Crossover is Probably Essential for the Ising Model on Trees

Dirk Sudholt
Fachbereich Informatik
Universität Dortmund
44221 Dortmund, Germany
Dirk.Sudholt@uni-dortmund.de

ABSTRACT

Due to experimental evidence it is incontestable that crossover is essential for some fitness functions. However, theoretical results without assumptions are difficult. So-called real royal road functions are known where crossover is proved to be essential, i. e., mutation-based algorithms have an exponential expected runtime while the expected runtime of a genetic algorithm is polynomially bounded. However, these functions are artificial and have been designed in such a way that crossover is essential only at the very end (or at other well-specified points) of the optimization process.

Here, a more natural fitness function based on a generalized Ising model is presented where crossover is essential throughout the whole optimization process. Mutation-based algorithms such as $(\mu+\lambda)$ EAs with constant population size are proved to have an exponential expected runtime while the expected runtime of a simple genetic algorithm with population size 2 and fitness sharing is polynomially bounded.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

General Terms

Theory, Algorithms, Performance

Keywords

Ising model, mutation vs. crossover, expected optimization time, theoretical analysis, fitness sharing

1. INTRODUCTION

In the history of evolutionary algorithms there have been long debates whether mutation or crossover is the "more important" search operator. Despite much experimental evidence of functions where crossover is indispensable it has long been an open question to prove rigorously (without assumptions) for a function that crossover is essential. Jansen and Wegener [7] present a simple fitness function and prove that a genetic algorithm optimizes the function in polynomial expected runtime while the expected runtime of mutation-based algorithms is superpolynomial.

In [6], Jansen and Wegener present a class of so-called *real royal road functions* where the performance gap between mutation-based algorithms and a genetic algorithm is even larger. They prove that mutation-based algorithms have an exponential expected runtime on real royal road functions while the expected runtime of a simple genetic algorithm is polynomially bounded. However, the population size of this genetic algorithm grows with the search space dimension.

Due to the large population size of Jansen's and Wegener's genetic algorithm, the question remains whether similar effects can be obtained if one restricts both types of algorithms to constant population sizes. Storch and Wegener [8] present another class of real royal road functions for constant population size where a genetic algorithm with the smallest possible population size, namely 2, suffices to obtain a polynomial expected runtime.

Both classes of real royal road functions have been constructed explicitly for the comparison of mutation and crossover and are rather artificial. In the analysis of genetic algorithms on real royal road functions, crossover is used only in one single step, namely in the very last step of the optimization process which creates a global optimum.

In this paper, we present another class of functions where $(\mu+\lambda)$ EAs with constant population size need an exponential runtime while a simple genetic algorithm with population size 2 and fitness sharing needs only polynomial expected runtime. In contrast to the functions presented by Jansen and Wegener and Storch and Wegener, this class of function has not been constructed explicitly to show the real royal road property; it has been derived more naturally from the investigation of generalized Ising models.

In Section 2, we define the generalized Ising model and the aforementioned class of fitness functions. Section 3 shows that $(\mu+\lambda)$ EAs with constant μ have an exponential expected runtime on these fitness functions. In Section 4, we define a simple genetic algorithm with fitness sharing, the (2+2) GA, and prove a polynomial upper bound for its expected runtime. Finally, Section 5 determines the practical behavior of the (2+2) GA with a closer look on the average runtime from an experimental perspective.

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'05, June 25–29, 2005, Washington, DC, USA.
Copyright 2005 ACM 1-59593-010-8/05/0006 ...$5.00.

2. THE ISING MODEL ON BINARY TREES

The Ising model from physics due to Ernst Ising [5] has become a popular model for the investigation of adaptation capabilities of evolutionary algorithms. It is based on an undirected graph $G = (V, E)$, $V = \{1, \ldots, n\}$ and a search point $x = (x_1, \ldots, x_n)$ represents a coloring of V. In its original form, an edge $e = \{u, v\}$ contributes the value $f_e(x) := w(e) \cdot x_u \cdot x_v$ to the fitness where $w(e)$ is the weight of the edge e and $x_u, x_v \in \{-1, +1\}$. The fitness of x, which is to be maximized, is the sum of all $f_e(x)$. In case of positive weights $w(e)$, the Ising model can be seen as an inverse graph coloring problem since the Ising function rewards monochromatic edges, i. e., edges $e = \{u, v\}$ where $x_u = x_v$.

Here, we only consider the simple case $w(e) = 1$ for all $e \in E$ and we apply an affine transformation to obtain the search space $\{0, 1\}^n$ instead of $\{-1, +1\}^n$. In this formulation, the Ising function Ising_G on $G = (V, E)$ simply counts the number of monochromatic edges. Since $x_u x_v + (1 - x_u)(1 - x_v) = 1$ iff $x_u = x_v$, the Ising function can be defined as quadratic function

$$\text{Ising}_G := \sum_{\{u,v\} \in E} (x_u x_v + (1 - x_u)(1 - x_v)).$$

An important property of Ising_G is *bit-flip symmetry* or *spin-flip symmetry*, i. e., $f(x) = f(\overline{x})$ if $\overline{x}$ is the bitwise complement of x.

