Convergence to Global Optima for Genetic Programming Systems with Dynamically Scaled Operators

Lothar M. Schmitt
The University of Aizu
Aizu-Wakamatsu City 965-8580, Japan
L@LMSchmitt.de

Stefan Droste
Universität Dortmund (LS Informatik 2)
44221 Dortmund, Germany
Stefan.Droste@udo.edu

ABSTRACT

This work shows asymptotic convergence to global optima for a family of dynamically scaled genetic programming systems where the underlying population consists of a fixed number of creatures (individuals) each of arbitrary size. The genetic programming systems use common mutation and crossover operators as well as fitness-proportional selection. In addition, the mutation and crossover rates are annealed to zero in predefined fashion over the course of the algorithm, and power-law scaling is used for the (possibly population-dependent) initial fitness function with (unbounded) logarithmic growth in the exponent.

We assume that a set of globally optimal creatures for the optimization problem instance exists. In addition, it is assumed that the ratio of the best fitness of globally optimal creatures *vs.* the fitness of other creatures is greater or equal a constant $\rho>1$ in any population they jointly reside in. We discuss how both conditions can usually be satisfied in application settings. Under the above conditions, a selected, traceable sequence of probability distributions over the possible states of the properly scaled genetic programming system converge in time towards the convex set of probability distributions over uniform populations that contain only globally optimal creatures.

Categories and Subject Descriptors

I.2.2 [**Automatic Programming**]: program synthesis.

General Terms

Algorithms

Keywords

Genetic programming, convergence to global optima.

1. INTRODUCTION

1.1. Principles of genetic programming. The term "*evolutionary algorithm*" denotes a class of randomized optimization heuristics based on principles held to be valid for natural evolution. Evolutionary algorithms employ a sequence of populations $p(t)$, $t\in\mathbb{N}$, where each $p(t)$ is a family of candidate solutions for the optimization problem instance, apply genetic operators like mutation and recombination (crossover) to generate new candidate solutions in the population $p(t)$, and tend to select the fittest candidate solutions in the current population to form the next population $p(t+1)$ with the expectation that some $p(t)$ will eventually contain an optimum. Candidate solutions are usually called creatures or individuals. Consult [3] for a detailed overview over such methods.

There are four main paradigms of evolutionary algorithms: *evolution strategies* [22], *evolutionary programming* [6], *genetic algorithms* [7] and *genetic programming* [9]. Since genetic programming was developed later than the other paradigms, it has borrowed many ideas from these earlier approaches. In particular, the similarity between a genetic algorithm and a genetic programming system is shown in [9, p. 29: Fig. 3.1] *vs.* [9, p. 76: Fig. 5.1], or [20, p. 183: Tbl. 1] *vs.* [11, p. 10: Fig. 1.6]. The following listing of pseudocode shows that a dynamically scaled genetic programming system with fixed population size $s\in\mathbb{N}$ as considered in this work resembles a dynamically scaled genetic algorithm setting with creatures of arbitrary size:

```
GeneticProgrammingSystem() := (
  integer s, t=1; (* fixed population size, time *)
  integer Θ(); (* time increment function *)
  real μ(), χ(); (* mutation, crossover rate functions *)
  p = initializeRandomPopulationOfSize(s);
  while ( stoppingCriterion != satisfied ) {
     for (θ=1, θ ≤ Θ(t), θ++) {
       p = mutation(μ(t),p); (* cf. section 3.2 *)
       p = crossover(χ(t),p); (* cf. section 3.3 *)
       p = selection(t,p); (* cf. section 3.5 *)
     };
     t = t+Θ(t);
  };
);
```

The main differences between the above paradigms of evolutionary algorithms lie in the representation of the search space $\mathcal{C}$ (*i.e.*, the set of creatures) and the operators used when optimizing an objective or fitness function[1] $f\colon \mathcal{C}\to\mathbb{R}$. Evolution strategies or genetic algorithms in their basic incarnations are applied when one is looking for a vector of

[1]We avoid a discussion of phenotype *vs.* genotype at this point.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
GECCO'06, July 8–12, 2006, Seattle, Washington, USA.
Copyright 2006 ACM 1-59593-186-4/06/0007 ...$5.00.

finite length ℓ in $\mathcal{C} = \mathbb{R}^\ell$, $\mathcal{C} = [0,1]^\ell$ or $\mathcal{C} = \{0,1,\ldots,k\}^\ell$, $k, \ell \in \mathbb{N}$, which optimizes the objective function. The search spaces for evolutionary programming and genetic programming systems are more complex since they try to find an optimal automaton/program in regard to the fitness function, and one often cannot limit the size of such an optimum before running the optimization procedure. In the case of a genetic programming system, $\mathcal{C}$ must therefore equal the *entire* set of programs generated from a given set $\mathcal{A}_f$ of function symbols and a given set $\mathcal{A}_t$ of terminal symbols. Thus, elements of $\mathcal{C}$ may have arbitrary depth or size. One therefore has to deal with an *infinite* dimensional underlying vector space, if the genetic programming system is to be modeled as (inhomogeneous) Markov chain in contrast to modeling genetic algorithms as, *e.g.*, in [20].

There are many different representations of the set of programs $\mathcal{C}$ in genetic programming systems which are accompanied by many different types of mutation and crossover operators. The most widely used representation is that of the *S-expression* [9, p. 69], but other forms, *e.g.*, linear genetic programming [4] are also used. S-expressions are defined as follows: an S-expression is a tree, where each inner node is labeled by an element of $\mathcal{A}_f$ and has as many children as the arity of its labeling function, and each leaf is labeled by an element of $\mathcal{A}_t$.

Random initialization of an S-expression starts at the root and chooses randomly a label from $\mathcal{A} = \mathcal{A}_f \cup \mathcal{A}_t$. If the label is in $\mathcal{A}_t$, then the process terminates. Otherwise the node gets as many children as the arity of the function symbol chosen, and the process is recursively applied for each of the children with a certain limit of depth for the overall generated tree (*cf.* [9, p. 106]). The limit of depth may be randomly determined: compare the end of section 2.3 and definition of mutation on a creature in section 3.1.

In its basic form, crossover for S-expressions first chooses in each of the two parents one node randomly. Then, the subtrees beginning at the chosen nodes are swapped. In its basic form, mutation for S-expressions chooses first one node of the parent (argument) at random. Then, the subtree beginning at the selected node is replaced by a random subtree, generated by the process used for initialization. Hence, mutation, in contrast to crossover, can generate any individual regardless of the parent.

Note that by the algorithm-design presented in this paper, replacing a creature by a random creature under mutation has a parent-independent, non-zero probability depending solely upon the mutation rate (*cf.* sections 3.1 and 3.2). Thus, mutation has a tendency to "pull back" the size of creatures towards an expected finite value[2] countering the phenomenon of bloat, *i.e.*, the uncontrolled growth of the creatures without any improvement of the function value (see, *e.g.*, [11]). In regard to the mathematical model developed in this work, this assures weak ergodicity and possibly strong ergodicity of the Markov chain describing the probabilistic behavior of the genetic programming system under consideration (*cf.* Lemmata 2.2.1 and 2.2.3, and section 4.1). Observe that Lemma 2.2.1 shows strong ergodicity for a genetic programming system with mutation as considered in this work where the operators are not scaled. Thus, the mutation operator introduced here prevents bloat in a wide variety of genetic programming system settings.

