solve (2.21) approximately. Then set $p^k = Z_k v_{j_k}$, where j_k is the first index that fulfils (2.22). Use the projected search strategy to determine the step length α_k and define x_{k+1} by (2.23). If (2.24) is satisfied, continue the conjugate gradient method.

It is claimed in [22] that if the termination of GPCG is occurred in a finite number of iterations, then it terminates at the solution of the problem. If the termination occurs in the global step, duo to standard result of the gradient projection method it terminates at the solution of the problem. Moreover, In local step the termination can only occur if the conjugate gradient method generates an iterate x_s with $\hat{g}_s = 0$ and $\mathscr{A}(x_s) = \mathscr{B}(x_s)$. Which implies that $g_{red}(x_s) = \hat{g}_s = 0$, so x_s is the solution.

The main theoretical result for convergence of GPCG is based on the following theorem which is represented in [22, p, 101]:

Theorem 2.4.1. *Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is a strictly convex quadratic function. If x_k is the sequence generated by GPCG for solving the problem* (2.1)*, then either x_k terminates at the solution x^* of the problem* (2.1) *in a finite number of iterations, or x_k converges to the solution x^*.*

2.5 A Trust Region Algorithm for Bound Constrained Problems

In this section we describe a method of trust region type for solving problem (1.1), where the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is a arbitrary function which is sufficiently smooth. We have no convexity assumption on f. Recall that the feasible region is the set of all $x \in \mathbb{R}^n$ for which $l \le x \le u$.

This method proposed by CONN, GOULD and TOINT [10, 9] is a combination of the active set method strategy and trust region method. That is, at each iteration of this algorithm, the objective function is approximated by a quadratic model within a region surrounding the current iteration. Then the algorithm chooses a new feasible point in that region for the next iterate such that it gives a sufficient decrease in the value of the quadratic model. If the value of objective function at this point decreases sufficiently as it is predicted by the quadratic model, then the computed feasible point within the region is accepted as the next iterate and the algorithm expands the region. Otherwise, the algorithm rejects this point and shrinks the region. Moreover, the set of active constraints changes in this method rapidly. The general convergence theory of this method is presented in [9].

2.5.1 Outline of the Algorithm

Before describing the method, we define the active set at point x with respect to vectors l and u as the set of all indices $i \in \{1,\dots,n\}$ for which either $x_i \le l_i$ or $x_i \ge u_i$,. We denote this set with $\mathscr{A}(x) = \mathscr{A}(x,l,u)$. Furthermore, $P[x,l,u]$ represents the projection on to the set $\{x \in \mathbb{R}^n \colon l \le x \le u\}$ computed by

$$(P[x,l,u])_i = \begin{cases} l_i & \text{if } x_i \le l_i, \\ u_i & \text{if } x_i \ge u_i, \\ x_i & \text{if } l_i < x_i < u_i. \end{cases}$$

Now suppose that we are at the kth iteration with x_k, the gradient g_k of the objective function f at x_k. The objective function f is approximated at x_k by a quadratic model m_k whose gradient coincides with g_k, and whose symmetric Hessian matrix B_k is computed by a limited memory quasi Newton method. In addition, we need a scaler Δ_k as the radius of the trust region at the kth iteration, i.e. a bound on the displacement around x_k. we believe that in this region the behaviour of the quadratic model

$$m_k(x+p) = f(x_k) + g_k^T p + \frac{1}{2} p^T B_k p, \qquad \text{s.t. } \|p\| \le \Delta_k \tag{2.27}$$

is similar to the objective function. Then a trial point $\bar{x}_{k+1} = x_k + p_k$ for next iteration is determined by finding an approximation to the solution of the following trust region problem

$$\begin{array}{ll} \text{minimize} & m_k(x) \\ \text{subject to} & l \le x \le u, \\ & \|x - x_k\| \le \Delta_k, \end{array} \tag{2.28}$$

where $\|.\|$ is the infinity norm. The acceptance of the trial point $\bar{x}_{k+1}$ as the new iterate and the modification of the trust region radius Δ_k are based on the agreement of the quadratic model m_k with the objective function f at the iteration x_k. That is, we compute the following ratio

$$\rho_k := \frac{f(x_k) - f(x_k + p_k)}{m_k(x_k) - m_k(x_k + p_k)}, \tag{2.29}$$

where the numerator and dominator of (2.29) are called *actual* and *predicted reduction*, respectively. Then we set

$$x_{k+1} = \begin{cases} x_k + p_k & \text{if } \rho_k > \mu, \\ x_k & \text{if } \rho_k \leq \mu, \end{cases}$$

and modify the radius Δ_k as follows

$$\Delta_{k+1} = \begin{cases} \gamma_1 \Delta_k & \text{if } \rho_k \leq \mu, \\ \Delta_k & \text{if } \mu < \rho_k < \eta, \\ \gamma_2 \Delta_k & \text{if } \rho_k \geq \eta, \end{cases} \tag{2.30}$$

where $0 < \gamma_1 < 1 < \gamma_2$, μ and η are fixed numbers. Now we describe how the solution of (2.28) is approximated at each iteration. Since $\|.\|$ in (2.28) is the infinity norm, the shape of the trust region is like a box. Hence, we can transform (2.28) to the following bound constrained problem

$$\begin{array}{ll} \text{minimize} & m_k(x) \\ \text{subject to} & l_k \leq x \leq u_k, \end{array} \tag{2.31}$$

where

$$\begin{aligned} (l_k)_i &:= \max(l_i, (x_k)_i - \Delta_k), \\ (u_k)_i &:= \min(u_i, (x_k)_i + \Delta_k), \end{aligned} \tag{2.32}$$

for all $i \in \{1, \ldots, n\}$.

According to [9], in order to fulfil the global convergence theory, it is required to find a feasible point of (2.31) at which the value of the quadratic model m_k is not greater than its value at the generalized Cauchy point (CP). Where the generalized Cauchy point is defined as the first local minimizer of the following univariate, piecewise quadratic function

$$q_k(t) := m_k(P[x_k - tg_k, l_k, u_k]). \tag{2.33}$$

In other words, the CP is the first minimizer of the quadratic model m_k along the piecewise linear path $P[x_k - tg_k, l_k, u_k]$ which is defined by projecting the steepest descent direction into the region

$$\{x \in \mathbb{R}^n : l_k \leq x \leq u_k\}.$$

Note that the generalized Cauchy point is a kind of the gradient projection methods that we have described in section (2.3). Therefore, it allows the algorithm to add and drop many indices from the active set at each iteration and it is also able to find optimal active set in a finite number of iterations, which is desirable for the case of large scale problems.

In order to obtain a fast asymptotic rate of convergence for the method, we need to find a better minimizer of (2.31) than the generalized Cauchy point. Hence, given x_k^{cp} the generalized Cauchy point for (2.31), and $\mathscr{A}(x_k^{cp}) = \mathscr{A}(x_k^{cp}, l_k, u_k)$ the set of active variables at the Cauchy point, we solve the following problem approximately

$$\begin{array}{lll} \text{minimize} & m_k(x) & \\ \text{subject to} & (l_k)_i \le x_i \le (u_k)_i, & i \notin \mathscr{A}(x_k^{cp}), \\ & x_i = (x_k^{cp})_i, & i \in \mathscr{A}(x_k^{cp}). \end{array} \tag{2.34}$$

For solving (2.34), Firstly we ignore the bound constraints on the free variables and treat the problem as an unconstrained optimization problem in the subspace of free variables which correspond to the set of indices $\{i \in \{1, \dots, n\} : i \notin \mathscr{A}(x_k^{cp})\}$. Then the conjugate gradient method is applied with the initial point $x = x_k^{cp}$ to the problem

$$\begin{array}{ll} \text{minimize} & m_k(x) \\ \text{subject to} & x_i = (x_k^{cp})_i, \quad i \in \mathscr{A}(x_k^{cp}). \end{array}$$

We terminate the conjugate gradient method at a new trial point $\bar{x}_{k+1}$, if one of the following conditions holds:

1. The residual in the conjugate gradient method is small enough.
2. One or more of the variables, whose index is in not contained in $\mathscr{A}(x_k^{cp})$ violates one of the bounds.
3. The conjugate gradient does not make efficient progress.

We describe the conjugate gradient method in more details later.

2.5.2 The Generalized Cauchy Point

In this section we outline an algorithm for computing the CP as the first local minimizer of the quadratic model m_k along the piece wise linear path

$$x(t) = P[x_k - tg_k, l_k, u_k], \quad \text{for } t \ge 0, \tag{2.35}$$

which is obtained by projecting the points along steepest descent direction in to the bound constraints. For convenience, we define $\hat{x} = x_k$ and drop the index k of the outer iteration throughout this section. Therefore, g, l, u and B represent, respectively, g_k, l_k, u_k and B_k. Subscripts are used to denote the components of a vector.

For deriving an explicit expression for piece wise linear path (2.35), we define the breakpoints in each coordinate by

$$t_i := \begin{cases} \frac{\hat{x}_i - u_i}{g_i} & \text{if } g_i < 0 \\ \frac{\hat{x}_i - l_i}{g_i} & \text{if } g_i > 0 \\ \infty & \text{otherwise,} \end{cases} \tag{2.36}$$

and we sort breakpoints t_i for $i = 1, \dots, n$ in an ascending order to obtain the ordered set $\{\bar{t}_j : \bar{t}_j \leq \bar{t}_{j+1}, j = 1, \dots, n\}$. Then the piecewise linear search path $x(t)$ can be expressed by

$$x_i(t) = \begin{cases} \hat{x}_i - t g_i & \text{if } t \leq t_i \\ \hat{x}_i - t_i g_i & \text{if } t > t_i, \end{cases} \tag{2.37}$$

which is linear on each interval $[\bar{t}_j, \bar{t}_{j+1}]$ with $j = 1, \dots, n-1$. For finding the CP we examine the intervals $[0, \bar{t}_1], [\bar{t}_1, \bar{t}_2], [\bar{t}_2, \bar{t}_3], \dots$ in turn until the one that contains the CP is located.

Assume that we have examined the intervals $[\bar{t}_{k-1}, \bar{t}_k]$ for $k = 1, \dots, j$ and determined that the local minimizer lies at some value $t \geq \bar{t}_j$. Now we are examining the interval $[\bar{t}_j, \bar{t}_{j+1}]$. We define the jth breakpoint by

$$x^j = x(\bar{t}_j).$$

Hence, on the interval $[\bar{t}_j, \bar{t}_{j+1}]$ we have

$$x(t) = x^j + \Delta t d^j, \tag{2.38}$$

where

$$\Delta t = t - \bar{t}_j, \qquad \Delta t \in [0, \bar{t}_{j+1} - \bar{t}_j], \tag{2.39}$$

and

$$d_i^j = \begin{cases} -g_i & \text{if } \bar{t}_j < t_i \\ 0 & \text{otherwise.} \end{cases} \tag{2.40}$$

In order to compute the generalized Cauchy point, we need to investigate the behaviour of the quadratic model

$$m(x) = f(\hat{x}) + g^T(x - \hat{x}) + \frac{1}{2}(x - \hat{x})^T B (x - \hat{x}) \tag{2.41}$$

for points lying on the interval $[\bar{t}_j, \bar{t}_{j+1}]$. By substituting (2.38) in (2.41), we can express the quadratic model m on the line segment $[x(\bar{t}_j), x(\bar{t}_{j+1})]$ as

$$m(x) = f(\hat{x}) + g^T(v^j + \Delta t d^j) + \frac{1}{2}(v^j + \Delta t d^j)^T B (v^j + \Delta t d^j),$$

where

$$v^j = x^j - \hat{x}. \tag{2.42}$$

Consequently, m can be written as a quadratic function in Δt,

$$\begin{aligned}\hat{m}(\Delta t) =& (f(\hat{x}) + g^T v^j + \frac{1}{2}(v^j)^T B v^j) + (g^T d^j + (d^j)^T B v^j)\Delta t \\ &+ ((d^j)^T B d^j)\Delta t^2.\end{aligned}$$

By expanding and grouping the coefficients 1, Δt, and Δt^2 we can write the above one dimensional quadratic function $\hat{m}$ as

$$\hat{m}(\Delta t) = f_j + f'_j \Delta t + \frac{1}{2} f''_j \Delta t^2, \tag{2.43}$$

where the coefficients f_j, f'_j, and f''_j are determined by

$$f_j := f(\hat{x}) + g^T v^j + \frac{1}{2}(v^j)^T B v^j,$$

$$f'_j := g^T d^j + (d^j)^T B v^j, \tag{2.44}$$

$$f''_j := (d^j)^T B d^j. \tag{2.45}$$

By differentiating $\hat{m}$ with respect to Δt and setting it to zero, we obtain $\Delta t^* = -f'_j / f''_j$. Now we have these cases:

- If $f''_j > 0$ and $\Delta t^* \in [0, \bar{t}_{j+1} - \bar{t}_j]$, we deduce that there exists a local minimizer of $m(x(t))$ at $t = \bar{t}_j + \Delta t^*$. Hence, the CP lies at $x(\bar{t}_j + \Delta t^*)$.

- Otherwise, the CP lies at $x(\bar{t}_j)$, provided that we have $f'_j > 0$.

- In all other cases the CP lies at $x(\bar{t}_{j+1})$ or outside of interval $[\bar{t}_j, \bar{t}_{j+1}]$. Therefore, we continue to search through the next interval. In these cases we need to calculate the new direction d^{j+1} from (2.40), and use this vector for calculating f_{j+1}, f'_{j+1} and f''_{j+1}. Since the difference between d^j and d^{j+1} is typically in just one component, we can made computational save by updating the coefficients f_{j+1}, f'_{j+1} and f''_{j+1} rather than calculating them from the scratch.

Now we need to be concerned with computing f'_{j+1} and f''_{j+1}. Assume that I is the set of indices corresponding to the variables that become active at $\bar{t}_{j+1}$. That is,

$$I_{j+1} := \{i \in \{1,\ldots,n\} : t_i = \bar{t}_{j+1}\}. \tag{2.46}$$

Then we have

$$d^{j+1} = d^j - \sum_{k\in I_{j+1}} d_k e_k,$$

where e_k is the kth column of the identity matrix. Now we can compute

$$\begin{aligned} f'_{j+1} &= f'_j + \Delta_j f''_j - (b^{j+1})^T x^{j+1} - \sum_{k\in I_{j+1}} d_k g'_k, \\ f''_{j+1} &= f''_j + (b^{j+1})^T \Big(\sum_{k\in I_{j+1}} d_k e_k - 2d^j \Big), \end{aligned} \tag{2.47}$$

where $g' := g - B\hat{x}$, $\Delta t_j := \bar{t}_{j+1} - \bar{t}_j$ and

$$b^{j+1} := B(\sum_{k\in I_{j+1}} d_k e_k) = \sum_{k\in I_{j+1}} d_k(Be_k). \tag{2.48}$$

Note that the computations (2.47) need only a matrix-vector multiplication (2.48) and two inner products of vectors. Furthermore, the matrix-vector multiplication includes columns of B which are indexed by I_{j+1}. Since $|I_{j+1}|$ is usually small, the product can be done very efficiently.

