Chapter 2
Optimization Techniques: An Overview

> *Since the fabric of the universe is most perfect, and is the work of a most wise Creator, nothing whatsoever takes place in the universe in which some form of maximum or minimum does not appear.*
>
> Leonhard Euler

It is an undeniable fact that all of us are optimizers as we all make decisions for the sole purpose of maximizing our quality of life, productivity in time, as well as our welfare in some way or another. Since this is an ongoing struggle for creating the best possible among many inferior designs, optimization was, is, and will always be the core requirement of human life and this fact yields the development of a massive number of techniques in this area, starting from the early ages of civilization until now. The efforts and lives behind this aim dedicated by many brilliant philosophers, mathematicians, scientists, and engineers have brought the high level of civilization we enjoy today. Therefore, we find it imperative to get to know first those major optimization techniques along with the philosophy and long history behind them before going into the details of the method detailed in this book. This chapter begins with a detailed history of optimization, covering the major achievements in time along with the people behind them. The rest of the chapter then draws the focus on major optimization techniques, while briefly explaining the mathematical theory and foundations over some sample problems.

2.1 History of Optimization

In its most basic terms, *Optimization* is a mathematical discipline that concerns the finding of the extreme (minima and maxima) of numbers, functions, or systems. The great ancient philosophers and mathematicians created its foundations by defining the optimum (as an extreme, maximum, or minimum) over several fundamental domains such as numbers, geometrical shapes optics, physics, astronomy, the quality of human life and state government, and several others. This era started with *Pythagoras of Samos* (569 BC to 475 BC), a Greek philosopher who made important developments in mathematics, astronomy, and the theory of music. He is often described as the first pure mathematician. His most important philosophical foundation is [1]: “that at its deepest level, reality is mathematical in nature.”

S. Kiranyaz et al., *Multidimensional Particle Swarm Optimization for Machine Learning and Pattern Recognition*, Adaptation, Learning, and Optimization 15,
DOI: 10.1007/978-3-642-37846-1_2, © Springer-Verlag Berlin Heidelberg 2014

Zeno of Elea (490 BC to 425 BC) who was a Greek philosopher famous for posing so-called paradoxes was the first to conceptualize the notion of extremes in numbers, or infinitely small or large quantities. He took a controversial point of view in mathematical philosophy, arguing that any motion is impossible by performing infinite subdivisions described by *Zeno's Dichotomy*. Accordingly, one cannot even start moving at all. Probably, *Zeno* was enjoying the challenging concept of "infinity" with his contemporaries without the proper formulation of the limit theory and calculus at the time.

Later, *Plato* (427 BC to 347 BC) who is one of the most important Greek philosophers and mathematicians, gained from the disciples of *Pythagoras*, and formed his idea [2], … "that the reality which scientific thought is seeking must be expressible in mathematical terms, mathematics being the most precise and definite kind of thinking of which we are capable. The significance of this idea for the development of science from the first beginnings to the present day has been immense." About 75 years earlier, *Euclid* wrote *The Elements, Plato* wrote *The Republic* around 375 BC, where he was setting his ideas on education: In that, one must study the five mathematical disciplines, namely arithmetic, plane geometry, solid geometry, astronomy, and harmonics. After mastering mathematics, one can proceed to the study of philosophy. The following dialog is a part of the argument he made:

> "…But when it is combined with the perception of its opposite, and seems to involve the conception of plurality as much as unity, then thought begins to be aroused within us, and the soul perplexed and wanting to arrive at a decision asks "What is absolute unity?" This is the way in which the study of the one has a power of drawing and converting the mind to the contemplation of reality."
>
> "And surely," he said, "this characteristic occurs in the case of one; for we see the same thing to be both one and infinite in multitude?"
>
> "Yes," I said, "and this being true of one, it must be equally true of all number?"
>
> "Certainly"

Aristotle (384 BC to 322 BC), who was one of the most influential Greek philosophers and thinkers of all times, made important contributions by systematizing deductive logic. He is perhaps best described by the authors of [3] as, "Aristotle, more than any other thinker, determined the orientation and the content of Western intellectual history. He was the author of a philosophical and scientific system that through the centuries became the support and vehicle for both medieval Christian and Islamic scholastic thought: until the end of the seventeenth century, Western culture was Aristotelian. And, even after the intellectual revolutions of centuries to follow, Aristotelian concepts and ideas remained embedded in Western thinking. "He introduced the well-known principle," The whole is more than the sum of its parts." Both the Greek philosophers, *Plato* and *Aristotle*, used their "powers of reasoning" to determine the best style of human life. Their goal was to develop the systematic knowledge of how the behavior of both individuals and society could be optimized. They particularly focused on questions of ethics (for optimizing the lifestyle of an individual) and politics (for optimizing the functioning of the state). At the end, both Plato and Aristotle recognized that the knowledge of how the members of society could *optimize* their lives was crucial. Both believed that the proper development of an individual's character traits was the key to living an *optimal* lifestyle.

As a follower of Plato's philosophy, *Euclid* of Alexandria (325 BC to 265 BC) was the most prominent antique Greek mathematician best known for his work on geometry, *The Elements,* which not only makes him the leading mathematician of all times but also one who influenced the development of Western mathematics for more than 2,000 years [4]. It is probable that no results in *The Elements* were first proved by *Euclid* but the organization of the material and its exposition are certainly due to him. He solved some of the earliest optimization problems in Geometry, e.g., in the third book, there is a proof that the greatest and least straight lines can be drawn from a point to the circumference of a circle; in the sixth book it is proven that a square has the *maximum* area among all rectangles with given total length of the edges.

Archimedes of Syracuse (287 BC to 212 BC) is considered by most historians of mathematics as one of the greatest mathematicians of all times [5]. He was the inventor of the water pump, the so-called *Archimedes' screw* that consists of a pipe in the shape of a helix with its lower end dipped in the water. As the device is rotated the water rises up the pipe. This device is still in use in many places in the world. Although he achieved great fame due to his mechanical inventions, he believed that pure mathematics was the only worthy pursuit. His achievements in calculus were outstanding. He perfected a method of integration which allowed him to find areas, volumes, and surface areas of many bodies by using the method of exhaustion, i.e., one can calculate the area under a curve by approximating it by the areas of a sequence of polygons. In Heath [6], it is stated that "Archimedes gave birth to the calculus of the infinite conceived and brought to perfection by Kepler, Cavalieri, Fermat, Leibniz and Newton." Unlike *Zeno* and other Greek philosophers, he and *Euclid* were the first mathematicians who were not troubled by the apparent contradiction of the *infinite* concept. For instance, they contrived the method of exhaustion technique to find the area of a circle *without* knowing the exact value of π.

Heron of Alexandria ($\sim$10 AC to $\sim$75 AC) who was an important geometer and worker in mechanics wrote several books on mathematics, mechanics, and even optics. He wrote the book, *Catoptrica*, which is attributed by some historians to *Ptolemy* although most now seem to believe that this was his genuine work indeed. In this book, *Heron* states that vision occurs as a result of light emissions by the eyes with infinite velocity. He has also shown that light travels between two points through the path of the *shortest* length.

Pappus of Alexandria ($\sim$290 AC to $\sim$350 AC) is the last of the great Greek geometers and made substantial contributions on many geometrical optimization problems. He proved what is known as the "honeycomb conjecture" that the familiar honeycomb shape, which is a repeating hexagonal pattern (volumetric hexagonal-shaped cylinders, stacked one against the other in an endless array) was the *optimal* way of storing honey. *Pappus* introduces this problem with one of the most charming essays in the history of mathematics, one that has frequently been excerpted under the title: *On the Sagacity of Bees*. In that, he speaks poetically of the divine mission of bees to bring from heaven the wonderful nectar known as honey, and says that in keeping with this mission they must make their honeycombs without any cracks through which honey could be lost. Having also a divine sense of symmetry, the bees had to choose among the regular shapes that could fulfill this condition, (e.g. *triangles, squares*, and *hexagons*). At the end they naturally chose the hexagon because a hexagonal prism required the *minimum* amount of material to enclose a given volume. He collected these ideas in his Book V and states his aim that [7], "Bees, then, know just this fact which is useful to them, that the hexagon is greater than the square and the triangle and will hold more honey for the same expenditure of material in constructing each. But we, claiming a greater share in wisdom than the bees, will investigate a somewhat wider problem, namely that, of all equilateral and equiangular plane figures having an equal perimeter, that which has the greater number of angles is always the greater, and the greatest of then all is the circle having its perimeter equal to them."

Also in Book V, *Pappus* discusses the 13 semi-regular solids discovered by *Archimedes* and solves other *isoperimetric* problems which were apparently discussed by the Athenian mathematician *Zenodorus* (200 BC to 140 BC). He compares the areas of figures with equal perimeters and volumes of solids with equal surface areas, proving that the sphere has the *maximum* volume among regular solids with equal surface area. He also proves that, for two regular solids with equal surface area, the one with the greater number of faces has the greater volume. In Book VII, *Pappus* defines the two basic elements of analytical problem solving, the analysis and synthesis [7] as, … "in analysis we suppose that which is sought to be already done, and inquire what it is from which this comes about, and again what is the antecedent cause of the latter, and so on until, by retracing our steps, we light upon something already known or ranking as a first principle… But in synthesis, proceeding in the opposite way, we suppose to be already done that which was last reached in analysis, and arranging in their natural order as consequents what were formerly antecedents and linking them one with another, we finally arrive at the construction of what was sought…"

During the time of the ancient great Greek philosophers and thinkers, arithmetic and geometry were the two branches of mathematics. There were some early attempts to do algebra in those days; however, they lacked the formalization of algebra, namely the arithmetic operators that we take for granted today, such as "$+$, $-$, $\times$, $\div$" and of course, "=". Much of the world, including Europe, also lacked an efficient numeric system like the one developed in the Hindu and Arabic cultures. *Al'Khwarizmi* (790–850) was a Muslim Persian mathematician who wrote on Hindu–Arabic numerals and was among the first to use the number *zero* as a place holder in positional base notation. Algebra as a branch of mathematics can be said to date to around the year 825 when *Al'Khwarizmi* wrote the earliest known algebra treatise, *Hisab al-jabr w'al-muqabala*. The word "algebra" comes from the Persian word *al'jabr* (that means "to restore") in the title. Moreover, the English term "algorithm," was derived from *Al'Khwarizmi* 's name as the way of a Latin translation and pronunciation: *Algoritmi*.

Ibn Sahl (940–1000) was a Persian mathematician, physicist, and optics engineer who was credited for first discovering the law of refraction, later called as the Snell's law. By means of this law, he computed the *optimum* shapes for lenses and curved mirrors. This was probably the first application of optimization in an engineering problem.

Further developments in algebra were made by the Arabic mathematician *Al-Karaji* (953–1029) in his treatise *Al-Fakhri*, where he extends the methodology to incorporate integer powers and integer roots of unknown quantities. Something close to a proof by mathematical induction appears in a book written by *Al-Karaji* who used it to prove the binomial theorem, Pascal's triangle, and the sum of integral cubes. The historian of mathematics, Woepcke in [8], credits him as *the first who introduced the theory of algebraic calculus*. This was truly one of the cornerstone developments for the area of optimization as it is one of the uses of calculus in the real world.

