Taming the Complexity of Natural and Artificial Evolutionary Dynamics

Riccardo Poli and Christopher R. Stephens

Abstract The study of complex adaptive systems is among the key modern tasks in science. Such systems show radically different behaviours at different scales and in different environments, and mathematical modelling of such emergent behaviour is very difficult, even at the conceptual level. We require a new methodology to study and understand complex, emergent macroscopic phenomena. Coarse graining, a technique that originated in statistical physics, involves taking a system with many microscopic degrees of freedom and finding an appropriate subset of collective variables that offer a compact, computationally feasible description of the system, in terms of which the dynamics looks "natural". This paper presents the key ideas of the approach and shows how it can be applied to evolutionary dynamics.

1 Introduction

Our understanding of evolution has itself evolved. The journey started with Darwin and Mendel but it was only with the understanding of the structure of the DNA and the formulation of the central dogma of Molecular Biology in the 1950s and 1960s that the microscopic mechanisms of evolution could start to be unravelled. The central dogma postulates that DNA can be seen as a sort of read-only memory which encodes all the features and functionality of adult individuals. Through the process of transcription the information contained in the DNA would then be transferred into RNA. Then through the process of translation, this information would be carried by messenger RNA to the ribosomes. These would finally be responsible

R. Poli (✉)
School of Computer Science and Electronic Engineering, University of Essex, Colchester, UK
e-mail: rpoli@essex.ac.uk

C.R. Stephens
Instituto de Ciencias Nucleares, UNAM, Mexico City, Mexico
e-mail: stephens@nucleares.unam.mx

for transforming the information into proteins. Thus, only the processes involved in reproduction could modify the DNA of an organism.

While of course this picture is a reasonable approximation of a particular function of the DNA and a particular way in which evolution can happen, over the last few decades biologists have discovered that there are many more mechanisms through which the DNA could be modified, even during the lifetime of an individual or of a cell, and thus there are many more mechanisms through which evolution is likely to have happened [1].

It is today clear that evolution is extremely complex. However, even if we stick with the 1970s view of it and we consider the super-simplified forms of evolution in the computer that were inspired by such a view and used within the field of evolutionary computation, there is still a huge amount of potential complexity in it. This is, however, hidden complexity, that we see only if we try to understand evolution at a deeper level: the level of theory.

What is a theory? A theory is a logically self-consistent framework for describing the behaviour of a related set of phenomena. It often originates from, or is supported by, experimental evidence. Thus, a theory is a systematic and formalised expression of previous observations that is predictive, logical and testable. Why is theory useful? A successful theory gives an intuitive *understanding* of the system being modelled, which permits one to deduce new consequences and explain phenomena. It allows quantitative *predictions*, albeit more often than not approximate, about the system.

Do we have a theory of evolutionary systems? Well, yes and no. In many areas of evolutionary computation and classical theoretical population genetics, we have well-defined, complete and precise mathematical frameworks. The models of population genetics and evolutionary algorithms (EAs) have a lot in common, and in some cases, it is even arguable that there has been more progress in modelling *natural* evolution in EA theory than in population genetics. However, making progress both with our understanding of evolutionary algorithms and with making predictions has been an exceedingly difficult task. In this chapter, we will try to illustrate the nature of the difficulties and show how a technique known as coarse graining has helped us make progress.

2 Physics and Probability Preliminaries

Let us start with some simple notions from physics. In many systems one can identify a minimum set of variables, called *degrees of freedom* (d.o.f.), which describe the state of the system. For example, the d.o.f. of a set of static marbles on a table would be the x and y coordinates of each marble. Similarly, the d.o.f. of the molecules of a gas are the x, y, z coordinates of each molecule and the components v_x, v_y, v_z of their velocities.

Note that the x and y of each marble would work as d.o.f. also if we glued the marbles together in some form of geometric arrangement, say a rectangle. However, now if we moved the rectangle, the marbles in the rectangle rigidly move together, i.e., their trajectories are now constrained. Indeed, the position of each marble is known if we know the position of the rectangle. This has only three d.o.f.—the coordinates x_R and y_R of its centre of mass and its rotation θ_R on the plane.

The x and y coordinates of each marble are said to be the microscopic d.o.f. of the system, while the rectangle's d.o.f., x_R, y_R and θ_R, are what a statistical physicist would call the *effective degrees of freedom* (e.d.o.f.) of the glued marbles. We will give here a personal definition of e.d.o.f.: *a set of effective degrees of freedom for a system is a minimum set of variables which, at a given scale, naturally, possibly approximately, describe the states of a system for many practical purposes.* So, while the x, y, z, v_x, v_y, v_z of a gas molecule's state are the microscopic d.o.f. for a gas, in many practical cases, pressure and temperature are a set of (macroscopic) e.d.o.f. for the gas.

Naturally, all d.o.f. are effective to some degree. Nearly always, in the real world, any chosen set of d.o.f. (even the most microscopic and complete ones) provide only an incomplete representation of reality. When we chose to represent the marbles using their x and y coordinates we had made some assumptions: (a) the marbles' z coordinates are constant, (b) the marbles' rotations are either unimportant or unobservable, (c) the marbles are stationary (no velocities), (d) everything else, e.g., the marbles' temperature, colour, etc., is irrelevant. For some situations this set of d.o.f. is sufficient to represent the behaviour of the real system. However, it wouldn't be appropriate if someone could give a push to a marble.

Related to the notion of effective d.o.f. is the notion of coarse graining. *Coarse graining* means taking a system with many microscopic d.o.f. and finding an appropriate set of e.d.o.f. for it. How do we choose a good set of e.d.o.f.? There are some criteria: we want e.d.o.f. that offer a more compact, appropriate and computationally tractable description of the system, and in terms of which the dynamics looks "natural". (Often this naturalness manifests itself in terms of finding variables that are as independent and uncoupled as possible.)

Normally one describes a systems using d.o.f. for a reason: we want to understand how and why the state of the system changes over time. This is what a physicist would call the *dynamics of the system*. Also, we may want to describe special states, e.g., equilibria, where the state variables (d.o.f.) have particular relationships, e.g., the gas law $PV = RT$ where P is pressure, V volume, T temperature and R is the ideal gas constant. In coarse graining, generally, we pass from a description with one set of d.o.f., and corresponding interactions, to another, where both the e.d.o.f. and their effective interactions are different as we change scale, i.e., as we change from one set of e.d.o.f. to another. So, the dynamics and laws governing a system change as we change d.o.f..

Since a number of sources of randomness influence evolution, models of evolution will need to make use of probabilities. In particular we will use probability/ event tree diagrams to model evolution. Tree diagrams allow us to see all the possible outcomes of an event and calculate their probability. Let us briefly discuss these tools.

