Lecture-Mate of
30E00100: Optimization

Kalyanmoy Deb
Helsinki School of Economics
Second Term in Spring 2008

Preface

This document is prepared for 30E00100 course on 'Optimization'. In this document, I have tried to put together a number of classical numerical optimization algorithms which provide a foundation on optimization basics. Since there is no single optimization method efficient for every problem, I have discussed a number of widely used techniques. But no attempt is made to outline every optimization algorithm in use todate. Most of the algorithms and some examples problems are copied from my book entitled 'Optimization for Engineering Design' published by Prentice-Hall of India in 1995 which is still available from http://www.amazon.co.uk. Hence, the reproduction or circulation of this material is strictly prohibited.

March 3, 2008

Kalyanmoy Deb
Email: Kalyanmoy.Deb@hse.fi

Contents

1 Introduction **iv**
1.1 Steps in an Optimization Simulation . . . iv
1.2 A Few Important Matters . . . iv

2 Single-variable Optimization **2**
2.1 Optimality Criteria . . . 2
2.2 Braketting Algorithms . . . 2
2.2.1 Exhaustive Search . . . 2
2.2.2 Bounding Phase Method . . . 3
2.3 Region-Elimination Methods . . . 3
2.3.1 Interval Halving . . . 4
2.3.2 Golden Section Search . . . 5
2.4 Point-Estimation Methods . . . 7
2.4.1 Successive Quadratic Estimation Method . . . 7
2.5 Methods Requiring Derivatives . . . 8
2.5.1 Newton-Raphson Method . . . 8
2.5.2 Bisection Method . . . 8
2.5.3 Secant Method . . . 9
2.5.4 Cubic Search Method . . . 9

3 Multi-variable Optimization **11**
3.1 Optimality Criteria . . . 11
3.2 Direct Search Methods . . . 11
3.2.1 Simplex Search Method . . . 11
3.2.2 Hooke-Jeeves Pattern Search Method . . . 13
3.2.3 Powell's Conjugate Direction Method . . . 14
3.3 Gradient-based Methods . . . 18
3.3.1 Numerical Gradient Approximation . . . 18
3.3.2 Cauchy's Method (Steepest Descent) . . . 19
3.3.3 Newton's Method . . . 19
3.3.4 Marquardt's Method . . . 19
3.3.5 Conjugate Gradient Method . . . 20
3.3.6 Variable-Metric Method (Davidon-Fletcher-Powell method) . . . 20

4 Constrained Optimization **24**
4.1 Kuhn-Tucker Conditions . . . 24
4.2 Transformation Methods . . . 25
4.2.1 Penalty Function Method . . . 25
4.2.2 Method of Multipliers . . . 30
4.3 Constrained Direct Search . . . 31
4.3.1 Random Search Methods . . . 31

4.4 Linearization Methods 32
4.4.1 Frank-Wolfe Algorithm: 32
4.4.2 Cutting Plane Method 33
4.5 Method of Feasible Directions 35
4.6 Generalized Reduced Gradient Method 42
4.7 Quadratic Approximation Method 44

5 Integer Programming **45**
5.1 Branch-and-Bound Algorithm 45

6 Nontraditional Optimization Techniques **53**
6.1 Genetic Algorithms 53
6.1.1 Fundamental Differences with Traditional methods 53
6.1.2 Reproduction Operator 54
6.1.3 Crossover Operator 54
6.1.4 Mutation Operator 55

A An Introduction to Linear Programming **62**
A.1 Simplex Method 63
A.2 Artificial Variables and Dual Phase Method 68

Chapter 1

Introduction

1.1 Steps in an Optimization Simulation

The following steps are usually involved in an optimal design formulation process.

Step 1: Need for optimization

Step 2: Choose decision variables

Step 3: Formulate constraints

Step 4: Formulate objective functions

Step 5: Set up variable bounds

Step 6: Choose an optimization algorithm

Step 7: Obtain solution(s)

Step 8: If not satisfied, move to a suitable step above.

The first step is to realize the need for using optimization in a specific design problem. Thereafter, the designer needs to choose the important decision variables associated with the design problem. The formulation of optimization problems involves other considerations, such as constraints, objective function, and variable bounds. As shown above, there is usually a hierarchy in the optimization process; although one consideration may get influenced by the other. Nevertheless, it is an iterative process.

1.2 A Few Important Matters

- Optimal design involves minimization of cost, effort, distance traveled, time elapsed or maximization of benefit, profit, performance, gain or a combination of both
- Duality Principle: A minimization problem has an equivalent maximization problem:

  $$\min f(x) \equiv \max -f(x)$$

  and vice versa
- No single optimization method is efficient to all problems
- Formulation of problem in a real-world situation is oftentimes difficult. Needs consideration of many aspects including aesthetics, reliability which are hard to quantify

- A typical optimal design problem:
  Obtain optimal set of variables so as to

Minimize or Maximize	Objective function
Subject to	Constraints
Satisfying	Variable bounds

Chapter 2

Single-variable Optimization

Only single-variable functions are considered in this chapter. These methods will be used in later chapters for multi-variable function optimization.

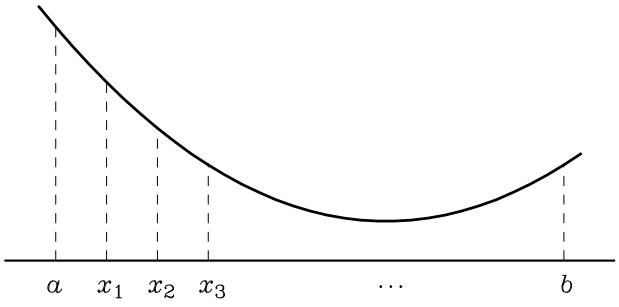
2.1 Optimality Criteria

Sufficient conditions of optimality:
Suppose at point x^*, the first derivative is zero and the first nonzero higher order derivative is denoted by n
(i) If n is odd, x^* is a point of inflection
(ii) If n is even, x^* is a local optimum
(a) If that derivative is positive, x^* is a local minimum
(b) If that derivative is negative, x^* is a local maximum.

2.2 Braketting Algorithms

Finds a lower and a upper bound of the minimum point

2.2.1 Exhaustive Search



Function values are evaluated at n equally spaced points.

Algorithm:

Step 1 Set $x^{(0)} = a$, $x^{(n+1)} = b$, $x^{(j)} = a + \frac{b-a}{n+1}j, \quad 1 \leq j \leq n$. Set $k = 1$

Step 2 If $f(x^{(k-1)}) \geq f(x^{(k)}) \leq f(x^{(k+1)})$,
the minimum lies in $(x^{(k-1)}, x^{(k+1)})$, **Terminate**

Else go to Step 3

Step 3 Is $k = n$? If no, set $k = k + 1$, go to Step 2
If yes, no minimum exists in (a, b)

Accuracy in the result: $\frac{2}{n+1}(b - a)$.
Average number of function evaluations to get to the optimum is $(\frac{n}{2} + 2)$.

2.2.2 Bounding Phase Method

This method is used to bracket the minimum of a function.
Algorithm:

Step 1 Choose an initial guess $x^{(0)}$ and an increment Δ. Set $k = 0$

Step 2 If $f(x^{(0)} - |\Delta|) \geq f(x^{(0)}) \geq f(x^{(0)} + |\Delta|)$ then Δ is positive

Else if $f(x^{(0)} - |\Delta|) \leq f(x^{(0)}) \leq f(x^{(0)} + |\Delta|)$ then Δ is negative

Else go to Step 1

Step 3 Set $x^{(k+1)} = x^{(k)} + 2^k\Delta$

Step 4 If $f(x^{(k+1)}) < f(x^{(k)})$, set $k = k + 1$ and go to Step 3

Else The minimum lies in the interval $(x^{(k-1)}, x^{(k+1)})$ and **Terminate**

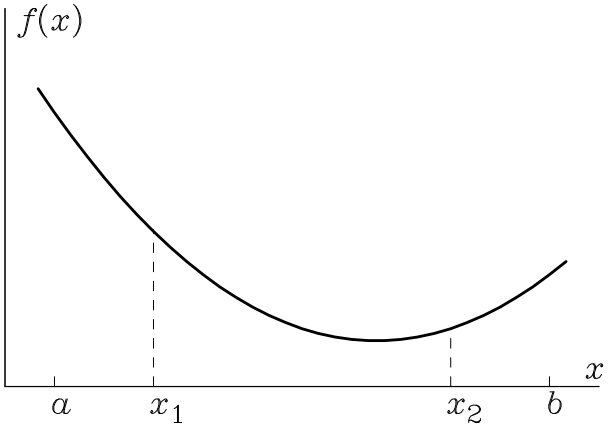
If Δ is large, poor bracketing.
If Δ is small, more evaluations.

2.3 Region-Elimination Methods

Fundamental rule for Region-elimination methods:

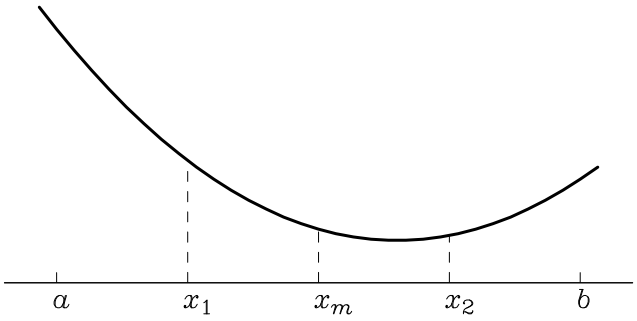
For $x_1 < x_2$ where x_1, x_2 lie in (a, b)

1. If $f(x_1) > f(x_2)$ then minimum does not lie in (a, x_1)
2. If $f(x_1) < f(x_2)$ then minimum does not lie in (x_2, b)



2.3.1 Interval Halving

Interval is reduced by half at every iteration.



Algorithm:

Step 1 Choose a lower bound a and an upper bound b. Choose also a small number ϵ. Let $x_m = (a+b)/2, \quad L_0 = L = b - a$; Compute $f(x_m)$

Step 2 Set $x_1 = a + L/4, \quad x_2 = b - L/4$, Compute $f(x_1)$ and $f(x_2)$

Step 3 If $f(x_1) < f(x_m)$ set $b = x_m$; $x_m = x_1$; go to Step 5

Else go to Step 4

Step 4 If $f(x_2) < f(x_m)$ set $a = x_m$; $x_m = x_2$; go to Step 5

Else set $a = x_1, b = x_2$; go to Step 5

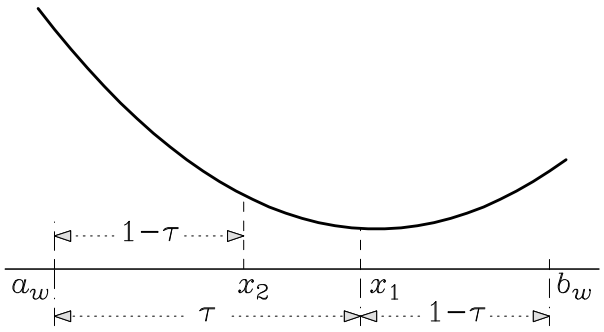
Step 5 Calculate $L = b - a$. If $|L| < \epsilon$, **Terminate**;

Else go to Step 2

Interval reduces to $0.5^{n/2}L_0$ after n function evaluations.
Two new function evaluations at each iterations.

2.3.2 Golden Section Search

Interval is reduced according to *golden* rule.
Properties: For only two trials, spread them equidistant from the center. Subinterval eliminated should be of the same length regardless of the outcome of the trial. Only one new point at each step is evaluated, other point remains from the previous step.



Formula: $1 - \tau = \tau^2$ yields $\tau = 0.618$
Golden section search is similar to Fibonacci search for large n.

Algorithm:

Step 1 Choose a lower bound a and an upper bound b. Also choose a small number ϵ. Normalize the variable x by using the equation, $w = (x - a)/(b - a)$. Thus, $a_w = 0$, $b_w = 1$, and $L_w = 1$. Set $k = 1$

Step 2 Set $w_1 = a_w + (0.618)L_w$ and $w_2 = b_w - (0.618)L_w$. Compute $f(w_1)$ or $f(w_2)$ depending on whichever was not evaluated earlier. Use the fundamental region-elimination rule to eliminate a region. Set new a_w and b_w.

Step 3 Is $|L_w| < \epsilon$ small? If no, set $k = k + 1$, go to Step 2
If yes, **Terminate**

Interval reduces to $(0.618)^{n-1}$ after n evaluations. One new function evaluation at each iteration.
Consider the following function:

$$f(x) = x^2 + 54/x.$$

Step 1 We choose $a = 0$ and $b = 5$. The transformation equation becomes $w = x/5$. Thus, $a_w = 0$, $b_w = 1$, and $L_w = 1$. Since the golden section method works with a transformed variable w, it is convenient to work with the transformed function:

$$f(w) = 25w^2 + 54/(5w).$$

In the w-space, the minimum lies at $w^* = 3/5 = 0.6$. We set an iteration counter $k = 1$.

Step 2 We set $w_1 = 0 + (0.618)1 = 0.618$ and $w_2 = 1 - (0.618)1$ or $w_2 = 0.382$. The corresponding function values are $f(w_1) = 27.02$ and $f(w_2) = 31.92$. Since $f(w_1) < f(w_2)$, the minimum cannot lie in any point smaller than $w = 0.382$. Thus, we eliminate the region (a, w_2) or $(0, 0.382)$. Thus, $a_w = 0.382$ and $b_w = 1$. At this stage, $L_w = 1 - 0.382 = 0.618$. The region being eliminated after this iteration is shown in Figure 2.1. The position of the exact minimum at $w = 0.6$ is also shown.

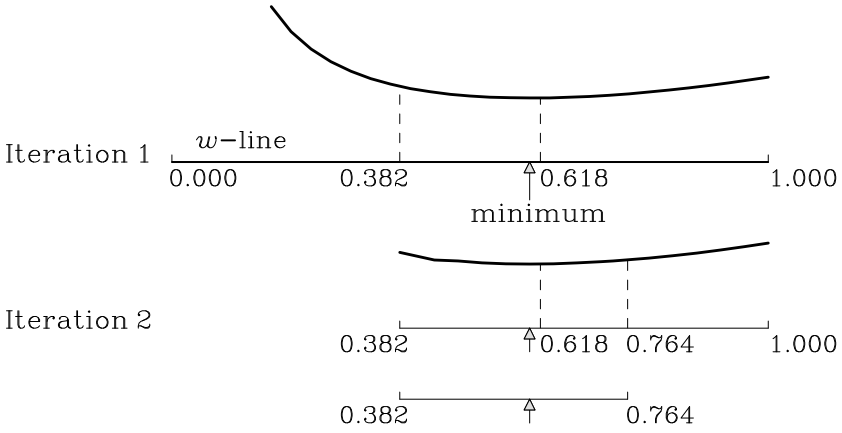


Figure 2.1: Region eliminations in the first two iterations of the golden section search algorithm.

Step 3 Since $|L_w|$ is not smaller than ϵ, we set $k = 2$ and move to Step 2. This completes one iteration of the golden section search method.

Step 2 For the second iteration, we set

$$w_1 = 0.382 + (0.618)0.618 = 0.764,$$
$$w_2 = 1 - (0.618)0.618 = 0.618.$$

We observe that the point w_2 was computed in the previous iteration. Thus, we only need to compute the function value at w_1: $f(w_1) = 28.73$. Using the fundamental region-elimination rule and observing the relation $f(w_1) > f(w_2)$, we eliminate the interval $(0.764, 1)$. Thus, the new bounds are $a_w = 0.382$ and $b_w = 0.764$, and the new interval is $L_w = 0.764 - 0.382 = 0.382$, which is incidentally equal to $(0.618)^2$! Figure 2.1 shows the final region after two iterations of this algorithm.

Step 3 Since the obtained interval is not smaller than ϵ, we continue to proceed to Step 2 after incrementing the iteration counter k to 3.

Step 2 Here, we observe that $w_1 = 0.618$ and $w_2 = 0.528$, of which the point w_1 was evaluated before. Thus, we compute $f(w_2)$ only: $f(w_2) = 27.43$. We also observe that $f(w_1) < f(w_2)$ and we eliminate the interval $(0.382, 0.528)$. The new interval is $(0.528, 0.764)$ and the new range is $L_w = 0.764 - 0.528 = 0.236$, which is exactly equal to $(0.618)^3$!

Step 3 Thus, at the end of the third iteration, $L_w = 0.236$. This way, Steps 2 and 3 may be continued until the desired accuracy is achieved.

We observe that at each iteration, only one new function evaluation is necessary. After three iterations, we have performed only four function evaluations. Thus, the interval reduces to $(0.618)^3$ or 0.236.

2.4 Point-Estimation Methods

- Region-elimination methods only take account of ordering of function values
- Point-estimation methods take account of the relative magnitudes of function values.
- Thus, Point-estimation methods are in general better than region-elimination methods

2.4.1 Successive Quadratic Estimation Method

Uses quadratic polynomial approximation with three points to guess the minimum iteratively. Given a quadratic function:

$$q(x) = a_0 + a_1(x - x_1) + a_2(x - x_1)(x - x_2).$$

If (x_1, f_1), (x_2, f_2), and (x_3, f_3) are three points on this function, then following relationships can be obtained:

$$\begin{aligned} a_0 &= f_1 \\ a_1 &= \frac{f_2 - f_1}{x_2 - x_1} \\ a_2 &= \frac{1}{x_3 - x_2}\left(\frac{f_3 - f_1}{x_3 - x_1} - \frac{f_2 - f_1}{x_2 - x_1}\right) \end{aligned}$$

The minimum of above function lies at

$$\bar{x} = \frac{x_2 + x_1}{2} - \frac{a_1}{2a_2}. \tag{2.1}$$

Algorithm:

Step 1 Let x_1 be an initial point and Δ be the step size. Compute $x_2 = x_1 + \Delta$

Step 2 Evaluate $f(x_1)$ and $f(x_2)$

Step 3 If $f(x_1) > f(x_2)$, let $x_3 = x_1 + 2\Delta$
Else let $x_3 = x_1 - \Delta$. Evaluate $f(x_3)$

Step 4 Determine $F_{\min} = \min(f_1, f_2, f_3)$ and $X_{\min}$ is the point x_i corresponds to $F_{\min}$

Step 5 Use points x_1, x_2, and x_3 to calculate $\bar{x}$ using equation (2.1)

Step 6 Are $|F_{\min} - f(\bar{x})|$ and $|X_{\min} - \bar{x}|$ small? If not, go to Step 7
Else the optimum is the best of current four points and **Terminate**

Step 7 Save the best point and two bracketting it, if possible; otherwise save the best three points. Relabel them according to $x_1 < x_2 < x_3$ and go to Step 4

2.5 Methods Requiring Derivatives

- Use gradient information.
- At local minimum, the derivative of the function is equal to zero.

Gradients are computed numerically as follows:

$$f'(x^{(t)}) = \frac{f(x^{(t)} + \Delta x^{(t)}) - f(x^{(t)} - \Delta x^{(t)})}{2\Delta x^{(t)}}, \tag{2.2}$$

$$f''(x^{(t)}) = \frac{f(x^{(t)} + \Delta x^{(t)}) - 2f(x^{(t)}) + f(x^{(t)} - \Delta x^{(t)})}{(\Delta x^{(t)})^2}. \tag{2.3}$$

The parameter $\Delta x^{(t)}$ is usually taken to be a small value. In all our calculations, we assign $\Delta x^{(t)}$ to be about 1 per cent of $x^{(t)}$:

$$\Delta x^{(t)} = \begin{cases} 0.01|x^{(t)}| & \text{if } |x^{(t)}| > 0.01 \\ 0.0001 & \text{otherwise.} \end{cases} \tag{2.4}$$

According to Equations (2.2) and (2.3), the first derivative requires two function evaluations and the second derivative requires three function evaluations.

2.5.1 Newton-Raphson Method

A linear approximation to the derivative is made to zero to derive the transition rule.
Algorithm:

Step 1 Choose initial guess x_1 and a small number ϵ. Set $k = 1$. Compute $f'(x_1)$

Step 2 Compute $f''(x_k)$

Step 3 Calculate $x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$. Compute $f'(x_{k+1})$

Step 4 If $| f'(x_{k+1}) | < \epsilon$, **Terminate**
Else set $k = k + 1$ and Go to step 2

Convergence of the algorithm depends on the initial point and the nature of the objective function.

2.5.2 Bisection Method

Both function value and sign of derivative are used to derive the transition rule.
Algorithm:

Step 1 Choose two points a and b such that $f'(a) < 0$ and $f'(b) > 0$, and a small number ϵ. Set $L = a$ and $R = b$

Step 2 Calculate $z = (R + L)/2$ and evaluate $f'(z)$

Step 3 If $| f'(z) | \leq \epsilon$, **Terminate**
Else if $f'(z) < 0$ set $L = z$ and Go to step 2; else if $f'(z) > 0$ set $R = z$ and Go to step 2

Consider again the function:

$$f(x) = x^2 + 54/x.$$

Step 1 We choose two points $a = 2$ and $b = 5$ such that $f'(a) = -9.501$ and $f'(b) = 7.841$ are of opposite sign. The derivatives are computed numerically using Equation (2.2). We also choose a small number $\epsilon = 10^{-3}$.

Step 2 We calculate a quantity $z = (x_1 + x_2)/2 = 3.5$ and compute $f'(z) = 2.591$.

Step 3 Since $f'(z) > 0$, the right-half of the search space needs to be eliminated. Thus, we set $x_1 = 2$ and $x_2 = z = 3.5$. This completes one iteration of the algorithm. This algorithm works more like the interval halving method described in Section 2.4. At each iteration, only half of the search region is eliminated, but here the decision about which half to delete depends on the derivatives at the mid-point of the interval.

Step 2 We compute $z = (2 + 3.5)/2 = 2.750$ and $f'(z) = -1.641$.

Step 3 Since $f'(z) < 0$, we set $x_1 = 2.750$ and $x_2 = 3.500$.

Step 2 The new point z is the average of the two bounds: $z = 3.125$. The function value at this point is $f'(z) = 0.720$.

Step 3 Since $|f'(z)| \not< \epsilon$, we continue with Step 2.

