Centrum voor Wiskunde en Informatica

Transmission function models of finite population genetic algorithms

C.H.M. van Kemenade, J.N. Kok, H La Poutré, D. Thierens

Software Engineering (SEN)

SEN-R9839 December 31, 1998

Report SEN-R9839
ISSN 1386-369X

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

Transmission Function Models of Finite Population Genetic Algorithms

Cees H.M. van Kemenade
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

Joost N. Kok
LIACS, Leiden University
P.O. Box 9512, 2300 RA Leiden, The Netherlands

Han La Poutré
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

Dirk Thierens
Utrecht University
P.O. Box 80.089 3508 TB Utrecht, The Netherlands

ABSTRACT

Infinite population models show a deterministic behaviour. Genetic algorithms with finite populations behave non-deterministicly. For small population sizes, the results obtained with these models differ strongly from the results predicted by the infinite population model. When the population size is increased towards infinity, a convergence to the results predicted by the infinite population models is observed. In real GA's random decisions are used during the run of the GA. These random decisions can lead to a behaviour that results in a deviation of the GA from the expected path of evolution. In this report four sources of non-determinism are identified. Finite population models are generated by explicitly modelling two of these sources. When comparing the results to runs of actual genetic algorithms, similar results are obtained. Hence, this model shows what are the most important sources of non-determinism in the GA for the problem at hand.

1991 Mathematics Subject Classification: 68T05, 68T20
1991 Computing Reviews Classification System: F.2.m, G.1.6, I.2.8
Keywords and Phrases: genetic algorithms, selection schemes, transmission functions, tracing models
Note: Work carried out under theme SEN4 "Evolutionary Computation".

In practical applications GA's with relatively small populations are used. Such GA's with a finite population can behave quite differently from the GA's with an infinite population. The transmission function model has been used to model genetic algorithms with an infitinite population [KKPT98]. In this report an extension of the transmission function model is introduced, such that the behaviour of GA's with a fixed population size can be modelled.

In section 1 the differences between finite and infinite population GA's are discussed. It is shown that the path of evolution predicted by distribution based models in general is not followed exactly by a finite population GA. However, if one increases the population size, then the path can be followed more closely. Section 2 introduces a framework that later is used to model GA's with finite populations. This framework is a modified version of the transmission function model used to described genetic algorithms with an infinite population size [KKPT98]. In section 3 two models for GA's with finite populations are introduced. The first model actually simulates a random run by means of sampling over distributions. By performing a large number of such runs and computing the averaged results, a model for the expected behaviour of a finite population GA

is obtained. The second model splits the search space in a small number of subsets and builds a branching tree by using binomial models. This model represents all possible paths of evolution in a single tree. To validate these models we make a comparison to experimental data obtained by running GA's. The experimental setup is described in section 4, followed by the results of the experiments and the simulations in section 5. The conclusions are given in section 6.

1. Finite population GA's

The population size can have a strong influence on the behaviour of a GA. Therefore, real GA's usually do not behave exactly as predicted by an infinite population model. In this section we discuss four steps during the evolution of a GA where randomness is introduced. These steps result in the non-deterministic behaviour of real GA's. Increasing the population size, usually results in a more deterministic behaviour. Next, we discuss which of the randomized steps of a GA are most sensitive to the size of the population, and therefore result in the largest differences between the infinite and finite population GA's.

1.1 Randomness in GA's

In practice we observe that if the population size of a GA is increased, then the behaviour of the GA often becomes more predictable. For example, population sizing has been investigated by Goldberg et al. [GDC93]. In that paper conservative lower bounds on the population size are given such that sufficient copies of all building blocks are present in the population.

A random generator is used to take many decisions during a run of a genetic algorithm. If the same type of decision has to be taken sufficiently often, then the average outcome over all such experiments will approximate the expected value, due to the law of large numbers.

In case of a canonical genetic algorithm the following randomized steps are used during the GA-run:

1. Initial population generation: the random initial population is used to represent a uniform distribution over the search space. The initial population is likely to approximate the uniform distribution better when the population size increases.

2. Parent-selection: given a parent-population, fitness proportional selection gives the proportion of copies that each individual gets in the intermediate population. Given that the expected proportion of a certain type of individuals is sufficiently large the actual proportion of individuals is likely to approximate this value closely.

3. Parent-pairing: given an intermediate population, the parent pairs are selected by uniform selection. If more copies of individuals A and B are present, then the expected proportion of parent-pairs AB found in the intermediate population is approximated better.

4. Evolutionary operators: the evolutionary operators use random decisions to generate offspring. If more instances of a parent pair are present, then the combined offspring of all these pairs will approximate the expected distribution of offspring better.

For each of these randomized steps the number of times that similar decisions have to be taken is proportional to the population size. This explains why GA's with larger populations behave more deterministic.

1.2 Consequences for finite population GA's

Next, we try to identify which is the step of the evolution process that is most sensitive to the size of the population. Distribution based models, like the transmission function model [Alt94, KKPT98], model the behaviour of an arbitrary GA with an infinite population size. Given the population of a GA one can compute the corresponding histogram. After normalization this histogram corresponds to a distribution. A finite population can only represent a limited number of distributions over search space $\mathcal{S}$. This number increases rapidly when the population size n increases. Given that $|\mathcal{S}| \gg n$, a population of size n can represent approximately $(|\mathcal{S}|^n/n!)$ distributions; Therefore, larger populations are better at approximating an arbitrary distribution. As a result, GA's with

large populations can better approximate distribution predicted by the transmission function model, and therefore can better follow the path predicted such an infinite population model.

The following two sections give a more detailed discussion on how well a GA with a finite population can approximate the behaviour of the infinite population model. Section 1.2.1 discusses the conditions that determine how well a distribution can be modelled by means of a finite sample. (This section may be skipped upon first reading). The consequence of the results of this section for distribution based models of genetic algorithm are then discussed in section 1.2.2.

1.2.1 Approximating distributions Next, we take a closer look at the conditions that determine how well a distribution can be modelled by means of a finite sample. Let $\vec{d}$ denote a distribution over a discrete search space $\mathcal{S}$ (which implies $\sum_{s\in\mathcal{S}} d_s = 1$). Let $C \subset \mathcal{S}$ denote the ϵ-support of $\vec{d}$ which we define to be the smallest subset of $\mathcal{S}$ such that $\sum_{s\in C} d_s \geq (1-\epsilon)$, where ϵ is a small positive number. Furthermore, for the ease of the argument, we assume that $|C|$ is even.

Now, we define the spread of a distribution $\vec{d}$ by $s(\vec{d}) = |C|$, and we conjecture that distributions with a smaller spread can usually be represented easier by means of a finite population. To make this more precise we define the distance between two distributions $\vec{d}$ and $\vec{p}$ as follows

$$d(\vec{d},\vec{p}) = \sum_{s\in C} |d_s - p_s|.$$

Now, let $\vec{d}$ represent an arbitrary distribution over $\mathcal{S}$, let C_d be the ϵ-support of $\vec{d}$, and let $\vec{p}_n$ be the distribution of the best matching population of size n. First, we describe a procedure that (given a $\vec{d}$) constructs the best matching $\vec{p}_n$. Next, we discuss a method to construct the $\vec{d}$ vectors that are most difficult to model by a finite population, and use these to estimate the distance between $\vec{d}$ and $\vec{p}_n$. If $|C_d| > n$, then we can construct this optimal $\vec{p}_n$ as follows. First note that each element of the population covers a fraction $1/n$ of the distribution given by $\vec{p}_n$. The maximal distance $d(\vec{d},\vec{p}_n)$ is two, where half of this distance is due to the fact that the elements of $\vec{p}_n$ are not covered by $\vec{d}$, and the other half comes from parts of $\vec{d}$ not covered by $\vec{p}_n$. The optimal $\vec{p}_n$ covers as much from $\vec{d}$ as possible. To construct this optimal $\vec{p}_n$ we first select all elements of $\vec{d}$ for which $d_s > 1/n$ and cover these with $\lfloor nd_s \rfloor$ elements from the population. The population always contains enough elements to perform this operation because $\sum_{s\in\mathcal{S}} d_s = 1$. Given that α elements of the population are used during this step the maximal distance with an arbitrary placement of the remaining elements is $d(\vec{d},\vec{p}_n) < 1 + \frac{n-\alpha}{n}$. Now the remaining elements are placed such that non-covered parts of $\vec{d}$ are covered as much as possible. Using this procedure the optimal $\vec{p}_n$ is obtained.

Now, the maximal distance $d(\vec{d},\vec{p}_n)$ is determined by construction of the $\vec{d}$ that can be approximated worst by $\vec{p}_n$. This worst case $\vec{d}$ is constructed by maximally "frustrating" the optimal construction procedure of $\vec{p}_n$. We consider the two cases, depending on the value of $|C|$. First, we consider the case that $n < |C| < 2n$. In this case a $\vec{d}$ is constructed, such that all elements d_s are either $\frac{1}{2n}$ or $\frac{3}{2n}$. This is possible because $|C|$ is even. Now for the optimal $\vec{p}_n$ we have $d(\vec{d},\vec{p}_n) = |C|\frac{1}{2n}$, so $\frac{1}{2} < d(\vec{d},\vec{p}_n) < 1$. For $|C| \geq 2n$ we can always construct a distribution with $2n$ elements having $d_s = \frac{1}{2n}$, which results in a distance $d(\vec{d},\vec{p}_n) = 1$. A distance-contribution of $\frac{1}{2n}$ is obtained for each element of $p_s > 0$, so in order to get an even worse distribution we should get a penalty term larger than $\frac{1}{2n}$ out of each element $p_s > 0$. This is possible if we use a $\vec{d}$ such that $d_s < 1/2n$ for all $s \in \mathcal{S}$. Now given an arbitrary distribution $\vec{d}$ with $d_s < 1/2n$, the construction method of $\vec{p_n}$ proceeds as follows. We define an ordering r_i of the elements of $\vec{d}$ such that if $i < j$, then $d_{r_i} \geq d_{r_j}$. The optimal $\vec{p}_n$ will now cover the each of the elements in the range r_1 up to r_n by an individual. Now we have

$$d(\vec{d},\vec{p}_n) = \sum_{i=1}^{n} \left(\frac{1}{n} - d_{r_i}\right) + \sum_{i=n+1}^{|C|} d_{r_i}.$$

Our goal is to find a $\vec{d}$ such that this sum is maximized. The first sum is maximized by having

all d_{r_i} as small as possible, and thus for all $i < n$ we should take $d_{r_i} = d_{r_n}$. The second sum is maximized by taking all d_{r_i} as large as possible, and thus for all $i > n$ we should take $d_{r_i} = d_{r_n}$. Because we have $\sum_{s \in C} d_s = (1 - \epsilon)$, the most difficult distribution to model is the uniform distribution over C given by $d_s = (1 - \epsilon)/|C|$ for all $s \in C$. So given that $|C| > 2n$ the upper bound on the distance is $d(\vec{d}, \vec{p}_n) = 2\left(1 - \frac{n}{|C|}\right)$. When combining these results we see that given values of $|C|$ and n, the worst case distance is given by

$$d(\vec{d}, \vec{p}_n) \leq \begin{cases} |C|\frac{1}{2n} & \text{if } n < |C| < 2n \\ 2\left(1 - \frac{n}{|C|}\right) & |C| \geq 2n \end{cases} .$$

Thus, we see that distributions $\vec{d}$ are more difficult to model when the spread $s(\vec{d})$ increases. Note that the worst case distribution is an almost uniform distribution. If $n < |C| < 2n$, then all d_s are either $\frac{1}{2n}$ or $\frac{3}{2n}$; If $|C| \geq 2n$, then $d_s = 1 - \epsilon/|C|$ for all $s \in C$.

1.2.2 Approximating distribution models We discuss the consequences of these results for the match between finite and infinite population models of GA's, and the influence of population size during the different randomized steps that GA performs. When comparing the behaviour of the infinite population model and the behaviour of a real GA with a finite population, we expect the largest differences when the $s(\vec{d})$ is large, where $\vec{d}$ is the distribution that is predicted by the infinite population model.

The initial population is used to represent this uniform distribution at the start of the evolution process. We have just seen that the uniform distribution is difficult to model by a finite population when $|\mathcal{S}| > 2n$.

The recombination steps usually increase the spread significantly. Therefore, we expect that the population size has a strong influence on how well the distribution after recombination can be approximated. For example, let us take a look at the uniform crossover with one offspring. If this crossover is applied to two binary strings that differ in m loci, then the offspring is one out of 2^m possible offspring-types. Each of these offspring-types has the same probability of being generated, but only one of these offspring-types actually is generated. The evolution of the GA can be influenced strongly by which individual is actually generated; When using a less disruptive crossover operator, the spread of the distribution is increased less.

We consider the influence of the population size during the selection-step to be less important, because selection only changes the proportions of the individuals that already exist in the population. In fact, selection is likely to reduce the spread of the distribution, because it increases the proportion of the best individuals in the distribution. Therefore the distribution after selection $Sel(\vec{d})$ often can be approximated better than the distribution before selection $\vec{d}$. If we want to approximate $\vec{d}$ and $Sel(\vec{d})$ with the same error, then the minimal population size to model $Sel(\vec{d})$ is smaller than or equal to the population size needed to model $\vec{d}$.