The optimization of the Ising function is trivial since 0^n and 1^n are always global optima. However, on most graph classes it is likely that, due to bit-flip symmetry, different parts of the graph may be colored with different colors. Connected subgraphs can be seen as building blocks and thus, 0-colored building blocks compete with 1-colored building blocks. This may lead to problems called *synchronization problems* by Goldberg, van Hoyweghen, and Naudts [4].

The two most known graph classes for the Ising model are the one-dimensional Ising model, also called ring, and the two-dimensional Ising model, also called torus. Fischer and Wegener [3] investigate the one-dimensional Ising model and compare the effects of mutation and crossover. They prove an upper bound of $O(n^2)$ for a genetic algorithm and an upper bound of $O(n^3)$ for the (1+1) EA which is asymptotically sharp under a reasonable assumption. The two-dimensional Ising model has been investigated by Fischer [2] and Briest et al. [1]. The latter also investigate other graph classes like partially connected cliques and the Boolean hypercube.

Here, we investigate another graph class, the class of complete binary trees. Let $G = (V, E)$ be a complete binary tree, then Ising_G can be seen as a hierarchical function where all subtrees represent building blocks. Since two-point crossover should be able to select building blocks, we choose an encoding that ensures that all vertices of a subtree form a coherent sequence within the bit string. This aim is accomplished by enumerating the vertices in V with an inorder traversal as seen in Figure 1. Note that we draw trees in downward direction, with the root at the top.

The hierarchical building block structure sketched in Figure 1 resembles the building block structure of H-IFF, see, e. g., Watson and Pollack [9]. Both functions are hierarchically consistent as defined in [9]. However, in H-IFF, the building blocks fit together tightly and the fitness contribution of monochromatic building blocks differs from the Ising function. While in the Ising function on trees, monochromatic building blocks of size m contribute the value $m - 1$ to the fitness, monochromatic building blocks of size m in H-IFF contribute the value $m \cdot ((\log m) + 1)$. Due to this superlinear function, H-IFF rewards few large monochromatic building blocks more than many small monochromatic building blocks. This strengthens the hierarchical structure of building blocks in H-IFF in comparison with the Ising function.

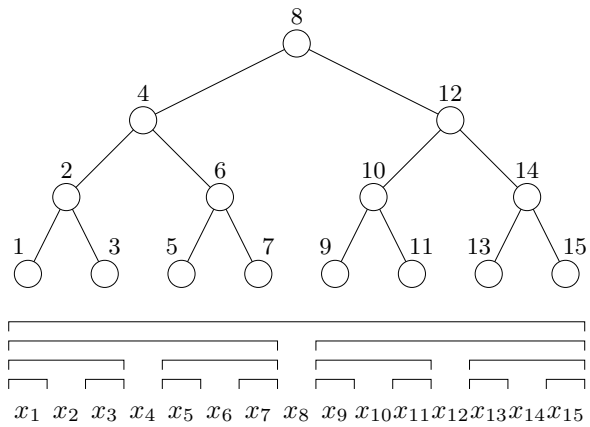

Figure 1: A visualization of the encoding of binary trees (top) and the resulting building block structure (bottom). The vertices are labelled with the inorder enumeration.

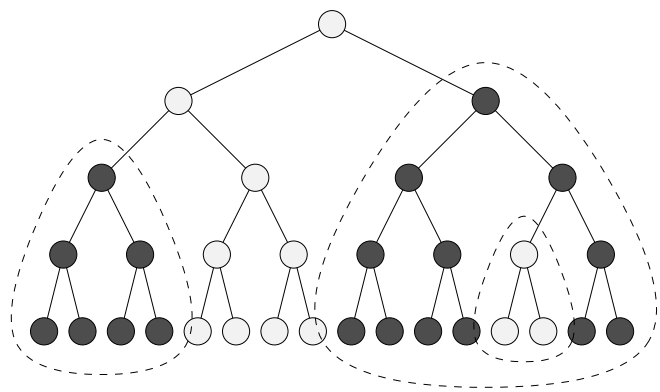

Figure 2: An example of a coloring with two colors. The coloring displayed contains three improving subtrees.

Let $x, y \in \{0, 1\}^n$, $n := |V|$, be colorings of V and $\{u, v\} \in E$, u parent of v, be dichromatic in x as well as in y. Let $T(v)$ be the subtree of v, then the bits belonging to $T(v)$ form a building block in x and y. If $T(v)$ is 0-colored in x and 1-colored in y, two-point crossover may exchange the colorings of the building block $T(v)$ in x and y, creating offspring x', y' where $\{u, v\}$ is monochromatic.

Since $T(v)$ may help to increase the fitness, $T(v)$ is called an *improving subtree* (see Figure 2 for an example).

Definition 1. Let $G = (V, E), n := |V|$, be a complete binary tree, $v \in V$, and $x \in \{0, 1\}^n$. A subtree $T(v)$ is called an *improving subtree* in x iff there is a parent $u \in V$ of v and $x_u \neq x_v$.

