12 Evolutionary computer programming

12.1 Reasons for the necessity of self-evolving computer programs

Nature offers a tremendous amount of extremely complicated problems which cannot easily be rationalized and resolved. Probably one has to accept that there are scientific and technological problems too complex to be directly rationalized by humans. A well-known example is the non-periodical movement of many gravitationally interacting bodies in space. Since we are not able to imagine their motions with a sufficient degree of perfection, we call it "chaos". Unfortunately, humans obviously have significant intellectual difficulties to find theoretical descriptions or models for phenomena which they do not comprehend. Similarly, as an ape cannot write a mathematical equation beyond its intellectual abilities, humans are not directly able to establish mathematical structures beyond their intellectual limits. However, further progression of science and technology urgently requires to overcome such limits.

One well-known example for a complicated problem is the so-called folding paradox (see, e.g., Nölting, 2005): how can a protein find its unique native conformation among the ~10^{30}–10^{200} possible conformations of an average small protein in the unfolded state? One of the major difficulties of complex phenomena like protein folding is often the lack of an efficient and solvable mathematical description. Often one is able to write down some equations which describe the physics of the system while not being able to solve them.

This chapter describes a method which can potentially provide solutions beyond current human intelligence: in order to overcome the limits of rational design, one lets a computer program evolve itself. The method is exemplarily applied on protein folding and structure predictions (Nölting et al., 2004) and the optimization of optical effects of nanoparticle arrays. In Sect. 12.5 the much wider scope of this method is discussed.

12.2 General features of the method

Figs. 12.1 and 12.3 show the principle of operation of the method of self-evolving programs. A so-called wild-type computer program is evolved towards higher efficiency by mutagenesis and selection in a similar way as species evolve in

nature: First a number of mutants of the wild-type program is created. The performance of the mutants is then tested and the best performing program serves

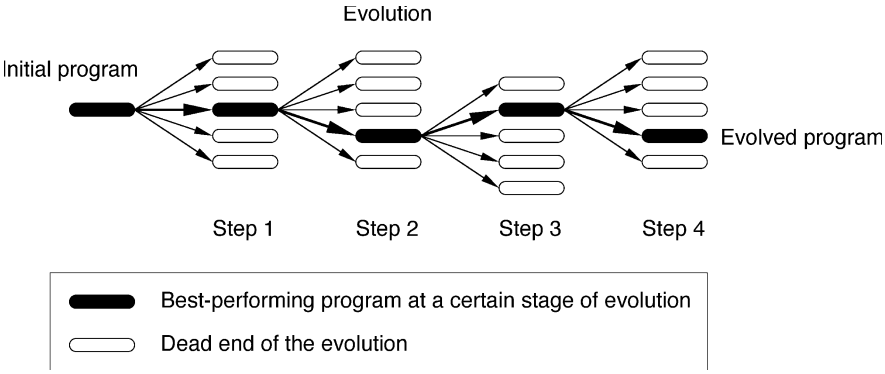

Fig. 12.1 Scheme of the method of evolutionary computer programming. The best performing program at a certain stage of the evolution is shown as a filled rectangle. Open rectangles indicate less-performing mutants which are usually sorted out. Essentially, the method is based on the mutation and selection of computer programs similarly as for species in the biological evolution. In this way, an initial program which contains mutatable parts becomes highly optimized in the course of successive rounds of evolution

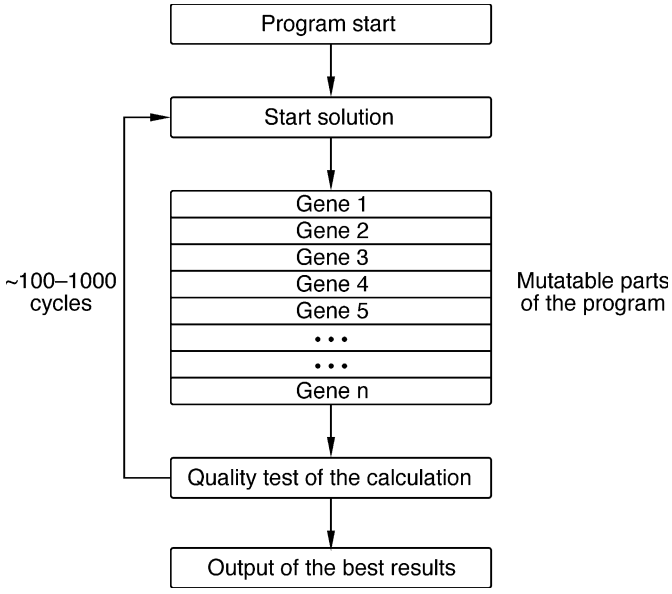

Fig. 12.2 A suitable structure of a program for self-evolution. The genes are the mutatable parts of the program. The genes improve the start solution of the given task