Fundamenta Informaticae XXI (2003) 1001–1017
IOS Press

Basic principles for understanding evolutionary algorithms

Karsten Weicker
Institute of Formal Methods in Computer Science
University of Stuttgart
Stuttgart, Germany
Karsten.Weicker@fmi.uni-stuttgart.de

Nicole Weicker
Institute of Formal Methods in Computer Science
University of Stuttgart
Stuttgart, Germany

Abstract. Evolutionary algorithms are often successfully applied to hard optimization problems. However, besides rules of thumb and experience, trial and error is still the leading design technique for evolutionary algorithms. A profound theoretical foundation guiding those applications is still missing. This article outlines a networked understanding of evolutionary algorithms. As a first step towards that goal, it reviews and interrelates major theoretical results and working principles in order to give an extensive insight into the internal processing of evolutionary algorithms. This not only helps to understand success and failure of evolutionary algorithms in applications but, in addition, could lead to a theory-guided design process enrichening and relieving today's heuristic techniques.

Keywords: evolutionary algorithm, optimization, theory, working principles, schema theory

1. Introduction

"Evolutionary algorithm" is a generic term for a wide number of algorithms mimicking the Darwinian principles of evolution for the purpose of optimization or problem solving. The basic components of these algorithms are populations of individuals, i.e. competing candidate solutions, the notion of quality or fitness to assess each individual, reproduction of candidate solutions using mechanisms of inheritance, and selective pressure which can take place in the parental selection or the survival competition. The

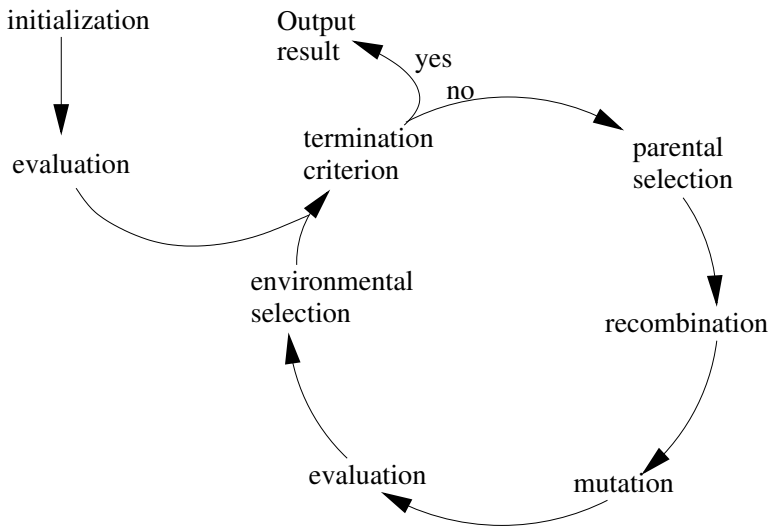

Figure 1. Schematic description of an evolutionary algorithm

basic algorithm is sketched in Figure 1. Contrary to other algorithms, they do not have any additional memory but the current set of individuals in the population. The idea of evolutionary algorithms has been discovered several times independently resulting in three main paradigms: genetic algorithms (GA) [21, 23, 18], evolution strategies (ES) [33, 38], and evolutionary programming (EP) [17, 15]. In its most general form an evolutionary algorithm works on a coded genotype space $\mathcal{G}$ which can be decoded into the search space $\mathcal{S}$ by a decoding function $dec : \mathcal{G} \rightarrow \mathcal{S}$. Note, that for certain algorithms $\mathcal{G} = \mathcal{S}$. The evolutionary operators, mutation and recombination, are defined on the genotype space and the evaluation of individuals is defined on the search space ($F : \mathcal{S} \rightarrow \mathbb{R}$). Selection introduces no new individuals but only changes the frequencies of the individuals for serving as parents or possible parents in the next generation. This article is not concerned with specific problem classes like constraint or multiobjective optimization.

Evolutionary algorithms have demonstrated their potential in numberless projects since they are easy to use. They are problem independent, i.e. they neither need a special representation for candidate solutions nor put any other restricting requirements on the problems to be tackled. Thus they can be viewed as universally applicable. However, practice shows that they are not general problem solvers since any successful application relies on careful tuning of operators, parameters, and problem-dependent techniques. This observation is supported by the "No Free Lunch" theorems [45, 9, 13, 37] which have shown that, on average, any deterministic or stochastic optimization algorithm is as good as any other algorithm on sets of problems that are closed under the permutation of fitness values. This implies that, on the one hand, each algorithm has a niche within the space of all problems and, on the other hand, for each optimization problem the algorithm must be tailored to the problem.

Today, designing evolutionary algorithms is primarily governed by certain rules of thumb, experience, and trial and error. However, without a theoretical foundation an algorithm's successful application remains a pure heuristic craft. Instead, foundations on how and why evolutionary algorithms work are necessary.

Current theoretical approaches either isolate single operators or parameters or the algorithm is analyzed from a global perspective. In the existing literature, often the single paradigms are analyzed and examined in depth on simple exemplary problems resulting in certain rules. But those rules are only in a restricted manner transferable to other problems or algorithms – especially problem-specific algorithms. Concerning the global perspective, theory in evolutionary computing tries to describe and analyze the processing as a whole, but the natural laws of evolutionary computing seem to be more complex and can probably better be understood as decentralized reciprocal actions. The "natural" representation for such an understanding is a network of interactions and interdependencies. Especially indirect quantities like the diversity of the population or characteristics of problems seem to play an outstanding role in such a network. Also the network must not be developed for one specific paradigm but should rather be formulated for a general evolutionary algorithm. Such a network of intra-process relations cannot be developed in one step and presumably today's theory is not suited to answer all possible questions. But as a first step the existing theoretical results are reviewed in this article and evaluated to what extend they make a statement that fits into a networked understanding.

The next section provides an example for the dynamics and interactions in evolutionary algorithms. It shows that the single components of evolutionary algorithms depend on each other in order to motivate the network of interdependencies. Sections 3 and 4 consider the selection methods and the evolutionary operators and sketches their interactions. The classical schema and forma theory is examined in Section 5 in respect of its contribution to such a network. Section 6 gives an overview on models of the complete search process and Section 7 concludes.