We can relax the condition that $T(v)$ is monochromatic in x as well as in y as follows. If $T(v)$ is an improving tree in x as well as in y and $x_v \neq y_v$, two-point crossover selecting $T(v)$ for exchange creates offspring x', y' where $\{u, v\}$ is monochromatic. Since the colorings of $T(v)$ are exchanged between x and y, the sum of the fitness values increases yielding $f(x') + f(y') = f(x) + f(y) + 2$. In case x', y' replace their parents x, y in the population, the fitness of the population increases.

We further remark that there are $\binom{n}{2}$ possibilities for choosing two crossover points while there are only n building blocks representing subtrees. Most two-point crossover operations choose a sequence within the bitstring not corresponding to subtrees. However, in the following we focus our analysis on two-point crossover operations selecting the bits belonging to a single subtree.

3. EXPECTED RUN TIME OF $(\mu+\lambda)$ EAS

The class of $(\mu+\lambda)$ EAs contains algorithms with population size μ working as follows.

Algorithm 1 (Scheme of a $(\mu+\lambda)$ EA).

1. *Choose a multiset P of μ individuals uniformly at random.*
2. *Select a multiset $P' \subseteq P$ of λ individuals according to some selection strategy.*
3. *For all $y \in P'$: flip each bit in y independently with a fixed mutation probability p_m.*
4. *Select a multiset $P^* \subseteq (P \cup P')$ of μ individuals with maximal fitness value according to some selection strategy. $P := P^*$.*
5. *Repeat steps 2–4.*

We regard the algorithm as an infinite stochastic process and are interested in results on the expected runtime, i. e., the expected time until a global optimum is evaluated. We do not further specify the selection strategies since the following lower bound holds for all non-specialized strategies.

Theorem 1. *Let $G = (V, E)$ be a complete binary tree with $n := |V|$ vertices and depth d. The expected number of mutation steps of an arbitrary non-specialized $(\mu+\lambda)$ EA with $\mu = O(1)$ and mutation probability p_m on $f := \mathrm{Ising}_G$ is bounded below by $2^{\Omega(n)}$.*

Sketch of Proof. Due to bit-flip symmetry, a mutation operator with mutation probability $p_m \geq 1/2$ works like a mutation operator with mutation probability $1 - p_m \leq 1/2$. So, w. l. o. g., we assume $p_m \leq 1/2$.

If $p_m < 2^{-(4\mu+1)}$, we assume the $(\mu+\lambda)$ EA starts with a population of worst-case search points, i. e., in all $x \in P$, both subtrees of the root are monochromatic and colored with different colors. This event occurs with probability $\Omega(2^{-\mu n})$ and then, all individuals in P have fitness $n-2$. In such a situation, there are only $O(n)$ different search points in $\{0,1\}^n$ which can be accepted for the next generation, namely all search points with at most one improving tree. Since the Hamming distance from any worst-case search point to both global optima and to all search points with improving subtrees of depth $d' < d-1$ is at least $(n+1)/4$, the expected number of mutation steps until a non-worst case search point is created and accepted is bounded below by

$$\begin{aligned}&\Omega(2^{-\mu n}) \cdot p_m^{-(n+1)/4}/O(n)\\ &< \Omega(2^{-\mu n}) \cdot 2^{(n+1)\cdot(4\mu+1)/4}/O(n) = 2^{-\Omega(n)}.\end{aligned}$$

If $2^{-(4\mu+1)} \leq p_m \leq 1/2$, the $(\mu+\lambda)$ EA resembles purely random search and it is hard to hit a specific search point. With probability $1 - 2^{-\Omega(n)}$, the $(\mu+\lambda)$ EA starts with a population of non-optimal search points. The probability to create a global optimum out of an arbitrary non-optimal search point is $2^{-\Omega(n)}$. Thus, the expected number of mutation steps is bounded below by $(1 - 2^{-\Omega(n)}) \cdot 2^{\Omega(n)} = 2^{\Omega(n)}$. □

4. EXPECTED RUN TIME OF A SIMPLE GA WITH FITNESS SHARING

Fitness sharing derates the "real" fitness of individuals by dividing the fitness $f(x)$ by the sharing function $Sh(x, P)$. The sharing function $Sh(x, P)$ measures the closeness from x to all individuals in the population P and a common formulation for $Sh(x, P)$ is

$$Sh(x, P) = \sum_{y \in P} \max\{0, (1 - d(x, y)/\sigma)^\alpha\}$$

where $d(x, y)$ is a measure for the distance between x and y. Here, we choose the Hamming distance $d(x, y) := H(x, y)$ and $\alpha := 1$. The σ-value indicates up to which distance two individuals should share their fitness. We choose $\sigma := n$ if n is the search space dimension so that individuals always share their fitness. Another consequence of this choice is that we can omit the max operator.