René Descartes (1596–1650) was a French mathematician and philosopher and his major work, *La Géométrie*, includes his linkage of algebra to geometry from which we now have the Cartesian geometry. He had a profound breakthrough when he realized he could describe any position on a 2D plane using a pair of numbers associated with a horizontal axis and a vertical axis—what we call today as "coordinates." By assigning the horizontal measurement with x's and the vertical measurement with y's, *Descartes* was the first to define any geometric object such as a line or circle in terms of algebraic equations. Scott in [9] praises his work for four crucial contributions:

1. He makes the first step toward a theory of invariants, which at later stages derelativises the system of reference and removes arbitrariness.
2. Algebra makes it possible to recognize the typical problems in geometry and to bring together problems which in geometrical dress would not appear to be related at all.
3. Algebra imports into geometry the most natural principles of division and the most natural hierarchy of method.
4. Not only can questions of solvability and geometrical possibility be decided elegantly, quickly, and fully from the parallel algebra, without it they cannot be decided at all.

The seminal construction of what we call *graphs* was obviously the cornerstone achievement without which any formulation of optimization would not be possible. In that, *Descartes* united the analytical power of algebra with the descriptive power of geometry into the new branch of mathematics, he named as *analytic geometry,* a term which is sometimes called as *Calculus with Analytic Geometry*. He was one of the first to solve the tangent line problem (i.e., the slope or the derivative) for certain functions. This was the first step toward finding the *maxima* or *minima* of any function or surface, the foundation of all analytical optimization solutions. On the other hand, when *Descartes* published his book, *La Géometrie* in 1637, his contemporary *Pierre de Fermat* (1601–1665) was already working on analytic geometry for about 6 years and he also solved the tangent line problem with a different approach, which is based on the approximation of the slope, converging to the exact value on the limit. This is the pioneer work for finding the derivative and henceforth calculating the optimum point when the slope is zero. Due to this fact, *Lagrange* stated clearly that he considers *Fermat* to be the inventor of the calculus. But at that time, such an approximation-based approach was perhaps why *Fermat* did not get the full credit for his work. There was an ongoing dispute between the two because *Descartes* thought that *Fermat's* work was reducing the importance of his own work *La Géometrie.* Fermat initiated the technique for solving $\mathrm{d}f(x)/\mathrm{d}x = 0$ to find the local optimum point of the function $f(x)$ and this is perhaps the basis in applied mathematics and its use in optimization. Another reason for the dispute between them might be that *Fermat* found a mistake in a book by *Descartes* and corrected it. *Descartes* attacked *Fermat's* method of maxima, minima, and tangents but in turn, *Fermat* proved correct and eventually *Descartes* accepted his mistake. Fermat's most famous work, called *Fermat's Last Theorem*, was the proof for the statement, $x^n + y^n = z^n$, has no integer solutions for $n > 2$. His proof remains a mystery till today since Fermat wrote it as, "I have discovered a truly remarkable proof which this margin is too small to contain." He also deduced the most fundamental optimization phenomenon in the optics, "the light always follows the shortest possible path", (or similarly, "the light follows the path which takes the shortest time").

Like most developments, the calculus too was the culmination of centuries of work. After these pioneers, the two most recognized discoverers of calculus are *Isaac Newton* of England (1643–1727) and a German, *Gottfried Wilhelm Leibniz* (1646–1716). Both deserve equal credit for independently coming up with calculus; however, at that time a similar rivalry and dispute occurred between the two as each accused the other of plagiarism for the rest of their lives. The mathematics community today has largely adopted *Leibniz's* calculus symbols but on the other hand, the calculus he discovered allowed *Newton* to establish the well-known physics laws which are sufficient in macro scale to explain many physical phenomena in nature to a remarkable accuracy. Their approach to calculus was also totally different, i.e., *Newton* considered functions changing in time, whereas *Leibniz* thought of variables x, y as ranging over sequences of infinitely close values, $\mathrm{d}x$ and $\mathrm{d}y$; however, he never thought of the derivative as a limit. In 1666, Newton found out the slope of a function by the derivative and solved the inverse problem by taking the integral, which he used to calculate the area under any function. Therefore, this work contains the first clear statement of the *Fundamental Theorem of Calculus.* As *Newton* did not publish his findings until 1687, unaware that he had discovered similar methods, Leibniz developed his calculus in Paris between 1673 and 1676. In November 1675, he wrote a manuscript using the common integral notation, $\int f(x)\,\mathrm{d}x$, for the first time. The following year, he discovered the power law of differentiation, $\mathrm{d}(x^n) = nx^{n-1}\mathrm{d}x$ for both integer and fractional n. He published the first account of differential calculus in 1684 and then published the explanation of integral calculus in 1686. There were rumors that *Leibniz* was following *Newton's* studies from their common colleagues and occasional discussion letters between the two, but despite of all his correspondences with *Newton*, he had already come to his own conclusions about calculus. In 1686 Leibniz published a paper, in *Acta Eruditorum*, dealing with the integral calculus with the first appearance of the integral notation in print. *Newton's* famous work, *Philosophiae Naturalis Principia Mathematica*, surely the greatest scientific book ever written, appeared in the following year. The notion that the Earth rotated around the Sun was already known by ancient Greek philosophers, but it was Newton who explained *why*, and henceforth, the great scientific revolution began with it.

During his last years, *Leibniz* published *Théodicée* claiming that the universe is in the *best* possible form but imperfect; otherwise, it would not be distinct from God. He invented more mathematical terms than anyone else, including "function," "analysis situ," "variable," "abscissa," "parameter," and "coordinate." His childhood IQ has been estimated as the second-highest in all of history, behind only Goethe [5]. Descriptions that have been applied to *Leibniz* include "one of the two greatest universal geniuses" (*da Vinci* was the other) and the "Father of the Applied Science." On the other hand, *Newton* is the genius who began revolutionary advances on calculus, optics, dynamics, thermodynamics, acoustics, and physics; it is easy to overlook that he too was one of the greatest geometers for he calculated the *optimum* shape of the bullet earlier than his invention of calculus. Among many brilliant works in mathematics and especially in calculus, he also discovered the Binomial Theorem, the polar coordinates, and power series for exponential and trigonometric functions. For instance, his equation, $e^x = \sum x^k / k!$, has been called as "the most important series in mathematics." Another optimization problem he solved is the *brachistochrone*, which is the curve of *fastest* descent between two points by a point-like body with a zero velocity while the gravity is the only force (with no friction). This problem had defeated the best mathematicians in Europe but it took Newton only a few hours to solve it. He published the solution anonymously, yet upon seeing the solution, *Jacob Bernoulli* immediately stated "I recognize the lion by his footprint."

After the era of *Newton* and *Leibniz,* the development of the calculus was continued by the Swiss mathematicians, *Bernoulli* brothers, *Jacob Bernoulli* (1654–1705), and *Johann Bernoulli* (1667–1748). Jacob was the first mathematician who applied separation of variables in the solution of a first-order nonlinear differential equation. His paper of 1690 was indeed a milestone in the history of calculus since the term integral appears for the first time with its *integration* meaning. *Jacob* liked to pose and solve physical *optimization* problems such as the catenary (which is the curve that an idealized hanging chain assumes under its own weight when supported only at its ends) problem. He was a pioneer in the field of calculus of variations, and particularly differential equations, with which he developed new techniques to many optimization problems. In 1697, he posed and partially solved the isoperimetric problem, which is a class of problems of the calculus of variations. The simplest of them is the following: among all curves of given length, find the curve for which some quantity (e.g., area) dependent on the curve reaches a minimum or maximum. Other optimization problems he solved include isochronous curves and curves of *fastest* descent. *Johann Bernoulli* on the other hand, followed a similar path like his brother. He mostly learned from him and also from *Leibniz*, and later on he alone supported *Leibniz* for the Newton–Leibniz controversy. He showed *Leibniz*'s calculus can solve certain problems where *Newton* had failed. He also became the principle teacher of *Leonhard Euler*. He developed the exponential calculus and together with his brother *Jacob*, founded the calculus of variations. Although their work of line was pretty similar, there were no common papers published, because after a while a bitter jealousy led to another famous rivalry, this time between the *Bernoulli* brothers, who—especially *Johann*—began claiming each other's work. Later, a similar jealousy arose between *Johann* and his son, *Daniel Bernoulli* (1700–1782), where this time *Johann* started to compete with him on his most important work, *Hydrodynamica* in 1734 which *Daniel* published in 1738 at about the same time as *Johann* published a similar version of it, *Hydraulica.* However, he discovered *L'Hôpital*'s Rule 50 years before *Guillaume de L'Hôpital* (1661–1704) did, and according to some mathematics historians, he solved the catenary problem before his brother did, although he used ideas that *Jacob* had given when he posed the problem. He attained great fame in his life and made outstanding contributions in calculus and physics for solving many real optimization problems, e.g., about vibrations, elastic bodies, optics, tides, and ship sails.

Calculus of variations is an area of calculus that deals with the *optimization* of *functionals*, which are mappings from a set of functions to real numbers and are often expressed as definite integrals involving functions and their derivatives. A physical system can be modeled by functionals, with which their variables can be optimized considering the constraints. Calculus of variations and the use of differential equations for the general solution of many optimization problems may not be possible without the Swiss mathematician, *Leonhard Euler* (1701–1783) who is probably the most influential mathematician ever lived. He is the father of mathematical analysis and his work in mathematics is so vast that we shall only name the few crucial developments herein for calculus and optimization. He took marvelous advantage of the analysis of *Fermat, Newton, Leibniz,* and the *Bernoulli* family members, extending their work to marvelous heights. To start with, many fundamental calculus and mathematical foundations that are used today were created by *Euler*, i.e., in 1734, he proposed the notation $f(x)$ for a function, along with three of the most important constant symbols in mathematics: e for the base of natural logs (in 1727), i for the square root of -1 (in 1777), π for pi, and several mathematical notations such as "$\sum$" for summation (in 1755), finite differences Δx, $\Delta^2 x$, and many others. In 1748, he published his major work, *Introductio in analysin infinitorum* in which he presented that mathematical analysis was the study of functions. This was the pioneer work, which bases the foundations of calculus on the theory of elementary functions rather than on geometric curves, as had been done earlier. Also in this work, he unifies the trigonometric and exponential functions by his famous formula: $e^{ix} = \cos x + i \sin x$. (The particular setting of $x = \pi$ yields $e^{i\pi} + 1 = 0$ that fits the three most important constants in a single equation.)