2. Balancing evolutionary algorithms

Evolutionary algorithms are characterized by many parameters which may be used for tuning an algorithm for a specific problem. Those parameters cannot be understood in isolation but rather as an interweaved, tangled network. Changing one parameter has significant impact on the effect of other parameters. Good parameter settings differ from problem to problem and cannot be transfered to algorithms using operators with different characteristics.

Theoretical examinations are primarily available for isolated single parameters and specific problems (e.g. an optimal mutation rate for genetic algorithms maximizing the number of ones in a binary string) which are of little help in general. Most knowledge is of heuristic nature and result of empirical examinations.

With a simple experiment the necessity of a networked representation is illustrated. In the Counting-Ones problem the number of "1"s in a 32 bit string must be maximized. A genetic algorithm with 2-point crossover with crossover rate 0.9, bit-flipping mutation and tournament selection of size 2 is used. The population size and the mutation rate are varied according to Figure 2 and in each optimization run the number of evaluations of the objective function was restricted to 512. Figure 2 shows the best fitness averaged over 200 independent experiments for each parameter combination. Even in this simple problem the two parameters are not separable and sequential tuning of population size and mutation rate (or vice versa) may lead to suboptimal performance. This experiment, which is a variant of the results in [10], shows that there exist local interactions among parameters that should be recorded. But it still leaves open the question how the parameters affect the processing and what quantities could be used to describe those effects.

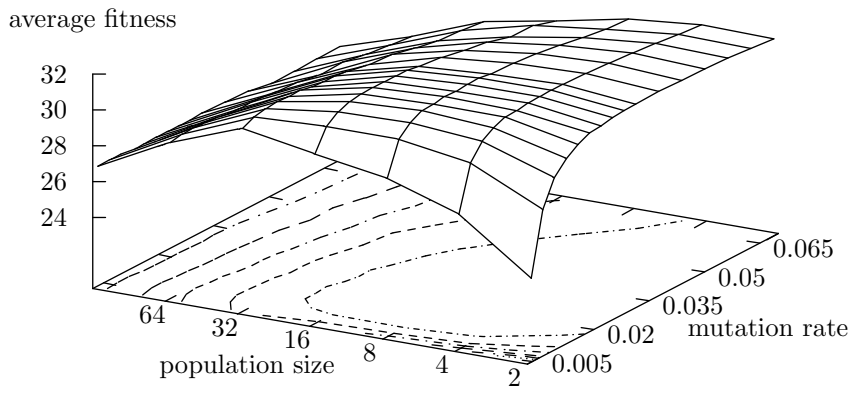

Figure 2. Interdependence between mutation rate and population size: The graph shows the average fitness with respect to the optimization of the Counting-Ones problem

Basically an algorithm should maintain a balance between certain aspects concerning the search process that might help in defining the proposed network. Two important aspects are the exploration of the search space and the exploitation of promising regions [12]. Mutation, recombination, and selection must hold the balance between both aspects. The operators define all possible search paths and must guarantee that there are individuals sufficing either aspect. This is the basis for a balance but selection has to focus the search process in a balanced way and should also preserve the acquisitions of past generations. Because of the dynamics of the evolutionary search process the emphasis on exploration and exploitation changes during search. Exploration could be more important at the beginning to find a tolerable solution, but towards the end of search exploitation should fine-tune the so-far best candidate solution. In addition, the algorithm's parameters determine two additional aspects that play also an important role during the search process: diversity in the population [3] and progress rate [4]. Exploration, exploitation, and diversity are examined more closely with respect to their impact in the following sections on selection and evolutionary operators.

3. Selection

This section examines the effects of selection. Selective pressure is essential for successful optimization since it focuses search on better candidate solutions. Selection does not introduce new candidate solutions and is usually a diversity diminishing and contracting operation. Essentially, it changes the frequencies of the individuals in the population (or of their inheritable genetic information) on the basis of evaluations using the objective function.

Selection is primarily responsible for the dynamics in the evolutionary search process. However the complex systems emerging by those dynamics and the resulting flow of genes is barely understood. This is even more true since there are many manifold selection operators in evolutionary algorithms. In the bottom line, it is known that high selective pressure increases the exploitation within the evolutionary search process and low selective pressure stresses the exploration.

Selection may occur in an evolutionary algorithm as parental selection or as environmental selection. Parental selection determines the number of offspring for each individual and environmental selection

determines those individuals that may be considered as parents in the next generation. Often it is sensible to create selective pressure only at one position in the algorithm. Then, the other position may be replaced by an identity mapping or selection with uniform probability. In the case of environmental selection, overlapping generations are an additional common theme in evolutionary algorithms. There, the next pool of possible parents is determined from the last parental population as well as the created offspring. Possible parameters are the selection or replacement strategy and the degree of overlap [36, 39].

Selection mechanisms may be distinguished by the usage of random influence. On the one hand there is probabilistic selection, where each individual is selected according to a certain probability, and on the other hand there is deterministic selection, where the fitness values define clearly which individuals will be selected. However if probabilistic techniques are used for parental selection each individual may be chosen more than once, i.e. the selection introduces duplicates. This is usually not the case in deterministic selection. Note, that duplicate-free probabilistic selection methods [14] behave rather like deterministic methods.

environmental ↓ parental →	uniform	identity	probab.
identity	no selective pressure		GA
dupl. free probab.	?	×	?
(overlap.)	?	EP (new)	?
probabilistic	Lemma 3.1	≡ GA	?
(overlap.)	(Lemma 3.1)	SA, TA, ...	stdstGA
deterministic	ES	×	?
(overlap.)	ES	EP (old)	stdstGA

Table 1. Possible combinations of selection operators: SA = simulated annealing, TA = threshold algorithm, stdstGA = steady state GA, × = inconstant population size. Lemma 3.1 makes a statement on uniform parental selection. Those combinations marked as restrictive use only a fraction of the parental population.

Table 1 shows the possible setups to create selective pressure. Since duplicate free probabilistic selection and deterministic selection reduce the number of individuals and the identity mapping keeps the number constant, certain combinations are not possible with a constant population size. They are marked by the symbol "×". Also the selection schemes of prominent evolutionary algorithms are included.