Definition 2. Let $H(x, y)$ be the Hamming distance between x and y. If n is the search space dimension and f is the real fitness, we define the sharing function as $Sh(x, P) := \sum_{y \in P}(1 - H(x, y)/n)$. The fitness of x with fitness sharing w. r. t. the population P is defined as

$$f(x, P) := \frac{f(x)}{Sh(x, P)} = \frac{f(x)}{\sum_{y \in P}(1 - H(x, y)/n)}.$$

The fitness of the population is then $f(P) := \sum_{x \in P} f(x, P)$.

If $P = \{x, y\}$, the definition of $f(P)$ can be simplified to

$$f(P) := \frac{f(x) + f(y)}{2 - H(x, y)/n}.$$

This term consists of two components: the Hamming distance $H(x, y)$ and the sum of the fitness values $f(x) + f(y)$. We will refer to the latter as the *fitness component*.

Now, we define a (2+2) GA with fitness sharing resembling the algorithm analyzed by Fischer and Wegener [3].

Algorithm 2 ((2+2) GA with fitness sharing).

1. *Choose $x, y \in \{0,1\}^n$ uniformly at random. $P := \{x, y\}$.*
2. *With probability $1/2$ execute only Step 3a, else execute only Step 3b.*

3a $(x', y') :=$ *two-point-crossover*(x, y). $P' := \{x', y'\}$.

3b $x' := mutate(x)$, $y' := mutate(y)$. $P' := \{x', y'\}$.

4. Replace P *by* P' *iff* $f(P') \geq f(P)$.

5. Repeat Steps 2–4.

The mutation operator flips each bit independently with probability $p_m := 1/n$. With a probability of approximately $1/e$, $e = 2.718\ldots$ the Eulerian constant, the mutation operator creates an offspring where no bit is flipped. So, the algorithm has the chance to execute steps where only one individual is really modified.

The selection operator selects either $P = \{x, y\}$ or $P' = \{x', y'\}$; these two populations are compared as a whole. This differs from the common strategy where a population $\hat{P} := P \cup P'$ of both parents and offspring is created and the individuals are evaluated with respect to $\hat{P}$. By comparing $f(P')$ with $f(P)$ directly, we evaluate both parents and offspring in their corresponding contexts and make sure the fitness of the current population is monotone over time.

THEOREM 2. *Let* $G = (V, E)$ *be a complete binary tree with* $n := |V|$ *vertices and depth* d. *The expected time until the (2+2) GA with fitness sharing on* Ising_G *reaches the optimal population* $\{0^n, 1^n\}$ *is bounded by* $O(n^3)$.

PROOF. We prove the theorem by showing that given a non-optimal population P, the probability of increasing $f(P)$ by at least $1/64$ in one generation is $\Omega(1/n^2)$. Since $f(P)$ is monotone and the maximum value is $f(\{0^n, 1^n\}) = 2n - 2$, the expected time until $P = \{0^n, 1^n\}$ is reached is bounded by $64 \cdot (2n-2) \cdot O(n^2) = O(n^3)$.

First, we compute the change in the $f(P)$ value according to changes in the two components. Let $P = \{x, y\}$ be the current population and $P' = \{x', y'\}$ be the population created in Step 3. Let $\Delta H := H(x', y') - H(x, y)$ and $\Delta f := (f(x') + f(y')) - (f(x) + f(y))$. Then,

$$f(P') - f(P) = \frac{f(x) + f(y) + \Delta f}{2 - (H(x,y) + \Delta H)/n} - \frac{f(x) + f(y)}{2 - H(x,y)/n}$$
$$= \frac{\Delta f(2 - H(x,y)/n) + \Delta H(f(x) + f(y))/n}{(2 - (H(x,y) + \Delta H)/n)(2 - H(x,y)/n)}.$$

Since the denominator is bounded above by 4,

$$f(P') - f(P) \geq \frac{\Delta f(2 - H(x,y)/n) + \Delta H(f(x) + f(y))/n}{4}$$
$$= \frac{\Delta f(2n - H(x,y)) + \Delta H(f(x) + f(y))}{4n} \quad (*)$$

if the nominator is positive (this will be the case in all applications of $(*)$).

An operation creating P' out of P is called an *improving operation* if $f(P') - f(P) \geq 1/64$. In different situations, there are different types of improving operations. We now present four cases of non-optimal populations and show for each case that the probability of an improving operation is $\Omega(1/n^2)$.

Case 1. $H(x, y) = n$.

Since $P \neq \{0^n, 1^n\}$ and $x = \overline{y}$, there is at least one subtree $T(v)$ such that $T(v)$ is an improving subtree in x as well as in y and $x_v \neq y_v$. Two-point crossover selecting $T(v)$ yields $\Delta f = 2$ and $\Delta H = 0$. The probability of such an operation is $\Omega(1/n^2)$ and

$$f(P') - f(P) \overset{(*)}{\geq} \frac{2n}{4n} = \frac{1}{2}.$$

Case 2. $H(x, y) < n$ and $H(x, y) + f(x) + f(y) < (3n-3)/2$.

We show that there is a 1-bit-mutation increasing the (real) fitness of a search point by at least 1.