Euler was first partially and later totally blind during a long period of his life. From the French Academy in 1735, he received a problem in celestial mechanics, which had required several months to solve by other mathematicians. *Euler*, using his improved methods solved it in 3 days (later, with his superior methods, *Gauss* solved the same problem within an hour!). However, the strain of the effort induced a fever that caused him the loss of sight in his right eye. Stoically accepting the misfortune, he said, "Now I will have less distraction." For a period of 17 years, he was almost totally blind after a cataract developed in his left eye in 1766. Yet he possessed a phenomenal memory, which served him well during his blind years as he calculated long and difficult problems on the blackboard of his mind, sometimes carrying out arithmetical operations to over 50 decimal places. The *calculus of variations* was created and named by *Euler* and in that he made several fundamental discoveries. His first work in 1740, *Methodus inveniendi lineas curvas* initiated the first studies in the calculus of variations. However, his contributions already began in 1733, and his treaty, *Elementa Calculi Variationum* in 1766 gave its name. The idea was already born with the *brachistochrone curve* problem raised by *Johann Bernoulli* in 1696. This problem basically deals with the following: Let a point particle of mass m on a string whose endpoints are at $a = (0,0)$ and $b = (x,y)$, where $y < 0$. If gravity acts on the particle with force $F = \mathrm{mg}$, what path of string *minimizes* its travel time from a to b, assuming no friction? The solution of this problem was one of the first accomplishments of the calculus of variations using which many optimization problems can be solved. Besides *Euler*, by the end of 1754, *Joseph-Louis Lagrange* (1736–1813) had also made crucial discoveries on the *tautochrone* that is the curve on which a weighted particle will always arrive at a fixed point in a fixed amount of time independent of its initial position. This problem too contributed substantially to the calculus of variations. *Lagrange* sent *Euler* his results on the *tautochrone* containing his method of maxima and minima in a letter dated 12 August 1755, and *Euler* replied on 6 September saying how impressed he was with *Lagrange's* new ideas (he was 19 years old at the time). In 1756, *Lagrange* sent *Euler* results that he had obtained on applying the calculus of variations to mechanics. These results generalized results which *Euler* had himself obtained. This work led to the famous *Euler–Lagrange* equation, the solution of which is applied on many optimization problems to date. For example using this equation, one can easily show that the closed curve of a given perimeter for which the area is a *maximum*, is a circle, the *shortest* distance between two fixed points is a line, etc. Moreover, *Lagrange* considered optimizing a functional with an added constraint and he turned the problem using the method of Lagrange multipliers to a single optimization equation that can then be solved by the *Euler–Lagrange* equation.

Euler made substantial contributions to differential geometry, investigating the theory of surfaces and curvature of surfaces. Many unpublished results by *Euler* in this area were rediscovered by *Carl Friedrich Gauss* (1777–1855). In 1737, *Euler* wrote the book, *Mechanica* which provided a major advance in mechanics. He analyzed the motion of a point mass both in a vacuum, in a resisting medium, under a central force, and also on a surface. In this latter topic, he solved various problems of differential geometry and geodesics. He then addressed the problem of ship propulsion using both theoretical and applied mechanics. He discovered the *optimal* ship design and first established the principles of hydrostatics.

Based on these pioneers' works, during the nineteenth century, the first optimization algorithms were developed. During this era, many brilliant mathematicians including *Jakob Steiner* (1796–1863), *Karl Theodor Wilhelm Weierstrass* (1815–1897), *William Rowan Hamilton* (1805–1865), and *Carl Gustav Jacob Jacobi* (1804–1851) made significant contributions to the field of calculus of variations. The first iterative optimization technique that is known as Newton's method or Newton–Raphson method was indeed developed by four mathematicians: *Isaac Newton, Joseph Raphson* (1648–1715), *Thomas Simpson* (1710–1761), and *Jean-Baptiste-Joseph Fourier* (1768–1830). *Newton* in 1664 found a non-iterative algebraic method of root finding of a polynomial and in 1687, he described an application of his procedure to a non-polynomial equation in his treaty *Principia Mathematica* where this was originated from *Kepler's* equation. Note that this was a purely algebraic and non-iterative method. In 1690, *Raphson* turned *Newton's* method into an iterative one, applying it to the solution of polynomial equations of degree up to 10. However, the method is still not based on calculus; rather explicit polynomial expressions are used in function form, $f(x)$ and its derivative, $f'(x)$. Simpson in 1740 was the first to formulate the *Newton–Raphson* method on the basis of calculus, extending it to an iterative solver for the multivariate minimization. *Fourier* in 1831 brought the method as we know of it today and published in his famous book, *Analyse des équations déterminées.* The method finds the root of a scalar function, $f(x) = 0$, iteratively by the following equation using only the first-order derivative, $f'(x)$.

$$x_{k+1} = x_k - f(x_k)/f'(x_k), k = 0, 1, 2, \ldots \quad (2.1)$$

where the initial guess, x_0, is usually chosen randomly. Similarly, finding the root off', which is equivalent to the optimum (or stationary) point of the function, $f(x)$, can similarly be expressed as,

$$x_{k+1} = x_k - f'(x_k)/f''(x_k), \quad k = 0, 1, 2, \ldots \quad (2.2)$$

Augustin-Louis Cauchy (1789–1857) did important work on differential equations and applications to mathematical physics. The four-volume series, *Exercices d'analyse et de physique mathématique* published during 1840–1847 was a major work in this area in which he proposed the method of the *steepest descent* (also called as *gradient descent*) in 1847. This method is perhaps one of the most fundamental and basic derivative-based iterative procedures for unconstrained minimization of a differentiable function. Given a differentiable function in N-D, $f(\bar{x})$, the gradient method in each step moves along the direction that minimizes ∇f, that is, the direction of *steepest* descent and thus perpendicular to the slope of the curve at that point. The method stops when it reaches to a local minimum (maximum) where $\nabla f = 0$ and thus no move is possible. Therefore, the update equation is as follows:

$$\bar{x}_{k+1} = \bar{x}_k - \lambda_k \nabla f(\bar{x}_k), \quad k = 0, 1, 2, \ldots \tag{2.3}$$

where $\bar{x}_0$ is the initial starting point in the N-D space. An advantage of *gradient descent* compared to *Newton–Raphson* method is that it only utilizes first-order derivative information about the function when determining the direction of movement. However, it is usually slower than *Newton–Raphson* to converge and it tends to suffer from very slow convergence especially as a stationary point is approached.

A crucial optimization application is the *least-square approximation*, which finds the approximate solution of sets of equations in which there are more equations than unknowns. At the age of 18, *Carl Friedrich Gauss* who was widely agreed to be the most brilliant and productive mathematician ever lived, invented a solution to this problem in 1795, although it was first published by *Lagrange* in 1806. This method basically minimizes the sum of the squares of the residual errors, that is, the overall solution minimizes the sum of the squares of the errors made in the results of every single equation. The most important application is in data fitting and the first powerful demonstration of it was made by *Gauss*, at the age of 24 when he used it to predict the future location of the newly discovered asteroid, *Ceres*. In June 1801, *Zach*, an astronomer whom *Gauss* had come to know two or three years earlier, published the orbital positions of *Ceres*, which was discovered by an Italian astronomer *Giuseppe Piazzi* in January, 1801. *Zach* was able to track its path for 40 days before it was lost in the glare of the sun. Based on this data, astronomers attempted to determine the location of *Ceres* after it emerged from behind the sun without solving the complicated *Kepler's* non-linear equations of planetary motion. *Zach* published several predictions of its position, including one by Gauss which differed greatly from the others. When *Ceres* was rediscovered by Zach on 7 December 1801, it was almost exactly where *Gauss* had predicted using the *least-squares method*, which was not published at the time.

The twentieth century brought the proliferation of several optimization techniques. Calculus of variations was further developed by several mathematicians including *Oskar Bolza* (1857–1942) and *Gilbert Bliss* (1876–1951). *Harris Hancock* (1867–1944) in 1917 published the first book on optimization, *Theory of Maxima and Minima*. One of the crucial techniques of the optimization, *Linear Programming* (LP), was developed in 1939 by the Russian mathematician, *Leonid Vitaliyevich Kantorovich* (1912–1986); however, the method was kept secret until the time the American scientist, *George Bernard Dantzig* (1914–2005) published the *Simplex* method in 1947. LP, sometimes called linear optimization, is a mathematical method for determining a way to achieve the optimal outcome in a given mathematical model according to a list of requirements that are predefined by some linear relationships. More formally, LP is a technique for the optimization of a linear objective function, subject to constraints expressed by linear (in)equalities. In an LP, the variables are continuous while the objective function and constraints must be linear expressions. An expression is linear if it can be expressed in the form, $c_1x_1 + c_2x_2 + \ldots + c_nx_n$ for some constants $c_1, c_2, \ldots, c_n$. The solution space corresponds to a convex polyhedron, which is a set defined as the intersection of finitely many half spaces, each of which is defined by a linear inequality. Its objective function that is to be optimized (under the given constraints) is a real-valued affine function defined on this polyhedron. In short, the LP method finds an optimum point in the polyhedron—if it exists. The *Simplex* method, on the other hand, is an efficient method for finding the optimal solution in one of the corners of the N-D polyhedron where N is the number of linear (in)equalities each intersecting to yield a corner. The *Simplex* method iteratively searches each corner to find the optimum one, which corresponds to the optimum solution. Finding each of the N corners is a matter of solving a system of N equations that can be done by Gaussian elimination method.

John Von Neumann (1903–1957) developed the theory of duality as an LP solution, and applied it in the field of game theory. If an LP exists in the maximization linear form, which is called the *primal* LP, its dual is formed by having one variable for each constraint of the primal (not counting the non-negativity constraints of the primal variables), and having one constraint for each variable of the primal (plus the non-negative constraints of the dual variables); then the maximization can be switched to minimization, the coefficients of the objective function are also switched with the right-hand sides of the inequalities, and the matrix of coefficients of the left-hand side of the inequalities are transposed. In 1928, *Von Neumann* proved the *minimax* theorem in the game theory, which indicates that there exists a pair of strategies for both players that allows each one to minimize his maximum losses. Each player examines every possible strategy and must consider all the possible responses of his adversary. He then plays out the strategy which will result in the minimization of his maximum loss. Such a strategy, which minimizes the maximum loss for each player is called the optimal *minmax* solution. Alternatively, the theorem can also be thought of as *maximizing* the minimum gain (*maximin*).

In 1939, *Nonlinear Programming* (NLP or *Nonlinear Optimization*) was first developed by a graduate student *William Karush* (1917–1997), who was also the first to publish the necessary conditions for the inequality constrained problem in his Master's thesis, *Minima of Functions of Several Variables with Inequalities as Side Constraints*. The optimal solution by the NLP was only widely recognized after a seminal conference paper in 1951 by *Harold William Kuhn* (born in 1925) and *Albert William Tucker* (1905–1995). Thus the theory behind NLP was called the *Karush–Kuhn–Tucker* (KKT) *Theory*, which provided necessary and sufficient conditions for the existence of an optimal solution to an NLP. NLP has a particular importance in optimal control theory and applications since optimal control problems are optimization problems in (infinite-dimensional) functional spaces, while NLP deals with the optimization problems in Euclidean spaces; optimal control can indeed be seen as a generalization of NLP.