[2]See [15] for some explicit computations in that regard.

Mutation was not considered to be a main operator in some early genetic programming systems. It was thought that "the occasional[3] usefulness of mutation in the conventional genetic algorithm ... is largely inapplicable to genetic programming." *cf.* [9, p. 106, pp. 73–77, pp. 599–600]. However, the question *'whether crossover or mutation is the main impelling operator in genetic programming'* was disputed until the end of the last century (see, *e.g.*, [2] or [13]). In the last few years, the number of papers addressing this question seemed to have decreased, possibly as a consequence of the *No Free Lunch Theorem* [23] which states that no search algorithm can be superior when averaged over all functions. Observe however that, as already pointed out in [9, p. 600], mutation in contrast to the crossover and selection operators has the ability to (re-)introduce elements of $\mathcal{A}$ that are not present in the current population. The discussion in [9, p. 105: top of sec. 6.5.1] ascertains to the usefulness of such algorithm capability in the course of exploring a highly non-linear search space. Due to the last point made, the No Free Lunch Theorem, and the fact that mutation counters bloat, we believe that a genetic programming system actually should apply mutation to creatures in the population with non-zero probability in every `while`-loop in the program listing given above.

Besides using mutation, the genetic programming systems analyzed in this work use standard genetic programming operators. In fact, any regular crossover method such as the method described above is allowed (*cf.* section 3.3). Furthermore, the genetic programming systems analyzed in this work use proportional fitness selection as used, *e.g.*, almost exclusively in [9] (*cf.* p. 604).

Since the size of creatures in a genetic programming system is not limited in general, special measures are usually taken to limit the growth of size of the creatures in the population. The most widely used measure is to take into account the size of a creature while computing its fitness value, *i.e.*, a smaller creature gets a higher fitness value than a larger one, if both show the same behavior or represent a solution to the optimization task within a predefined margin of error or similarity (see, *e.g.*, [9, p. 612: sec. 25.12] or [12]). This so-called *parsimony pressure* is also beneficial, as it can further prevent the phenomenon of bloat already mentioned above. Furthermore, parsimony pressure can increase the generalization capabilities of the programs found according to the principle of *Occam's Razor* (*cf.* [5]). We observe that if the standard method of *'computing the fitness value of a candidate solution on a finite set of training inputs'* is employed, *e.g.*, by using the combined distance of the actual output to the desired output, and parsimony pressure is applied properly, then the number of optimal programs can always be made even finite. As a consequence of the above, we shall work under the assumption that a set $\mathcal{C}_{\max}$ of globally optimal creatures exists for the genetic programming systems analyzed below.

While the above discussion gives only a rough insight into principles of genetic programming systems, the following important points for our analysis become apparent:

1.1.1. The search space is infinite due to the *a priori* undetermined size of an optimal program. The associated mathematical formalization requires dealing with an infinite di-

[3]In [18, Sec. 7.5], the necessity of mutation in a genetic algorithm is discussed in order to prevent genetic drift.

mensional Banach space ($\ell^1(\wp)$, *cf.* section 2.2) and bounded linear operators acting on it.

1.1.2. Mutation can generate any search point with a certain probability regardless of the parent. This insures weak ergodicity and eliminates bloat which can be proven within the Banach space framework considered here.

1.1.3. A set $\mathcal{C}_{\max}$ of global optima exists due to reasonable assumptions in regard to the genetic programming system under consideration such as the application of parsimony pressure. $\mathcal{C}_{\max}$ can even be assumed to be finite.

Most theoretical results on genetic programming systems so far concentrate on schema theory. Consult [11] for a very good overview in that regard. These schema theories allow to understand mainly the one-step behavior of genetic programming systems[4] and, thus, are possibly helpful to prove convergence (*cf.* [14]), but thus far this seems not been accomplished.

1.2. Outline of new results and proofs. This paper, while following a similar path of exploration as, *e.g.*, [20], contains a number of results that are new even in the genetic algorithm context. For example, Proposition 3.5.2 allows the adaptation of results on convergence of dynamically scaled genetic algorithms to global optima in [19], [20] and [21] for an extended family of crossover operators. In addition, results on genetic algorithms obtained in [19], [20] and [21] for various distinct settings[5] are (or can be) generalized to genetic programming systems based upon the results presented below.

Sections 2.1–2.2 mostly introduce notation in regard to inhomogeneous Markov chains on a discrete (infinite) space $\wp$ extending [18, Sec. 2] and [20, Sec. 1]. The main results are Lemmata 2.2.1 and 2.2.3 which establish weak ergodicity for the inhomogeneous Markov chain describing the probabilistic behavior for the family of the dynamically scaled genetic programming systems studied in this work.

Sections 2.3–2.4 mainly describe the special notation needed in regard to creatures and populations used in genetic programming systems. Interesting are the tensor-representations of the underlying vector space $\ell^1(\wp)$ given in lines (*7*) and (*8*). These are used in the analysis of the mutation and crossover operators respectively. In turn, this is used in part of the proof of the steady-state flow inequality, *i.e.*, line (*19*). Lemma 2.4.1 is quite obvious but very useful in regard to proving convergence toward uniform populations.

Sections 3.1–3.2 define and investigate mutation. First mutation is described as an action on a creature, and then as action on a population. Many variants of the mutation operator are allowed in our approach. The main result is Proposition 3.2.1 which establishes that mutation satisfies the prerequisites of Lemmata 2.2.1 and 2.2.3, the description of the mutation operator as a tensor product of more elementary operators (used in the proof of line (*19*)), and an estimate for the probability for retaining a population under mutation (used in the proof of line (*18*)).

Section 3.3 defines and investigates crossover. Many variants of the crossover operator are allowed in our approach. The main observation is Proposition 3.3.1 which establishes the description of the crossover operator as a tensor product of more elementary operators, and an estimate for the probability for retaining a population under crossover.

Section 3.4 establishes the *mixing flow inequality* (Proposition 3.4.1.2) for the combined mutation-crossover operator based upon the above two estimates to retain a population. If the mutation rate $\mu(t)$ and the crossover rate $\chi(t)$ both converge to 0 as $t\to\infty$, then the mixing flow inequality proves convergence towards uniform populations for the genetic programming systems studied in this work.

First, section 3.5 lists some known results on fitness-proportional selection in order to fix notation. The scaling of the initial fitness function[6] and, consequently, of the selection operator is achieved by exponentiation with a function $g(t) = B \cdot \log(t)$, *cf.* line (*14*), where $B>0$ must satisfy a certain condition, *cf.* top of section 4.3. In fact, B has to be chosen large enough such that the *conservative* tendency of selection asymptotically outweighs the *destructive* noise of mixing. In addition, Proposition 3.5.2 lists a strong result that provides an estimate for "non-uniformity" in the steady state probability distributions $v_t = G_t v_t$ of the infinite stochastic matrices G_t describing the individual steps of a dynamically scaled genetic programming system (*cf.* line (*17*)).