If an identity mapping is used at either position there is no difference where we use the other selection method (shown in the entry of "≡ GA"). Also, uniform environmental selection seems to make no sense since this simply means that with a considerable probability some newly created individuals are not considered independent of their fitness. There are no findings on the combinations marked by a "?" – probably two true selection operators create too much selective pressure, but those techniques are sometimes used with constraint and multiobjective optimization or in steady-state algorithms. The usage of uniform selection is examined in the following lemma.

Lemma 3.1. A population of size 3 (with no duplicates) is given. Then, a probabilistic selection inducing duplicates as parental selection has a smaller probability to choose two equal individuals than

uniform selection from a population of size 3 created by the same probabilistic selection as environmental selection.

Proof:
Let pr_A, pr_B, and pr_c be the probabilities to choose the individuals ($pr_A + pr_B + pr_C = 1.0$). Then, for mere parental selection the probability to choose two identical individuals equals $P_{parent} = pr_A^2 + pr_B^2 + pr_C^2$. If used as environmental selection, the probability to get a new population containing three identical individuals A is $pr_{AAA} = pr_A^3$, for a population with two individuals A it is $pr_{AA} = 3pr_A^2(1 - pr_A)$, and analogously for individuals B and C. The total probability to choose two identical individual results in

$$\begin{aligned} & pr_{AAA} + pr_{BBB} + pr_{CCC} + \\ & \left(\frac{2}{3}\frac{2}{3} + \frac{1}{3}\frac{1}{3}\right)(pr_{AA} + pr_{BB} + pr_{CC}) + \frac{1}{3}pr_{ABC} \\ \geq \; & \frac{1}{3}\left(5pr_A^2 - 2pr_A^3 + 5pr_B^2 - 2pr_B^3 + 5pr_C^2 - 2pr_C^3\right) \\ \geq \; & pr_A^2 + pr_B^2 + pr_c^2 \; = \; P_{parent}. \end{aligned}$$

The latter transformation holds since for all $x \in \{A, B, C\}$: $0 \leq pr_x \leq 1$. □

This lemma shows that uniform parental selection should be avoided. Deterministic environmental selection combined with uniform parental selection (like in ES) makes sense if a recombination is used and the uniform sampling generates various different pairs of individuals. However, the number of chosen individuals must be considerably high to consider all available individuals. Otherwise the unconsidered individuals should have an opportunity to survive as is usually the case in overlapping populations.

The different selection schemes may further be compared by their selection pressure. One appropriate measure is the selection intensity which is defined as the selection differential defined by the average population fitness of two successive generations, normalized by the fitness standard deviation before selection: $I = \frac{\bar{f}^{t+1} - \bar{f}^t}{\sigma_t}$. The following lemma compares the fitness proportional selection, a ranking selection where the best individual has probability $\frac{2-\epsilon}{\mu}$ (μ is the population size) and the probabilities are decreasing linearly to the worst probability $\frac{\epsilon}{\mu}$, and deterministic selection where the population shrinks from λ individuals to μ individuals.

Lemma 3.2. The following selection intensities hold.

$$I_{prop.} = \frac{\sigma}{\bar{f}(t)} \tag{1}$$

$$I_{ranking} = \frac{1 - \epsilon}{\sqrt{\pi}} \tag{2}$$

$$I_{determ.} \approx \frac{\lambda}{\mu}\phi(\mathbf{\Phi}^{-1}\left(1 - \frac{\mu}{\lambda}\right)) \tag{3}$$

where $\phi(x) = \frac{1}{\sqrt{2\pi}}\exp(-\frac{x^2}{2})$ and $\Phi = \int_{-\infty}^{x}\phi(t)dt$. In addition, (2) and (3) require a distribution of the fitness values according to a Gaussian pdf.

For the proof of Lemma 3.2 refer to [26] for (1), to [8] for (2), and to [2] for (3).

This shows in particular that selection proportional to the individuals' fitness depends directly on the constitution of the population. This may lead to comparable little selection pressure if the diversity in the population decreases or the average fitness gets rather high towards the end of an optimization. This is not the case with both other selection schemes: here the selection pressure is independent of the current population. In the case of linear ranking, the parameter ϵ may be used to adjust the selection intensity and, in the case of deterministic selection, the pressure depends directly on the sizes of parent and offspring populations. Note, that depending on the selection mechanism different interdependencies are active in the required network.

Because of its contracting character the selection operator is usually not able to maintain a large number of individuals with similar good fitness values. To prevent the concentration of the population in one point of the search space, additional techniques are necessary, e.g. fitness sharing [20, 24] where the fitness of individuals in clusters is decreased. In addition recombination may be restricted to individuals within one cluster. In [20] the similarity of individuals is based on a distance metric leading to an expensive operation. Alternatively the assignment of an individual to a cluster can be attached to each individual as additional information [40]. This enables the algorithm to self-adapt the technique to the characteristics of the problem landscape.

4. Evolutionary Operators

This section examines the evolutionary operators with respect to the network of interdependencies. Evolutionary operators are the counterpart to selection – they create new individuals and, as a consequence, determine the possible directions search can take. Most evolutionary algorithms use two different kinds of operators for the production of new individuals: unary and binary (or more general k-ary) operators, namely mutation and recombination. The different responsibilities of an ensemble of operators are exploration and exploitation as already mentioned above, but also the reachability of all points in the search space and the maintenance of the diversity.

A mutation operator takes one already existing individual and varies it. Bigger changes to the phenotype support the exploration, small (local) changes to the phenotype promote exploitation.

The interplay of mutation and selection is the foundation of Darwinism and is supposed to lead to continual improvement. How well a mutation operator supports such a local search process can be examined for finite search spaces by the analysis of operator-induced neighborhood landscapes [25], i.e. a graph structure on the search space where connections between candidate solutions are induced by the variation of the operator. On such a landscape the adaptedness of the operator to a problem can be measured by the number of mutation-induced local optima. The number of these optima can often be reduced by mutations making little changes concerning the specific problem (cf. [15]). But often the landscape is determined by a solution encoding, i.e. a mapping from a genotype space to the search space, and the mutation operator is working on the genotype. Such an approach may introduce even more local optima. However, smart choices of decoder functions can preserve properties of the search space and thus induce less new local optima, e.g. Rana and Whitley examined standard binary encodings and Gray encoding for three problems and found that Gray encodings can be expected to introduce less local optima [32]. In particular, Gray encodings embed phenotypic neighborhood relations into the genotypic neighborhood landscape.