In this section we mentioned four randomized steps used in a GA. These steps lead to the non-deterministic behaviour of GA with a small population. Due to these randomized steps the GA with finite population behaves different from the GA with an infinite population. In order to behave similar to the infinite population GA, a finite population GA has to approximate the distributions of the population given by the infinite population model. Approximation of such a distribution gets more difficult when the spread of the distribution is larger. If we look at the different randomized steps of a GA, then we see that the initial distribution has a large spread, recombination increases the spread of the distribution, and selection decreases the spread of the distribution distribution. Thus, the initial distribution and the distributions after recombination have the largest spread, and therefore are the relatively difficult to approximate with a small population. Therefore, in the rest of the report, we will concentrate on models that explicitly consider the consequences of finite populations for those steps that introduce new offspring (i.e. generation of the initial population and the recombination step).

2. Modified transmission function models

We are going to extend the transmission function model. Our extension allows us to obtain both the distributions of the surviving parents and of the newly generated offspring. First, a general outline of the framework is given, followed by the implementation details for the different GA's.

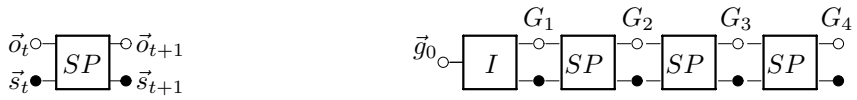

Figure 1: Schematic representation of single evolutionary step of the model (left) and a model for a complete evolution of a genetic algorithm (right).

In order to get a general framework evolutionary algorithms are modelled by means of SP-boxes, which will be described next. A SP-box is shown in the left part of Figure 1. A SP-box represents a single step of the evolution. The S stands for Selection and P stands for production; The inputs on the left are the distribution of the offspring $\vec{o}_t$ of the previous evolutionary step and the distribution of the surviving parents $\vec{s}_t$ of the previous step. The outputs are the distribution of the newly generated offspring $\vec{o}_{t+1}$ and the distribution of the surviving parents $\vec{s}_{t+1}$. These surviving parents are needed to model GA's involving elitism. The generational genetic algorithm fits easily into this schema; There are no surviving parents and the distribution $\vec{s}_t$ can be discarded during the computation of the output distributions. In the case of evolution strategies where selection is done after production of a large number of offspring one has to "reorder" the operations such that selection comes before the production step. This reordering is required because the distribution of the offspring is needed as an output.

The infinite distribution model is obtained by concatenating boxes as shown on the right hand side of Figure 1. Here G_i denotes the distribution of population after i steps of production. This distribution is obtained by combining the distributions $\vec{o}_i$ and $\vec{s}_i$ into a single distribution. Later we will describe exactly how one obtains this distribution. To start one needs a special box, called the I-box, which models the first step of evolution. For generational GA's the I-box is just an SP-box, except for that there is only one input (recall that for generational GA's the distribution $\vec{s}_i$ can be discarded). In case of evolution strategies the I-box contains a production step.

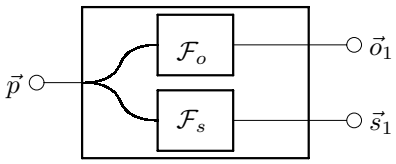

Figure 2: Implementation of a P-box.

A schematic representation of a P-box is shown in Figure 2. The P-box takes a parent distribution as its input, and produces two outputs. The first output represents the produced offspring, while the second output represents the surviving parents. Two functions are used inside the P-box. The first function is $\mathcal{F}_o : P \to P$, which is the part of the transmission function that produces the offspring: it takes a parent distribution as its input, and it generates the distribution of the offspring. The second function is $\mathcal{F}_s : P \to P$, which gives the distribution of the surviving parents. These functions are specific for the different GA's; Instances of these functions for different GA's will be given below.

A schematic representation of a SP-box is given in Figure 3, the dashed box inside the SP-box is a P-box, which represents the production part of the SP-box. The selection step is performed by a function $\mathcal{F} : P \times P \to P$. The actual implementation of $\mathcal{F}$ will depend on the particular GA

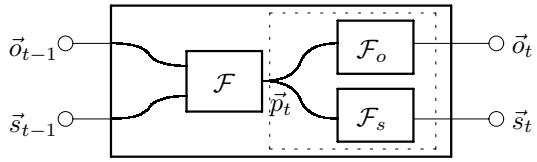

Figure 3: Implementation of a SP-box.

that is modelled. The parents population is obtained by

$$\vec{p}_t = \mathcal{F}(\vec{s}_{t-1}, \vec{o}_{t-1}).$$

The distribution of the parents and the offspring is obtained by

$$\vec{o}_t = \mathcal{F}_o(\vec{p}_t) \quad \text{and} \quad \vec{s}_t = \mathcal{F}_s(\vec{p}_t).$$

2.1 Instances for the different genetic algorithms
In a number of subsections, the implementation of the functions $\mathcal{F}$, $\mathcal{F}_o$, and $\mathcal{F}_s$ is discussed for different evolutionary algorithms. These models are based on the transmission function models for GA's with infinite populations, which were given in section 3 of report [KKPT98]. In the case that an evolutionary algorithm does not involve elitism the parents are always discarded in the sense that one takes $\mathcal{F}_s(\vec{p}) = \vec{u}$ where $\vec{u}$ is the uniform distribution.

2.1.1 Canonical genetic algorithm In case of the canonical GA the function $\mathcal{F}_o$ corresponds to the transmission function model. This results in

$$\mathcal{F}_o(\vec{p}) = \vec{f}(p), \quad \text{where} \quad f_i(\vec{p}) = \sum_{j,k} T_o(i \leftarrow j,k) \frac{f_j f_k}{\bar{f}^2} p_j p_k.$$

When a generational model is applied there are no surviving parents, and therefore we can set $\mathcal{F}_s(\vec{p}) = \vec{u}$ and $\mathcal{F}(\vec{s}, \vec{o}) = \vec{o}$.

2.1.2 Deterministic n-tournament selection In case of the generational GA's the function $\mathcal{F}_o$ corresponds to the transmission function model, which result in

$$\mathcal{F}_o(\vec{p}) = \vec{f}(p), \quad \text{where} \quad f_i(\vec{p} = \sum_{j,k} T_o(i \leftarrow j,k) P^{(n)}_{tour}(j, \vec{p}) P^{(n)}_{tour}(k, \vec{p}).$$

When a generational model is applied there are no surviving parents, and therefore we can set $\mathcal{F}_s(\vec{p}) = \vec{u}$ and $\mathcal{F}(\vec{s}, \vec{o}) = \vec{o}$.

2.1.3 (μ, λ) and BGA selection In case of (μ, λ)-selection [BHmS91, Rec94, Sch95] we get

$$\mathcal{F}_o(\vec{p}) = \vec{f}(p), \quad \text{where} \quad f_i(\vec{p}) = \sum_{j,k} T_o(i \leftarrow j,k) p_j p_k$$

and $\mathcal{F}_s(\vec{p}) = \vec{u}$. The function $\mathcal{F}$ equals

$$\mathcal{F}(\vec{s}, \vec{o}) = \mathrm{Tr}(\vec{o}, \frac{\mu}{\lambda}),$$

so the distribution of $\vec{s}$ is discarded during the application of $\mathcal{F}$. (For details, see section 3.3 of report [KKPT98].)

2.1.4 $(\mu + \lambda)$ and CHC selection In case of the $(\mu + \lambda)$-selection [BHmS91, Rec94, Sch95] we get

$$\mathcal{F}_o(\vec{p}) = \vec{f}(p), \quad \text{where} \quad f_i(\vec{p}) = \sum_{j,k} T_o(i \leftarrow j,k) p_j p_k.$$

and $\mathcal{F}_s(\vec{p}) = \vec{p}$ (so it yields the parent distribution without modifications), and

$$\mathcal{F}(\vec{s},\vec{o}) = \mathrm{Tr}\left(\mathrm{Bl}(\vec{s},\vec{o},\frac{\mu}{\mu+\lambda}),\frac{\mu}{\mu+\lambda}\right).$$

(For details, see section 3.4 of report [KKPT98].)

2.1.5 Triple-competition selection In case of the triple-competition selection scheme [vK97], the formula given in section 3.5 of report [KKPT98] can be split in two parts. The first part, corresponding to the distribution of the offspring, is

$$\mathcal{F}_o(\vec{p}) = \vec{f}(p), \quad \text{where} \quad f_i(\vec{p}) = \sum_{j,k} T_o(i \leftarrow j,k) P(j,k,\vec{p}),$$

and the second part, corresponding to the distribution of the parents, is

$$\mathcal{F}_o(\vec{p}) = \vec{g}(p), \quad \text{where} \quad g_i(\vec{p}) = \sum_{k} P(i,k,\vec{p}).$$

These two parts are combined by

$$\mathcal{F}(\vec{s},\vec{o}) = \mathrm{Bl}(\vec{s},\vec{o},\frac{2}{3}).$$

2.1.6 Elitist recombination The original transmission function model of elitist recombination [TG94], as given in section 3.6 of report [KKPT98], already discriminates between newly generated offspring and surviving parents. Both are combined in $T_{er}(i \leftarrow j,k)$. We rewrite the transmission function model for elitist recombination such that these two types of individuals are separated:

$$T^{(o)}_{er,i}(\vec{p}) + T^{(s)}_{er,i}(\vec{p}) = \sum_{j,k} T_{er}(i \leftarrow j,k) p_j p_k,$$

where $T^{(o)}_{er,i}(\vec{p})$ denotes the offspring labelled i, and $T^{(s)}_{er,i}(\vec{p})$ denotes the surviving parents labelled i. Next, we define

$$\mathcal{F}_o(\vec{p}) = \frac{\overrightarrow{T^{(o)}_{er}}(\vec{p})}{|\overrightarrow{T^{(o)}_{er}}(\vec{p})|} \quad \text{and} \quad \mathcal{F}_s(\vec{p}) = \frac{\overrightarrow{T^{(s)}_{er}}(\vec{p})}{|\overrightarrow{T^{(s)}_{er}}(\vec{p})|}.$$

Now, if we introduce the additional parameter

$$\beta = |\overrightarrow{T^{(s)}_{er}}(\vec{p})|,$$

then we get

$$\mathcal{F}(\vec{s},\vec{o}) = \mathrm{Bl}(\vec{s},\vec{o},\beta).$$

The additional parameter β is needed because the proportion of parents that survives will vary during evolution. If the search progresses slowly, then β is close to one. In order to be able to sample over the offspring an integer number of offspring is needed. Given that the population size is n, we set the number of offspring equal to

$$\max\{\mathrm{Round}(n|\overrightarrow{T^{(o)}_{er}}(\vec{p})|), 1\},$$

where this value is bounded from below by one (the reason being that the null-vector is not normalized).

3. Finite population models

In this section it is discussed how sampling in combination with the (extended) transmission function model can be used to model the influence of the generation of a finite number of offspring.

In section 1 four randomized steps used in GA's were discussed. New individuals are produced during the generation of the initial population, and by the application of evolutionary operators. In section 1 we made plausible that during these two randomized steps the influence of the population size is relatively large. We develop two hybrid models, and use these models to check this assumption. These models approximate a finite population during the generation of the initial population and the recombination, and use an infinite population model to perform the other randomized steps.

The first model is a simulation model that uses the transmission function model to evolve a distribution. After each generation the loss of information due to the finite population size is simulated by a sampling step. A single run of this simulation model mimics a single run of a GA. The second model is a branching model, which constructs a branching tree that models all possible paths of evolution. The next subsections discuss these models in detail.

3.1 Modelling finite populations by simulation

The infinite population model is combined with simulation-steps that mimic the behaviour of the recombination step on a finite population, and with simulation-steps that mimic the influence of the finite initial population. In this way a hybrid model is obtained that follows the infinite population models during selection and mating, while behaving like a finite population GA during the generation of the initial population and during recombination. We first discuss the conditions under which we can simulate a finite sample given the distribution of the infinite sample. Next, it is discussed during which steps of the GA such a simulation approach is possible, and the details of the corresponding hybrid model are given.

We start with a discussion about the type of distributions for which a finite sample can be modelled. Let us assume that we know the expected distribution of a sample, denoted by $\vec{d}$. Now, if a sample consists of independent identically distributed sample-points, then $\vec{d}$ can be interpreted in two different ways. First, $\vec{d}$ represents the expected distribution of a large sample; Second, $\vec{d}$ can be interpreted as the distribution used to draw a single sample-point. Now, given the expected distribution of the sample, one can simulate a population of size n by generating sample-points according to $\vec{d}$. After normalization of the histogram of this sample a new distribution is obtained, which we denote by $\vec{p}$. This step actually corresponds to approximating $\vec{d}$ by a sample of size n. For a small sample this actual distribution can differ quite a lot from the expected distribution. If the size of the sample n is increased, then the difference between $\vec{p}$ and $\vec{d}$ is likely to decrease, and in the limit of $n \to \infty$ the $\vec{p}$ converges to $\vec{d}$. Thus, $\vec{p}$ can be considered as a randomized simulation $\vec{d}$, and this simulation is likely to approximate $\vec{d}$ better when increasing the size of the sample that $\vec{p}$ corresponds to.

Next, we consider the use of such a simulation approach to model the influence of the population size during certain steps of the GA. Such a simulation is possible for distributions representing a sample of independent identical distributed sample points. Thus, such an approach can only be used for a population in which all individuals are generated independently of each other. This is the case during the generation of the initial population and during application of a recombination operator that produces only one offspring. In case of probabilistic selection (such as fitness proportional selection or tournament selection) the individuals in a population are also generated independently of each other (provided that selection variance reduction mechanism is used, see section 2.1 of report [KKPT98] for details). Deterministic selection schemes do not select individuals independently of each other, and thus cannot be simulated in this manner.