Thus, at the end of 10 function evaluations, we have obtained an interval $(2.750, 3.125)$, bracketing the minimum point $x^* = 3.0$. The guess of the minimum point is the mid-point of the obtained interval or $x = 2.938$. This process continues until we find a point with a vanishing derivative. Since at each iteration, the gradient is evaluated only at one new point, the bisection method requires two function evaluations per iteration. In this method, exactly half the region is eliminated at every iteration; but using the magnitude of the gradient, a faster algorithm can be designed to adaptively eliminate variable portions of search region—a matter which we discuss in the following subsection.

2.5.3 Secant Method

Both function value and derivative are used to derive the transition rule.
Algorithm:
Same as Bisection method except step 2 is modified as follows:

Step 2 Calculate $z = R - \frac{f'(R)}{(f'(R)-f'(L))/(R-L)}$ and evaluate $f'(z)$.

2.5.4 Cubic Search Method

A cubic approximation function

$$\bar{f}(x) = a_0 + a_1(x - x_1) + a_2(x - x_1)(x - x_2) + a_3(x - x_1)^2(x - x_2)$$

is approximated by two points and derivatives at those points (x_1, f_1, f_1'), (x_2, f_2, f_2'). By making the derivative of the above equation to zero, the approximation to the minimum of the original function is obtained:

$$\bar{x} = \begin{cases} x_2, & \text{if } \mu = 0; \\ x_2 - \mu(x_2 - x_1), & \text{if } 0 \le \mu \le 1; \\ x_1, & \text{if } \mu > 1. \end{cases} \tag{2.5}$$

where

$$\begin{aligned} z &= \frac{3(f_1 - f_2)}{x_2 - x_1} + f_1' + f_2', \\ w &= \frac{x_2 - x_1}{|\, x_2 - x_1 \,|}\sqrt{(z^2 - f_1' f_2')}, \\ \mu &= \frac{f_2' + w - z}{f_2' - f_1' + 2w}. \end{aligned}$$

Algorithm:

Step 1 Choose an initial point $x^{(0)}$, a step size Δ, and two termination parameters ϵ_1 and ϵ_2. Compute $f'(x^{(0)})$. If $f'(x^{(0)}) > 0$, set $\Delta = -\Delta$. Set $k = 0$.

Step 2 Compute $x^{(k+1)} = x^{(k)} + 2^k\Delta$.

Step 3 Evaluate $f'(x^{(k+1)})$. If $f'(x^{(k+1)})f'(x^{(k)}) \leq 0$, set $x_1 = x^{(k)}$ and $x_2 = x^{(k+1)}$, and go to Step 4.

Else set $k = k + 1$ and go to Step 2.

Step 4 Calculate the point $\bar{x}$ using equation (2.5)

Step 5 If $f(\bar{x}) < f(x_1)$, go to Step 6

Else set $\bar{x} = \bar{x} - \frac{1}{2}(\bar{x} - x_1)$ until $f(\bar{x}) \leq f(x_1)$ is achieved

Step 6 Compute $f'(\bar{x})$. If $|f'(\bar{x})| \leq \epsilon_1$ and $|(\bar{x} - x_1)/\bar{x}| \leq \epsilon_2$, **Terminate**,

Else if $f'(\bar{x})f'(x_1) < 0$, set $x_2 = \bar{x}$

Else set $x_1 = \bar{x}$.
Go to Step 4

Chapter 3

Multi-variable Optimization

Functions of multiple variables (N variables) are considered for minimization here. Duality principle can be used to apply these methods to maximization problems.

3.1 Optimality Criteria

A stationary point $\bar{x}$ is minimum, maximum, or saddle-point if $\nabla^2 f(\bar{x})$ is positive definite, negative definite, or $\nabla^2 f(\bar{x}) \leq\geq 0$.
Necessary Conditions: For x^* to be a local minimum, $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive semidefinite.
Sufficient Conditions: If $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive definite than, x^* is an isolated local minimum of $f(x)$.

3.2 Direct Search Methods

Use only function values; no gradient information is used.

3.2.1 Simplex Search Method

- A simplex is a set of $(N + 1)$ points. At each iteration, a new simplex is created from the current simplex by fixed transition rules.
- Originally proposed by Spendley, Hext, & Himsworth (1962) and later modified by Nelder and Meade (1965):
- The worst point is projected a suitable distance through the centroid of the remaining points.

Algorithm:

Step 1 Choose $\gamma > 1$, $\beta \in (0, 1)$, and a termination parameter ϵ. Create an initial simplex[1]

Step 2 Find x_h (the worst point), x_l (the best point), and x_g (next to the worst point). Calculate $x_c = \frac{1}{N} \sum_{i=1, i\neq h}^{N+1} x_i$

[1]One of the ways to create a simplex is to choose a base point x^0 and a scale factor C. Then $(N + 1)$ points are $x^{(0)}$ and for $i, j = 1, 2, \ldots, N$

$$x_j^{(i)} = \begin{cases} x_j^{(0)} + C & \text{if } j = i; \\ x_j^{(0)} + C\Delta, & \text{otherwise,} \end{cases} \quad \text{where } \Delta = \begin{cases} 0.25, & \text{if } N = 3; \\ \frac{\sqrt{N+1}-2}{N-3}, & \text{otherwise.} \end{cases}$$

Step 3 Calculate the reflected point $x_r = 2x_c - x_h$. Set $x_{\text{new}} = x_r$

If $f(x_r) < f(x_l)$, set $x_{\text{new}} = (1+\gamma)x_c - \gamma x_h$ (expansion)

Else if $f(x_r) \geq f(x_h)$, set $x_{\text{new}} = (1-\beta)x_c + \beta x_h$ (contraction)

Else if $f(x_g) < f(x_r) < f(x_h)$, set $x_{\text{new}} = (1+\beta)x_c - \beta x_h$ (contraction)

Calculate $f(x_{\text{new}})$ and replace x_h by x_{new}

Step 4 If $\left\{ \sum_{i=1}^{N+1} \frac{(f(x_i)-f(x_c))^2}{N+1} \right\}^{1/2} \leq \epsilon$, **Terminate**

Else go to Step 2

3.2.2 Hooke-Jeeves Pattern Search Method

- Main motivation in Simplex search is to find search direction.
- Work with a set of directions that completely spans the search space
- 'One-variable-at-a-time' technique fails on elongated and distorted contours.
- Powell has shown that searching along a fixed set of directions is not adequate
- HJ method is a combination of exploratory moves and heuristic pattern moves.

Exploratory Move: Assume current solution (the base point) is denoted by x^c. Assume also that the variable x_i^c is perturbed by Δ_i. Set $i = 1$ and $x = x^c$

Step 1 Calculate $f = f(x)$, $f^+ = f(x_i + \Delta_i)$ and $f^- = f(x_i - \Delta_i)$

Step 2 Find $f_{\min} = \min(f, f^+, f^-)$. Set x corresponds to $f_{\min}$

Step 3 Is $i = N$? If no, set $i = i + 1$; go to Step 1

Else x is the result and go to Step 4

Step 4 If $x \neq x^c$, **success**, Else **failure**

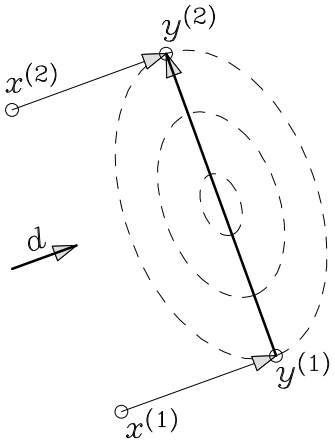
Pattern move: A single step from the present base point, $x^{(k)}$, along the previous, $x^{(k-1)}$, to the current base point:

$$x_p^{(k+1)} = x^{(k)} + (x^{(k)} - x^{(k-1)}).$$

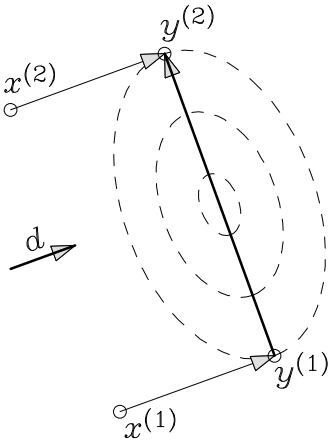
3.2.3 Powell's Conjugate Direction Method

- Most successful direct search method
- Uses history of iterations to create new search directions
- Based on a quadratic model
- Generate N conjugate directions and perform one-dimensional search in each direction one at a time

Parallel Subspace Property: Given a quadratic function $q(x)$, two arbitrary but distinct points $x^{(1)}$ and $x^{(2)}$, and a direction d. If $y^{(1)}$ is the solution to $\min q(x^{(1)} + \lambda d)$ and $y^{(2)}$ is the solution to $\min q(x^{(2)} + \lambda d)$, then the direction $(y^{(2)} - y^{(1)})$ is C-conjugate to d or $(y^{(2)} - y^{(1)})^T C d =$ Diagonal matrix.



(a)



(a)

Instead of using two points and a direction vector to create one conjugate direction, one point and coordinate directions can be used to create conjugate directions.
Extended Parallel Subspace Property: In higher dimensions, if from $x^{(1)}$ the point $y^{(1)}$ is found after searches along each of $m(< n)$ conjugate directions, and similarly if from $x^{(2)}$ the point $y^{(2)}$ is found after searches along each of m conjugate directions, $s^{(1)}$, $s^{(2)}$, ..., $s^{(m)}$, then the vector $(y^{(2)} - y^{(1)})$ will be the conjugate to *all* of the m previous directions.
Algorithm:

Step 1 Choose a starting point $x^{(0)}$ and a set of N linearly independent directions; possibly $s^{(i)} = e^{(i)}$ for $i = 1, 2, \ldots, N$

Step 2 Minimize along N unidirectional search directions using the previous minimum point to begin the next search. Begin with the search direction $s^{(1)}$ and end with $s^{(N)}$. Thereafter, perform another unidirectional search along $s^{(1)}$

Step 3 Form a new conjugate direction d using the extended parallel subspace property

Step 4 If $\|d\|$ is small or search directions are linearly independent, **Terminate**

Else replace $s^{(j)} = s^{(j-1)}$ for all $j = N, N-1, \ldots, 2$. Set $s^{(1)} = d/\|d\|$ and go to Step 2

A test is required to ensure linear independence of conjugate directions. If the function is quadratic, exactly N loops through steps 2 to 4 are required. If the function is quadratic, exactly $(N-1)$ loops through Steps 2 to 4 is required. Since in every iteration of the above

algorithm exactly $(N+1)$ unidirectional searches are necessary, a total of $(N-1)\times(N+1)$ or (N^2-1) unidirectional searches are necessary to find N conjugate directions. Thereafter, one final unidirectional search is necessary to obtain the minimum point. Thus, in order to find the minimum of a quadratic objective function, the conjugate direction method requires a total of N^2 unidirectional searches.
Disadvantages:

- It takes usually more than N cycles for nonquadratic functions
- One-dimensional searches may not be exact, so directions may not be conjugate
- May halt before the optima is reached

Consider again the Himmelblau function:

$$\text{Minimize} \quad f(x_1,x_2)=(x_1^2+x_2-11)^2+(x_1+x_2^2-7)^2$$

in the interval $0\le x_1,x_2\le 5$.

Step 1 We begin with a point $x^{(0)}=(0,4)^T$. We assume initial search directions as $s^{(1)}=(1,0)^T$ and $s^{(2)}=(0,1)^T$.

Step 2 We first find the minimum point along the search direction $s^{(1)}$. Any point along that direction can be written as $x^p=x^{(0)}+\alpha s^{(1)}$, where α is a scalar quantity expressing the distance of the point x^p from $x^{(0)}$. Thus, the point x^p can be written as $x^p=(\alpha,4)^T$. Now the two-variable function $f(x_1,x_2)$ can be expressed in terms of one variable α as

$$F(\alpha)=(\alpha^2-7)^2+(\alpha+9)^2,$$

which represents the function value of any point along the direction $s^{(1)}$ and passing through $x^{(0)}$. Since we are looking for the point for which the function value is minimum, we may differentiate the above expression with respect to α and equate to zero. But in any arbitrary problem, it may not be possible to write an explicit expression of the single-variable function $F(\alpha)$ and differentiate. In those cases, the function $F(\alpha)$ can be obtained by substituting each variable x_i by x_i^p. Thereafter, any single-variable optimization methods, as described in Chapter 2, can be used to find the minimum point. The first task is to bracket the minimum and then the subsequent task is to find the minimum point. Here, we could have found the exact minimum solution by differentiating the single-variable function $F(\alpha)$ with respect to α and then equating the term to zero, but we follow the more generic procedure of numerical differentiation, a method which will be used in many real-world optimization problems. Using the bounding phase method in the above problem we find that the minimum is bracketed in the interval $(1,4)$ and using the golden section search we obtain the minimum $\alpha^*=2.083$ with three decimal places of accuracy. Thus, $x^{(1)}=(2.083,4.000)^T$. The above procedure of obtaining the best point along a search direction is also described in Section 3.2.

Similarly, we find the minimum point along the second search direction $s^{(2)}$ from the point $x^{(1)}$. A general point on that line is

$$x(\alpha)=x^{(1)}+\alpha s^{(2)}=(2.083,(4+\alpha))^T.$$

The optimum point found using a combined application of the bounding phase and the golden section search method is $\alpha^*=-1.592$ and the corresponding point is $x^{(2)}=(2.083,2.408)^T$.

From the point $x^{(2)}$, we perform a final unidirectional search along the first search direction $s^{(1)}$ and obtain the minimum point $x^{(3)}=(2.881,2.408)^T$.

Step 3 According to the parellel subspace property, we find the new conjugate direction

$$d = x^{(3)} - x^{(1)} = (2.881, 2.408)^T - (2.083, 4.000)^T = (0.798, -1.592)^T.$$

Step 4 The magnitude of search vector d is not small. Thus, the new conjugate search directions are

$$\begin{aligned} s^{(2)} &= (1, 0)^T, \\ s^{(1)} &= (0.798,\ -1.592)^T / \|(0.798,\ -1.592)^T\| \\ &= (0.448,\ -0.894)^T. \end{aligned}$$

This completes one iteration of Powell's conjugate direction method. Figure 3.1 shows the new conjugate direction on a contour plot of the objective function. With these new search directions we now proceed to Step 2.

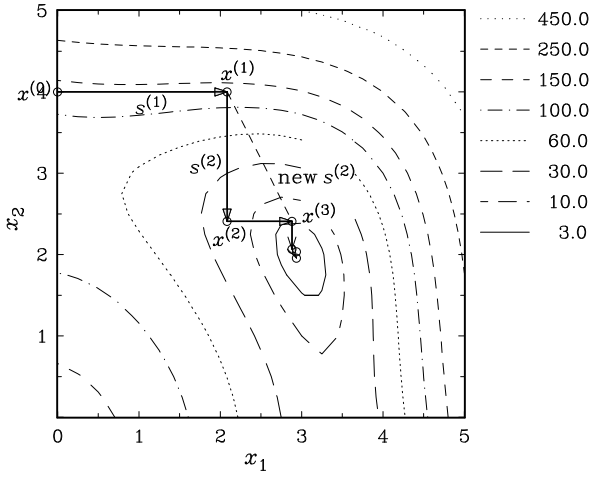


Figure 3.1: Two iterations of Powell's conjugate direction method.

Step 2 A single-variable minimization along the search direction $s^{(1)}$ from the point $x^{(3)} = (2.881, 2.408)^T$ results in the new point $x^{(4)} = (3.063, 2.045)^T$. If the objective function were a quadratic function, we would have achieved that the optimum point at this step. However, it is interesting to note that the solution $x^{(4)}$ is close to the true minimum of the function.

One more unidirectional search along $s^{(2)}$ from the point $x^{(4)}$ results in the point $x^{(5)} = (2.988, 2.045)^T$. Another minimization along $s^{(1)}$ results in $x^{(6)} = (3.008, 2.006)^T$ ($f(x^{(6)} = 0.004)$.

Step 3 The new conjugate direction is

$$d = (x^{(6)} - x^{(4)}) = (0.055, -0.039)^T.$$

The unit vector along this direction is $(0.816, -0.578)^T$.

Step 4 The new pair of conjugate search directions are $s^{(1)} = (0.448, -0.894)^T$ and $s^{(2)} = (0.055, -0.039)^T$, respectively. The search direction d (before normalizing) may be considered to be small and therefore the algorithm may terminate.

We observe that in one iteration of Step 2, $(N + 1)$ unidirectional searches are necessary. Thus, computationally this method may be expensive. In terms of the storage requirement, the algorithm has to store $(N+1)$ points and N search directions at any stage of the iteration.

3.3 Gradient-based Methods

- Direct search methods are expensive
- Gradient-based methods assume existence of $f(x)$, $\nabla f(x)$, and $\nabla^2 f(x)$
- All methods employ
$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} s(x^{(k)})$$
where $\alpha^{(k)}$ is the step length and $s(x^{(k)})$ is the search direction.
- Usually, $\alpha^{(k)}$ is selected so as to minimize the function in $s(x^{(k)})$ direction. Update search direction iteratively.

3.3.1 Numerical Gradient Approximation

- In real-world engineering problems, gradients are oftentimes difficult to calculate.
- Even if they are available, their computation could be error prone
- Numerically gradients are computed:

$$\begin{aligned}
\left.\frac{\partial f(x)}{\partial x_i}\right|_{x^{(t)}} &= \left(f(x_i^{(t)}+\Delta x_i^{(t)}) - f(x_i^{(t)}-\Delta x_i^{(t)})\right) \Big/ 2\Delta x_i^{(t)}, \\
\left.\frac{\partial^2 f(x)}{\partial^2 x_i^2}\right|_{x^{(t)}} &= \left(f(x_i^{(t)}+\Delta x_i^{(t)}) - 2f(x^{(t)}) + f(x_i^{(t)}-\Delta x_i^{(t)})\right) \Big/ \left(\Delta x_i^{(t)}\right)^2. \\
\left.\frac{\partial^2 f(x)}{\partial x_i \partial x_j}\right|_{x^{(t)}} &= \Big[f(x_i^{(t)}+\Delta x_i^{(t)},\, x_j^{(t)}+\Delta x_j^{(t)}) \\
&\quad -f(x_i^{(t)}+\Delta x_i^{(t)},\, x_j^{(t)}-\Delta x_j^{(t)}) - f(x_i^{(t)}-\Delta x_i^{(t)},\, x_j^{(t)}+\Delta x_j^{(t)}) \\
&\quad + f(x_i^{(t)}-\Delta x_i^{(t)},\, x_j^{(t)}-\Delta x_j^{(t)})\Big] \Big/ \left(4\Delta x_i^{(t)} \Delta x_j^{(t)}\right).
\end{aligned}$$

The computation of the first derivative with respect to each variable requires two function evaluations, thus totaling $2N$ function evaluations for the complete first derivative vector. The computation of the second derivative $\partial^2 f/\partial x_i^2$ requires three function evaluations, but the second-order partial derivative $\partial^2 f/(\partial x_i \partial x_j)$ requires four function evaluations. Thus, the computation of Hessian matrix requires $(2N^2 + 1)$ function evaluations (assuming the symmetry of the matrix).

3.3.2 Cauchy's Method (Steepest Descent)

Greatest decrease in $s(x^{(k)}) = -\nabla f(x)$ direction.
Algorithm:

Step 1 Choose a maximum number of iterations M to be performed, an initial point $x^{(0)}$, two termination parameters ϵ_1, ϵ_2, and set $k = 0$

Step 2 Calculate $\nabla f(x^{(k)})$, the first derivative at the point $x^{(k)}$

Step 3 If $\|\nabla f(x^{(k)})\| \leq \epsilon_1$, **Terminate**

Else if $k \geq M$, **Terminate**

Else go to Step 4

Step 4 Perform a unidirectional search to find $\alpha^{(k)}$ using ϵ_2 such that $f(x^{(k+1)}) = f(x^{(k)} - \alpha^{(k)}\nabla f(x^{(k)}))$ is minimum. One criterion for termination is when $\|\nabla f(x^{(k+1)}) \cdot \nabla f(x^{(k)})\| \leq \epsilon_2$

Step 5 Is $\left\|\frac{x^{(k+1)} - x^{(k)}}{x^{(k)}}\right\| \leq \epsilon_1$? If yes, **Terminate**

Else set $k = k + 1$ and go to Step 2

Cauchy's method works well when $x^{(0)}$ is far away from x^*.

3.3.3 Newton's Method

Use a search direction $s(x^{(k)}) = -\nabla^2 f(x^{(k)})^{-1} \nabla f(x^{(k)})$.
Algorithm: Same as Cauchy's method except step 4 is modified:
Step 4 Perform a line search to find $\alpha^{(k)}$ using ϵ_2 such that $f(x^{(k+1)}) = f(x^{(k)} - \alpha^{(k)} \nabla^2 f(x^{(k)})^{-1} \nabla f(x^{(k)}))$ is minimum

Newton's method works well when $x^{(0)}$ is close to x^*.

3.3.4 Marquardt's Method

- A compromise between Cauchy's and Newton's method.
- When the point is far away from the optimum, Cauchy's method is used and when the point is close to the optimum Newton's method is used.

Algorithm:

Step 1 Choose a starting point, $x^{(0)}$, the maximum number of iterations, M, and a termination parameter, ϵ. Set $k = 0$ and $\lambda^{(0)} = 10^4$ (a large number)

Step 2 Calculate $\nabla f(x^{(k)})$

Step 3 If $\|\nabla f(x^{(k)})\| \leq \epsilon$? or $k \geq M$, **Terminate**

Else go to Step 4

Step 4 Calculate $s(x^{(k)}) = -[H^{(k)} + \lambda^{(k)} I]^{-1} \nabla f(x^{(k)})$. Set $x^{(k+1)} = x^{(k)} + s(x^{(k)})$

Step 5 Is $f(x^{(k+1)}) < f(x^{(k)})$? If yes, go to Step 6

Else go to Step 7

Step 6 Set $\lambda^{(k+1)} = \frac{1}{2}\lambda^{(k)}$, $k = k + 1$, and go to Step 2

Step 7 Set $\lambda^{(k)} = 2\lambda^{(k)}$ and go to Step 4

Need to compute Hessian matrix, which may be cumbersome.