Where the interaction of small changes and selection leads to local search, an operator with a very local behavior might be trapped in local optima such that the potential reachability of all points in the search space gets lost. However, a high diversity of the individuals in a population counteracts those effects. Also the recombination requires often a certain degree of diversity which is discussed in the next paragraph. As a consequence, diversity must be maintained by one of the operators – if there is no other specific operator available for this task (e.g. the hypermutation in dynamic optimization) the normal mutation operator has to maintain the diversity. Usually, sucessfully applied mutation operators like ES mutation or GA mutation balance small and large changes to a certain degree.

Recombination operators get two (or more) individuals to produce at least one new offspring. Their intuitive working principle is the recombination of different parts of various individuals. This does not only enable the formation of new innovative combinations of existing individuals but can also be understood as a parallel search in hyperplanes of the search space according to early genetic algorithm theory [23, 18]. This issue is discussed in more detail in Section 5. However, recombination depends on a high degree of diversity in the population such that the parallel search is based on sufficient distinct samples in the search space. If diversity is lacking, many individuals are very similar to each other. Then, another possible aspect of recombination operators can become active: operators with a rather contracting behavior may remove single huge mutation errors towards the end of optimization by stochastic effects (genetic repair principle) – an effect that has been discovered for intermediate recombination on real valued search spaces by Beyer who also transfered it to discrete search spaces [6].

Lemma 4.1. Let $\mathcal{S} = \mathbb{R}^n$ be a real valued search space with optimum $\mathbf{0} \in \mathcal{S}$. Then, a mutation of individual $x \in \mathcal{S}$ may be written as $M(x) = x + a\ \overrightarrow{x\mathbf{0}} + y$ where y is orthogonal to the optimal search direction $\overrightarrow{x\mathbf{0}}$. Then, intermediate recombination with r parents reduces the deviation from the optimal search direction by the factor $\frac{1}{r}$.

For the proof of Lemma 4.1 refer to [5].

Sometimes recombination operators are also used to identify new, promising areas within the search space using the current individuals' fitness information. Those operators have an extrapolating character and are sometimes referred to as directed recombination or implicit mutation.

The parallel processing and the local neighborhood search have been antipodal main motivations for the different paradigms since the former requires a rather fine-scaled coding and the latter a rather coarse-grained representation preserving a phenotypic neighborhood. However, the principles may be understood in a more general sense. Combining those principles, interdependencies among the operators become visible. On the one hand, rather big mutations increase the diversity and promote a more effective sampling in parallel hyperplane search using recombination. On the other hand, small mutations decrease diversity leading to further damping of the mutation by contracting recombinations. This illustrates the difficulty in balancing evolutionary algorithms.

Optimal parameter settings for operators have been one focus of research for many years. However, as the discussion above already implies, it has been shown that no mutation operator is optimal during the complete search process with respect to a point-wise measure combining the success rate of the operator $s_M(x) = \frac{\#\ better\ offspring\ M(x)}{\#\ all\ offspring\ M(x)}$ and the average improvement $i_M(x)$ over all possible improving mutations.

Lemma 4.2. Let $\mathcal{S}$ be an arbitrary search space and $Imp_M(x) = s_M(x)i_M(x)$ be the relative expected improvement. Consider two operators M_1 and M_2 with comparable offspring distributions (cf. [42]) but different variance in offspring fitness $V[M_1(x)] < V[M_2(x)]$. Then, there exist fitness ranges R_1 close to the optimum and R_2 close to the center of the fitness values in the search space for which

$$\begin{aligned} Imp_{M_1}(x) > Imp_{M_2}(x) \quad & \text{if } f(x) \in R_1 \text{ and} \\ Imp_{M_2}(x) > Imp_{M_1}(x) \quad & \text{if } f(x) \in R_2. \end{aligned}$$

For the proof of Lemma 4.2 refer to [42, 44]. This result is strongly related to the shift from exploration to exploitation during search, mentioned in Section 2. As a consequence, adaptation of the operators during optimization is necessary. The basic techniques for such a parameter control are predetermined adaptation (i.e. a fixed schedule of parameter values), adaptation on the population level e.g. on the basis of statistics, and self-adaptation for each candidate solution where "good" parameters are evolved in each individual by means of the evolutionary process [11].

Where we discussed so far the two kinds of operators separately there is work which stresses rather the similarities, e.g. theoretical findings where recombination operators emulate the behavior of self-adaptive mutation [7] and the viewpoint of recombination as little phenotypic variation [16]. These are hints that the operators are probably more interchangeable than is assumed today.

5. Schemata, Formae, and Price

One main motivation for and explanation of genetic algorithms has been the controversially discussed schema theorem [22] which predicts the growth of certain genotypic characteristics within the population by analyzing the parallel processing in the hyperplanes. The idea is that the population searches in parallel in many different hyperplanes of the search space for useful building blocks which are used in the further processing of the algorithm to construct competitive complex solutions. We discuss the schema theorem within the more general formae theory [30] and examine how these findings may be interpreted within the network of local interactions.

The complete search space may be divided into several equivalence classes by considering certain genotypic (or phenotypic) characteristics. Such an equivalence class is called a forma. Examples are certain bit patterns at fixed positions in a bit string – such a forma is called schema – or the occurrence of certain edges in a candidate solution for an instance of the traveling salesman problem. The following lemma states how the occurrence of a forma in the population is affected by one generation.

Lemma 5.1. Let $P^{(t)} = \left\langle A^{(t,i)} \right\rangle_{1 \le i \le n}$ be a population at generation t. Let v be a forma. Then, the sequential application of parental selection Sel, recombination R, and mutation M changes the frequency of instances of the forma in the population as follows,

$$E(v, t+1) \quad \geq \quad E_{Sel}(v, P^{(t)}) \left(1 - P_{M,v} - P_{R,v}\right), \tag{4}$$

where $E_{Sel}(v, P)$ is the expected number that instances of v are chosen as parent, $P_{M,v}$ is the probability that M destroys the forma property, and $P_{R,v}$ analogously for recombination R with an arbitrary partner.