We will first consider what tree diagrams can do for us using very simple examples. Let us start with modelling repeated coin tosses. Spinning a coin has

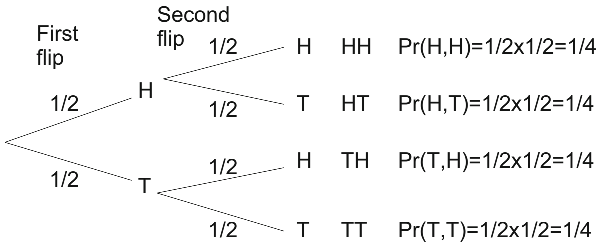

Fig. 1 Spinning a coin twice can be modelled using a tree diagram with two levels and four possible outcomes. The probability of an outcome of a double flip is the product of the probabilities $(1/2)$ encountered along the edges connecting the root node to the outcome

two outcomes: heads (H) or tails (T). Spinning it twice can be modelled using a tree diagram with two levels and four possible outcomes: HH, HT, TH and TT. In general, in a tree, we label each edge with the probability of an event following that edge and resulting in the corresponding outcome. The probability of an outcome (e.g., heads followed by another heads) is the product of the probabilities along the path connecting it to the root of the tree. If the coin is fair, the probabilities along each edge are all $1/2$ as shown in Fig. 1. The probability of the four outcomes of a double coin flip are the products of the probabilities encountered along the edges connecting the root node to an outcome: in this case $1/4$ for all outcomes.

Once the tree diagram is instantiated, one can use it to compute the probability of more complex events, such as the probability of having at least one head in two coin flips, by simply labelling as "success" all the relevant outcomes and adding together the probabilities of such "successful" outcomes.

Naturally, probability trees can also handle multiple outcomes, as in the case of drawing beads from a bag containing 2 blue, 3 red and 5 green beads, and, again, the product rule for computing the probability of outcomes applies. Also in this case, one can compute the probability of more complex events by simply adding up the probabilities of the relevant outcomes.

More generally, with probability trees we can find answers to questions such as: If we have a α_R (α_B, α_G) probability of getting a red (blue, green) bead in one draw, what's the probability of getting exactly n_R red, n_B blue and n_G green beads after M draws? For $M = 2$ we can use a tree diagram to answer the question. However, for bigger M and also for general values of α_R, α_B and α_G, we would really need a formula to find out the answers. Fortunately, the answer is given in probability textbooks. The repeated draw of beads from the bag follow a *multinomial distribution*. In other words,

$$\Pr(n_{\mathrm{R}}, n_{\mathrm{B}}, n_{\mathrm{G}}) = \binom{M}{n_{\mathrm{R}}, n_{\mathrm{B}}, n_{\mathrm{G}}} \alpha_{\mathrm{R}}^{n_{\mathrm{R}}} \alpha_{\mathrm{B}}^{n_{\mathrm{B}}} \alpha_{\mathrm{G}}^{n_{\mathrm{G}}}$$

where $\binom{M}{n_{\mathrm{R}}, n_{\mathrm{B}}, n_{\mathrm{G}}} = \frac{M!}{n_{\mathrm{R}}! n_{\mathrm{B}}! n_{\mathrm{G}}!}$ are multinomial coefficients.

Naturally, tree diagrams are useful also to model sequences of events that are not all of the same type. For example, we can alternate coin flips and bead draws, and find what's the probability of the event T,R,T,R from the corresponding tree. The only problem is that we just cannot use one of the multinomial distribution shortcuts to add up probabilities for us. Also, tree diagrams work even if events are not independent. For example, if we were performing bead draws but did not put the bead drawn back into the bag, the probabilities of drawing beads of different colour would change after each draw. Nonetheless, the probability of an outcome (e.g., R,R) is still the product of the probabilities along the path connecting it to the root.

Note that, in the case just described, in the second level of the tree we are performing a different kind of draw depending on the result of the first draw. We could be even more radical, and in fact consider a completely different set of events and outcomes for each outcome of the first draw. For example, we might draw another bead (with outcomes R, B and G) if the first bead drawn was red, we might flip a coin (with outcomes H and T) if the bead was blue and we might roll a dice (with outcomes 1, 2, 3, 4, 5 and 6) if the bead was green. While the exercise and the resulting tree (with its set of inhomogeneous outcomes) might seem odd, the product rule to compute the probability of outcomes would still apply.

3 Models of Evolutionary Algorithms

Modelling EAs means modelling the different events that take place during the creation of offspring, then modelling the iteration of such events which lead to the creation of a new generation and, finally, modelling the iterated construction of a generation to model a full run of the algorithm. We will do this in the following subsections.

3.1 Modelling the Genetic Operators

For simplicity, we will assume that we use a binary fixed-length representation and that offspring can be created by either the selection of one parent followed by mutation or the selection of two parents followed by crossover. More complex forms of creation can be modelled following the same principles.

The first question we need to answer is: What happens when an offspring is created in one particular generation? Irrespective of the genetic operators used, the creation of an offspring at a given time depends only on: what's in the population at that particular time (which is variable), the fitness function (which we will assume to be fixed) and the parameters of the EA, such as the population size (which we will also assume to be fixed). So, the thing on which offspring creation depends is the current population.

Table 1 Degrees of freedom *(left)* and possible configurations *(right)* for a population of three binary strings of length four

d_1	d_2	d_3	d_4
d_5	d_6	d_7	d_8
d_9	d_{10}	d_{11}	d_{12}

0 0 0 0	0 0 0 0		1 1 1 1
0 0 0 0	0 0 0 0	···	1 1 1 1
0 0 0 0	0 0 0 1		1 1 1 1

4,096

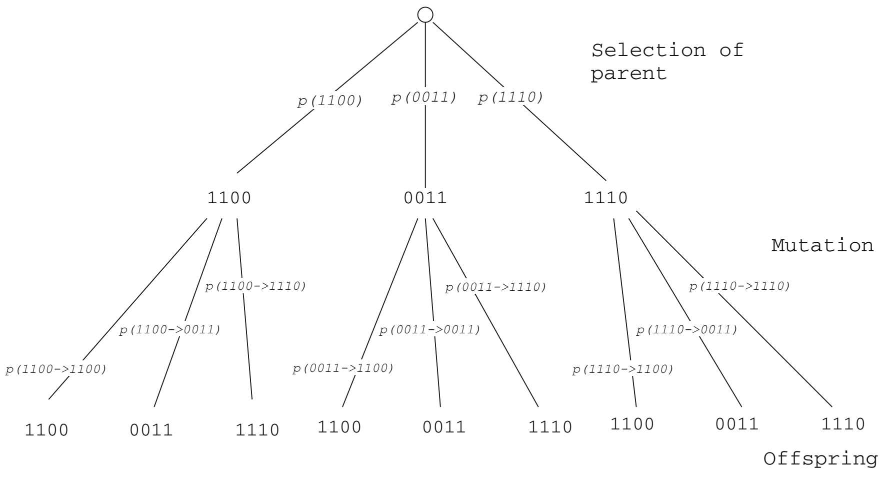

Fig. 2 Simplified tree model of the selection of a parent followed by its mutation

We will need to formalise this dependency in some way. The next question then is: What are the microscopic d.o.f. of a binary population? Clearly, the d.o.f. of a population are the bits in every individual of the population. For example, a population of three four-bit strings has 12 d.o.f. and there are $2^{12} = 4{,}096$ different configurations, as shown in Table 1.

Let us first consider the case of selection followed by point mutation. Suppose our current population is one of the configurations in Table 1(right), namely: $\{1100, 0011, 1110\}$. The creation of offspring by selection and mutation is represented by the tree diagram in Fig. 2, which is simplified for display purposes in that it assumes that only strings 1100, 0011 and 1110 can ever be generated (which, of course, isn't true).