An example of simulating a finite population by sampling over a distribution is given in Figure 4. In the top row the evolution of a single generation is shown; Selection, mating, and recombination are performed in sequence on a distribution. These distribution represent a population of infinite size. After the recombination step a distribution $\vec{d}$ is obtained. Next, a random sample of size n is generated according to this distribution. The distribution of this sample, denoted by $\vec{p}$, is computed. Next, $\vec{p}$ is used as an input for the selection step of the infinite population model. The

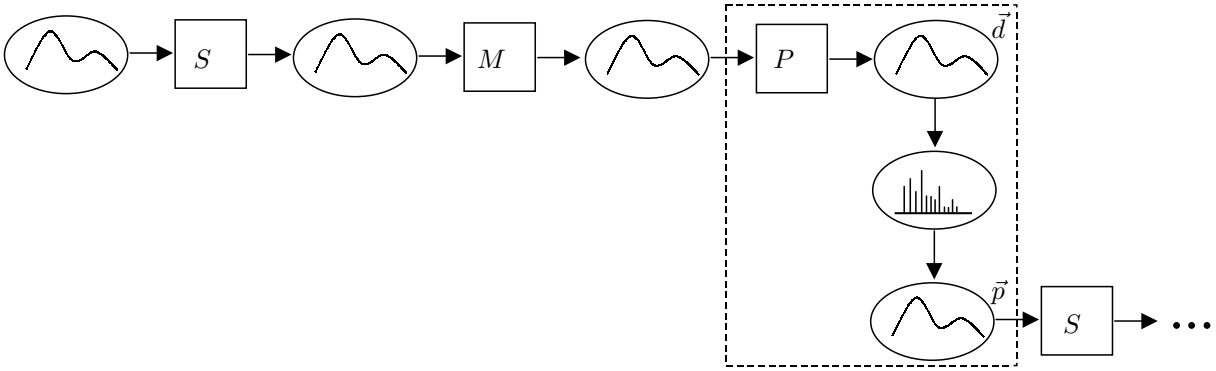

Figure 4: Simulate a finite recombination-step by sampling for a generational GA.

operations inside the dashed box mimic recombination applied to a finite population.

Figure 4 shows a single generation out of a run of the simulation model for a generational GA. Using the SP-boxes a run of an arbitrary GA can be modelled. The right-hand side of Figure 1 shows the model based on SP-boxes. In the SP-box a separation is made between the newly generated offspring, denoted by $\vec{o}_t$, and the surviving parents, denoted by $\vec{s}_t$. The places indicated by open circles represent the points where simulation of a sample is used to model the behaviour of of a finite population. These open circles indicate all points where new individuals are generated.

Hence, we introduced in this section a simulation model that uses an infinite population during the selection step and the mating step, and a finite population during the generation of the initial population and during recombination. This simulation model is a hybrid model in the sense that part of the evolution process is performed with an infinite population and part is performed with a finite population. This model allows us to differentiate between the influence of the population size during the different steps of evolution.

3.2 Branching over binomially distributed finite populations

Next we are going to consider a model that represents all possible paths of evolution in a single tree. This model splits the search space in a small number of subsets and builds a branching tree using binomial distributions.

Figure 1 denotes only a single possible path of evolution. When modelling all possible paths of evolution a tree-like structure is obtained. The input of each node is the current distribution of surviving parents and newly generated offspring, and the output is a set of branches, each one denoting a possible actual distribution of the population of size n. These branches are then fed to different SP-boxes.

Following all possible traces of evolution for a finite population GA requires a lot of computation. Given the cardinality of the search space $|\mathcal{S}|$, and the population size n, a single node has approximately $(|\mathcal{S}|^n/n!)$ outputs, when $|\mathcal{S}| \gg n$. Even when a representation by means of equivalence classes is used, the amount of outputs is going to be large. The total amount of computation for the complete model is very large, because the number of nodes in the tree describing the complete evolution is approximately $(|\mathcal{S}|^n/n!)^G$, where G is the number of generations. To reduce the amount of computation an approximate model is introduced, where a relatively small number of branches is generated for each node. To do so, the set of all possible bit-strings $\mathcal{S}$ is partitioned in two disjoint subsets, S and $\overline{S}$, such that $S \cup \overline{S} = \mathcal{S}$. These sets are for example chosen in such a way that elements of S contain certain parts of the optimal solution, while the elements of $\overline{S}$ do not contain these parts. The number of elements in S can vary between 0 and n, so the number of outputs of a node is $n + 1$.

The same procedure can be repeated by splitting the set S in two disjoint sets S_1 and $\overline{S_1}$. In this way it is possible to continue until the subsets exactly defines a specific population. Using only a few subsets one can already observe a good match between the model and actual GA-runs,

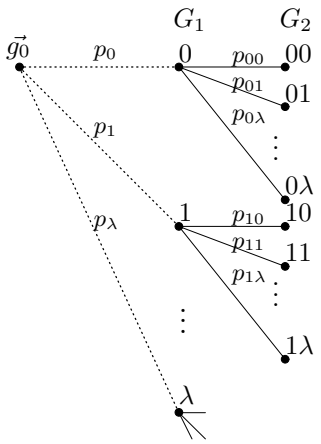

Figure 5: Picture of a branching tree for a finite-population model

as will be shown in the rest of this report.

We will focus on the case where the search space $\mathcal{S}$ is split in two disjoint sets S and $\overline{S}$. The tree corresponding to this case is shown in Figure 5. A more detailed description of the computation is given by the dataflow graphs in Figures 6 and 7, which will be discussed later in this section. Figure 5 shows the computation by means of a tree. The nodes are labelled with a sequences of numbers. These numbers correspond to the number of elements in the population that belong to S. For example a node labelled 134 corresponds to a path where one individual belonging to S is present in the initial population, three such individuals are present in the first generation, and four such individuals are present in the second generation. The number of offspring produced during a single generation is λ. The arcs are labelled with the probability of the corresponding transition. The root of the tree corresponds to the distribution used to draw the initial population (denoted by $\vec{g}_0$). The first branch corresponds to the case that no individual in the population is contained in S, the second branch corresponds to case that exactly one individual in the population belongs to the set S, and the branch with label k corresponds to case that k individuals belong to the set S. The probability of branch k is denoted by p_k.

3.2.1 Distributions corresponding to finite populations Assume that we have a transition to a node in which k individuals should belong to set S (out of λ). If the proportion of individuals belonging to S in an distribution $\vec{g}$ is equal to p, then the distribution corresponding to k out of λ individuals belonging to the set S is constructed as follows. First, the actual proportion $p' = k/\lambda$ is computed. The proportion of all elements that belong to the set S are multiplied by factor p'/p, while the proportions of the other elements are multiplied by a factor $(1-p')/(1-p)$. So, the relative proportions of individuals that belong to the same set remain unchanged, but the relative proportions of individuals that belong to different sets change by this procedure. Let $\mathcal{G} : \mathbb{R} \times P \to P$ be the function that performs this operation, where the real-valued parameter is the proportion of individuals that belong to the set S, and where P is the space of all possible distributions over the search space. The following identity holds

$$\sum_{k=0}^{\lambda} \mathrm{Bin}(\lambda, k, p)\mathcal{G}\left(\frac{k}{\lambda}, \vec{g}\right) = \vec{g}.$$

Here $\mathrm{Bin}(n, k, p)$ is the binomial distribution that represents the probability of obtaining k successes out of n independent random events, where p is the probability of success. To prove this equality let us consider a single element g_i from distribution $\vec{g}$, and let us assume that element i belong to the set S. We have $\sum_{k=0}^{\lambda} \mathrm{Bin}(\lambda, k, p)\frac{k}{\lambda} g_i = g_i$. After some rewriting one obtains

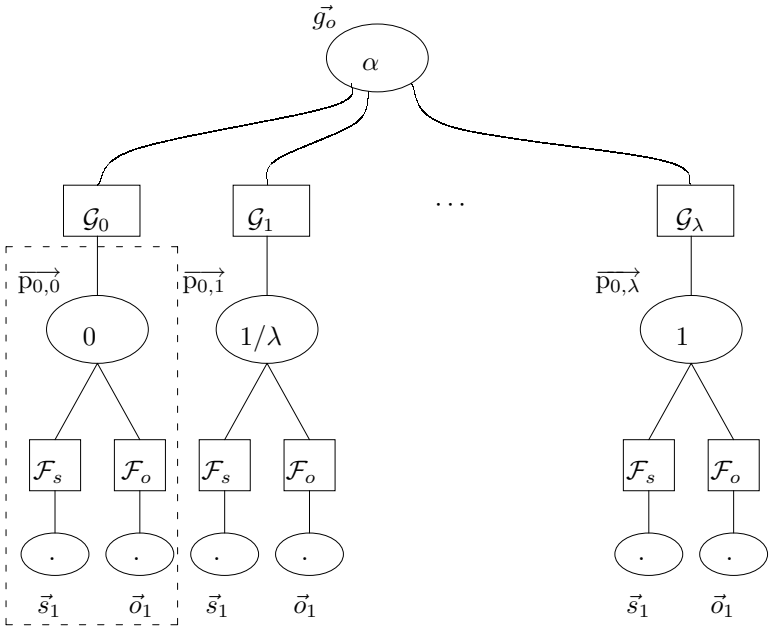

Figure 6: Data flow for generation G_0 (for ES)

$\sum_{k=0}^{\lambda} \mathrm{Bin}(\lambda, k, p)k = \lambda p$, and hence the identity holds. The same result holds for an element of the set $\overline{S}$, using $\mathrm{Bin}(\lambda, k, p) = \mathrm{Bin}(\lambda, \lambda - k, 1 - p)$.

3.2.2 Transitions from generation G_0 The root of the branching tree is the expected distribution of the initial population. A detailed view of the computation at generation G_0 is given by the dataflow graph in Figure 6. The ovals represent distributions, while the boxes represent operations on distributions. The dashed box marks a set of operations that correspond to a single P-box. At the top the distribution of the initial population, with label $\vec{g}_0$, is shown. The expected proportion of individuals in set S is denoted by α. Next, the functions $\mathcal{G}_k$ that produce the new distribution in which proportions $\frac{k}{\lambda}$ $(k = 0, ..., \lambda)$ of individuals belonging to the set S are given. The resulting distributions are labelled $\overrightarrow{p_{0,k}}$. Given one such a distribution, the surviving parents and offspring are computed by means of the functions $\mathcal{F}_s$ and $\mathcal{F}_o$. So, after a single generation each trace is described by means of two distributions $\vec{s}$ and $\vec{o}$.

Now let us assume that the population of parents contains μ individuals that are used to produce λ offspring, and the distribution of the parents in the initial population is $\vec{g}_0$. The distribution of the node labelled k of the first generation is

$$\overrightarrow{p_{0,k}} = \mathcal{G}(\frac{k}{\mu}, \vec{g}_0),$$

which is denoted by the symbol $\mathcal{G}_k$ in Figure 6. The probability of this transition is

$$p_k = \mathrm{Bin}(\mu, k, p),$$

where k is the proportion of individuals from $\vec{g}_0$ that belongs to the set S.

3.2.3 Transitions from generation G_t A detailed view of the computation of generation G_{t+1} from generation G_t is given by the dataflow graph in Figure 7. The dashed box denotes a set of operations that correspond to a single SP-box. The evolution of a single trace is shown. On top the inputs of a node consisting of the two distributions $\vec{s}_t$ and $\vec{o}_t$ are given. The function G_k is

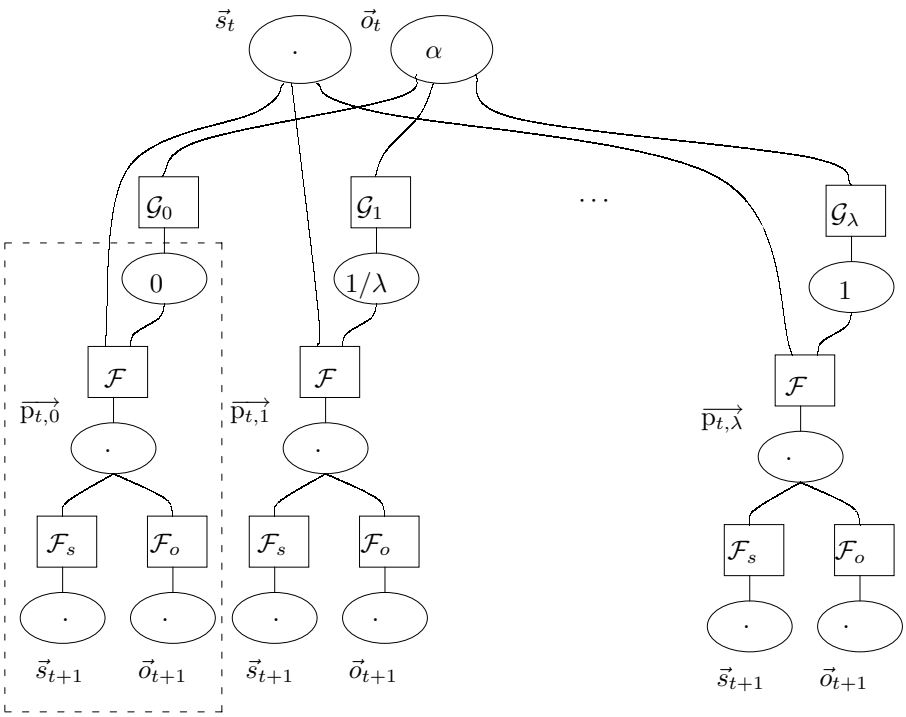

Figure 7: Picture of the data flow for a single branch of generation G_t for $t > 0$

applied to this distribution in order to get a distribution where a proportion k/λ individuals are in the set S. The function $\mathcal{F}$ is applied to this modified offspring distribution and the distribution of surviving parents to obtain the distribution $\overrightarrow{p_{t,k}}$:

$$\overrightarrow{p_{t,k}} = \mathcal{F}(\vec{s}_{t-1}, \mathcal{G}_k(\frac{k}{\lambda}, \vec{o}_{t-1})).$$

Its probability is given by

$$p_{t,k} = p_x \mathrm{Bin}(\mu, k, p),$$

where p_x is probability of reaching node $P_{t,x}$ that is the predecessor of the node under consideration. There is a unique path from the root of the tree to the node $P_{t,x}$ and the probability p_x is the product of the individual transition probabilities along this path.