In section 4.1, a family of scalings for mutation is described such that the inhomogeneous Markov chain describing the probabilistic behavior of the algorithm becomes weakly ergodic. The crossover rate is set to $\chi(t) = \mu(t)^{1/m}$, where $m\in\mathbb{N}$ is fixed such that $s/(2m)>1$. Here, s denotes the population size. Thus for large population size, crossover can be annealed to 0 much slower than mutation and asymptotically can become the dominant mixing operator.

In sections 4.2–4.3, two results are obtained for the invariant probability distributions v_t of the G_t which are crucial for convergence: line (*18*) shows convergence of the v_t towards the set of probability distributions over uniform populations; the steady-state flow inequality, *i.e.*, line (*19*) shows convergence of the v_t towards the set of probability distributions over populations with creatures taken from the set $\mathcal{C}_{\max}$ of global optima.

Using the above, we scale the mutation/crossover rates and selection pressure to be piecewise constant in order to to employ the contraction of iterates of (fixed) G_t, $t\in\mathbb{N}$, towards v_t. This assures that for selected, predetermined and computable times t, the probabilistic behavior of the genetic programming systems considered in this work follows the trajectory of the v_t which asymptotically converge to probability distributions over uniform populations with optimal creatures. Thus, we obtain: a dynamically scaled genetic programming system as described in this work is a non-elitist, non-memory, randomized algorithm that can easily be implemented and is able, *i.e.*, inherently directed to find global optima asymptotically with probability one.

2. NOTATION AND PRELIMINARIES

2.1. The underlying vector space. For a given (possibly infinite) set $\wp$ let the Banach space $\ell^1(\wp)$ be given by

[4]R. Poli: Oral Statement. In W. B. Langdon, R. Poli: *Advanced Tutorial on Genetic Programming Theory I & II.* GECCO 2003, Holiday Inn Mart Plaza, Chicago, Ill, USA.

[5]Vose-Liepins genetic algorithm [19], convergence to global optima for non-binary genetic algorithms with dynamically scaled local mutation [20], and coevolution [21].

[6]The initial fitness function for a genetic programming systems as studied here is allowed to be rank based upon another given fitness function acting on the system.

$$\ell^1(\wp) = \{(v_p)_{p\in\wp} : ||v||_1 = \sum_{p\in\wp} |v_p| < \infty\}. \quad (1)$$

We shall understand elements of $\ell^1(\wp)$ as *column-vectors*. For $p \in \wp$, let the base vector b_p be given by $b_p = (\delta_{p,q})_{q\in\wp} \in \ell^1(\wp)$ where δ is the Kronecker delta. It is a notational convenience to identify $p \in \wp$ with b_p. Since $\ell^1(\wp) \subset \ell^2(\wp)$, the canonical inner product of $v = (v_p)_{p\in\wp}, w = (w_p)_{p\in\wp} \in \ell^1(\wp)$ is well-defined and given by

$$<w, v> = \sum_{p\in\wp} \bar{w}_p v_p. \quad (2)$$

Let $\mathcal{S}_\wp \subset \ell^1(\wp)$ be the set of probability distributions over the discrete set $\wp$.

2.2. Infinite matrices. Let $\mathcal{B}(\ell^1(\wp))$ denote the complex Banach algebra of bounded linear operators on $\ell^1(\wp)$. The identity map $\ell^1(\wp) \to \ell^1(\wp)$ will be denoted by $\mathbf{1}$. Every $X \in \mathcal{B}(\ell^1(\wp))$ can be identified with an infinite two-dimensional array or infinite matrix $(X_{q,p})_{q,p\in\wp}$ given by

$$X_{q,p} = <b_q, Xb_p> = <q, Xp>, \qquad q, p \in \wp. \quad (3)$$

The infinite matrix $(X_{q,p})_{q,p\in\wp} \in \mathcal{B}(\ell^1(\wp))$ operates by canonical matrix multiplication from the left on column vectors in $\ell^1(\wp)$. $X \in \mathcal{B}(\ell^1(\wp))$ is called positive, if $X_{q,p} \geq 0$ for all $q, p \in \wp$. This shall be denoted as $X \in \mathcal{B}(\ell^1(\wp))^+$. $X \in \mathcal{B}(\ell^1(\wp))^+$ is called column-stochastic or for short *stochastic*, if $Xp \in \mathcal{S}_\wp$ for every $p \in \wp$.

The operator norm of $X \in \mathcal{B}(\ell^1(\wp))$ is given by

$$||X||_1 = \sup\{||Xp||_1 : p \in \wp\} < \infty. \quad (4)$$

See [19, proof of eq. 7]. Thus, a stochastic matrix $X \in \mathcal{B}(\ell^1(\wp))$ satisfies $||X||_1 = 1$.

If $v \in \mathcal{S}_\wp$, then let $P(v) \in \mathcal{B}(\ell^1(\wp))$ be defined by $P(v)p = v$ for every $p \in \wp$. $P(v)$ is the stochastic projection onto the one-dimensional space generated by v. We have:

$$P(v)(w - \tilde{w}) = v - v = 0, \quad w, \tilde{w} \in \mathcal{S}_\wp. \quad (5)$$

In contrast to some conventions, we shall use the Π-symbol to denote products of possibly *non-commuting* operators as follows: for $s, t \in \mathbb{Z}$, $s \neq t$ let

$$\prod_{\tau=t}^{s} X_\tau = X_t \cdot X_{t+\mathrm{sign}(s-t)} \cdots X_s, \quad X_\tau \in \mathcal{B}(\ell^1(\wp)). \quad (6)$$

2.2.1. Lemma. Let $w_o, w, \tilde{w} \in \mathcal{S}_\wp$, and let $M, X, Y \in \mathcal{B}(\ell^1(\wp))$ be stochastic. In addition, let $\epsilon > 0$ be such that $M = \epsilon P(w_o) + N$ where $N \in \mathcal{B}(\ell^1(\wp))$ is positive. Then:

1. $||XMY(w - \tilde{w})||_1 \leq (1 - \epsilon)\, ||w - \tilde{w}||_1$.
2. XMY has a uniquely determined invariant eigenvector $v = XMYv \in \mathcal{S}_\wp$.
3. $\lim_{t\to\infty} (XMY)^t = P(v)$.

Proof: Using line (4), we see that $||N||_1 = 1 - \epsilon$. Using line (5), we have:

$$||XMY(w - \tilde{w})||_1 \leq ||M(Yw - Y\tilde{w})||_1$$
$$= ||N(Yw - Y\tilde{w})||_1 \leq (1 - \epsilon)||w - \tilde{w}||_1.$$

This shows statement (1). Now, Banach's fixed point theorem [10, p. 338] yields statements (2) and (3). Q.E.D.

2.2.2. Remark. One can obtain by induction that the powers of M converge to $\epsilon(\mathbf{1} - N)^{-1}P(w_o)$ which must be a projection. Lemma 2.2.1.3 then shows that the invariant eigenvector of M in $\mathcal{S}_\wp$ is given by $\epsilon(\mathbf{1} - N)^{-1}w_o$.