Since point mutation acts on every bit independently, in the diagram, the probability (along the bottom edges) of mutating a string y to a string x is given by $p(y \to x) = p_m^{h(x,y)} \times (1 - p_m)^{\ell - h(x,y)}$ where $h(x, y)$ is the Hamming distance between x and y and ℓ is the string length. Let us further assume that we use fitness proportionate selection to select parents. In this form of selection, the selection probability for a string x, $p(x)$, is simply the ratio between the fitness of x and sum of the fitnesses of all individuals in the population.

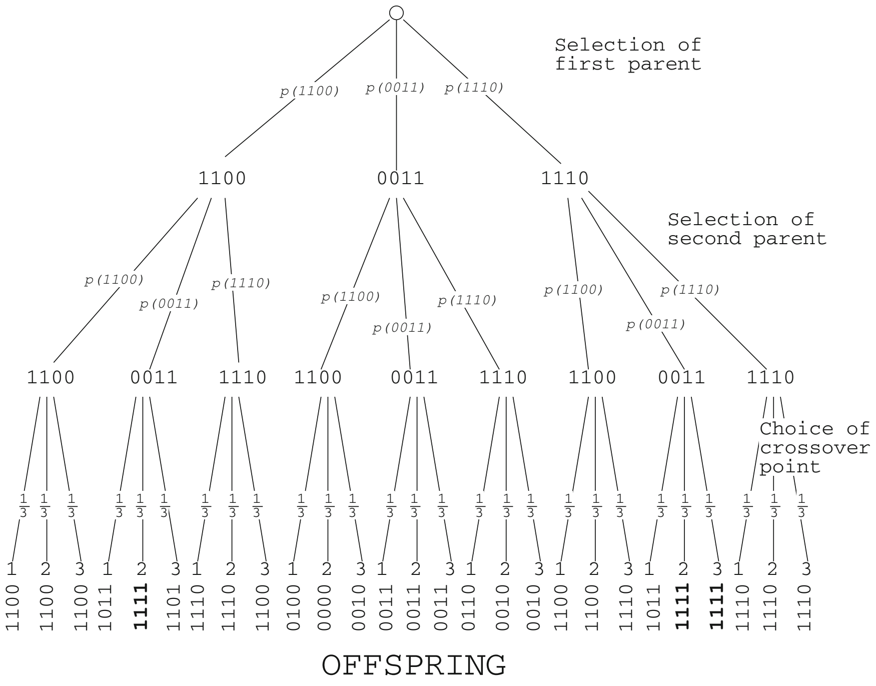

Fig. 3 Tree diagram for the selection of two parents followed by crossover

Then, if we know the mutation rate and the fitness function, we can turn the generic tree in Fig. 2 into a concrete probability tree. For example, if $p_m = 0.25$, then $p(1100 \to 1100) = 0.25^0 \times 0.75^4 = 0.316$, $p(1100 \to 0011) = 0.25^4 \times 0.75^0 = 0.004$, $p(1100 \to 1110) = 0.25^1 \times 0.75^3 = 0.105$, etc. So, all lower level edges have numerical probabilities associated with them. Also, suppose we are solving the OneMax problem,[1] then $f(1100) = f(0011) = 2$, $f(1110) = 3$ and the sum of fitnesses in the population is seven. So, $p(1100) = p(0011) = \frac{2}{7} = 0.286$ and $p(1110) = \frac{3}{7} = 0.429$. So, the probabilities in the upper part of the tree diagram are also defined. Of course, the multiplication rule still applies, but there are multiple paths leading to the same outcome (offspring). So, the offspring creation probabilities, α, are *sums* of products.

Let us now consider the process of generating offspring by selection followed by one-point crossover. The selection-crossover diagram for $\{1100, 0011, 1110\}$ is shown in Fig. 3. Clearly there are now three events taking place (two selections and one crossover), so the tree has three levels. The first two levels are exact copies of the first level of Fig. 2, since they simply represent the selection of the parents (with reselection allowed). As for the bottom level, since we use four-bit strings,

[1] In OneMax, fitness is the number of 1s in a bit string and the objective is to maximise that number.

one-point crossover can choose among three cut points. Each has a probability of $\frac{1}{3}$ of occurring.

If we look at the outcomes at the bottom of the tree, we can see that while we started from a population containing the three strings 1100, 0011, 1110, crossover can produce nine different strings: 0000, 0010, 0100, 0110, 1011, 1100, 1101, 1110 and 1111. Naturally, the sum-of-products rule still applies even if some probabilities in the tree are left unspecified. So, we can use the diagram to compute the creation probability $\alpha(x)$ for the nine outcomes (offspring strings): $\alpha(0000) = \frac{1}{3}p(0011)p(1100)$, …, $\alpha(1111) = \frac{1}{3}p(1100)p(0011) + \frac{1}{3}p(1110)p(0011) + \frac{1}{3}p(1110)p(0011)$.

As for mutation, if we know the fitness, we can work out selection probabilities $p(x)$, and from these the creation probability for all strings that can be created in the next generation. Again, if we just focus on the outcomes, we can represent the process with a tree diagram with just one level.

It is traditional for the process of creation of offspring via selection and crossover to first require the selection of the two parents and then execute the crossover operation, which in turns requires selecting a random crossover point. It is, however, quite clear that choosing the crossover point is totally independent from the selection of parental types. So, one could reorder these operations without altering the outcome. For example, selecting crossover points before selecting parents doesn't affect results in any way. Naturally, this different way of ordering events leads to a different but equivalent tree diagram model.

3.2 Coarse Graining and Generalising Models

Having developed models for the process of creating individuals via selection-mutation and selection-crossover for the specific population $\{1100, 0011, 1110\}$ we may ask: What if we had the population $\{1100, 1100, 0011, 1110\}$ which contains two copies of the string 1100 instead of just one?

Of course, we could just follow the same steps as before and redevelop tree diagrams for the events associated with such a population. For example, Fig. 4 shows the diagram for offspring generation via selection-crossover for the new population. We should note, however, that there is a lot of duplication (identical sub-trees, identical outcomes) in this diagram with respect to the one for the original population (Fig. 3).

Let us try to simplify the tree a little. First, we should note that if we doubled some of the probabilities labelling some of the edges in the tree, we could obtain the entirely equivalent, but more compact model in Fig. 5.

We now see clearly that this is the same diagram as for $\{1100, 0011, 1110\}$ except four probabilities have doubled. What is this diagram trying to tell us? Why the factor 2? Note: 2 is also the number of copies of the string 1100 in the population. Is this a coincidence?