In 1952, *Richard Bellman* (1920–1984) made the first publication on *dynamic programming* (DP), which is a commonly used method of optimally solving complex problems by breaking them down into simpler problems. DP is basically an algorithmic technique, which uses a recurrent formula along with one or more starting states. A subsolution of the problem is constructed from those previously found. DP solutions have a polynomial complexity, which assures a much faster running time than other techniques such as backtracking and brute-force. The problem is first divided into "states," each of which represents a sub-solution of the problem. The state variables chosen at any given point in time are often called the "control" variables. Finally, the *optimal* decision rule is the one that achieves the best possible value from the objective function, which is written as a function of the state, called the "value" function. *Bellman* showed that a DP problem in discrete time can be stated in a recursive form by writing down the relationship between the value function in one period and the value in the next period. The relationship between these two value functions is called the *Bellman* equation. In other words, the *Bellman* equation, also known as a DP equation, is a necessary condition for optimality. The Bellman equation can be solved by *backwards induction*, either analytically in a few special cases, or numerically on a computer. Numerical *backwards induction* is applicable to a wide variety of problems, but may be infeasible when there are many state variables, due to the "Curse of Dimensionality," which is a term coined by Bellman to describe the problem caused by the exponential increase in volume associated with adding extra dimensions to the (search) space. One implication of the curse of dimensionality is that some methods for numerical solution of the Bellman equation require vastly more computer time when there are more state variables in the value function.

All of the optimization methods described till now have been developed for deterministic processes applied over known differentiable (and double differentiable) functions. Optimization by *Stochastic Approximation* (SA) aims at finding the minima or maxima of an unknown function with unknown derivatives, both of which can be confounded by random error. Therefore, SA methods belong to the family of iterative stochastic optimization algorithms, which converge to the optimum points of such functions that cannot be computed directly, but only estimated via noisy observations. SA is a part of the stochastic optimization (SO) methods that generate and use random variables, which appear in the formulation of the optimization problem itself, along with the random objective functions or random constraints. Stochastic approximation was introduced in 1951 by the American mathematicians *Herbert Ellis Robbins* (1915–2001) and his student, *Sutton Monro* (1919–1995) [10]. This algorithm is a root finder of the equation, $\theta(x) = 0$ which has a unique root at $x = \alpha$. It is assumed that one cannot observe directly the function, $\theta(x)$, rather we have the noisy measurements of it, $N(X)$, where $\mathrm{E}(N(X)) = \theta(x)$ ($E(.)$ is the mathematical expectation operation). The algorithm then iterates toward the root in the form: $x_{k+1} = x_k + c_k(\alpha - N(x))$ where $c_1, c_2, \ldots$is a sequence of positive step sizes. They suggested the form of $c_k = c/k$ and proved that under certain conditions, x_k converges to the root, α.

Motivated by the publication of the *Robbins-Monro* algorithm in 1951, the *Kiefer-Wolfowitz* algorithm [11] was introduced in 1952 by the Polish mathematician, *Jacob Wolfowitz* (1910–1981) and the American statistician, *Jack Kiefer* (1924–1981). This is the first stochastic optimization method which seeks the maximum point of a function. This method suffers from heavy function computations since it requires $2(d + 1)$ function computations for each gradient computation, where d is the dimension of the search space. This is a particularly a significant drawback in high dimensions. To address this drawback, *James Spall* in [12], proposed the use of simultaneous perturbations to estimate the gradient. This method would require only two function computations per iteration, regardless of the dimension. For any SA method, applied over unimodal functions, it can be shown that the method can converge to the local optimum point with probability one.

However, for multimodal functions, SA methods, as all other gradient-based deterministic algorithms may be stuck on a local optimum. The convergence to the global optimum point is a crucial issue, yet most likely infeasible by any of these gradient-based methods. This brought the era of probabilistic metaheuristics. The American physicist *Nicholas Metropolis* (1915–1999) in 1953 co-authored the paper, *Equation of State Calculations by Fast Computing Machines*, a technique that was going to lead to the first probabilistic metaheuristics method, now known as *simulated annealing*. After this landmark publication, *Keith Hastings* (born in 1930) extended it to a more general case in 1970, by developing the *Metropolis–Hastings* algorithm, which is a Markov chain Monte-Carlo method for creating a series of random numbers from a known probability density function. In 1983, the adaptation of this method led to the simulated annealing method [13], which is a generic probabilistic metaheuristics for the global optimization problem so as to converge to the global optimum of any function in a large search space. The name *annealing* basically mimics the process undergone by misplaced atoms in a metal when it is first heated and then slowly cooled. With a similar analogy, each step of the simulated annealing attempts to replace the current solution with a new solution chosen randomly according to a certain probability distribution. This new solution may then be accepted with a probability that depends both on the difference between the corresponding function values and also on a global parameter T (called the temperature) that is gradually decreased during the process (*annealing*). When T is large, the choice between the new and the previous solution becomes almost purely random and as T goes to zero; it consistently selects the best solution between the two, mimicking a steepest descent (or ascent) method. Therefore, especially when T is large (during the early stages of *simulated annealing*'s iterative algorithm), it prevents the early trappings to a local minima and then yields the convergence to the optimum point as T goes to zero. While this technique cannot guarantee finding the optimum solution, it can often find a suboptimum point in the close vicinity of it, even in the presence of noisy data.

The 1950s and early 1960s were the times when the use of the computers became popular for a wide range of optimization problems. During this era, *direct search methods* first appeared, whereas the name "direct search" was introduced in 1961 by *Robert Hooke* and *T. A. Jeeves*. This was a pattern search method, which is better than a random search due to its search directions by exploration in the search space. After this key accomplishment, in 1962 the first simplex-based direct search method was proposed by *W. Spendley, G. R. Hext,* and *F. R. Himsworth* in their paper, *Sequential Application of Simplex Designs in Optimisation and Evolutionary Operation*. Note that this is an entirely different algorithm than the *Simplex* method for LP as discussed earlier. It uses only two types of transformations to form a new simplex (e.g., vertices of a triangle in 2D) in each step: *reflection* away from the worst vertex (the one with the highest function value), or *shrinking* toward the best vertex (the one with the lowest function value). For each iteration, the angles between simplex edges remain constant during both operations, so the working simplex can change in size, but not in shape. In 1965, this method was modified by *John Ashworth Nelder* (1924–2010) and *Roger Mead* who added two more operators: *expansion* and *contraction* (in and out), which allow the simplex to change not only its size, but also its shape [14]. Their modified simplex method, known as *Nelder-Mead (or simplex) method,* became immediately famous due to its simplicity and low storage requirements, which makes it an ideal optimization technique especially for the primitive computers at that time. During the 1970s and 1980s, it was used by several software packages while its popularity grew even more. It is now a standard method in MATLAB© where it can be applied by the command: *fminsearch.* Nowadays, despite its long past history, the *simplex method,* is still one of the most popular heuristic optimization techniques in use.

During the 1950s and 1960s, the concept of artificial intelligence (AI) was also born. Along with the AI, a new family of metaheuristic optimization algorithms in stochastic nature was created: *evolutionary algorithms* (EAs). An EA uses mechanisms inspired by biological *evolution* such as reproduction, *mutation*, *recombination*, and *selection*. It is also a stochastic method as in simulated annealing; however, it is based on the collective behavior of a population. A potential solution of the optimization problem plays the role of a member in the population, and the fitness function determines the search space within which the solutions lie. The earliest instances of EAs appeared during the 1950s and early 1960s, simulated on computers by evolutionary biologists who were explicitly seeking to model aspects of natural evolution. At first, it did not occur to any of them that this approach might be generally applicable to optimization problems. The EAs were first used by a Norwegian-Italian mathematician; *Nils Aall Barricelli* (1912–1993) who applied to evolutionary simulations. By 1962, several researchers developed evolution-inspired algorithms for function optimization and machine learning, but at the time their work only attracted little attention. The first development in this field for optimization came in 1965, when the German scientist *Ingo Rechenberg* (born in 1934), developed a technique called *evolution strategy* (ES), which uses natural problem-dependent representations, and primarily mutation and selection, as search operators in a loop where each iteration is called *generation*. The sequence of generations is continued until a termination criterion is met.

The next EA member came in 1966, when an American aerospace engineer, *Lawrence Fogel* (1928–2007) developed *evolutionary programming* (EP) where a potential solution of a given problem in hand is represented by simple finite-state machines as predictors. Similar to evolution strategies, EP performs random *mutation* of a simulated machine and keeps the best one. However, both EAs still lack a crucial evolutionary operator, the *crossover*. As early as 1962, *John Holland* (born in 1929) performed the pioneer work on adaptive systems, which laid the foundation for a new EA, *genetic algorithms* (GAs). Holland was also the first to explicitly propose crossover and other recombination operators. In 1975 he wrote the ground-breaking book on GA, "Adaptation in Natural and Artificial Systems." Based on earlier work on EAs by himself and by colleagues at the University of Michigan, this book was the first to systematically and rigorously present the concept of adaptive digital systems using evolutionary operators such as *mutation*, *selection* and *crossover*, simulating processes of natural evolution. In a GA, a population of strings (called chromosomes), which encodes potential solutions (called individuals, creatures, or phenotypes) of an optimization problem, evolves toward better solutions using these operators in an iterative way. Traditionally, solutions are represented in binary strings of 0s and 1s, but other encodings are also possible. The evolution usually starts from a population of randomly generated individuals and in each generation, the fitness of every individual in the population is evaluated, multiple individuals are stochastically selected from the current population (based on their fitness), and modified (recombined and possibly randomly mutated) to form a new generation, and so on. The GA process is terminated either when a successful solution or a maximum number of generations is reached. These foundational works established more widespread interest in evolutionary computation. By the early to mid-1980s, GAs were being applied to a broad range of fields, from abstract mathematical problems to many engineering problems such as pipeline flow control, pattern recognition and classification, and structural optimization.

In 1995, differential evolution (DE) as the most recent EA was developed by *Rainer Storn* and *Kenneth Price*. Similar to GA and many other EAs, DE is a population-based technique, which performs evolutionary operators, *mutation, crossover,* and *selection* in a certain way and the candidate solutions are represented by agents based on floating point number arrays (or vectors). As any other EA, it is a generic optimization method that can be used on optimization problems that are noisy, dynamic, or not even continuous. In each generation, it creates new candidate solutions by combining existing ones according to its simple expression, and then keeping whichever candidate solution has the best score or fitness. This is a typical process of an EA, especially resembling GA; however, DE has a distinct property of interaction among individuals, that is, each individual (agent) is mutated with respect to three others. A similar concept of interaction became the basis and the key element in one of the latest and the most successful methods in the era of probabilistic metaheuristics, the *Particle Swarm Optimization* (PSO), which was proposed in 1995 by *Russell C. Eberhart* and *James Kennedy*. PSO was first intended as a simulation program for the social behavior and stylized representation of the movement of organisms in a bird flock or fish school. The algorithm was simplified and it was observed to be performing optimization. We shall cover the details and the philosophy behind the PSO in Chap. 3 and, therefore, in the forthcoming sections in this chapter, we shall now detail the major optimization methods prior to PSO.