Using [16, Thms. 10.20, 10.31.b, 10.28.b] and investigating powers of XMY, one sees that 1 is an isolated point in the spectrum of XMY and the only element of modulus 1. ⌋

Note that Lemma 2.2.1.3 shows strong ergodicity of a genetic programming system with a mutation operator as considered in this work, if the genetic operators are not dynamically scaled.

Lemma 2.2.1.2 will be used to show that the stochastic operator G_t associated with an individual step of a dynamically scaled genetic programming system as considered here (*cf.* line (17)) has a positive, invariant eigenvector v_t. Using the contraction property established in Lemma 2.2.1.1, we then keep the individual steps of the genetic programming system constant for certain intervals of time such that the overall trajectory of the scaled genetic programming system follows the trajectory of the v_t.

The next Lemma follows directly from Lemma 2.2.1.1.

2.2.3. Lemma. Let $w, \tilde{w}, w_t \in \mathcal{S}_\wp$, $t \in \mathbb{N}$. Let $M_t, X_t, Y_t \in \mathcal{B}(\ell^1(\wp))$ be stochastic. In addition, let $\epsilon_t \in \mathbb{R}^+$ be such that $\sum_{t\in\mathbb{N}} \epsilon_t = \infty$ and $M_t = \epsilon_t P(w_t) + N_t$ where N_t is an element of $\mathcal{B}(\ell^1(\wp))^+$. Then we have:

$$\lim_{t\to\infty} (\prod_{\tau=t}^{1} (X_\tau \cdot M_\tau \cdot Y_\tau))(w - \tilde{w}) = 0. \quad ⌋$$

Proposition 3.2.1.1 together with the settings of section 4.1 shows that mutation contributes matrices satisfying the conditions in Lemma 2.2.3 to the inhomogeneous Markov chain which represents the algorithm. Thus, Lemma 2.2.3 allows for a simple treatment of weak ergodicity in this work. Compare [8, pp. 142–151, p. 151: Thm. V.3.2].

2.3. The alphabet and creatures. The alphabet $\mathcal{A}$ used in the model for the optimization algorithm described in this work shall consist of two subsets: a finite set $\mathcal{A}_t$ of terminal symbols, and a finite set $\mathcal{A}_f$ of function symbols [9, p. 80]. Thus, $\mathcal{A} = \mathcal{A}_t \cup \mathcal{A}_f$.

We assume that the closure property [9, Sec. 6.1.1] holds, *i.e.*, that all output of all elements in $\mathcal{A}$ is admissible as input to all functions in $\mathcal{A}_f$. The output of a terminal symbol is by definition its value upon input.

Following [9, p. 81] and our outline in the introduction, we define a creature (candidate solution, candidate program, individual) as an S-expression, *i.e.*, a rooted, labeled tree with elements of $\mathcal{A}_f$ labeling the non-leaves and elements of $\mathcal{A}_t$ labeling the leaves.

Let $\mathcal{C}_\ell$ be the set of all creatures of depth $\ell \in \mathbb{N}$. Since every function-symbol admits a specific number of arguments, and since $\mathcal{A}_f$ is finite, $\mathcal{C}_\ell$ is finite. Let $\mathcal{C} = \cup_{\ell=1}^{\infty} \mathcal{C}_\ell$ be the set of all creatures.

We assume that the reader is familiar with one procedure generating a (pseudo-)random tree of exact depth ℓ specifying a creature in $\mathcal{C}_\ell$ as above. This procedure is hereafter denoted as `randomCreature(`ℓ`)`. See the introduction (section 1.1) for a short description of one possible initialization procedure for a random creature.

2.4. Populations. The set $\wp$ of populations, to which the genetic programming system is applied, is the set of s-tuples of creatures, $s \in \mathbb{N}$. That is $\wp = \mathcal{C}^s$. We shall assume that s is even, and $s \geq 4$. If $p = (c_1, c_2, \ldots, c_s)$ is a population, $c_\sigma \in \mathcal{C}$, $1 \leq \sigma \leq s$, then we define $\mathrm{set}(p) = \{c_\sigma : 1 \leq \sigma \leq s\}$. If $c \in \mathcal{C}$, then we shall write $c \in p$, if $c \in \mathrm{set}(p)$. A *spot* in a population p is, by definition, the position of one of the creatures in p.

The vector space $\ell^1(\wp)$ is the vector space underlying our model for the genetic programming system considered in this paper. We observe that $\ell^1(\wp)$ can be identified (as a

Banach space, *cf.* [17, p. 92: Ch. III.6]) with the s-fold closed projective tensor product of $\ell^1(\mathcal{C})$ as follows:

$$\ell^1(\wp) = \bigotimes_{\sigma=1}^{s} \ell^1(\mathcal{C}). \tag{7}$$

Under this identification, a population $p = (c_1, c_2, \ldots, c_s) \in \ell^1(\wp)$, $c_\sigma \in \mathcal{C}$, $1 \leq \sigma \leq s$ is mapped to $c_1 \otimes c_2 \otimes \cdots \otimes c_s \in \bigotimes_{\sigma=1}^{s} \ell^1(\mathcal{C})$, where the c_σ are seen as base vectors in $\ell^1(\mathcal{C})$. The tensor product description of $\ell^1(\wp)$ in equation (*7*) can be used to analyze the mutation operator (see Proposition 3.2.1.2, 3.2.1.4). Consult also [20, Prop. 2.2.2.2–3 and further comment].

Similarly to the above, we can identify $\ell^1(\wp)$ in the following way:

$$\ell^1(\wp) = \bigotimes_{\sigma=1}^{s/2} \ell^1(\mathcal{C}^2). \tag{8}$$

This shall be used in the analysis of the crossover operator.

Let $\mathcal{U} \subset \ell^1(\wp)$ be the closed subspace of $\ell^1(\wp)$ generated by the populations which are uniform, *i.e.*, which consist of s copies of a single creature. Let $\wp \cap \mathcal{U}$ denote the set of uniform populations. In addition, $P_{\mathcal{U}} \in \mathcal{B}(\ell^1(\wp))$ shall denote the continuous projection onto $\mathcal{U}$ given by $P_{\mathcal{U}} p = p$ for every $p \in \wp \cap \mathcal{U}$, and $P_{\mathcal{U}} p = 0$ for every $p \notin \wp \cap \mathcal{U}$. In subsequent proofs, the following Lemma replaces [20, Lemma 1.6.1].

2.4.1. Lemma. Let $X \in \mathcal{B}(\ell^1(\wp))$ be such that $Xp = p$ for every $p \in \wp \cap \mathcal{U}$. Then X satisfies $XP_{\mathcal{U}} = P_{\mathcal{U}}$, and $(\mathbf{1} - P_{\mathcal{U}})X = (\mathbf{1} - P_{\mathcal{U}})X(\mathbf{1} - P_{\mathcal{U}})$.

