11

Nonstationary and Noisy Function Optimisation

Unlike most of the examples we have used so far, real-world environments typically contain sources of uncertainty. This means that if we measure the fitness of a solution more than once, we will not always get the same result. Of course, biological evolution happens in just such a dynamic environment, but there are also many EA applications in environments featuring change or noise when solutions are evaluated. In these **nonstationary** situations the search algorithm has to be designed so that it can compensate for the unpredictable environment by monitoring its performance and altering some aspects of its behaviour. An objective of the resulting adaptation is not to find a single optimum, but rather to select a sequence of values over time that maximise or minimise some measure of the evaluations, such as the average or worst. This chapter discusses the various sources of unpredictability, and describes the principal adaptations to the basic EA in response to them.

11.1 Characterisation of Nonstationary Problems

At this stage we must consider some basic facts about the process of going from a representation of a solution (genotype) x, to measuring the quality of a candidate solution for the task at hand, $f(x)$. For illustration we will use a simple household example — designing a mop for cleaning spills of various different liquids from a floor. We will assume that the candidate solution in fact describes the structure of the sponge — that is to say the size of pores, elasticity, shape of contact area, etc.[1] The quality recorded for a given solution may be unpredictable for one or more of the following reasons.

[1] In general, δq stands for a small random change in the value of some property q.

A.E. Eiben, J.E. Smith, *Introduction to Evolutionary Computing*,
Natural Computing Series, DOI 10.1007/978-3-662-44874-8_11

The Genotype to Phenotype Mapping Is Not Exact and One-to-One

If the fitness is to be measured via simulations, the genotype may use double-precision floating point numbers to code for the design parameters, but in the simulation there might be differences in the resolution of the models. If the fitness is to be measured via physical effects, the manufactured phenotype may not perfectly reflect the encoded design parameters. Thus, for our example, we might be measuring the cleaning ability of a slightly different sponge to the one we thought. In terms of a search landscape, the fitness observed may be that of a point in the region of x: $f_{observed}(x) = f(x + \delta x)$.

The Act of Measurement Itself Is Prone to Error or Uncertainty

This might arise from, for example, human error, small random fluctuations in shape of physical objects as their molecules vibrate, randomness in the movement of electrons through a sensor or wire, or the collected randomness of more complex organisms such as people, markets, packets on computer network or physical traffic. In terms of our mop, we might mismeasure the quantities of fluid absorbed by the sponge. This means a rethink of our landscape metaphor: the unique surface in the altitude dimension representing the fitness of points in the search space is replaced by a 'cloud' — a probability distribution from which we sample when we measure fitness, whose 'thickness' may vary across the space. Many different models may be used to characterise the noise. The most straightforward is to break down a quality function into two components: $f_{observed}(x) = f_{mean}(x) + f_{noise}(x)$. Here the first component represents the average that we would find if we measured fitness many times, and the second noise component is typically modelled as a random drawing from a normal distribution $N(0, \sigma)$.

The Environment Changes Over Time

This may be because the external environment is inherently volatile, or it may be that the very act of evaluating solutions affects subsequent fitness. For example, in an interactive EA, each interaction potentially increases user fatigue and changes the expectations (Sect. 14.1). If our mop is being tested in an environment with significant seasonal fluctuations in temperature, then this may affect either the absorbency of the sponge material, or the viscosity of fluids tested. This could mean that if we measured the same design every day for a year we would observe seasonal cyclic changes in the fitness. In the context of a search landscape, this means that the locations of the optima are now time-dependent, i.e., $f_{observed}(x) = f(x, t)$.

In many real-world problems, one or more of these effects occur in combination. It remains for the algorithm designer to decide which will be present, take account of the context in which the tool created will be used, and select appropriate modifications from those listed in subsequent sections.

11.2 The Effect of Different Sources of Uncertainty

A number of researchers have proposed mechanisms for dealing with uncertainty, and examined their performance on test cases and real-world problems. Algorithms are typically compared by running them for a fixed period and calculating two time-averaged metrics, which correspond to different types of real-world applications.

The first of these is the online measure [102] and is simply the average of all calls to the evaluation function during the run of the algorithm. This measure relates to applications where it is desirable to maintain consistently good solutions, e.g., online process control [164, 444] or financial trading. The second metric considered is the offline performance and is the time-averaged value of the best-performing member of the current population. Unlike the online metric, offline performance is unaffected by the occasional generation of individuals with very low fitness, and so is more suitable for problems where the testing of such individuals is not penalised, e.g., parameter optimisation using a changing design model.