Let z be a search point without fitness-improving 1-bit-mutations and let $\deg_z^+(v)$ for $v \in V$ denote the number of monochromatic edges adjacent to vertex v w. r. t. the coloring z. Then the root and the $(n+1)/2$ leaves must be adjacent to at least one monochromatic edge and the $(n-3)/2$ other vertices must be adjacent to at least two monochromatic edges. Hence,

$$\sum_{v \in V} \deg_z^+(v) \geq 1 + \frac{n+1}{2} + 2 \cdot \frac{n-3}{2} = \frac{3n-3}{2}.$$

Since the sum counts all monochromatic edges twice, $f(z) \geq (3n-3)/4$. If there is no fitness-improving 1-bit-mutation in x and y, this implies $f(x) + f(y) \geq (3n-3)/2$ in contradiction to $H(x, y) + f(x) + f(y) < (3n-3)/2$.

We have shown that there is a fitness-improving 1-bit-mutation in a search point $z \in \{x, y\}$. If the (2+2) GA executes Step 3*b*, performs a fitness-improving 1-bit-mutation on z and a 0-bit-mutation on the other search point, $\Delta f \geq 1$ and $\Delta H \geq -1$. The probability for such an operation is $\Omega(1/n)$ and

$$f(P') - f(P) \overset{(*)}{\geq} \frac{2n - H(x,y) - f(x) - f(y)}{4n}$$
$$> \frac{2n - (3n-3)/2}{4n} > \frac{1}{8}.$$

Case 3. $H(x, y) < n$ and $H(x, y) + f(x) + f(y) > (33/16)n$.

In this case, we rely on 1-bit-mutations increasing the Hamming distance at the expense of the fitness component. To be precise, we show that there is a 1-bit-mutation flipping a vertex u in a search point $z \in \{x, y\}$ yielding $\Delta H = 1$ and $\Delta f \geq -1$, so the loss in the fitness component is rather small.

Since $H(x, y) < n$ there is a vertex v with $x_v = y_v$. If v is a leaf or if there is a dichromatic edge adjacent to v in a search point $z \in \{x, y\}$, we choose $u := v$ and a 1-bit-mutation flipping z_u has the desired properties. Otherwise, if v is an inner vertex and all edges incident on v are monochromatic in x and y, there is a child v' of v and we consider v' instead of v. This process terminates with an appropriate vertex u at the latest in case a leaf is reached.

If the (2+2) GA executes Step 3*b*, performs a 1-bit-mutation flipping z_u on z and a 0-bit-mutation on the other search point,

$$f(P') - f(P) \overset{(*)}{\geq} \frac{-2n + H(x,y) + f(x) + f(y)}{4n}$$
$$> \frac{-2n + (33/16)n}{4n} = \frac{1}{64}.$$

The probability for such an operation is $\Omega(1/n)$.

Case 4. $H(x, y) < n$ and $(3n-3)/2 \leq H(x, y) + f(x) + f(y) \leq (33/16)n$.

In this special case, we cannot rely on 1-bit-mutations described in Cases 2 and 3 since the existence of fitness-improving 1-bit-mutations cannot be guaranteed and even if these 1-bit-mutations exist, the gain in $f(P)$ is very low. Hence, we search for more gainful operations, e. g. operations where $\Delta f + \Delta H > 0$.

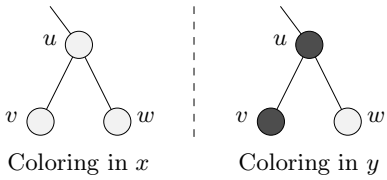

Figure 3: An example of a subtree of depth 1 and its partial coloring in x and y.

Operations increasing one component and not decreasing the other one yield a gain in $f(P)$ of at least $1/8 - O(1/n)$, since $H(x,y) + f(x) + f(y) \geq (3n-3)/2 \Rightarrow f(x) + f(y) \geq n/2 - O(1)$ and in $(*)$ both Δ-values are weighted by terms of at least $1/8 - O(1/n)$. For sufficiently large n, $1/8 - O(1/n) \geq 1/64$, so that these operations are improving operations.

We now describe three scenarios of partial colorings where improving operations can be found. Afterwards, we prove the existence of sufficiently many instances of these scenarios yielding a lower bound of $\Omega(1/n^2)$ for the probability to execute an improving operation.

Scenario 1. There is an inner vertex v where all incident edges are dichromatic in a search point $z \in \{x, y\}$.

Since $\deg(v) \geq 2$, the fitness value $f(z)$ is raised by at least 2 if a 1-bit-mutation flips z_v. The (2+2) GA may execute Step 3*b*, perform a 1-bit-mutation of z_v and flip no bit in the other search point, yielding $\Delta f \geq 2$ and $\Delta H \geq -1$. The probability for such an operation is $\Omega(1/n)$ and

$$\begin{aligned} f(P') - f(P) &\overset{(*)}{\geq} \frac{4n - 2H(x,y) - f(x) - f(y)}{4n} \\ &= \frac{4n - (H(x,y) - f(x) - f(y)) - H(x,y)}{4n} \\ &> \frac{4n - (33/16) \cdot n - H(x,y)}{4n} > \frac{15}{64}. \end{aligned}$$

Scenario 2. There is a subtree $T(v)$ such that $T(v)$ is an improving tree in x as well as in y and $x_v \neq y_v$.