3.3.5 Conjugate Gradient Method

- Uses only first-order derivatives
- Assuming quadratic functions, conjugate directions can be written in terms of first-order derivatives.
- **Fletcher-Reeves:** Conjugate directions are related as follows:

$$s^{(k)} = -\nabla f(x^{(k)}) + \left[\frac{\|\nabla f(x^{(k)})\|^2}{\|\nabla f(x^{(k-1)})\|^2}\right] s^{(k-1)}$$

Algorithm:

Step 1 Choose $x^{(0)}$ and termination parameters ϵ_1, ϵ_2, ϵ_3

Step 2 Find $\nabla f(x^{(0)})$ and set $s^{(0)} = -\nabla f(x^{(0)})$

Step 3 Find $\lambda^{(0)}$ such that $f(x^{(0)} + \lambda^{(0)} s^{(0)})$ is minimum with termination parameter ϵ_1. Set $x^{(1)} = x^{(0)} + \lambda^{(0)^*} s^{(0)}$ and $k = 1$. Calculate $\nabla f(x^{(1)})$

Step 4 Set

$$s^{(k)} = -\nabla f(x^{(k)}) + \frac{\|\nabla f(x^{(k)})\|^2}{\|\nabla f(x^{(k-1)})\|^2} s^{(k-1)}$$

Step 5 Find $\lambda^{(k)}$ such that $f(x^{(k)} + \lambda^{(k)} s^{(k)})$ is minimum with termination parameter ϵ_1. Set $x^{(k+1)} = x^{(k)} + \lambda^{(k)^*} s^{(k)}$

Step 6 Is $\left\|\frac{(x^{(k+1)} - x^{(k)})}{x^{(k)}}\right\| \leq \epsilon_2$ or $\|\nabla f(x^{(k+1)})\| \leq \epsilon_3$?
If yes, **Terminate**

Else set $k = k + 1$ and go to Step 4

3.3.6 Variable-Metric Method (Davidon-Fletcher-Powell method)

- Uses positive characteristics of Newton's method using only first-order information
- The search direction is

$$s(x^{(k)}) = -A^{(k)} \nabla f(x^{(k)}), \tag{3.1}$$

where $A^{(k)}$ is the Hessian matrix.

- Starting with an identity matrix ($A^{(0)} = I$), iteratively evaluate the Hessian matrix with $g(x^{(k)}) = \nabla f(x^{(k)})$:

$$\begin{aligned} A^{(k)} &= A^{(k-1)} + \frac{\Delta x^{(k-1)} \Delta x^{(k-1)^T}}{\Delta x^{(k-1)^T} \Delta e(x^{(k-1)})} \\ &\quad - \frac{A^{(k-1)} \Delta e(x^{(k-1)}) (A^{(k-1)} \Delta e(x^{(k-1)}))^T}{\Delta e(x^{(k-1)})^T A^{(k-1)} \Delta e(x^{(k-1)})}. \end{aligned} \tag{3.2}$$

- The above relation preserves the positive definiteness of the matrix A.

Algorithm:

Same as Fletcher-Reeves algorithm except the expression for search direction $s(x^{(k)})$ in step 4, which is set according to equation 3.1.

Let us consider the Himmelblau function again:

$$\text{Minimize} \quad f(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2.$$

The inverse of the Hessian matrix at the minimum point $x^* = (3, 2)^T$ is found to be

$$H^{-1}(x^*) = \begin{pmatrix} 0.016 & -0.009 \\ -0.009 & 0.035 \end{pmatrix}. \tag{3.3}$$

Successive iterations of the DFP method transform an initial identity matrix into the above matrix. This allows the DFP search to become similar to Newton's search after a few iterations. The advantage of this method over Newton's method is that the inverse of the Hessian matrix need not be calculated. Thus, the algorithm has the effect of a second-order search, but the search is achieved only with first-order derivatives.

Step 1 Once again we begin with the initial point $x^{(0)} = (0, 0)^T$. The termination parameters are all set to be 10^{-3}.

Step 2 The derivative vector at the initial point is equal to $\nabla f(x^{(0)}) = (-14, -22)^T$. The search direction is $s^{(0)} = (14, 22)^T$.

Step 3 A unidirectional search from $x^{(0)}$ along $s^{(0)}$ gives the minimum point $x^{(1)} = (1.788, 2.810)^T$, as found in Step 3 of the Exercise 3.4.5. We calculate the gradient at this point:

$$\nabla f(x^{(1)}) = (-30.707, 18.803)^T.$$

Step 4 In order to calculate the new search direction, we first compute the parameters required to be used in Equation (3.2):

$$\begin{gathered} \Delta x^{(0)} = x^{(1)} - x^{(0)} = (1.788, 2.810)^T, \\ \Delta e(x^{(0)}) = e(x^{(1)}) - e(x^{(0)}) = (-16.707, 40.803)^T, \\ A^{(0)} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \end{gathered}$$

With these parameters, we compute the next estimate of the matrix A as follows:

$$\begin{aligned} A^{(1)} &= A^{(0)} + \frac{\Delta x^{(0)} \Delta x^{(0)^T}}{\Delta x^{(0)^T} \Delta e(x^{(0)})} - \frac{A^{(0)} \Delta e(x^{(0)}) (A^{(0)} \Delta e(x^{(0)}))^T}{\Delta e(x^{(0)})^T A^{(0)} \Delta e(x^{(0)})}, \\ &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \frac{\begin{pmatrix} 1.778 \\ 2.810 \end{pmatrix} (1.788,\ 2.810)}{(1.788, 2.810) \begin{pmatrix} -16.707 \\ 40.803 \end{pmatrix}} \\ &\quad - \frac{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} -16.707 \\ 40.803 \end{pmatrix} \left(\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} -16.707 \\ 40.803 \end{pmatrix} \right)^T}{(-16.707, 40.803) \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} -16.707 \\ 40.803 \end{pmatrix}}, \\ &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} 0.038 & 0.060 \\ 0.060 & 0.093 \end{pmatrix} - \begin{pmatrix} 0.144 & -0.351 \\ -0.351 & 0.856 \end{pmatrix}, \\ &= \begin{pmatrix} 0.894 & 0.411 \\ 0.411 & 0.237 \end{pmatrix}. \end{aligned}$$

Knowing $A^{(1)}$, we now obtain the search direction $s^{(1)}$ using Equation (3.1) as follows:

$$s^{(1)} = -\begin{pmatrix} 0.894 & 0.411 \\ 0.411 & 0.237 \end{pmatrix} \begin{pmatrix} -30.707 \\ 18.803 \end{pmatrix} = \begin{pmatrix} 19.724 \\ 8.164 \end{pmatrix}.$$

The unit vector along this direction is $(0.924, 0.382)^T$. It is interesting to observe that this search direction is similar to the first search direction $s^{(1)}$ obtained by using the Fletcher-Reeves algorithm (see Exercise 3.4.5). But soon we shall see how subsequent search directions obtained by the DFP method vary from the Fletcher-Reeves method.

Step 5 Performing a unidirectional search along $s^{(1)}$ from the point $x^{(1)}$, we obtain the minimum point: $x^{(2)} = (2.248, 2.989)^T$ having a function value $f(x^{(2)}) = 26.237$ (Figure 3.2).

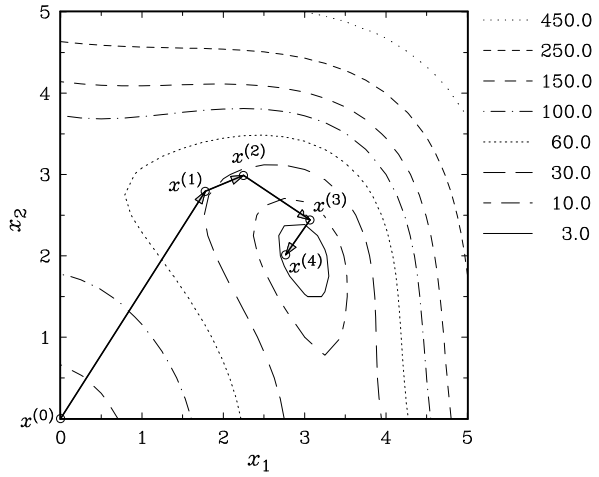


Figure 3.2: Three iterations of the DFP method shown on a contour plot of Himmelblau's function.

Step 6 We observe that the point $x^{(2)}$ is very different from the point $x^{(1)}$ and calculate the gradient at $x^{(2)}$: $\nabla f(x^{(2)}) = (-18.225, 44.037)^T$. Since $\|\nabla f(x^{(2)})\|$ is not close to zero, we increment k and proceed to Step 4. This completes one iteration of the DFP method. The initial point $x^{(0)}$, the point $x^{(1)}$, and the point $x^{(2)}$ are shown in Figure 3.2.

Step 4 The second iteration begins with computing another search direction $s^{(2)} = -A^{(2)}\nabla f(x^{(2)})$. The matrix $A^{(2)}$ can be computed by calculating $\Delta x^{(1)}$ and $\Delta e(x^{(1)})$ as before and using Equation (3.2). By simplifying the expression, we obtain the new matrix

$$A^{(2)} = \begin{pmatrix} 0.070 & -0.017 \\ -0.017 & 0.015 \end{pmatrix}$$

and the new search direction is found to be

$$s^{(2)} = (0.731, -0.976)^T.$$

Note that this direction is more descent than that obtained in the Fletcher-Reeves method after one iteration.

Step 5 The unidirectional search along $s^{(2)}$ finds the best point: $x^{(3)} = (2.995, 1.991)^T$ with a function value equal to $f(x^{(3)}) = 0.003$.

Another iteration updates the A matrix as follows:

$$A^{(3)} = \begin{pmatrix} 0.030 & -0.005 \\ -0.005 & 0.020 \end{pmatrix}.$$

The new search direction is found to be $s^{(3)} = (0.014, 0.095)^T$. The unidirectional search method finds the new point $x^{(4)} = (2.997, 2.003)^T$ with a function value equal to 3×10^{-4}. Another iteration finds the following A matrix:

$$A^{(4)} = \begin{pmatrix} 0.018 & -0.011 \\ -0.011 & 0.036 \end{pmatrix}$$

and the new search direction $s^{(4)} = (0.003, -0.003)^T$. A unidirectional search finds the point $x^{(5)} = (3.000, 2.000)^T$ with a function value equal to zero. One final iteration finds the A matrix:

$$A^{(5)} = \begin{pmatrix} 0.016 & -0.009 \\ -0.009 & 0.035 \end{pmatrix},$$

which is identical to the inverse of the Hessian matrix of the Himmelblau function at the minimum point (Equation (3.3)). The above calculation shows how the original identity matrix $A^{(0)}$ has transformed into $H^{-1}(x^*)$ matrix in successive iterations. From the fourth iteration onwards, the search direction is very similar to that in the Newton's method, because the matrix $A^{(k)}$ is similar to the inverse of the Hessian matrix. Since, during that iteration the solution is also close to the minimum, the search converges faster to the minimum point. One difficulty with this method is that the matrix A sometimes becomes ill-conditioned due to numerical derivative computations and due to inaccuracy involved in unidirectional searches.

Chapter 4

Constrained Optimization

Single and multi-variable functions with equality or inequality constraints or both are considered.

4.1 Kuhn-Tucker Conditions

We assume that objective function $f(x)$ (x is an N-dimensional array), inequality constraints $g_j(x), j = 1, 2, \ldots, J$, and equality constraints $h_k(x), k = 1, 2, \ldots, K$ are all differentiable. The NLP problem is as follows:

$$\begin{array}{rll} \text{Minimize} & f(x) & \\ \text{Subject to} & g_j(x) \geq 0 & j = 1, 2 \ldots, J \\ & h_k(x) = 0 & k = 1, 2 \ldots, K \end{array}$$

The Kuhn-Tucker problem is to find vectors x (size N), u (size J), and v (size K) that satisfy

$$\begin{array}{rcll} \nabla f(x) - \sum_{j=1}^{J} u_j \nabla g_j(x) - \sum_{k=1}^{K} \nabla h_k(x) & = & 0 & N \text{ equations} \\ g_j(x) & \geq & 0 & j = 1, 2, \ldots, J \\ h_k(x) & = & 0 & k = 1, 2, \ldots, K \\ u_j g_j(x) & = & 0 & j = 1, 2, \ldots, J \\ u_j & \geq & 0 & j = 1, 2, \ldots, J \end{array}$$

If $g_j(\bar{x}) = 0$, the constraint j is active or binding at $\bar{x}$.
If $g_j(\bar{x}) > 0$, the constraint j is inactive or nonbinding at $\bar{x}$.

Kuhn-Tucker Necessity Theorem:
Consider the NLP problem shown above. Let f, g, and h be differentiable functions and x^* be a feasible solution to NLP. Let $I = \{j | g_j(x^*) = 0\}$. Furthermore, $\nabla g_j(x^*)$ for $j \in I$ and $\nabla h_k(x^*)$ for $k = 1, 2, \ldots, K$ are linearly independent (*Constraint qualification*). If x^* is an optimal solution to NLP, there exists a (u^*, v^*) such that (x^*, u^*, v^*) solves Kuhn-Tucker problem.

If a feasible point does not satisfy constraint qualification, K-T necessity theorem can be used to prove that the point is not optimal but not vice versa.

Kuhn-Tucker Sufficiency Theorem:
Let the objective function be convex, the inequality constraints $g_j(x)$ be all concave functions for $j = 1, 2, \ldots, J$ and equality constraints $h_k(x)$ for $k = 1, 2, \ldots, K$ be linear. If there exists a solution (x^*, u^*, v^*) that satisfies the K-T conditions, then x^* is an optimal solution to the NLP problem.

4.2 Transformation Methods

- Original constrained problem is transformed into a sequence of unconstrained problems via penalty functions
- Methods vary according to the way constraints are handled.
  Interior penalty method: Each sequence contains feasible points
  Exterior penalty method: Each sequence contains infeasible points
  Mixed penalty method: Each sequence contains feasible or infeasible points

4.2.1 Penalty Function Method

- Penalty concept:
$$P(x, R) = f(x) + \Omega(R, g(x), h(x))$$
where R is a set of penalty parameters, Ω is the penalty term so selected to favor the selection of feasible points over infeasible points:

  1. Parabolic Penalty: $\Omega = R\{h(x)\}^2$. Used for equality constraints and it is an exterior penalty term. A small value of R is started with and increased gradually.
  2. Infinite Barrier Penalty: $\Omega = 10^{20} \sum_{j \in \bar{J}} |g_j(x)|$. It assigns an infinite penalty to infeasible points.
  3. Log Penalty: $\Omega = -R \ln[g(x)]$. A large value of R is started with and reduced to zero gradually. It is an interior penalty term.
  4. Inverse Penalty: $\Omega = R\left[\frac{1}{g(x)}\right]$. Like Log penalty, the value of R starts from large value and reduces to zero. It is also an interior penalty term.
  5. Bracket Operator Penalty: $\Omega = R\langle g(x)\rangle^2$, where $\langle \alpha \rangle = \alpha$, when α is negative; zero, otherwise. Here R starts from a small value and increases to a large value. This is an exterior penalty term.

Algorithm:

Step 1 Choose two termination parameters ϵ_1, ϵ_2, an initial solution $x^{(0)}$, a penalty term Ω, and an initial penalty parameter $R^{(0)}$. Choose a parameter c to update R such that $0 < c < 1$ is used for interior penalty terms and $c > 1$ is used for exterior penalty terms. Set $t = 0$

Step 2 Form $P(x^{(t)}, R^{(t)}) = f(x^{(t)}) + \Omega(R^{(t)}, g(x^{(t)}), h(x^{(t)}))$

Step 3 Starting with a solution $x^{(t)}$, find $x^{(t+1)}$ such that $P(x^{(t+1)}, R^{(t)})$ is minimum for a fixed value of $R^{(t)}$. Use ϵ_1 to terminate the unconstrained search

Step 4 Is $|P(x^{(t+1)}, R^{(t)}) - P(x^{(t)}, R^{(t-1)})| \leq \epsilon_2$?
If yes, set $x^T = x^{(t+1)}$ and **Terminate**

Else go to Step 5

Step 5 Choose $R^{(t+1)} = cR^{(t)}$. Set $t = t + 1$ and go to Step 2

Consider the constrained Himmelblau's function:

$$\text{Minimize} \quad (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$$

subject to

$$(x_1 - 5)^2 + x_2^2 - 26 \geq 0, \qquad x_1, x_2 \geq 0.$$

The inclusion of the constraint changes the unconstrained optimum point. The feasible region and the optimum point of this NLP is shown in Figure 4.1. The figure shows that the original optimum point $(3, 2)^T$ is now an infeasible point. The new optimum is a point on the constraint line that touches a contour line at that point.

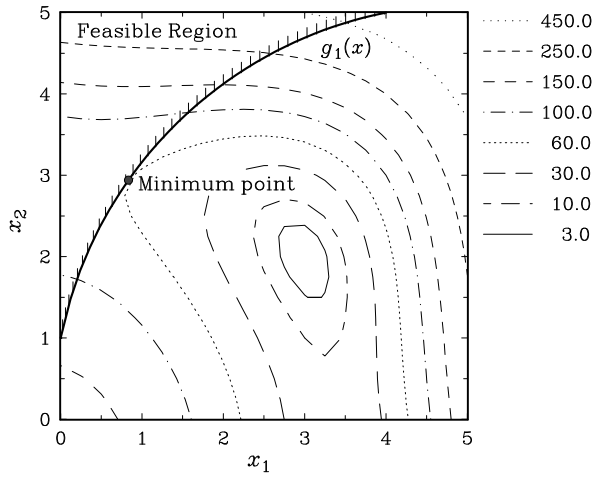


Figure 4.1: The feasible search space and the true minimum of the constrained problem. The hatches on the constraint boundary are wrongly placed on this plot, but the 'Feasible Region' is correctly marked.

Step 1 We use the bracket-operator penalty term to solve this problem. The bracket operator penalty term is an exterior penalty term. We choose an infeasible point $x^{(0)} = (0, 0)^T$ as the initial point. We also choose a small value for the penalty parameter: $R^{(0)} = 0.1$. We choose two convergence parameters $\epsilon_1 = \epsilon_2 = 10^{-5}$.

Step 2 The next task is to form the penalty function:

$$\begin{aligned} P(x, R^{(0)}) &= (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2 \\ &\quad + 0.1 \times \langle (x_1 - 5)^2 + x_2^2 - 26 \rangle^2. \end{aligned}$$

In the above formulation, the variable bounds must also be included as inequality constraints. For clarity and ease of illustration, we have not included the variable bounds in our formulation.

Step 3 The above unconstrained function can now be minimized using one of the methods described in the previous chapter. Here, we use the steepest descent method to solve the above problem (the FORTRAN code presented at the end of Chapter 3 is used). We begin the algorithm with an initial solution $x^{(0)} = (0, 0)^T$ having $f(x^{(0)}) = 170.0$. At this point, the constraint violation is -1.0 and the penalized function value $P(x^{(0)}, R^{(0)}) = 170.100$. Intermediate points obtained by the steepest descent algorithm are tabulated in Table 4.1, and some of these points are shown in Figure 4.2. After 150 function evaluations, the solution $x^* = (2.628, 2.475)^T$ having a function value equal to $f(x^*) = 5.709$ is obtained. At this point, the constraint violation is equal to -14.248, but has a penalized function value equal to 25.996, which is smaller than that at the initial point. Even though the constraint violation at this point is greater than that at the initial point, the steepest descent method has minimized the penalized function $P(x, R^{(0)})$ from 170.100 to 25.996. We set $x^{(1)} = (2.628, 2.475)^T$ and proceed to the next step.

Table 4.1: Tabulation of the Intermediate Points Obtained Using the Steepest Descent Algorithm in Exercise 4.2.1

Sequence t	$R^{(t)}$	Solution $x^{(t)}$	$P(x^{(t)}, R^{(t)})$	Constraint violation
1	0.1	$(0, 0)^T$	170.100	-1.000
		$(2.569, 2.294)^T$	27.119	-14.828
		$(2.657, 2.455)^T$	26.019	-14.483
		$\vdots$	$\vdots$	$\vdots$
		$(2.628, 2.475)^T$	25.996	-14.248
2	1.0	$(2.628, 2.475)^T$	208.70	-14.248
		$(1.730, 3.412)^T$	75.140	-3.655
		$(1.166, 2.871)^T$	60.986	-3.058
		$\vdots$	$\vdots$	$\vdots$
		$(1.011, 2.939)^T$	58.757	-1.450
3	10.0	$(1.011, 2.939)^T$	77.591	-1.450
		$(0.906, 3.016)^T$	60.530	-0.143
		$\vdots$	$\vdots$	$\vdots$
		$(0.844, 2.934)^T$	60.233	-0.119

Step 4 Since this is the first iteration, we have no previous penalized function value to compare with; thus we move to Step 5.

Step 5 At this step, we update the penalty parameter $R^{(1)} = 10 \times 0.1 = 1.0$ and move to Step 2. This is the end of the first sequence. It is important here to note that with a different initial point, we could have also converged to the same point. But simulation runs with certain initial points may have taken a longer time to converge than with other points. However, solutions in subsequent sequences will be identical for all simulations.

Step 2 The new penalized function in the second sequence is as follows:

$$\begin{aligned} P(x, R^{(1)}) &= (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2 \\ &\quad +1.0 \times \langle (x_1 - 5)^2 + x_2^2 - 26 \rangle^2. \end{aligned}$$

Step 3 At this step, we once again use the steepest descent method to solve the above problem from the starting point $(2.628, 2.475)^T$. Table 4.1 shows intermediate points of the simulation run. The minimum of the function is found after 340 function evaluations and is $x^{(2)} = (1.011, 2.939)^T$. At this point, the constraint violation is equal to -1.450, which suggests that the point is still an infeasible point. The penalized function and the minimum of the function are both shown in Figure 4.3. The progress of the previous sequence is also shown using dashed lines. Observe that this penalized function is distorted with respect to the original Himmelblau function. This distortion is necessary to shift the minimum point of the current function closer to the true constrained minimum point. Also notice that the penalized function at the feasible region is undistorted.

