Linear programming (LP) was first studied systematically by the mathematician Leonid Kantorovich (1939). It was one of the first applications of computers; the **simplex algorithm** (Dantzig, 1949) is still used despite worst-case exponential complexity. Karmarkar (1984) developed the far more efficient family of **interior-point** methods, which was shown to have polynomial complexity for the more general class of convex optimization problems by Nesterov and Nemirovski (1994). Excellent introductions to convex optimization are provided by Ben-Tal and Nemirovski (2001) and Boyd and Vandenberghe (2004).

Work by Sewall Wright (1931) on the concept of a **fitness landscape** was an important precursor to the development of genetic algorithms. In the 1950s, several statisticians, including Box (1957) and Friedman (1959), used evolutionary techniques for optimization problems, but it wasn't until Rechenberg (1965) introduced **evolution strategies** to solve optimization problems for airfoils that the approach gained popularity. In the 1960s and 1970s, John Holland (1975) championed genetic algorithms, both as a useful optimization tool and as a method to expand our understanding of adaptation (Holland, 1995).

The **artificial life** movement (Langton, 1995) took this idea one step further, viewing the products of genetic algorithms as *organisms* rather than just solutions to problems. The Baldwin effect discussed in the chapter was proposed roughly simultaneously by Conwy Lloyd Morgan (1896) and James Baldwin (1896). Computer simulations have helped to clarify its implications (Hinton and Nowlan, 1987; Ackley and Littman, 1991; Morgan and Griffiths, 2015). Smith and Szathmáry (1999), Ridley (2004), and Carroll (2007) provide general background on evolution.

Most comparisons of genetic algorithms to other approaches (especially stochastic hill climbing) have found that the genetic algorithms are slower to converge (O'Reilly and Oppacher, 1994; Mitchell *et al.*, 1996; Juels and Wattenberg, 1996; Baluja, 1997). Such findings are not universally popular within the GA community, but recent attempts within that community to understand population-based search as an approximate form of Bayesian learning (see Chapter 20) might help close the gap between the field and its critics (Pelikan *et al.*, 1999). The theory of **quadratic dynamical systems** may also explain the performance of GAs (Rabani *et al.*, 1998). There are some impressive practical applications of GAs, in areas as diverse as antenna design (Lohn *et al.*, 2001), computer-aided design (Renner and Ekart, 2003), climate models (Stanislawska *et al.*, 2015), medicine (Ghaheri *et al.*, 2015), and designing deep neural networks (Miikkulainen *et al.*, 2019).

The field of **genetic programming** is a subfield of genetic algorithms in which the representations are programs rather than bit strings. The programs are represented in the form of syntax trees, either in a standard programming language or in specially designed formats to represent electronic circuits, robot controllers, and so on. Crossover involves splicing together subtrees in such a way that the offspring are guaranteed to be well-formed expressions.

Interest in genetic programming was spurred by the work of John Koza (1992, 1994), but it goes back at least to early experiments with machine code by Friedberg (1958) and with finite-state automata by Fogel *et al.* (1966). As with genetic algorithms, there is debate about the effectiveness of the technique. Koza *et al.* (1999) describe experiments in the use of genetic programming to design circuit devices.

The journals *Evolutionary Computation* and *IEEE Transactions on Evolutionary Computation* cover evolutionary algorithms; articles are also found in *Complex Systems*, *Adaptive Behavior*, and *Artificial Life*. The main conference is the *Genetic and Evolutionary Com-*