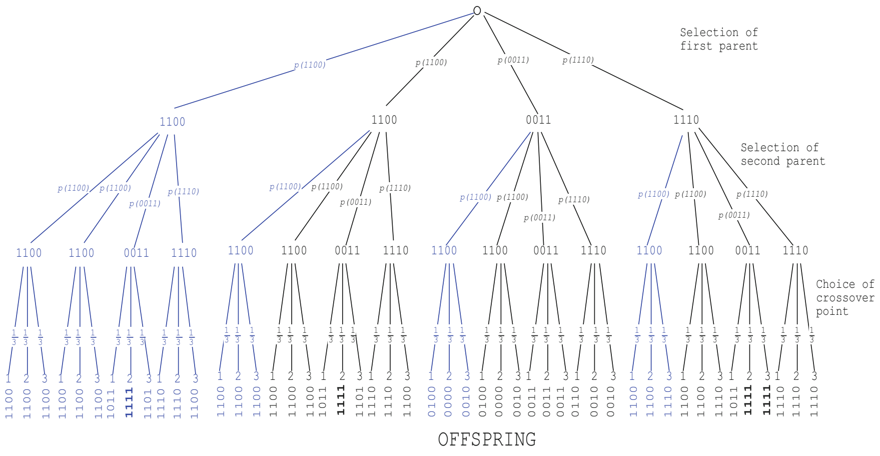
Selection of first parent
Selection of second parent
Choice of crossover point
OFFSPRING

Fig. 4 Tree diagram model of the creation of offspring via selection-crossover for the population {1100, 1100, 0011, 1110}

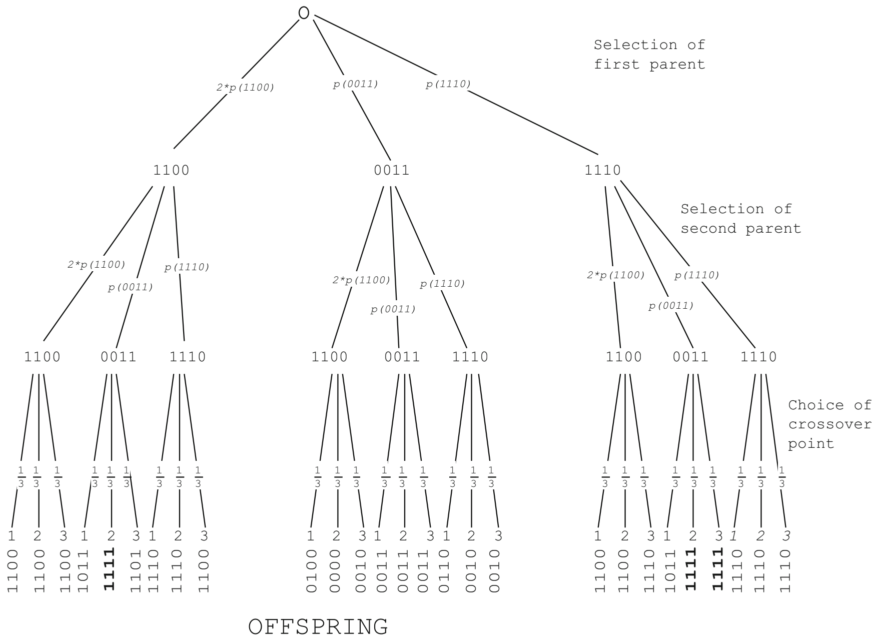
Selection of first parent
Selection of second parent
Choice of crossover point
OFFSPRING

Fig. 5 More compact, but equivalent, version of the model in Fig. 4

Naturally, it is not a coincidence. The original tree diagram in Fig. 3 is *valid for both populations* provided we interpret the selection of first and second parent events as the *selection of first and second parental types* not of particular individuals and we interpret the $p(x)$'s as probabilities of selecting a particular *type*.

What is a type? The notion of type is best explained with an example. The population {A, B, B, C, A, A, B} contains seven individuals but only three distinct types: A, B and C. So, the selection probability for a type is defined as

$$p(\text{type } x) = \sum_{s \text{ of type } x \text{ in pop}} p(s) \;=\; \#(s \text{ of type } x) \times p(s),$$

which for fitness proportionate selection becomes

$$p(x) = \frac{\#\text{ individuals of type } x \text{ in population} \times f(x)}{\text{sum of fitnesses of all individuals in population}} .$$

With these changes of interpretation, the model of crossover in Fig. 3 works for all populations having any number of copies of 1100, 0011 and 1110, e.g., $\{1100, 1100, 1100, 0011, 0011, 1110, 1110, 1110\}$. Similarly we could coarse grain selection-mutation models.

A question that immediately comes to mind is: Would the model work for also the population {0011, 1110} which contains zero copies of the string 1100? The answer is yes. There are extra outcomes in the tree in Fig. 3 which for the population {0011, 1110} have a zero probability of occurring, but the model is still formally correct.

In other words, the probability of selection of types automatically adjusts for the number of copies (including 0) of a type. Thus, we could generalise the selection part of the model to any population if we added all possible string types of a given length as outcomes of the selection process. Naturally, only a small subset of outcomes would have non-zero probability for any given population.

If we also generalise crossover to strings of a generic length ℓ, we would get a *general model of the selecto-recombination operator* which is both independent from the particular population at hand and from the length of the representation. This is shown in Fig. 6. The same approach would produce a general selecto-mutation model.

Note that the models in Figs. 2 and 6 can be collapsed down to a tree with a single level if we consider the selection-mutation and selection-crossover processes as a single (composite) event, respectively. Naturally, as shown in Fig. 7, we need to use an appropriate set of probabilities to label the edges that lead to the outcomes (offspring), namely the quantities $\alpha(x)$ for all possible values of x.

Let us reconsider at this point the question of how we compute the creation probabilities α. The tree we have just defined has 2^ℓ nodes at the first level, $2^\ell \times 2^\ell$ at the second and $(\ell - 1) \times 2^\ell \times 2^\ell$ at the third level. So, for any realistic value of ℓ it is *immense*. Interestingly, however, it can have "only" 2^ℓ distinct outcomes (offspring). Since $2^\ell \ll (\ell - 1) \times 2^\ell \times 2^\ell$, we should expect to have to multiply and add an exponential number of probabilities to compute $\alpha(x)$ for each outcome.

Obviously, we cannot do this by hand. So, we introduce a function to help us do it: $\lambda(y, z, n, x) = 1$ if crossing over y and z at position n produces x, and 0

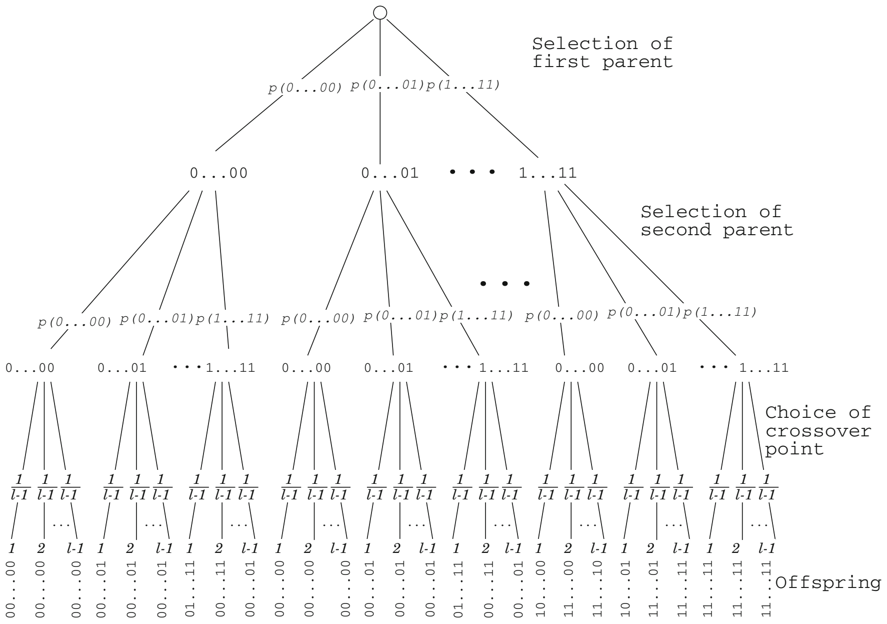
Selection of first parent
p(0...00)
p(0...01)
p(1...11)
0...00
0...01
1...11
Selection of second parent
Choice of crossover point
1/(l-1)
1
2
l-1
Offspring