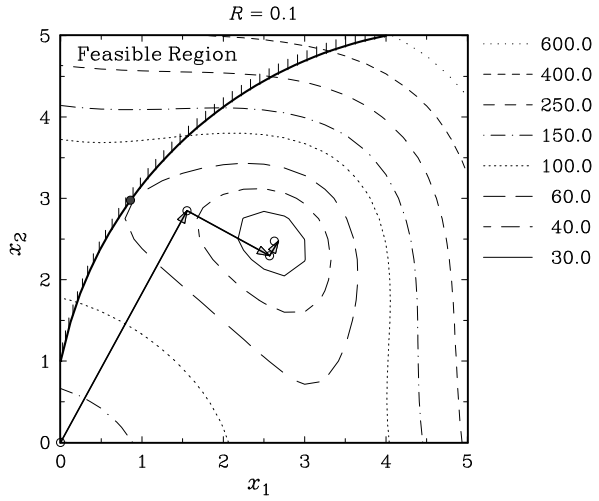


Figure 4.2: A simulation of the steepest descent method on the penalized function with $R = 0.1$. The hatches on the constraint boundary are wrongly placed on this plot, but the 'Feasible Region' is correctly marked.

Step 4 Comparing the penalized function values, we observe that $P(x^{(2)}, 1.0) = 58.664$ and $P(x^{(1)}, 0.1) = 25.996$. Since they are very different from each other, we continue with Step 5.

Step 5 The new value of the penalty parameter is $R^{(2)} = 10.0$. We increment the iteration counter $t = 2$ and go to Step 2.

In the next sequence, the penalized function is formed with $R^{(2)} = 10.0$. The penalized function and the corresponding solution is shown in Figure 4.4. This time the steepest descent algorithm starts with an initial solution $x^{(2)}$. The minimum point of the sequence is found to be $x^{(3)} = (0.844, 2.934)^T$ with a constraint violation equal to -0.119. Figure 4.4 shows the extent of distortion of the original objective function. Compare the contour levels shown at the top right corner of Figures 4.2 and 4.4. With $R = 10.0$, the effect of the objective function $f(x)$ is almost insignificant compared to that of the constraint violation in the infeasible search region. Thus, the contour lines are almost parallel to the constraint line. Fortunately in this problem, the increase in the penalty parameter R only makes the penalty function steeper in the infeasible search region. In problems with a sufficiently nonlinear objective function and with multiple constraints, a large value of the penalty parameter may create one or more artificial local optima in the search space, thereby making it difficult for the unconstrained search to obtain the correct solution. The advantage of using the unconstrained search method sequentially is that the unconstrained search is always started from the best point found in the previous sequence. Thus, despite the presence of many local optima in the search space, the search at every sequence is initiated from a point near the correct optimum point. This makes it easier for the unconstrained search to find the correct solution.

After another sequence (iteration) of this algorithm, the obtained solution is

$$x^{(4)} = (0.836, 2.940)^T$$

with a constraint violation of only -0.012. This point is very close to the true constrained optimum solution. A few more iterations of the penalty function method may be performed to get a solution with the desired accuracy. Although a convergence check with a small difference in the penalized function value at two consecutive sequences is used in this algorithm, any

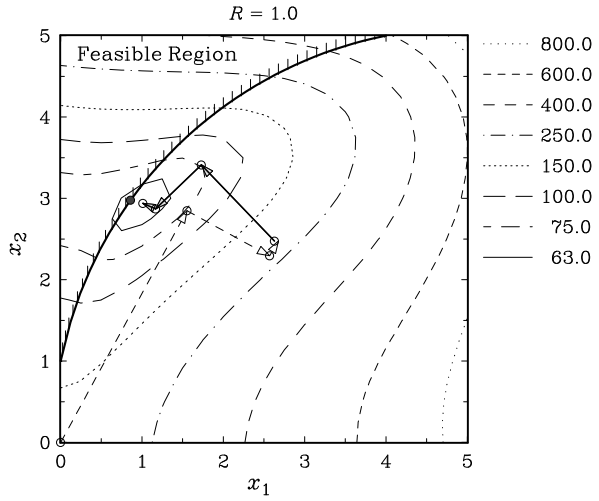


Figure 4.3: Intermediate points using the steepest descent method for the penalized function with $R = 1.0$ (solid lines). The hatches on the constraint boundary are wrongly placed on this plot, but the 'Feasible Region' is correctly marked.

other convergence criteria (for example, a small difference in the x-vector of the solutions in two successive sequences) may also be used. At the end of this chapter, we present a FORTRAN code implementing the above penalty function method.

In the presence of multiple constraints, it is observed that the performance of the penalty function method improves considerably if the constraints and the objective functions are first normalized before constructing the penalized function. An inequality constraint $g_j(x) \geq 0$ can be normalized as follows:

$$\frac{g_j(x)}{g_{\max}} \geq 0,$$

where $g_{\max}$ is the maximum value of the constraint $g_j(x)$ in the search space. Often, engineering design problems contain constraints restraining resource or capacity of b_j as $g'_j(x) \leq b_j$. The constraint can be normalized as follows:

$$1 - \frac{g'_j(x)}{b_j} \geq 0.$$

If an upper bound of the objective function is known, the objective function can also be normalized as shown above, but the normalization of the constraints is more important than that of the objective function.

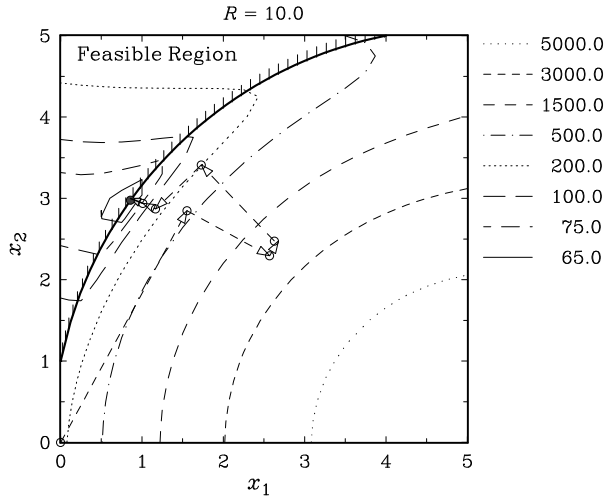


Figure 4.4: Intermediate points obtained using the steepest descent method for the penalized function with $R = 10.0$ (solid lines near the true optimum). Notice the distortion in the function. The hatches on the constraint boundary are wrongly placed on this plot, but the 'Feasible Region' is correctly marked.

4.2.2 Method of Multipliers

- Distortion of functions is avoided by using fixed parameter penalty method
- The following penalty function and update rules are used:

$$\begin{aligned} P(x, \sigma^{(t)}, \tau^{(t)}) &= f(x) + R \sum_{j=1}^{J} \left\{ \left(\langle g_j(x) + \sigma_j^{(t)} \rangle \right)^2 - \left(\sigma_j^{(t)} \right)^2 \right\} \\ &\quad + R \sum_{k=1}^{K} \left\{ \left(h_k(x) + \tau_k^{(t)} \right)^2 - \left(\tau_k^{(t)} \right)^2 \right\}. \end{aligned}$$

The final solution x^T is a K-T point. Furthermore, the Lagrange multipliers are computed easily:

$$\begin{aligned} u_j &= -2R\sigma_j^T \\ v_k &= -2R\tau_k^T \end{aligned}$$

Algorithm:

Step 1 Choose a penalty parameter R, termination parameters ϵ_1 and ϵ_2. Choose an initial solution $x^{(0)}$. Set multipliers $\sigma_j^{(0)} = \tau_k^{(0)} = 0$ and the iteration counter $t = 0$

Step 2 Next, form the penalized function:

$$\begin{aligned} P(x, \sigma^{(t)}, \tau^{(t)}) &= f(x) + R \sum_{j=1}^{J} \left\{ \left(\langle g_j(x) + \sigma_j^{(t)} \rangle \right)^2 - \left(\sigma_j^{(t)} \right)^2 \right\} \\ &\quad + R \sum_{k=1}^{K} \left\{ \left(h_k(x) + \tau_k^{(t)} \right)^2 - \left(\tau_k^{(t)} \right)^2 \right\}. \end{aligned}$$

Step 3 Use an unconstrained optimization technique to solve the penalized function $P(x, \sigma^{(t)}, \tau^{(t)})$ from the starting point $x^{(t)}$ with an convergence factor ϵ_1. During this optimization, $\sigma^{(t)}$ and $\tau^{(t)}$ are kept fixed; only the vector x is varied. Let us say the solution is $x^{(t+1)}$

Step 4 Is $\|P(x^{(t+1)}, \sigma^{(t)}, \tau^{(t)}) - P(x^{(t)}, \sigma^{(t-1)}, \tau^{(t-1)})\| \leq \epsilon_2$?
If yes, set $x^T = x^{(t+1)}$ and **Terminate**

Else go to Step 5

Step 5 Update $\sigma_j^{(t+1)} = \langle g_j(x^{(t)}) + \sigma_j^{(t)} \rangle$ for all $j = 1, 2, \ldots, J$ and $\tau_k^{(t+1)} = h_k(x^{(t)}) + \tau_k^{(t)}$ for all $k = 1, 2, \ldots, K$. Set $t = t + 1$ and go to Step 2

4.3 Constrained Direct Search

- Detailed structure of constraints are considered
- Functions may be discontinuous and non-differentiable
- Algorithms are heuristic in nature
- Algorithms start from a feasible point. When a new point is created by a fixed rule make sure that the point is feasible. If not, modify by a predefined rule.

4.3.1 Random Search Methods

Points are generated at random.

Luus and Jaakola Algorithm:

Step 1 Given an initial feasible point x^0, an initial range z^0 such that the minimum, x^*, lies in $(x^0 - \frac{1}{2}z^0, x^0 + \frac{1}{2}z^0)$. Choose a parameter $0 < \epsilon < 1$. For each of Q blocks, initially set $q = 1$ and $p = 1$

Step 2 For $i = 1, 2, \ldots, N$, create points using a uniform distribution of r in the range $(-0.5, 0.5)$. Set $x_i^{(p)} = x_i^{q-1} + r z_i^{q-1}$

Step 3 If $x^{(p)}$ is infeasible and $p < P$, repeat Step 2
If $x^{(p)}$ is feasible, save $x^{(p)}$ and $f(x^{(p)})$, increment p and repeat Step 2

Else if $p = P$, set x^q to be the point that has the lowest $f(x^{(p)})$ over all feasible $x^{(p)}$ including x^{q-1} and reset $p = 1$

Step 4 Reduce the range via $z_i^q = (1 - \epsilon) z_i^{q-1}$

Step 5 If $q > Q$, **Terminate**

Else increment q and continue with Step 2

Suggested values of parameters are $\epsilon = 0.05$, $P = 100$, and Q is related to the desired reduction in variable uncertainity.

4.4 Linearization Methods

- Problems are converted into linear constrained problems
- At $x^{(0)}$, the function is approximated as

$$f(x) = f(x^{(0)}) + \nabla f(x^{(0)})(x - x^{(0)}).$$

- Linear Programming methodology is used recursively

4.4.1 Frank-Wolfe Algorithm:

Used for linearly constrained case:

$$\begin{array}{rl} \text{Minimize} & f(x) \quad \text{(could be nonlinear)} \\ \text{Subject to} & Ax \leq b \\ & x \geq 0 \end{array}$$

Algorithm: (Assumes convex feasible region)

Step 1 Given an initial point $x^{(0)}$, two convergence parameters ϵ and δ. Set an iteration counter $t = 0$

Step 2 Calculate $\nabla f(x^{(t)})$. If $\|\nabla f(x^{(t)})\| \leq \epsilon$, **Terminate**

Else go to Step 3

Step 3 Solve the following LP problem:

$$\begin{array}{rll} \text{Minimize} & f(x^{(t)}) + \nabla f(x^{(t)})(x - x^{(t)}) & \\ \text{subject to} & & \\ & g_j(x^{(t)}) + \nabla g_j(x^{(t)})(x - x^{(t)}) \geq 0, & j = 1, 2, \ldots, J; \\ & h_k(x^{(t)}) + \nabla h_k(x^{(t)})(x - x^{(t)}) = 0, & k = 1, 2, \ldots, K; \\ & x_i^{(L)} \leq x_i \leq x_i^{(U)}, & i = 1, 2, \ldots, N. \end{array}$$

Let $y^{(t)}$ be the optimal solution to the above LP problem

Step 4 Find $\alpha^{(t)}$ that minimizes $f(x^{(t)} + \alpha(y^{(t)} - x^{(t)}))$ in the range $\alpha \in (0, 1)$

Step 5 Calculate $x^{(t+1)} = x^{(t)} + \alpha^{(t)}(y^{(t)} - x^{(t)})$

Step 6 If $\|x^{(t+1)} - x^{(t)}\| < \delta\|x^{(t)}\|$ and if $\|f(x^{(t+1)}) - f(x^{(t)})\| < \epsilon\|f(x^{(t)}\|$, **Terminate**

Else $t = t + 1$ and go to Step 2

Comments:

1. Frank-Wolf algorithm is similar to unconstrained gradient search methods
2. Convergence to a Kuhn-Tucker point is proved by Zangwill.
3. In general NLP case, the following functions and constraints may be used:

$$\begin{array}{rll} \text{Minimize} & f(x^{(t)} + \nabla f(x^{(t)})(x - x^{(t)}) & \\ \text{Subject to} & g_j(x^{(t)}) + \nabla g_j(x^{(t)})(x - x^{(t)}) \geq 0 & \forall j \\ & h_k(x^{(t)}) + \nabla h_k(x^{(t)})(x - x^{(t)}) = 0 & \forall k \\ & x_i^{(L)} \leq x_i \leq x_i^{(U)} & \end{array}$$

4.4.2 Cutting Plane Method

- Used only for inequality constraints
- The feasible region is expected to be convex
- Problem is transformed into a linearly constrained problem
- Starting from assumed large search space, constraints are added and the search space is reduced, each time solving the LP problem

Given a problem:

$$\begin{array}{rl} \text{Minimize} & \sum_{i=1}^{N} c_i x_i \\ \text{Subject to} & g_j(x) \geq 0 \\ & x_i^{(L)} \leq x_i \leq x_i^{(U)} \end{array}$$

a suitable LP and a constraint tolerance $\epsilon > 0$ and initial bound

$$Z^0 = \{x : x_i^{(L)} \leq x_i \leq x_i^{(U)}, i = 1, 2, \ldots, N\}$$

such that $Z^0 \supset$ Feasible region.
Algorithm:

Step 1 Choose a constraint tolerance ϵ and an initial search space Z^0. Solve the following LP problem:

$$\begin{array}{rl} \text{Minimize} & \sum_{i=1}^{N} c_i x_i \\ \text{subject to} & x \in Z^0. \end{array}$$

Let us say that the solution is $x^{(1)}$. Set a counter $k = 1$

Step 2 Find m such that

$$-g_m(x^{(k)}) = \max\left[-g_j(x^{(k)}), 0; \quad j = 1, 2, \ldots, J\right].$$

If $g_m(x^{(k)}) > -\epsilon$, **Terminate**

Else go to Step 3

Step 3 Construct a cutting plane:

$$p^{(k)}(x) \equiv \tilde{g}_m(x; x^{(k)}) = g_m(x^{(k)}) + \nabla g_m(x^{(k)})(x - x^{(k)}) \geq 0.$$

Let $H^{(k)}$ define the space $H^{(k)} = \{x : p^{(k)}(x) \geq 0\}$. Solve the following LP problem:

$$\begin{array}{rl} \text{Minimize} & \sum_{i=1}^{N} c_i x_i \\ \text{subject to} & x \in Z^{(k-1)} \cap H^{(k)}. \end{array}$$

Designate the solution $x^{(k+1)}$

Step 4 Set $Z^{(k)} = Z^{(k-1)} \cap H^{(k)}$, $k = k + 1$, and go to Step 2

Extension: If $f(x)$ is nonlinear, introduce a new variable x_0

$$x_0 \geq f(x)$$

and the NLP problem becomes as follows:

$$\begin{array}{rl} \text{Minimize} & x_0 \\ \text{Subject to} & g_j(x) \geq 0 \quad \text{for } j = 1, 2, \ldots, J \\ & x_0 - f(x) \geq 0 \\ & x_i^{(L)} \leq x_i \leq x_i^{(U)} \end{array}$$

Above algorithm can now be used on this problem.
Comments:

1. Algorithm generates a sequence of infeasible points, hence can not be terminated early.

2. Number of constraints increases by one at each iteration

3. Use cut-deletion procedure, that deletes constraints generated prior to subproblem t provided they are both nonbinding at $x^{(t+1)}$ and that there is an improvement in function value:
   Let $I^{(t)}$ be the set of indices of cutting planes accumulated but not yet deleted by iteration t. Replace step 4 with following:

   Step 4a Determine the set D of cutting planes to be deleted. For each $i \in I^{(t)} \cup \{t\}$, if both
   $$p^{(i)}(x^{(t+1)}) > 0 (> \epsilon) \quad \text{and} \quad f(x^{(t+1)}) \geq f(x^{(i)}) + p^{(i)}(x^{(i)})$$
   then include i in the set D. Let $I^{(t+1)} = I^{(t)} \cup \{t\} - D$

   Step 4b Set $Z^{t+1} = Z^0 \cap \{x : p^{(i)}(x) \geq 0, \quad i \in I^{(t+1)}\}$, $t = t + 1$, and Go to step 2

4.5 Method of Feasible Directions

- Rather than relying on solutions to LP problems, use linear approximations to determine a locally good search direction
- Similar to gradient-based unconstrained search
- **Zoutendijk**'s method: Have a search direction d that is descent ($\nabla f(x^{(t)}) \cdot d < 0$) and feasible ($\nabla g_j(x^{(t)}) \cdot d \geq 0$).

Algorithm:

Step 1 Set an iteration counter $t = 0$. Choose an initial feasible point $x^{(0)}$ and a parameter for checking the constraint violation, ϵ

Step 2 At the current point $x^{(t)}$, let $I^{(t)}$ be the set of indices of active constraints. In other words,

$$I^{(t)} = \{j : 0 \leq g_j(x^{(t)}) \leq \epsilon, \quad j = 1, 2, \ldots, J\}.$$

If $I^{(t)}$ is empty, use $d^{(t)} = \theta^{(t)} = -\nabla f(x^{(t)})$, normalize $d^{(t)}$ and go to Step 4

Step 3 Solve the following LP:

$$\begin{array}{ll} \text{Maximize} & \theta \\ \text{subject to} & \\ & \nabla f(x^{(t)})d \leq -\theta, \\ & \nabla g_j(x^{(t)})d \geq \theta, \quad j \in I^{(t)}; \\ & -1 \leq d_i \leq 1, \quad i = 1, 2, \ldots, N. \end{array}$$

Label the solution $d^{(t)}$ and $\theta^{(t)}$

Step 4 If $\theta^{(t)} \leq 0$, **Terminate**

Else $\bar{\alpha} = \min[\alpha : g_j(x^{(t)} + \alpha d^{(t)}) = 0, j = 1, 2, \ldots, J, \text{and } \alpha > 0]$.
If no $\bar{\alpha} > 0$ exists, set $\bar{\alpha} = \infty$

Step 5 Find $\alpha^{(t)}$ such that

$$f(x^{(t)} + \alpha^{(t)} d^{(t)}) = \min[f(x^{(t)} + \alpha d^{(t)}) : 0 \leq \alpha \leq \bar{\alpha}].$$

Set $x^{(t+1)} = x^{(t)} + \alpha^{(t)} d^{(t)}$, $t = t + 1$, and go to Step 2

Comments:

1. Only a subset of constraints are used to define a subproblem, thus LP is smaller
2. Since only binding constraints are used, zigzag iteration pattern results

Consider the constrained Himmelblau function again:

$$\text{Minimize} \quad f(x) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$$

subject to

$$\begin{array}{rcl} g_1(x) & = & 26 - (x_1 - 5)^2 - x_2^2 \geq 0, \\ g_2(x) & = & 20 - 4x_1 - x_2 \geq 0, \\ x_1, x_2 & \geq & 0. \end{array}$$

Let us recall that the minimum point of the above problem lies at $x^* = (3, 2)^T$ with a function value equal to zero.

Step 1 Let us choose an initial feasible point $x^{(0)} = (0, 0)^T$ and a tolerance parameter $\epsilon = 10^{-3}$. We also set the iteration counter $t = 0$.

Step 2 At this step, let us find the active constraints at point $x^{(0)}$. It turns out that only variable bounds are active. Calling these two inequality constraints $g_3(x) = x_1 \geq 0$ and $g_4(x) = x_2 \geq 0$, we update the active constraint set $I^{(0)} = \{3, 4\}$. Since this set is not empty, we continue with Step 3.

Step 3 At this step, we have to find a descent direction which is maximally away from both constraints g_3 and g_4. At the initial point, both active constraints are orthogonal to each other. Thus, the desired search direction may make equal angle with all active constraints. But in any other situation, this may not be true. Thus, we find the optimal search direction by solving an LP problem. Calculating the derivative of the objective function and the active constraints at $x^{(0)}$ numerically and denoting the search direction $d = (d_1, d_2)^T$, we obtain the following LP problem:

$$\text{Maximize} \quad \theta$$

subject to

$$\begin{gathered} -14d_1 - 22d_2 \leq -\theta, \\ d_1 \geq \theta, \\ d_2 \geq \theta, \\ -1 \leq d_1, d_2 \leq 1. \end{gathered}$$

There exist a number of difficulties with the above formulation to be directly solved using the simplex method of LP technique. First of all, in the above formulation, the variables can take negative values, which are not allowed in a LP technique. Thus, we first substitute $t_i = d_i + 1$ for $i = 1, 2$ such that the variables t_i can take only positive values in the range $(0, 2)$. We rewrite the above LP problem in terms of the new variables:

$$\text{Maximize} \quad \theta$$

subject to

$$\begin{gathered} 14t_1 + 22t_2 - \theta \geq 36, \\ t_1 - \theta \geq 1, \\ t_2 - \theta \geq 1, \\ 0 \leq t_1, t_2 \leq 2. \end{gathered}$$

Secondly, the simplex method can handle only equality constraints. Slack variables are usually added or subtracted to convert inequality constraints to equality constraints. Therefore, for each of the above constraints we add a slack variable (y_1 to y_5). Thirdly, we observe that the problem variables and the slack variables do not constitute an initial basic feasible solution (Refer to the Appendix for details). Thus, we introduce three more artificial variables (y_6, y_7, and y_8) to constitute an initial basic feasible solution for the first phase of the dual simplex search method. Thus, the underlying LP problem becomes as follows:

$$\left.\begin{array}{rc} & \text{Maximize} \quad \theta \\ \text{subject to} & \\ & \begin{gathered} 14t_1 + 22t_2 - \theta - y_1 + y_6 = 36, \\ t_1 - \theta - y_2 + y_7 = 1, \\ t_2 - \theta - y_3 + y_8 = 1, \\ t_1 + y_4 = 2, \\ t_2 + y_5 = 2, \\ t_1, t_2, y_1, y_2, t_3, y_4, y_5, y_6, y_7, y_8 \geq 0. \end{gathered} \end{array}\right\} \tag{4.1}$$

The three-variable problem of Exercise 4.6.1 now becomes an 11-variable problem. At first, we solve the above problem for the objective:

$$\text{Maximize} \quad -(y_6 + y_7 + y_8). \tag{4.2}$$

Since all artificial variables must also be nonnegative, the solution to the above problem would have $y_6 = y_7 = y_8 = 0$, because the above objective function at this point would be zero. This solution will then be a candidate solution for the initial basic feasible solution of the problem presented in Equation (4.1). The dual phase method is described in the Appendix. The successive tables for the first problem of obtaining a feasible starting solution are shown in Tables 4.5 to 4.8.