If we use the time-dependent notation for the fitness function as $f(x,t)$ and denote the best individual in a population $P(t)$ at time t by $best(P(t))$, then we can formalise the two metrics over a period T as follows:

$$online = \frac{1}{T} \times \sum_{t=1}^{T} \frac{1}{|P(t)|} \sum_{x \in P(t)} f(x,t)),$$

$$offline = \frac{1}{T} \times \sum_{t=1}^{T} f((best(P(t)),t)).$$

Finally, let us note that in some cases it may be appropriate to consider both metrics in a multiobjective approach, since optimising the mean fitness may be the principle desiderata, but evaluating low-fitness solutions might be catastrophic. In this case one approach might be to use a surrogate model to screen out such potential fatal errors.

The three different sources of uncertainty identified in the previous section affect the performance of the EA in different ways. Considering errors in the genotype–phenotype mapping, we can note that using the average of n repeated fitness evaluations $\frac{1}{n} \times \sum_n f(x + \delta x)$ to determine the fitness of any given x means using a sample of n points from the neighbourhood around x. However, the sampled neighbourhoods of adjacent solutions can overlap, that is, $x + \delta x$ can coincide with $y + \delta y$. Hence, fine-grained features of the fitness landscape will be smoothened out possibly removing local optima in the process. In practice this is often a good thing. From the search perspective it creates gradients around steps and plateaus in the landscape, and from a problem-solving perspective it reduces the attractiveness of high-quality solutions that are surrounded by much lower quality neighbours, which might be considered very 'brittle'.

Considering noise in the act of measurement itself, the average of n repeated measurements is $f(x) + \frac{1}{n} \times \sum_n N(0, \sigma)$ and the second term will approximate zero as n is increased. In other words, when repeatedly sampling from random noise, the deviations cancel out and you get an estimate of the mean. Unlike the case above, there is no smoothing of landscape features. This is illustrated in Figure 11.1, which shows a fitness function $f(x) = 1/(0.1 + x^2)$ and the values estimated after five samples with two different sorts of uncertainty present. Both types of noise were drawn uniformly from a distribution between +/− 0.4. Already becoming apparent is that the errors in the genotype–phenotype mapping reduce the height of the estimated local optimum and make it wider. In contrast, the effect of noise in the measurement alone is already being reduced to near-zero after five samples.

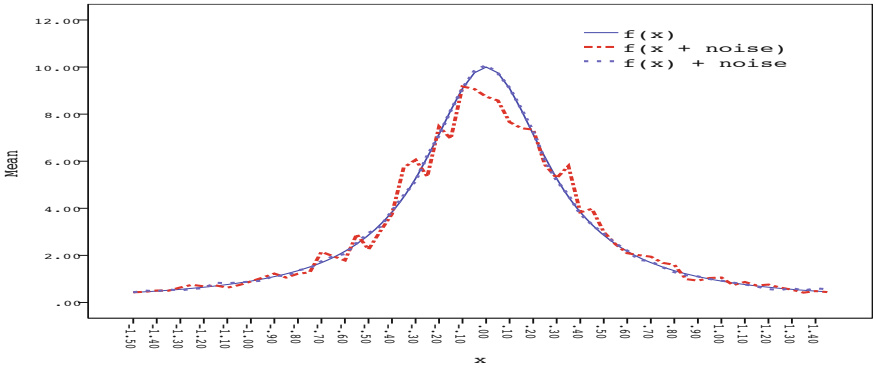

Fig. 11.1. Effect of different types of uncertainty on estimated fitness. Curves show mean values estimated after five samples for each value of x

Regarding the third situation of nonstationary fitness functions, Cobb [84] defines two ways of classifying these:

- Switching versus continuous, which is based on time-scale of change with respect to the rate of evaluation — the former providing sudden and the latter more gradual changes. Continuous changes might be cyclical (e.g., related to seasonal effects) or reflect a more uniform movement of landscape features (for example, when arising from gradual wear-and-tear of physical parts).
- Markovian versus state dependent. In the first of these, the environment at the next time step is purely derived from the current one, whereas in the latter far more complex dynamics may play out.