PROOF: A simple check shows $XP_{\mathcal{U}}p = P_{\mathcal{U}}p$ for $p \in \wp$. Thus, $XP_{\mathcal{U}} = P_{\mathcal{U}}$ by continuity of the operators involved. Now consult the proof of [20, Lemma 1.6.1]. Q.E.D.

3. THE GENETIC OPERATORS

The operators used in the genetic programming system described in this work are as follows: (1) mutation M_μ acts sequentially on creatures and with a certain probability μ_o completely replaces a creature at spot σ in the population with a randomly generated new creature; (2) practically any known crossover operator C_χ; and (3) scaled fitness-proportional selection S_t. For selection, the initial fitness-value of creatures in the population is exponentiated (scaled) in accordance with a predetermined schedule of logarithmic growth (*cf.* lines (*14*,*15*)). Note that fitness-proportional selection is used as the standard selection operator in [9].

3.1. Mutation on creatures. Let $\mu_o \in (0, 1]$ be fixed. In what follows, μ_o shall denote the probability by which mutation acting on a creature shall replace the creature by a randomly generated new creature. Mutation acts on a creature $c \in \mathcal{C}$ in accordance with the following listing of pseudo-code:

```
mutationOnCreature(c) := (
  r = randomNumberInUnitInterval();
  if (r < μo) then {
     d = randomCreature(randomInteger());
  } else {
     d = changeCreature(c);
  };
  return(d);
);
```

Thereby, the unit interval is $[0, 1]$; the procedure `randomInteger()` selects a random integer ℓ according to some fixed probability distribution such as a discrete Poisson distribution; the procedure `randomCreature(`ℓ`)` generates a random creature of depth ℓ; and the procedure `changeCreature` mutates a creature in a predefined, customized way. For example, one can follow the procedure described in the introduction (section 1.1).

The above procedure determines a creature-mutation operator M_c. The operator M_c defines a stochastic infinite matrix in $\mathcal{B}(\ell^1(\mathcal{C}))$ which shall also be denoted as M_c. The stochastic matrix M_c describes transition probabilities for creatures. Let $v_c \in \ell^1(\mathcal{C})$ be the strictly positive probability distribution corresponding to the execution of `randomCreature(randomInteger())`, *i.e.*, generating a random creature with the procedures described above. Then we have:

$$M_c = \mu_o P(v_c) + N_c \tag{9}$$

where $N_c \in \mathcal{B}(\ell^1(\mathcal{C}))^+$ corresponds to the action of `changeCreature`.

3.2. Mutation on populations. Let $\mu \in (0, 1]$. In what follows, μ describes the mutation rate which equals the probability for every individual creature in a population to be subject to action of `mutationOnCreature`. Consider $p = (c_1, \ldots, c_s) \in \wp$. Then mutation acts on p in accordance with the following listing of pseudo-code:

```
mutation(μ,p) := (
   for(σ=1; σ ≤ s; σ++) {
     r = randomNumberInUnitInterval();
     if (r < μ) then { cσ = mutationOnCreature(cσ); };
   };
);
```

The above procedure determines a mutation operator M_μ. The operator M_μ determines an infinite stochastic matrix in $\mathcal{B}(\ell^1(\wp))$ which shall also be denoted as M_μ. The stochastic matrix M_μ describes transition probabilities for entire populations. Let $v_c \in \ell^1(\mathcal{C})$ be given as at the end of section 3.1. Let

$$v_\pi = v_c \otimes v_c \otimes \cdots \otimes v_c \in \ell^1(\wp) \tag{10}$$

where the tensor product has s factors v_c.

3.2.1. Proposition. Let $\mu \in (0, 1]$ denote the mutation rate for the multiple-spot mutation operator $M_\mu \in \mathcal{B}(\ell^1(\wp))$. Let v_π be given by line (*10*). Then we have:

1. $M_\mu = (\mu\mu_o)^s P(v_\pi) + N_\pi$ where $N_\pi \in \mathcal{B}(\ell^1(\wp))^+$.
2. $M_\mu = \bigotimes_{\sigma=1}^{s}((1-\mu)\,\mathbf{1} + \mu\, M_c)$ where the identification in line (*7*) is taken into account.
3. If $p \in \wp$, then $<p, M_\mu p> \geq (1-\mu)^s$.
4. If $\mu<1/2$, then M_μ is an invertible element of $\mathcal{B}(\ell^1(\wp))$.

PROOF: M_μ applies `randomCreature(randomInteger())` in every spot with probability $(\mu\mu_o)^s$ according to the definitions of M_c and M_μ. If that case is executed, then the probabilities to generate a (new) particular population are determined by v_π and are obviously independent from the current (old) population. This shows statement (1). If one compares the actions of M_μ and $\bigotimes_{\sigma=1}^{s}((1-\mu)\,\mathbf{1} + \mu\, M_c)$ on base vectors $p \in \ell^1(\wp)$ while taking into account the identification in line (*7*), then statement (2) follows immediately from the continuity of the operators involved. Statement (3) follows from statement (2). Finally, we show statement (4). M_c is an infinite stochastic matrix and, consequently, has operator norm 1 using line (*4*). Now statement (4) follows as in [20, Prop. 2.2.2.3]. Q.E.D.

3.2.2. Remark. Suppose that $p = (c_1, c_2, \ldots, c_s)$ and $q = (d_1, d_2, \ldots, d_s)$ are populations such that $c_\sigma \neq d_\sigma$ for every $\sigma \in [1, s] \cap \mathbb{N}$. Using Proposition 3.2.1.2, we obtain

$$<q, M_\mu p> = \prod_{\sigma=1}^{s} <d_\sigma, ((1-\mu)\mathbf{1} + \mu M_c)\, c_\sigma> \\ = \mu^s \cdot <q, M_1 p> \qquad (11)$$

Thus, a summation in line (*11*) over all possible q is bounded by μ^s. Similarly, if we consider populations q that differ only in s_o creatures from p, $s_o \in [1, s] \cap \mathbb{N}$, then the corresponding sum over $<q, M_\mu p>$ is bounded by a term in the order of μ^{s_o}. ⌋

3.3. Crossover. We shall assume that a local pair-crossover procedure[7] `pairCrossover(`c_1`,`c_2`)` is defined on pairs of creatures $(c_1, c_2) \in \mathcal{C}^2$. Output of `pairCrossover(`$c_1$`,`$c_2$`)` is an element in $\mathcal{C}^2$. Thus, essentially all known crossover methods can be incorporated in our model. In addition, we shall assume that `pairCrossover(,)` induces a stochastic pair-crossover operator $C_{**} \in \mathcal{B}(\ell^1(\mathcal{C}^2))$.