Table 4.2: The First Tableau for the First Phase of the Dual Simplex Search Method

		0	0	0	0	0	0	0	0	−1	−1	−1		
c_B	Basic	t_1	t_2	θ	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8		
−1	y_6	14	22	−1	−1	0	0	0	0	1	0	0	36	$\frac{36}{22} = 1.64$
−1	y_7	1	0	−1	0	−1	0	0	0	0	1	0	1	$\frac{1}{0} = \infty$
−1	y_8	0	1	−1	0	0	−1	0	0	0	0	1	1	$\frac{1}{1} = 1$ ←
0	y_4	1	0	0	0	0	0	1	0	0	0	0	2	$\frac{2}{0} = \infty$
0	y_5	0	1	0	0	0	0	0	1	0	0	0	2	$\frac{2}{1} = 2$
$(\Delta f)_q$		15	23	−3	−1	−1	−1	0	0	0	0	0	$f = -38$	
			↑											

The objective function value at the Table 4.5 is $f = -38$. (Equation (4.2) is used as the objective.) In the table, it is clear that the nonbasic variable t_2 corresponds to a maximum increase in the function value. (The quantity $(\Delta f)_q$ is larger for t_2.) Thus, we choose t_2 as the new basic variable.

It turns out from the minimum ratio rule (see the Appendix for the definition of the rule) that the basic variable y_8 must be replaced by the the variable t_2 in the next iteration. We formulate the next row-echelon matrix. The calculation procedure for forming the row-echelon format is also discussed in the Appendix. The outcome of the calculation is shown in Table 4.6.

Note that the objective function value has improved considerably from the previous iteration. Here, we also observe that the nonbasic variable y_3 corresponds to the maximum value of the quantity $(\Delta f)_q$. Thus, we choose y_3 as the new basic variable. Using the minimum ratio rule, we also observe that the current basic variable y_6 must be replaced by the variable y_3 (Table 4.7). At the end of the third iteration, we observe that the nonbasic variable t_1 must replace the basic variable y_7. The objective function value at this iteration is $f = -1$. We form the next row-echelon matrix in Table 4.8. At this stage, we observe that all artificial variables are zero and the objective function is also equal to zero. This is the termination criterion for the first phase of the dual phase method. The solution of the above iteration is $t_1 = 1$, $t_2 = 1$, $y_1 = 0$, $y_2 = 0$, $y_3 = 0$, $y_4 = 1$, and $y_5 = 1$. This solution was not obvious in the formulation of the problem presented in Equation (4.1).

We begin the second phase with the above solution as the initial solution. The objective in the second phase is to maximize the original function: $f(x) = \theta$. Since the artificial variables are no more required, we discontinue with them in subsequent computations. The row-echelon matrix is shown in Table 4.9. The objective function value at this iteration is $f(x) = 0$. Table 4.6 shows that the basic variable y_3 must be replaced by the nonbasic variable θ.

Table 4.3: The Second Tableau for the First Phase of the Dual Simplex Search Method

		0	0	0	0	0	0	0	0	−1	−1	−1		
c_B	Basic	t_1	t_2	θ	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8		
−1	y_6	14	0	21	−1	0	**22**	0	0	1	0	−22	14	0.64 ←
−1	y_7	1	0	−1	0	−1	0	0	0	0	1	0	1	$\frac{1}{0} = \infty$
0	t_2	0	1	−1	0	0	−1	0	0	0	0	1	1	−ve
0	y_4	1	0	0	0	0	0	1	0	0	0	0	2	$\frac{2}{0} = \infty$
0	y_5	0	0	1	0	0	1	0	1	0	0	−1	1	$\frac{1}{1} = 1$
$(\Delta f)_q$		15	0	20	−1	−1	22	0	0	0	0	−23	$f = -15$	
							↑							

Table 4.4: The Third Tableau for the First Phase of the Dual Simplex Search Method

		0	0	0	0	0	0	0	0	−1	−1	−1		
c_B	Basic	t_1	t_2	θ	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8		
0	y_3	0.64	0	0.95	−0.04	0	1	0	0	0.04	0	−1	0.64	
−1	y_7	**1.00**	0	−1.00	0.00	−1	0	0	0	0.00	1	0	1.00	←
0	t_2	0.64	1	−0.04	−0.04	0	0	0	0	0.04	0	0	1.64	
0	y_4	1.00	0	0.00	0.00	0	0	1	0	0.00	0	0	2.00	
0	y_5	−0.64	0	0.04	0.04	0	0	0	1	−0.04	0	0	0.36	
$(\Delta f)_q$		1.00	0	−1.00	0.00	−1	0	0	0	−1.00	0	−1	$f = -1$	
		↑												

Ratios: 1 (first row), 1 (second row), 2.57 (third row), 2 (fourth row),
and $-ve$ (fifth row)

The new set of basic and nonbasic variables are formed in Table 4.10. At this iteration, the objective function value does not change, but the algorithm moves into a new solution. The table shows that the nonbasic variable y_1 must become basic. The minimum ratio rule suggests that the basic variable y_4 must be replaced with the new basic variable y_1.

It is important to note that even though the ratio calculated in the right-most column for the first row is zero, it is considered to be negative. In the implementation of the linear programming method, care should be taken to check the sign for both numerator and denominator. If they are of opposite sign, that row must be excluded from consideration. We form the new row-echelon matrix in Table 4.11. In this table, the quantity $(\Delta f)_q$ corresponding to all nonbasic variables is nonpositive. Thus, we have obtained the optimum solution and we terminate the linear programming method. Thus, the final solution is $t_1 = 2$, $t_2 = 2$, $\theta = 1$, $y_1 = 35$, $y_2 = 0$, $y_3 = 0$, $y_4 = 0$, and $y_5 = 0$. These values satisfy all constraints in the problem presented in Equation (4.1). Now, we get back to Step 3 of the feasible direction search method.

Step 3 (cont.) The solution of the LP problem is $t = (2, 2)^T$ or $d^{(0)} = (1, 1)^T$ and $\theta^{(0)} = 1$. This solution implies that the resulting search direction makes equal angles with each of the two active constraints.

Step 4 Since $\theta^{(0)} = 1 > 0$, we do not terminate the algorithm. Instead, we calculate the limits along the direction $d^{(0)}$ before an infeasible point is found. Any generic point along $d^{(0)}$ from $x^{(0)}$ can be written as $x(\alpha) = x^{(0)} + \alpha d^{(0)}$ or $x(\alpha) = (\alpha, \alpha)^T$. The upper limit on α can be calculated by finding points along $d^{(0)}$ that intersect with each constraint. The problem of finding the intersection of a straight line and any generic curve can be posed as a root-finding problem, which can be solved using an optimization algorithm discussed in Chapter 2. We substitute the expression for $x_1 = \alpha$ and $x_2 = \alpha$ in each constraint and then minimize the following problem:

$$\text{Minimize} \quad \text{abs}\,[g_j(x(\alpha))]. \tag{4.3}$$

For example, the upper limit along $d^{(0)}$ can be found for the first constraint by minimizing the unidirectional function:

$$\text{abs}\,[26 - (\alpha - 5)^2 - \alpha^2].$$

Since the absolute value of the argument is always considered, the above function allows only positive values. Since we are looking for points for which the constraint has a value zero, those points correspond to the minimum value of the above expression. Note that the problem described in Equation (4.3) is a single-variable function. Thus, we

Table 4.5: The Fourth Tableau for the First Phase of the Dual Simplex Search Method

		0	0	0	0	0	0	0	0	−1	−1	−1	
c_B	Basic	t_1	t_2	θ	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	
0	y_3	0	0	1.59	−0.04	0.64	1	0	0	0.04	−0.64	−1	0
0	t_1	1	0	−1.00	0.00	−1.00	0	0	0	0.00	1.00	0	1
0	t_2	0	1	0.59	−0.04	0.64	0	0	0	0.04	−0.64	0	1
0	y_4	0	0	1.00	0.00	1.00	0	1	0	0.00	−1.00	0	1
0	y_5	0	0	−0.59	0.04	0.04	0	0	1	−0.04	0.64	0	1
$(\Delta f)_q$		0	0	0.00	0.00	0	0	0	0	−1.00	−1.00	−1	$f = 0$

Table 4.6: The First Tableau for the Second Phase of the Dual Simplex Search Method

		0	0	1	0	0	0	0	0		
c_B	Basic	t_1	t_2	θ	y_1	y_2	y_3	y_4	y_5		
0	y_3	0	0	1.59	−0.04	0.64	1	0	0	0	$\frac{0}{1.59} = 0$ ←
0	t_1	1	0	−1.00	0.00	−1.00	0	0	0	1	−ve
0	t_2	0	1	0.59	−0.04	0.64	0	0	0	1	$\frac{1}{0.59} = 1.69$
0	y_4	0	0	1.00	0.00	1.00	0	1	0	1	$\frac{1}{1} = 1$
0	y_5	0	0	−0.59	0.04	0.04	0	0	1	1	−ve
$(\Delta f)_q$		0	0	1.00	0.00	0.00	0	0	0	$f(x) = 0$	
				↑							

first bracket the minimum and then minimize the function. Using the bounding phase method from a starting point $\alpha^{(0)} = 5$ and $\Delta = 1$, we obtain the bracketing interval $(4, 6)$. Next, we use the golden section search in that interval to obtain the minimum point with three decimal places of accuracy: $\alpha_1^* = 5.098$. The same solution can also be obtained by solving the quadratic expression $g_1(x(\alpha)) = 0$. Similarly, the limit on the second constraint can also be calculated: $\alpha_2^* = 4.0$. Other constraints produce upper limits $\alpha_3^* = \alpha_4^* = 0$, which are not acceptable. Thus, the true upper limit is $\overline{\alpha} = 4.0$.

Step 5 Once the lower and upper limit on α are found, we perform another one dimensional search with the given objective function to find the minimum point along that direction. Using the golden section search, we obtain the minimum point in the interval $(0, 4)$: $\alpha^* = 2.541$, which corresponds to the new point $x^{(1)} = (2.541, 2.541)^T$. At this point, we increment the iteration counter and go to Step 2. This completes one iteration of the feasible direction method. The progress of this iteration is shown in Figure 4.5.

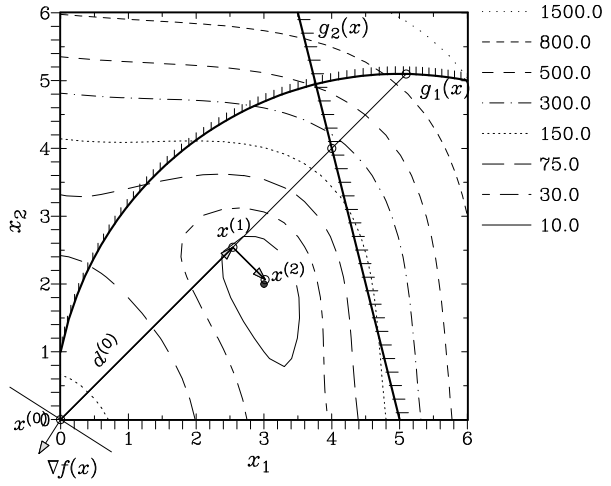


Figure 4.5: A number of iterations of the feasible direction method.

Step 2 At the new point, we find that no constraints are active. Thus, $I^{(1)} = \emptyset$, which means that the point is not on any constraint boundary and we are free to search in any direction locally. Therefore, we choose the steepest descent direction and the search direction is

Table 4.7: The Second Tableau for the Second Phase of the Dual Simplex Search Method

		0	0	1	0	0	0	0	0		
c_B	Basic	t_1	t_2	θ	y_1	y_2	y_3	y_4	y_5		
1	θ	0	0	1	−0.03	0.40	0.63	0	0	0	−ve
0	t_1	1	0	0	−0.03	−0.60	0.63	0	0	1	$\frac{1}{0} = \infty$
0	t_2	0	1	0	−0.03	0.40	−0.37	0	0	1	−ve
0	y_4	0	0	0	0.03	0.60	0.63	1	0	1	$\frac{1}{0.03} = 35$ ←
0	y_5	0	0	0	0.03	−0.40	0.37	0	1	1	$\frac{1}{0.03} = 35$
$(\Delta f)_q$		0	0	0	0.03	−0.4	−0.63	0	0	$f(x) = 0$	

↑

Table 4.8: The Third Tableau for the Second Phase of the Dual Simplex Search Method

		0	0	1	0	0	0	0	0	
c_B	Basic	t_1	t_2	θ	y_1	y_2	y_3	y_4	y_5	
1	θ	0	0	1	0	1	0	1	0	1
0	t_1	0	0	0	0	0	0	1	0	2
0	t_2	0	1	0	0	1	−1	1	0	2
0	y_1	0	0	0	1	21	−22	1	0	35
0	y_5	0	0	0	0	−1	1	−1	1	0
$(\Delta f)_q$		0	0	0	0	−1	0	−1	0	$f(x) = 1$

set according to the negative of the gradient of the objective function at the new point:

$$d^{(1)} = -\nabla f(x^{(1)}) = (16.323, -16.339)^T,$$

which is computed numerically. At this point the function value is $f(x^{(1)}) = 8.0$.

Step 4 Once again, we compute the upper limit along the search direction $d^{(1)}$. Posing the root-finding problem as an optimization problem as shown in the previous iteration, we obtain the parameter $\overline{\alpha} = \min[0.162, 0.149] = 0.149$.

Step 5 Performing a unidirectional search along $d^{(1)}$ in the domain $(0, 0.149)$, we obtain $\alpha^* = 0.029$. The corresponding point is $x^{(2)} = (3.018, 2.064)^T$ with an objective function value $f(x^{(2)}) = 0.107$.

This process continues until a point with a small derivative of the objective function is found. If the intermediate points fall on the constraint boundary frequently, this method may be expensive in terms of the overall computational time required to solve the problem. This situation may happen for problems with a narrow feasible region.

4.6 Generalized Reduced Gradient Method

- Implicit variable elimination algorithm using equality constraints or active inequality constraints
- If $K(< N)$ equations need be used, label first K variables as basic variables ($\hat{x}$) and remaining $N-K$ variables as nonbasic variables ($\bar{x}$).
- Partition the gradients of the constraints as follows:

$$\begin{aligned} J &= (\nabla\hat{h}_1, \nabla\hat{h}_2, \ldots \nabla\hat{h}_k)^T \quad K \times K \text{ matrix} \\ C &= (\nabla\bar{h}_1, \nabla\bar{h}_2, \ldots, \nabla\bar{h}_k)^T \quad (K \times N-K) \text{ matrix} \end{aligned}$$

- Using matrix algebra, the reduced gradient is calculated as:

$$\nabla\tilde{f}(x^{(t)}) = \nabla\bar{f}(x^{(t)}) - \nabla\hat{f}(x^{(t)})J^{-1}C \tag{4.4}$$

Algorithm:

Step 1 Choose an initial point $x^{(0)}$, a termination factor ϵ. Set $t = 0$. Transform all inequality constraints into equality constraints by adding slack variables

Step 2 Calculate the following quantity for all variables, $i = 1, 2, \ldots, N$:

$$y_i^{(t)} = \min\{(x_i^{(t)} - x_i^{(L)}), (x_i^{(U)} - x_i^{(t)})\}/(x_i^{(U)} - x_i^{(L)}).$$

Arrange $y_i^{(t)}$ in descending order of magnitude. Choose the first K variables as basic variables. Calculate the matrices J and C from the first derivative of constraints. Compute the reduced gradient $\nabla\tilde{f}(x^{(t)})$ using equation (4.4)

Step 3 If $\|\nabla\tilde{f}\| \leq \epsilon$, **Terminate**

Else set

$$\bar{d} = \begin{cases} 0, & \text{if } \bar{x}_i = x_i^{(L)} \text{ and } (\nabla\tilde{f})_i > 0; \\ 0, & \text{if } \bar{x}_i = x_i^{(U)} \text{ and } (\nabla\tilde{f})_i < 0; \\ -(\nabla\tilde{f})_i, & \text{otherwise.} \end{cases}$$

$\hat{d} = -J^{-1}C\bar{d}$, $\hat{d}$ is a column vector.

$$d = \begin{pmatrix} \hat{d} \\ \bar{d} \end{pmatrix}$$

Step 4 Find $\alpha^{(t)}$ such that $f(x^{(t)} + \alpha d)$ is minimum.
Set $x^{(t+1)} = x^{(t)} + \alpha^{(t)}d$, increment t, and go to Step 2

The basic algorithm may generate infeasible point at Step 4. Use an iterative procedure to find a feasible point.

Step 4 (revised) Set $\alpha = \alpha^0$. Set $i = 1$

(a) Calculate $v^{(i)} = x^{(t)} + \alpha d$. If $|h_k(v^{(i)})| \leq \epsilon_2; \quad k = 1, 2, \ldots, K$, go to Step 4(d)

Else continue

(b) Let $\hat{v}^{(i+1)} = \hat{v}^{(i)} - J^{-1}(v^{(i)}) \cdot h(v^{(i)})$ and $\bar{v}^{(i+1)} = \bar{v}^{(i)}$

(c) If $\|\hat{v}^{(i+1)} - \hat{v}^{(i)}\| > \epsilon_3$, set $i = i + 1$, and go to Step 4(b)

Else if $|h_k(v^{(i+1)})| \leq \epsilon_2$ for $k = 1, 2, \ldots, K$, go to Step 3(d)

Else set $\alpha = \gamma\alpha$ and go to Step 4(a)

(d) If $f(x^{(t)}) \leq f(v^{(i)})$ set $\alpha = \gamma\alpha$, $i = 1$, and go to Step 4(a)

Else set $x^{(t+1)} = v^{(i)}$, $t = t + 1$, and go to Step 2

4.7 Quadratic Approximation Method

- Quadratic approximation to objective functions
- Linear approximations to constraints to make the calculations easier

Algorithm:

Step 1 Given $x^{(0)}$ and a suitable method for solving QP problem

Step 2 Formulate QP

$$\begin{aligned} \text{Minimize} \quad & \nabla f(x^{(t)})^T d + \tfrac{1}{2} d^T \nabla^2 f(x^{(t)}) d \\ \text{Subject to} \quad & h_k(x^{(t)}) + \nabla h_k(x^{(t)})^T d = 0 \qquad k = 1, 2, \ldots, K \\ & g_j(x^{(t)}) + \nabla g_j(x^{(t)})^T d \geq 0 \qquad j = 1, 2, \ldots, J \end{aligned}$$

Step 3 Solve the QP problem and set $x^{(t+1)} = x^{(t)} + d$

Step 4 Check for convergence. If not converged, Go to step 1

- Lead to nonvertex solutions
- The term $\nabla^2 f(x^{(t)})$ in the objective function may be replaced by a Lagrangian term or a variable-metric approximation for Hessian for easier computation

This procedure can be repeatedly used to solve general non-linear programming problems by formulating a QP problem at the current best solution. This methodology is called Sequential Quadratic Programming or SQP. Many commercial optimization software implements this algorithm in which the resulting QP problem is solved using a quasi-Newton multi-variable optimization algorithm.

Chapter 5

Integer Programming

Used when some variables are integers.

5.1 Branch-and-Bound Algorithm

This method is widely used.
Algorithm:

Step 1 Assume that all variables are real. Solve the NLP problem. Let the solution be $\{x_I^{(t)}, x_R^{(t)}\}$. Initialize a counter $k = 2$

Step 2 For each unfathomed NLP, choose a particular integer variable $i \in I$ for which $x_i^{(t)}$ is non-integer. Branch into two NLP problems each with an extra constraint. Form the first NLP (call it NLP-k) by adding the constraint $x_i \leq \lfloor x_i^{(t)} \rfloor$ and form the second NLP (call it NLP-$(k+1)$) by adding the constraint $x_i \geq \lceil x_i^{(t)} \rceil$. Solve NLP-$k$ and NLP-$(k+1)$. Increment k by 2

Step 3 Fathom all NLPs one at a time, if one or more of the following conditions are satisfied:

1. The optimal solution of the NLP for all $i \in I$ is integer valued
2. The NLP problem is infeasible
3. The optimal function value of the NLP is not better than the current best function value

Step 4 If all nodes have been fathomed, **Terminate**

Else go to Step 2

The selection of the integer variable to be branched is usually guided by one of the following rules (Reklaitis, Ravindran, and Ragsdell, 1983):

1. The integer variable corresponding to the better function value is used.

2. The most important integer variable is used first (guided by the experience of the user).

3. An arbitrary integer design variable is used.

When a number of NLP problems in any level are required to be solved, there is usually no preference for solving one NLP over the other. This is because all created NLPs in any level must be solved before proceeding to the next level.

We present hand simulation of this method on a numerical problem.

Table 5.1: Intermediate Points Obtained Using the Penalty Function Method for NLP-I. (Steepest descent method is used as the unconstrained optimization technique.)