The expected distribution $\vec{g}_t$ is given by:

$$\vec{g}_t = \sum_{k=0}^{\lambda} p_{t,k} P_{t,k}$$

i.e. a weighted average over the distributions at this depth in the tree.

3.2.4 Merging of transitions A complete evolution tree consists of $(N+1)^G$ branches, where N is the size of the population, and G is the number of generations. It is not feasible to follow all these branches. In order to obtain a feasible computation the set of followed branches is limited to those branches that have a probability above a certain threshold δ. Branches that are not followed will be put together with the nearest branch that is being followed. The proportion of individuals in the set S is assigned a weighted average according to the formula $k' = \sum_{k=l}^{m} p_{xk} p_x k$, where x is the label of the node in the preceding generation from which the branch originates, l and m denote the range of the branches that are combined to a single branch, and p_x denotes the probability that the evolution ends in node x. The resulting value of k' does not have to be integer anymore in case branches are merged. This probability of the joined branch is set to $\sum_{k=l}^{m} p_{xk}$. If none of the

branches has a probability above the threshold, then the net effect is that the infinite population model is traced.

Given the lower bound on the probability of a traced branch, the total number of followed branches of a single generation is smaller than $\lfloor \frac{1}{\delta} \rfloor$. The maximal width of the tree at generation G_i is n^i, so the upper bound on total number of branches followed is given by

$$n_t \leq \sum_{i=0}^{G} \min\{\lfloor \frac{1}{\delta} \rfloor, n^i\} \leq 1 + \lfloor \frac{1}{\delta} \rfloor G.$$

A tighter upper bound is obtained by observing that the $n^i \leq \lfloor \frac{1}{\delta} \rfloor$ for generations i such that $i < {}^n\log(\lfloor \frac{1}{\delta} \rfloor)$. Using this bound one obtains

$$n_t \leq n^{\alpha}/ln(n) + \beta \lfloor \frac{1}{\delta} \rfloor,$$

where $\alpha = \max\{g+1, G\}$ and $\beta = \min\{0, G-g\}$.

Each node of the tree given in Figure 5 corresponds to a constant amount of computation because each node corresponds to a single evaluation of the functions $\mathcal{G}$, $\mathcal{F}$, $\mathcal{F}_s$, and $\mathcal{F}_o$. Therefore these bounds are also the bounds on the total amount of computation.

4. Experiment setup

To test the models we implemented the models for different GA's, implemented the corresponding real GA's, and applied these to the cross-competition problem. This problem (which is defined in section 6 of report [KKPT98]) consists of two parts, the first part corresponds to a oneMax problem, and the second part correspond to a deceptive trap-function. The individuals are coded as bit-strings of length 12 for the real GA's, and are coded by means of 49 equivalence classes (see section 6 of report [KKPT98] for details) for the models.

When coding individuals in linear strings of bits, genetic drift influences the performance of finite population genetic algorithms. The cross-competition problem is susceptible to genetic drift. Within a partition corresponding to a single function of unitation all loci have similar importance, so there is no reason for some loci to converge faster than other loci. But due to genetic drift some loci might converge faster than others within one specific run of the GA. This type of genetic drift will be ruled out because a formulation in terms of equivalence classes will be used. The underlying assumption is that all loci belonging to the same partition are governed by the same probability of containing a one-bit. In order to get rid of the genetic drift in the real genetic algorithm a modified version of the uniform crossover operator is applied. This crossover accepts two parent individuals. For both individuals an intermediate individual is created in which all bits within the same partition are shuffled. Such an intermediate individual has the same fitness as the original individual because the fitness of a part is given by a function of unitation. Next, uniform crossover is applied to these intermediate individuals. Due to this shuffling, the probability of observing a one-bit in each of the loci of a partition becomes equal again, so genetic drift of individual loci is removed. The removal of genetic drift during the experiments allows us to focus on the influence of the other assumptions.

Next, the details about the implemented GA's are presented. For the generational genetic algorithms, and $(\mu \dot{+} \lambda)$-selection schemes, we have two implementations. The first implementation is a straightforward implementation of the algorithm, while the second implementation contains a selection method that reduces the selection variance (see section 2.1 of report [KKPT98] for details).

In case of triple-competition and elitist recombination the algorithms are defined in such a way that the selection variance is small already.

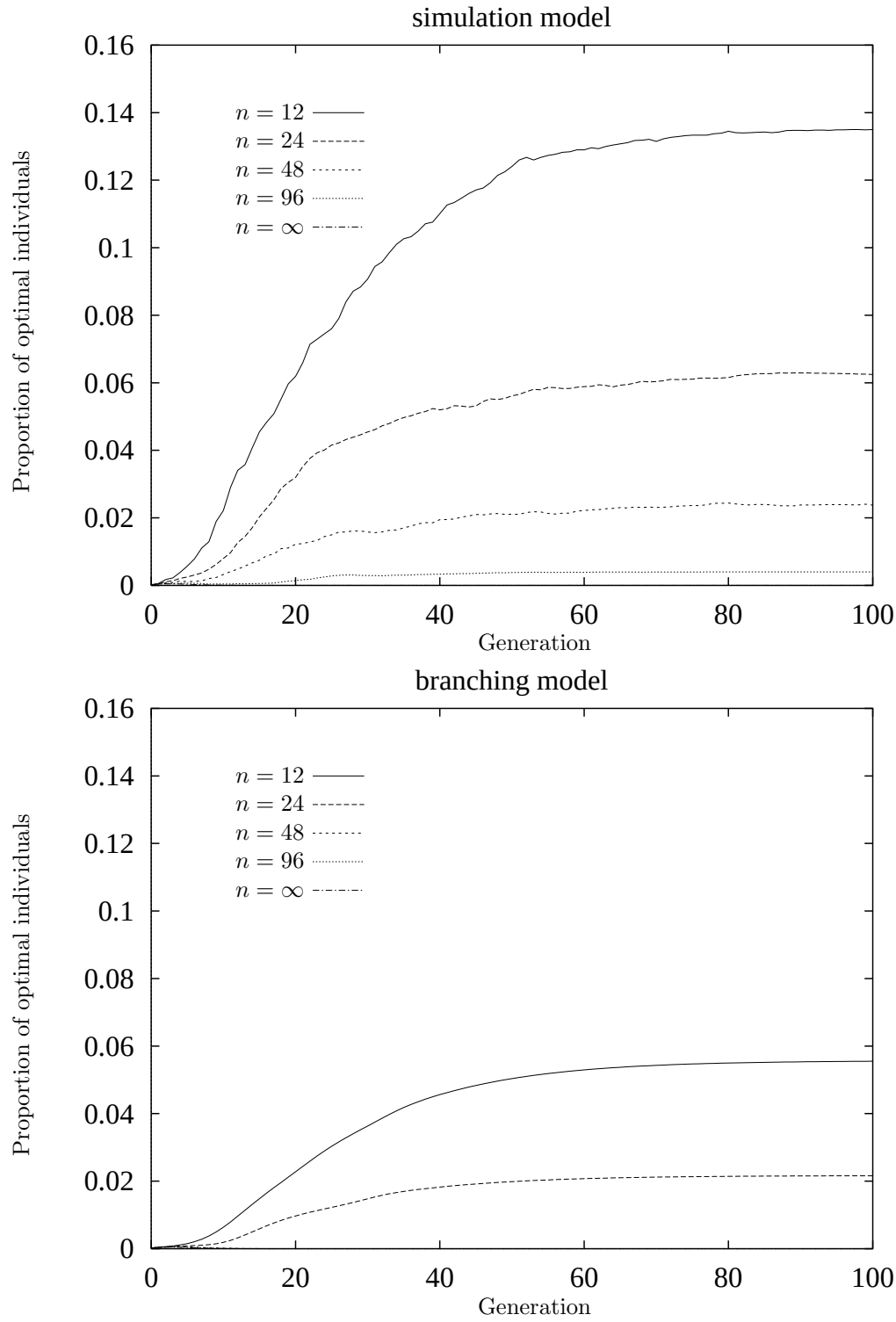

Figure 8: Proportion of optimal solutions for the canonical genetic algorithm both for the simulation model (top) and the branching model (bottom).

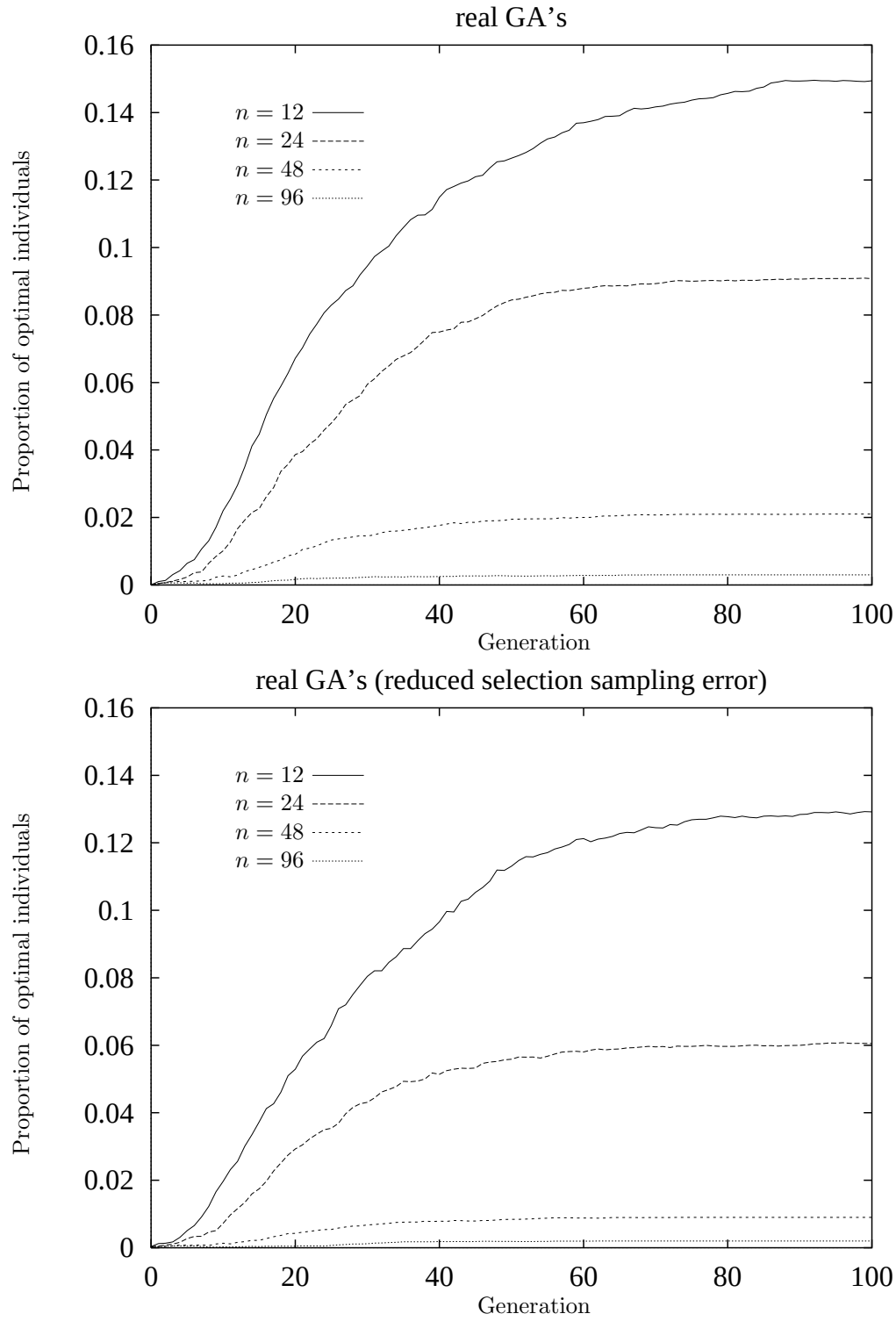

Figure 9: Proportion of optimal solutions for the canonical genetic algorithm both real GA's both without (top) and with (bottom) reduction of selection variance.