Fig. 6 General population- and length-independent model of the process of creating offspring via selection and recombination

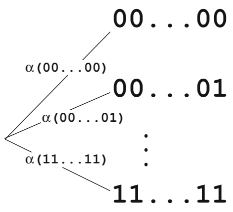

Fig. 7 The tree representation for a general selection-crossover process and a general selection-mutation process, represented as a single (*composite*) event

otherwise. With this, we can now write the creation probability via selection and crossover for a generic type x, in a general form:

$$\alpha(x) = \sum_{y\in\Omega}\sum_{z\in\Omega}\sum_{n=1}^{\ell-1}\lambda(y,z,n,x)\left(\frac{p(y)p(z)}{\ell-1}\right) \tag{1}$$

where Ω is the space of all possible strings of length ℓ.[2]

[2]Naturally, we would like to have an explicit form for $\lambda(y,z,n,x)$. It exists, but for now we will not look at it.

Note that Fig. 7 represents also a general model for the selection-mutation process, although now we have $\alpha(x) = \sum_{y \in \Omega} p(y)p(y \to x)$ where $p(y \to x)$ is the probability of mutation transforming an individual of type y to an individual of type x (which we have already computed).

3.3 Modelling One Generation

Now that we have models of the process of creating one offspring, we are in a position to start constructing models of the iterated application of the offspring-creation process, in view of modelling a generation.

What happens when an offspring is created in one particular generation? We have already seen that, irrespective of the genetic operators used, offspring creation is similar to drawing beads from a bag. Generally, the process is modelled in Fig. 7 for a *generic* operator or set of operators (where $\alpha(x)$ is the probability of creating individual x with the chosen operators). Naturally, the creation process will have many more than the three possible outcomes of the bead-draw process, but the principles at work are exactly the same. Then, what happens if we iterate the creation process, e.g., to create a full new generation? In a generational EA, *within a generation the α's, i.e., the offspring creation probabilities, are constant*. So, for a generation we have a probability tree diagram very much like the one for iterated bead draws as illustrated in Fig. 8 for a population of three individuals.

Naturally, in general, the general model in Fig. 7 would need to be used at each level of the tree diagram representing one generation. In either case, this time the outcomes of the process are populations. For example, for the model in Fig. 8 there are 27 outcomes:

1111	1111	1111	1111	1111	1111	1111	**1111**	1111		0000
1111	1111	1111	0000	0000	0000	1010	**1010**	1010	...	0000
1111	0000	1010	1111	0000	1010	1111	**0000**	1010		0000

Again, like for multiple bead draws, the multiplication rule applies: when we want to compute the probability of an outcome, we simply need to multiply the probabilities along the edges of the path that goes from the root of the tree to the outcome of interest. So, for example, the probability of the next generation being the eighth outcome (in boldface) is $\alpha(1111) \times \alpha(1010) \times \alpha(0000)$.

At this stage we note the possibility of a further coarse graining: coarse graining on positional symmetries. In other words, do we generally care about the differences between populations, such as the second and the fourth above, which have exactly the same strings but in a different order?

We might, if some genetic operator depends on position, e.g., if selection only allowed mating of neighbouring individuals. However, typically *genetic operators are position independent*, so we don't care about positional differences. Because these populations are effectively equivalent, we don't really need to distinguish them using the original set of d.o.f.. We only care about how many individuals of any

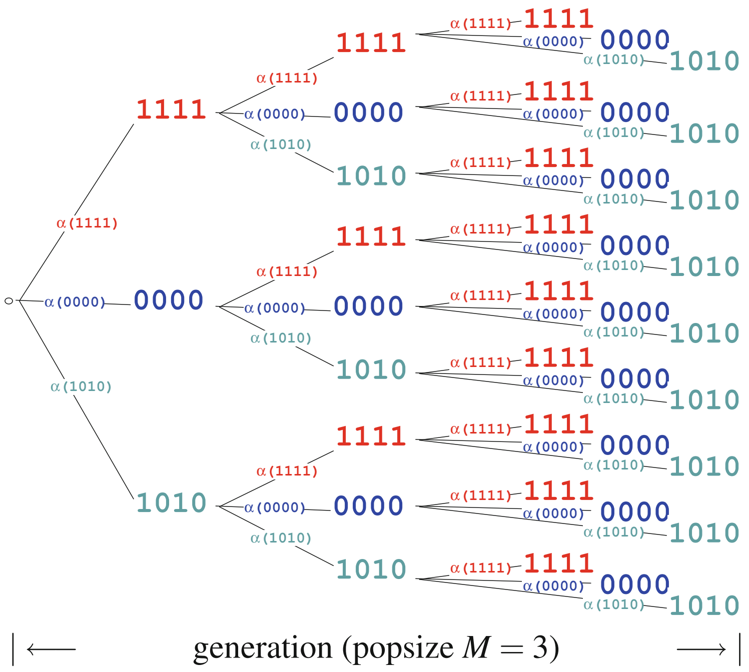

Fig. 8 Simplified tree diagram model of a generation for a population of three individuals

given *type* are present in the population. E.g., two of type 1111 and one of type 0000. So, we can define new (coarse grained) e.d.o.f.: using the number n_x of individuals of a given type x (for all x) as a representation. So, a population with two individuals of type 1111 and one of type 0000 can be represented as $n_{1111} = 2$, $n_{0000} = 0$ and $n_{1010} = 1$.

More generally, for $\ell = 4$ and a population of size M, our e.d.o.f. are

0000	n_{0000}	0100	n_{0100}	1000	n_{1000}	1100	n_{1100}
0001	n_{0001}	0101	n_{0101}	1001	n_{1001}	1101	n_{1101}
0010	n_{0010}	0110	n_{0110}	1010	n_{1010}	1110	n_{1110}
0011	n_{0011}	0111	n_{0111}	1011	n_{1011}	1111	n_{1111}

with the constraint $\sum_x n_x = M$.

Having now coarse-grained on positional symmetries, have we actually saved anything in terms of complexity? Simple counting arguments can show that with strings of length ℓ there are $\binom{M+2^\ell-1}{2^\ell-1}$ possible populations of M individuals. This is in general much smaller than the $2^{M\times\ell}$ populations we would have to consider if we did not coarse grain. So, the saving is huge.