Sequence	Penalty parameter	Solution	Penalized function value
1	1.0	$(0, 0)^T$	130.000
		$(2.000, 2.580)^T$	8.494
		$(2.629, 2.095)^T$	$4.08(10^{-4})$
		$\vdots$	$\vdots$
		$(2.628, 2.091)^T$	$4.63(10^{-5})$
2	10.0	$(2.628, 2.091)^T$	$4.15(10^{-8})$

We consider the INLP problem used in the previous exercise problem and apply the branch-and-bound algorithm to solve the problem:

$$\text{Minimize} \quad f(x) = (x_1^2 + x_2 - 9)^2 + (x_1 + x_2^2 - 7)^2$$

subject to

$$\begin{aligned} g_1(x) &= 26 - (x_1 - 5)^2 + x_2^2 \geq 0, \\ g_2(x) &= 20 - 4x_1 - x_2 \geq 0, \\ x_1, x_2 &\geq 0, \qquad x_1, x_2 \text{ integers.} \end{aligned}$$

Recall that the optimal integer solution lies at the point $(3, 2)^T$ with a function value equal to 4.0.

Step 1 At first, we assume that both variables can take any real values. We solve the resulting NLP problem using the penalty function method described in the previous chapter starting from the initial point $x^{(1)} = (0, 0)^T$. We choose $R^{(0)} = 1.0$ and obtain the solution $x^{(1)} = (2.628, 2.091)^T$. The second sequence is performed with $R^{(1)} = 10.0$ and with $x^{(1)}$ as the initial point. We obtain the point $x^{(2)} = (2.628, 2.091)^T$ having an objective function value equal to $f(x^{(2)}) = 4.152 \times 10^{-7}$. Since these solutions $x^{(1)}$ and $x^{(2)}$ are the same up to three decimal places of accuracy, we terminate the search process. Intermediate points are tabulated in Table 5.1 and the points are shown on a contour plot in Figure 5.1. This solution requires 306 function evaluations.

Step 2 At this point, both solutions are real-valued. Thus, we branch along any one of the variables. Let us assume that we choose to branch on x_1 variable. Thus, we form two NLP problems as shown in Table 5.2. Let us solve the NLP-II problem first. We use the penalty function method with the steepest descent method as the unconstrained optimizer and the golden section search as the unidirectional optimizer. We begin with an initial solution $x^{(0)} = (0, 0)^T$ and $R^{(0)} = 1.0$. As shown in Table 5.3, four successive sequences of the penalty function method find the minimum point $(2.001, 2.324)^T$. Since the solution to the first variable is within the tolerance level (which is assumed to be $\epsilon_I = 0.001$), we set the solution to be $(2, 2.324)^T$. At this point, the function value is 7.322. The penalty function method requires 850 function evaluations. The intermediate points of the penalty function method are shown in Figure 5.2.

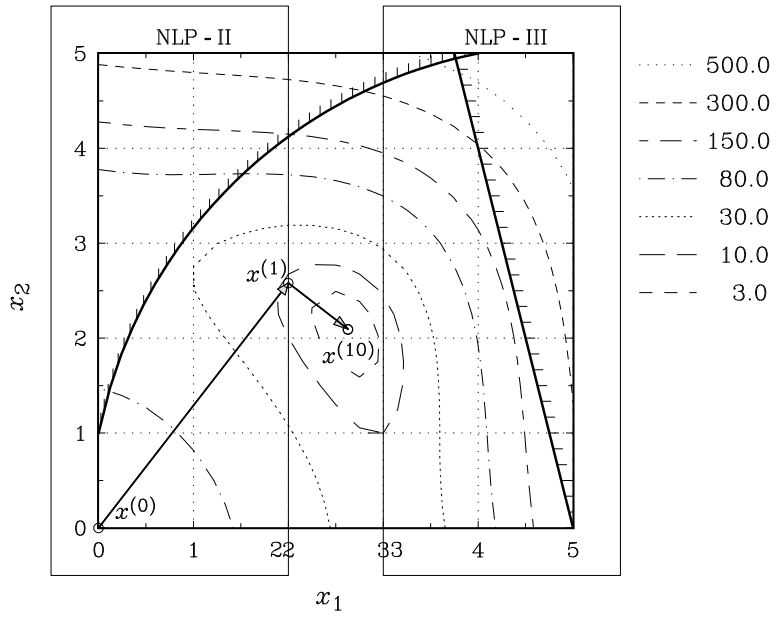


Figure 5.1: Intermediate points for NLP-I on a contour plot of the objective function $f(x)$ for real values of x_1 and x_2. Based on the obtained optimum of NLP-I, the search space is divided into two nonoverlapping regions (NLP-II and NLP-III). Note that no feasible point in NLP-I is eliminated in forming the two NLPs. The choice of division along x_1 is arbitrary.

Table 5.2: NLP-II and NLP-III Problems Formed from the Solution of NLP-I. (One extra constraint is added to the original INLP problem to form two NLP problems. Note that no feasible solution is eliminated.)

NLP-II		NLP-III	
Minimize	$f(x)$	Minimize	$f(x)$
subject to		subject to	
	$g_1(x) \geq 0,$		$g_1(x) \geq 0,$
	$g_2(x) \geq 0,$		$g_2(x) \geq 0,$
	$x_1 \leq 2.0,$		$x_1 \geq 3.0,$
	$x_1, x_2 \geq 0.$		$x_1, x_2 \geq 0.$

Table 5.3: Intermediate Points Obtained Using the Penalty Function Method for NLP-II

Sequence	Penalty parameter	Solution	Penalized function value
1	1.0	$(0, 0)^T$	130.000
		$(2.006, 2.580)^T$	8.494
		$(2.605, 2.113)^T$	0.381
		$\vdots$	$\vdots$
		$(2.603, 2.104)^T$	0.379
2	10.0	$(2.603, 2.104)^T$	3.650
		$(2.450, 2.104)^T$	2.839
		$\vdots$	$\vdots$
		$(2.427, 2.187)^T$	2.718
3	100.0	$(2.427, 2.187)^T$	19.128
		$(2.102, 2.187)^T$	6.787
		$\vdots$	$\vdots$
		$(2.091, 2.324)^T$	6.377
4	1000.0	$(2.001, 2.324)^T$	7.311

After the solution of NLP-II, we solve the NLP-III problem from the same starting point and penalty parameters. This time, we obtain the solution $(2.999, 1.925)^T$ (refer to Table 5.4). This solution can be accepted to be the solution $(3, 1.925)^T$ with the specified tolerance level. At this point, the function value is 3.792. The number of function evaluations required to solve this problem is 952. Figure 5.2 shows how the penalty function method finds this minimum point.

Step 3 Since none of the problems NLP-II and NLP-III satisfy any of the three conditions for fathoming a node, we proceed to Step 4.

Step 4 Since NLP-II and NLP-III are not yet fathomed, we move to Step 2. This completes one iteration of the branch-and-bound method.

Step 2 From each node (NLP-II and NLP-III), we now have to branch into two more nodes. This makes a total of four NLP problems. Recalling that the solution obtained from NLP-II is $(2, 2.324)^T$, we branch only on the second variable. Two problems arising from NLP-II are tabulated in Table 5.5. Using the penalty function method and starting from the point $(0, 0)^T$, we solve NLP-IV and obtain the solution $(2.001, 2.000)^T$. We approximate this solution to be $(2, 2)^T$ with a function value equal to 10.0. This procedure requires 510 function evaluations.

Solving NLP-V, we obtain the solution $(1.999, 3.000)^T$. This solution can be approximated as $(2, 3)^T$ with a function value equal to 20.0. The number of function evaluations required to solve NLP-V is 612.

In NLP-III, the solution obtained is $(3, 1.925)^T$. The progress of this simulation is not shown in Figure 5.2, for the sake of brevity. We branch on the second variable only. Two NLP problems arising from NLP-III are shown in Table 5.6.

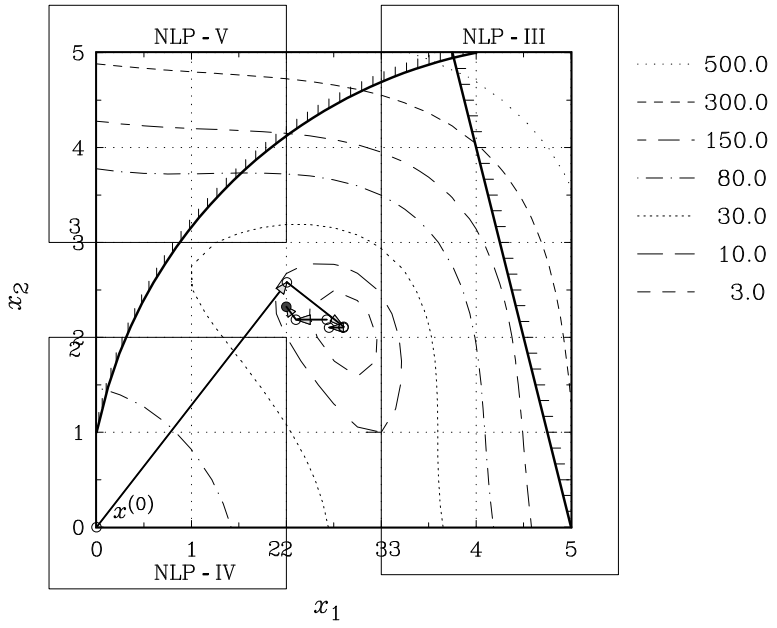


Figure 5.2: Intermediate points for NLP-II and NLP-III on a contour plot of the objective function $f(x)$ for real values of x_1 and x_2. NLP-II is branched into NLP-IV and NLP-V based on the obtained solution of NLP-II.

Solutions to these two problems are also found using the penalty function method from an initial point $(0, 0)^T$. The solution for NLP-VI is $(3.000, 1.000)^T$ with a function value equal to 10.0 and the solution for NLP-VII is found to be $(3.000, 2.001)^T$. This solution is approximated to be $(3, 2)^T$ with a function value equal to 4.0. The function evaluations required to solve these NLPs are 544 and 754, respectively.

Step 3 Since the solution to NLP-IV is a feasible solution (having integer values for both variables), we fathom this node. Similarly, we fathom the node at NLP-V. The current best solution for problems NLP-IV and NLP-V is $(2, 2)^T$ with a function value equal to 10.0. The intermediate points of the simulation are shown in Figure 5.3.

The solutions to both NLP-VI and NLP-VII are also feasible (integer valued). Thus, we fathom each of these NLPs. The current best solution at this stage is found as $(3, 2)^T$ with a function value equal to 4.0. Two NLPs and their solutions using the penalty function method are shown in Figure 5.3. We now move to Step 4.

Step 4 Since all open nodes are now fathomed, we terminate the algorithm. Therefore, the solution to the original INLP problem is $(3, 2)^T$ with optimal function value equal to 4.0.

The flowchart of the progress of the algorithm is outlined in Figure 5.4. The figure shows how an NLP is branched into two NLPs with an extra constraint and how an NLP is fathomed. The complete solution of the INLP problem requires a total of $4,528$ function evaluations; whereas there are only 22 feasible points in the specified search space (Figure 5.2). This computational complexity of this algorithm increases as the number of integer variables increases. Due to this reason, it is advisable not to use this algorithm in problems where the number of integer variables exceeds ten or so.

Table 5.4: Intermediate Points Obtained Using the Penalty Function Method for NLP-III. (The steepest descent method is used as the unconstrained optimization technique.)

Sequence number	Penalty parameter	Solution	Penalized function value
1	1.0	$(0, 0)^T$	139.000
		$(2.510, 2.559)^T$	0.810
		$(2.653, 2.100)^T$	0.143
		$\vdots$	$\vdots$
		$(2.643, 2.083)^T$	0.132
2	10.0	$(2.643, 2.083)^T$	1.279
		$(2.723, 2.083)^T$	1.019
		$\vdots$	$\vdots$
		$(2.734, 2.035)^T$	0.983
3	100.0	$(2.734, 2.035)^T$	7.351
		$(2.911, 2.035)^T$	3.071
		$\vdots$	$\vdots$
		$(2.919, 1.925)^T$	2.887
4	1000.0	$(2.919, 1.925)^T$	8.791
		$(2.989, 1.925)^T$	3.670
		$\vdots$	$\vdots$
		$(2.999, 1.925)^T$	3.791

Table 5.5: NLP-IV and NLP-V Problems Formed from the Solution of NLP-II. (One extra constraint is added to NLP-II to form two NLP problems.)

NLP-IV		NLP-V	
Minimize	$f(x)$	Minimize	$f(x)$
subject to		subject to	
	$g_1(x) \geq 0,$		$g_1(x) \geq 0,$
	$g_2(x) \geq 0,$		$g_2(x) \geq 0,$
	$x_1 \leq 2.0,$		$x_1 \leq 2.0,$
	$x_2 \leq 2.0,$		$x_2 \geq 3.0,$
	$x_1, x_2 \geq 0.$		$x_1, x_2 \geq 0.$

Table 5.6: NLP-VI and NLP-VII Problems Formed from the Solution of NLP-III. (One extra constraint is added to NLP-III to form to NLP problems.)

NLP-VI		NLP-VII	
Minimize	$f(x)$	Minimize	$f(x)$
subject to		subject to	
	$g_1(x) \geq 0,$		$g_1(x) \geq 0,$
	$g_2(x) \geq 0,$		$g_2(x) \geq 0,$
	$x_1 \geq 3.0,$		$x_1 \geq 3.0,$
	$x_2 \leq 1.0,$		$x_2 \geq 2.0,$
	$x_1, x_2 \geq 0.$		$x_1, x_2 \geq 0.$

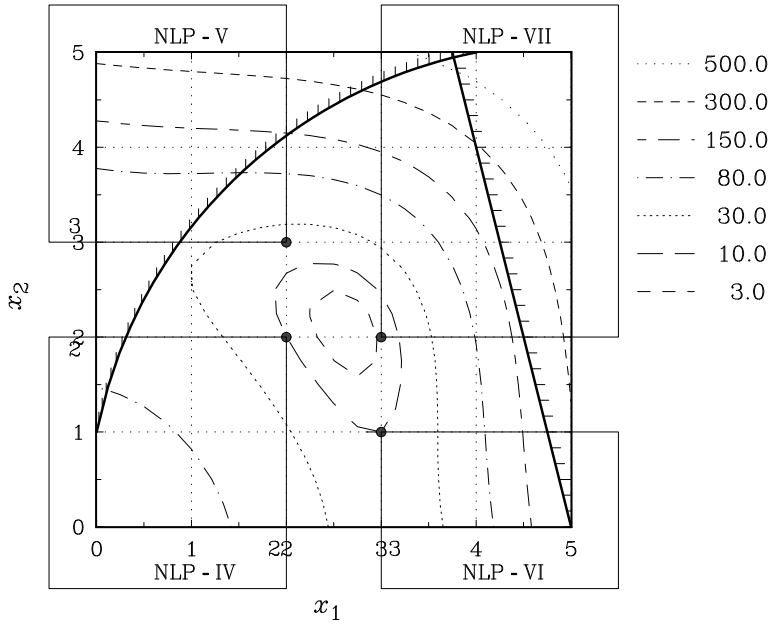


Figure 5.3: The optimum points for NLP-IV, NLP-V, NLP-VI, and NLP-VII on a contour plot of the objective function $f(x)$ for real values of x_1 and x_2. No feasible solution in NLP-IV is eliminated.

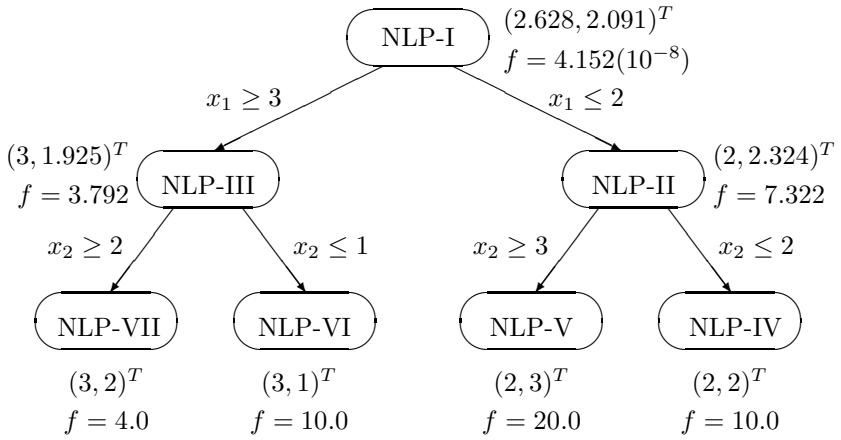


Figure 5.4: Different NLPs in the branch-and-bound method.

Chapter 6

Nontraditional Optimization Techniques

This chapter makes a brief description of a nontraditional search and optimization methods. Further details will be supplied by separate sources.

6.1 Genetic Algorithms

Genetic algorithms (abbreviated as GAs) are computerized search and optimization algorithms designed based on the mechanics of natural genetics and natural selection.

6.1.1 Fundamental Differences with Traditional methods

1. GAs work on a coding of parameters, instead of parameters. GAs exploit the coding similarities to achieve a parallel search.
2. GAs work on a population of points, instead of a single point. That is why GAs are likely to find the global solutions.
3. GAs do not require any derivative or auxiliary information. This extends the application of GAs to a wide variety of problem domains. That is why GAs are robust.
4. GAs use probabilistic transition rules, instead of deterministic transition rules. This reduces the bias in the search. Initially the search direction is random and as iteration progresses, GAs obtain a directed search adaptively.

Algorithm:

Step 1 Choose a coding to represent problem parameters, a selection operator, a crossover operator, and a mutation operator. Choose population size, N, crossover probability, p_c, mutation probability, p_m. Initialize a random population of strings of size N. Set $t = 0$

Step 2 Evaluate each string in the population

Step 3 If $t > t_{\max}$ or other termination criteria is satisfied,
Terminate
Else Go to step 4

Step 4 Reproduction on the population

Step 5 Crossover on random pairs of strings

Step 6 Mutation on every string

Step 7 Set $t = t + 1$ and Go to step 2

6.1.2 Reproduction Operator

Selects good strings from a population. Some of popular reproduction operators are as follows:

Proportionate Selection Strings are selected according to their fitness. Specifically, a string with fitness f_i is allocated $\frac{f_i}{f_{avg}}$ number of copies. Better strings are allocated more copies under this scheme.

Tournament Selection Usually, s strings are selected at random from a population, and the best is chosen. This procedure is continued until the whole population is filled up. This scheme can be performed with and without replacement. When performed without replacement, the best string gets s copies.

Ranking Selection All strings are first ranked from best to worst and ranked in a way so that the best gets s copies and the worst gets zero copies. Each string is then selected with a probability depending on its rank in the population.

6.1.3 Crossover Operator

Exchanges information between two strings selected at random. Some popular operators are as follows:

Single-point Crossover A cross site is chosen at random. The contents on one side of the site are exchanged between parent strings:

$$\begin{array}{cc|ccc} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{array} \Rightarrow \begin{array}{cc|ccc} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{array}$$

Two-point Crossover Two cross sites are chosen at random. The contents between two sites are exchanged between parent strings:

$$\begin{array}{c|cc|cc} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{array} \Rightarrow \begin{array}{c|cc|cc} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \end{array}$$

Uniform Crossover Each bit-position is exchanged between parent strings with a probability 0.5:

```
1 1 1 1 1        1 0 0 1 0
         ⇒
0 0 0 0 0        0 1 1 0 1
```

The search power of uniform crossover is most among these three operators, but the destruction probability of good partial substrings in parent strings is also more in uniform crossover.

In order to preserve some previously obtained good solutions, all strings in the population is not used in crossover. The proportion of population used in crossover is known as probability of crossover. Usually, a proportion of 0.60–0.95 is used.

6.1.4 Mutation Operator

Alters a bit value to another with a small probability. In a binary string, mutation operation is shown below:

$$\texttt{11111} \Rightarrow \texttt{11101}$$

Mutation maintains diversity in the population.

• Other advanced operators exist and tried

The objective is to minimize the function

$$f(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$$

in the interval $0 \leq x_1, x_2 \leq 6$. Recall that the true solution to this problem is $(3, 2)^T$ having a function value equal to zero.

Step 1 In order to solve this problem using genetic algorithms, we choose binary coding to represent variables x_1 and x_2. In the calculations here, 10-bits are chosen for each variable, thereby making the total string length equal to 20. With 10 bits, we can get a solution accuracy of $(6-0)/(2^{10}-1)$ or 0.006 in the interval $(0, 6)$. We choose roulette-wheel selection, a single-point crossover, and a bit-wise mutation operator. The crossover and mutation probabilities are assigned to be 0.8 and 0.05, respectively. We decide to have 20 points in the population. The random population created using Knuth's (1981) random number generator[1] with a random seed equal to 0.760 is shown in Table 6.1. We set $t_{\max} = 30$ and initialize the generation counter $t = 0$.

Step 2 The next step is to evaluate each string in the population. We calculate the fitness of the first string. The first substring (`1100100000`) decodes to a value equal to $(2^9+2^8+2^5)$ or 800. Thus, the corresponding parameter value is equal to $0 + (6-0) \times 800/1023$ or 4.692. The second substring (`1110010000`) decodes to a value equal to $(2^9+2^8+2^7+2^4)$ or 912. Thus, the corresponding parameter value is equal to $0 + (6-0) \times 912/1023$ or 5.349. Thus, the first string corresponds to the point $x^{(1)} = (4.692, 5.349)^T$. These values can now be substituted in the objective function expression to obtain the function value. It is found that the function value at this point is equal to $f(x^{(1)}) = 959.680$. We now calculate the fitness function value at this point using the transformation rule: $\mathcal{F}(x^{(1)}) = 1.0/(1.0+959.680) = 0.001$. This value is used in the reproduction operation. Similarly, other strings in the population are evaluated and fitness values are calculated. Table 6.1 shows the objective function value and the fitness value for all 20 strings in the initial population.

Step 3 Since $t = 0 < t_{\max} = 30$, we proceed to Step 4.

[1] A FORTRAN code implementing the random number generator appears in the GA code presented at the end of this chapter.

Table 6.1: Evaluation and reproduction of a random population is illustrated.