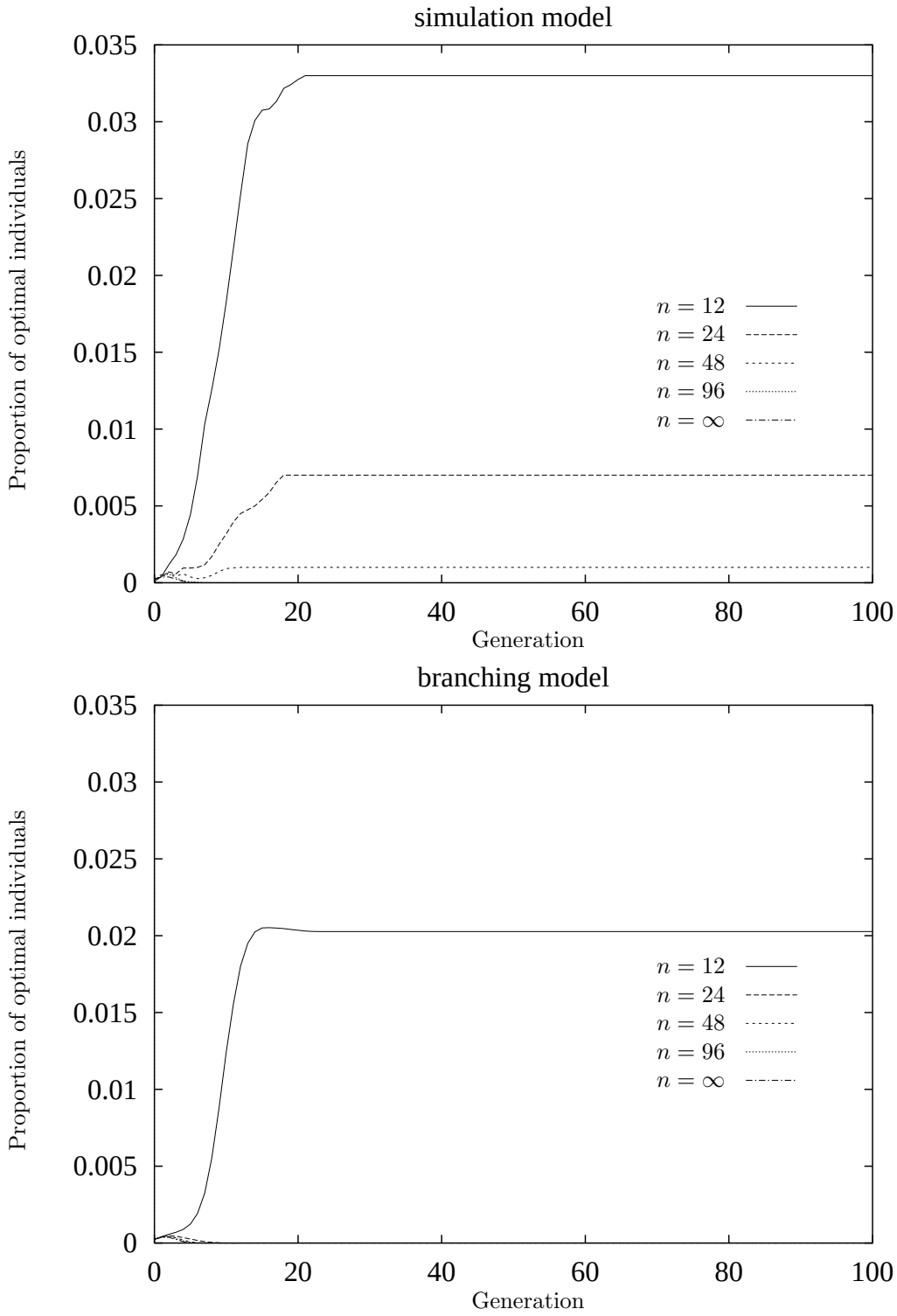

Figure 10: Proportion of optimal solutions for the generational genetic algorithm with 2-tournament selection both for the simulation model (top) and the branching model (bottom).

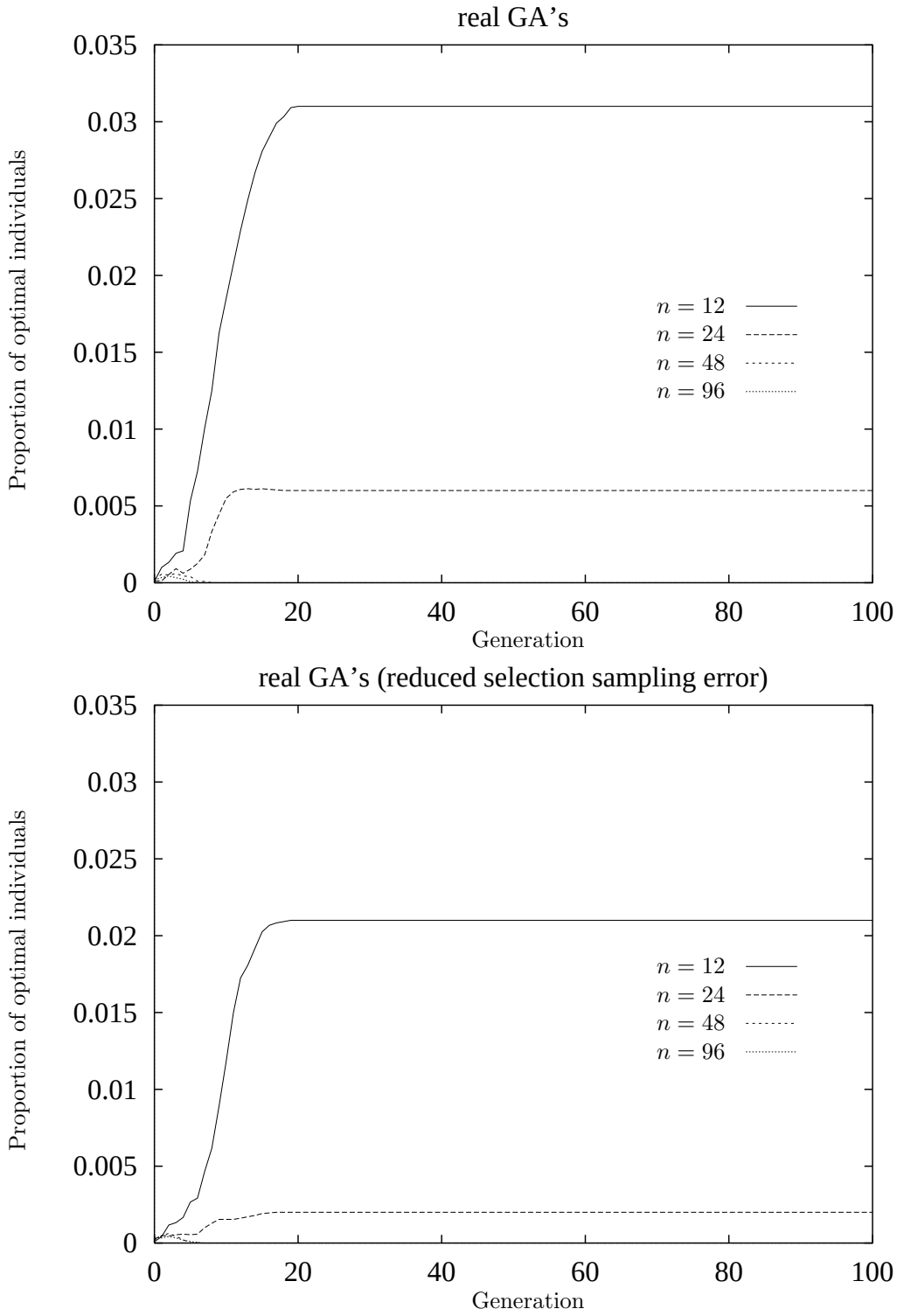

Figure 11: Proportion of optimal solutions for the generational genetic algorithm with 2-tournament selection both without (top) and with (bottom) reduction of selection variance.

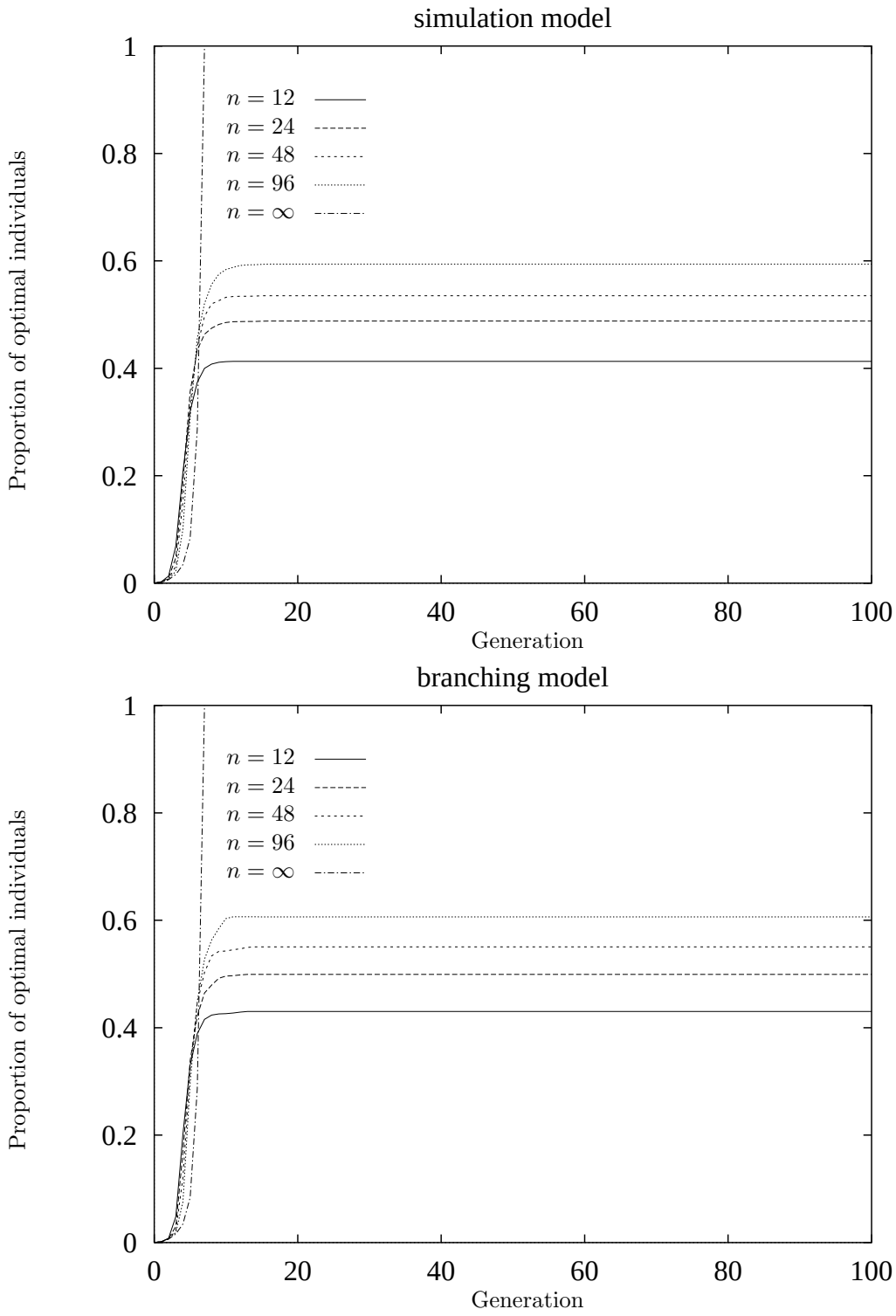

Figure 12: Proportion of optimal solutions for the (μ, λ) selection (BGA) both for the simulation model (top) and the branching model (bottom).

