Level-Based Analysis of Genetic Algorithms and Other Search Processes

Dogan Corus, Duc-Cuong Dang, Anton V. Eremeev, and Per Kristian Lehre

***Abstract*—Understanding how the time complexity of evolutionary algorithms (EAs) depend on their parameter settings and characteristics of fitness landscapes is a fundamental problem in evolutionary computation. Most rigorous results were derived using a handful of key analytic techniques, including drift analysis. However, since few of these techniques apply effortlessly to population-based EAs, most time complexity results concern simple EAs, such as the (1+1) EA. We present the *level-based theorem*, a new technique tailored to population-based processes. It applies to any nonelitist process where offspring are sampled independently from a distribution depending only on the current population. Given conditions on this distribution, our technique provides upper bounds on the expected time until the process reaches a target state. The technique is demonstrated on pseudo-Boolean functions, the sorting problem, and approximation of optimal solutions in combinatorial optimization. The conditions of the theorem are often straightforward to verify, even for genetic algorithms and estimation of distribution algorithms which were considered highly nontrivial to analyze. The proofs for the example applications are available in the supplementary materials. Finally, we prove that the theorem is nearly optimal for the processes considered. Given the information the theorem requires about the process, a much tighter bound cannot be proved.**

***Index Terms*—Approximation, estimation of distribution algorithm (EDA), genetic algorithm (GA), runtime analysis.**

I. Introduction

THE THEORETICAL understanding of evolutionary algorithms (EAs) has advanced significantly over the last decades. Significant progress in developing and understanding a formal model of canonical genetic algorithms (GAs) and their generalizations was made in the nineties using dynamical systems [53]. Notably, the behavior of the dynamical systems model is closely related to the local optima structure of the problem in the case of binary search spaces [54]. However, most of these findings relate to the infinite population limit, from which it is difficult to derive statements about performance. Researchers from theoretical computer science argued in the early 2000s that EC theory had attempted to make either too general, or too precise statements [3]. Instead, one should develop techniques for deriving rigorous statements about worst-case optimization time, starting from the simplest possible settings. Much of the work assumed a population size of one and no crossover operator, e.g., the (1+1) EA [20].

Early analyses of EAs with larger population sizes often ignored the recombination operator. The family tree technique was introduced in [56] to analyze the (μ+1) EA. The performance of the (μ+μ) EA for different settings of the population size was analyzed in [30], using Markov chains to model the search processes, and in [4], using a similar argument to fitness levels. The analysis of parallel EAs in [35] also made use of the fitness levels argument. The inefficiency of standard fitness proportionate selection without scaling was shown in [36] and [44], using drift analysis [28]. In the recently introduced switch analysis, the progress of the EA is analyzed relative to a reference process that is easier to understand [58].

Runtime analyses considering recombination often aimed at understanding how evolutionary search can benefit from sexual reproduction. For the OneMax problem, it is known that the simple genetic algorithm (SGA) [53] is inefficient, even when crossover is enabled [45]. For a variant of the (μ+1) EA, crossover can speed up by a factor of 2 compared to the (1+1) EA [51]. GAs with even higher (nonconstant) speedups on OneMax are known, but they rely on nonconventional reproduction mechanisms [16], [17]. The unrestricted black-box complexity of OneMax is $\Omega(n/\log(n))$ [21], the speedup of any unrestricted black-box algorithm relative to the (1+1) EA is therefore at most $\mathcal{O}(\log(n)^2)$. More complex settings are required to show further speedups. So-called convex search algorithms, which include nonelitist GAs with gene pool recombination and no mutation, have been analyzed on *quasi-concave fitness landscapes* [41]. As a special case, convex search algorithm has expected runtime $\mathcal{O}(n \log n)$ on LeadingOnes, i.e., a speedup of $\Theta(n/\log(n))$ compared with the (1+1) EA. Prügel-Bennett *et al.* [47] considered nonelitist GAs, also without mutation, on noisy OneMax. The (μ+1) GA decreases the runtime on the Jump problem, however this was first only shown for artificially small crossover probabilities [31], [32]. For realistic crossover probabilities, the (μ+1) GA decreases the runtime by an exponential factor on instances of an FSM testing problem, however, this result assumes a deterministic crowding diversity mechanism [39]. It was recently shown that

