8.1 Aims of this Chapter

The issue of setting the values of various parameters of an evolutionary algorithm is crucial for good performance. In this chapter we discuss how to do this, beginning with the issue of whether these values are best set in advance or are best changed during evolution. We provide a classification of different approaches based on a number of complementary features, and pay special attention to setting parameters on-the-fly. This has the potential of adjusting the algorithm to the problem while solving the problem.

This chapter differs from most in this book in that it presents rather more of a survey than a set of prescriptive details concerning how to implement an EA for a particular type of problem. For this reason, rather than end with one or two example applications, we have chosen to interleave a number of examples throughout the text. Thus we hope to both clarify the points we wish to raise as we present them, and also to give the reader a feel for some of the many possibilities available for controlling different parameters.

8.2 Introduction

The previous chapters presented a number of evolutionary algorithms. The description of a specific EA contains its components, thereby setting a framework while still leaving quite a few items undefined. For instance, a simple GA might be given by stating it will use binary representation, uniform crossover, bit-flip mutation, tournament selection, and generational replacement. For a full specification, however, further details have to be given, for instance, the population size, the probability of mutation $p_m$ and crossover $p_c$, and the tournament size. These data – called the algorithm parameters or strategy parameters – complete the definition of the EA and are necessary to produce an executable version. The values of these parameters greatly determine whether the algorithm will find an optimal or near-optimal solution, and
whether it will find such a solution efficiently. Choosing the right parameter values is, however, a hard task.

Globally, we distinguish two major forms of setting parameter values: **parameter tuning** and **parameter control**. By parameter tuning we mean the commonly practised approach that amounts to finding good values for the parameters before the run of the algorithm and then running the algorithm using these values, which remain fixed during the run. Later on in this section we give arguments that any static set of parameters having the values fixed during an EA run seems to be inappropriate. Parameter control forms an alternative, as it amounts to starting a run with initial parameter values that are changed during the run.

Parameter tuning is a typical approach to algorithm design. Such tuning is done by experimenting with different values and selecting the ones that give the best results on the test problems at hand. However, the number of possible parameters and their different values means that this is a very time-consuming activity. Considering four parameters and five values for each of them, one has to test $5^4 = 625$ different setups. Performing 100 independent runs with each setup, this implies 62,500 runs just to establish a good algorithm design.

The technical drawbacks to parameter tuning based on experimentation can be summarised as follows:

- Parameters are not independent, but trying all different combinations systematically is practically impossible.
- The process of parameter tuning is time consuming, even if parameters are optimised one by one, regardless of their interactions.
- For a given problem the selected parameter values are not necessarily optimal, even if the effort made for setting them was significant.

This picture becomes even more discouraging if one is after a “generally good” setup that would perform well on a range of problems or problem instances. During the history of EAs considerable effort has been spent on finding parameter values (for a given type of EA, such as GAs), that were good for a number of test problems. A well-known early example is that of [98], determining recommended values for the probabilities of single-point crossover and bit mutation on what is now called the DeJong test suite of five functions. About this and similar attempts [181, 334], it should be noted that genetic algorithms used to be seen as robust problem solvers that exhibit approximately the same performance over a wide range of problems [172, page 6]. The contemporary view on EAs, however, acknowledges that specific problems (problem types) require specific EA setups for satisfactory performance [26]. Thus, the scope of “optimal” parameter settings is necessarily narrow. There are also theoretical arguments that any quest for generally good EA, thus generally good parameter settings, is lost a priori, cf. the discussion of the No Free Lunch theorem [430] in Chap. 11.

To elucidate another drawback of the parameter tuning approach recall how we defined it: finding good values for the parameters before the run of the