Now we give an algorithm for computing the generalized Cauchy point.

Algorithm 2.5.1. *Generalized Cauchy Point*

Step 0: *(Initialization)*

Given x_k, l_k, l, u_k, u, g_k, *and* B_k, *Initialize*

$$\begin{aligned} x &:= x_k, \\ g &:= g_k - B_k x_k \\ d &:= \lim_{t\to 0^+} P[x_k - tg_k, l, u] - x_k, \\ f' &:= (g_k)^T d, \text{ and} \\ f'' &:= d^T B_k d, \end{aligned}$$

If $f' \geq 0$, *go to step 4*

Step 1: *(Finding the next breakpoint)*

Compute

$$\Delta t := \max\{t : l_k \le x + td \le u_k\},$$

and find index set I of the indices corresponding all the variables that become active at $x + \Delta t d$.

Step 2: *(examining whether the CP has been found)*

If $f'' > 0$ and $0 < -(f'/f'') < \Delta t$, set

$$x := x - (f'/f'')d,$$

and go to step 4

Step 3: *(Updating the line derivatives)*

Compute

$$b := B_k\Big(\sum_{i\in I} d_i e_i\Big),$$

Reset

$$\begin{aligned} x &:= x + \Delta t d \\ f' &:= f' + \Delta t f'' - b^T x - \sum_{i\in I} d_i g_i, \\ f'' &:= f'' + b^T\Big(\sum_{i\in I} d_i e_i - 2d\Big), \\ d_i &:= 0 \textit{ for all } i \in I. \end{aligned}$$

If $f' \ge 0$, go to step 4.

Otherwise, go to step 1

Step 4: *(Termination with CP)*

Set $x^{cp} := x$.

2.5.3 A Conjugate Gradient Type Method

In this subsection we are concerned with solving the problem

$$\begin{array}{lll} \text{minimize} & m_k(x) & \\ \text{subject to} & (l_k)_i \le x_i \le (u_k)_i, & i \notin \mathscr{A}(x_k^{cp}), \\ & x_i = (x_k^{cp})_i, & i \in \mathscr{A}(x_k^{cp}). \end{array} \tag{2.49}$$

In particular, the variables in which the generalized Cauchy point lies at the bounds remain fixed and we minimize $m_k(x)$ over the subspace of free variables subject to the bound constraints. The index set of free variables at the iteration k defined by

$$\mathscr{F}_k = \{1,2,\ldots,n\} \setminus \mathscr{A}(x_k^{cp}).$$

Suppose that t denotes the number of free variables, and Z is a $n \times t$ matrix whose columns are unit vectors which span the subspace of free variables, Now we can transform (2.49) to the following quadratic problem for $\hat{x} \in \mathbb{R}^t$

$$\text{minimize} \quad \hat{m}_k(\hat{x}) := \frac{1}{2}(\hat{x}-\hat{x}_k)^T \hat{B}_k(\hat{x}-\hat{x}_k) + (\hat{x}-\hat{x}_k)^T r_k \tag{2.50}$$

$$\text{subject to} \quad (\hat{l}_k) \leq \hat{x} \leq (\hat{u}_k), \tag{2.51}$$

where

$$\hat{B}_k := Z_k^T B_k Z_k, \tag{2.52}$$

is the reduced Hessian matrix,

$$\hat{x}_k := Z^T x_k \quad r_k := Z^T g_k, \quad \hat{l}_k := Z^T l_k, \text{ and } \quad \hat{u}_k := Z^T u_k \tag{2.53}$$

represent, respectively, the projection of the x_k, g_k, l_k, and u_k onto the subspace of free variables.

For Solving the above problem, we ignore the bound constraints (2.51) and apply the conjugate gradient method with the starting point $Z^T x^{cp}$ to the system of linear equations

$$\hat{B}_k\hat{x} = -r_k - \hat{B}_k\hat{x}_k, \tag{2.54}$$

and terminate the iteration when one or more of the bound constraints (2.51) is violated, or when an excessive number of iterations has been done by the conjugate gradient method, or when the residual is smaller than

$$\delta_k := \min(0.1, \sqrt{\|\bar{g}_k\|})\|\bar{g}_k\|, \tag{2.55}$$

where $\bar{g}_k$ is projected gradient at x_k and defined by

$$\bar{g}_k := P[x_k - g_k] - x_k.$$

Algorithm 2.5.2. *Cunjugate Gradient Method*

Step 0: *(Initialization.)*

Compute $\hat{B}_k$, $\hat{l}_k$, $\hat{u}_k$ *from* (2.52) *and* (2.53)*, and* δ_k *from* (2.55)*. Set*

$$\begin{aligned} x &:= x^{cp}, \\ \hat{x} &:= Z^T x, \\ \hat{r} &:= -Z^T(g_k + B_k(x - x_k)), \\ \hat{p} &:= 0, \\ \rho_1 &:= 1, \\ \rho_2 &:= \hat{r}^T \hat{r}, \end{aligned}$$

Step 1: *(Test for required accuracy)*

If $\rho_2 < \delta_k$*, go to step 3.*

Step 2: *(Conjugate gradient iterations)*

Set

$$\begin{aligned} \beta &:= \rho_2/\rho_1, \\ \hat{p} &:= \hat{r} + \beta \hat{p}, \\ \hat{y} &:= \hat{B}_k \hat{p}, \\ \alpha_1 &:= \max\{\alpha : \hat{l} \leq \hat{x} + \alpha \hat{p} \leq \hat{u}\}, \end{aligned}$$

If $\hat{p}^T \hat{y} \leq 0$*, set* $\hat{x} := \hat{x} + \alpha_1 \hat{p}$ *and go to step 3.*

Otherwise, compute $\alpha_2 := \rho_2/\hat{p}^T \hat{y}$.

If $\alpha_2 > \alpha_1$*, set* $\hat{x} := \hat{x} + \alpha_1 \hat{p}$ *and go to step 3.*

Otherwise, reset

$$\begin{aligned} \hat{x} &:= \hat{x} + \alpha_1 \hat{p}, \\ \hat{r} &:= \hat{r} - \alpha_2 \hat{y}, \\ \rho_1 &:= \rho_2, \\ \rho_2 &:= \hat{r}^T \hat{r}, \end{aligned}$$

and return step 1.

Step 3: *(Termination of conjugate gradient algorithm)*

Set for $i = 1, \ldots, n$

$$(\bar{x}_{k+1})_i = \begin{cases} x_i^{cp} & \text{if } i \notin \mathscr{F} \\ (Z_k \hat{x})_i & \text{if } i \in \mathscr{F}. \end{cases}$$

2.5.4 The Algorithm

In this subsection, we summarize the method, which we have described in 2.5.1, in an algorithm. Before that we give some explanations about the approximated Hessian matrix B_k. We should mention that for global convergence of the algorithm, it is required that the Hessian approximations satisfy the following condition

$$\sum_{k=0}^{\infty} 1/(1+\min_{0\leq i\leq k} \|B_i\|) = \infty$$

The above condition is satisfied if, for example, we have

$$\|B_k\| \leq \lambda_1 + k\lambda_2 \tag{2.56}$$

for $k \in \mathbb{N}$ and fixed positive numbers λ_1 and λ_2.

The Hessian matrix can be approximated at each iteration by a number of different methods including BFGS,

$$B_{k+1} = B_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k},$$

and DFP,

$$B_{k+1} = B_k + \frac{r_k y_k^T + y_k r_k^T}{y_k^T s_k} - \frac{r_k^T s_k y_k y_k^T}{(y_k^T s_k)^2},$$

updating strategies. Where s_k and y_k represent, respectively, the displacement $x_{k+1} - x_k$ and the change of the gradient $g_{k+1} - g_k$, and r_k defined by

$$r_k := y_k - B_k s_k.$$

In both of the cases, we update only if the new approximation B_{k+1} is guaranteed to be positive definite, more precisely, the updates are performed provided s_k, y_k satisfy the following condition

$$\frac{y_k^T s_k}{y_k^T y_k} \geq 10^{-8}.$$

It has been shown in [15] that for the case of convex problems, the BFGS updates remain uniformly bounded, therefore, the condition (2.56) is automatically satisfied in this case. Another method for approximating the Hessian matrix is the symmetric rank-one(SR1) method which has the following update formula

$$B_{k+1} = B_k + \frac{r_k r_k^T}{r_k^T s_k},$$

This method does not guarantee that the new approximation is positive definite and must be controlled so that the condition (2.56) holds. Moreover, we skip the update whenever the condition

$$\|\frac{r_k r_k^T}{r_k^T s_k}\| > 10^8$$

holds for the rank-one correction.

Now we are in the position, in which we can specify the algorithm.

Algorithm 2.5.3. *A Trust region type algorithm for bound constrained optimization problems*

Step 1: *(Initialization)*

Given the initial feasible point x_0 with the gradient g_0, an initial trust region radius Δ_0, and an initial symmetric approximation of the Hessian matrix B_0. In addition, the positive constant $\gamma_0 < 1 < \gamma_2, \mu, \eta$ and ε are defined. Set

$$k = 0.$$

Step 1: *(Test for convergence)*

Compute the projected gradient

$$\bar{g}_k := P[x_k - g_k] - x_k.$$

If $\|\bar{g}_k\| < \varepsilon$, Stop

Step 2: *(Computing the Generalized Cauchy Point)*

Determine the bounds l_k and u_k from (2.32). *Compute the generalized Cauchy point by algorithm 2.5.1.*

Step 3: *(Finding the new iteration)*

Determine the active set $\mathscr{A}(x_k^{cp}, l_k, u_k)$. Use the conjugate gradient algorithm 2.5.2 to find an approximation $\bar{x}_{k+1}$ to the solution of the problem (2.34).

Step 4: *(Computing the ratio of actual reduction to predicted reduction in the function value)*

Compute $f(\bar{x}_{k+1})$ and set

$$\rho_k := (f(x_k) - f(\bar{x}_{k+1}))/(m_k(x_k) - m_k(\bar{x}_{k+1})).$$

Step 5: *(Updating)*

Set

$$x_{k+1} = \begin{cases} \bar{x}_{k+1} & \text{if } \rho_k > \mu, \\ x_k & \text{if } \rho_k \leq \mu, \end{cases}$$

$$g_{k+1} = \begin{cases} g(\bar{x}_{k+1}) & \text{if } \rho_k > \mu, \\ g_k & \text{if } \rho_k \leq \mu, \end{cases}$$

and update Δ_{k+1} from (2.30). *Apply a method for approximating the Hessian matrix B_{k+1} at x_{k+1} while ensuring that the condition* (2.56) *is satisfied. Set*

$$k := k+1,$$

and go to step 1

Remark 2.5.1. *It can be shown by short computation that*

$$P[x_k - g_k] - x_k = -g_{red}(x_k),$$

therefore, the termination condition

$$\|P[x_k - g_k] - x_k\| < \varepsilon$$

in the algorithm 2.5.3 is equivalent to the following condition

$$\|g_{red}(x_k)\| < \varepsilon,$$

where g_{red} is reduced gradient defined by (1.4). *That is, the algorithm 2.5.3 terminates whenever a stationary point is reached.*

2.6 L-BFGS-B

In this section we give the description of the limited memory BFGS method for solving the bound constrained optimization problem (1.1) with a large number of variables n. This method, which was proposed by BYRD, LU, ZHU and NOCEDAL [6], uses a limited quasi-Newton update method for approximating the Hessian matrices in such manner that the required storage is linear in n. Similar to all the algorithms that we have described in this chapter, the gradient projection method is applied to identify the set of active variables at each iteration. In fact, using limited memory BFGS matrices and the line search strategy are the main properties that make this method distinguished from the other methods, especially, from the trust region type method described in the previous section.

2.6.1 Outline of the Algorithm

Similar to 2.5 at each iteration k with the iterate x_k, we approximate the objective function f with a quadratic model of the form

$$m_k(x) = f(x_k) + g_k^T(x - x_k)^T B_k(x - x_k), \tag{2.57}$$

where g_k is the gradient of objective function at x_k, B_k is a positive definite matrix computed by the limited memory BFGS method. Then the quadratic model m_k is approximately minimized with respect to the feasible region $\{x \in \mathbb{R}^n : l \le x \le u\}$. This task is done by, first, applying the gradient projection method to fix some variables in one of their bounds and then minimizing the quadratic model over the subspace of free variables.

The gradient projection method is performed by first considering the piece-wise linear path

$$x_k(t) = P[x_k - tg_k, l, u],$$

which is obtained by projecting the steepest descent direction onto the feasible region, where the projection $P[,,]$ is computed from (2.5.1). Then we find the generalized Cauchy point x^{cp} as the first local minimizer the piece-wise quadratic univariate function

$$q_k(t) = m_k(x_k(t)).$$

After x_k^{cp} has been computed, the values of variables whose index belongs to $\mathscr{A}(x_k^{cp}) = \mathscr{A}(x_k^{cp}, l, u)$ are held fixed. Then we solve approximately the following quadratic programming over the subspace of free variables

$$\begin{array}{lll} \text{minimize} & m_k(x) & \\ \text{subject to} & l_i \le x_i \le u_i, & i \notin \mathscr{A}(x_k^{cp}), \\ & x_i = (x_k^{cp})_i, & i \in \mathscr{A}(x_k^{cp}), \end{array} \tag{2.58}$$

where $\mathscr{A}(x_k^{cp}, l, u)$, is defined as the set of indices corresponding to the variables whose value at x^{cp} is at one of their bounds. To solve (2.58), first we ignore the bound constraints and minimize the quadratic model $m_k(x)$ over the subspace of free variables, which can be done either by a direct method or an iterative method with respect to the subspace of free variables, or by a dual method in which the active bounds are dealt by Lagrange multipliers. Then the path toward the solution is truncated in such a way that the condition

$$l_i \le x_i \le u_i, \quad i \notin \mathscr{A}(x_k^{cp}) \tag{2.59}$$

is fulfilled.