Of course, as for the beads, if x_i is the i-th possible offspring, the probability that in the next generation one gets n_1 individuals of type x_1, n_2 of type x_2, etc., is just given by the multinomial distribution

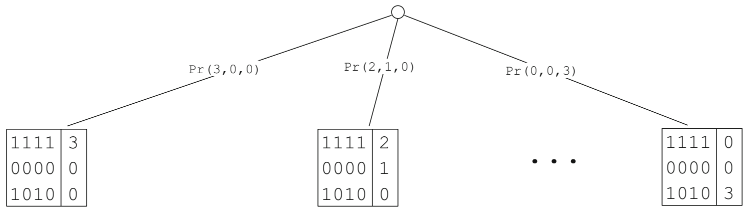

Fig. 9 Tree model of the process of creating a generation as a single *(composite)* event. (Here, for simplicity, we assume that only strings 1111, 0000 and 1010 can be generated. So, there are only ten possible outcomes)

$$\Pr(n_1, n_2, \cdots) = \binom{M}{n_1, n_2, \cdots} \alpha(x_1)^{n_1} \alpha(x_2)^{n_2} \cdots$$

Effectively this equation is a representation for the *dynamics* of the system in terms of the new e.d.o.f. n_x. With it, we can get (probabilistic) information about the next generation. For example, we know all the moments of the distribution of next-generation populations. For instance, the expected number of copies of any x_i in the next generation is simply $M\alpha(x_i)$.

As we did before with the α's, knowledge of $\Pr(n_1, n_2, \cdots)$ allows us to model a generation, which is the result of a series of offspring-creation events, as one (composite) generation-creation event, the outcomes of which are populations. A sample (simplified) tree diagram for the population $\{1111, 0000, 1010\}$ is shown in Fig. 9.

3.4 Modelling Runs

Having now "tamed" the complexity of the generation-creation process using the $\Pr(n_1, n_2, \cdots)$ and tree diagrams such as Fig. 9, we are now in a position to model entire runs.

It is clear that tree diagrams can be used also to model multiple generations and, thus, runs as illustrated in simplified form in Fig. 10, where we assumed that we start runs from a given (known) population. In the figure, different edges in the tree leading to the *same population* are labelled by *different probabilities*. The reason for this is that the action of the genetic operators (e.g., selection) depends on who is in the population, so the α's and consequently the probabilities $\Pr(n_1, n_2, \cdots)$ depend on it. So, in general we should expect $\Pr(3, 0, 0) \neq \Pr'(3, 0, 0)$ and similarly for most other labels.[3]

[3]As a result, we cannot use the multinomial distribution to predict the future over multiple generations.

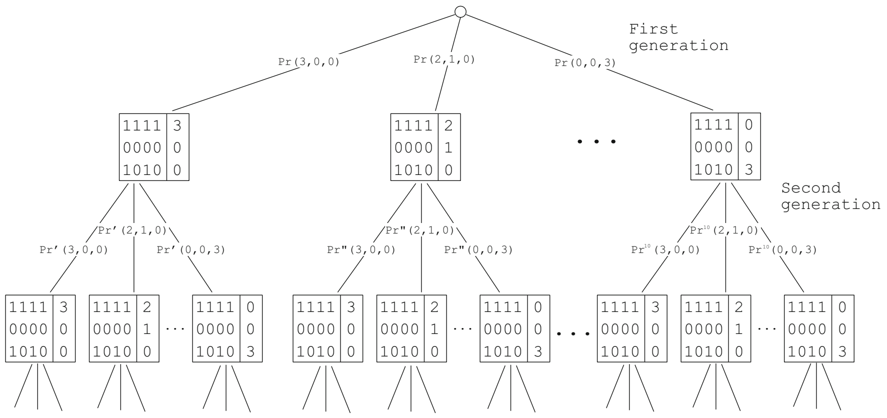

Fig. 10 Tree model of the process of running an EA for multiple generations

Of course, outcomes are now trajectories in population space. For example, if we ran an algorithm for three generations, an outcome could be

$$\text{trajectory} = \left(\text{Initial population} \Rightarrow \begin{array}{|c|c|}\hline 1111 & 1 \\ \hline 0000 & 2 \\ \hline 1010 & 0 \\ \hline \end{array} \Rightarrow \begin{array}{|c|c|}\hline 1111 & 2 \\ \hline 0000 & 1 \\ \hline 1010 & 0 \\ \hline \end{array} \Rightarrow \begin{array}{|c|c|}\hline 1111 & 3 \\ \hline 0000 & 0 \\ \hline 1010 & 0 \\ \hline \end{array} \right)$$

Applying the multiplication rule to this sample trajectory we obtain

$$\Pr(\text{trajectory}) = \Pr(1,2,0) \times \Pr{}'(2,1,0) \times \Pr{}''(3,0,0)$$

where

$$\Pr(1,2,0) = \Pr\left(\text{getting} \begin{array}{|c|c|}\hline 1111 & 2 \\ \hline 0000 & 1 \\ \hline 1010 & 0 \\ \hline \end{array} \text{from the initial population} \right),$$

$$\Pr{}'(2,1,0) = \Pr\left(\text{getting} \begin{array}{|c|c|}\hline 1111 & 2 \\ \hline 0000 & 1 \\ \hline 1010 & 0 \\ \hline \end{array} \text{from population} \begin{array}{|c|c|}\hline 1111 & 1 \\ \hline 0000 & 2 \\ \hline 1010 & 0 \\ \hline \end{array} \right),$$

$$\Pr{}''(3,0,0) = \Pr\left(\text{getting} \begin{array}{|c|c|}\hline 1111 & 3 \\ \hline 0000 & 0 \\ \hline 1010 & 0 \\ \hline \end{array} \text{from population} \begin{array}{|c|c|}\hline 1111 & 2 \\ \hline 0000 & 1 \\ \hline 1010 & 0 \\ \hline \end{array} \right),$$

are *conditional probabilities*. In other words, the probability on the edge from a population $\mathscr{P}$ to population $\mathscr{P}'$ is the probability of $\mathscr{P}'$ being the next generation *when we use the creation probabilities α computed for population $\mathscr{P}$*. We write this probability as $\Pr(\mathscr{P}' \mid \mathscr{P})$.

Now, let us imagine we numbered all possible populations: $\mathscr{P}_1$, $\mathscr{P}_2$, and so on. The probabilities of the edges between all possible pairs of populations could be

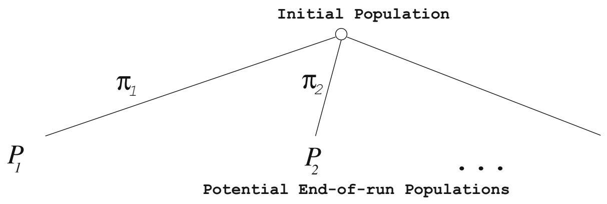

Fig. 11 Tree model of the process of performing a run of an EA as a single (*composite*) event

represented as a matrix $\mathbf{Q} = \left(\Pr(\mathscr{P}_i \mid \mathscr{P}_j)\right)$, which is effectively the *transition matrix for a Markov chain* whose states are all possible populations: $\mathscr{P}_1$, $\mathscr{P}_2$, and so forth. So, the probability of the population following a particular trajectory during a run is the product of a set of elements of $\mathbf{Q}$.

However, in many cases one is not particularly interested in the trajectory followed by a run, but, more simply, in the population obtained at the end of that run. In this case, there is a further opportunity for coarse graining: we could consider populations as outcomes of runs of the EA and ignore all other aspects of the dynamics that led to such end-of-run populations. For the properties of Markov chains, we can easily compute the probability of each such outcome. Indeed, the probability distribution, π_t, of the EA being in any state (having a particular population) at a future generation t is simply

$$\pi = \mathbf{Q}^t \pi_0$$

where π_0 is the initial probability distribution over states, which for a known initial population is simply a vector with one unitary component (representing the initial population) and zeros everywhere else.[4] So, if we know the initial population, again, we can treat the complex chain of events taking place in a run as one (composite) event which we can model with a tree diagram with only one level. This tree is represented in Fig. 11 where π_1, π_2, etc. are first, second, etc. component of the vector π, respectively.