2.2 Deterministic and Analytic Methods

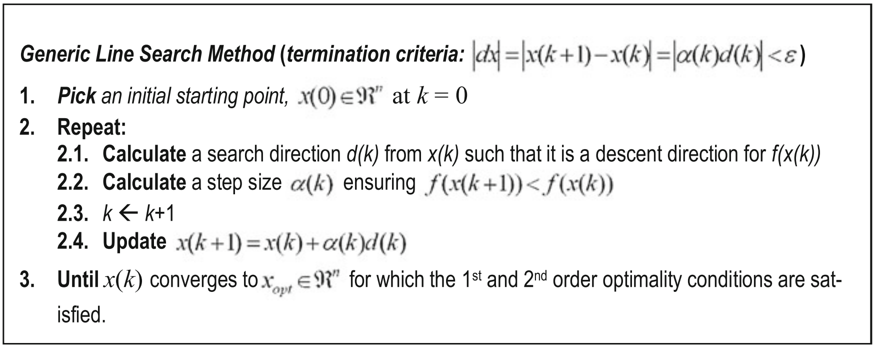
Assume an unconstrained optimization problem, such as $\min\limits_{x \in \Re^n} f(x)$, where the objective function $f : \Re^n \to \Re$ is sufficiently smooth with continuous second derivative. It is well known from the theory of functions from Calculus that the necessary and sufficient conditions for x^* to be a local minimum are (1) *gradient* $f'(x^*) = 0$ and (2) *Hessian* $H(x^*)$ $(= \nabla^2 f(x))$ is positively definite. For some problems, the solution can be obtained analytically by determining the zeros of the gradient and verifying positive definiteness of the *Hessian* matrix at these points. One particularly interesting property of an objective function is convexity. If f is a convex function, satisfying $f(\alpha x + (1-\alpha)y) \le \alpha f(x) + (1-\alpha) f(y), \alpha \in [0,1]$, then it has only one (global) minimum. There are effective methods for solving convex optimization problems [15].

For one-dimensional (and possibly multi-dimensional) unconstrained optimization problems, such as $\min\limits_{x \in \Re^n} f(x)$, search methods explore the parameter space *iteratively* by adjusting the search direction and the search range in every iteration in order to find lower values of the objective function. Search methods are generally classified into three groups based on their use of (1) objective function evaluations, (2) gradient of the objective function, and (3) Hessian of the objective function. There are several iterative search methods, usually called "line search methods" and designed to solve one-dimensional, *unimodal* unconstrained optimization problems [16]. Some of these methods can be analogously applied to multi-dimensional unconstrained problems. The *generic* pseudo-code for a line search method is given in Table 2.1, where the steps 2.1 and 2.2 will differ for a specific search method.

2.2.1 Gradient Descent Method

When the search direction is chosen as the gradient descent direction, $-\nabla f(x)$, the corresponding iterative search is called the method of gradient descent (also known as steepest descent or Cauchy's method). The direction of the negative gradient along which the objective function decreases fastest is the most natural choice. This simple algorithm for continuous optimization uses gradient of the objective function in addition to the function value itself, hence f must be a

Table 2.1 Pseudo-code for generic line search method



```
Generic Line Search Method (termination criteria: $|dx| = |x(k+1) - x(k)| = |\alpha(k)d(k)| < \varepsilon$)
1. Pick an initial starting point, $x(0) \in \Re^n$ at $k = 0$
2. Repeat:
    2.1. Calculate a search direction d(k) from x(k) such that it is a descent direction for f(x(k))
    2.2. Calculate a step size $\alpha(k)$ ensuring $f(x(k+1)) < f(x(k))$
    2.3. k ← k+1
    2.4. Update $x(k+1) = x(k) + \alpha(k)d(k)$
3. Until $x(k)$ converges to $x_{opt} \in \Re^n$ for which the 1st and 2nd order optimality conditions are satisfied.
```

differentiable function. The principle of the gradient descent algorithm can be obtained by setting the search direction as $\mathrm{d}(k) = -\nabla f(x)$ in step 2.1 and the *optimum* step size as $\alpha(k) = \underset{\alpha \in \Re^+}{\arg\min} f(x(k) - \alpha(k-1)\Delta f(x))$, in step 2.2 of the generic line search method, resulting in the position update as

$$x(k+1) = x(k) - \left(\underset{\alpha \in \Re^+}{\arg\min} f(x(k) - \alpha \nabla f(x))\right) \nabla f(x) \tag{2.4}$$

By using the optimum $\alpha(k)$ the gradient descent technique is guaranteed to converge to a local minimum from any starting point $x(0)$. Additionally, for the *exact* line search version of the algorithm described, it can be shown that the next step will be taken in the direction of the negative gradient at this new point and the step size will be chosen such that the successive search directions are orthogonal. In practice, there are inexact line search methods that use different criteria to find a suitable step size avoiding too long or too short steps to improve efficiency. The termination criterion is usually of the form $\|\nabla f(x)\| \leq \eta$ where η is small $(1e-6)$ and positive. However, the gradient method requires a large number of iterations for convergence when the Hessian of f near minima has a large condition number (linear dependence). The plots of the gradient descent method with the fixed $(\alpha(k) = 0.001)$ and the exact (optimal $\alpha(k)$) step size over Rosenbrock (banana) function,$f(x, y) = 100(y - x^2)^2 + (1 - x)^2$,are illustrated in Fig. 2.1. The total numbers of iterations for the corresponding plots are 26093 and 6423, respectively.

2.2.2 Newton–Raphson Method

Assuming the objective function $f(x)$ is a twice differentiable function, *Newton–Raphson* method is based on the second order Taylor series expansion of the function f around the point x:

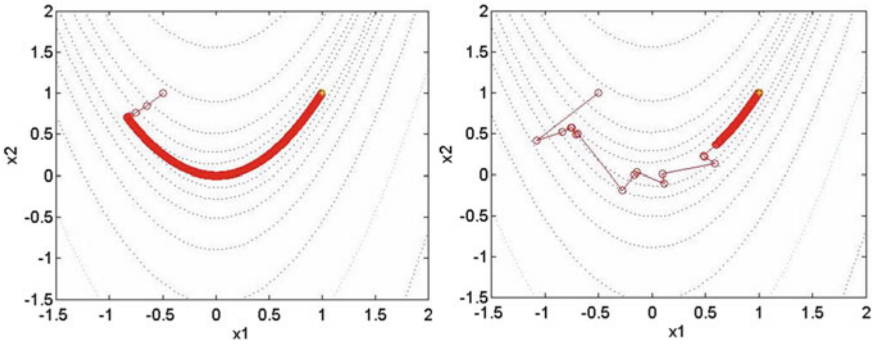


Fig. 2.1 Iterations of the fixed (*left*) and optimum $\alpha(k)$ (*right*) line search versions of gradient descent algorithm plotted over the Rosenbrock objective function, $x(0) = [-0.5, 1.0]$

$$f(x+\delta) \cong f(x) + \Delta f(x)\delta + \frac{1}{2}\delta^T H_f(x)\delta \tag{2.5}$$

where the Hessian matrix $H_f(x) = \nabla^2 f(x)$ is assumed to be positive definite near local minimum x^*. Therefore, *Newton–Raphson* method utilizes both the first and the second partial derivatives of the objective function to find its minimum. Similar to the gradient descent method, it can be implemented as an iterative line search algorithm using $\mathrm{d}(k) = -H_f(x)^{-1}\nabla f(x)$ as the search direction (Newton direction or the direction of the curvature) in 2.1 yielding the position update:

$$x(k+1) = x(k) - \alpha(k)H_f(x)^{-1}\nabla f(x) \tag{2.6}$$

At each iteration, *Newton–Raphson* method approximates the objective function by a quadratic function around $x(k)$ and moves toward its minima. In the original algorithm, the step size, $\alpha(k)$, is fixed to 1. While the convergence of *Newton–Raphson* method is fast in general, being quadratic near x^*, the computation and the storage of the inverse Hessian is costly. Quasi-Newton methods, which compute the search direction (through inverse Hessian approximation) with less computation can be alternatively employed. In the left plot in Fig. 2.2, iterations of the Quasi-Newton method over the Rosenbrock function are shown. Note that the convergence to the optimum point, (1, 1) is impressively faster than of gradient descent as shown in Fig. 2.1 (33 iterations versus 6,423 iterations with the optimal $F(x,y) = 2y^2 + x^2$). In the right plot, iterations of both gradient descent and Quasi-Newton methods over a quadratic objective function, $F(x,y) = 2y^2 + x^2$, are shown. It took 6 iterations for Quasi-Newton and 13 iterations for gradient descent to converge ($\alpha(0) = 0.05$ for both methods).

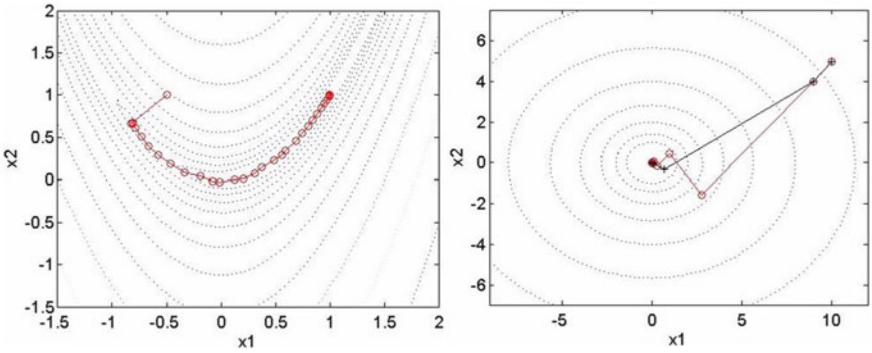


Fig. 2.2 (*Left*) Iterations of the Quasi-Newton method plotted over Rosenbrock function, $x_0 = [-0.5, 1.0]$. (*Right*) Iterations of the gradient descent (*red*) versus Quasi-Newton (*black*) methods plotted over a quadratic objective function, $x(0) = [10, 5]$

2.2.3 Nelder–Mead Search Method

Nelder–Mead or downhill simplex method [14], is a heuristic algorithm for multidimensional unconstrained optimization problems. The Nelder–Mead algorithm falls in the more general class of direct search algorithms which use only the function values, thus it depends on neither the first nor the second order gradients. This heuristic search method depends on the comparison of the objective function values at the $(n + 1)$ vertices of a general simplex, followed by replacement of the vertex with the highest function value by another point. It is, therefore, based on an iterative simplex search, keeping track of $n + 1$ points in n dimensions as vertices of a simplex (i.e., a triangle in 2 dimensions, a tetrahedron in 3 dimensions, and so on). It includes features, which enable the simplex to adapt to the local landscape of the cost function, i.e., at each iteration, the simplex moves toward the minimum by performing one of *reflection*, *expansion*, and *contraction* (in and out) operations. The stopping criterion is based on the standard deviation of the function value over the simplex. This is indeed a “greedy” method in the sense that the expansion point is kept if it improves the best function value in the current simplex. The convergence of a Nelder–Mead operation over the Rosenbrock’s function is shown in the left plot of Fig. 2.3 and the right plot demonstrates the consecutive simplex operations during iterations 3–30 where the total number of iterations is 94. Note that the four operations are annotated in the plot.