After the approximated solution $\bar{x}_{k+1}$ of (2.58) has been obtained, the new iterate x_{k+1} is computed by a line search strategy which enforces the strong Wolfe conditions along the descent direction $P_k := \bar{x}_{k+1} - x_k$. That is, the new iterate is computed by

$$x_{k+1} = x_k + \alpha_k p_k \tag{2.60}$$

where α_k is a step-length that satisfies in the sufficient decrease condition

$$f(x_{k+1}) \leq f(x_k) + \gamma_1 \alpha_k g_k^T p_k, \tag{2.61}$$

and the curvature condition

$$|g_{k+1}^T p_k| \leq \gamma_2 |g_k^T p_k|, \tag{2.62}$$

where $\gamma_1 < \gamma_2 < 1$ are fixed positive numbers. Now it remains to show that the direction p_k is a descent direction. Firstly, since the generalized Cauchy point x^{cp} is the minimizer of the quadratic model m_k along the projected steepest descent direction, we have

$$m_k(x_k) > m_k(x^{cp}),$$

unless the projected gradient is equal to zero. Moreover, $\bar{x}_{k+1}$ lies on a path from x^{cp} to the point that minimizes the quadratic model m_k over the subspace of free variables. Hence, the value of the quadratic model m_k decreases along this path, in particular, $m_k(\bar{x}_{k+1})$ can not be greater than $m_k(x^{cp})$ and we have

$$f(x_k) = m_k(x_k) > m_k(x^{cp}) \geq m_k(\bar{x}_{k+1}) = f(x_k) + g_k^T p_k + \frac{1}{2} p_k^T B_k p_k,$$

which implies that

$$g_k^T p_k + \frac{1}{2} p_k^T B_k p_k < 0.$$

Since in this algorithm at each iteration, the limited memory BFGS method is used to approximate the Hessian matrix, B_k is positive definite. Consequently, we have

$$g_k^T p_k < 0.$$

2.6.2 Limited Memory BFGS Update

In this section we describe a method in which computing the limited memory BFGS can be done efficiently. BYRD, NOCEDAL and SCHNABEL [7] have derived the compact representation form of the BFGS method which allows us to implement the limited memory BFGS method efficiently.

Suppose that we are at the iterate x_k, and the pair correcter vectors $\{s_i, y_i\}$ for $i = 0, \dots, k-1$ are defined by

$$s_i =: x_{i+1} - x_i, \quad y_i := g_{i+1} - g_i, \tag{2.63}$$

and also for $i = 0, \dots, k-1$ the curvature condition

$$s_i^T y_i > 0 \tag{2.64}$$

is satisfied. Then the new BFGS matrix is computed by formula

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}. \tag{2.65}$$

Now we have the following theorem [7, p, 6]

Theorem 2.6.1 (Compact Representation Of BFGS)**.** *Let B_0 be symmetric and positive definite and k pairs $\{s_i, y_i\}_{i=0}^{k-1}$ satisfy the curvature condition* (2.64)*. Moreover, assume that k times update are applied to B_0 by using pairs vectors $\{s_i, y_i\}_{i=0}^{k-1}$ and the direct BFGS formula* (2.65)*, then the symmetric positive definite matrix B_k can be expressed as*

$$B_k = B_0 - \begin{bmatrix} B_0 S_k & Y_k \end{bmatrix} \begin{bmatrix} S_k^T B_0 S_k & C_k \\ C_k^T & -D_k \end{bmatrix}^{-1} \begin{bmatrix} S_k^T B_0 \\ Y_k^T \end{bmatrix} \tag{2.66}$$

Where $C_k \in \mathbb{R}^{k \times k}$ and diagonal matrix $D_k \in \mathbb{R}^{k \times k}$ are defined by

$$(C_k)_{i,j} := \begin{cases} s_{i-1}^T y_{j-1} & \text{if } i > j \\ 0 & \text{otherwise,} \end{cases} \tag{2.67}$$

$$D_k := \operatorname{diag}[s_0^T y_0, \dots, s_{k-1}^T y_{k-1}], \tag{2.68}$$

and the correction matrices $S_k, Y_k \in \mathbb{R}^{n \times k}$ have the form

$$S_k := [s_0, \dots, s_{k-1}], \quad Y_k = [y_0, \dots, y_{k-1}]. \tag{2.69}$$

Now we can use the update procedure, which is described in the above theorem, to compute the limited memory BFGS matrices. In the case of limited memory BFGS, at each iteration we store a small number r of recent correction pairs $\{s_i, y_i\}$ in the correction matrices S_k and Y_k. This matrices are updated by removing the oldest pair and adding the new pair to them. That is, the correction matrices belong to $\mathbb{R}^{n \times r}$ and have the forms

$$S_k := [s_{k-r}, \dots, s_{k-1}], \quad Y_k := [y_{k-r}, \dots, y_{k-1}]. \tag{2.70}$$

Moreover, we choose $B_0^{(k)} = \omega_k I$ as the initial matrix where ω_k is a positive scaler computed by

$$\omega_k := \frac{y_{k-1}^T y_{k-1}}{s_{k-1}^T y_{k-1}}. \tag{2.71}$$

Now if the curvature condition (2.64) holds for correction pairs $\{s_i, y_i\}$ with $i = k-r, \dots, k-1$, then the limited memory BFGS matrix B_k which is computed by applying r times updates to the initial matrix $\omega_k I$, using the correction pairs $\{s_i, y_i\}_{k-r}^{k-1}$ and the BFGS formula can be expressed as

$$B_k = \omega_k I - U_k N_k U_k^T \tag{2.72}$$

where

$$U_k := \begin{bmatrix} Y_k & \omega_k S_k \end{bmatrix}, \tag{2.73}$$

$$N_k := \begin{bmatrix} -D_k & C_k^T \\ C_k & \omega_k S_k^T S_k \end{bmatrix}^{-1}, \tag{2.74}$$

and C_k and D_k belong to $\mathbb{R}^{r \times r}$ and defined as

$$(C_k)_{i,j} := \begin{cases} s_{k-r-1+i}^T y_{k-r-1+j} & \text{if } i > j \\ 0 & \text{otherwise,} \end{cases} \tag{2.75}$$

$$D_k := \operatorname{diag}\,[s_{k-r}^T y_{k-r}, \dots, s_{k-1}^T y_{k-1}]. \tag{2.76}$$

It should be mentioned that the matrix N_k is a $2r \times 2r$ matrix and since the positive integer r is chosen to be small, the computation of N_k^{-1} is cheap. Furthermore, throughout the algorithm we do not compute B_k explicitly.

It has been shown in [7] that similar to (2.72), the inverse limited memory BFGS matrix H_k can be represented by

$$H_k = \frac{1}{\omega_k} I - \bar{U}_k \bar{N}_k \bar{U}_k^T, \tag{2.77}$$

where

$$\bar{U}_k := \begin{bmatrix} \frac{1}{\omega_k} Y_k & S_k \end{bmatrix}, \tag{2.78}$$

$$\bar{N}_k := \begin{bmatrix} 0 & -Q_k^{-1} \\ -Q_k^{-T} & Q_k^{-T}(D_k + \frac{1}{\omega_k} Y_k^T Y_k) Q_k^{-1} \end{bmatrix}, \tag{2.79}$$

and $Q_k \in \mathbb{R}^{r\times r}$ defined as

$$(Q_k)_{i,j} := \begin{cases} s_{k-r-1+i}^T y_{k-r-1+j} & \text{if } i \leq j \\ 0 & \text{otherwise.} \end{cases} \tag{2.80}$$

$$\tag{2.81}$$

Note that although the line search procedure, which is used in this method, imposes the strong Wolfe conditions, the condition $s_k^T y_k > 0$ may not always hold. This is due to presence of bound constraints. Hence, to preserve the positive definiteness of the limited memory BFGS matrices, we skip to update the limited memory BFGS matrice B_k and discard the correction pair $\{s_k, y_k\}$ if the curvature condition

$$s_k^T y_k > eps\|y\|^2 \tag{2.82}$$

does not hold, where eps is the machine precision. In this situation we do not remove the oldest correction pair from the correction matrices (2.70).

2.6.3 The Generalized Cauchy Point

The computation of the generalized Cauchy point in the L-BFGS-B method is similar as it has been described in 2.5.2. Except that because of the special structure of limited memory BFGS matrices, the computations and updating of the direction derivations f'_j, and f''_j are different and can be done in a more efficient manner.

Suppose that we have searched through the intervals $[\bar{t}_{k-1}, \bar{t}_k]$ with $k = 1, \ldots, j$ and the generalized Cauchy point has been not found. Now we are examining the interval $[\bar{t}_j, \bar{t}_{j+1}]$. we set

$$x^{j+1} = x^j + \Delta t_j d^j, \quad \text{with} \quad \Delta t_j = \bar{t}_{j+1} - \bar{t}_j. \tag{2.83}$$

Suppose that only one variable with index b becomes active at $\bar{t}_{j+1}$. we compute the search direction d^{j+1} by updating d^j as follows

$$d^{j+1} = d^j + g_b e_b, \tag{2.84}$$

where e_b is the unit vector with one at the bth component and zeros elsewhere. According to the definitions (2.42) and (2.83) we have

$$v^{j+1} = v^j + \Delta_j d^j \tag{2.85}$$

Hence by using (2.44), (2.45), (2.84) and (2.85) we have

$$\begin{aligned} f'_{j+1} =& g^T d^{j+1} + (d^{j+1})^T B v^{j+1} \\ =& g^T d^j + g_b^2 + (d^j)^T B v^j + \Delta t_j (d^j)^T B d^j + g_b e_b^T v^{j+1} \\ =& f'_j + \Delta t_j f''_j + g_b^T + g_b e_b^T B v^{j+1} \end{aligned} \tag{2.86}$$

and

$$\begin{aligned} f''_{j+1} =& (d^{j+1})^T B d^{j+1} \\ =& (d^j) B d^j + 2 g_b e_b^T B d^j + g_b^2 e_b^T B e_b \\ =& f''_j + 2 g_b e_b^T B d^j + g_b^2 e_b^T B e_b. \end{aligned} \tag{2.87}$$

Note that since B is a dense matrix, the only expensive computations in (2.86) and (2.87) are

$$e_b^T B v^{j+1}, \quad e_b^T B d^j, \text{ and } e_b^T B e_b,$$

which can be accomplished in order $O(n)$ operations . Nevertheless, by using the limited memory BFGS formula

$$B = \omega I + U N U^T, \tag{2.88}$$

and the definition (2.40), we can express the updating formula (2.86) and (2.87) as

$$f'_{j+1} = f'_j + \Delta t_j f''_j + g_b^2 + \omega g_b v_b^{j+1} - g_b U_{b:}^T N U^T u^{j+1}, \tag{2.89}$$

$$f''_{j+1} = f''_j - 2\omega g_b^2 - 2 g_b U_{b:}^T N U^T d^j + \omega g_b^2 - g_b^2 U_{b:}^T N u_b, \tag{2.90}$$

where $U_{b:}^T$ represents the bth row of the matrix U. Note that in (2.89) and (2.90) only the computations of $U^T v^j$ and $U^T d^j$ require $O(n)$ operations. However, because from (2.84) and (2.85), the vectors v^{j+1} and d^{j+1} can be updated at each iteration by simple computation, we can accomplished these computations in order $O(r^2)$ operations for a small integer r, provided we maintain the following $2r$-vectors at each iteration

$$\begin{aligned} q^{j+1} :=& U^T d^{j+1} = U^T (d^j + g_b e_b) = q^j + g_b u_b, \\ w^{j+1} :=& U^T v^{j+1} = U^T (v^j + \Delta t_j d^j) = w^j + \Delta t_j q^j. \end{aligned}$$

By using q^{j+1} and w^{j+1}, we can express the directional derivatives f'_{j+1} and f''_{j+} as

$$\begin{aligned} f'_{j+1} &= f'_j + \Delta t_j f''_j + g_b^2 + \omega g_b v_b^{j+1} - g_b U_{b:}^T N w^{j+1}, \\ f''_{j+1} &= f''_j - \omega g_b^2 - 2 g_b U_{b:}^T N q^j - g_b^2 U_{b:}^T N U_{b:}. \end{aligned}$$

If more than one variables will be active at $\bar{t}_{j+1}$ the above process, which outlined above, will be repeated again.

Now we are in the position that we can specify an algorithm for computing CP point:

Algorithm 2.6.1. *Generalized Cauchy Point*

Step 0: *(Initialization)*

Given x, l, u, g, and $B = \omega I + UNU^T$, for $i = 1, \ldots, n$ compute the break points and the components of the direction d in each coordinate by

$$t_i := \begin{cases} \frac{\hat{x}_i - u_i}{g_i} & \text{if } g_i < 0 \\ \frac{\hat{x}_i - l_i}{g_i} & \text{if } g_i > 0 \\ \infty & \text{otherwise,} \end{cases}$$

$$d_i := \begin{cases} 0 & \text{if } t_i = 0 \\ -g_i & \text{otherwise} \end{cases}$$

Initialize

$$\begin{aligned} \mathscr{F} &:= \{i : t_i > 0\}, \text{ (The set of indices corresponding to the free variables)} \\ q &:= U^T d, \\ w &:= 0, \\ t &:= \min_{i \in \mathscr{F}} t_i, \text{ (using the heapsort algorithm)} \\ t_{old} &:= 0, \\ \Delta t &:= t - t_{old} = t - 0, \\ b &:= i \text{ such that } t_i = t. \text{ Remove the } b \text{ from } \mathscr{F}. \end{aligned}$$

step 1: *(Examining the first interval $[0, \bar{t}_1]$)*

Compute

$$\begin{aligned} f' &:= g^T d = -d^T d, \\ f'' &:= \omega d^T d - d^T UNU^T d = -\omega f' - q^T nq \\ \Delta t_{min} &:= -f'/f''. \end{aligned}$$

If $\Delta t_{min} < \Delta t$ go to Step 4.

Step 2: *(examining the subsequent segments)*

Set

$$x_r^{cp} := \begin{cases} u_b & \text{if } d_b > 0 \\ l_b & \text{if } d_b < 0. \end{cases}$$

Compute

$$\begin{aligned} v_b &:= x_b^{cp} - x_b, \\ w &:= w + \Delta t q, \\ f' &:= f' + \Delta t_j f'' + g_b^2 + \omega g_b v_b - g_b U_{b:}^T N w, \\ f'' &:= f'' - \omega g_b^2 - 2 g_b U_{b:}^T N q - g_b^2 U_{b:}^T N U_{b:}. \\ q &:= q + g_b U_{b:} \end{aligned}$$

Set

$$\begin{aligned} d_b &:= 0, \\ \Delta t_{min} &:= -f'/f'', \\ t_{old} &:= t, \\ t &:= \min_{i \in \mathscr{F}} t_i, \\ \Delta t &:= t - t_{old}, \\ b &:= i \text{ such that } t_i = t. \text{ Remove the } b \text{ from } \mathscr{F}. \end{aligned}$$

Step 3: *(Loop)*

If $\Delta t_{min} \geq \Delta t$ go to step 2.