This lemma can be directly derived from the sequential application of the operators [22].

One instance of this lemma is the schema theorem for binary strings and standard genetic algorithms where, as property, predefined values are required at certain defining bit positions – the other positions may be filled with arbitrary values. Then, the theorem predicts an above average growth for those schemata with above average fitness and few defining positions that are arranged close together. Lemma 5.1 is relatively weak since only the preservation and not the creation of schema instances is considered. Recently, a number of schema theorems have been shown which predict the growth in the population in a more exact way [41]. These schema theorems are exact models of the search space dynamics.

When the schema theory came up, the conclusion was drawn that for all problems a coding using an alphabet of low cardinality should be chosen [19] in order to maximize the number of hyperplanes that are searched. But the "No Free Lunch" theorems, mentioned above, imply that a good encoding, creating sensible schemata or building blocks for a set of problems, will not work for another set of problems. As a consequence the encoding or representation must be chosen problem specific. The general notion of formae provides certain principles [30, 31] how the ensemble of representation, mutation, and recombination should be chosen to get the positive effects of building block growth.

First of all, if redundancy in the encoding of the problem cannot be avoided, the redundant candidate solutions with high fitness should be gathered in the same formae. Then, those characteristics responsible for high fitness are identical and may be identified by the search algorithm. Also, the fitness values of formae with low precision, i.e. covering big fractions of the search space, should be correlated. The reason for this principle is that low precision formae may be understood as building blocks which may be combined to more complex candidate solutions – but only if the average fitness of instances of such a forma is representative, which should be guaranteed by the correlation, they can spread their genes. In order to guarantee a proper refinement within the search process and combination of the different characteristics, the set of all considered formae should be closed under intersection. The recombination operator should respect any common forma of two parents and produce only offspring that are also instances of this forma. This guarantees that common properties are inherited by the offspring. Furthermore, if two formae are compatible, i.e. there exists an individual which is instance of both formae, recombination should also be possible for any pair of individuals where one individual is instance of the first forma and the other of the second to create an individual which is element of both formae. This last principle guarantees that it is actual possible to use the refinement of formae discussed above and build up candidate solutions using those building blocks. In order to have a mere combining recombination it is also possible to require that each atomic forma (i.e. it cannot be split up into several formae) within an offspring is already contained in at least one parent. Otherwise the recombination performs implicit mutations. However, if all those design principles are fulfilled each operator gets a strict role assigned. Recombination is supposed to be able to combine and conserve positive characteristics and mutation guarantees that each candidate solution in the search space is reachable. Note, that this is only one possible setup for the interaction of operators.

However, all those schema (or forma) theorems make only a meaningful statement if the above average fitness of a forma is still present in the new offspring. That means it assumes that there is a high correlation between the parents' and the offspring's fitness. This correlation is made explicit as a prerequisite in Price's Covariance and Selection Theorem introduced to evolutionary computation by Altenberg [1]. From this theorem, not only a variant of the schema theorem may be derived but also the following theorem which gives important insight into the internal processing of evolutionary algorithms. It quantifies the change in the fitness distribution directly using the recombination operator and the kind

of schemata processed by this operator, the covariance concerning the fitness between parents and offspring, the frequency of the schemata in the population, and the average fitness of schemata and the total population.

Theorem 5.1. ("Missing" schema theorem)
The change in the fitness distribution from generation t to generation $t+1$ can be measured by the percentage $\overline{QuInd}_w$ of individuals better than quality threshold w. ($QuInd_w(A)$ is the indicator function whether $F(A) > w$). Then, for an algorithm with fitness proportional selection, a mere combining recombination, and no mutation the change results in

$$\overline{QuInd}_w^{(t+1)} - \overline{QuInd}_w^{(t)} =$$
$$\sum_{\theta \in \mathcal{P}(\{1,\ldots,l\})} p_\theta \left(Cov\left[QuInd_w(A), \frac{\overline{F}_{H_\theta(A)} \overline{F}_{H_{\widetilde{\theta}}(A)}}{\overline{F}^2} \right] - \right.$$
$$\left. \sum_{A \in \mathcal{G}} (p_A - p_{H_\theta(A)} p_{H_{\widetilde{\theta}}(A)}) (QuInd_w(A) - \overline{QuInd}_w) \frac{\overline{F}_{H_\theta(A)} \overline{F}_{H_{\widetilde{\theta}}(A)}}{\overline{F}^2} \right)$$

where the mask θ is an index set determining the genes from the first parent and p_θ is the probability that mask θ is used by the operator. $\widetilde{\theta} = \{1, \ldots, l\} \setminus \theta$ is the index set of the genes from the second parent. $H_\theta(A)$ and $H_{\widetilde{\theta}}(A)$ are the schemata describing possible parents for the creation of A, p_H denotes the respective frequencies of instances of the schema in the population, and $\overline{F}_{H_\theta(A)}$ and $\overline{F}_{H_{\widetilde{\theta}}(A)}$ are the average fitness values of those instances.

For a proof refer to [1].

Several conclusions can be drawn from this theorem. First, contrary to the schema theorem not all schemata are relevant to the search process but only those which are actually created by the functioning of the recombination operator (quantified in the probability p_θ of mask θ). Moreover, those schemata occur in complementary pairs ($H_\theta(A)$ and $H_{\widetilde{\theta}}(A)$) covering all indices of the individuals. In order to get a considerable effect on the fitness distribution, there should be a positive covariance between the average fitness of these schemata and the quality of the created offspring. Furthermore, it makes the coherence between the schemata's and their offspring's occurrence and the offsprings above or below average quality obvious with respect to the fitness distribution change. As a consequence this theoretical result may be used to examine recombination operators with respect to their effects on the search process and to tune operators by increasing the probabilities of the masks θ with a positive effect on the fitness distribution. The effect of recombination masks can be determined statistically.