To illustrate these differences, consider a simple example many of us encounter daily – modelling and predicting traffic levels on a commute. These will change gradually during each day, building up to, and then tailing off

from, peak values at rush hours. When planning a journey we would expect that the likely travel time of a direct route involving a busy road will vary during the day. However, there will also be more switching behaviour as the flows are affected by one-off events such as public holidays, and between different periods — for example the amount of variation and overall levels of traffic are often less during school holidays. On these days we might happily decide take the most direct route. Considering ten-minute intervals over a single day in a city, we might view traffic levels as driven by some Markovian process — the value at the next period depends on traffic levels now, but not prior to that, since the cause is the aggregated behaviour of lots of independent people, mostly travelling to, from, and in the course of work. However, if we turn our attention to modelling air traffic, the situation is still clearly time-varying, but is highly state-dependent as disruption to an airline's schedule in one place can have enormous knock-on effects due to planes being in the wrong place, etc.

11.3 Algorithmic Approaches

11.3.1 Approaches That Increase Robustness or Reduce Noise

The only viable approach for reducing the effect of noise, whether in the fitness function evaluation, or in the genotype-to-phenotype mapping, is to repeatedly re-evaluate solutions and take an average. This might be done either explicitly, or (as described further below) implicitly via the population management processes of parent and survivor selection.

The principle question that arises in explicit approaches is, how many times should the fitness be sampled? Bearing in mind that EAs naturally contain some randomness in their processes anyway, the key issue from the perspective of evolution is being able to reliably distinguish between good and bad members of the population. Thus, a common approach is to monitor the degree of variation present, and resample when this is greater than the range of estimated fitnesses in the population. This reasoning suggests that the rate of resampling would increase as the population converges towards high-quality solutions.

When calculating how large a sample to take, there is also the law of diminishing returns: in general, the standard deviation observed decreases only as fast as the square root of the number of measurements taken.

Finally, it is worth mentioning that it is often worth the extra book-keeping of making resampling decisions independently for each solution, since the amount of noise will often not be uniform across the search space.

11.3.2 Pure Evolutionary Approaches to Dynamic Environments

The distributed nature of the genetic search provides a natural source of power for exploring in changing environments. As long as sufficient diversity

remains in the population, the EA can respond to a changing search landscape by reallocating future trials. However, the tendency of EAs, especially of GAs, to converge rapidly results in the population becoming homogeneous, which reduces the ability of the EA to identify regions of the search space that might become more attractive as the environment changes. In such cases it is necessary to complement the standard EA with a mechanism for maintaining a healthy exploration of the search space. (Recall the self-adaptation example from Sect. 4.4.2.)

In [84] the behaviour of a standard GA on a parabolic function with the optima moving sinusoidally in space was observed. This was done for a range of bitwise mutation rates. It was found that the offline performance decreased as the rate of change increased, for all mutation probabilities. As the rate of change increased, the mutation rate that gave optimal offline performance increased. Finally, it was noted that as problem difficulty increased, the rate of change that GA could track decreased.

In the light of these findings, various approaches have been proposed that are aimed at responding to different types of environmental change.

11.3.3 Memory-Based Approaches for Switching or Cyclic Environments

The first strategy expands the memory of the EA in order to build up a repertoire of ready responses for various environmental conditions. The main examples of this approach are the GA with diploid representation [194] and the structured GA [94]. Goldberg and Smith examined the use of diploid representation and dominance operators to improve performance of an EA in an oscillating environment [402], while Dasgupta and McGregor presented a modified "structured GA" with a multilayered structure of the chromosome which constitutes a "long-term distributed memory".

11.3.4 Explicitly Increasing Diversity in Dynamic Environments

The second modification strategy effectively increases diversity in the population directly (i.e., without extending the EA memory) in order to compensate for changes encountered in the environment. Examples of this strategy involve the GA with a hypermutation operator [84, 85], the random immigrants GA [199], the GA with a variable local search (VLS) operator [443, 444], and the thermodynamic GA [306].

The hypermutation operator temporarily increases the mutation rate to a high value, called the hypermutation rate, during periods when the time-averaged best performance of the EA worsens. In his 1992 study, Grefenstette noted that under certain conditions hypermutation might never get triggered [199].

The random immigrants mechanism replaces a fraction of a standard GA's population by randomly generated individuals in each generation in order to

maintain a continuous level of exploration of the search space. It was reported that 30% replacement gave the best off-line tracking: if the value is too high the algorithm is unable to converge between changes; however, off-line performance decreases with proportion replaced.