Manuscript received October 27, 2016; revised February 21, 2017 and June 13, 2017; accepted July 29, 2017. Date of publication September 18, 2017; date of current version September 28, 2018. This work was supported in part by the European Union Seventh Framework Programme (FP7/2007-2013, SAGE) under Grant 618091, and in part by the Russian Foundation for Basic Research under Grant 15-01-00785 and Grant 16-01-00740. *(Corresponding author: Per Kristian Lehre.)*

D. Corus was with the School of Computer Science, University of Nottingham, Nottingham NG8 1BB, U.K., and is now with the Department of Computer Science, University of Sheffield, Sheffield S1 4DP, U.K.

D.-C. Dang is an Independent Researcher.

A. V. Eremeev is with the Sobolev Institute of Mathematics SB RAS, Omsk 644099, Russia.

P. K. Lehre is with the School of Computer Science, University of Birmingham, Birmingham B15 2TT, U.K. (e-mail: p.k.lehre@cs.bham.ac.uk).

This paper has supplementary downloadable multimedia material available at http://ieeexplore.ieee.org provided by the authors.

Digital Object Identifier 10.1109/TEVC.2017.2753538

the standard (μ+1) GA has a speed up of $\Omega(n/\log(n))$ on JUMP compared to the mutation-only variant (μ+1) EA [10].

Estimation of distribution algorithms (EDAs), a relatively new type of EAs [34], build explicit probabilistic models in every generation from the fittest individuals, from which the next generation is sampled. The sequence of probability distributions should converge to a distribution concentrated on the optimal search points. Traditional EAs can be seen as special cases of EDAs, where the probability distributions are given implicitly via their genetic operators and selection mechanisms. Most theoretical studies of EDAs have considered convergence and scalability properties [27], [43], [46], [50], [59], and rigorous runtime analyses of EDAs are still rare. Droste's $\Theta(Kn)$ bound on the expected runtime on linear functions for compact GA (cGA) with update parameter $K \geq n^{1+\varepsilon}$, is an early rigorous result on the runtime of EDAs [19]. Recent runtime analyses have demonstrated the noise robustness of cGA [25], as well as the impact of its update parameter [52]. The runtime of univariate marginal distribution algorithm (UMDA), a more complex EDA [42], has been analyzed in a series of papers [5]–[8], [13], [37], [57]. Chen *et al.* [6] described easy and hard functions for the UMDA under the so-called "no-random-error" assumption and with a sufficiently large population. This assumption was lifted in [8], but the analysis still assumed an unrealistically large population size, leading to a high bound on the expected runtime. It is usually necessary to impose margins on the probability distribution of the UMDA [7], however, the only known setting where the UMDA outperforms the (1+1) EA assumes the UMDA without margins [5]. An early variant of the level-based method provided the first upper bound of $\mathcal{O}(n\lambda \log \lambda)$ on the runtime of UMDA with realistic population size $\lambda = \Omega(\log n)$ on ONEMAX [13]. Recently, a more precise application of the level-based method tightened this bound to $\mathcal{O}(n\lambda)$ under the assumption that the parent population size is $\mu = \mathcal{O}(\sqrt{n})$ [37]. For offspring population sizes $\lambda = \mathcal{O}(\log n)$, this runtime bound is tight, because it matches the lower bound of $\Omega(\mu n + n \log n)$ shown via drift analysis [33]. A different argument than the level-based method yielded a similar upper bound [57] for the UMDA.

We show that all nonelitist EAs with or without crossover, and even EDAs, can be cast and analyzed in the same framework. An early version of this paper was published in [9]. This followed from work dating back to the introduction of a fitness level technique for *nonelitist* EAs with linear ranking selection [40], later on generalized to many selection mechanisms and *unary variation operators* [36], with a refined result in [14]. The original fitness level technique and its generalization to the level-based technique have already found several applications, including analysis of EAs in uncertain environments, such as partial information [14], noisy [12], and dynamic fitness functions [11]. It has also been used to analyze self-adaptive EAs [15], pointing out multimodal fitness landscapes where they outperform classical, elitist EAs.