Naturally, many EAs start from a random population, not a known population. It should, however, be clear by now, that the process of randomising the initial generation before starting a run simply adds an extra level to the tree-diagram model of runs in Fig. 10. This does not change at all the resulting model of runs as single composite events shown in Fig. 11. All that changes is the initial distribution over populations, π_0. So, the case of initial random populations, too, is covered by our analysis.

[4]This is Michael Vose's model for a genetic algorithm [2].

4 Coarse Graining and the Emergence of Schemata

So far we have already used coarse-graining a number of times: we coarse-grained over types when performing selection, we coarse-grained over positions in the population and, finally, we coarse-grained over population dynamics focusing only on end-of-run populations.

In theory, with **Q** one can compute everything that can be computed about the future of a run. For example, we could compute the probability of solving a problem in say 50 generations. In practice, however, **Q** is just too big for one to be able to create and use it. We can only study its properties mathematically. So, while the theory presented above is an exact theory, it is a theory that is hard to use to make predictions and to understand why an EA behaves the way it does. The problem is that the model and its e.d.o.f. are still too *microscopic* to show us the regularities of an algorithm. We need something else.

4.1 In Search of New Effective Degrees of Freedom

Earlier we expressed the creation probability via selection and crossover as in Eq. (1). It is now time to learn more about $\lambda(y, z, n, x)$.

Let us fix the type of interest as $x = 11$ ($\ell = 2$). From the selection-crossover event diagram[5] we find that $\lambda(y, z, n, x) \equiv 0$ except for

$$\lambda(10, 01, 1, 11) = \lambda(10, 11, 1, 11) = \lambda(11, 01, 1, 11) = \lambda(11, 11, 1, 11) = 1.$$

So, $\alpha(11) = p(10)p(01) + p(10)p(11) + p(11)p(01) + p(11)p(11)$, which includes only 4 terms out of the possible 16. $\alpha(11)$ can thus can be written as

$$\alpha(11) = (p(10) + p(11)) \times (p(01) + p(11)).$$

Is this factorisation a coincidence, or is the equation trying to tell us that $A = p(10) + p(11)$ and $B = p(01) + p(11)$ would lead to a more natural description of the creation process as $\alpha(11) = A \times B$? To answer these questions we will need to find (and work with) an explicit form of λ.

The function λ is 1 only if there is a match between the offspring's bits and the first parent bits before the crossover point *and* there is a match between the offspring's bits and the second parent bits after the crossover point. We can write this conjunction of multiple requirements as

$$\lambda(y, z, n, x) = \prod_{i \leq n} \delta(x_i = y_i) \prod_{i > n} \delta(x_i = z_i)$$

[5]For $\ell = 2$ there is only one valid crossover point ($n = 1$).

where x_i, y_i and z_i are the bits in x, y and z, respectively, and $\delta(\mathbf{expression}) = 1$ if *expression* is true, and 0 otherwise.

Substituting this description of λ into Eq. (1) yields

$$\alpha(x) = \sum_{y\in\Omega}\sum_{z\in\Omega}\sum_{n=1}^{\ell-1}\left(\prod_{i\leq n}\delta(x_i = y_i)\prod_{i>n}\delta(x_i = z_i)\right)\left(\frac{p(y)p(z)}{\ell-1}\right).$$

We can then reorder the calculation in an interesting way:

$$\alpha(x) = \sum_{n=1}^{\ell-1}\frac{1}{\ell-1}\left(\sum_{y\in\Omega}p(y)\prod_{i\leq n}\delta(x_i = y_i)\right)\times\left(\sum_{z\in\Omega}p(z)\prod_{i>n}\delta(x_i = z_i)\right)$$

That is, like $\alpha(11) = A \times B$, also $\alpha(x)$ can be expressed in a simpler form:

$$\alpha(x) = \sum_{n=1}^{\ell-1}\frac{1}{\ell-1}A_n \times B_n,$$

for $A_n = \sum_{y\in\Omega} p(y)\prod_{i\leq n}\delta(x_i = y_i)$ and $B_n = \sum_{z\in\Omega} p(z)\prod_{i>n}\delta(x_i = z_i)$. It is then natural to ask: What are the factors A_n and B_n? Why do things look so much simpler if we calculate α in this way?

4.2 Coarse Graining from Types to Sets

The action of the δ's in A_n and B_n is to zero some terms in the corresponding summations over Ω, i.e., they *limit* the range of the summations. That is

$$A_n = \sum_{y\in L_n(x)} p(y) \quad \text{and} \quad B_n = \sum_{z\in R_n(x)} p(z),$$

for an appropriate choice of the two sets $L_n(x)$ and $R_n(x)$, namely $L_n(x) = \{y \in \Omega : y_1 = x_1, \cdots, y_n = x_n\}$ and $R_n(x) = \{z \in \Omega : z_{n+1} = x_{n+1}, \cdots, z_\ell = x_\ell\}$. This suggests that there is a further level of coarse-graining that we can do to tame the complexity of evolution: moving from types to sets of types.

We can easily extend the definition of $p(x)$ *from types to sets* as follows:

$$p(A) = \Pr\{\text{selecting a individual of a type belonging to set } A\}.$$

Because all events in A are mutually exclusive $p(A) = \sum_{x\in A} p(x)$. Thus, we can express $A_n = p(L_n(x))$, $B_n = p(R_n(x))$, and

$$\alpha(x) = \sum_{n=1}^{\ell-1}\frac{1}{\ell-1}p(L_n(x)) \times p(R_n(x)).$$

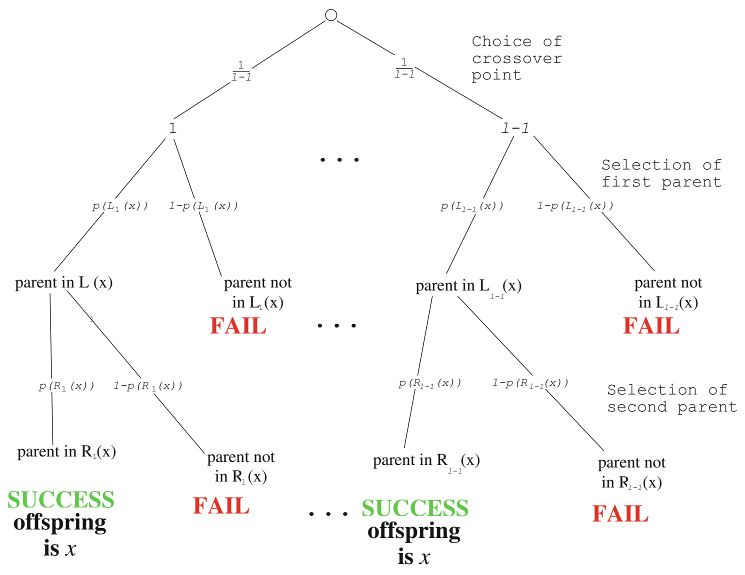

Fig. 12 Tree model of the selection-crossover process coarse-grained using sets of types

Let us analyse this result. We see that the probability of creating an offspring of type x is now decomposed as a sum of products of three probabilities: the probability of choosing a particular crossover point $\frac{1}{\ell-1}$, the probability of selecting a first parent belonging to the set $L_n(x)$ and the probability of selecting a second parent belonging to the set $R_n(x)$. Note that these two sets depend on both x and the choice of crossover point n. This suggests that the most coarse-grained tree diagram we can construct to represent these events is one that starts with the choice of a crossover point at the first level of the tree. Then it focuses on different kinds of selection events depending on the chosen crossover point, thereby forming a hybrid hierarchy of the same kind considered at the end of Sect. 2. The different kinds of selection are the selection of types belonging to the sets $L_n(x)$ for the second level of the tree and the sets $R_n(x)$ for the third level of the tree. Naturally at every level we must consider not only just the positive outcome (the selected type is in the set) but also its corresponding negative one (the type is not in the set).