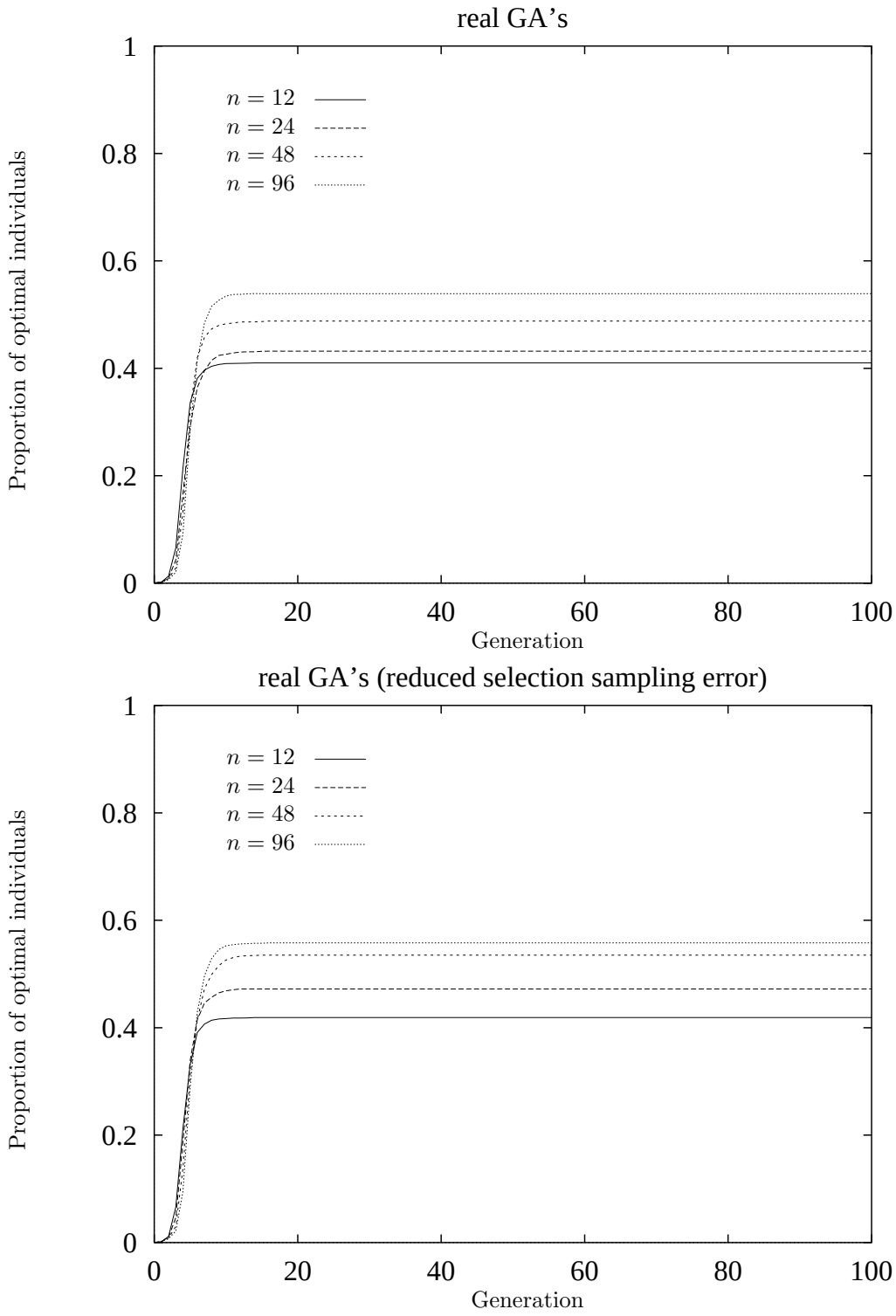

Figure 13: Proportion of optimal solutions for the (μ, λ) selection (BGA) both without (top) and with (bottom) reduction of selection variance.

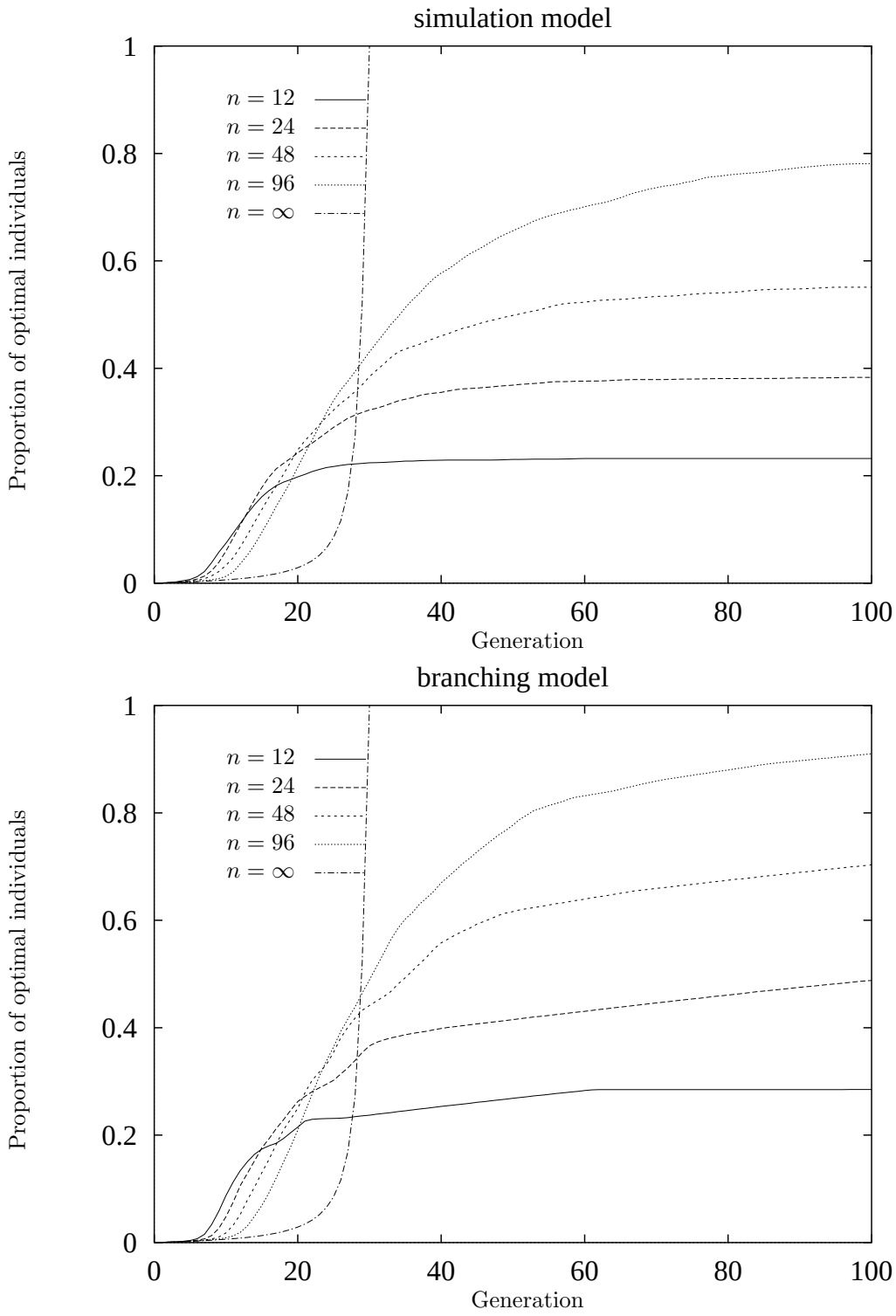

Figure 14: Proportion of optimal solutions for the $(\mu + \lambda)$ selection (CHC) both for the simulation model (top) and the branching model (bottom).

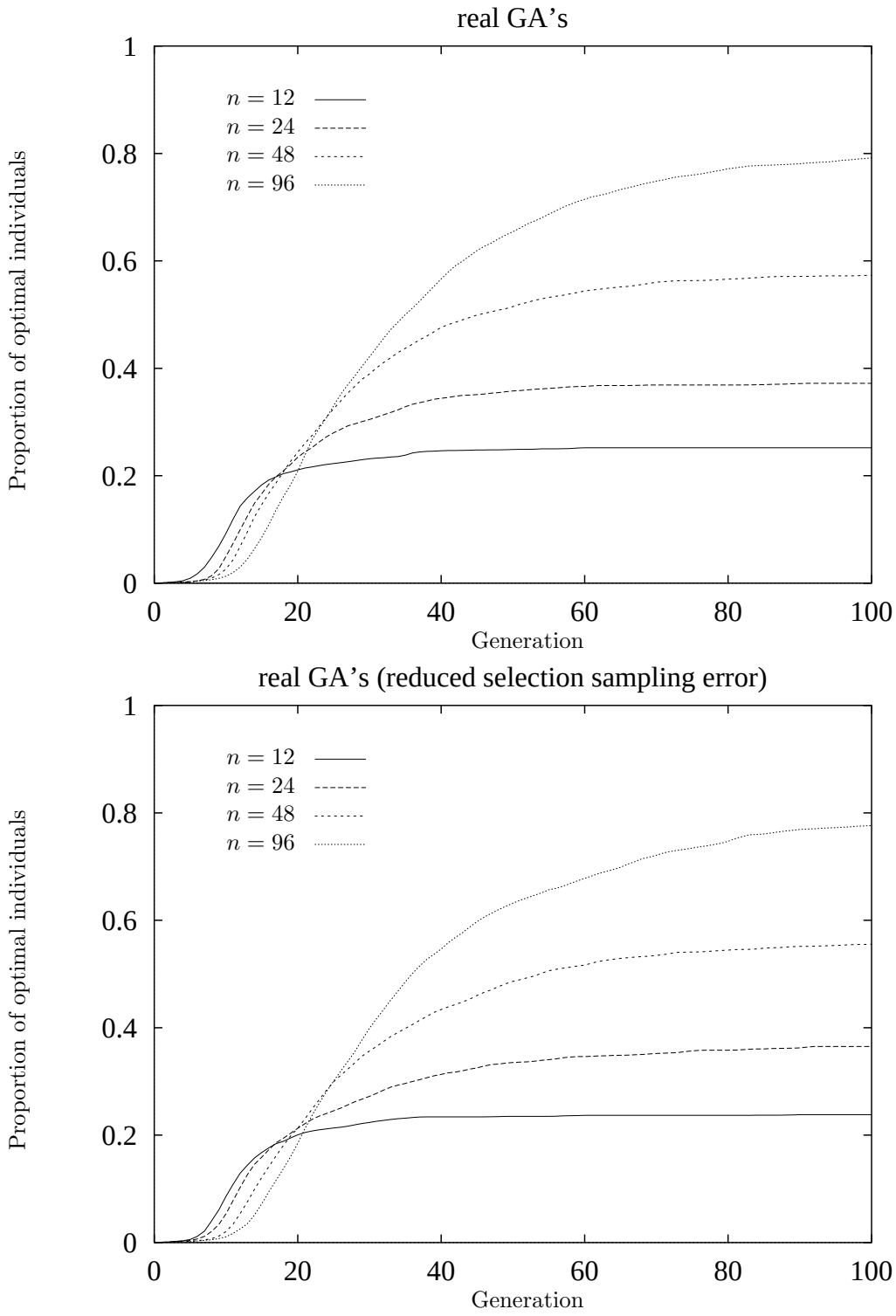

Figure 15: Proportion of optimal solutions for the $(\mu + \lambda)$ selection (CHC) both without (top) and with (bottom) reduction of selection variance.

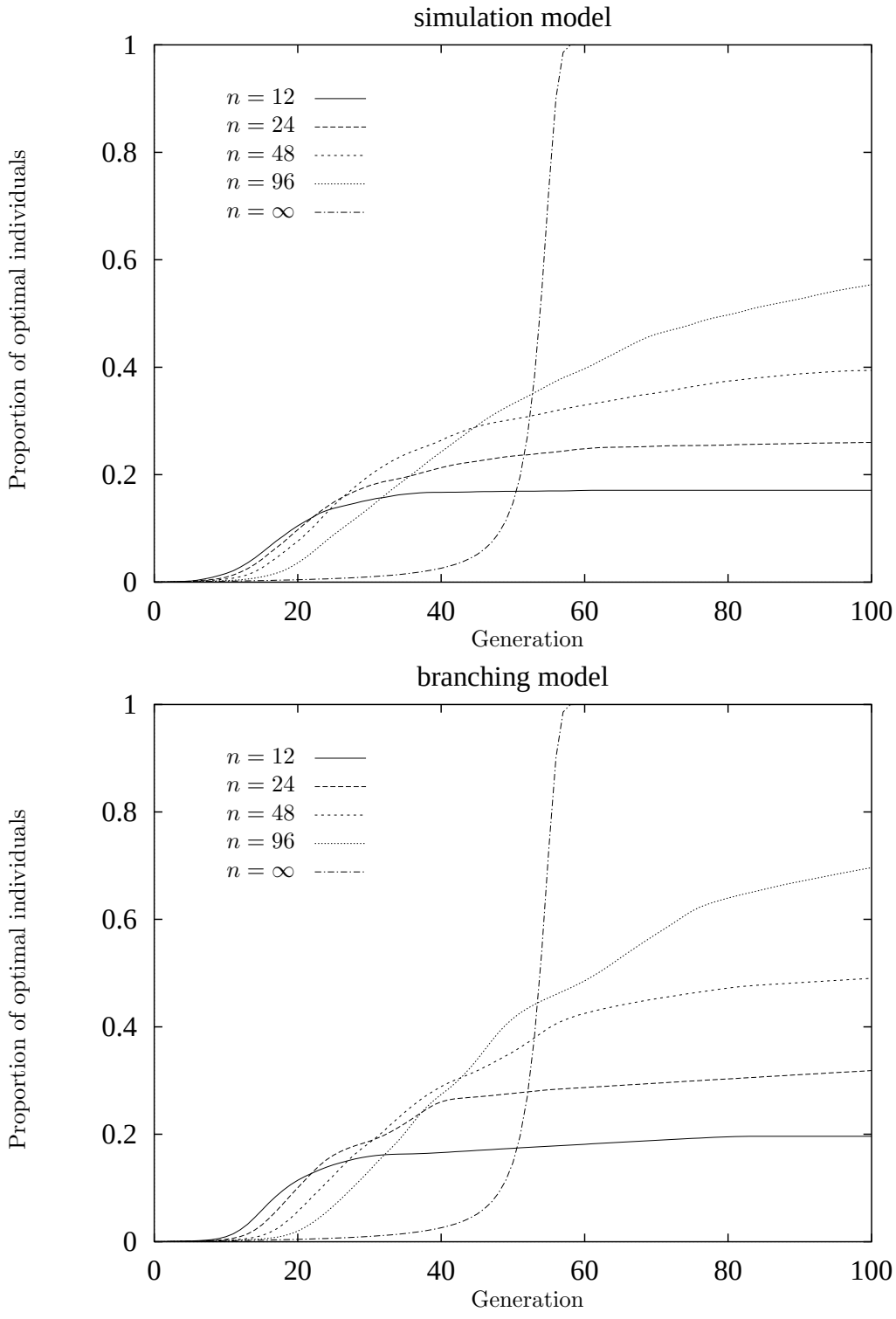

Figure 16: Proportion of optimal solutions for triple-competition both for the simulation model (top) and the branching model (bottom).