Two-point-crossover selecting $T(v)$ yields $\Delta f = 2$ and $\Delta H = 0$. The probability for such an operation is $\Omega(1/n^2)$ and $f(P') - f(P) \geq 1/64$.

Scenario 3. There is a subtree $T(u)$ of depth 1 (two leaves v, w with their common parent u) such that

$$\neg(x_u = x_v = x_w \neq y_u = y_v = y_w)$$

(see Figure 3 for an example).

A coloring with $x_u = x_v = x_w \neq y_u = y_v = y_w$ is called a *locally optimal coloring* since in the subgraph induced by $T(u)$ both Hamming distance and fitness component is maximized. If $T(u)$ is not colored locally optimal, there is a mutation of at most 3 bits in $B := \{x_u, x_v, x_w, y_u, y_v, y_w\}$ leading to the nearest locally optimal coloring. A closer examination of all non-locally optimal colorings of bits in B reveals that this operation is an improving operation. This holds even if we pessimistically assume that if x_u or y_u is flipped, the edge from u to its parent becomes dichromatic in the corresponding coloring.

We only present the resulting Δ-values of a mutation leading to a nearest locally optimal coloring. If $T(u)$ is monochromatic in x as well as in y, $\Delta f \geq -1$ and $\Delta H = 3$ holds and it is easy to show that $f(P') - f(P) \geq 1/64$. Otherwise, the result of such a mutation is either $\Delta f \geq 0$ and $\Delta H \geq 1$ or, if the colorings of $T(u)$ are complementary, $\Delta f \geq 2$ and $\Delta H \geq 0$. So, in all cases there is an improving operation occurring with probability $\Omega(1/n^3)$.

We have shown that there are many different scenarios that guarantee improving operations. Due to the variety of scenarios, it is likely that in the current population, several instances of these scenarios are given. To complete the analysis of Case 4, we now show that there are more than cn improving operations for a constant $c > 0$. We do this by contradiction. If there are at most cn instances of Scenarios 1–3, we conclude that $H(x,y) + f(x) + f(y)$ is large, contradicting the assumption $H(x,y) + f(x) + f(y) \leq (33/16)n$.

Assume that there are at most cn improving operations described in Scenario 3. Since there are $(n+1)/4$ disjoint subtrees of depth 1, $s := (n+1)/4 - cn$ subtrees of depth 1 must be colored locally optimal. This is a clue for large values of both Hamming distance $H(x,y)$ and $f(x) + f(y)$. There are $3s = 3(n+1)/4 - 3cn$ vertices on the last two levels whose bits are complementary in x and y, thus

$$H(x,y) \geq 3(n+1)/4 - 3cn.$$

Let $E_{i,j} \subseteq E$ for $i < j$ be the set of edges $\{u, v\} \in E$ such that u and v each are located on levels in the interval $[i, j]$. For $E' \subseteq E$, let $f_{x,y}(E')$ be the number of edges in E' monochromatic in x plus the number of edges in E' monochromatic in y. It is obvious that $f(x) + f(y) = f_{x,y}(E)$.

In $E_{d-1,d}$, $2s = (n+1)/2 - 2cn$ edges are monochromatic in x as well as in y, thus contributing to $f(x) + f(y)$ an additive term of

$$f_{x,y}(E_{d-1,d}) \geq n + 1 - 4cn.$$

If, in addition, there are at most cn improving operations described in Scenario 2, we show that the edges in $E_{d-1,d-2}$ also contribute a large value to $f(x)+f(y)$. Let $e = \{u, v\} \in E_{d-2}$ be an edge leading to a locally optimal colored subtree $T(v)$ of depth 1. If e is dichromatic in x as well as in y, $T(v)$ is an improving subtree in x as well as in y and $x_v \neq y_v$. This corresponds to the situation described in Scenario 2; an example is sketched in Figure 4.

So, if there are at most cn improving operations in Scenario 2, at least $s - cn = (n+1)/4 - 2cn$ edges in E_{d-2} have to be monochromatic either in x or in y. The edges in E_{d-2} contribute to $f(x) + f(y)$ another additive term of

$$f_{x,y}(E_{d-2,d-1}) \geq \frac{n+1}{4} - 2cn.$$

The remaining edges in $E_{0,d-2}$ cannot all be dichromatic since we assume that there are at most cn improving operations in Scenario 1. Let $V'_z \subseteq V$ for $z \in \{x, y\}$ be the set of vertices v such that all edges incident to v are in $E_{0,d-2}$ and not all edges incident to v are dichromatic in z: $\deg^+_z(v) \geq 1$. If we subtract the numbers of vertices on levels $d, d-1$, and $d-2$, we obtain

$$|V'| = \left(n - \frac{n+1}{2} - \frac{n+1}{4} - \frac{n+1}{8}\right) - cn = \frac{n-7}{8} - cn$$