Step 4: *Set*

$$\begin{aligned} \Delta t_{min} &:= \max(\Delta t_{min}, 0), \\ t_{old} &:= t_{old} + \Delta t_{min}, \\ x_i^{cp} &:= x_i + t_{old} d_i, \text{ for all } i \text{ such that } t_i \geq t, \\ \text{For all } i \in \mathscr{F} & \text{ with } t_i = t, \text{ remove } i \text{ from } \mathscr{F}. \\ w &:= w + \Delta t_{min} q \end{aligned}$$

Note that at last step of the above algorithm, the vector w is updated so that at the termination we have

$$w = U^T (x^{cp} - x_k). \tag{2.91}$$

we will use this vector in the primal direct and the conjugate gradient method.

2.6.4 Subspace Minimization

After the generalized Cauchy point x^{cp} has been determined, we deal with approximating the solution of (2.58) which consists of minimizing the quadratic model m_k over the subspace of free variables, and enforcing the bound constraints on the free variables. In this section we shall present three methods including a directed primal method, a conjugate gradient method which is similar to 2.5.3 and a dual method. In all of the three methods, first, we disregard the bound constraints and solve the unconstrained problem with objective function m_k over the subspace of free variables. Then the path obtained by the solution of unconstrained problem is truncated so that the bound constrained conditions are satisfied.

From now on, we denote the index set corresponding to the free variables at x^{cp} by $\mathscr{F}$, which can express alternatively as

$$\mathscr{F}_k = \{1,2,\dots,n\} \setminus \mathscr{A}(x_k^{cp}),$$

where the subscript k stands for the outer iteration. Moreover, we define Z_k to be a matrix with unit columns which span the subspace of free variables.

A Directed Primal Method

In the directed primal method, the variables in which the generalized Cauchy point lies at the bound remain fixed and we minimize m_k over the subspace of free variables by starting from x^{cp} and enforcing the bound constraints corresponding to $\mathscr{F}$. Hence, we consider only points $x \in \mathbb{R}^n$ of the form

$$x = x^{cp} + Z_k p,$$

where vector p belongs to the subspace of free variables. Now we can rewrite the quadratic model $m_k(x)$ as

$$\begin{aligned} m_k(x) &= f(x_k) + g_k^T(x - x^{cp} + x^{cp} - x_k) + \frac{1}{2}(x - x^{cp} + x^{cp} - x_k)^T B_k (x - x^{cp} + x^{cp} - x_k) \\ &= (g_k + B_k(x^{cp} - x_k))^T (x - x^{cp}) + \frac{1}{2}(x - x^{cp})^T B_k (x - x^{cp}) + \zeta \\ &:= p^T r_k + \frac{1}{2} p^T \hat{B}_k p + \zeta, \end{aligned}$$

where ζ is a constant,

$$\hat{B}_k := Z_k^T B_k Z_k \tag{2.92}$$

is the reduced Hessian matrix, and

$$r_k := Z_k^T (g_k + B_k(x^{cp} - x_k))$$

represents the projected gradient of the m_k at x_k onto the subspace of free variables. By using (2.91) and (2.72) the projected gradient r_k can be rewritten as

$$r_k = Z_k^T(g_k + \omega_k(x^{cp} - x_k) - U_k N_k w), \tag{2.93}$$

where the vector w has been already computed while computing the generalized Cauchy point. Now we can transform (2.58) to the following problem

$$\text{minimize} \quad m_k(p) := \frac{1}{2}p^T \hat{B}_k p + r_k^T p \tag{2.94}$$

$$\text{subject to} \quad l_i - x_i^{cp} \le p_i \le u_i - x_i^{cp}, \quad i \in \mathscr{F}_k. \tag{2.95}$$

As we have mentioned to solve the above problem, we ignore the bound constraints (2.95) and solve the unconstrained problem (2.94), which is done by

$$p^u := -\hat{B}_k^{-1} r_k. \tag{2.96}$$

Since the limited memory BFGS matrix B_k is a small-rank correction of a diagonal matrix, for computing B_k^{-1} we can apply the Sherman-Morrison-Woodbury formula, In particular, by using (2.72) we can express the reduced Hessian matrix $\hat{B}$ as

$$\hat{B} = \omega I - Z^T U(NU^T Z),$$

where the subscript k of the outer iteration is dropped for simplicity. By using Sherman-Morrison-Woodbury formula we obtain

$$\hat{B}^{-1} = \frac{1}{\omega}I + \frac{1}{\omega}Z^T U(I - \frac{1}{\omega}NU^T ZZ^T U)^{-1} NU^T Z \frac{1}{\omega}.$$

By substituting above expression in (2.96), the solution of unconstrained problem can computed by

$$p^u = \frac{1}{\omega}r + \frac{1}{\omega^2}Z^T U(I - \frac{1}{\omega}NU^T ZZ^T U)^{-1} NU^T Zr. \tag{2.97}$$

Once the Newton direction p^u has been computed, we impose the bound constraints (2.95) which is done by setting

$$p^* := \alpha^* p^u, \tag{2.98}$$

where the positive number α^* is defined by

$$\alpha^* = \max\{\alpha : \alpha \le 1, \quad l_i - x_i^{cp} \le \alpha p_i^u \le u_i - x_i^{cp}, \quad i \in \mathscr{F}\}. \tag{2.99}$$

Thus, we can express the approximated solution $\bar{x}$ of (2.58) as

$$\bar{x}_i = \begin{cases} x_i^{cp} & \text{if } i \notin \mathscr{F} \\ x_i^{cp} + (Z_k p^*)_i & \text{if } i \in \mathscr{F}. \end{cases} \tag{2.100}$$

Algorithm 2.6.2. *(Direct Primal method)*

Given x, l, u, g, w and $B = \omega I + UNU^T$.(Note that the subscript k for the outer iteration have been omitted for simplicity)

Step 1: *(Computing $NU^T Zr$)*

Compute

$$\begin{aligned} Zr &:= ZZ^T(g + \omega(x^{cp} - x) - UNw) \\ v &:= U^T Zr \\ v &:= Nv \end{aligned}$$

Step 2: *(Computing $M := I - \frac{1}{\omega} NU^T ZZ^T U$)*

Set

$$\begin{aligned} M &:= \frac{1}{\omega} NU^T ZZ^T U \\ M &:= I - NM \end{aligned}$$

Step 3: *(Computing the Newton direction p^u)*

Compute

$$\begin{aligned} v &:= M^{-1} v \\ p^u &:= -(\frac{1}{\omega} r + \frac{1}{\omega^2} Z^T U v) \end{aligned}$$

Step 4: *(Imposing the bound constraints (2.95))*

Compute

$$\begin{aligned} &\alpha^* := \max\{\alpha : \alpha \leq 1, \quad l_i - x_i^{cp} \leq \alpha p_i^u \leq u_i - x_i^{cp}, \quad i \in \mathscr{F}\} \\ &p^* := \alpha^* p^u \\ &\bar{x}_i := \begin{cases} x_i^{cp} & \text{if } i \notin \mathscr{F} \\ x_i^{cp} + (Z_k p^*)_i & \text{if } i \in \mathscr{F}. \end{cases} \end{aligned}$$

A Primal Conjugate Gradient Type Method

Another approach for approximating the solution of (2.58) is to solve the positive definite linear system

$$\hat{B}_k p^u = -r, \tag{2.101}$$

iteratively, by applying the conjugate gradient method. Similar to 2.5.3 the algorithm terminates whenever one the following conditions is satisfied:

- One or more of the bound constraints (2.95) is violated.
- The residual in the conjugate gradient method is smaller than

$$\delta_k := \min(0.1, \sqrt{||r_k||})||r_k||. \tag{2.102}$$

It should be mentioned that since the limited memory BFGS matrix B_k is positive definite and also all of its eigenvalues are almost identical, it is suitable to use the conjugate gradient method.

Algorithm 2.6.3. *(Conjugate Gradient Method)*

Given $x, l, u, g, w, B = \omega I + UNU^T$,and δ from (2.102).*(Note that the subscript k of the outer iteration have been omitted for simplicity)*

```
Step 0: (Initialization.)
    Set
        $d := 0,$
        $\hat{r} := Z^T(g + \omega(x^{cp} - x) - UNw),$
        $\hat{p} : -\hat{r},$
        $\rho_1 := 1,$
        $\rho_2 := \hat{r}^T\hat{r},$
Step 1: (Test for required accuracy)
    If $\rho_2 < \delta^2$, go to step 3.
Step 2: (Conjugate gradient iterations)
    Set
        $\beta := \rho_2/\rho_1,$
        $\hat{p} := -\hat{r} + \beta\hat{p},$
        $\hat{y} := \hat{B}\hat{p},$
        $\alpha_1 := \max\{\alpha : l_i \leq x_i^{cp} + d_i + \alpha\hat{p}_i \leq u_i\},$
        $\alpha_2 := \rho_2/\hat{p}^T\hat{y}.$
    If $\alpha_2 > \alpha_1$, set $d := d + \alpha_1\hat{p}$ and go to step 3.
    Otherwise, reset
        $d := d + \alpha_1\hat{p},$
        $\hat{r} := \hat{r} + \alpha_2\hat{y},$
        $\rho_1 := \rho_2,$
        $\rho_2 := \hat{r}^T\hat{r},$
    and return step 1.
```

Step 3: *(Termination of conjugate gradient algorithm)*

Set for $i = 1, \dots, n$

$$(\bar{x}_{k+1})_i := \begin{cases} x_i^{cp} & \text{if } i \notin \mathscr{F} \\ x_i^{cp} + (Z_k d)_i & \text{if } i \in \mathscr{F}. \end{cases}$$

A Dual Method

In practice, it is usually observed that the number of active variables is small in comparison to the dimension of the problem. Thus, it should be appropriate to deal with active bounds by corresponding Lagrange multipliers. This kind of methods is referred to as a dual method. We consider the point $x \in \mathbb{R}^n$ of form

$$x := x_k + p,$$

where $x_k + p$ is restricted to lie on the subspace of free variables at x^{cp}. Which is done by enforcing the condition

$$L_k^T p = L_k^T (x^{cp} - x_k), \tag{2.103}$$

where L_k is a matrix whose columns are unit vectors which constitute a basis for the subspace of active variables at x^{cp}. In addition, we have

$$L_k^T Z_k = 0, \quad L_k L_k^T + Z_k Z_k^T = I.$$

Now by stetting

$$b_k := L_k^T (x^{cp} - x_k),$$

we form the following subspace problem

$$\begin{aligned} &\text{minimize} && g_k^T p + \frac{1}{2} p^T B_k p && (2.104) \\ &\text{subject to} && L_k^T p = b_k && (2.105) \\ & && l \leq x_k + p \leq u. && (2.106) \end{aligned}$$

Similar to other approaches in this section first we solve this problem without considering the bound constraints (2.106). This is done by writing the optimality conditions

$$g_k + B_k p^* + L_k \mu^* = 0, \tag{2.107}$$

$$L_k^T p^* = b_k. \tag{2.108}$$

for (2.104)-(2.105), and multiplying (2.107) by $L_k^T H_k$ from the right-hand-side, where H_k represents the inverse of B_k. This yields

$$L_k^T H_k g_k + L_k^T P^* + L_k^T H_k L_k \mu^* = 0,$$

by using (2.108) we obtain

$$(L_k^T H_k L_k)\mu^* = -L_k^T H_k g_k - b_k. \tag{2.109}$$

Since L_k is a full rank matrix and whose columns are unit vectors, $L_k^T H_k L_k^T$ is a principal sub-matrix of H_k. Hence we can determine μ^* by solving (2.109). Furthermore, since H_k is limited memory BFGS matrix and $L_k^T L_k = I$, we can again use the Sherman-Morrison-Woodbury formula to solve the linear system (2.109). To see this, we have

$$L_k^T H_k L_k = \frac{1}{\omega_k} I + (L_k^T \bar{U}_k)(\bar{N}_k \bar{U}_k^T L_K).$$

By using the Sherman-Morrison-Woodbury formula we obtain

$$(L_k^T H_k L_k)^{-1} = \omega_K I - \omega_k L_k^T \bar{U}_k (I + \omega_k \bar{N}_k \bar{U}_k^T L_k L_k^T \bar{U}_k)^{-1} \bar{N}_k \bar{U}_k^T L_k \omega_k. \tag{2.110}$$

Once μ^* has been determined, we compute p^* by solving the linear system

$$B_k p^* = -L_k \mu^* - g_k. \tag{2.111}$$

Finally, the approximated solution $\bar{x}$ of (2.58) is computed by

$$\bar{x} = x^{cp} + \alpha^* (x_k + p^* - x^{cp}), \tag{2.112}$$

where the positive number α^* is defined by

$$\alpha^* = \max\{\alpha : l_i - x_i^{cp} \le \alpha((x_k)_i + p_i^* - x_i^{cp}) \le u_i - x_i^{cp}, i \in \mathscr{F}\}. \tag{2.113}$$

Algorithm 2.6.4. *(Dual Method)*
Given $x, l, u, g, w, H = \frac{1}{\omega} I + \bar{U}\bar{N}\bar{U}.^T$ and we define $n_{\mathscr{A}}$ the number of active variables at x^{cp}, In other words,

$$n_{\mathscr{A}} = |\mathscr{A}(x^{cp}, l, u)|.$$

(Note that the subscript k of the outer iteration have been omitted for simplicity)

Step 1: *If $n_{\mathscr{A}} = 0$, Compute*

$$\begin{aligned} u &:= \bar{U}^T g \\ u &:= \bar{N} u \\ p^* &:= -\frac{1}{\omega} g - \bar{U} u \end{aligned}$$

and go to step 5

Step 2: *(Computing the right-hand-side of* (2.109)*)*

Compute

$$\begin{aligned} b &:= -L^T(x^{cp} - x_k) \\ v &:= \bar{U}^T g \\ v &:= \bar{M}u \\ q &:= L^T \bar{U} v \\ q &:= -\frac{1}{\omega} L^T g - q - b \end{aligned}$$

Step 3: *(Computing μ^*)*

Reset

$$u := \bar{U}^T Lq$$

Form $M = (I + \omega \bar{N} \bar{U}^T L L^T \bar{U})$ as follows

$$\begin{aligned} \bar{M} &:= \omega \bar{U}^T L L^T \bar{U} \\ \bar{M} &:= I + \bar{N}\bar{M} \end{aligned}$$

Compute

$$\begin{aligned} u &:= \bar{M}^{-1} \bar{N} u \\ \mu^* &:= \omega^2 L^T \bar{U} u \\ \mu^* &:= -\omega q + \mu^* \end{aligned}$$

Step 4: *(Computing p^*)*

Compute

$$\begin{aligned} u &:= \bar{U}^T L \mu^* \\ u &:= \bar{N} u + v \\ p^* &:= -\frac{1}{\omega}(L\mu^* + g) + \bar{U} u \end{aligned}$$

Step 5: *Compute*

$$\alpha^* = \max\{\alpha : l_i - x_i^{cp} \leq \alpha((x_k)_i + p_i^* - x_i^{cp}) \leq u_i - x_i^{cp}, i \in \mathscr{F}\},$$

and set

$$\bar{x} = x^{cp} + \alpha^*(x_k + p^* - x^{cp})$$

2.6.5 The Algorithm for L-BFGS-B

Now, at this moment we can summarize the L-BFGS-B method in to an algorithm. Similar to 2.5.3 we use

$$\|P[x_k - g_k] - x_k\|_\infty < 10^{-5} \tag{2.114}$$

as the termination condition.