6. Modeling the search process

All previous sections provide only a microscopic insight into the working principles of evolutionary algorithms. Although these principles are important for the understanding and the design of effective algorithms, a macroscopic view is also of interest where the complete search process is the focus of an examination.

One common technique is the use of Markov models [34, 35]. They enable statements on the limit behavior, e.g. the convergence to the global optimum, as well as on finite time behavior, e.g. convergence time. However, the latter results have only been shown for rather simple problems and algorithms.

Another technique to analyze genetic algorithms macroscopically is given by statistical mechanics models [29]. In this model the population dynamics are described by few macroscopic quantities like cumulants of the fitness distribution of the population assuming that the remaining microscopic degrees of freedom can be averaged out. In formulating a genetic algorithm as a statistical mechanics model it is possible to predict the average convergence time and the average fitness of the final population at least for simple problems.

A third exact modeling technique are the already mentioned exact schema theorems [41, 28, 27] that can be used to give exact predictions on how the number of schemata in the population changes.

Apparently, those techniques help to increase the understanding of evolutionary algorithms – especially for very simple algorithms and problems where interesting statements are possible. Then, their global perspective can help to evaluate the complete algorithm. But, the omission of complex problems and the restriction to the complete ensemble of operators and selection helps little for the development of the proposed networked understanding. In addition, they cannot guide the design of new algorithms which makes them also less interesting to the practitioner.

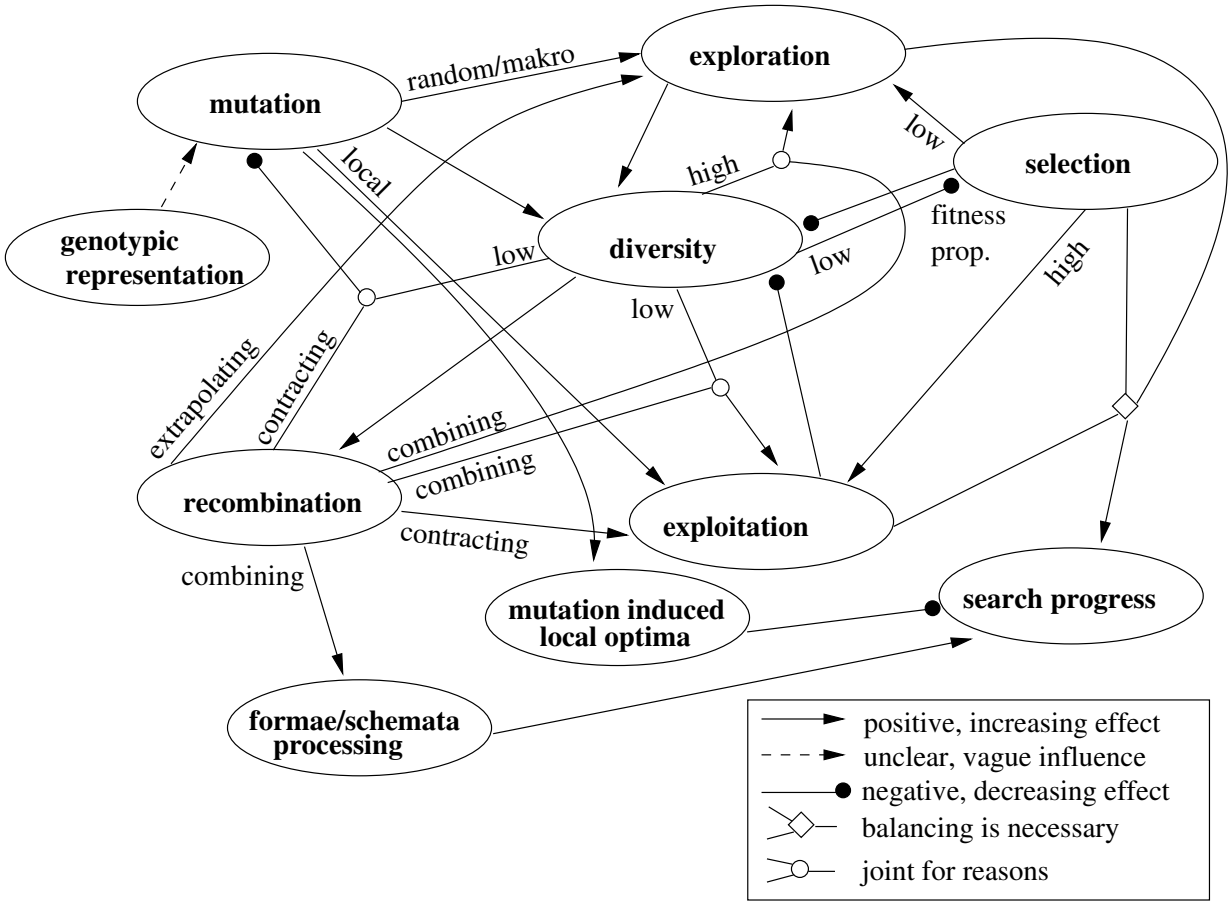

Figure 3. Exemplary network of relations and interactions unfolded in this article. The semantic of the lines is explained in Table 2 in detail.

source	target	description
genotype	mutation	possible operating schemes of the mutation are influenced by the genotype
mutation	exploration	rather random mutations support exploration
mutation	exploitation	phenotypic local mutations support exploitation
mutation	diversity	the strength of the mutation has a direct effect on the diversity
mutation	local optima	very local mutation operators preserve phenotypic local optima, discontinuous operators might induce even more local optima
recombination	exploration	extrapolating operators support exploration
recombination	exploitation	contracting operators support exploitation
div./recomb.	mutation	if diversity is low a contracting recombination operator may dampen huge mutation outliers (genetic repair theory)
diversity	recombination	high diversity supports manifold recombination
selection	exploration	low selective pressure supports exploration
selection	exploitation	high selective pressrue supports exploitation
selection	diversity	selection decreases diversity usually
div./recomb.	exploration	combining recombination supports exploration if the diversity is high
div./recomb.	exploitation	combining recombination supports exploitation if the diversity is low
exploration	diversity	high exploration increases the diversity
exploitation	diversity	high exploitation decreases the diversity
diversity	selection	low diversity (implying similar fitness values) has usually a negative effect on fitness proportional selection
recombination	formae proc.	fulfilled forma design rules support an effective formae/schemata processing
formae proc.	search progress	if forma processing works according to the schema theorem and the missing schema theorem, positive effects concerning the search progress can be assumed
local optima	search progress	many local optima hinder the search progress
expl./expl./sel.	search progress	good balance between exploration, exploitation, and selective pressure is essential for the search progress

Table 2. Additional remarks to the interactions in Figure 3.