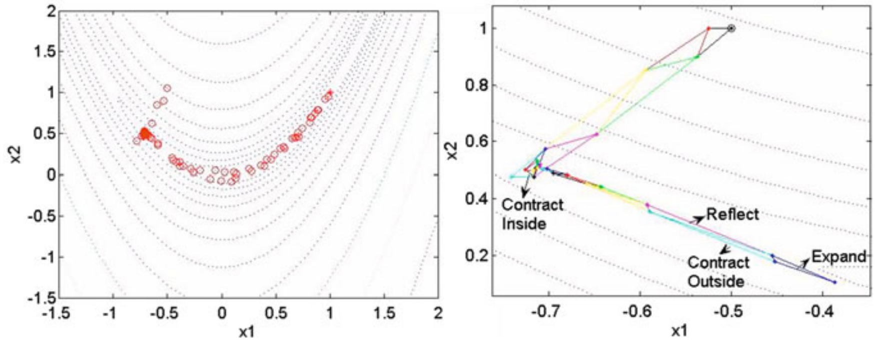


Fig. 2.3 *Left* Iterations of the Nelder–Mead method plotted over Rosenbrock function with $x(0) = [-0.5,\ 1.0]$. The vertices with the minimum function values are only plotted. *Right* consecutive simplex operations during the iterations 3–30

2.3 Stochastic Methods

2.3.1 Simulated Annealing

Metropolis in his groundbreaking paper, *Equation of State Calculations by Fast Computing Machines* in 1953 introduced the algorithm which simulates the evolution of a solid in a heat bath to thermal equilibrium. In physics, a thermal process for obtaining low energy states of a solid in a heat bath consists of the following two steps:

1. Increase the temperature of the heat bath to a maximum value at which the solid melts;
2. Slowly decrease the temperature until the particles arrange themselves in the ground state of the solid.

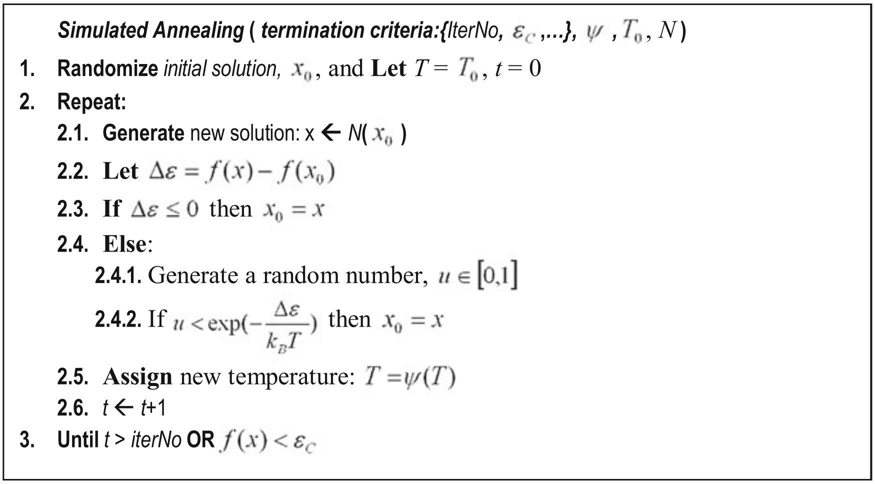
In the liquid phase, all particles are distributed randomly, whereas in the ground state of the solid the particles are arranged in a highly structured lattice, for which the corresponding energy is minimal. The ground state of the solid is obtained only if the maximum value of the temperature is sufficiently high and the cooling is performed slowly. Otherwise, the solid will be obtained in a meta-stable state rather than in the true ground state. This is the key for achieving the optimal ground state, which is the basis of the *annealing* as an optimization method. The *simulated annealing* method is a Monte Carlo-based technique and generates a sequence of states of the solid. Let i and j be the current and the subsequent state of the solid and ε_i and ε_j their energy levels. The state j is generated by applying a perturbation mechanism, which transforms the state i into j by a little distortion, such as a mere displacement of a particle. If $\varepsilon_j \le \varepsilon_i$, the state j is accepted as the current state; otherwise, the state j may still be accepted as the current state with a probability

$$P(i \Rightarrow j) = \exp\left(\frac{\varepsilon_i - \varepsilon_j}{k_B T}\right) \tag{2.7}$$

where k_B is a physical constant called the Boltzmann constant; recall from the earlier discussion that T is the temperature of the state. This rule of acceptance is known as *Metropolis* criterion. According to this rule, the *Metropolis–Hastings* algorithm generates a sequence of solutions to an optimization problem by assuming: (1) solutions of the optimization problem are equivalent to the state of this physical system, and (2) the cost (fitness) of the solution is equivalent to the energy of a state. Recall from the earlier discussion that the temperature is used as the control parameter that is gradually (iteratively) decreased during the process (*annealing*). Simulated annealing can thus be viewed as an iterative *Metropolis–Hastings* algorithm, executed with the gradually decreasing values of T. With a given cost (fitness) function, f, let ψ be the continuously decreasing temperature function, T_0 is the initial temperature, N is the neighborhood function, which changes the state (candidate solution) with respect to the previous state in an appropriate way, ε_C is the minimum fitness score aimed, and x is the variable to be optimized in N-D search space. Accordingly, the pseudo-code of the simulated annealing algorithm is given in Table 2.2.

Note that a typical characteristic of the simulated annealing is that it accepts deteriorations to a limited extent. Initially, at large values of temperature, T, large deteriorations may be accepted; as T gradually decreases, the amount of deteriorations possibly accepted goes down and finally, when the temperature reaches absolute zero, deteriorations cannot happen at all—only improvements. This is why it mimicks the family of steepest descent methods as T goes to zero. On the other hand, recall that simulated annealing and the family of Evolutionary

Table 2.2 Pseudo-code of the simulated annealing algorithm



```
Simulated Annealing ( termination criteria:{IterNo, $\varepsilon_C$,...}, $\psi$ , $T_0$, $N$ )
1. Randomize initial solution, $x_0$, and Let $T = T_0$, $t = 0$
2. Repeat:
    2.1. Generate new solution: x ← N( $x_0$ )
    2.2. Let $\Delta\varepsilon = f(x) - f(x_0)$
    2.3. If $\Delta\varepsilon \le 0$ then $x_0 = x$
    2.4. Else:
        2.4.1. Generate a random number, $u \in [0,1]$
        2.4.2. If $u < \exp(-\frac{\Delta\varepsilon}{k_B T})$ then $x_0 = x$
    2.5. Assign new temperature: $T = \psi(T)$
    2.6. t ← t+1
3. Until t > iterNo OR $f(x) < \varepsilon_C$
```

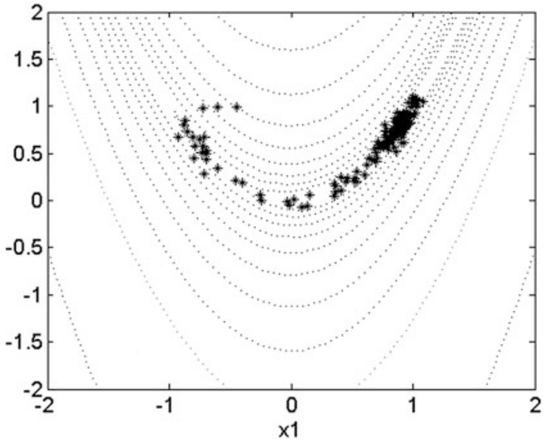


Fig. 2.4 The plot of 1,532 iterations of the simulated annealing method over Rosenbrock function with $x(0) = [-0.5, 1.0]$, $\varepsilon_C = 10^{-3}$, $T_0 = 1$, $\psi(T) = 0.95T$, $N(x_0) = x_0 + 0.01x_{\text{range}}r$ where $r \in N(0, 1)$, and x_{range} is the dimensional range, i.e., $x_{\text{range}} = 2 - (-2) = 4$

Algorithms (EAs) are sometimes called *meta-heuristics*, which make few or no assumptions about the problem being optimized and can thus search for the global optimum over a large set of candidate solutions. However, besides the population-based nature of EAs, this particular property is another major difference between them since EAs are all based on the "survival of the fittest" philosophy, whereas for the simulated annealing, worse solutions (generations in GA or particle positions in PSO) can still be "tolerated" for the sake of avoiding a local optimum.

Figure 2.4 shows the simulated annealing iterations plotted over the Rosenbrock function. The parameters and functions (for the temperature and neighborhood) used are: $\varepsilon_C = 10^{-3}$, $T_0 = 1$, $\psi(T) = 0.95T$, $N(x_0) = x_0 + 0.01x_{\text{range}}r$ where $r \in N(0, 1)$, and $x_{\text{range}}$is the dimensional range, i.e., $x_{\text{range}} = 2 - (-2) = 4$. Note that as $T \to 0$, it mimics the gradient descent method and hence took a longer time to converge to the global optimum.

2.3.2 Stochastic Approximation

Recall that the goal of deterministic optimization methods is to minimize a loss function $L : R^p \to R^1$, which is a differentiable function of θ and the minimum (or maximum) point θ^* corresponds to zero-gradient point, i.e.,

$$g(\theta) \equiv \frac{\partial L(\theta)}{\partial \theta}\bigg|_{\theta = \theta^*} = 0 \tag{2.8}$$

As mentioned earlier, in cases where more than one point satisfies this equation (e.g., a multi-modal problem), then such algorithms may only converge to a local minimum. Moreover, in many practical problems, the exact gradient value, g, is not readily available. This makes the stochastic approximation (SA) algorithms quite popular. The general SA takes the following form:

$$\widehat{\theta}_{k+1} = \widehat{\theta}_k - a_k \widehat{g}_k\left(\widehat{\theta}_k\right) \tag{2.9}$$

where $\widehat{g}_k\left(\widehat{\theta}_k\right)$ is the estimate of the gradient $g(\theta)$ at iteration k and $a_k$is a scalar gain sequence satisfying certain conditions. Unlike any steepest (gradient) descent method, SA assumes no direct knowledge of the gradient. To estimate the gradient, there are two common SA methods: finite difference stochastic approximation (FDSA) and simultaneous perturbation SA (SPSA) [17]. FDSA adopts the traditional Kiefer-Wolfowitz approach to approximate gradient vectors as a vector of p partial derivatives where p is the dimension of the loss (fitness) function. Accordingly, the estimate of the gradient can be expressed as follows:

$$\widehat{g}_k\left(\widehat{\theta}_k\right) = \begin{bmatrix} \frac{L\left(\widehat{\theta}_k + c_k\Delta_1\right) - L\left(\widehat{\theta}_k - c_k\Delta_1\right)}{2c_k} \\ \frac{L\left(\widehat{\theta}_k + c_k\Delta_2\right) - L\left(\widehat{\theta}_k - c_k\Delta_2\right)}{2c_k} \\ \cdot \\ \cdot \\ \cdot \\ \frac{L\left(\widehat{\theta}_k + c_k\Delta_p\right) - L\left(\widehat{\theta}_k - c_k\Delta_p\right)}{2c_k} \end{bmatrix} \tag{2.10}$$

where Δ_k is the unit vector with a 1 in the kth place and c_k is a small positive number that gradually decreases with k. Note that separate estimates are computed for each component of the gradient, which means that a p-dimensional problem requires at least $2p$ evaluations of the loss function per iteration. The convergence theory for the FDSA algorithm is similar to that for the root-finding SA algorithm of Robbins and Monro. These are: $a_k > 0, c_k > 0,\ \lim\limits_{k\to\infty} a_k = 0, \lim\limits_{k\to\infty} c_k = 0,$ $\sum_{k=0}^{\infty} a_k < \infty$ and $\sum_{k=0}^{\infty} a_k / c_k < \infty$. The selection of these gain sequences is critical to the performance of the FDSA. The common choice is the following:

$$a_k = \frac{a}{(k + A + 1)^{\upsilon}} \text{ and } c_k = \frac{c}{(k+1)^{\tau}}, \tag{2.11}$$

where a, c, υ and τ are strictly positive and $A \geq 0$. They are usually selected based on a combination of the theoretical restrictions above, trial-and-error numerical experimentation, and basic problem knowledge.