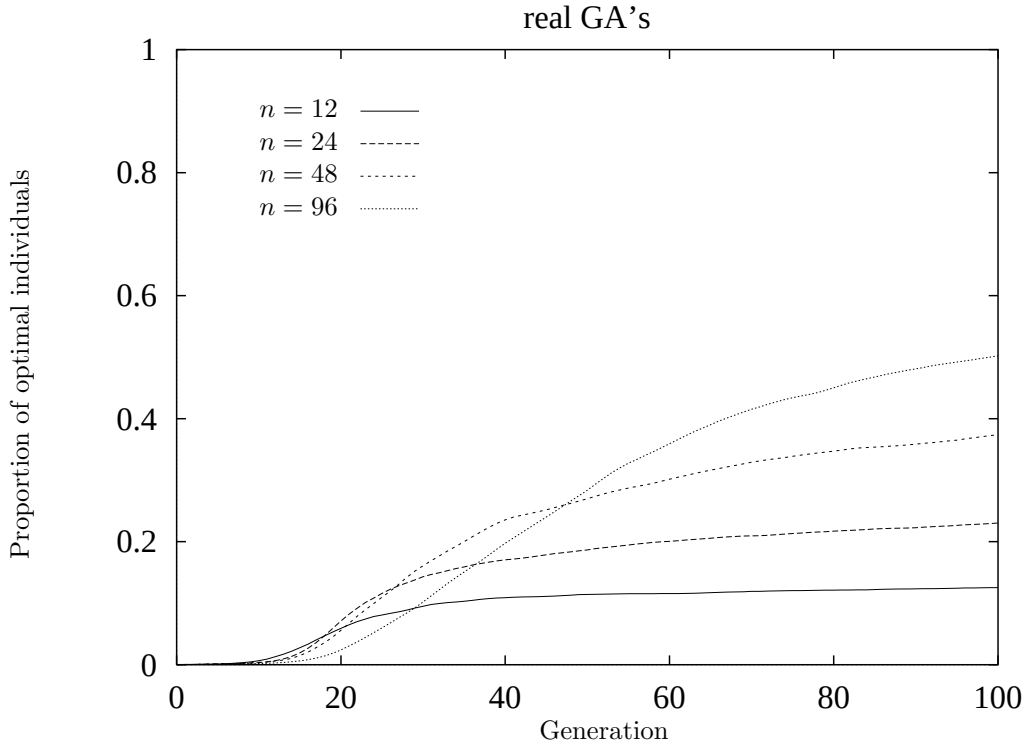

Figure 17: Proportion of optimal solutions for triple-competition.

We consider the population size n of the $(\mu \dot{+} \lambda)$-selection to be equal to μ, because this is the size of the smallest population that is used to pass information to subsequent generations; Therefore, this is considered as an appropriate measure for the maximal amount of information that can be propagated to subsequent generations. Given this definition of n the amount of computation required to evaluate a generation G_t, where $t > 0$, differs per algorithm. The generational GA's, the $(\mu + \lambda)$-selection, and the elitist recombination use n function evaluations per generation, the (μ, λ)-selection uses $n\lambda/\mu$ function evaluations per generation ($\lambda = 7\mu$), and the triple-competition uses $n/3$ function evaluations per generation.

5. Experiments

In this section the results of the experiments are discussed.

The simulation model can be applied directly to the problem. In case of the branching model the set S has to be chosen. Here, we define the set S to correspond to exactly those elements of $\mathcal{S}$ that contain an optimal building block in the deceptive part. Because elements of the set S are relatively likely to result in convergence to the global optimum, it is of interest to know which proportion of the population is part of this set. This proportion might differ between different runs, but it is possible to estimate the probabilities that a certain proportion of the population is contained in the set S in generation t. The probability that k individuals out of a set of λ offspring belong to the set S is given by the binomial distribution $\mathrm{Bin}(\lambda, k, p)$ where p denotes the expected proportion of elements that belong to the set S. Given that k out of λ offspring belong to the set S the actual proportion is $p' = k/\lambda$. For large populations p' will be close to p, but for small population sizes the difference between these two might be rather larger.

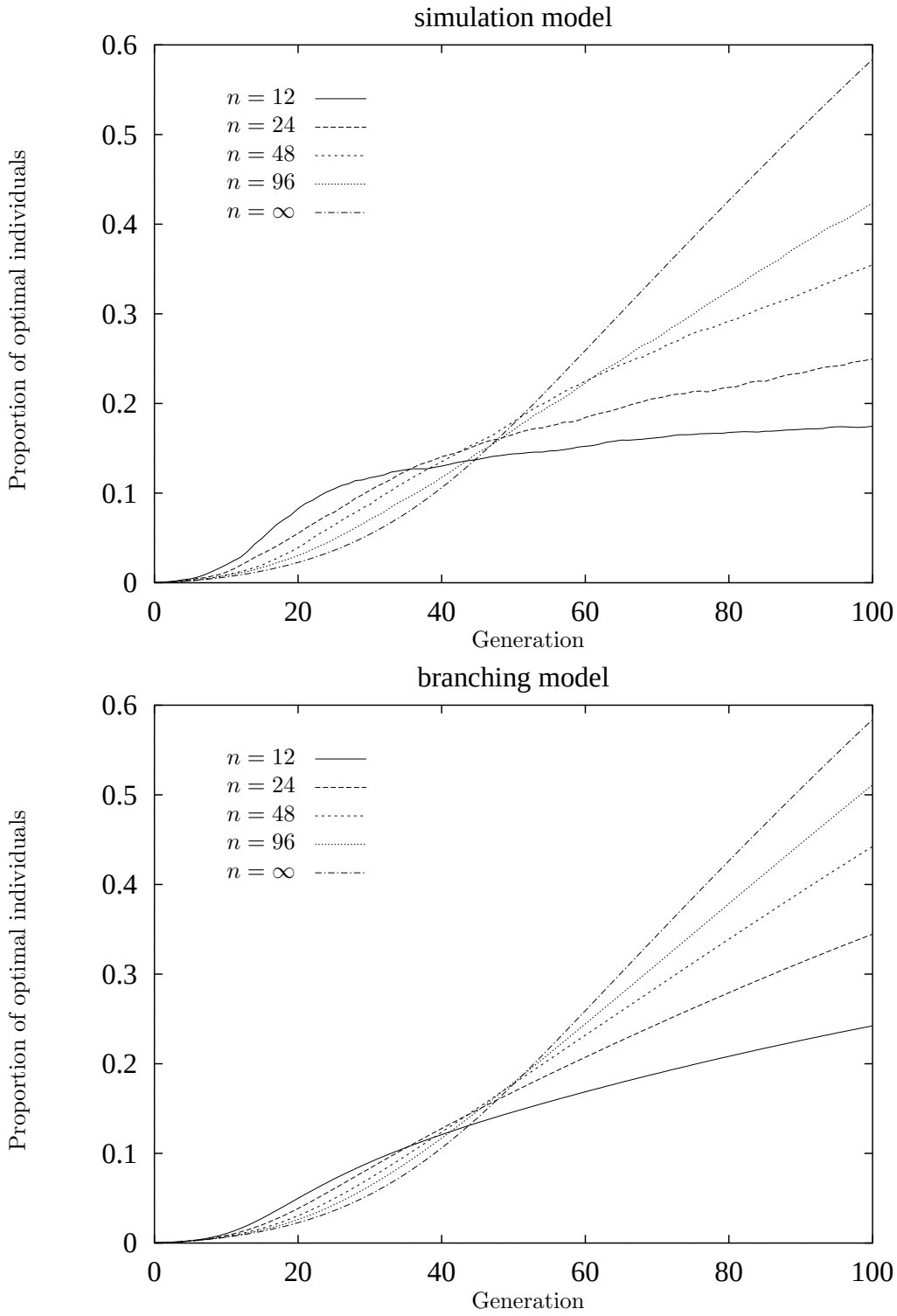

Figure 18: Proportion of optimal solutions for elitist recombination both for the simulation model (top) and the branching model (bottom).

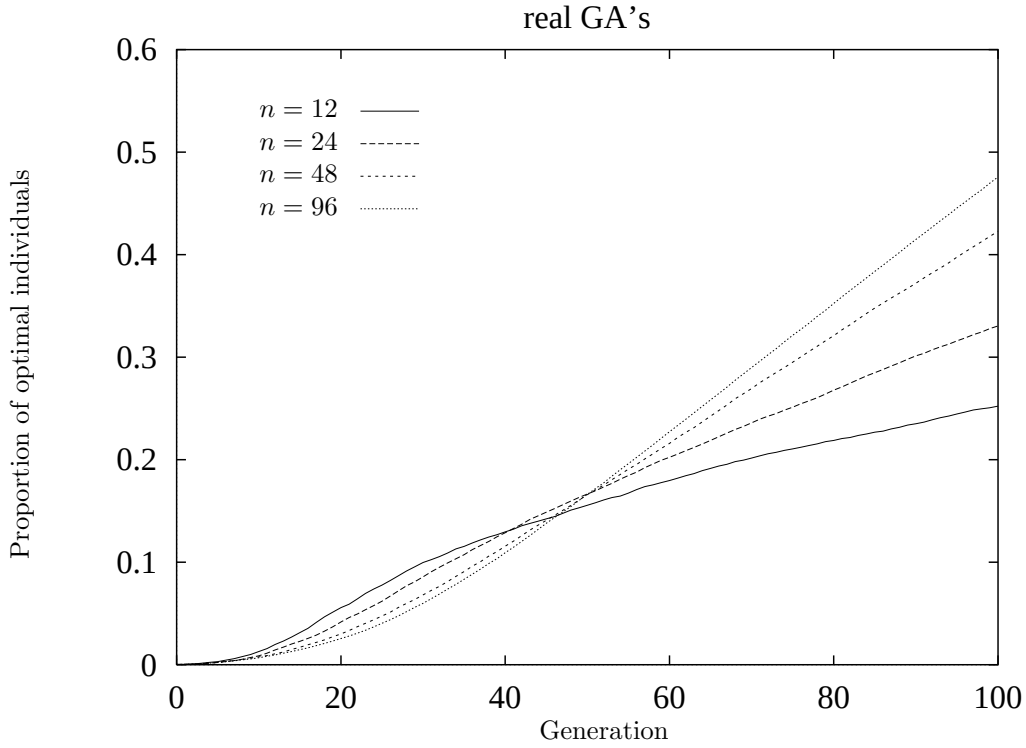

Figure 19: Proportion of optimal solutions for elitist recombination.

Both tracing of the models and real experiments have been performed for all the selection schemes discussed in previous sections. The modelling is done by tracing the behaviour of both the finite population model and the infinite population model. Plots are shown for population sizes $n = 12, 24, 48, 96,$ and ∞. (Recall that n refers to the number of parent individuals that are used to create the set of offspring.) The results for the simulation model and the real GA's are obtained by computing the averages over 1,000 independent runs.

For all selection schemes a number of plots are shown. The first plot shows the results for the simulation model, the second plot the results for the branching model, the third plot shows results of the real GA, and the fourth plot shows the results for a real GA with reduction of selection variance (if applicable). In these plots the horizontal axis shows the index of the generation, and the vertical axis shows the proportion of optimal individuals.

Figures 8 and 9 show the results when using a generational genetic algorithm with fitness proportional selection. This genetic algorithm is not likely to find the global optimal solution. The results of the simulation model are similar to the results obtained with the real GA's. The simulation corresponds best to the implementation with reduction of selection variance. In the branching model the performance of the GA deteriorates when the population size is increased. The same behaviour is observed for the real GA's. The branching model underestimates the probability that the optimum is located.