Under the above assumptions, we let $\chi \in [0, 1]$ denote the crossover rate and define the crossover operator on populations as follows: Let $p = (c_1, \ldots, c_s) \in \wp$. Then crossover acts on p in accordance with the following listing of pseudocode:

```
crossover(χ,p) := (
   for(σ=1; σ ≤ s/2; σ++) {
     r = randomNumberInUnitInterval();
     if (r < χ) then {
       (c_{2σ-1}, c_{2σ}) = pairCrossover(c_{2σ-1},c_{2σ});
     };
   };
);
```

The above procedure determines a crossover operator C_χ. The operator C_χ defines a stochastic element in $\mathcal{B}(\ell^1(\wp))$ which shall also be denoted as C_χ. The stochastic matrix C_χ describes transition probabilities for entire populations. Using the identification in line (*8*), we obtain similarly to the proof of Proposition 3.2.1.2–4:

3.3.1. Proposition. Let $\chi \in [0, 1]$ and C_χ denote the stochastic infinite matrix associated with the crossover operator described above. Then we have:

1. $C_\chi = \bigotimes_{\sigma=1}^{s/2}((1-\chi)\,\mathbf{1} + \chi\, C_{**})$ where the identification in line (*8*) is taken into account.
2. If $p \in \wp$, then $<p, C_\chi p> \geq (1-\chi)^{s/2}$.
3. If $\chi < 1/2$, then C_χ is invertible. ⌋

3.4. The mixing flow inequality. The next result is one of the key ingredients in showing convergence to uniform populations by properly scaled genetic programming systems. We follow here mainly the discussion in [19, Prop. 3.1.1]. However, the following result is new even in the genetic algorithm context in that we do not suppose that the crossover operator keeps uniform populations uniform. This yields generalizations of all major results in [19], [20], [21].

[7] It quite obvious how to generalize the approach to crossover presented in this paper to crossover operators that use three or more creatures as input. Details such as the adaptation of the steady-state flow inequality (*cf.* line (*19*)) and a corresponding proof are left to the reader.

3.4.1. Proposition. Let $\mu \in (0, 1]$ denote the mutation rate for the stochastic multiple-spot mutation operator M_μ defined in section 3.2. Let $\chi \in [0, 1)$ denote the crossover rate for the stochastic crossover operator C_χ defined in section 3.3. Let the mixing operator $\mathcal{M}(\mu, \chi)$ be given by either $\mathcal{M}(\mu, \chi) = C_\chi \cdot M_\mu$ or $\mathcal{M}(\mu, \chi) = M_\mu \cdot C_\chi$. Then we have:

1. $\beta_{\mu,\chi} = \inf\{||P_{\mathcal{U}}\mathcal{M}(\mu, \chi)p||_1 : p \in \wp \cap \mathcal{U}\}$ satisfies $(1-\chi)^{s/2}(1-\mu)^s \leq \beta_{\mu,\chi} < 1$.
2. $||(\mathbf{1} - P_{\mathcal{U}})\mathcal{M}(\mu, \chi)v||_1 \leq 1 - \beta_{\mu,\chi} + \beta_{\mu,\chi}||(\mathbf{1} - P_{\mathcal{U}})v||_1$ for every $v \in \mathcal{S}_\wp$.

PROOF: If $p \in \wp \cap \mathcal{U}$, then changing a single spot in the population under mutation makes p non-uniform. Using the definition of the mutation operator, we see that the probability for this to happen is strictly positive. If crossover is executed before or after mutation, the above situation is retained with probability $(1-\chi)^{s/2} > 0$. Thus, $\beta_{\mu,\chi} < 1$. Combining, Propositions 3.2.1.3 and 3.3.1.2 yields $\beta_{\mu,\chi} \geq (1-\chi)^{s/2}(1-\mu)^s$. This completes the proof of statement (1). To show statement (2), one mainly follows the argument in the proof of [19, Prop. 3.1.1]. Q.E.D.

3.5. Selection. The following brief discussion of selection is mostly identical to the discussion of selection in [20, Sec. 2.6]. However, we present a slight generalization in the spirit of [21].

We shall assume that there is a given non-constant (raw) fitness function $\Phi : D_\Phi \to \mathbb{R}^+$ where $D_\Phi \subset \mathcal{C} \times \wp$ is the set of all pairs (c, p) such that $c \in p$. Suppose that Φ is non-zero on every $p \in \wp$. As discussed in the introduction (section 1.1), we shall assume that a non-empty set $\mathcal{C}_{\max} \subset \mathcal{C}$ exists such that for any $p \in \wp$, $c \in \mathcal{C} \setminus \mathcal{C}_{\max}$ and $d \in \mathcal{C}_{\max}$:

$$c, d \in p \Rightarrow \Phi(c, p) < \Phi(d, p), \qquad (12)$$

i.e., the elements of $\mathcal{C}_{\max}$ behave strictly superior in every population they reside in. The genetic programming system is supposed to *maximize* Φ in the sense of finding an element of $\mathcal{C}_{\max}$.

Let $\mathcal{S}_{\max} \subset \mathcal{S}_\wp$ be the convex hull of uniform populations containing creatures from $\mathcal{C}_{\max}$. Let $P_{\max}$ be the projection (*i.e.*, $\{0, 1\}$-diagonal matrix) onto $\mathrm{span}_{\mathbb{C}}(\mathcal{S}_{\max})$.

Based upon Φ, the fitness function f used in the genetic programming system studied in this work can then, *e.g.*, be identical to Φ or can be rank based upon Φ in the sense of [18, Sec. 7.3]. Such f satisfy the condition in line (*12*). In addition to the above, **suppose** that

$$\rho_2(f) = \inf\{f(c, p)^{-1} \max\{f(d, p) : d \in \mathrm{set}(p) \cap \mathcal{C}_{\max} \neq \emptyset\} : \\ p \in \wp,\, c \in \mathrm{set}(p) \setminus \mathcal{C}_{\max} \neq \emptyset\} \;>\; 1. \qquad (13)$$

$\rho_2(f)$ measures the strength of "second-to-best" creatures $c \in \mathcal{C} \setminus \mathcal{C}_{\max}$ in populations containing elements $d \in \mathcal{C}_{\max}$. One has $\rho_2(f) = s/(s-1)$, if the fitness function f is given by rank.

Next, we define *power-law scaling* of the fitness function. Let the function $g : \mathbb{N} \to \mathbb{R}^+$ be given by:

$$g(t) = B \cdot \log(t) \quad \text{for } B \in \mathbb{R}^+_*,\, t \in \mathbb{N}. \qquad (14)$$

Now, set for $(c, p) \in D_f$:

$$f_t(c, p) = (f(c, p))^{g(t)}. \qquad (15)$$

In addition, let $f_t(c, p) = 0$, if $(c, p) \in (\mathcal{C} \times \wp) \setminus D_f$.

Finally, dynamically scaled fitness-proportional selection is defined as follows: For $p = (c_1, c_2, \ldots, c_s)$, $q = (d_1, d_2, \ldots,$

$d_s) \in \wp$ with $c_\sigma, d_\sigma \in \mathcal{C}$, $1 \leq \sigma \leq s$, let $\#(d_\sigma, p)$ denote the number of copies of d_σ in the population p. In this situation, the stochastic operator $S_t \in \mathcal{B}(\ell^1(\wp))$ describing probabilistic passage from p to q under dynamically scaled fitness-proportional selection is given by

$$<q, S_t p> = (\textstyle\sum_{\sigma=1}^{s} f_t(c_\sigma, p))^{-s} \cdot \prod_{\sigma=1}^{s} \#(d_\sigma, p)\, f_t(d_\sigma, p). \qquad (16)$$

A procedural description to determine $S_t p$ corresponding to a listing of `selection(t,p)` considered in the introduction (pseudocode, section 1.1) can be found in [19, Def. 2.3.1].