Algorithm 2.6.5. *(L-BFGS-B)*

Step 0: *(Initialization)*

Choose an initial point x_0, and an integer r to be the number of limited memory corrections stored. Define the initial limited memory Hessian matrix B_0 to be identity. Set

$$k := 0$$

Step 1: *(Test for convergence)*

If the condition (2.114) *is satisfied, stop*

Step 2: *(Computing the generalized Cauchy point)*

Compute x^{cp} by algorithm 2.6.1.

Step 3: *(Computing the descent direction)*

Use one of the methods described in 2.6.4, including the directed primal method, the conjugate gradient method or the dual method, to determine the approximated solution $\bar{x}_{k+1}$ of (2.58). *Then set*

$$p_k := \bar{x}_{k+1} - x_k$$

Step 4: *(Line search iteration)*

Implement a line search iteration along the direction p_k, with respect to the bound constraints, to determine the step length α_k, and set

$$x_{k+1} = x_k + \alpha_k p_k$$

Note that the line search starts with the unit as the initial value and satisfies the strong Wolfe conditions (2.61) *and* (2.64) *with $\gamma_1 = 10^{-4}$ and $\gamma_2 = 0.9$.*

Step 5: *(Updating gradient)*

Set

$$g_{k+1} := \nabla f(x_{k+1})$$

Step 6: *If y_k and s_k satisfy the condition* (2.82) *with* $eps = 2.2 \times 10^{-16}$*, add the vectors y_k and s_k to the correction matrices S_k and Y_k.*

If both of the matrices S_k and Y_k have more than r update columns, remove the oldest columns from both of them.

Step 7: *(Updating limited memory BFGS)*

Update $S_k^T S_k, Y_k^T Y_k, C_k$ and Q_k, and set

$$\omega = \frac{y_k^T y_k}{y_k^T s_k}.$$

Step 8: *(Loop)*

Reset

$$k := k+1,$$

and go to step 1

Chapter 3

A New Quasi Newton Method for Bound Constrained problems

3.1 Introduction

In this chapter we introduce a new approach [24] to solve the bound constrained problem (1.1). Our proposed method is an active set method that uses a combination of the steepest descent directions and quasi Newton directions to identify the optimal active bound constraints. Once the optimal set of active variables has been identified, these variables are fixed at their bounds and the problem is reduced to a smaller problem over the subspace of free variables. This reduced problem is solved by performing the bent line search method along the quasi Newton direction. Particularly, the quasi Newton direction is computed by help of the limited memory symmetric rank-one matrix. As it is known the SR1 matrices are not necessarily positive definite, consequently, the quasi-Newton direction need not be a descent direction. In such a case, we regularize this direction so that it will become a descent direction. The convergence theory of the algorithm is also provided.

3.2 Bent line Searches

In this subsection, we give a description of a line search approach for the case of bound constrained optimization problems

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & l \le x \le u \end{array} \tag{3.1}$$

In general, a line search method generates a descent sequence of feasible points which is constructed at each iterate x_k by choosing a search direction p_k satisfying the descent condition

$$g_k^T p_k < 0, \tag{3.2}$$

and a corresponding search path $x_k + \alpha p_k$ for $\alpha > 0$. Then the positive parameter α_k is chosen in such a way that $x_{k+1} := x_k + \alpha_k p_k$ satisfies the condition

$$f(x_{k+1}) < f(x_k), \tag{3.3}$$

where the positive parameter α_k is referred to as step-length.

From now on throughout this chapter, we drop the subscript k of the outer iteration for simplicity. Therefore, x, g, and p represent x_k, g_k and p_k, and we write $\hat{x}$ for x_{k+1}. Given a current iterate x, we call a component x_i and the corresponding index i **active** if either $x_i = l_i$ or $x_i = u_i$. Otherwise, if $x_i \in (l_i, u_i)$, we call the ith component and its corresponding index **nonactice** or **free**. In the case of bound constrained optimization problems, a search direction p must be also a feasible direction at x in addition to satisfy the descent condition.

Definition 3.2.1. *Given a feasible point x, we call the direction vector d feasible at x if there exists a sufficiently small $\alpha > 0$ such that $x + \alpha d$ is feasible.*

The feasibility of p at x requires that

$$\left.\begin{array}{ll} p_i \geq 0 & \text{if } x_i = l_i, \\ p_i \leq 0 & \text{if } x_i = u_i. \end{array}\right\} \tag{3.4}$$

If the gradient $g = g(x)$ has a nonzero component g_i at a nonactive index i, we can move from x_i along both of the directions $\pm g_i$ while remaining within feasible region. Therefore the value of objective function is reduced by moving from x_i along the direction $-g_i$. However, if x_i is active in one of their bounds, only changes x_i in one direction is possible, since by moving from x_i along the opposite direction we would lose the feasibility. Therefore, for an active variable i the objective function f decreases if we have

$$\left.\begin{array}{ll} g_i < 0 & \text{if } x_i = l_i, \\ g_i > 0 & \text{if } x_i = u_i. \end{array}\right\} \tag{3.5}$$

In this case, we say the variable x_i can be freed from its bound.

As we have mentioned in 1.2, at a local minimizer $\bar{x}$ of the bound constrained optimization problem (1.1), we have

$$g_{red}(\bar{x}) = 0, \tag{3.6}$$

where g_{red} is the reduced gradient.

Definition 3.2.2. *For an arbitrary feasible point x of the bound constrained optimization problem* (1.1)*, we call an active bound on x_i* ***strongly active*** *if one of the following conditions is satisfied.*

1. $g_i > 0$ *and* $x_i = l_i$,
2. $g_i < 0$ *and* $x_i = u_i$.

Consequently, an active bound called ***weakly active*** *if it is not strongly active.*

While creating a line search method for bound constrained optimization problems, we face several difficulties. First of all, as we have mentioned, in addition to being a descent direction the search direction p must be also a feasible direction. Moreover, the search path must be modified since linear search path may leave the feasible region. To be more accurate, let p be a search direction and $x+\alpha p (\alpha > 0)$ be the corresponding linear search path, then for an index i the bound constraint $[l_i, u_i]$ is violated, provided either $p_i > 0$ and $u_i < \infty$, or $p_i < 0$ and $l_i > -\infty$. To overcome this difficulty, we define the following piecewise linear **Bent line search** path

$$x(\alpha) = P[x+\alpha p, l, u], \tag{3.7}$$

which is obtained by projecting the linear path $x+\alpha p(\alpha > 0)$ onto the bound constraints

$$\{x \in \mathbb{R}^n : l \leq x \leq u\},$$

where $P[,,]$ is a projection defined by

$$(P[x,l,u])_i = \begin{cases} l_i & \text{if } x_i \leq l_i, \\ u_i & \text{if } x_i \geq u_i, \\ x_i & \text{if } l_i < x_i < u_i. \end{cases}$$

The bent line search path is linear on each interval $[\alpha_{i-1}, \alpha_i]$ for $i = 1, ..., m+1$. Where

$$0 = \alpha_0 < \alpha_1 < \cdots < \alpha_m < \alpha_{m+1} = \infty,$$

and the break points $\alpha_1, \ldots, \alpha_m$ belong to the set

$$S := \left\{ \frac{u_i - x_i}{p_i} : p_i > 0 \right\} \cup \left\{ \frac{l_i - x_i}{p_i} : p_i < 0 \right\} \setminus \{0, \infty\}. \tag{3.8}$$

Proposition 3.2.1. *Suppose p is a feasible direction at x for which the following conditions hold*

1. $g_{red}^T p < 0$,

2. *$p_i = 0$ if either $x_i = l_i$, $g_i > 0$ or $x_i = u_i$, $g_i < 0$,*

then for sufficiently small $\alpha > 0$ the bent search path (3.7) *is feasible, and also we have $f(x(\alpha)) < f(x)$.*

Proof. Due to the definition of the reduced gradient and second condition, it can easily be shown that

$$g_i p_i = (g_{red})_i p_i \quad \text{for all } i = 1, \dots, n.$$

Hence, we have

$$g^T p = g^T_{red} p \leq 0.$$

Since p is a feasible direction, we have $x(\alpha) = x + \alpha p$ and consequently $f(x(\alpha)) = f(x + \alpha p) = f(x) + \alpha g^T p + o(\alpha)$ if $\alpha > 0$ is sufficiently small. □

There are many ways to construct a bent line search. According to our convergence theory we need only to impose the following conditions

B1 If $S = \emptyset$ or $\alpha < \min S$, the line search is efficient.

B2 If $\alpha \geq \min S$, the condition

$$f(x(\alpha)) < f(x) \tag{3.9}$$

is satisfied.

To implement the bent line search satisfying the above conditions. First, we determine the values of α for which each component reaches its bound along the chosen search direction p by

$$\bar{\alpha}_i := \begin{cases} (u_i - x_i)/p_i & \text{if } p_i > 0 \text{ and } u_i < \infty \\ (l_i - x_i)/p_i & \text{if } p_i < 0 \text{ and } l_i > -\infty \end{cases} \tag{3.10}$$

Then we eliminate the duplicate values of $\bar{\alpha}_i$ from the set $\{\bar{\alpha}_1, \dots, \bar{\alpha}_n\}$ and sort the remaining elements in an increasing ordered sequence $\alpha_1, \alpha_2, \dots$ so that

$$0 = \alpha_0 < \alpha_1 < \cdots < \alpha_m < \infty.$$

Then we start with the first trial value $\alpha = \alpha_j$ with $j = m$. If (3.9) holds we terminates the line search method and accept α_m as the step length. Otherwise, we replace j by $\lfloor \frac{j}{2} \rfloor$ iteratively until one of the following cases is occurred:

1. An index $j > 0$ is obtained, for which the condition

$$f(x(\alpha_j)) < f(x)$$

is satisfied.

2. $j = 0$

In the first case, we apply a bisection procedure on the index set to improve index j until it is determined that there is no neighbouring breakpoint than has a smaller function value. Then the corresponding breakpoint α_j is accepted as the step length. In the second case, the breakpoint search failed. Therefore, we perform an ordinary line search iteration by using an efficient line search method with $\alpha \in (0, \alpha_1)$. Note that while performing the bent line search method, the set of active variables is changed either by dropping some variables when $p_i \neq 0$ for some active components, or by adding atleast one variable when a break point is accepted as a step length.

3.3 Overview of the Algorithm

In the case of bound constrained optimization problems, the choice of the search direction is a crucial point for proving global convergence of algorithms. Therefore close attention should be devoted to that. As it is has been mentioned in 1.2, the zigzagging behaviour is the major cause that makes the algorithms inefficient. In this situation, algorithms are unable to identify the set of optimal active bound constraints, therefore, they free and fix the same variable alternatively in a large number of successive iterations. Indeed, we have no full control over whether the Bent line search fixes variables, because often the Bent line search is not able to find a efficient step length $\alpha < \min S$, and consequently accepts a break point as the step length. Therefore, it remains just one way to prevent the bad zigzagging behaviour which is to control the conditions under which the variable are freed.

In order to obtain the fast rate of locally convergence, we have to perform locally steps which are done by means of an efficient unconstrained optimization method over the subspace of nonactive variables; but this is reasonable only if the resulting step is not bent. Therefore such a **local step** is performed whenever in the previous step no variable has been fixed in one of its bounds. On the other hand, if in the previous step a new variable was fixed in its bounds, it does not make sense to perform the local step as the set of active variables is likely to change again. In such a case we perform a **standard step** which is done by performing the bent line search iteration along the steepest descent direction over the subspace of nonactive variables. The search direction p used in a standard step is negative of the scaled gradient direction in the subspace of nonactive variables, which consists of components

$$p_i = \begin{cases} -d_i g_i & \text{if } i \in I, \\ 0 & \text{otherwise}, \end{cases} \tag{3.11}$$

where d_i are positive scaling numbers of the order of square of typical change in the ith component, and I consists of indices corresponding the variables which either are nonactive or can be freed, provided the previous step was a freeing step. In all other cases I consists only of indices corresponding nonactive variables.

Suppose that in the previous step no variable has been fixed in one of its bounds, now we have to make a decision between performing a freeing step or local step. In order to prevent the zigzagging behaviour, first we perform a local step. Then we check whether a new variable became active. If a variable has been active, we perform a standard step. After that, again we check whether a active variable can be freed. If a variable can be free, we do a freeing step. Otherwise, we repeat the local step again.

while implementing the algorithm, We terminate the algorithm whenever one of the following conditions are satisfied

1. The norm of reduced gradient (1.4) is sufficiently small.
2. A large number of iterations have been done.

Note that if the first condition is satisfied, the algorithm terminates at a stationary point of problem and consequently the first optimality condition holds for this point.

In order to analyse the convergence of the algorithm, we need to consider infinity many iterations. Thus, in the following idealized version of the algorithm, we only terminate the algorithm whenever the reduced gradient is zero.

Algorithm 3.3.1. *(Proposed Algorithm)*

Step 1: *(Initialization)*

Given a feasible initial point x^0 with function value f^0 and gradient vector g^0. Set

$$I := \{i : l_i < x_i^0 < u_i \text{ or } (g_{red}^0)_i \neq 0\}$$

Step 2: *(Standard step)*

If the reduced gradient is zero, Stop

Otherwise, if $I = \emptyset$, go to step 4.

Otherwise, Perform the bent line search method along the standard direction over the subspace of variables corresponding to I to compute the new x. Set

$$I := \{i : l_i < x_i < u_i\}.$$

Then, if some bound have been fixed, repeat step 2

Step 3: *(Local step)*

If reduced gradient is zero, stop

Otherwise, perform the bent line search method along the direction determined by a quasi Newton method over the subspace of variables corresponding to I to compute new x. Set

$$I := \{i : l_i < x_i < u_i\}.$$

Then, if a new bound has been fixed, go to step 2.

If no variable can be freed, repeat step 3

Step 4: *(Freeing step)*

Set

$$I := \{i : l_i < x_i < u_i \text{ or } (g_{red})_i \neq 0\},$$

and continue with step 2.