7. Discussion

During the last years the isolated view of the paradigms has been replaced by a general understanding of evolutionary computation. This trend continues with small steps towards a unified theory. For a long time the major working principles, parallel search in hyperplanes by recombination and local improvement by mutation, have been antagonistic philosophies for using evolutionary algorithms. In the first case an encoding with an alphabet of small cardinality was postulated [19] where the latter philosophy requires rather problem-given representations [15]. A unified theory, integrating both working principles, might overcome those restricted viewpoints. This could also lead to a better understanding of the nuances and trade-offs between those extremes.

This article reviews a number of prominent theoretical results and working principles with the goal to identify local interactions. In their sum, they result in a networked understanding of evolutionary algorithms. The identified network is shown in Figure 3 and explained in detail in Table 2. Due to the rather superficial approach within this article, many details are missing and certain interactions are rather vague. There are also many important factors which are still unknown – leaving certain gaps within the network.

Although the approach of this article is rather superficial, the resulting network is a good starting point for reasoning on how operators influence each other. The integration of different views within one network helps to hypothesize on an abstract level on possible interactions within the search dynamics. Recently, an approach has been proposed how the presented network may be extended to a qualitative model of typical evolutionary algorithm dynamics [43]. This qualitative model has the valuable advantage that problem characteristics may be considered which is necessary to support the development of problem-specific evolutionary algorithms. In [43] a representation for the qualitative model as well as the empirical derivation of one interaction from experimental data has been shown.

The presented networked understanding of evolutionary algorithms appears to be a promising contribution to integrate both research and development of evolutionary algorithms. The next years will show how the presented model and the qualitative model in [43] "evolve". The goal is the establishment of a top-down design process supported by theoretical findings and empirical investigations.

References

[1] Altenberg, L.: The Schema Theorem and Price's Theorem, *Foundations of Genetic Algorithms III* (L. D. Whitley, M. D. Vose, Eds.), Morgan Kaufmann, San Francisco, CA, 1995.

[2] Bäck, T.: Generalized Convergence Models for Tournament- and (μ, λ)-Selection, *Proc. of the Sixth Int. Conf. on Genetic Algorithms* (L. J. Eshelman, Ed.), Morgan Kaufmann, San Francisco, CA, 1995.

[3] Bedau, M. A., Ronneburg, F., Zwick, M.: Dynamics of Diversity in an Evolving Population, *Parallel problem solving from nature 2* (R. Männer, B. Manderick, Eds.), North-Holland, Amsterdam, 1992.

[4] Beyer, H.-G.: *Towards a Theory of Evolution Strategies: Results from the N-dependent (μ, λ) and the Multi-Recombinant $(\mu/\mu, \lambda)$ Theory*, Technical Report SYS-5/94, Systems Analysis Research Group, University of Dortmund, Dortmund, Germany, 1994.

[5] Beyer, H.-G.: Toward a Theory of Evolution Strategies: On the Benefits of Sex—the $(\mu/\mu, \lambda)$ Theory, *Evolutionary Computation*, **3**(1), 1995, 81–111.

[6] Beyer, H.-G.: An alternative explanation for the manner in which genetic algorithms operate, *BioSystems*, **41**, 1997, 1–15.

[7] Beyer, H.-G., Deb, K.: On the Desired Behaviors of Self-Adaptive Evolutionary Algorithms, *Parallel Problem Solving from Nature – PPSN VI* (M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, H.-P. Schwefel, Eds.), Springer, Berlin, 2000.

[8] Blickle, T., Thiele, L.: A Comparison of Selection Schemes Used in Evolutionary Algorithms, *Evolutionary Computation*, **4**(4), 1997, 361–394.

[9] Culberson, J. C.: On the Futility of Blind Search: An Algorithmic View of "No Free Lunch", *Evolutionary Computation*, **6**(2), 1998, 109–127.

[10] Deb, K., Agrawal, S.: Understanding Interactions among Genetic Algorithm Parameters, in: *Foundations of Genetic Algorithms 5* (W. Banzhaf, C. Reeves, Eds.), Morgan Kaufmann, San Francisco, CA, 1999, 265–286.

[11] Eiben, A. E., Hinterding, R., Michalewicz, Z.: Parameter Control in Evolutionary Algorithms, *IEEE Trans. on Evolutionary Computation*, **3**(2), 1999, 124–141.

[12] Eiben, A. E., Schippers, C. A.: On Evolutionary Exploration and Exploitation, *Fundamenta Informaticae*, **35**, 1998, 35–50.

[13] English, T. M.: Some Information Theoretic Results on Evolutionary Optimization, *1999 Congress on Evolutionary Computation*, IEEE Service Center, Piscataway, NJ, 1999.

[14] Fogel, D. B.: *Evolutionary computation: toward a new philosophy of machine intelligence*, IEEE Press, New York, 1995.

[15] Fogel, D. B.: An Overview of Evolutionary Programming, in: *Evolutionary Algorithms* (L. D. Davis, K. De Jong, M. D. Vose, L. D. Whitley, Eds.), Springer, New York, 1999, 89–109.

[16] Fogel, D. B., Atmar, J. W.: Comparing Genetic Operators with Gaussian Mutations in Simulated Evolutionary Processes Using Linear Systems, *Biological Cybernetics*, **63**(2), 1990, 111–114.

[17] Fogel, L. J., Owens, A. J., Walsh, M. J.: *Artificial Intelligence Through simulated Evolution*, John Wiley and Sons, New York, 1966.

[18] Goldberg, D. E.: *Genetic algorithms in search, optimization, and machine learning*, Addison-Wesley, Reading, MA, 1989.