3.5.1. Proposition. (*Cf.* [20, Prop. 2.6.1]). Let p and q be populations as above and let the dynamically scaled fitness selection operator S_t be defined as in equation (*16*). Let $\theta = 1 - s^{-s+1}$. Then we have

1. If p is a uniform population, then $S_t p = p$.
2. $||P_{\mathcal{U}} S_t p||_1 \geq 1 - \theta$.
3. If $v \in \mathcal{S}_\wp$, then $||(\mathbf{1} - P_{\mathcal{U}}) S_t v||_1 \leq \theta \cdot ||(\mathbf{1} - P_{\mathcal{U}}) v||_1$. ⌋

Based upon the mixing flow inequality (Proposition 3.4.1.2) and Proposition 3.5.1.3, one can prove a complete analogue and generalization[8] of [20, Thm. 3.1.1]. In particular, one obtains:

3.5.2. Proposition. Let $\mu \in (0, 1]$ denote the mutation rate for the stochastic multiple-spot mutation operator M_μ defined in section 3.2. Let $\chi \in [0, 1)$ denote the crossover rate for the stochastic crossover operator C_χ defined in section 3.3. Let the mixing operator $\mathcal{M}(\mu, \chi)$ be given by either $\mathcal{M}(\mu, \chi) = C_\chi \cdot M_\mu$ or $\mathcal{M}(\mu, \chi) = M_\mu \cdot C_\chi$. Let the selection operator S_t, $t \in \mathbb{N}$, be defined as in line (*16*). Let $\theta = 1 - s^{-s+1}$. Let Γ be given by either $\Gamma = S_t \cdot \mathcal{M}(\mu, \chi)$ or $\Gamma = \mathcal{M}(\mu, \chi) \cdot S_t$. Then we have:

1. Γ has a uniquely determined invariant eigenvector $v \in \mathcal{S}_\wp$, *cf.* Proposition 3.2.1.1 and Lemma 2.2.1.2.
2. $||(\mathbf{1} - P_{\mathcal{U}}) v||_1 \leq (1 - \beta_{\mu,\chi})/(1 - \beta_{\mu,\chi}\theta)$, *cf.* proof of [20, Thm. 3.1.1]. ⌋

4. CONVERGENCE TO GLOBAL OPTIMA

4.1. Weak ergodicity. Set $\mu(t) = t^{-1/s}$, $t \in \mathbb{N}$. In addition, let $\chi(t) = \mu(t)^{1/m}$, where $m \in \mathbb{N}$ is fixed such that $s/(2m) > 1$. Thus, we allow crossover to be annealed at a slower rate than mutation provided that the population size s is chosen large enough. Let

$$G_t = S_t C_{\chi(t)} M_{\mu(t)}. \qquad (17)$$

Alternatively, we can reverse the order of crossover and mutation in the definition of G_t. In the discussion that follows, both cases are handled simultaneously.

Applying Lemma 2.2.3, we obtain weak ergodicity of the inhomogeneous Markov chain $(G_t)_{t \in \mathbb{N}}$ and any similar chain where the mutation rates are equal or larger than the $\mu(t)$ set at the beginning of this section.

Using Lemma 2.2.1.2, let $v_t = G_t v_t \in \mathcal{S}_\wp$ be the uniquely determined invariant eigenvector of G_t in $\mathcal{S}_\wp$.

[8] Here, in regard to an infinite set of creatures but also in regard to more general crossover operators in the genetic algorithm setting.

4.2. Convergence to uniform populations for the v_t. Since $\lim_{t\to\infty} \mu(t) = \lim_{t\to\infty} \chi(t) = 0$, Proposition 3.4.1.1 shows that $\lim_{t\to\infty} \beta_{\mu(t),\chi(t)} = 1$. Now, Proposition 3.5.2.2 implies that

$$\lim_{t\to\infty} (\mathbf{1} - P_{\mathcal{U}}) v_t = 0. \qquad (18)$$

4.3. Convergence to global optimum for the v_t. Now, set the parameter B determining the scaling in the selection operator S_t such that $1 < sB \log(\rho_2(f)) + 1/m$. Let $\Omega = (\mathcal{C}_{\max})^s$. Let P_Ω be the projection (*i.e.*, $\{0, 1\}$-diagonal matrix) onto $\mathrm{span}_{\mathbb{C}}(\Omega)$. Let $\omega(t) = ||P_\Omega v_t||_1$. Using the reasoning in the proof of [19, Thm. 3.4.1], we obtain a steady-state flow inequality for some $K > 0$ as follows:

$$\tfrac{1-\omega(t)}{\omega(t)} \leq K(\mu(t)^{s/(2m)-1} + \mu(t)^{sB \log(\rho_2(f))+1/m-1}) \qquad (19)$$

Line (*19*) shows that $\omega(t)$ converges to 1 as $t \to \infty$. Combining the latter fact with line (*18*), we obtain

$$\lim_{t\to\infty} d(v_t, \mathcal{S}_{\max}) = \lim_{t\to\infty} d(P_{\max} v_t, \mathcal{S}_{\max}) = 0 \qquad (20)$$

where $d(x, \mathcal{S}_{\max}) = \min\{||x - y||_1 : y \in \mathcal{S}_{\max}\}$ for $x \in \ell^1(\wp)$. Observe that the convergence in line (*20*) also holds for a subsequence of the v_t. This is employed in section 4.4.

4.3.1. Remark. In regard to obtaining line (*19*), one may copy the proof of [19, Thm. 3.4.1] almost verbatim with the following two exceptions.

First, one disregards the selector-mask J in the proof of [19, Thm. 3.4.1], *i.e.*, one uses $J = [1, s] \cap \mathbb{N}$, $\#(J) = s$.

Second, the only non-trivial change in the proof of [19, Thm. 3.4.1] is to replace the arguments in the discussion of *Part 2b*, Cases 1–2. In fact, let $p \in \Omega$ be fixed,

$$\Omega^+ = \{q \in \wp : \mathrm{set}(q) \cap \mathcal{C}_{\max} \neq \emptyset\},$$

$q^c \in \wp \setminus \Omega^+$, and $q^+ \in \Omega^+ \setminus \Omega$. In the genetic algorithm case, it is sufficient to estimate the order of the mutation rate $\mu(t)$ in *finitely* many matrix coefficients for mixing, *i.e.*, one considers

$$<q^c, \mathcal{M}(\mu(t), \chi(t)) p>, \quad \text{and} \quad <q^+, \mathcal{M}(\mu(t), \chi(t)) p>.$$

In the genetic programming system case, one has to estimate the order of $\mu(t)$ in the sums of the above coefficients over *infinitely* many q^c and q^+ respectively. This can be achieved using arguments similar to the techniques shown in Remark 3.2.2. ⌋