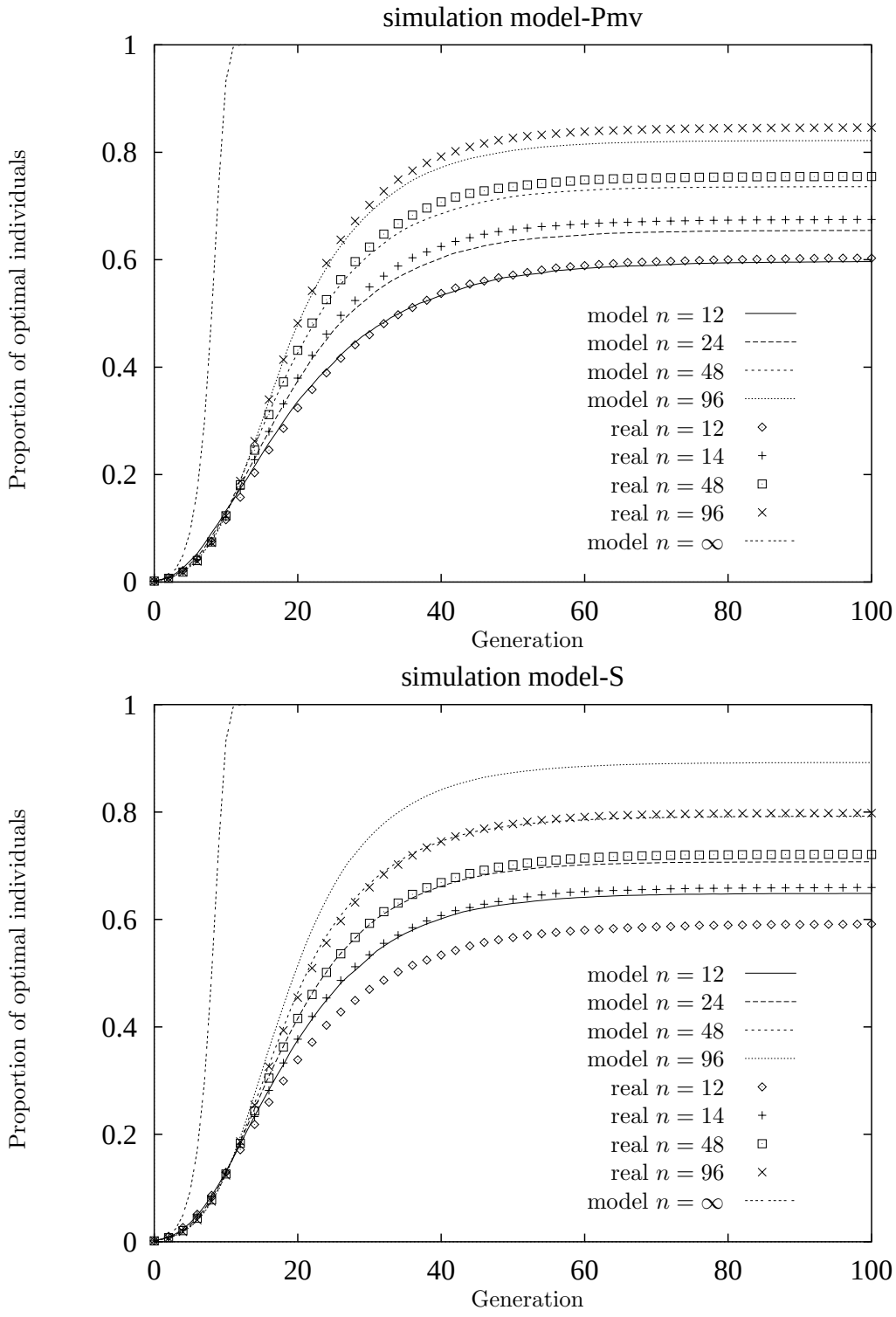

Figure 20: Problem 2: proportion of optimal solutions for the generational genetic algorithm with fitness proportional selection for real GA with reduction of sampling errors during selection and the a model simulating finite recombination (top) and for a real GA without reduction of sampling erros during selection and a model simulating finite selection (bottom).

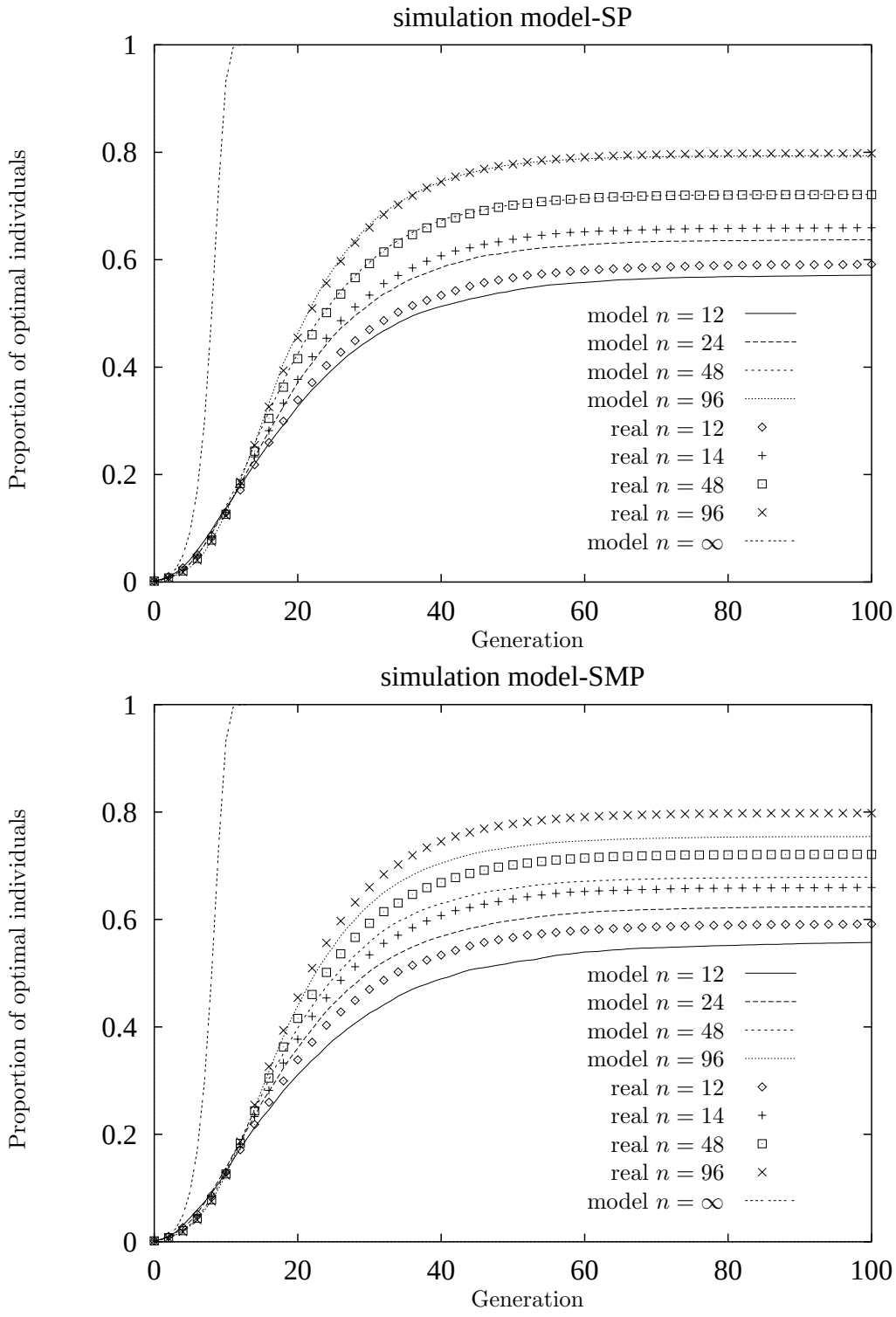

Figure 21: Problem 2: proportion of optimal solutions for the generational genetic algorithm with fitness proportional selection for real GA without reduction of sampling errors during selection and the simulating finite selection and recombination (top) and when simulating finite selection, mating, and recombination (bottom).

Figures 10 and 10 show the results when applying tournament selection. Tournament selection results in a very fast convergence. The optimal solution should be found fast, otherwise it will not be found at all. We see roughly the same qualitative behaviour, but the actual probabilities differ between the simulation and the experiments.

Figures 12 and 13 show the results for the (μ, λ)-selection. The qualitative behaviour of the simulated and experimental results match well. The results for the real GA's show that selection variance is small for this selection scheme. This is to be expected, because the parents are selected more often than in case of the generational GA's. As a result, the ratio between the expected and the actual number of copies of a parent is likely to be relatively close to one.

Figures 14 and 15 show the results for a $(\mu + \lambda)$-selection. The simulation model predicts the results of the runs of the real GA's very well. The branching model overestimates the proportions of optimal solutions for all population sizes. If we compare the real GA with and without reduction of selection variance, then the differences are small. This is again due to the fact that well-performing individuals produce many copies. A well-performing individual might get no chance to reproduce once in a while, but if this individual survives during several generations, then this event does not have a strong influence on the progress of the evolution process. For this reason we expect that minimization of sampling errors during selection is not very important for the $(\mu + \lambda)$-selection.

Figures 16 and 17 show the results for the triple-competition. Both models show approximately the same results as the real GA. Again the branching model overestimates the proportions during all runs.

Figures 18 and 19 show the results for the elitist recombination algorithm. Elitist recombination converges quite slow, when compared to the other GA's. The results of the models match the results of the real GA run reasonably. In the model we only sample over the offspring. However, in the elitist recombination parents and offspring are in direct competition. Whether the parents survive depends on the fitness of their direct offspring. Therefore, computing both distributions independently is not correct. This discrepancy between the models and the real elitist recombination is probably the reason for the difference between the results predicted by the models and the results obtained from the real GA runs. If we would like to model this interaction correctly, then we should generate the distribution of the parents dependent upon the distribution of the offspring after sampling. In order to compute this parent distribution we should keep track of the which parents produced which offspring.

The branching model assumes a building block processing approach to optimization (recall that set S corresponds to those individuals that contain the optimal building block in the deceptive part). If we compare the results of the branching model to the results of the real runs, then a good match is observed for all GA's, except for the generational GA's. When using the simulation model these generational GA's are also modelled quite well, so the SP-boxes seem to work correctly. An explanation for the difference is that the underlying assumption of the branching model is not true for the two generational genetic algorithms: the generational GA's do not process the order-six deceptive building block. This explanation is further strengthened by the fact that the generational GA's are not well able to optimize this problem.

A second smaller problem instance is introduced in which the oneMax part consists of six bits and the deceptive part consists of three bits. The generational GA with fitness proportional selection is applied to this problem. The results are shown in Figures 20 and 21. The upper graph of Figure 20 shows the results for the model (given by the lines) and the results for the real GA with reduction of sampling errors during selection (given by the markers). The match between the model and the real GA is good, especially for small population sizes. The lower graph shows the results when using sampling of the distribution of initial population and sampling of the distributions after the selection step by means of a sampling step, while recombination is done by means of the transmission function model. The sampling errors by selection are here modelled explicitly, and hence we do added the results of a GA without reduction of the sampling error during selection to this plot. The model in this case results in too optimistic predictions. This effect can be explained by means of the following example. Let A be a highly fit string, while B is a string with a very low fitness. Assume that the recombination produces a distribution containing an equal proportion of strings A and B and that this proportion is smaller than $1/n$. After the

selection the proportion of A can easily be larger than $1/n$, while the proportion of B is close to zero. Hence, if we only sample after selection, then it is not surprising that the resulting model results in too optimistic predictions.

Figure 21 shows the results for the real GA without reduction of the sampling error during selection. The upper graph shows the results when simulating a finite selection and a finite recombination by means of sampling. The results predicted by the model match well with the results obtained with the real GA's. The lower graph shows the results when simulating a finite selection, mating and recombination. In this case the match between the model and the results of the actual GA is not so good. We expect this to be due to the fact that our model assumption, which is that the individuals are generated independently, does not hold here.

The best performance is obtained when using finite recombination, as was already suggested by the theoretical analysis. The distribution after recombination has quite a large spread and therefore the discrepancies are quite large at this point during evolution. Furthermore we observe a good match for the simulation-P with a GA with reduction of sampling errors during selection, and for the simulation-SP with a real GA without this reduction of sampling errors. This indicates that we can make a separation between these two sampling errors by means of our model.

6. Conclusions

Infinite population models can behave quite differently from their finite population counterparts. We conjecture that the influence of population size is strongest when the expected distribution of the population has the largest spread.

We introduced two models of finite population GA's, applied these models to two test-problems, and compared the behaviour of the models to the behaviour shown by real GA's. The simulation model predicts the behaviour of all selection schemes well. The branching model only fails on the generational GA's, which suggests that generational GA's are not able to process order-six building blocks efficiently.

The experiments show that modelling the consequences of finite populations during recombination gives a good match between models and real GA's, and that modelling finite population after the selection step results in a less good match.

In case of generational GA with fitness proportional selection we have shown how to model the consequences of the finite population during selection and during recombination (both together and separately). The results further strengthened our suspicion that finite population sizes have a strong influence during the recombination phase.

References

[Alt94] L. Altenberg. The evolution of evolvability in genetic programming. In Kenneth E. Kinnear, Jr., editor, *Advances in Genetic programming*, chapter 3, pages 47–74. M.I.T. Press, 1994.

[BHmS91] T. Bäck, F. Hoffmeister, and H.-P. Schwefel. A survey of evolution strategies. In R.K. Belew and L.B. Booker, editors, *Proceedings of the 4th International Conference on Genetic Algorithms*, pages 2–9. Morgan Kaufmann, 1991.

[GDC93] D.E. Goldberg, K. Deb, and J.H. Clark. Accounting for noise in the sizing of populations. In G. Rawlins, editor, *Foundations of Genetic Algorithms-2*, pages 127–151. Morgan Kaufmann, 1993.

[KKPT98] C.H.M. Van Kemenade, J.N. Kok, J.A. La Poutré, and D. Thierens. Transmission function models of infinite population genetic algorithms. Technical Report SEN-R98xx, CWI, Amsterdam, 1998.

[Rec94] I. Rechenberg. *Evolutionsstrategie '94*. Frommann-Holzboog, 1994.

[Sch95] H.-P. Schwefel. *Evolution and Optimum Seeking*. Sixth-Generation Computer Technology Series. Wiley, New York, 1995.

[TG94] D. Thierens and D.E. Goldberg. Elitist recombination: an integrated selection recombi-

nation GA. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, pages 508–512. IEEE Press, 1994.

[vK97] C.H.M. van Kemenade. Modeling elitist genetic algorithms with a finite population. In *Proceedings of the Third Nordic Workshop on Genetic Algorithms*, pages 1–10, 1997.