The **initial point** in the first step is either chosen by user, or is given by

$$x_i^0 := \begin{cases} l_i & \text{if } l_i > 0, \\ u_i & \text{if } u_i < 0, \end{cases}$$

as the absolutely smallest point x^0 in the feasible region.

3.4 Convergence Analysis

In this section, we derive the global convergence result for our method. To ensure global convergence, not only the step length must be well chosen but also the search direction must be well chosen. In particular, additional to be feasible, an another requirement is imposed on the angle between the search direction and the steepest descent direction.

Definition 3.4.1. *let $s_k =: x_{k+1} - x_k$ be the step and p_k be a descent search direction, along which x_{k+1} is obtained by performing a line search method. We call this line search method* ***efficient*** *if at each iteration, it produces a step such that satisfies the condition*

$$\inf_{0 \le l \le k} \frac{(f_l - f_{l+1})\|s_l\|^2}{(g_l^T s_l)^2} > 0, \tag{3.12}$$

where $||.||$ is an arbitrary norm in $\mathbb{R}^n$. Equivalently, a line search procedure is referred to as efficient if there exists an $\varepsilon > 0$ such that for all steps $0 \leq l \leq k$, we have

$$f_{l+1} \leq f_l - \varepsilon \frac{g_l^T s_l}{||s_l||^2}.$$

Efficiency of the line search procedure is one of the key requirements which is need to be satisfied in order to obtain global convergence of the algorithms involving the line search strategies.

In order to obtain the convergence result, it is required that the sequence of iterations $\{x_k\}_k$ to be bounded. This condition is satisfied whenever for some index k the level set

$$\{x \in \mathscr{F} : f(x) \leq f(x_k)\}$$

is bounded, where $\mathscr{F}$ is the feasible region of (1.1) consisting of only bound constraints. In most applications, this is satisfied for some iterates x_k.

In the following convergence result, the degenerate stationary point is the only case in which the weakly active constraints might pop in and out of the active set in the zigzagging manner.

Theorem 3.4.1. *Let f be continuously differentiable on the feasible region $\mathscr{F}$. Assume that all search directions p are feasible and also satisfy*

$$p_i = 0 \text{ if either } x_i = l_i, g_i > 0 \text{ or } x_i = u_i, g_i < 0, \tag{3.13}$$

and the ***reduced angle condition***

$$\sup \frac{g_{red}^T(x)p}{||g_{red}(x)||\,||p||} < 0. \tag{3.14}$$

If the sequence of iteration points $\{x_k\}_k$ is bounded then:

(i) *The the reduced gradient $g_{red}(x_k)$ satisfy*

$$\inf_{k \geq 0} ||g_{red}(x_k)|| = 0. \tag{3.15}$$

(ii) *If the algorithm does not terminate but $x_k \to \hat{x}$ for $k \to \infty$, then $\hat{x}$ satisfies the first order optimality conditions $g_{red}(\hat{x}) = 0$, and for all i and sufficiently large k, we have*

$$\begin{aligned} (x_k)_i = \hat{x}_i = l_i \text{ if } g_i(\hat{x}) > 0, \\ (x_k)_i = \hat{x}_i = u_i \text{ if } g_i(\hat{x}) < 0. \end{aligned} \tag{3.16}$$

Note that the condition (3.14) is automatically satisfied for the steepest descent direction, consequently this condition always holds at the standard step.

Proof. Every thing is trivial when the algorithm terminates at a stationary point; therefore, we might assume that infinitely many iterations are done. W.l.o.g., we can assume that $l_i < u_i$ for all i.

(i) Suppose that the line search is efficient only finitely often then there exists an integer L such that for all iterations $k > L$, condition **B1** is violated, and consequently the condition **B2** is satisfied, in particular, the set of breakpoints is not empty, and atleast a new bound is fixed. Hence ultimately, at each iteration some new bounds are fixed. But this means that ultimately, only step 2 is executed. Since the number of variables n is finite and no bound can be freed in the step 2, this can happen only in a finite number of iterations. This is a contradiction with our assumption.

Therefore, the line search is infinitely often efficient, and by (3.4.1), there exists a number $\delta > 0$ such that infinitely often

$$\frac{(f_k - f_{k+1}\|s_k\|^2)}{g_k^T s_k} \geq \delta. \tag{3.17}$$

Now by (3.13) and definition of the reduced gradient, for each component $i \in \{1, ..., n\}$ we have one of the following cases

1. $g_{red}(x_k)_i = (g_k)_i$,
2. $g_{red}(x_k)_i = 0 \leq (g_k)_i$ and $(x_k)_i = l_i$,
3. $g_{red}(x_k)_i = 0 \geq (g_k)_i$ and $(x_k)_i = u_i$.

By (3.13) for the cases 2 and 3 we have

$$(x_{k+1})_i - (x_k)_i = (s_k)_i = \alpha(p_k)_i = 0.$$

Therefore for all cases, $g_{red}(x_k)_i(s_k)_i = (g_k)_i s_i$, and by summing over all $i \in \{1, ..., n\}$ we obtain

$$g_k^T s_k = g_{red}^T(x_k)s_k.$$

Since the sequence of iterations $\{x_k\}_k$ is bounded and the objective function f is continuous, the sequence $f_k := f(x_k)$ is also bounded. Therefore $\hat{f} := \inf f_k$ is finite. Moreover, Since we generate a descent sequence, we have $\lim f_k = \hat{f}$. Hence, after infinitely many iterations satisfying (3.17), we have (writing $-\varepsilon < 0$ for the left hand side of (3.14))

$$\varepsilon\|g_{red}(x_k)\| \leq -\frac{g^T(x_k)p_k}{\|p_k\|} = \frac{|g_{red}^T(x_k)p_k|}{\|p_k\|} = \frac{|g_{red}^T(x_k)s_k|}{\|s_k\|} \leq \sqrt{\frac{f_k - f_{k+1}}{\delta}} \to 0,$$

as $k \to \infty$, and consequently we have

$$\inf_{k \geq 0} \|g_{red}(x_k)\| = 0.$$

(ii) By continuity of the gradient, we have

$$\hat{g} := g(\hat{x}) = \lim_{l \to \infty} g_k.$$

If i is an index for which $\hat{g}_i > 0$ and $(x_k)_i = l_i$, then there exists a number L'' such that $(g_k)_i > 0$ for $k > L''$, analogously, there exist a number L' such that $(g_k)_i < 0$ for all $k > L'$ provided $\hat{g}_i < 0$ and $(x_k)_i = u_i$. Now due to the definition (1.4) of the reduced gradient we have for all iterations $k > L := \max(L', L'')$,

$$(g_{red}(x_k))_i = \begin{cases} 0 & \text{if } (x_k)_i = l_i, \\ 0 & \text{if } (x_k)_i = u_i, \\ (g_k)_i & \text{otherwise.} \end{cases}$$

By part (i), there is a subsequence of $g_{red}(x_k)$ which converges to zero. Now we show that the active variables $(x_k)_i = l_i$ and $(x_k)_i = u_i$ remain fixed at their bounds for infinitely many iterations $k > L$. Since for each active variable $(x_k)_i = l_i\big((x_k)_i = u_i\big)$ we have $(g_k)_i > 0\big((g_k)_i < 0\big)$ for all $k > L$, all active variables are strongly active. Therefore, they can not be freed anymore. This fact implies that $(x_k)_i = l_i\big((x_k)_i = u_i\big)$ for all $k > L$, and consequently, $\hat{x}_i = \lim_{k \to \infty}(x_k)_i = l_i\big(\hat{x}_i = \lim_{k \to \infty}(x_k)_i = u_i\big)$. Therefore the conditions (3.16) are satisfied.

From (3.16) and the definition of the reduced gradient, we conclude that $g_{red}(\hat{x}) = \lim g_{red}(x_k) = 0$. Hence the first order optimality condition is satisfied.

□

3.5 Local Step

In this section we are dealing with computation of the search direction in the local step of the algorithm 3.3.1. As we have mentioned, in order to obtain fast locally convergence we apply a limited quasi Newton method for unconstrained optimization problems in the local step. To be more accurate, suppose that while performing algorithm 3.3.1, no variable has been fixed after execution of the standard step and also the feasible point x with $I := \{i : l_i < x_i < u_i\}$ has been determined. Now in local step, the quasi Newton search direction is computed by solving the following optimization problem

$$\begin{aligned} &\text{minimize} \quad p^T Bp + g^T p + f(x) \\ &\text{subject to} \quad p_i = 0, \quad i \notin I, \end{aligned} \tag{3.18}$$

where g is the gradient of f at feasible point x, and B is a limited memory SR1 approximation of the Hessian matrix $\nabla^2 f$. Now the problem (3.18) can be transformed to the following unconstrained problem defined over the subspace of nonactive variables corresponding to the set I

$$\min_{\hat{p} \in \mathbb{R}^{|I|}} \hat{p}^T \hat{B} \hat{p} + \hat{g}^T \hat{p}, \tag{3.19}$$

where

$$\hat{B} := Z^T BZ \tag{3.20}$$

is the reduced Hessian matrix, and

$$\hat{g} := Z^T g \tag{3.21}$$

represents the projection of the gradient vector $g(x)$ onto the subspace of nonactive variables corresponding to I, and Z is a matrix whose columns are unit vectors spanning the subspace of nonactive variables corresponding to I. As we know from general theory of unconstrained optimization, the solution of (3.19) is computed by

$$\hat{p}_f := -\hat{B}^{-1} \hat{g}. \tag{3.22}$$

Hence the solution of (3.18) is determined by

$$p_f := Z \hat{p}_f. \tag{3.23}$$

Since the matrix $\hat{B}$ is computed by the limited memory SR1 method, we have no guarantee that the search direction p_f is a descent direction. However, from our convergence result, the search directions p_f must satisfy the angle condition (3.14). In following we will explain, how this search direction is regularized in order to satisfy the condition (3.14) in case it fails to hold for (3.23). First we give a description about updating the limited memory SR1 matrices.

3.5.1 Limited Memory SR1 Update

Similar to the BFGS matrices, the compact representation of SR1 matrices can be derived. This compact representation introduced by BYRD, NOCEDAL and

SCHNABEL [7] is similar to ones developed for the BFGS formula, except that under some conditions it requires less storage. The SR1 matrices are updated at each iteration k by the following update formula

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{(y_k - B_k s_k)^T s_k}, \tag{3.24}$$

where

$$s_k := x_{k+1} - x_k, \quad y_k := g_{k+1} - g_k.$$

It should be mentioned that the above update is well defined only if

$$(y_k - B_k s_k)^T s_k \neq 0.$$

In our implementation it has been observed that, the SR1 method does not break down and performs well if the update is skipped whenever the denominator of (3.24) is small. In particular, the update formula (3.24) is applied whenever the following condition

$$|(y_k - B_k s_k)^T s_k| \geq 10^{-8} \|s_k\| \|y_k - B_k s_k\| \tag{3.25}$$

holds. Now we present the compact representation of SR1 matrices by means of the following theorem.

Theorem 3.5.1. *(Compact Representation of SR1 [7, p, 20]) Suppose that the symmetric matrix B_0 is updated k times by the SR1 formula* (3.24) *using the pairs $\{s_i, y_i\}_{i=0}^{k-1}$, and assume that each update is well defined. In another words, we have $(y_j - B_j s_j)^T s_j \neq 0$ for all $j = 1, \dots, k-1$. Then the symmetric SR1 matrix B_k can be expressed as*

$$B_k = B_0 + (Y_k - B_0 S_k) N_k^{-1} (Y_k - B_0 S_k)^T, \tag{3.26}$$

where the correction matrices $S_k, Y_k \in \mathbb{R}^{n \times k}$ have the form

$$S_k := [s_0, \dots, s_{k-1}], \quad Y_k = [y_0, \dots, y_{k-1}], \tag{3.27}$$

the matrix

$$N_k := D_k + C_k + C_k^T - S_k^T B_0 S_k \tag{3.28}$$

is nonsingular, $C_k \in \mathbb{R}^{k \times k}$ and diagonal matrix $D_k \in \mathbb{R}^{k \times k}$ are defined by

$$(C_k)_{i,j} := \begin{cases} s_{i-1}^T y_{j-1} & \textit{if } i > j \\ 0 & \textit{otherwise,} \end{cases} \tag{3.29}$$

$$D_k := \operatorname{diag}[s_0^T y_0, \dots, s_{k-1}^T y_{k-1}]. \tag{3.30}$$

Similar to the limited memory BFGS method described in 2.6.2, we can develop the above update formula for the limited memory SR1 matrices. To see this, the initial matrix B_0 is replaced with a basic matrix $B_k^0 = \omega_k I$ at the kth iteration, where ω_k is positive scaling number or can be computed by

$$\omega_k := \frac{y_{k-1}^T y_{k-1}}{s_{k-1}^T y_{k-1}}. \tag{3.31}$$

Moreover, in the case of limited memory the correction matrices Y_k and S_k contain only the r most recent correction vectors

$$S_k := [s_{k-r}, \dots, s_{k-1}], \quad Y_k := [y_{k-r}, \dots, y_{k-1}], \tag{3.32}$$

where r is a fixed integer number. At each iteration, these matrices are updated by removing the oldest correction pair and inserting the new correction pair. Now we can express the limited memory SR1 matrices as follows:

$$B_k = \omega_k I + U_k N_k^{-1} U_k^T, \tag{3.33}$$

where

$$U_k := Y_k - \omega_k S_k, \tag{3.34}$$

$$N_k := D_k + C_k + C_k^T - \omega_k S_k^T S_k, \tag{3.35}$$

C_k and D_k belong to $\mathbb{R}^{r \times r}$ and are defined as

$$(C_k)_{i,j} := \begin{cases} s_{k-r-1+i}^T y_{k-r-1+j} & \text{if } i > j \\ 0 & \text{otherwise,} \end{cases} \tag{3.36}$$

$$D_k := \operatorname{diag}[s_{k-r}^T y_{k-r}, \dots, s_{k-1}^T y_{k-1}]. \tag{3.37}$$

Note that the matrix N_k belongs to $\mathbb{R}^{r \times r}$, furthermore, the positive integer r is chosen to be small. Therefore, the computation of N_k is cheap at each iteration.