This paper improves the main result of [9] in many aspects. A more careful analysis of the population dynamics leads to a much tighter expression of the runtime bound compared to [9], and close to optimal for the class of search processes the theorem applies to. This immediately implies improved results in the previously mentioned applications. In particular, the leading term in the runtime is improved by a factor of $\Omega(\delta^{-3})$, where δ characterizes how fast good individuals can populate the population. This significantly improves the results of [12] and [14] concerning noisy optimization, for which δ is often very small (e.g., $1/n$). We also recommend a stepwise guideline for how to apply the theorem to new settings. Example applications are given for the cases of GAs and UMDA in optimizing standard pseudo-Boolean functions, a simple combinatorial problem, and in searching for local optima of NP-hard problems.

In this paper, Section II describes the class of algorithms covered by the level-based theorem, showing that many GAs and EDAs are special cases. The section then states the main theorem and corollaries for special cases, followed by their proofs. Sections IV and V apply the level-based theorem to the simple genetic algorithm (SGA) and UMDA on example problems. Section VI proves that the level-based theorem is tight for the class of algorithms considered. Section VII concludes this paper. Some proofs have been omitted due to space restrictions, but are in the supplementary materials.

Algorithm 1 Population-Based Algorithm

Require: A finite state space $\mathcal{X}$, and population size $\lambda \in \mathbb{N}$, a mapping D from $\mathcal{X}^\lambda$ to the space of prob. dist. over $\mathcal{X}$, and an initial population $P_0 \in \mathcal{X}^\lambda$.

1: **for** $t = 0, 1, 2, \ldots$ until termination condition met **do**
2: Sample $P_{t+1}(i) \sim D(P_t)$ independently for all $i \in [\lambda]$.

II. MAIN RESULT

A. Abstract Algorithmic Scheme

We consider population-based processes as stochastic processes $(P_t)_{t\in\mathbb{N}}$, where for each "generation" $t \in \mathbb{N}$, $P_t = (x_1, \ldots, x_\lambda) \in \mathcal{X}^\lambda$ is a vector of λ *individuals*, and where the set $\mathcal{X}$ represents the "search space" or "genospace." Our goal is to estimate the expected number of generations until the population contains at least one element in some given subset of $\mathcal{X}$. Our main assumption is that for every generation $t \in \mathbb{N}$, each individual in generation P_{t+1} is obtained by independent sampling from a distribution $D(P_t)$. Intuitively, D describes the randomized process determining how new individuals are produced, and may include fitness evaluations, selection, variation operators, external noise etc. Formally, D is a mapping from the set of all possible populations $\mathcal{X}^\lambda$ into the space of probability distributions over $\mathcal{X}$. This scheme is summarized in Algorithm 1.

Algorithm 1 covers many nonelitist search heuristics, such as stochastic beam search [53], EDA, and GAs [26]. For example, the GA given by Algorithm 2 is a special case of Algorithm 1, where the operator D corresponds to lines 3–5 in Algorithm 2. Here, the random operator `select`: $\mathcal{X}^\lambda \to [\lambda]$ represents a selection mechanism, which given a vector of λ individuals, returns the index of the individual to be selected. The selection mechanism is typically defined relative to a fitness function $f\colon \mathcal{X} \to \mathbb{R}$. The GA uses the two variation operators `mutate`: $\mathcal{X} \to \mathcal{X}$, and `crossover`: $\mathcal{X}\times\mathcal{X} \to \mathcal{X}$.

B. Level-Based Theorem

We now state the main result of this paper: a general technique for obtaining upper bounds on the expected runtime of

Algorithm 2 GA

Require: A finite set $\mathcal{X}$, population size $\lambda \in \mathbb{N}$, recombination rate $p_c \in (0, 1]$, and initial pop. $P_0 \in \text{Unif}(\mathcal{X}^\lambda)$.
1: **for** $t = 0, 1, 2, \ldots$ until termination condition met **do**
2: **for** $i = 1$ to λ **do**
3: $u \coloneqq P_t(\texttt{select}(P_t))$, $v \coloneqq P_t(\texttt{select}(P_t))$.
4: With prob. p_c, $x \coloneqq \texttt{crossover}(u, v)$ **else** $x \coloneqq u$.
5: $P_{t+1}(i) \coloneqq \texttt{mutate}(x)$.