Figure 2.5 shows the FDSA iterations plotted over the Rosenbrock function with the following parameters: $a = 20$, $\mathrm{A} = 250$, $c = 1$, $\upsilon = 1$ and $\tau = 0.75$. Note that during the early iterations, it performs a random search due to its stochastic nature with large a_k, c_k values but then it mimics the gradient descent algorithm. The large number of iterations needed for the convergence is another commonality with the gradient descent.

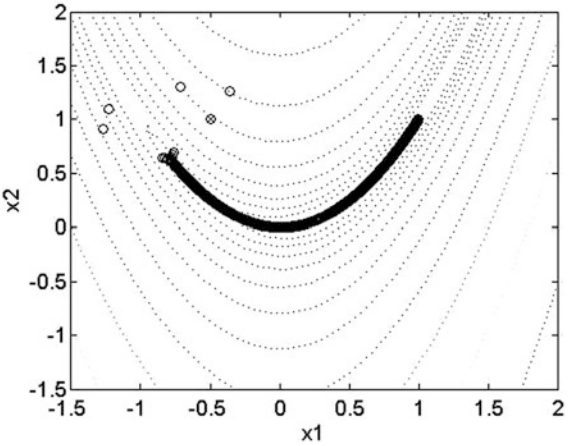


Fig. 2.5 The plot of 25,000 iterations of the FDSA method over Rosenbrock function with $x(0) = [-0.5, 1.0]$, $\varepsilon_C = 10^{-3}$, $a = 20$, $A = 250$, $c = 1$, $\upsilon = 1$, and $\tau = 0.75$

2.4 Evolutionary Algorithms

Among the family members of EAs, this section will particularly focus on GAs and DEs leaving out the details of both EP and ES while the next chapter will cover the details of PSO.

2.4.1 Genetic Algorithms

In nature, every living organism has a set of rules, a blueprint so to speak, describing how that organism is created (designed). The *genes* of an organism represent these rules and they are connected together into long strings called *chromosomes*. Each gene represents a specific property of the organism, such as eye or hair color and the collective set of gene settings are usually referred to as an organism's *genotype*. The physical expression of the genotype—the organism itself—is called the *phenotype*. The process of recombination occurs when two organisms mate and the genes are shared in the resultant offspring. In a rare occasion, a mutation occurs on a gene; however, this mutated gene will usually not affect the creation of the phenotype. Yet in rare cases, it will be expressed in the organism as a completely new trait. The ongoing cycle of natural selection, recombination, and mutation brought the evolution of the life on earth in addition to all such variations among the living organisms and of course their adaptation and survival instincts. The gene mutation plays a crucial role in the famous *Darwinian* rule of evolution, "the survival of the fittest." Genetic Algorithms (GAs) which are all inspired from the *Darwinian* evolution mimic all these natural evolutionary processes so as to search and find the optimum solution of the problem in hand.

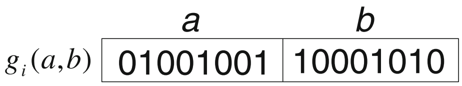


Fig. 2.6 A sample chromosome representation for the two problem variables, *a* and *b*

In order to accomplish this, the first step is the encoding of the problem variables into genes in a way of strings. This can be a string of real numbers but more typically a binary bit string (series of 0s and 1s). This is the genetic representation of a potential solution. For instance, consider the problem with two variables, a and $b \ni 0 \leq a,\ b < 256$. A sample chromosome representation for the ith chromosome, $g_i(a, b)$, is shown in Fig. 2.6 where both a and b are encoded with 8-bits, therefore, the chromosome contains 16 bits. Note that examining the chromosome string alone yields no information about the optimization problem. It is only with the decoding of the chromosome into its phenotypic (real) values that any meaning can be extracted for the representation. In this case, as described below, the GA search process will operate on these bits (chromosomes), rather than the real-valued variables themselves, except, of course, where real-valued chromosomes are used.

The second requirement is a proper fitness function which calculates the fitness score of any potential solution (the one encoded in the chromosome). This is indeed the function to be optimized by finding the optimum set of parameters of the system or the problem in hand. The fitness function is always problem dependent. In nature, this corresponds to the organism's ability to operate and to survive in its present environment. Thus, the objective function establishes the basis for the proper selection of certain organism pairs for mating during the reproduction phase. In other words, the probability of selection is proportional to the chromosome's fitness. The GA process will then operate according to the following steps:

1. **Initialization**: The initial population is created while all chromosomes are (usually) randomly generated so as to yield an entire range of possible solutions (the search space). Occasionally, the solutions may be "seeded" in areas where optimal solutions are likely to be found. The population size depends on the nature of the problem, but typically contains several hundreds of potential solutions encoded into chromosomes.
2. **Selection**: For each successive generation, first the selection of a certain proportion of the existing population is performed to breed a new generation. As mentioned earlier, the selection process is random, however, favors the chromosomes with higher fitness scores. Certain selection methods rate the fitness of each solution and preferentially select the best solutions.
3. **Reproduction**: For each successive generation, the second step is to generate the next generation chromosomes from those selected through genetic operators such as *crossover* and *mutation*. These genetic operators ultimately result in the child (next generation) population of chromosomes that is different from the initial generation but typically shares many of the characteristics of its parents.

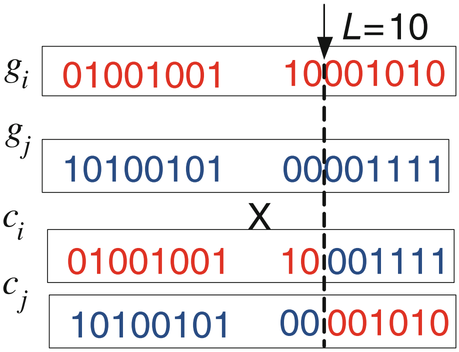


Fig. 2.7 A sample crossover operation over two chromosomes g_i and g_j after $L = 10$ bits. The resultant child chromosomes are c_i and c_j

4. **Evaluation**: The child chromosomes are first decoded and then evaluated using the fitness function and they replace the least-fit individuals in the population so as to keep the population size unchanged. This is the only link between the GA process and the problem domain.
5. **Termination**: During the **Evaluation** step, if any of the chromosomes achieves the objective fitness score or the maximum number of generations is reached, then the GA process is terminated. Otherwise, steps 2 to 4 are repeated to produce the next generation.

To perform a crossover operation, an integer position, *L*, is selected uniformly at random between 1 and the chromosome string length minus one, and the genetic information exchanged between the individuals about this point, then two new offspring strings are produced. A sample operation for $L = 10$ is shown in Fig. 2.7. The crossover operation is applied with a probability, *Px,* over the pairs chosen for breeding. Crossover is a critical operator in GA due to two reasons: it greatly accelerates the search early in the evolution of a population and it leads to effective combination of subsolutions on different chromosomes. There is always a trade-off when setting its value, i.e., assigning a too high *Px* may lead to premature convergence to a local optimum and a too low value may deteriorate the rate of convergence.

The other genetic operator, *mutation*, is then applied to certain genes (bits) of the child chromosomes with a probability, *Pm.* In the binary string representation, a mutation will cause a single bit to change its state, i.e., $0 \Rightarrow 1$ or $1 \Rightarrow 0$. Assigning a very low *Pm* leads to genetic drift (which is non-ergodic in nature) and the opposite may lead to loss of good solutions unless there is an elitist selection. Without a proper *Pm* setting, GAs may converge toward local optima or even some arbitrary (non-optimum) points rather than the global optimum of the search space. This indicates that a sufficiently high *Pm* setting should be assigned to teach the algorithm how to “sacrifice” a short-term fitness in order to gain a longer term fitness. In contrast to the binary GA, the real-valued GA uses real values in chromosomes without any encoding and thus the fitness score of each chromosome can be computed without decoding. This is a more straightforward, faster, and efficient scheme than the binary counterpart. However, both crossover and mutation operations might

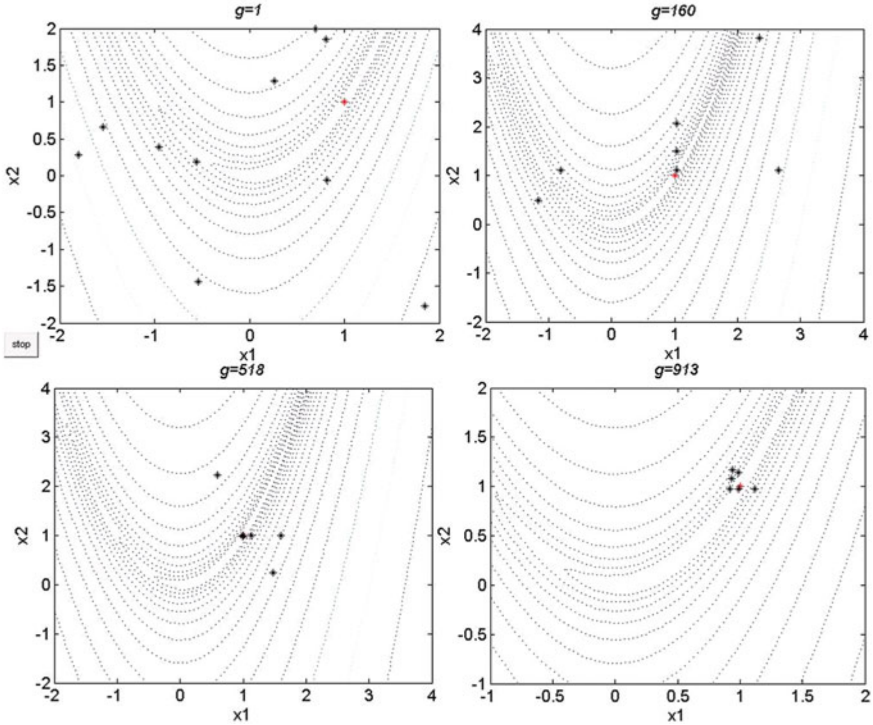


Fig. 2.8 The distributions of real-valued GA population for generations, $g = 1$, 160, 518, and 913 over Rosenbrock function with $S = 10$, $Px = 0.8$, and σ linearly decreases from x_{range} to 0

be different and there are numerous variants performing different approaches for each. A similar crossover operation as in binary GA can still be used (flipping with a probability, *Px,* over the pairs chosen for breeding). Another crossover operation common for real-valued GA is to exploit the idea of creating the child chromosome between parents via arithmetic recombination (linear interpolation), i.e., $z_i = \alpha$ $x_i + (1\text{–}\alpha)\, y_i$ where x_i, y_i ,and z_i are ith parent and child chromosomes, respectively, and $\alpha : 0 \leq \alpha \leq 1$. The parameter α can be a constant, or a variable changing according to some function or a random number. On the other hand, the most common mutation method is to shift by a random deviate, applied to each chromosome separately, taken from Gaussian distribution $N(0, \sigma)$ and then curtail it to the problem range. Note that the standard deviation, σ, controls the amount of shift. Figure 2.8 shows the distributions of a real-valued GA population for generations, $g = 1$, 160, 518, and 913 over Rosenbrock function with the parameter settings as, $S = 10, Px = 0.8$, and an adaptive σ linearly decreasing from x_{range} to 0, where x_{range} is the dimensional range, i.e., $x_{\mathrm{range}} = 2 - (-2) = 4$ for the problem shown in the figure. It took 160 generations for a GA chromosome to converge to the close vicinity of the optimum point, (1, 1).