	String		x_2	x_1	$f(x)$	$\mathcal{F}(x)$	Expected	Probability	Cumulative	Random	String	Actual	Mating
	Substring-2	Substring-1					Count	of copying	probability	number	number	count	pool
1	1110010000	1100100000	5.349	4.692	959.680	0.001	0.13	0.007	0.007	0.472	10	0	0010100100 1010101010
2	0001001101	0011100111	0.452	1.355	105.520	0.009	1.10	0.055	0.062	0.108	3	1	1010100001 0111001000
3	1010100001	0111001000	3.947	2.674	126.685	0.008	0.98	0.049	0.111	0.045	2	1	0001001101 0011100111
4	1001000110	1000010100	3.413	3.120	65.026	0.015	1.85	0.093	0.204	0.723	14	2	1110011011 0111000010
5	1100011000	1011100011	4.645	4.334	512.197	0.002	0.25	0.013	0.217	0.536	10	0	0010100100 1010101010
6	0011100101	0011111000	1.343	1.455	70.868	0.014	1.71	0.086	0.303	0.931	19	2	0011100010 1011000011
7	0101011011	0000000111	2.035	0.041	88.273	0.011	1.34	0.067	0.370	0.972	19	1	0011100010 1011000011
8	1110101000	1110101011	5.490	5.507	1436.563	0.001	0.12	0.006	0.376	0.817	17	0	0111000010 1011000110
9	1001111101	1011100111	3.736	4.358	265.556	0.004	0.49	0.025	0.401	0.363	7	1	0101011011 0000000111
10	0010100100	1010101010	0.962	4.000	39.849	0.024	2.96	0.148	0.549	0.189	4	3	1001000110 1000010100
11	1111101001	0001110100	5.871	0.680	814.117	0.001	0.14	0.007	0.556	0.220	6	0	0011100101 0011111000
12	0000111101	0110011101	0.358	2.422	42.598	0.023	2.84	0.142	0.698	0.288	6	3	0011100101 0011111000
13	0000111110	1110001101	0.364	5.331	318.746	0.003	0.36	0.018	0.716	0.615	12	1	0000111101 0110011101
14	1110011011	0111000010	5.413	2.639	624.164	0.002	0.24	0.012	0.728	0.712	13	1	0000111110 1110001101
15	1010111010	1010111000	4.094	4.082	286.800	0.003	0.37	0.019	0.747	0.607	12	0	0000111101 0110011101
16	0100011111	1100111000	1.683	4.833	197.556	0.005	0.61	0.030	0.777	0.192	4	0	1001000110 1000010100
17	0111000010	1011000110	2.639	4.164	97.699	0.010	1.22	0.060	0.837	0.386	9	1	1001111101 1011100111
18	1010010100	0100001001	3.871	1.554	113.201	0.009	1.09	0.054	0.891	0.872	18	1	1010010100 0100001001
19	0011100010	1011000011	1.326	4.147	57.753	0.017	2.08	0.103	0.994	0.589	12	2	0000111101 0110011101
20	1011100011	1111010000	4.334	5.724	987.955	0.001	0.13	0.006	1.000	0.413	10	0	0010100100 1010101010

Step 4 At this step, we select good strings in the population to form the mating pool. In order to use the roulette-wheel selection procedure, we first calculate the average fitness of the population. By adding the fitness values of all strings and dividing the sum by the population size, we obtain $\overline{\mathcal{F}} = 0.008$. The next step is to compute the expected count of each string as $\mathcal{F}(x)/\overline{\mathcal{F}}$. The values are calculated and shown in column A of Table 6.1. In other words, we can compute the probability of each string being copied in the mating pool by dividing these numbers with the population size (column B). Once these probabilities are calculated, the cumulative probability can also be computed. These distributions are also shown in column C of Table 6.1. In order to form the mating pool, we create random numbers between zero and one (given in column D) and identify the particular string which is specified by each of these random numbers. For example, if the random number 0.472 is created, the tenth string gets a copy in the mating pool, because that string occupies the interval $(0.401, 0.549)$, as shown in column C. Column E refers to the selected string. Similarly, other strings are selected according to the random numbers shown in column D. After this selection procedure is repeated n times (n is the population size), the number of selected copies for each string is counted. This number is shown in column F. The complete mating pool is also shown in the table. Columns A and F reveal that the theoretical expected count and the true count of each string more or less agree with each other. Figure 6.1 shows the initial random population and the mating pool after reproduction. The points marked with an enclosed box are the points

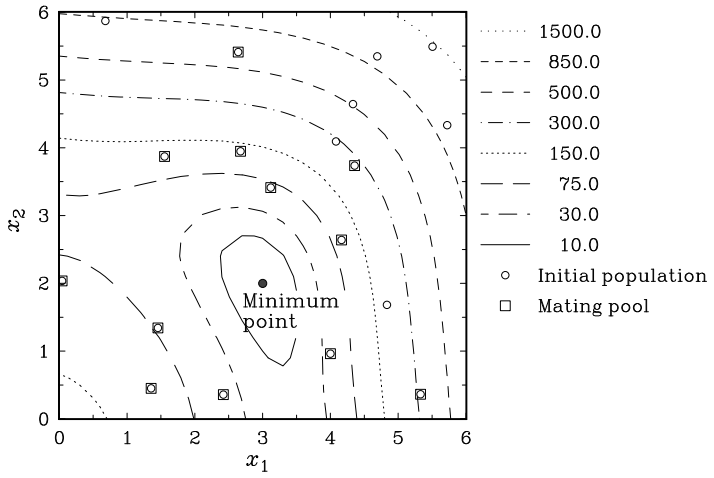


Figure 6.1: The initial population (marked with empty circles) and the mating pool (marked with boxes) on a contour plot of the objective function. The best point in the population has a function value 39.849 and the average function value of the initial population is 360.540.

in the mating pool. The action of the reproduction operator is clear from this plot. The inferior points have been probabilistically eliminated from further consideration. Notice that not all selected points are better than all rejected points. For example, the 14th individual (with a fitness value 0.002) is selected but the 16th individual (with a function value 0.005) is not selected.

Although the above roulette-wheel selection is easier to implement, it is noisy. A more stable version of this selection operator is sometimes used. After the expected count for each individual string is calculated, the strings are first assigned copies exactly equal to the mantissa of the expected count. Thereafter, the regular roulette-wheel selection is implemented using the decimal part of the expected count as the probability of selection.

This selection method is less noisy and is known as the *stochastic remainder* selection.

Step 5 At this step, the strings in the mating pool are used in the crossover operation. In a single-point crossover, two strings are selected at random and crossed at a random site. Since the mating pool contains strings at random, we pick pairs of strings from the top of the list. Thus, strings 3 and 10 participate in the first crossover operation. When two strings are chosen for crossover, first a coin is flipped with a probability $p_c = 0.8$ to check whether a crossover is desired or not. If the outcome of the coin-flipping is true, the crossing over is performed, otherwise the strings are directly placed in an intermediate population for subsequent genetic operation. It turns out that the outcome of the first coin-flipping is true, meaning that a crossover is required to be performed. The next step is to find a cross-site at random. We choose a site by creating a random number between $(0, \ell - 1)$ or (0, 19). It turns out that the obtained random number is 11. Thus, we cross the strings at the site 11 and create two new strings. After crossover, the children strings are placed in the intermediate population. Then, strings 14 and 2 (selected at random) are used in the crossover operation. This time the coin-flipping comes true again and we perform the crossover at the site 8 found at random. The new children strings are put into the intermediate population. Figure 6.2 shows how points cross over and form new points. The points marked with a small box are the points in the mating pool and the

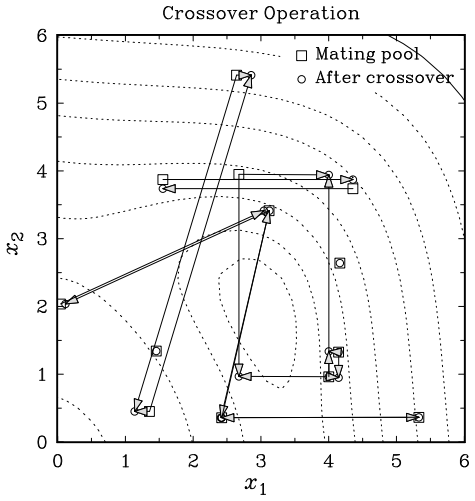


Figure 6.2: The population after the crossover operation. Two points are crossed over to form two new points. Of ten pairs of strings, seven pairs are crossed.

points marked with a small circle are children points created after crossover operation. Notice that not all 10 pairs of points in the mating pool cross with each other. With the flipping of a coin with a probability $p_c = 0.8$, it turns out that fourth, seventh, and tenth crossovers come out to be false. Thus, in these cases, the strings are copied directly into the intermediate population. The complete population at the end of the crossover operation is shown in Table 6.2. It is interesting to note that with $p_c = 0.8$, the expected number of crossover in a population of size 20 is $0.8 \times 20/2$ or 8. In this exercise problem, we performed seven crossovers and in three cases we simply copied the strings to the intermediate population. Figure 6.2 shows that some good points and some not-so-good points are created after crossover. In some cases, points far away from the parent points are created and in some cases points close to the parent points are created.

Step 6 The next step is to perform mutation on strings in the intermediate population.

Table 6.2: Crossover and mutation operators are shown.

Mating pool		Cross-over	Site	Intermediate population		Mutation		x_1	x_2	$f(x)$	$\mathcal{F}x$
Substring-2	Substring-1			Substring-2	Substring-1	Substring-2	Substring-1				
0010100100	1010101010	yes	9	0010100101	0111001000	0010101**1**101	0111001000	1.015	2.674	18.886	0.050
1010100001	0111001000	yes	9	1010100000	1010101010	1010100001	1010101010	3.947	4.000	238.322	0.004
0001001101	0011100111	yes	12	0001001101	0011000010	0001001101	00**0**1000010	0.452	0.387	149.204	0.007
1110011011	0111000010	yes	12	1110011011	0111100111	1110011011	01**0**1100**0**11	5.413	2.082	596.340	0.002
0010100100	1010101010	yes	5	0010100010	1011000011	0010100010	1011000011	0.950	4.147	54.851	0.018
0011100010	1011000011	yes	5	0011100100	1010101010	0011100100	1**1**10101010	1.337	5.501	424.583	0.002
0011100010	1011000011	no		0011100010	1011000011	0011100011	1011**1**00011	1.331	4.334	83.929	0.012
0111000010	1011000110	no		0111000010	1011000110	01**0**101**0**010	1011000110	1.982	4.164	70.472	0.014
0101011011	0000000111	yes	14	0101011011	0000010100	0101011011	0000010100	2.035	0.117	87.633	0.011
1001000110	1000010100	yes	14	1001000110	1000000111	100101**1**0110	1000000111	3.507	3.044	72.789	0.014
0011100101	0011111000	yes	1	0011100101	0011111000	0011100101	0011111000	1.343	1.455	70.868	0.014
0011100101	0011111000	yes	1	0011100101	0011111000	0011100101	0011111000	1.343	1.455	70.868	0.014
0000111101	0110011101	no		0000111101	0110011101	00001**0**1101	011**1**011100	0.264	2.792	25.783	0.037
0000111110	1110001101	no		0000111110	1110001101	0000111110	1110001101	0.364	5.331	318.746	0.003
0000111101	0110011101	yes	18	0000111101	0110011100	0000111101	0110011100	0.358	2.416	42.922	0.023
1001000110	1000010100	yes	18	1001000110	1000010101	1001000110	**0**000010101	3.413	0.123	80.127	0.012
1001111101	1011100111	yes	10	1001111101	0100001001	1001111101	0100001001	3.736	1.554	95.968	0.010
1010010100	0100001001	yes	10	1010010100	1011100111	1010010100	101**0**100111	3.871	3.982	219.426	0.005
0000111101	0110011101	no		0000111101	0110011101	0000111101	0110011101	0.358	2.422	42.598	0.023
0010100100	1010101010	no		0010100100	1010101010	0010100100	1010101010	0.962	4.000	39.849	0.024

For bit-wise mutation, we flip a coin with a probability $p_m = 0.05$ for every bit. If the outcome is true, we alter the bit to `1` or `0` depending on the bit value. With a probability of 0.05, a population size 20, and a string length 20, we can expect to alter a total of about $0.05 \times 20 \times 20$ or 20 bits in the population. Table 6.2 shows the mutated bits in bold characters in the table. As counted from the table, we have actually altered 16 bits. Figure 6.3 shows the effect of mutation on the intermediate population. In some cases, the mutation operator changes a point locally and in some other it can bring a large change. The points marked with a small circle are points in the intermediate population. The points marked with a small box constitute the new population (obtained after reproduction, crossover, and mutation). It is interesting to note that if only one bit is mutated in a string, the point is moved along a particular variable only. Like the crossover operator, the mutation operator has created some points better and some points worse than the original points. This flexibility enables GA operators to explore the search space properly before converging to a region prematurely. Although this requires some extra computation, this flexibility is essential to solve global optimization problems.

Step 7 The resulting population becomes the new population. We now evaluate each string as before by first identifying the substrings for each variable and mapping the decoded values of the substrings in the chosen intervals. This completes one iteration of genetic algorithms. We increment the generation counter to $t = 1$ and proceed to Step 3 for the next iteration. The new population after one iteration of GAs is shown in Figure 6.3 (marked with empty boxes). The figure shows that in one iteration, some good points have been found. Table 6.2 also shows the fitness values and objective function values of the new population members.

The average fitness of the new population is calculated to be 0.015, a remarkable improvement from that in the initial population (recall that the average in the initial population was

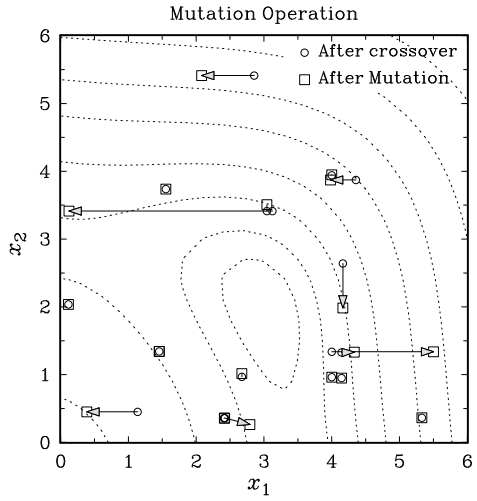


Figure 6.3: The population after mutation operation. Some points do not get mutated and remain unaltered. The best point in the population has a function value 18.886 and the average function value of the population is 140.210, an improvement of over 60 per cent.

0.008). The best point in this population is found to have a fitness equal to 0.050, which is also better than that in the initial population (0.024). This process continues until the maximum allowable generation is reached or some other termination criterion is met. The population after 25 generation is shown in Figure 6.4. At this generation, the best point is found to be

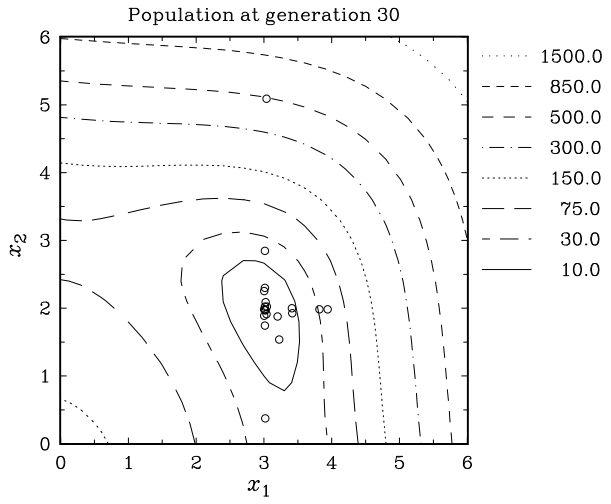


Figure 6.4: All 20 points in the population at generation 25 shown on the contour plot of the objective function. The figure shows that most points are clustered around the true minimum.

$(3.003, 1.994)^T$ with a function value 0.001. The fitness value at this point is equal to 0.999 and the average population fitness of the population is 0.474. The figure shows how points are clustered around the true minimum of the function in this generation. A few inferior points are still found in the plot. They are the result of some unsuccessful crossover events. We also observe that the total number of function evaluations required to obtain this solution is $0.8 \times 20 \times 26$ or 416 (including the evaluations of the initial population).

In order to show the efficiency of GAs in arriving at a point close to the true optimum, we perform two more simulations starting with different initial populations. Figure 6.5 shows how the function value of the best point in a population reduces with generation number. Although all three runs have a different initial best point, they quickly converge to a solution

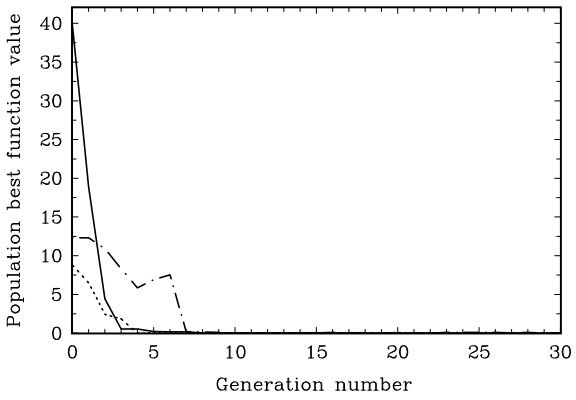


Figure 6.5: The function value of the best point in the population for three independent GA runs. All runs quickly converge to a point near the true optimum.

close to the true optimum (recall that the optimum point has a function value equal to zero).

In order to illustrate the schema processing through genetic operators, we investigate the growth of a particular schema $H = (0 * \ldots * \ * \ldots *)$. This schema represents all points in the range $0 \leq x_2 < 3$. The optimum point lies in this region. With reference to Equation (6.3), we observe that the order, defining length, and the fitness of the schema are such that it is a building block. This schema contains more good points than it's competitor $H^c = (1 * \ldots * \ * \ldots *)$ which represents the range $3 \leq x_2 \leq 6$. According to Equation (6.3), the schema H must increase exponentially due to the action of genetic operators. We observe that in the random initial population the schema H has nine strings and the schema H^c has 11 strings. At the end of one generation, the population has 14 strings representing the schema H and only six strings representing the schema H^c. We may also investigate other interesting regions in the search space and observe their growth in terms of the number of representative points in the population. Other low-order and above-average schemata are also processed similarly and are combined to form higher-order and good schemata. This processing of several schemata happens in parallel without any extra book-keeping (Goldberg, 1989). Eventually, this processing forms the optimum or a near-optimum point.

Suggested Reading on GAs:

1. Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning.* Reading: Addison-Wesley.

2. Holland, J. H. (1975). *Adaptation in natural and artificial systems.* Ann Arbor: University of Michigan Press

3. Proceedings of International Conference on Genetic Algorithms and Their Applications (1985, 1987) Lawrence Elbaum (1987, 1989, 1991, 1993) San Mateo: Morgan Kaufmann

4. Proceedings of Foundation of Genetic Algorithms (1991, 1993) San Mateo: Morgan Kaufmann

Appendix A

An Introduction to Linear Programming

Linear programming methods can solve problems having linear objective function and linear constraints. Since all constraints are linear, the feasible region is usually an enclosed region surrounded by linear hyperplanes. Since the objective function is also linear, the optimum point is usually one of the corner points (or hyperplanes) of the feasible region. Here we consider a maximization problem and discuss two ways to handle minimization problems later. Let us consider an N-variable problem having J equality constraints suitable for an LP method:

$$\left.\begin{aligned} \text{Maximize} \quad & f(x) = c_1x_1 + c_2x_2 + \ldots + c_Nx_N \\ \text{subject to} \quad & \\ & a_{11}x_1 + a_{12}x_2 + \cdots + a_{1N}x_N = b_1, \\ & a_{21}x_1 + a_{22}x_2 + \cdots + a_{2N}x_N = b_2, \\ & \vdots \qquad\qquad\qquad\qquad \vdots \\ & a_{J1}x_1 + a_{J2}x_2 + \cdots + a_{JN}x_N = b_J, \\ & x_i \geq 0, \qquad i = 1, 2, \ldots, N. \end{aligned}\right\} \tag{A.1}$$

Even though all constraints in the above problem are shown to be equality constraints, inequality constraints can also be handled using the linear programming method. An inequality constraint is usually transformed into an equivalent equality constraint by introducing a slack variable. If the inequality constraint is of lesser-than-equal-to type, a slack variable is added to the left-side expression. For example, the constraint $g_j(x) \leq 0$ can be converted into an equality constraint by adding a slack variable x_{N+j} as follows:

$$g_j(x) + x_{N+j} = 0,$$

where $x_{N+j} \geq 0$. The idea is that since the quantity $g_j(x)$ is less than or equal to zero, we add a positive quantity to make the sum zero. On the other hand, if the constraint is of greater-than-equal-to type, a slack variable is subtracted from the left side expression. The constraint $g_j(x) \geq 0$ is converted to

$$g_j(x) - x_{N+j} = 0,$$

where $x_{N+j} \geq 0$. The final set of constraints in Equation (A.1) requires all variables to take nonnegative values. If any variable is expected to have a negative value, that variable can

be substituted by another variable (which can take only nonnegative values) using a suitable linear transformation. In the above transformations of inequality to equality constraints, the slack variables are fixed to take nonnegative values. Thus, any linear problem having inequality or equality constraints can be rewritten in the form given in problem (A.1).

In general, in a real-world optimization problem, there are more variables than the number of constraints. In those situations, equality constraints can be used to express J variables in terms of other $(N - J)$ variables. The equality constraints can be written in the *row-echelon* form as follows (Strang, 1980):

$$\left.\begin{array}{ccccccccccc} x_1 & & & + & a'_{1(J+1)}x_{J+1} & + & \cdots & + & a'_{1N}x_N & = & b'_1, \\ & x_2 & & + & a'_{2(J+1)}x_{J+1} & + & \cdots & + & a'_{2N}x_N & = & b'_2, \\ & & \ddots & & \vdots & & \vdots & & \vdots & & \vdots \\ & & & x_J & + & a'_{J(J+1)}x_{J+1} & + & \cdots & + & a'_{JN}x_N & = & b'_J. \end{array}\right\} \quad \text{(A.2)}$$

The variables x_1 to x_J are expressed in terms of other variables (x_{J+1} to x_N) and are known as *basic* variables. The variables x_{J+1} to x_N can take any values and are known as *nonbasic* variables. It is important to note that even though the above equations show that the first J variables (x_1 to x_J) are basic variables, in practice, any J variables can be considered as basic variables. In fact, basic variables are chosen using some guidelines, which we shall discuss later. In the above row-echelon formulation, the coefficients corresponding to the basic variables are always one and b'_j values are always positive. For a set of values of the nonbasic variables, a feasible solution to the above problem can be obtained by computing other J variables. Therefore, there are infinite feasible solutions to the above problem satisfying all the above equations. Since the objective in problem (A.1) is to maximize the function $f(x)$, the feasible solution(s) corresponding to the maximum function value is(are) the optimum solution(s) to the problem.