In an extensive study, Cobb and Grefenstette compared hypermutation with random immigrants and simple GA (with high mutation rate) [85]. They noted that there was a qualitative difference in the nature of the mutation operator in the three algorithms:

- Simple Genetic Algorithm (SGA) – uniform in population and time
- Hypermutation – uniform in population, not in time
- Random immigrants – uniform in time, not in population

They used two landscapes, and three types of change: a linear motion in the first problem (moving 1 step along an axis every 2 or 5 generations), randomly shifting the optima in the first problem every 20 generations, and swapping between the two problems every 2 or 20 generations. Their findings were:

- SGA: A high mutation probability of 0.1 was reasonably good at the translation tasks, but gave very poor online performance. It was unable to track the steadily moving optimum or oscillation. In general, the mutation probability needs to be matched to the degree of change.
- Hypermutation: High variances in performance were noted, and the higher mutation rate needed careful tuning to the problem instance. It was much better at tracking sudden changes than SGA and gave better online performance than SGA or random immigrants when the rate of change was slow enough to allow a lower rate of mutation.
- Random Immigrants: This strategy was not very good at tracking linear movement, but was the best at the oscillating task. They hypothesised that this was because it allowed the preservation of niches. The strategy displayed poor performance on stationary and slowly changing problems.

The VLS operator uses a similar triggering mechanism to hypermutation, and it enables local search around the location of the population members before the environmental change. The range of the search is gradually extended using a heuristic that attempts to match the degree of change.

The thermodynamic GA can maintain a given level of diversity in population by evaluating the entropy and free energy of the GA's population. The free energy function is effectively used to control selection pressure during the process of creating a new population.

11.3.5 Preserving Diversity and Resampling: Modifying Selection and Replacement Policies

In [441, 442] the suitability of generational GAs (GGAs) and steady-state GAs (SSGAs) was studied for use in dynamic environments. Results showed

that the SSGA with a "delete-oldest" replacement strategy can adapt to environmental changes with reduced degradation of offline and particularly online performance. The improved performance of the SSGA can be explained by the fact that an offspring is immediately used as a part of the mating pool, making a shift towards the optimal solution possible in a relatively early phase of search. The authors concluded that the steady-state model was better suited tor use in nonstationary environments, particularly for on-line applications.

Selection is a vital force in any evolutionary algorithm, and an understanding of the nature of its effects is necessary if effective algorithms are to be developed. For GGAs selection has been well studied, and methods have been developed that reduce much of the noise inherent in the stochastic algorithm, e.g., SUS [32]. Unfortunately, the very nature of SSGAs precludes the use of such methods and those available are inherently more noisy.

In [400] a Markov chain analysis of the takeover probability versus time was used to investigate sources of noise in several replacement strategies. Some variations in performance arise from losing the only copy of the current best in the population, which happened approximately 50% of the time for delete random, and 10% of the time for delete-oldest. Performance comparisons on static landscapes demonstrated that the extent to which this affects the quality of the solutions obtained depends on the ability of the reproductive operators to rediscover the lost points. In [78] other strategies, e.g., deletion by exponential ranking, were also shown to lose the optimum.

A common way of avoiding this problem is to incorporate elitism, often in the form of a delete-worst strategy. Chakraborty [78, 79] showed that this exhibits increased selection pressure, which can lead to premature convergence and poor performance on higher dimensional problems.

In [399] a number of replacement strategies were compared in combination with two different ways of achieving elitism. The first was the common method described in Section 5.3.2, and the elite member can either be preserved with its original fitness value, or be reevaluated and the new fitness value saved. The second, "conservative selection" is an implicit mechanism introduced in [444]. Here each parent was selected by a binary tournament between a randomly selected member of the population and the member about to be replaced. If the latter is the current best, then it will win both tournaments, so recombination will have no effect, and (apart from the effects of mutation) elitism is attained. In [400] this was shown to guarantee takeover by the optimal class, but at a much slower rate than delete-worst or elitist delete-oldest. In total, ten selection strategies were evaluated for their online and offline performance on two different test problems. Deletion of the oldest, worst, and random members was done in conjunction with both standard and conservative tournaments. Additionally, a delete-oldest policy was tested with four variants of elitism. These were:

1. Oldest is kept if it is one of the current best, but is re-evaluated.
2. Oldest is kept if it is the sole copy of the current best and is re-evaluated.