2.4.2 Differential Evolution

Differential Evolution (DE) is an evolutionary algorithm, showing particular similarities to GA and hence can be called as a genetic-type method. DE has certain differences in that it is applicable to real-valued vectors, rather than bit-encoded strings. Accordingly, the ideas of mutation and crossover are substantially different. Particularly, the mutation operator is entirely different in a way that it is difficult to see why it is called mutation, except perhaps it serves the same purpose of avoiding early local trappings. DE has a notion of population similar to PSO rather than GA as its population members are called *agents* rather than *chromosomes*.

Suppose we optimize a real-valued (fitness) function in N-D, having N real variables. The ath agent in the population in the generation, g, represents the candidate solution of this function in the following array form: $x_a^g = \left[x_{a,1}^g, x_{a,2}^g, \ldots, x_{a,N}^g\right], \quad a \in [1, S]$ where S represents the size of the population. Then the DE process will follow the same path as GA except that the selection is performed after the reproduction, as follows:

1. **Initialization**: The initial population is created with $S > 3$. The range of each agent is defined for $g = 0$, i.e., $x_d^{\min} < x_{a,\,d}^0 < x_d^{\max}$and agent vector elements are randomly initialized within this range, $\left[x_d^{\min}, x_d^{\max}\right]$.
2. **Reproduction**: For each successive generation, $g = 1, 2, \ldots$, first mutation and then the crossover operators are applied on each agent's vector. To perform the mutation over the ath agent vector, x_a^g, three distinct agents, b, c, and d, are first randomly chosen such that $a \neq b \neq c \neq d$. This is why $S > 3$. The so-called donor vector for agent a is formed as follows:

$$y_a^{g+1} = x_b^g - \mathrm{Fr}\left(x_c^g - x_d^g\right) \tag{2.12}$$

where $r \sim U(0,1)$ is a random variable with a uniform distribution and F is a constant, usually assigned to 2. This is the mutation operation, which adds the weighted difference of the two of the vectors to the third, hence gives the name "differential" evolution. The following crossover operation then forms a trial vector from the elements of the agent vector, x_a^g, and the elements of the donor vector, y_a^{g+1}, each of which enters the trial vector with probability R.

$$u_{a,j}^{g+1} = \left\{ \begin{array}{lllll} y_{a,j}^{g+1} & \text{if} & r \leq R & \text{or} & j = \delta \\ x_{a,j}^g & \text{if} & r > R & \text{and} & j \neq \delta \end{array} \right\} \tag{2.13}$$

where $1 \leq \delta < N$ is a random integer ensuring that $u_{a,j}^{g+1} \neq x_{a,j}^g$. In other words, at least one element from the donor vector is ensured into the trial vector.

3. **Selection (with Evaluation)**: For each successive generation once the trial vector is generated, the agent vector, x_a^g, is compared with the trail vector, $u_{a,j}^{g+1}$, and the one with the better fitness is admitted to the next generation.

$$x_a^{g+1} = \left\{ \begin{array}{cc} u_a^{g+1} & \text{if } f\left(u_a^{g+1}\right) \le f\left(x_a^g\right) \\ x_a^g & \text{else} \end{array} \right\} \tag{2.14}$$

4. **Termination**: During the previous step, if any agent achieves the objective fitness score or the maximum number of generations is reached, then the DE process is terminated. Otherwise, steps 2 and 3 are repeated to produce the next generation.

Figure 2.9 illustrates the generation of the trial vector on a sample 2-D function. Note that the trial vector, u_a^{g+1}, gathers the first-dimensional element $\left(u_{a,1}^{g+1}\right)$ from the agent vector, $x_{a,1}^g$ and the second-dimensional element $\left(u_{a,2}^{g+1}\right)$ from the donor vector, $y_{a,2}^{g+1}$.

The choice of DE parameters F, S, and R can have a large impact on the optimization performance and how to select good parameters that yield good performance has therefore been subject to much research, e.g., see Price et al. [18] and Storn [19]. Figure 2.10 shows the distributions of DE population for generations, $g = 1, 20, 60$, and 86 over Rosenbrock function with the parameter settings as, $S = 10$, $F = 0.8$, and $R = 0.1$. Note that as early as 20th generations, a member of DE population already converged to the close vicinity of the optimum point, (1, 1).

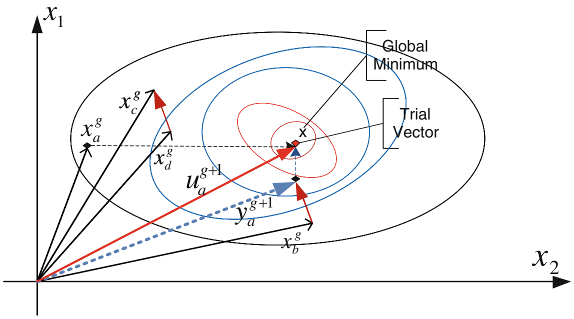


Fig. 2.9 A sample 2-D fitness function and the DE process forming the trial vector

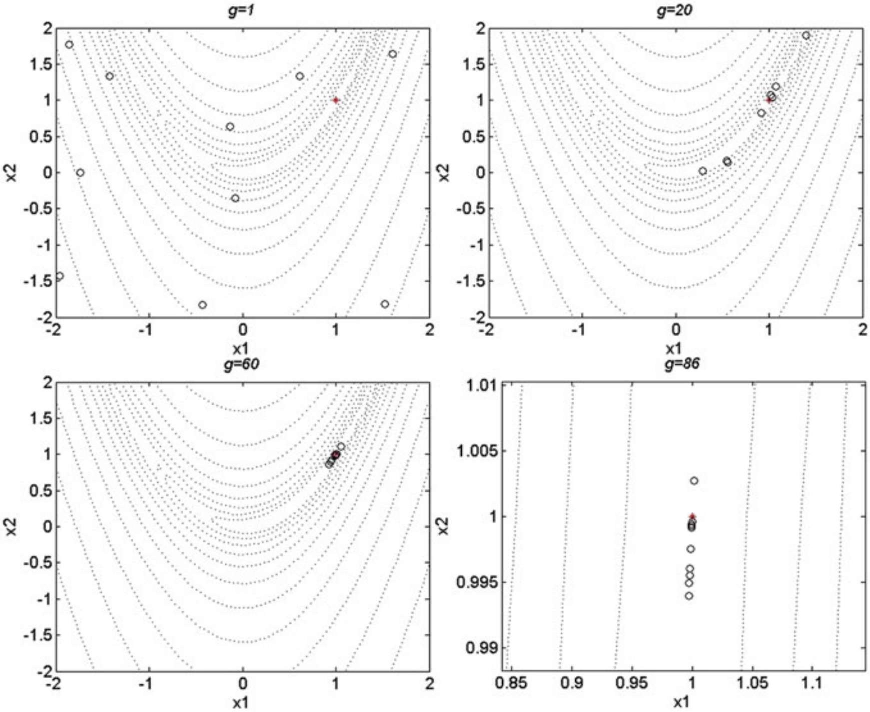


Fig. 2.10 The distributions of DE population for generations, $g = 1$, 20, 60, and 86 over Rosenbrock function with $S = 10$, $F = 0.8$ and $R = 0.1$

References

1. Biography in *Encyclopaedia Britannica.* http://www.britannica.com/eb/article-9062073/Pythagoras
2. C Field, *The Philosophy of Plato* (Oxford, 1956)
3. Biography in *Encyclopaedia Britannica.* http://www.britannica.com/eb/article-9108312/Aristotle
4. The MacTutor History of Mathematics archive. http://turnbull.dcs.st-and.ac.uk/history/
5. List of Greatest Mathematicians. http://fabpedigree.com/james/grmatm3.htm
6. T L Heath, *A History of Greek mathematics* II (Oxford, 1931)
7. I. Bulmer-Thomas, Selections illustrating the History of Greek mathematics II (London, 1941)
8. F. Woepcke, *Extrait du Fakhri, traité d'Algèbre par Abou Bekr Mohammed Ben Alhacan Alkarkhi* (1853)
9. J. F. Scott, *The Scientific Work of René Descartes* (1987)
10. H. Robbins, S. Monro, A stochastic approximation method. Ann. Math. Stat. **22**, 400–407 (1951)
11. J. Kiefer, J. Wolfowitz, Stochastic estimation of the maximum of a regression function. Ann. Math. Stat. **23**, 462–466 (1952)
12. J.C. Spall, Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. IEEE Trans. Autom. Control **37**, 332–341 (1992)

13. S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing. Science **220**, 671–680 (1983)
14. J.A. Nelder, R. Mead, A simplex method for function minimization. Comput. J. **7**, 308–313 (1965)
15. S. Boyd, L. Vandenberghe, *Convex Optimization* (Cambridge University Press, UK, 2004)
16. A. Antoniou, W.-S. Lu, *Practical Optimization, Algorithms and Engineering Applications* (Springer, USA, 2007)
17. R. Silipo, et al., ST-T segment change recognition using artificial neural networks and principal component analysis. Comput. Cardiol., 213–216, (1995)
18. K. Price, R. M. Storn, J. A. Lampinen, Differential Evolution: A Practical Approach to Global Optimization. Springer. ISBN 978-3-540-20950-8 (2005)
19. R. Storn, "On the usage of differential evolution for function optimization". Biennial Conference of the North American Fuzzy Information Processing Society (NAFIPS). pp. 519–523, (1996)

http://www.springer.com/978-3-642-37845-4

Multidimensional Particle Swarm Optimization for
Machine Learning and Pattern Recognition
Kiranyaz, S.; Ince, T.; Gabbouj, M.
2014, XXVIII, 321 p. 95 illus., 78 illus. in color.,
Hardcover
ISBN: 978-3-642-37845-4