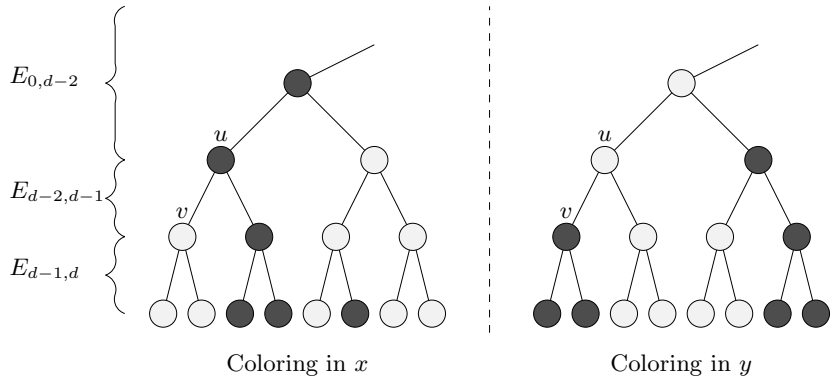

Figure 4: An extract of two colorings x and y and an edge $\{u, v\}$ leading to a locally optimal colored subtree $T(v)$ of depth 1, thus creating an instance of Scenario 2.

and

$$\sum_{v \in V'} \deg_z^+(v) \geq \frac{n-7}{8} - cn.$$

Since every edge monochromatic in z contributes at most 2 to this sum, the number of monochromatic edges in $E_{0,d-2}$ w. r. t. z is at least $(n-7)/16 - cn/2$. Thus,

$$f_{x,y}(E_{0,d-2}) = \frac{n-7}{8} - cn.$$

Adding up all three fitness contributions yields

$$\begin{aligned} f(x) + f(y) &= f_{x,y}(E_{0,d-2}) + f_{x,y}(E_{d-2,d-1}) + f_{x,y}(E_{d-1,d}) \\ &\geq n + 1 - 4cn + \frac{n+1}{4} - 2cn + \frac{n-7}{8} - cn \\ &> \frac{11}{8} \cdot n - 7cn. \end{aligned}$$

Hence,

$$H(x, y) + f(x) + f(y) > \frac{17}{8} \cdot n - 10cn$$

and, choosing $c \leq 1/160$, this contradicts $H(x, y) + f(x) + f(y) \geq (33/16) \cdot n$.

Since there are $\Omega(n)$ improving operations and every improving operation is executed with probability $\Omega(1/n^3)$, the probability of executing any improving operation is $\Omega(1/n^2)$ completing the analysis of Case 4 and proving the theorem. □

5. EXPERIMENTAL SUPPLEMENTS

The question remains whether the good performance of the (2+2) GA with fitness sharing is really based on crossover or if fitness sharing is the key component for an efficient optimization. If we modify the (2+2) GA with fitness sharing by omitting crossover and always executing Step 3*b*, we obtain the (2+2) EA with fitness sharing. The lower bound presented in Theorem 1 also holds for the (2+2) EA with fitness sharing (in the case of small p_m, we assume the population is initialized with two *complementary* worst-case search points).

In addition to this theoretical result, experiments were done to compare the two algorithms. Experiments for the (2+2) GA were run on complete binary trees of depth 1–10, i. e., $n = 3, 7, 15, \ldots, 2047$, the experiments for the (2+2) EA were run on trees of depth 1–3. The average runtime in independent runs was measured; the number of runs was 1,000 runs for each setup except for 100 runs for $n = 15$ and the (2+2) EA and 30 runs for $n = 2047$ and the (2+2) GA. The results are shown in Table 1.

n	(2+2) GA	(2+2) EA
3	39.019	2.511
7	127.950	61.955
15	471.141	26,739,085.010
31	2,029.305	–
63	9,672.528	–
127	47,511.313	–
255	224,340.916	–
511	1,072,733.151	–
1023	5,032,030.214	–
2047	22,223,437.900	–

Table 1: Average runtimes of the (2+2) GA with fitness sharing and the (2+2) EA with fitness sharing.

function class	a	mean square error
an^2	5.26988	$3.60934 \cdot 10^{10}$
$an^2 \log n$	0.69545	$4.49334 \cdot 10^{8}$
an^3	0.00262463	$5.57783 \cdot 10^{11}$

Table 2: The results of a regression analysis with the function classes an^2, $an^2 \log n$, and an^3.

For $n = 15$, the (2+2) GA with fitness sharing clearly outperforms the (2+2) EA with fitness sharing. Note that for very small n, $n \leq 7$, the opposite holds. This does not contradict our theoretical results since these results are asymptotic ones and they do not make assertions for small n such as $n \leq 7$. We conclude that, for n large enough, fitness sharing alone does not suffice for an efficient optimization.