There are two different ways one can get a feasible solution of the original problem. One way to achieve this is to choose the set of basic variables a priori, assign several sets of values to nonbasic variables, and then obtain corresponding basic variables in each case using (A.2). The other way is to choose several sets of basic variables, assign all nonbasic variables to zero, and then compute the values of the basic variables in each case using (A.2). In the method described in the following section, the latter approach is used. In this approach, any combination of J variables can be chosen to create the row-echelon formulation. Since there are N variables from which to choose J variables, there are a total of $\binom{N}{J}$ different solutions to compare[1]. A basic feasible solution is defined as a solution at which all basic variables are nonnegative and all nonbasic variables are zero. Thus, there are a total of $\binom{N}{J}$ basic feasible solutions possible, of which one or more corresponds to the optimum point. A naive approach would be to compare all such basic feasible solutions and obtain the optimum. An efficient approach would be to compare only a small fraction of all basic feasible solutions to obtain the optimum. In the following section, we discuss one such efficient algorithm.

A.1 Simplex Method

The simplex method was developed by G. B. Dantzig in 1963. The main concept of the simplex method is to compare *neighbouring* basic feasible solutions in an efficient way. A neighbouring solution is a basic feasible solution which differs from the current basic feasible solution in

[1]The combination $\binom{N}{J}$ is calculated by computing the factorials of N, J, and $(N-J)$ as follows: $\binom{N}{J} = N!/(J!(N-J)!)$, where $N! = N(N-1)(N-2)\cdots 2 \times 1$.

exactly one basic variable. That is, one basic variable in the current basic feasible solution is made nonbasic and a currently nonbasic variable is made basic. This can be achieved in a total of $J(N-J)$ different ways. Therefore, there are two decisions to be made:

1. Which one of the $(N-J)$ nonbasic variables is to be chosen for the basic variable?
2. Which one of the J basic variables is to be chosen for the nonbasic variable?

We answer these questions now.

It is clear that the choice of different nonbasic variables will result in different objective function values. The simplex method chooses that nonbasic variable which causes a maximum increase in the objective function value. By simple algebraic calculations, it can be shown that the increase in the objective function value $f(x)$ due to a unit increase in a nonbasic variable x_q from zero to one is given by

$$(\Delta f)_q = c_q - \sum_{j=1}^{J} c_j a'_{jq}.$$

Thus, we can calculate $(\Delta f)_q$ for all nonbasic variables and choose the one with the maximum positive value. In the case of a tie (same $(\Delta f)_q$ for more than one nonbasic variables), any nonbasic variable can be chosen at random. It is worth mentioning here that if for all remaining nonbasic variables, the quantity $(\Delta f)_q$ is nonpositive, no increment in the objective function value is possible. This suggests that the optimum solution has been obtained, and we terminate the algorithm.

Once a nonbasic variable (x_q) is chosen, the next question is, which of the basic variables has to be made nonbasic. It is clear that as x_q is increased from zero to one, the objective function value will also increase. Therefore, we may want to increase the nonbasic variable x_q indefinitely. But there is a limit to the extent of this increment. When x_q is increased, all basic variables must either be increased, decreased, or kept the same in order to make the solution feasible. Recall that all variables in a linear program must be nonnegative. Thus, the critical basic variable is the one which, when reduced, becomes zero first. Any more increase in the chosen nonbasic variable will make that basic variable negative. From the row-echelon formulation, we can write the value of a basic variable as follows:

$$x_j = b'_j - a'_{jq} x_q, \qquad j = 1, 2, \ldots, J.$$

A basic variable becomes zero when $x_q = b'_j / a'_{jq}$. Since, in the row-echelon form, all b'_j are nonnegative, this can happen only when a'_{jq} is positive. In order to find the critical basic variable, we compute the quantity b'_j / a'_{jq} and choose the basic variable x_q for which this quantity is minimum. This rule is also known as *minimum ratio* rule in linear programming.

The simplex method begins with an initial feasible solution. Thereafter, a basic variable is replaced by a nonbasic variable chosen according to rules described above. Thus, the simplex method is an iterative method which works by alternating among various basic and nonbasic variables so as to achieve the optimum point efficiently. Since all constraints and the objective function are linear, these points are the corners of the feasible search region. In the following, we describe the algorithm:

Algorithm

Step 1 Choose a basic feasible solution. Set all nonbasic variables to zero.

Step 2 Calculate the quantity $(\Delta f)_q$ for all nonbasic variables and choose the one having the maximum value. If $(\Delta f)_q \leq 0$ for all nonbasic variables, **Terminate**;

Else use the minimum ratio rule to choose the basic variable to be replaced.

Step 3 Perform a row-echelon formulation for new basic variables and go to Step 2.

The algorithm assumes that the underlying problem can be written in the form shown in Equation (A.1). If some variables in the original problem take negative values, they need to be suitably transformed so that the new variables can only take nonnegative values. Inequality constraints are converted into equality constraints by adding or subtracting slack variables. The algorithm stated above works only for maximization problems. There are two ways the above algorithm can be used to solve minimization problems. The duality principle (by multiplying the objective function by -1) can be used to convert the minimization problem into an equivalent maximization problem. In the other approach, the criterion for choosing a nonbasic variable to make it basic needs to be changed. Recall that in the maximization problem, the nonbasic variable for which the quantity $(\Delta f)_q$ is maximally nonnegative is chosen. That variable increases the objective function at the highest rate. Similarly, the nonbasic variable for which the quantity $(\Delta f)_q$ is maximally nonpositive will decrease the objective function value at the highest rate. Thus, the above algorithm can be used for the minimization problems by selecting nonbasic variables according to the most negative $(\Delta f)_q$. The rest of the algorithm can be used as above. The other change is that the algorithm is terminated when all nonbasic variables have nonnegative values of the quantity $(\Delta f)_q$.

We illustrate the working of the simplex algorithm by considering a simple constrained optimization problem.

EXERCISE A.1.1

Consider the following problem:

$$\text{Maximize} \quad f(x) = 2x_1 + 3x_2$$

subject to

$$x_1 \leq 6,$$

$$x_1 + 2x_2 \leq 10,$$

$$x_1, x_2 \geq 0.$$

The feasible solution space and the optimum solution are shown in Figure A.1. It is clear that the optimum solution is $(6, 2)^T$ with a function value equal to 18. Let us investigate how we can obtain this solution using the simplex method.

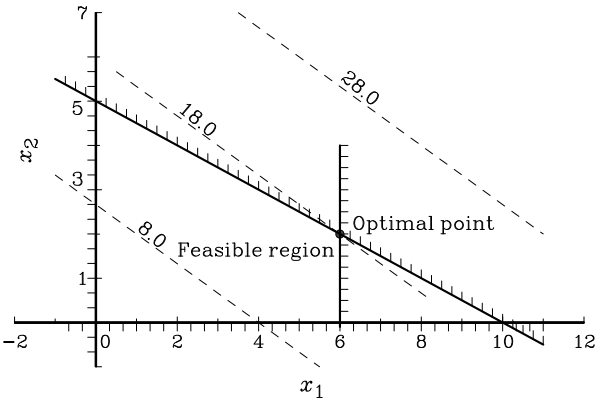


Figure A.1: The linear program described in Exercise A.2.1. The feasible region and the optimal solution are shown.

Step 1 First of all, we have to convert each inequality constraint into an equality constraint using a slack variable. Two inequality constraints are converted into the two equality constraints as follows:

$$x_1 \qquad + x_3 \qquad = 6,$$

$$x_1 + 2x_2 \qquad + x_4 = 10.$$

The variables x_3 and x_4 are slack variables and can take only nonnegative values. Thus, the optimization problem becomes as follows:

$$\text{Maximize} \quad f(x) = 2x_1 + 3x_2$$

subject to

$$x_1 \qquad + x_3 \qquad = 6,$$

$$x_1 + 2x_2 \qquad + x_4 = 10,$$

$$x_1, x_2, x_3, x_4 \geq 0.$$

This problem is now well represented by the problem described in Equation (A.1). We also observe that both constraints are already in row-echelon form with x_3 and x_4 being basic variables. Thus, we begin the simplex method with the initial feasible solution: $x_1 = 0$, $x_2 = 0$, $x_3 = 6$, and $x_4 = 10$. The constraints can be represented in the following tableau:

		2	3	0	0		
c_B	Basic	x_1	x_2	x_3	x_4		
0	x_3	1	0	1	0	6	$\frac{6}{0} = \infty$
0	x_4	1	2	0	1	10	$\frac{10}{2} = 5$ ←
$(\Delta f)_q$		2	3	0	0	$f(x) = 0$	
			↑				

The top row of the table represents the coefficients of the variables in the objective function. The next row shows all variables. The next two rows show the coefficients (a_{ij}) in the constraint expressions. There will be as many rows as there are constraints. The seventh column shows the corresponding b_j values in the constraints. The second column shows the basic variables and the first column shows the corresponding coefficients in the objective function. The bottom row shows the quantity $(\Delta f)_q$ for all variables. For basic variables, this quantity must be zero. For nonbasic variables, this quantity is calculated in Step 2. The above table represents the current linear programming problem and it contains all the information necessary to form the next row-echelon table. The table also represents the current solution—all nonbasic variables are zero and basic variables (x_q) are set equal to b_j where the row j corresponds to the coefficient $a_{qj} = 1$ of the basic variable. It is important to note that in the column of every basic variable there must be only one entry of $a_{ij} = 1$ and rest of the entries must be zero. This initial solution is marked as A in Figure A.2.

Step 2 The quantities $(\Delta f)_q$ are calculated for the nonbasic variables and are shown in the bottom row. It turns out that the quantity $(\Delta f)_q$ is maximum for the variable x_2. Thus, the nonbasic variable x_2 is to be made a basic variable. Since $(\Delta f)_2$ is positive, the optimal solution is not found. We use the minimum ratio rule to determine the basic

variable which is to be replaced. The right-most column shows these calculations. By calculating the quantity b_j/a_{2j}, we observe that the quantity is minimum for the basic variable x_4. Thus, the new basic variables are x_2 and x_3, and the new nonbasic variables are x_1 and x_4.

Step 3 We create a new row-echelon table using variables x_2 and x_3 as basic variables. The new parameter values are shown in the following tableau:

		2	3	0	0		
c_B	Basic	x_1	x_2	x_3	x_4		
0	x_3	1	0	1	0.0	6	$\frac{6}{1} = 6$ ←
3	x_2	0.5	1	0	0.5	5	$\frac{5}{0.5} = 10$
$(\Delta f)_q$		0.5	0	0	−1.5	$f(x) = 15$	
		↑					

The above table represents the solution $x_1 = 0$, $x_2 = 5$, $x_3 = 6$ and $x_4 = 0$. This solution is marked as B in Figure A.2. Notice that in one iteration, the simplex method moves to a neighbouring point of the initial point.

Step 2 Next, the quantity $(\Delta f)_q$ is calculated for all nonbasic variables. It is observed that only $(\Delta f)_1$ is positive. Thus, the variable x_1 is chosen to be the new basic variable. In order to decide which of the two older basic variables to replace, we use the minimum ratio rule. It turns out that the variable x_3 has the smallest b_j/a_{1j} value. Thus, the new basic variables are x_1 and x_2.

Step 3 At this step, we create a new row-echelon matrix with x_1 and x_2 as basic variables and obtain the following tableau:

		2	3	0	0	
c_B	Basic	x_1	x_2	x_3	x_4	
2	x_1	1	0	1.0	0.0	6
3	x_2	0	1	−0.5	0.5	2
$(\Delta f)_q$		0	0	−0.5	−1.5	$f(x) = 18$

The solution at this step is $x_1 = 6$, $x_2 = 2$, $x_3 = 0$, and $x_4 = 0$. This solution is marked as C in Figure A.2.

Step 2 At this step, the quantity $(\Delta f)_q$ for all nonbasic variables is nonpositive. Thus, the optimal solution is found. The optimal solution is $x^* = (6, 2)^T$ and the optimal function value is $f(x^*) = 18$.

Figure A.2 shows that the simplex method begins at a corner point of the feasible search region and visits only the neighbouring corner points in successive iterations. Since the optimal solution is bound to be one of the corner points of the feasible bounded region, the simplex method guarantees the convergence to the optimal solution in linear programming problems.

In the case of an unbounded optimal solution (where the optimal function value is not finite), the constraint entries for all basic variables corresponding to the chosen nonbasic variables are negative.

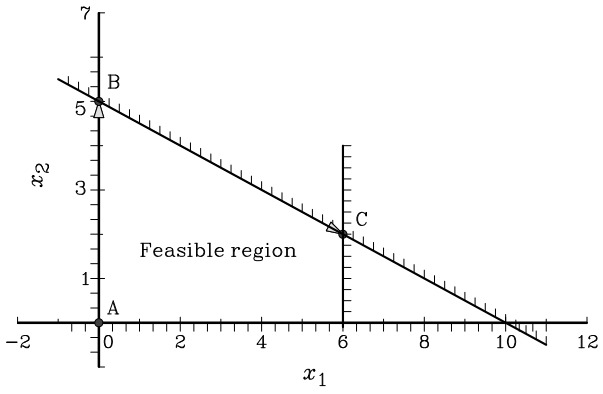


Figure A.2: The intermediate points found in the simulation of the simplex algorithm. The figure shows how neighbouring basic feasible solutions are found iteratively and the optimal solution is finally obtained.

A.2 Artificial Variables and Dual Phase Method

The working principle of the simplex method is straightforward and is easy to implement. There exist some commercial codes implementing the simplex method (IBM MPSX-370 or Management Science Systems MPS-III). However, a successful LP code must handle a difficulty usually encountered in many LP problems. The difficulty is that in some LP problems it may not be possible to obtain an initial basic feasible solution directly from the given constraints. In those situations, *artificial* variables are added. Artificial variables have no meaning as far as the given problem is concerned. They are simply included to obtain a basic feasible solution. Once a basic feasible solution is found, the artificial variables can be excluded from further consideration. Usually, a dual phase strategy is used for this purpose. In the first phase, the simplex method is used to find a basic feasible solution comprising the design variables, slack variables, and artificial variables (if any). The objective of the search strategy is to maximize the negative of the sum of the artificial variables. Since the artificial variables are also constrained to be nonnegative, the optimal solution of the first phase is a solution at which all artificial variables are zero. Since at the end of the first phase all artificial variables are zero, the basic variables are either design variables or slack variables. This constitutes a basic feasible solution which can be used as the initial solution for the next phase, where $f(x)$ is used as the objective function. We illustrate the dual phase method by solving the following exercise problem.
Let us add one more inequality constraint to the problem used in the previous exercise. Thus, the new problem is as follows:

$$\text{Maximize} \quad f(x) = 2x_1 + 3x_2$$

subject to

$$x_1 \leq 6,$$

$$x_1 + 2x_2 \leq 10,$$

$$x_1 + x_2 \geq 2,$$

$$x_1, x_2 \geq 0.$$

The solution to the above problem remains the same as that in the previous problem: $x^* = (6, 2)^T$.

Step 1 In order to convert the new inequality constraint into an equality constraint, we use another slack variable:

$$x_1 + x_2 - x_5 = 2,$$

where x_5 is nonnegative. We write all three constraints in the row-echelon form:

$$\begin{aligned} x_1 \quad & + x_3 \quad & = 6, \\ x_1 + 2x_2 \quad & + x_4 \quad & = 10, \\ x_1 + \ x_2 \quad & - x_5 & = 2, \\ x_1, x_2, x_3, x_4, x_5 & \geq 0. \end{aligned}$$

Since there are three constraints, there must be three basic variables. The variables x_3 and x_4 can be considered as basic variables. But the variables x_1, x_2, and x_5 cannot be considered as basic variables, because none of them have a coefficient one in one equation and zero in other equations. Thus, we add an artificial variable x_6 at the third constraint and form the following row-echelon form:

$$\begin{aligned} x_1 \quad & + x_3 \quad & = 6, \\ x_1 + 2x_2 \quad & + x_4 \quad & = 10, \\ x_1 + \ x_2 \quad & - x_5 + x_6 & = 2, \\ x_1, x_2, x_3, x_4, x_5, x_6 & \geq 0. \end{aligned}$$

Now, the variables x_3, x_4, and x_6 can be considered as basic variables. Thus, variables x_1, x_2, and x_5 are nonbasic variables. The initial basic feasible solution is $x_1 = x_2 = x_5 = 0$, $x_3 = 6$, $x_4 = 10$, and $x_6 = 2$. This is the initial solution of the first phase where the objective function is $-x_6$. We form the following tableau:

		0	0	0	0	0	−1		
c_B	Basic	x_1	x_2	x_3	x_4	x_5	x_6		
0	x_3	1	0	1	0	0	0	6	$\frac{6}{0} = \infty$
0	x_4	1	2	0	1	0	0	10	$\frac{10}{2} = 5$
−1	x_6	1	1	0	0	−1	1	2	$\frac{2}{1} = 2$ ←
$(\Delta f)_q$		1	1	0	0	−1	0	$f(x) = -2$	
			↑						

The initial solution is $(0, 0)^T$ and is marked as A in Figure A.3. It is interesting to note that this solution is not a feasible solution to the original problem, violating the third constraint.

Step 2 The quantity $(\Delta f)_q$ is calculated for variables x_1, x_2 and x_5. It is found from the above table that the quantity $(\Delta f)_q$ is maximum for both variables x_1 and x_2. We choose the variable x_2 (at random) to be the new basic variable. Next, we use the minimum ratio rule to choose the next new basic variable. The above table shows that the variable x_6 must be replaced with the variable x_2.

Step 3 Thus, we replace the variable x_6 by x_2 and form the next row-echelon tableau.

		0	0	0	0	0	−1	
c_B	Basic	x_1	x_2	x_3	x_4	x_5	x_6	
0	x_3	1	0	1	0	0	0	6
0	x_4	−1	0	0	1	2	−2	6
0	x_2	1	1	0	0	−1	1	2
$(\Delta f)_q$		0	0	0	0	0	−1	$f(x) = 0$

At this step, the solution is $(0, 2)^T$ and is marked as B in Figure A.3.

Step 2 Since the quantities $(\Delta f)_q$ for all basic variables are nonpositive, the optimal solution for the first phase is found. Therefore, we terminate the first phase and use this solution as the initial solution for the next phase. Here the solution corresponds to $x_1 = 0$, $x_2 = 2$, $x_3 =$, $x_4 = 6$, $x_5 = 0$, and $x_6 = 0$. The artificial variable at this solution is zero. Thus, this solution is a feasible solution.

Step 1 In the second phase, the objective function is the same as in the original problem: $f(x) = 2x_1 + 3x_2$. Since the artificial variable x_6 was introduced to obtain an initial basic feasible solution, we need not continue with that variable in the second phase. Thus, we tabulate the new row-echelon form:

		2	3	0	0	0		
c_B	Basic	x_1	x_2	x_3	x_4	x_5		
0	x_3	1	0	1	0	0	6	$\frac{6}{0} = \infty$
0	x_4	−1	0	0	1	$\boxed{2}$	6	$\frac{6}{2} = 3$ ←
3	x_2	1	1	0	0	−1	2	$\frac{2}{0} = \infty$
$(\Delta f)_q)$		−1	0	0	0	3		$f(x) = 6$
						↑		

At this step, the solution is $(0, 2)^T$.

Step 2 The quantity $(\Delta f)_q$ is calculated for all nonbasic variables in the above table. We observe that the variable x_5 needs to be made basic. Using the minimum ratio rule, we also observe that the variable x_4 needs to be exchanged with x_5. Notice that the objective function value is now calculated using the original function.

Step 3 We create the new row-echelon form as follows:

		2	3	0	0	0		
c_B	Basic	x_1	x_2	x_3	x_4	x_5		
0	x_3	$\boxed{1.0}$	0	1	0.0	0	6	$\frac{6}{1} = 6$ ←
0	x_5	−0.5	0	0	0.5	1	3	$\frac{3}{0} = \infty$
3	x_2	0.5	1	0	0.5	0	5	$\frac{5}{0.5} = 10$
$(\Delta f)_q$		0.5	0	0	−1.5	0		$f(x) = 15$
		↑						

At this step, the solution is $(0, 5)^T$ and is marked as C in Figure A.3. Notice that the objective function value at the current point is better than that in the previous iteration.

Step 2 The quantity $(\Delta f)_q$ is calculated for all nonbasic variables and only $(\Delta f)_1$ is positive. Again, the minimum ratio rule suggests that the basic variable x_3 needs to be made nonbasic.

Step 3 Thus, we form the new row-echelon table.

		2	3	0	0	0	
c_B	Basic	x_1	x_2	x_3	x_4	x_5	
2	x_1	1	0	1.0	0.0	0	6
0	x_5	0	0	0.5	0.5	1	6
3	x_2	0	1	−0.5	0.5	0	2
$(\Delta f)_q$		0	0	−0.5	−1.5	0	$f(x) = 18$

The solution obtained from the above table is $(6, 2)^T$ with an objective function value equal to 18.

Step 2 Since the quantities $(\Delta f)_q$ for all nonbasic variables (x_3 and x_4) are nonpositive, the optimal solution is obtained. Thus, we terminate the simplex method. Therefore, the final solution is $(6, 2)^T$ as marked D in Figure A.3.

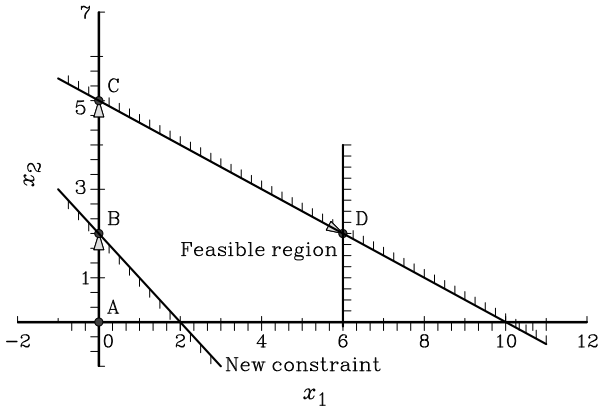


Figure A.3: The intermediate points found in the simulation of the dual phase method of the simplex algorithm. The first phase begins with the point A and the basic feasible solution B is found. In the next phase, the optimal solution D is obtained after two iterations.