The resulting tree diagram is shown in Fig. 12. Note how much smaller and simpler than the original in Fig. 6 this is. There are only $O(\ell)$ nodes in it as opposed to the original $O(2^\ell)$. Clearly, $L_n(x)$ and $R_n(x)$ are really good e.d.o.f. for describing the creation of instances of x. But what are these sets?

4.3 *Schemata as Effective Degrees of Freedom*

Holland [3] introduced the notion of *schema* as a tool for analysing EAs. A schema is a set of individuals represented with a particular pattern: a string of symbols from the alphabet $\{0, 1, *\}$. The semantics of the symbols is this: (a) all individuals in the schema must have the specified pattern of 0s and 1s within them, and (b) the $*$ symbols mean we "don't care" to check the corresponding bit in the individuals. For example, the schema $11{*}{*}$ represents all strings of length 4 which start with 11, i.e., $11{*}{*} = \{1100, 1101, 1110, 1111\}$.

With this notion in hand, it is now easy to see that for a fixed x and n, the two sets $L_n(x)$ and $R_n(x)$ are particular *schemata*, namely $L_n(x) = x_1 \cdots x_n * \cdots *$ and $R_n(x) = * \cdots * x_{n+1} \cdots x_\ell$. Hence

$$\alpha(x) = \sum_{n=1}^{\ell-1} \frac{1}{\ell - 1} p(x_1 \cdots x_n * \cdots *) p(* \cdots * x_{n+1} \cdots x_\ell).$$

In other words, schemata are the right type of coarse-graining to represent the operation of creating an individual of a particular type through selection and crossover. Note, however, that there is a slight asymmetry in this equation: we have used schemata as effective d.o.f. for the right-hand side, but types as e.d.o.f. for the left-hand side. Could we coarse grain creation events even further by extending the interpretation of the domain of α from types to *sets of types*? If A is a schema, i.e., $A = s_1 s_2 \cdots s_\ell$ with $s_n \in \{0, 1, *\}$, and we define $\alpha(A) = \sum_{x \in A} \alpha(x)$, it is possible to prove [4] that this coarse graining leads to an equation of exactly the same form as that for $\alpha(x)$, namely:

$$\alpha(s_1 s_2 \cdots s_\ell) = \sum_{n=1}^{\ell-1} \frac{1}{\ell - 1} p(s_1 \cdots s_n * \cdots *) p(* \cdots * s_{n+1} \cdots s_\ell).$$

This reveals the hierarchical nature of creation events across multiple generations. For example, the probability of creating individuals of type 111 at generation t is determined by the selection probabilities of individuals of the sets $1{*}{*}$, ${*}11$, $11{*}$ and ${*}{*}1$ at that generation. These selection probabilities depend not only on the fitness function but also on the number of individuals within each set at generation t. These were created in the previous generation, $t - 1$. Their distribution is entirely determined by the probability of creating individuals in each sets at generation $t - 1$. For example, the number of individuals in ${*}11$ depends on $\alpha({*}11)$ at generation $t - 1$. In turn $\alpha({*}11)$ is controlled by the individuals in sets ${*}1{*}$ and ${*}{*}1$ at generation $t - 1$. Their number is stochastic, but, of course, depends on $\alpha({*}1{*})$ and $\alpha({*}{*}1)$ at generation $t - 2$.[6]

[6]Note that $\alpha(s_1 s_2 \cdots s_\ell)$ becomes particularly simple when all $s_i = *$ except one. In that case it is easy to verify that $\alpha(s_1 s_2 \cdots s_\ell) = p(s_1 s_2 \cdots s_\ell)$.

5 Conclusions

Here we have shown that what makes even the simplest forms of evolution so complicated to analyse mathematically is the explosive number of possible outcomes of each operation or sequence of operations. We have also shown that coarse graining has helped us formulate more intuitive models of the dynamics of EAs, thereby taming their complexity, at least to some degree.

Naturally, many people have attempted to model EAs for many years. So, different models and different types of coarse grainings have been used. For example, the models presented here have been extended to variable-length strings, to non-binary alphabets, to more general forms of crossover and mutation, to tree-like structures, to diploidy/polyploidy and multiple chromosomes, to more recently discovered genetic operations (inversion, transposition, gene duplication, gene deletion, etc.), etc. [5–8]. Also, there are systems where the natural e.d.o.f. are Fourier (Walsh) modes and systems that can be characterised by interpreting the crossover operation as a low-pass filter. It is also possible to apply the Renormalisation Group [9] to model EAs. Also, search in continuous spaces can be modelled using the finite element method (coarse-graining the states of the system).

There are, however, also a number of still unresolved issues including, for instance, deriving convergence proofs using coarse-grained variables, deriving problem-difficulty indicators based on such variables, relating no-free-lunch theory to coarse-grained models, and many others.

While coarse graining the d.o.f. and dynamics of evolution is difficult, most of what has been achieved for EAs is relevant for biological evolution, too. For example, the key notion of schema and the hierarchical nature of creation via crossover has been almost completely neglected in population genetics. Much might be learnt about natural evolution by applying such notion.

References

1. Shapiro, J.A.: Evolution: A View from the 21st Century. FT Press, Upper Saddle River (2011)
2. Vose, M.: Modeling simple genetic algorithms. In FOGA-92, Foundations of Genetic Algorithms, pp. 24–29. Vail, Colorado, (1992)
3. Holland, J.H.: Adpatation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)
4. Stephens, C.R., Waelbroeck, H.: Schemata evolution and building blocks. Evol. Comp. **7**, 109–124 (1999)
5. Poli, R., Stephens C.R.: Understanding the biases of generalised recombination: Part I. Evol. Comput. **14**(4), 411–432 (2006)
6. Poli, R., Stephens C.R.: Understanding the biases of generalised recombination: Part II. Evol. Comput. **15**(1), 95–131 (2007)
7. Poli, R., McPhee, N.F.: General schema theory for genetic programming with subtree-swapping crossover: Part I. Evol. Comput. **11**(1), 53–66 (2003)
8. Poli, R., McPhee, N.F.: General schema theory for genetic programming with subtree-swapping crossover: Part II. Evol. Comput. **11**(2), 169–206 (2003)
9. Kadanoff, L.P.: Statistical Physics: Statics, Dynamics and Renormalization. World Scientific, Singapore (2000)