3.5.2 Computing the Search Direction

Now we are in the position that we can explain how the quasi Newton search direction is computed in order to satisfy the condition (3.14). As it has been mentioned before the first quasi Newton direction is computed by solving the following linear system

$$\hat{B}\hat{p} = -\hat{g} \tag{3.38}$$

By using (3.33) and (3.20), we can rewrite the above linear system as

$$(\omega \hat{I} + \hat{U} N^{-1} \hat{U}^T)\hat{p} = -\hat{g}, \tag{3.39}$$

where $\hat{U} := Z^T U$ and $\hat{I} \in \mathbb{R}^{r\times r}$ is the identity matrix. Note that the subscription k of the outer iteration is eliminated for simplicity. Now by using (3.39) and defining vector

$$h := N^{-1}\hat{U}^T \hat{p}, \tag{3.40}$$

we have

$$\hat{p} = -\frac{1}{\omega}\hat{g} - \frac{1}{\omega}\hat{U}h. \tag{3.41}$$

Furthermore, by substituting (3.41) in (3.40) we have

$$h = N^{-1}\hat{U}^T(-\frac{1}{\omega}\hat{g} - \frac{1}{\omega}\hat{U}h), \tag{3.42}$$

where (3.42) is a linear system containing only an unknown vector h. By solving this linear system we have

$$h = (N + \frac{1}{\omega}\hat{U}^T\hat{U})^{-1}(-\frac{1}{\omega}\hat{U}^T\hat{g}). \tag{3.43}$$

Now we can define our search direction as a function of κ as follows

$$\hat{p}(\kappa) = -\frac{1}{\omega}\hat{g} - \kappa q, \tag{3.44}$$

where

$$q := \frac{1}{\omega}\hat{U}h.$$

In our proposed method, first we set $\kappa = 1$ and test whether the condition

$$\frac{\hat{g}^T\hat{p}(\kappa)}{\|\hat{g}\|\|\hat{p}(\kappa)\|} \le -\delta, \tag{3.45}$$

is satisfied for a fixed positive number $\delta \in (0,1)$. If it is satisfied then $p(1)$ is accepted as the search direction in the local step. But if the above condition does not hold for $\kappa = 1$, we choose κ such that the condition (3.45) will be satisfied for chosen κ.

Now, it remains to show how κ is chosen in order to satisfy (3.45). By substituting (3.44) into the condition (3.45) we have

$$\frac{-\frac{1}{\omega}\|\hat{g}\|^2 - \kappa\hat{g}^T q}{\|\hat{g}\|\| - \frac{1}{\omega}\hat{g} - \kappa q\|} \le -\delta, \tag{3.46}$$

and consequently

$$\|\hat{g}\|\frac{1}{\delta}\Big(\frac{1}{\omega} + \kappa\frac{\hat{g}^T q}{\|\hat{g}\|}\Big) \ge \| - \frac{1}{\omega}\hat{g} - \kappa q\|. \tag{3.47}$$

Now by defining

$$\tau := \frac{1}{\delta}\Big(\frac{1}{\omega} + \kappa\frac{\hat{g}^T q}{\|\hat{g}\|^2}\Big), \tag{3.48}$$

we can rewrite (3.47) as

$$\|\hat{g}\|\tau \geq \|\frac{1}{\omega}\hat{g} + \kappa q\|. \tag{3.49}$$

Since the both sides the above inequality are nonnegative, we can compute the squares of them without switching the inequality direction as follows

$$\|\hat{g}\|^2\tau^2 \geq \frac{1}{\omega^2}\|\hat{g}\|^2 + \kappa^2\|q\|^2 + \frac{2\kappa}{\omega}\hat{g}^T q. \tag{3.50}$$

By computing κ from (3.48), we obtain

$$\kappa := (\delta\tau - \frac{1}{\omega})\frac{\|\hat{g}\|^2}{\hat{g}^T q} \tag{3.51}$$

By using (3.50) and (3.51) and eliminating $\|\hat{g}\|^2$ from the both sides of the inequality we have

$$\tau^2 \geq -\frac{1}{\omega^2} + \frac{2\delta\tau}{\omega} + (\delta\tau - \frac{1}{\omega})^2\frac{\|\hat{g}\|^2\|q\|^2}{(\hat{g}^T q)^2}. \tag{3.52}$$

Now by adding $-\delta^2\tau^2$ to the both sides of the above inequality we obtain

$$(1-\delta^2)\tau^2 \geq -\frac{1}{\omega^2} + \frac{2\delta\tau}{\omega} - \delta^2\tau^2 + (\delta\tau - \frac{1}{\omega})^2\frac{\|\hat{g}\|^2\|q\|^2}{(\hat{g}^T q)^2}, \tag{3.53}$$

which can be express as

$$(1-\delta^2)\tau^2 \geq (\delta\tau - \frac{1}{\omega})^2\Big(\frac{\|\hat{g}\|^2\|q\|^2}{(\hat{g}^T q)^2} - 1\Big), \tag{3.54}$$

Since $(1-\delta^2)$ is positive we can rewrite (3.54) as

$$\frac{\tau^2}{(\delta\tau - \frac{1}{\omega})^2} \geq \Big(\frac{\|\hat{g}\|^2\|q\|^2 - (\hat{g}^T q)^2}{(1-\delta^2)(\hat{g}^T q)^2}\Big), \tag{3.55}$$

which implies that

$$\frac{\tau}{\frac{1}{\omega} - \delta\tau} \geq \Delta, \tag{3.56}$$

where

$$\Delta := \sqrt{\frac{\|\hat{g}\|^2\|q\|^2 - (\hat{g}^T q)^2}{(1-\delta^2)(\hat{g}^T q)^2}}. \tag{3.57}$$

By using (3.56), we have

$$\begin{aligned} & \tau \geq (\frac{1}{\omega} - \delta\tau)\Delta, \\ \Leftrightarrow \quad & \tau(1+\Delta\delta) \geq \frac{\Delta}{\omega}, \\ \Leftrightarrow \quad & \tau \geq \frac{\Delta}{(1+\Delta\delta)\omega}. \end{aligned} \tag{3.58}$$

By substituting (3.48) into (3.58), we obtain

$$\frac{1}{\delta}\Big(\frac{1}{\omega} + \kappa\frac{\hat{g}^T q}{\|\hat{g}\|^2}\Big) \geq \frac{\Delta}{(1+\Delta\delta)\omega}, \tag{3.59}$$

which yields

$$\kappa\frac{\hat{g}^T q}{\|\hat{g}\|^2} \geq \frac{\delta\Delta}{(1+\delta\Delta)\omega} - \frac{1}{\omega} = \frac{-1}{(1+\delta\Delta)\omega}, \tag{3.60}$$

and consequently

$$\begin{cases} \kappa \geq \Big(\dfrac{-1}{(1+\Delta\delta)\omega}\Big)\dfrac{\|\hat{g}\|^2}{\hat{g}^T q} & \text{if } \hat{g}^T q > 0, \\ \kappa \leq \Big(\dfrac{-1}{(1+\Delta\delta)\omega}\Big)\dfrac{\|\hat{g}\|^2}{\hat{g}^T q} & \text{if } \hat{g}^T q < 0. \end{cases} \tag{3.61}$$

In our proposed method, we choose

$$\kappa = \Big(\frac{-1}{(1+\Delta\delta)\omega}\Big)\frac{\|\hat{g}\|^2}{\hat{g}^T q}, \tag{3.62}$$

where Δ is computed from (3.57). After κ has been computed, we define the search direction as

$$p := Z\hat{p}(\kappa) = -Z(\frac{1}{\omega}\hat{g} + \kappa q) \tag{3.63}$$

Now we can summarize the description outlined above into the following algorithm

Algorithm 3.5.1. *(Local step)*

Step 2: *(Initialization)*

Given ω, U, N, a fixed number $\delta \in (0,1)$ and the gradient g of the current iterate x. Assume that the set of free variable I at x has been determined.

Step 1: *(Computing $\hat{p} = \hat{B}^{-1}\hat{g}$)*

Set

$$\begin{aligned}
\theta &:= \frac{1}{\omega}\\
\hat{g} &:= Z^T g\\
\hat{U} &:= Z^T U\\
T &:= N + \theta \hat{U}^T \hat{U}\\
h &:= -\theta \hat{U}^T \hat{g}\\
h &:= T \backslash h\\
q &:= \theta \hat{U} h\\
\hat{p} &:= -\theta \hat{g} - q
\end{aligned}$$

Step 3: *(Testing the angle condition* (3.45)*)*

Set

$$\begin{aligned}
n_{\hat{g}} &:= \|\hat{g}\|\\
n_{\hat{p}} &:= \|\hat{p}\|
\end{aligned}$$

If $\frac{\hat{g}^T \hat{p}}{n_{\hat{p}} n_{\hat{g}}} \leq \delta$, then go to step 4

Otherwise,set

$$\begin{aligned}
\eta &:= \hat{g}^T q\\
n_q &:= \|q\|\\
\Delta &:= \sqrt{\frac{n_{\hat{g}}^2 n_q^2 - \eta^2}{\eta^2(1-\delta^2)}}\\
\kappa &:= \left(\frac{-\theta}{1+\Delta\delta}\right)\frac{n_{\hat{g}}^2}{\eta}\\
\hat{p} &:= -\theta \hat{g} - \kappa q
\end{aligned}$$

Step 4: *(Computing the search direction)*

Set

$$p := Z\hat{p}$$

Chapter 4

Numerical Experiments

4.1 Introduction

In this chapter, we test our proposed algorithm by solving a list of bound constrained optimization problems. These problems are selected from CUTEr collection [14] and most of them are large scale problems. Then, the performance of our algorithm is demonstrated for different choices of the limited memory parameter r. Finally, we compare the performance of our algorithm with L-BFGS-B.
Throughout our implementations, we use

$$\|g_{red}(x)\|_\infty \leq 10^{-5} \tag{4.1}$$

as the termination condition. Moreover, while performing the bent line search approach if the break point search fails, we perform the efficient line search by means of the routine of MORÉ and THUENTE [23] which tries to impose the strong Wolfe conditions. After testing a small number of test problems, we have realized that the algorithm works well with the choice of $\delta = 0.01$ for the most of problems, where δ is defined in the angle condition (3.14). All of codes have been written in MATLAB and we have used the new MATLAB interface of CUTEr testing environment from `http://www.cuter.rl.ac.uk/download.html`.

4.2 Numerical Results

First, we have tested our algorithm with different choices of the limited memory parameter r which represents the number of stored correction pairs in the correction matrices. Namely, we set $r = 3$, $r = 5$, $r = 17$ and $r = 29$. The corresponding numerical results are represented by table 4.1, where the first column in this table reveals the names of test problems, the second column represents the dimensions

n of the problems. We record the total run time $Time$, the number of function evaluations nfe and the number of gradient evaluations nge for each test problem.

Problem	n	$r = 3$			$r = 5$			$r = 17$			$r = 29$		
		Time	nfe	nge	Time	nfe	nge	Time	nfe	nge	Time	nfe	nge
TORSION1	5476	1.962	2880	318	1.313	1933	207	2.225	2049	241	3.685	2062	242
TORSION2	5476	1.884	2406	266	1.222	1711	196	2.034	1780	211	5.246	2786	342
TORSION3	5476	1.221	1395	191	1.119	1590	230	1.206	1092	152	2.700	1487	212
TORSION4	5476	1.144	1382	187	0.666	929	128	0.928	844	114	3.321	1807	267
TORSION6	5476	0.610	638	97	0.377	501	70	0.611	514	74	0.643	446	66
BDEXP	5000	0.081	2	2	0.015	2	2	0.005	2	2	0.010	2	2
BQPGASIM	50	0.220	347	64	0.035	249	43	0.033	173	30	0.040	173	30
JLBRNGA	10000	9.422	8158	853	6.703	5543	569	13.467	6354	671	35.113	10719	1217
LINVERSE	1999	10.479	27162	3187	3.168	8334	810	5.040	9284	842	6.640	8395	740
MCCORMCK	5000	0.408	221	41	0.364	273	80	0.326	184	34	0.308	168	32
PROBPENL	500	0.436	5	3	0.026	5	3	0.004	5	3	0.002	5	3
NONSCOMP	5000	0.151	8	4	0.025	8	4	0.019	8	4	0.006	8	4
OBSTCLAE	10000	9.303	8095	657	4.668	3992	363	8.196	4261	370	13.395	4682	363
OBSTCLAL	10000	3.728	3144	290	2.973	2569	237	3.647	1946	178	5.330	1821	174
OBSTCLBL	10000	3.696	3016	258	3.611	3022	259	5.011	2663	218	7.445	2644	215
OBSTCLBM	10000	3.265	2683	238	2.439	2036	174	4.721	2461	233	6.831	2350	218
OBSTCLBU	10000	3.811	3197	287	3.696	3120	270	4.441	2372	207	5.655	2033	170

Table 4.1: Results of the algorithm 3.3 for various choices of the limited memory parameter r

An important observation on the results in table 4.1 is that on these problems in the case of $r = 5$ the algorithm works more efficient than in the other cases. In other word, in the case of $r = 5$, the algorithm requires less time, function evaluations and gradient evaluations for the most of problems.

In order to analyse the result of table 4.1, we use the performance profile introduced by DOLAN and MORÉ [11]. This performance profile is defined by

$$\rho_{Alg}(\tau) = \frac{\text{number of test problems for which} \quad \log_2(r_{P,Alg}) \leq \tau}{\text{number of all test problems}}, \tag{4.2}$$

where $\tau \geq 0$, and $r_{P,Alg}$ is the **performance ratio** defined by

$$r_{P,Alg} = \frac{\text{the time required to solve problem } P \text{ by algorithm } Alg}{\text{the shortest time required to solve problem } P}. \tag{4.3}$$

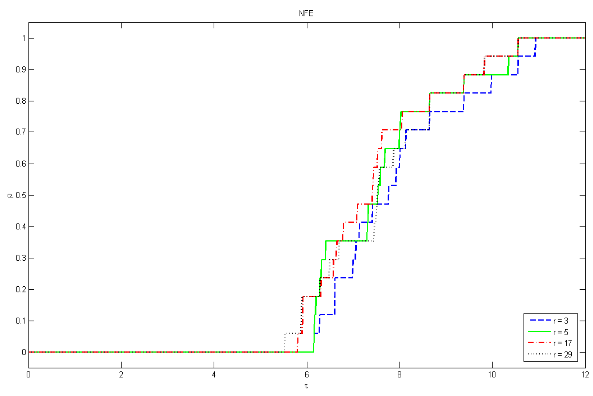
Note that, the performance ratio (4.2) can be computed with respect to an arbitrary performance measure, for instance, the number of function evaluations or the number of gradient evaluations. Furthermore, if an algorithm Alg fails to solve a test problem P, the corresponding performance ratio $r_{P,Alg}$ is set to be infinity (or a large number).

The nondecreasing, piecewise constant function $\rho_{Alg} : \mathbb{R} \to [0,1]$ is the cumulative distribution function for performance ratio. In the performance profiles the value of $\rho_{Alg}(\tau)$ at $\tau = 0$ yields the percentage of the test problems for which the algorithm Alg is the best and the value of $\rho_{Alg}(\tau)$ for large enough values of τ gives the percentage of the test problems that the algorithm Alg can solve. In addition, the value of $\rho_{Alg}(\tau)$ at $\tau = 1$ gives the probability that the algorithm Alg will win over the rest of algorithms. Therefore, if the number of wins is of our interest, we need to compare the value of $\rho_{Alg}(1)$ for all of the algorithms.