Another interesting question is whether the upper bound $O(n^3)$ of Theorem 2 is sharp. Experiments show that in a typical run of the (2+2) GA with fitness sharing, the algorithm reaches a population with two complementary search points rather quickly. Then, the algorithm is likely to maintain the maximal Hamming distance and spends most of the runtime waiting for the right crossover operations. It is

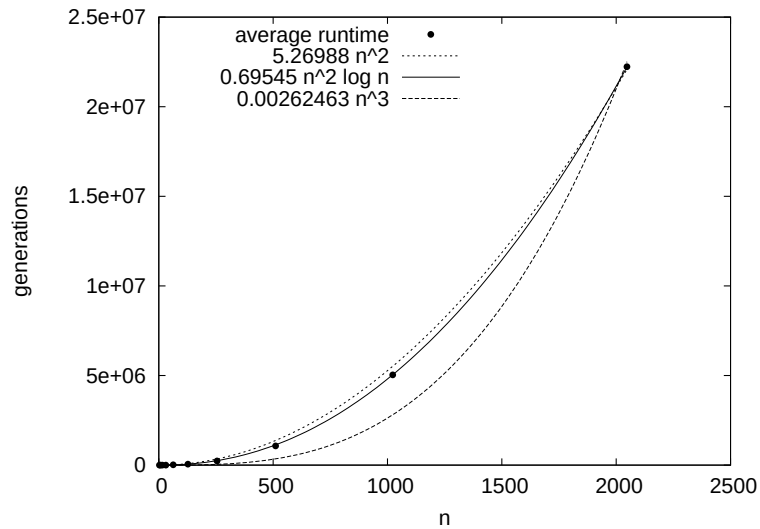

Figure 5: The average runtime of the (2+2) GA and the three fitted functions.

easy to show that, in case the algorithm maintains a maximal Hamming distance, the expected waiting time for the right crossover operations is bounded by $\sum_{i=1}^{n-2} 2/i \cdot \binom{n}{2} = O(n^2 \log n)$. So, we expect the true average runtime to be $\Theta(n^2 \log n)$.

The average runtimes for the (2+2) GA from Table 1 were used in a regression analysis with gnuplot 3.7 using the standard settings.

Among the function classes $an^2, an^2 \log n$, and an^3, the function $0.69545n^2 \log n$ resulted in the best fit with a mean square error of $4.49334 \cdot 10^8$. The results of the regression analysis are shown in Table 2, a plot of the data and the fitted functions is shown in Figure 5. The observable differences are in accordance with differences in mean square errors.

6. CONCLUSIONS

We have presented a fitness function derived from a generalized Ising model and proved that, for this specific fitness function, crossover is essential for evolutionary algorithms with constant population size. $(\mu+\lambda)$ EAs with constant population size using the common mutation operator have an exponential runtime. A simple (2+2) GA with fitness sharing is proved to have an expected runtime bounded by $O(n^3)$. Experiments show that this bound differs from the true average optimization time by a factor of $n/\log n$.

Moreover, both theory and experiments show that fitness sharing alone does not suffice for an efficient optimization implying that crossover is an essential component for the problem investigated here.

7. REFERENCES

[1] P. Briest, D. Brockhoff, B. Degener, M. Englert, C. Gunia, T. Jansen, O. Heering, M. Leifhelm, K. Plociennik, H. Röglin, A. Schweer, D. Sudholt, S. Tannenbaum, and I. Wegener. The Ising model: simple evolutionary algorithms as adaptation schemes. In *Proceedings of the Eighth Conference on Parallel Problem Solving from Nature (PPSN VIII)*, volume 3242 of *Lecture Notes in Computer Science*, pages 31–40. Springer-Verlag, 2004.

[2] S. Fischer. A polynomial upper bound for a mutation-based algorithm on the two-dimensional Ising model. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) 2004*, volume 3102 of *Lecture Notes in Computer Science*, pages 1100–1112. Springer-Verlag, 2004.

[3] S. Fischer and I. Wegener. The Ising model on the ring: Mutation versus recombination. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) 2004*, volume 3102 of *Lecture Notes in Computer Science*, pages 1113–1124. Springer-Verlag, 2004.

[4] D. E. Goldberg, C. Van Hoyweghen, and B. Naudts. Spin-flip symmetry and synchronization. *Evolutionary Computation*, 10:317–344, 2002.

[5] E. Ising. Beitrag zur Theorie des Ferromagnetismus. *Z. Physik*, 31:235–288, 1925.

[6] T. Jansen and I. Wegener. Real royal road functions – functions where crossover provably is essential. In L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001)*, pages 375–382. Morgan Kaufmann, 2001.

[7] T. Jansen and I. Wegener. On the analysis of evolutionary algorithms – a proof that crossover really can help. *Algorithmica*, 34(1):47–66, 2002.

[8] T. Storch and I. Wegener. Real royal road functions for constant population size. *Theoretical Computer Science*, 320:123–134, 2004.

[9] R. A. Watson and J. B. Pollack. Hierarchically consistent test problems for genetic algorithms. In P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, editors, *Proceedings of the Congress on Evolutionary Computation*, volume 2, pages 1406–1413, Mayflower Hotel, Washington D.C., USA, 6-9 1999. IEEE Press.