Algorithm 3 Example Algorithm to Illustrate Theorem 1

1: Sample an initial population $P_0 \sim \text{Unif}([m]^\lambda)$ u.a.r.
2: **for** $t = 0, 1, 2, \ldots$ until termination condition met **do**
3: Sort $P_t = (x_1, \ldots, x_\lambda)$ s.t. $x_1 \geq x_2 \geq \cdots \geq x_\lambda$.
4: **for** $i = 1$ to λ **do**
5: $z \coloneqq x_k$, where $k \sim \text{Unif}([\lambda/2])$.
6: $y \coloneqq z + \text{Unif}(\{-c, 0, 1\})$ for any fixed $c \in [m]$
7: $P_{t+1}(i) \coloneqq \max\{1, \min\{y, m\}\}$.

any process that can be described in the form of Algorithm 1. We use the following notation. The natural logarithm is denoted by $\ln(\cdot)$, and $[n] = \{1, \ldots, n\}$ denotes the first n natural numbers. Suppose that for some m there is an ordered partition of $\mathcal{X}$ into subsets $(A_1, \ldots, A_m)$ which we call *levels*. We define $A_{\geq j} \coloneqq \bigcup_{i=j}^{m} A_i$, i.e., the union of all levels above level j. The *canonical* partition of $\mathcal{X}$ with respect to a fitness function $f\colon \mathcal{X} \to \mathbb{R}$, is $x, y \in A_j$ if and only if $f(x) = f(y)$ (see [36]). The partition is called f-based if $f(x) < f(y)$ for all $x \in A_j$, $y \in A_{j+1}$ and all $j \in [m-1]$. As a result of the algorithmic abstraction, our main theorem is not limited to this particular type of partition. Let $P \in \mathcal{X}^\lambda$ be a population vector of a finite number $\lambda \in \mathbb{N}$ of individuals. Given any subset $A \subseteq \mathcal{X}$, we define $|P \cap A| \coloneqq |\{i \mid P(i) \in A\}|$, i.e., the number of individuals in P that belong to A.

Theorem 1: Given a partition $(A_1, \ldots, A_m)$ of $\mathcal{X}$, define $T \coloneqq \min\{t\lambda \mid |P_t \cap A_m| > 0\}$, where for all $t \in \mathbb{N}$, $P_t \in \mathcal{X}^\lambda$ is the population of Algorithm 1 in generation t. If there exist $z_1, \ldots, z_{m-1}, \delta \in (0, 1]$, and $\gamma_0 \in (0, 1)$ such that for any population $P \in \mathcal{X}^\lambda$

(G1): for each level $j \in [m-1]$, if $|P \cap A_{\geq j}| \geq \gamma_0\lambda$; then

$$\Pr_{y \sim D(P)}\left(y \in A_{\geq j+1}\right) \geq z_j$$

(G2): for each level $j \in [m-2]$, and all $\gamma \in (0, \gamma_0]$ if $|P \cap A_{\geq j}| \geq \gamma_0\lambda$ and $|P \cap A_{\geq j+1}| \geq \gamma\lambda$, then

$$\Pr_{y \sim D(P)}\left(y \in A_{\geq j+1}\right) \geq (1+\delta)\gamma$$

(G3): and the population size $\lambda \in \mathbb{N}$ satisfies

$$\lambda \geq \left(\frac{4}{\gamma_0\delta^2}\right)\ln\left(\frac{128m}{z_*\delta^2}\right), \text{ where } z_* \coloneqq \min_{j \in [m-1]}\{z_j\}$$

then

$$E[T] \leq \left(\frac{8}{\delta^2}\right)\sum_{j=1}^{m-1}\left(\lambda \ln\left(\frac{6\delta\lambda}{4 + z_j\delta\lambda}\right) + \frac{1}{z_j}\right).$$

The level-based theorem gives an upper bound on the expected time until the algorithm discovers an element in the last level A_m, given that certain conditions on the operator D and population size λ are satisfied. Time is measured by the random variable T, which is defined as the number of individuals sampled in line 2 of Algorithm 1 until the end of the first generation having an individual in level A_m. This is an upper bound on the total number of sampled search points until a search point in A_m is conceived for the first time.