[19] Goldberg, D. E.: Zen and the Art of Genetic Algorithms, *Proc. of the Third Int. Conf. on Genetic Algorithms* (J. D. Schaffer, Ed.), Morgan Kaufmann, San Mateo, CA, 1989.

[20] Goldberg, D. E., Richardson, J.: Genetic Algorithms with Sharing for Multimodal Function optimization, *Genetic algorithms and their applications: Proc. of the Second Int. Conf. on Genetic Algorithms* (J. J. Grefenstette, Ed.), Lawrence Erlbaum Assoc., Hillsdale, NJ, 1987.

[21] Holland, J. H.: A new kind of turnpike theorem, *Bulletin of the American Mathematical Society*, **75**(6), 1969, 1311–1317.

[22] Holland, J. H.: *Adaptation in Natural and Artifical Systems*, University of Michigan Press, Ann Arbor, MI, 1975.

[23] Holland, J. H.: *Adaptation in Natural and Artifical Systems*, MIT Press, Cambridge, MA, 1992.

[24] Horn, J.: Resource-Based Fitness Sharing, *Parallel Problem Solving from Nature - PPSN VII* (J. J. Merelo Guervós, P. Adamidis, H.-G. Beyer, J.-L. Fernández-Villacañas, H.-P. Schwefel, Eds.), Springer, Berlin, 2002.

[25] Jones, T.: *Evolutionary Algorithms, Fitness Landscapes and Search*, Ph.D. Thesis, The University of New Mexico, Albuquerque, NM, 1995.

[26] Mühlenbein, H., Schlierkamp-Voosen, D.: Predictive Models for the Breeder Genetic Algorithm: I. Continuous Parameter Optimization, *Evolutionary Computation*, **1**(1), 1993, 25–49.

[27] Poli, R.: Recursive Conditional Schema Theorem, Convergence and Population Sizing in Genetic Algorithms, in: *Foundations of Genetic Algorithms 6* (W. N. Martin, W. M. Spears, Eds.), Morgan Kaufmann, San Francisco, 2001, 143–163.

[28] Poli, R., McPhee, N. F.: Exact Schema Theorems for GP with One-Point and Standard Crossover Operating on Linear Structures and Their Application to the Study of the Evolution of Size, *Genetic Programming: 4th European conference* (J. Miller, M. Tomassini, P. L. Lanzi, C. Ryan, A. G. B. Tettamanzi, W. B. Langdon, Eds.), Springer, Berlin, 2001.

[29] Prügel-Bennett, A., Shapiro, J. L.: An Analysis of Genetic Algorithms Using Statistical Mechanics, *Physical Review Letters*, **72**, 1994, 1305–1309.

[30] Radcliffe, N. J.: Equivalence Class Analysis of Genetic Algorithms, *Complex Systems*, **5**, 1991, 183–205.

[31] Radcliffe, N. J., Surry, P. D.: Fitness variance of formae and performance prediction, *Foundations of Genetic Algorithms III* (L. D. Whitley, M. D. Vose, Eds.), Morgan Kaufmann, San Mateo, CA, 1995.

[32] Rana, S., Whitley, L. D.: Search, Binary Representations and Counting Optima, in: *Evolutionary Algorithms* (L. D. Davis, K. De Jong, M. D. Vose, L. D. Whitley, Eds.), Springer, New York, 1999, 177–189.

[33] Rechenberg, I.: *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*, frommann-holzbog, Stuttgart, 1973, German.

[34] Rudolph, G.: *Convergence properties of evolutionary algorithms*, Kovač, Hamburg, 1997.

[35] Rudolph, G.: Finite Markov Chain Results in Evolutionary Computation: A Tour d'Horizon, *Fundamenta Informaticae*, **35**, 1998, 67–89.

[36] Sarma, J., De Jong, K.: Generation gap methods, in: *Handbook of Evolutionary Computation* (T. Bäck, D. B. Fogel, Z. Michalewicz, Eds.), Institute of Physics Publishing and Oxford University Press, Bristol, New York, 1997, C2.7:1–5.

[37] Schumacher, C., Vose, M. D., Whitley, L. D.: The No Free Lunch and Problem Description Length, *GECCO 2001: Proc. of the Genetic and Evolutionary Computation Conf.* (L. Spector, E. D. Goodman, Eds.), Morgan Kaufmann, San Francisco, 2001.

[38] Schwefel, H.-P.: *Evolution and optimum seeking*, Wiley, New York, 1995.

[39] Smith, J. E., Vavak, F.: Replacement Strategies in Steady State Genetic Algorithms: Static Environments, in: *Foundations of Genetic Algorithms 5* (W. Banzhaf, C. Reeves, Eds.), Morgan Kaufmann, San Francisco, CA, 1999, 219–233.

[40] Spears, W. M.: Simple Subpopulation Schemes, *Evolutionary Programming: Proc. of the Third Annual Conf.* (A. V. Sebald, L. J. Fogel, Eds.), World Scientific Press, Singapore, 1994.

[41] Stephens, C. R., Waelbroeck, H., Aguirre, R.: Schemata as Building Blocks: Does Size Matter?, in: *Foundations of Genetic Algorithms 5* (W. Banzhaf, C. Reeves, Eds.), Morgan Kaufmann, San Francisco, CA, 1999, 117–133.

[42] Weicker, K., Weicker, N.: Locality vs. Randomness – dependence of operator quality on the search state, in: *Foundations of Genetic Algorithms 5* (W. Banzhaf, C. Reeves, Eds.), Morgan Kaufmann, San Francisco, CA, 1999, 147–163.

[43] Weicker, K., Weicker, N.: Towards Qualitative Models of Interactions in Evolutionary Algorithms, in: *Foundations of Genetic Algorithms VII* (K. A. De Jong, R. Poli, J. E. Rowe, Eds.), Morgan Kaufmann, San Francisco, 2003, 365–382.

[44] Weicker, N.: *Qualitative No Free Lunch Aussagen für Evolutionäre Algorithmen*, Cuvillier, Göttingen, 2001.

[45] Wolpert, D. H., Macready, W. G.: No Free Lunch Theorems for Optimization, *IEEE Trans. on Evolutionary Computation*, **1**(1), 1997, 67–82.