4.4. Approaching the global optimum. Finally, set the time-increment function as

$$\Theta(t) = \mathrm{ceiling}(\mu_o^{-s} t \, \log(\log(t + 2))). \qquad (21)$$

The switching times t^*_τ, $\tau \in \mathbb{N}$, for the time-variable t in the genetic programming system as described in the introduction (pseudocode, section 1.1) are then given by $t^*_1 = 1$ (corresponding to initialization) and $t^*_{\tau+1} = \Theta(t^*_\tau) + t^*_\tau$. For $T \in \mathbb{N}$ define $H_T = \prod_{\tau=T}^{1} G_{t^*_\tau}^{\Theta(t^*_\tau)}$. Using Lemma 2.2.1.1 it is easy to show, that the accumulated contraction factor of $G_t^{\Theta(t)}$ is bounded by $1/\log(t+2)$. Let $w_T = H_T w_o$ for some fixed initial probability distribution $w_o \in \mathcal{S}_\wp$. w_T is the probability distribution over the possible states of the dynamically scaled genetic programming system after completion of T `while`-loops in the pseudocode listed in section 1.1. Consequently, w_T describes "the state" of the algorithm at that time. Then we have:

$$||w_T - v_{t^*_T}|| = ||G_{t^*_T}^{\Theta(t^*_T)} (w_{T-1} - v_{t^*_T})|| \leq 2/\log(t^*_T + 2). \qquad (22)$$

The latter converges to 0 as $T \to \infty$. This shows that the

w_T converge in time towards the convex set of probability distributions over uniform populations that contain globally optimal creatures.

4.5. Conjecture. The authors conjecture that not only weak but strong ergodicity for the inhomogeneous Markov chain $(G_t)_{t\in\mathbb{N}}$ defined in line (*17*) holds. The proof of such a result would likely use [8, p. 160: Thm. V.4.3], [19, Thm. 3.3.2] or a similar theorem. In case this conjecture is true, one could set $\Theta(t) = 1$ in the discussion above, *i.e.*, one could dispense with the inner `for`-loop in the pseudocode listed in the introduction (section 1.1). In addition, a method of showing strong ergodicity for $(G_t)_{t\in\mathbb{N}}$ should also yield strong ergodicity for the algorithm described here which includes the `for`-loop. Thus, not only the w_T but the states for any time would converge towards the convex set of probability distributions over uniform populations that contain globally optimal creatures.

5. CONCLUSION

We have obtained an all-purpose, dynamically scaled genetic programming system for which a *global optimization theorem* holds. The global optimization theorem can be stated by saying that at selected, predetermined and computable times, the combined probability for the dynamically scaled genetic programming system to be in some uniform population containing copies of a single optimal creature approaches 1 over the course of the algorithm regardless of the initial population.

The proposed algorithm is similar to the setting of the simulated annealing algorithm [1]. It is realistic in that the population-size and consequently evaluation-time for the fitness function stay relatively small. Explicit annealing schedules for crossover/mutation and exponentiation schedules for the fitness-function scaling are given such that the proposed algorithm, in fact, can be easily implemented.

Our analysis has, in addition, uncovered a number of new techniques which allow for significant improvements of known results, in particular, results on global optimization for dynamically scaled genetic algorithms as discussed in [19], [20] and [21].

6. REFERENCES

[1] E. Aarts and P. van Laarhoven. Simulated annealing: An introduction. *Statist. Neerlandica*, 43:31–52, 1989.

[2] P. Angeline. Subtree crossover: Building block engine or macromutation. In J. R. Koza*, et al*, editors, *Genetic Programming 97*, pages 9–17. Morgan Kaufmann, San Francisco, CA, USA, 1997.

[3] T. Bäck, D. Fogel, and Z. Michalewicz. *Handbook of Evolutionary Computation*. Inst. of Physics Publishing & Oxford Univ. Press, New York, NY, USA, 1997.

[4] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone. *Genetic Programming — An Introduction; On the Automatic Evolution of Computer Programs and its Applications*. Morgan Kaufmann, San Francisco, CA, USA, 1998.

[5] S. Droste. Genetic programming with guaranteed quality. In J. R. Koza*, et al*, editors, *Genetic Programming 98*, pages 54–59. Morgan Kaufmann, San Francisco, CA, USA, 1998.

[6] D. B. Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. IEEE Press, Piscataway, NY, USA, 1995.

[7] J. H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, Cambridge, MA, USA, 1992.

[8] D. L. Isaacson and R. W. Madsen. *Markov Chains: Theory and Applications*. Prentice-Hall, Upper Saddle River, NJ, USA, 1961.

[9] J. R. Koza. *Genetic Programming*. MIT Press, Cambridge, MA, USA, 1992.

[10] S. Lang. *Analysis I*. Addison-Wesley, Boston, MA, USA, 1969.

[11] W. B. Langdon and R. Poli. *Foundations of Genetic Programming*. Springer, Berlin, Germany, 2002.

[12] S. Luke and L. Panait. Lexicographic parsimony pressure. In W. B. Langdon*, et al*, editors, *Genetic and Evolutionary Computation Conference*, pages 829–836. Morgan Kaufmann, San Francisco, CA, USA, 2002.

[13] S. Luke and L. Spector. A revised comparison of crossover and mutation in genetic programming. In J. R. Koza*, et al*, editors, *Genetic Programming 98*, pages 208–213. Morgan Kaufmann, San Francisco, CA, USA, 1998.

[14] R. Poli. Recursive conditional schema theorem, convergence and population sizing in genetic algorithms. In W. N. Martin and W. M. Spears, editors, *Foundations of Genetic Algorithms 6*, pages 143–163. Morgan Kaufmann, San Francisco, CA, USA, 2000.

[15] J. E. Rowe and N. F. McPhee. The effects of crossover and mutation operators on variable length linear structures. In L. Spector*, et al*, editors, *Genetic and Evolutionary Computation Conference*, pages 535–542. Morgan Kaufmann, San Francisco, CA, USA, 2001.

[16] W. Rudin. *Functional Analysis*. McGraw-Hill, New York, NY, USA, 1973.

[17] H. H. Schaefer. *Topological Vector Spaces*. Springer, Berlin, Germany, 1970.

[18] L. M. Schmitt. Theory of genetic algorithms. *Theoretical Computer Science*, 259:1–61, 2001.

[19] L. M. Schmitt. Asymptotic convergence of scaled genetic algorithms to global optima – a gentle introduction to the theory. In A. Menon, editor, *Frontiers of Evolutionary Computation*, pages 157–192. Kluwer, Dordrecht, The Netherlands, 2004.

[20] L. M. Schmitt. Theory of genetic algorithms II – Models for genetic operators over the string-tensor representation of populations and convergence to global optima for arbitrary fitness function under scaling. *Theoretical Computer Science*, 310:181–231, 2004.

[21] L. M. Schmitt. Classification with scaled genetic algorithms in a coevolutionary setting. In R. Poli*, et al*, editors, *Genetic and Evolutionary Computation Conference*, pages 138–149. Springer, Berlin, Germany, 2004. LNCS 3103.

[22] H.-P. Schwefel. *Evolution and Optimum Seeking*. John Wiley & Sons, New York, NY, USA, 1995.

[23] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.