3. As 1, but without re-evaluation (original fitness value kept).
4. As 2, but without re-evaluation (original fitness value kept).

This was done for algorithms with and without hypermutation on two different classes of problems. The results obtained clearly confirmed that for some algorithms an extra method for creating diversity (in this case, hypermutation) can improve tracking performance, although not all of the strategies tested were able to take advantage of this. However, two factors are immediately apparent from these results which hold with or without hypermutation.

Exploitation: Strategies such as delete-oldest or delete-random, which can lose the sole copy of the current population best, performed poorly. This matched the theoretical analysis and results on static landscapes noted above. Therefore some form of elitism is desirable.

Reevaluation: In potentially dynamic environments it is essential that the fitness of points on the landscape is continuously and systematically reevaluated. Failure to do so leads to two effects. First, the population can get 'dragged back' to the original peak position, as solutions near there are selected to be parents on the basis of out-of-date information. Second, it can also lead to a failure to trigger the hypermutation mechanism. Although this was obvious for the third and fourth variants of elitism tested, it also applies to the much more common delete-worst policy. In this case if the population had converged close to the optimum prior to the change, the worst members that get deleted may be the only ones with a true fitness value attached. The importance of systematic reevaluation was clear from the difference between conservative delete-oldest and conservative delete-random. The former always produced better performance than the latter, and very significantly so when hypermutation was present.

Of all the policies tested, the conservative delete-oldest was the best suited to the points noted above and produced the best performance. The improvement over the elitist policy with reevaluation is believed to result not merely from the reduced selection pressure, but from the fact that the exploitation of good individuals is not limited to preserving the very best, but will also apply (with decreasing probability) to the second-best member, and so on. Since the implicit elitism still allows changes via mutation, there is a higher probability of local search around individuals of high fitness, while worse members are less likely to win tournaments, and so they are replaced with offspring created by recombination. The result is that even without hypermutation the algorithm was able to track environmental changes of modest size.

11.3.6 Example Application: Time-Varying Knapsack Problem

This problem is a variant of that described in [306]. As discussed in Sect. 3.4.2, we have a number of items each having a value (v_i^t) and a weight or cost (c_i^t) associated with them, and the problem is to select a subset that maximises the sum of the elements' values while meeting a (time-varying) total capacity constraint $C(t)$.

In [399], Smith and Vavak outline a series of experiments on this problem aimed at investigating the effect of different survivor selection policies. In the particular case investigated, the values v_i and costs c_i attached to the items remained constant, but the capacity constraint $C(t)$ alternated between 50%, 30%, and 80% of C_{sum}, changing once every 20,000 evaluations.

The algorithm used was a binary-coded SSGA with 100 members. Parent selection was by binary tournaments, with the fitter member always selected. In some cases the conservative tournament selection operator was used. Uniform crossover was used (with probability 1.0) to generate offspring, as this shows no positional bias (Sect. 16.1). The rest of the parameter settings were decided after some initial experimentation to establish robust values.

The hypermutation operator was implemented as it is currently the most commonly used method for tracking. It was triggered if the running average of the best performing members of the population over an equivalent of three generations of the generational GA (in this case, 300 evaluations) drops by an amount that exceeds a predefined threshold. In this case a value of threshold TH=3 was used. The best performing member of the population was re-evaluated for 100 evaluations. Once it had been triggered, the hypermutation rate (0.2) was switched back to the baseline mutation rate (0.001) as soon as the best performing member of the population reached 80% of its value before the environmental change occurred. The setting of the parameters (80% and hypermutation rate 0.2) was found to provide good results for the given problem. A prolonged period of high mutation for values higher than 80% has a negative effect on on-line performance because diversity is introduced into the population despite the correct region of the search space having already been identified. Similarly to the choice of the threshold level described previously, the values of both parameters were selected empirically.

As hinted above, the best results came from the combination of conservative tournaments for parent selection policy, with a delete-oldest policy. Here each member has a fixed lifespan, but when its turn comes to be deleted it enters the tournament to be a parent of the offspring that will replace it. The algorithm using this policy, along with hypermutation, was able to successfully track the global optimum in both a switching environment, as here, and also in a problem with a continuously moving optimum.

For exercises and recommended reading for this chapter, please visit www.evolutionarycomputation.org.