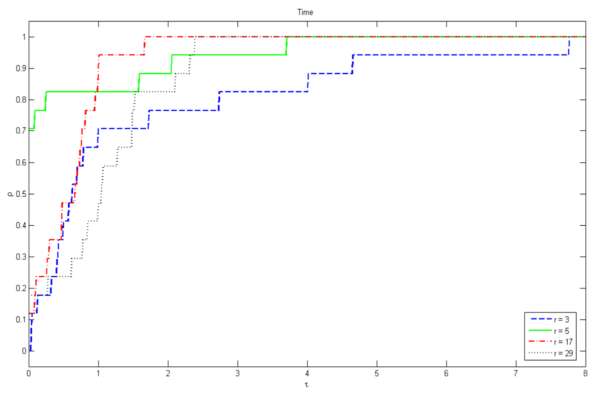
The figures 4.1 show the performance profile of the algorithm 3.3 based on the table 4.1. These figures reveal the relatively performance of the algorithm for different setting of the limited memory parameter r with respect to the performance measures including the number of objective function evaluations, the number of gradient evaluations and the required time. Furthermore, the relative efficiency of the algorithm in each case can be directly observed from the figures. In other word, the higher is the particular curve, the better is the corresponding case for the algorithm.

As we can see from the figure 4.1(b), in the case $r = 5$ the algorithm has the most wins, consequently, it has the highest probability for being the optimal solver when the computational time is compared. However as the figures 4.1(c) , 4.1(a) reveal, the performance of the algorithm is almost the same in all the cases when the number of function or gradient evaluations is compared. In the case of $r = 3$ and $r = 5$ updating the Hessian matrix is cheaper than the other cases while in these cases the algorithm needs more iterations to solve the problem in comparison to the cases $r = 17$ and $r = 29$. Since in cases $r = 19$ and $r = 29$ the updating Hessian matrix require longer time in comparison to evaluation of the objective function and gradient, the algorithm has the better performance in the case of $r = 5$. Although in case $r = 3$, updating the Hessian matrix is cheaper than in the rest of cases, the algorithm needs more iterations to reach the stationary point. This is due to the fact that in the case $r = 3$, the limited memory Hessian matrix is updated by using just the three most recent correction vectors at each iteration. Therefore, the limited memory Hessian matrix can not be good approximation for the Hessian matrix and consequently the algorithm performs many iterations to solve the problem.

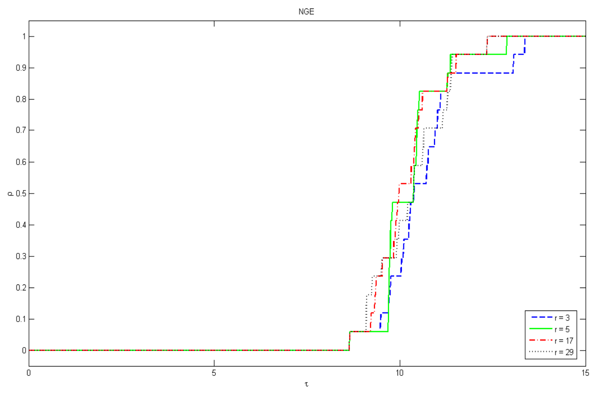
In the next part of this chapter we compare the performance of our algorithm in



(a) Function evaluations



(b) Time



(c) Gradient evaluations

Figure 4.1: Performance profiles of the algorithm 3.3 based on the table 4.1

tow cases with L-BFGS-B method described in 2.6. The L-BFGS-B codes is written in MATLAB from [6]. The test functions are the same test function presented in table 4.1. we test our algorithm with different bent line search approaches:

Alg 1: The bent line search used in the algorithm performs the bisection procedure to improve the index at each iteration.

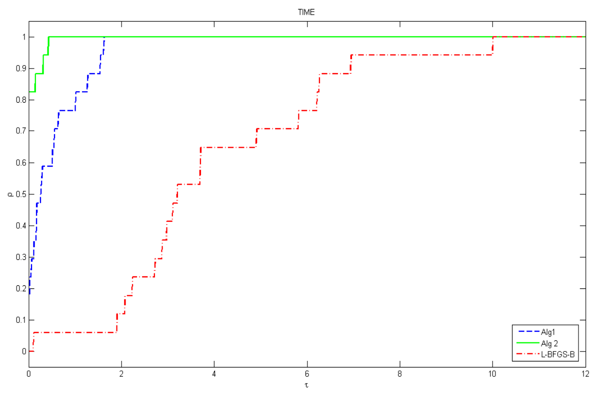
Alg 2: The bent line search used in the algorithm does not perform the bisection procedure.

The figure 4.2 shows the performance profiles of our algorithm in comparison to L-BFGS-B. As performance measures, we use number of objective function evaluations (4.2(b)), number of gradient evaluations (4.2(c)), and required computational time (4.2(a)). In figure 4.2(a) we see that our algorithm in both of cases dominates L-BFGS-B. This stems from the following facts: first, updating the limited memory SR-1 matrices is cheaper that updating the limited memory BFGS matrices. In addition, at each iteration our algorithm needs to solve just a system of linear equations of order r, while at each iteration the L-BFGS-B algorithm needs to compute the inverse of a $2r \times 2r$ matrix and a linear system of order r for computing the direction in direct primal method. Finally, after solving the chosen test functions by these algorithms, we have realized that the L-BFGS-B needs more iterations compared to both variants of our algorithm. This means that our algorithm is able to identify the optimal active set of bound constraints faster.

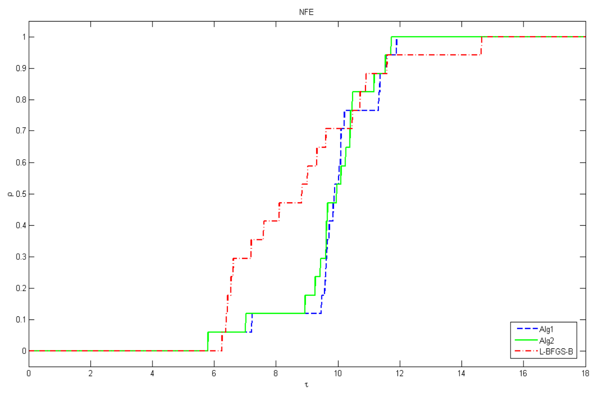
In contrast to figure 4.2(a), as it is illustrated in figure 4.2(c), the L-BFGS-B dominates our algorithms when the number of objective function evaluations is compared. While performing the bent line search, our algorithm evaluates objective function at many breakpoints. Specially in the case of large scale problem, the number of break point is usually large at each iteration. We consider this as a major drawback of our algorithm.

As we have mentioned before, our algorithms compared to L-BFGS-B need less iterations to solve most of the chosen test problems. This fact can be easily observed from figure 4.2(c). As we can see from figure 4.2(c), the curves corresponding to Alg1 and Alg2 are higher than the red curve corresponding to L-BFGS-B. Moreover, in all of the algorithms, we have only a gradient evaluation at each iteration.

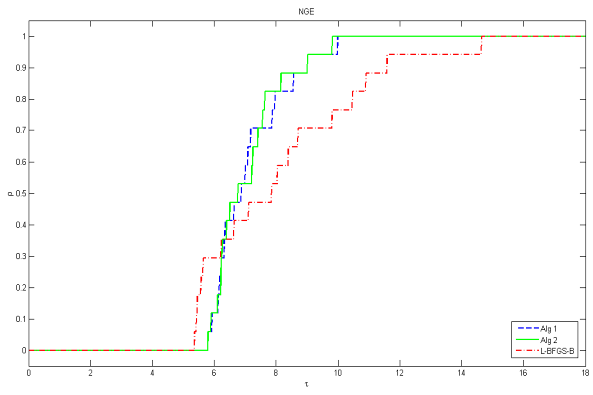
Now if we consider the two above variants of our algorithm, it can be easily observed that the Alg2 is faster than Alg1 with respect to the chosen test functions. This is due the fact that at each iteration Alg1 performs many objective function



(a) Time



(b) Function evaluations



(c) Gradient evaluations

Figure 4.2: Performance profiles of the algorithm 3.3 compared to L-BFGS-B

evaluations during the bisection procedure. This consideration is demonstrated by figure 4.2(b).

Chapter 5

Conclusion and Future Work

In this thesis, we introduced a new quasi Newton method for solving large scale bound constrained optimization problems. This method contains a number of features that make it distinguished from the other algorithms: firstly, we use the combination of the steepest descent directions and quasi Newton directions in order to find the optimal active variables. Moreover, in order to prevent the zigzagging behaviour, a new schema for active set method is presented. Lastly, the quasi Newton direction is computed by limited memory SR-1 matrices. As it is known the SR1 matrices are not necessarily positive definite. Therefore the quasi Newton direction computed by these matrices need not to be a descent direction. In such case we regularize the direction such that it will be a descent direction. We reported numerical results of our algorithm applied to a list of bound constrained problems from CUTEr. In addition, the relative performance of our algorithm compared to L-BFGS-B have been demonstrated.

Although our proposed algorithm appears to be effective already, there of some further points to be considered in order to make algorithm more efficient. As we have mentioned in previous chapter, the bent line search method involved in our algorithm requires too many objective function evaluations to find the step length at each iteration. This fact makes our algorithm inefficient in case of large scale problems. In future work we expect to propose a new bent line search that requires less function evaluations. Moreover, during the implementation of our algorithm, we have observed for a few test problems at some iterations the linear system (3.43) is ill-conditioned. As further work we would like to equip our algorithm with a preconditioning method.

Bibliography

[1] R. Andreani, A. Friedlander, and J. M. Martínez. On the solution of finite-dimensional variational inequalities using smooth optimization with simple bounds. *J. Optim. Theory Appl.*, 94:635–657, September 1997.

[2] Dimitri P. Bertsekas. On the Goldstein-Levitin-Polyak gradient projection method. *IEEE Transactions on Automatic Control*, 21:174–184, 1976.

[3] Dimitri P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA 02178-9998, second edition, 1999.

[4] Dimitri P. Bertsekast. Projected newton methods for optimization problems with simple constraints. *SIAM J.Control, Opimization*, 20(2):221–246, 1982.

[5] Ernesto G. Birgin, Ivan Chambouleyron, and José Mario Martínez. Estimation of the optical constants and the thickness of thin films using unconstrained optimization. *J. Comput. Phys.*, 151(2):862–880, May 1999.

[6] Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190, 1995.

[7] Richard H Byrd, Jorge Nocedal, and Robert B Schnabel. Representations of quasi-newton matrices and their use in limited memory methods. *Mathematical Programming*, 63(1-3):129–156, 1994.

[8] Paul Calamai and Jorge Moré. Projected gradient methods for linearly constrained problems. *Mathematical Programming*, 39(1):93–116, September 1987.

[9] A R Conn, N I M Gould, and Ph L Toint. Global convergence of a class of trust region algorithms for optimization with simple bounds. *SIAM Journal on Numerical Analysis*, 25(2):433, 1988.

[10] Andrew R Conn, Nicholas I M Gould, and Philippe L Toint. Testing a class of methods for solving minimization problems with simple bounds on the variables. *Mathematics of Computation*, 50(182):399–430, 1988.

[11] Elizabeth D Dolan and Jorge J Mor. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):13, 2001.

[12] J. C. Dunn. On the convergence of projected gradient processes to singular critical points. *J. Optim. Theory Appl.*, 55:203–216, 1987.

[13] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Practical optimization*. Academic Press Inc. [Harcourt Brace Jovanovich Publishers], London, 1981.

[14] Nicholas I. M. Gould, Dominique Orban, and Philippe L. Toint. Cuter and sifdec: A constrained and unconstrained testing environment, revisited. *ACM Trans. Math. Softw.*, 29(4):373–394, December 2003.

[15] A. Griewank and Ph. L. Toint. Partitioned variable metric updates for large structured optimization problems. *Numerische Mathematik*, 39(1):119–137, 1982.

[16] W. W. Hager. Dual techniques for constrained optimization. *J. Optim. Theory Appl.*, 55(1):37–71, October 1987.

[17] W. W. Hager. Analysis and implementation of a dual algorithm for constrained optimization. *J. Optim. Theory Appl.*, 79(3):427–462, December 1993.

[18] Y. Lin and C. W. Cryer. An alternating direction implicit algorithm for the solution of linear complementarity problems arising from free boundary problems. *Appl. Math. Optimization*, 13(1):1–17, 1987.

[19] P Lötstedt. Solving the minimal least squares problem subject to bounds on the variables. *BIT*, 24:206–224, 1984.

[20] Jos Mario Martnez. Box-quacan and the implementation of augmented lagrangian algorithms for minimization with inequality constraints. 1998.

[21] J J Moré and G Toraldo. Algorithms for bound constrained quadratic programming problems. *Numerische Mathematik*, 55(4):377–400, 1989.

[22] J J Moré and G Toraldo. On the solution of large quadratic programming problems with bound constraints. *SIAM Journal on Optimization*, 1(1):93–113, 1991.

[23] Jorge J. Moré and David J. Thuente. Line search algorithms with guaranteed sufficient decrease. *ACM Trans. Math. Softw.*, 20(3):286–307, September 1994.

[24] Arnold Neumaier. Optimization theory and algorithms. University Lecture, November 2011.

Behzad Azmi

Curriculum Vitae

Date of Birth: 18.09.1982
Place of Birth: Shemiran, Iran
Nationality: Iranian
Marital Status: not married

Education

Oct. 2009– Now.	**Master of Science in Applied Mathematics and Scientific Computing**, *University of Vienna*, Vienna, Austria.
Sep. 2006– Jan. 2009	**Master of Science in Applied Mathematics**, *Islamic Azad University(Science and Research branch)*, Tehran, Iran, *17.73/20.00* or *A*.
Sep. 2000– Jul. 2006	**Bachelor of Science in Applied Mathematics**, *Islamic Azad University*, Tehran, Iran.

Master Thesis

Title	*Benchmarking in Data Envelopment Analysis with Fuzzy Data*
Advisor	Dr. Farhad Hosseinzadeh Lotfi

Publication

G.R. Jahanshahloo, F. Hosseinzadeh Lotfi, and B. Azmi (2009). "Target Setting in the Case of Bounded Aggregate Outputs," Int. Journal of Math. Analysis, Vol. 3, 19, 931 - 942.

Language

Persian	**native speaker**	
English	**fluent**	
German	**fluent**	*Österreichisches Sprachdiplom B2*