Informally, the two first conditions require a relationship between the current population P and the distribution $D(P)$ of the individuals in the next generation. Condition (G1) demands that the probability of creating an individual at level $j+1$ or higher is at least z_j when some fixed portion γ_0 of the population has reached level j or higher. Furthermore, if the number of individuals at level $j+1$ or higher is at least $\gamma\lambda > 0$, condition (G2) requires that their number tends to increase further, by a multiplicative factor of $1 + \delta$. Finally, (G3) requires a sufficiently large population. When all conditions are satisfied, an upper bound on the expected time for the algorithm to create an individual in A_m is guaranteed.

We recommend these steps when applying the theorem.

1) Identify a partition of $\mathcal{X}$ reflecting the "typical" progress of the population to the target set A_m.
2) Find parameter settings of the algorithm and corresponding parameters γ_0 and δ which allow condition (G2) to be satisfied, possibly by adjusting the partition of $\mathcal{X}$.
3) For each level $j \in [m-1]$, estimate lower bounds z_j such that condition (G1) holds.
4) Determine the lower bound on the population size λ in (G3), using the parameters obtained in the previous steps.
5) Compute the upper bound on $E[T]$. The terms $\ln([(6\delta\lambda)/(4 + z_j\delta\lambda)])$ can be bounded by underestimating the denominator, either by 4, or by $z_j\delta\lambda$, leading to the upper bounds $\ln(3\lambda/2)$, respectively, $\ln(6/z_j)$.

We now illustrate this methodology on a simple example.

Corollary 1: Algorithm 3 with $\lambda \geq 72(\ln(m) + 9)$ produces less than $216(m-1)(\lambda+1)$ individuals in expectation before it discovers m.

To illustrate Theorem 1, we estimate the time until the element m is contained in the population of Algorithm 3. The search space $\mathcal{X}$ is the set of natural numbers between 1 and m. Following the scheme of Algorithm 1, the operator D corresponds to lines 3–7. The new individual y is obtained by first selecting uniformly at random one of the best $\lambda/2$ individuals in the population (lines 3 and 5) and mutating this individual by adding 1 subtracting c or doing nothing, with equal probabilities. We will see that the value of c does not matter in our analysis.

We now carry out the steps described previously.

Step 1: We use the partition $A_j \coloneqq \{j\}$ for all $j \in [m]$.

Step 2: Assume that the *current level* is $j < m-1$. This means that in P_t, there are $\gamma_0\lambda$ individuals in $A_{\geq j}$, i.e., with fitness at least j, and at least $\gamma\lambda$ but less than $\gamma_0\lambda$ individuals in $A_{\geq j+1}$, i.e., with fitness at least $j+1$. We need to estimate $\Pr_{y \sim D(P_t)}(y \in A_{\geq j+1})$, i.e., the probability of producing an individual with fitness at least $j+1$. We say that a selection event is "good" if in line 5, the algorithm selects an individual in $A_{\geq j+1}$, i.e., with fitness at least $j+1$. If $\gamma \leq 1/2$, then the

probability of a good selection event is at least $\gamma\lambda/(\lambda/2) = 2\gamma$. We say that a mutation event is good if in line 6, the algorithm does not subtract c from the selected search point. The probability of a good mutation event is 2/3. Selection and mutation are independent events, hence we have shown for all $\gamma \in (0, 1/2]$ that $\Pr_{y\sim D(P_t)}(y \in A_{\geq j+1}) \geq (2\gamma)(2/3) = \gamma(1+(1/3))$. Condition (G2) is therefore satisfied with $\delta = 1/3$ for any $\gamma_0 \leq 1/2$. We will choose the parameter γ_0 later.

Step 3: Assume that population P_t has at least $\gamma_0\lambda$ individuals in $A_{\geq j}$. The algorithm can then produce an individual in $A_{\geq j+1}$ by selecting an individual in $A_{\geq j}$, and mutate this individual into $A_{\geq j+1}$ by adding 1 in line 6. We can conveniently fix $\gamma_0 := 1/2$, so that the probability of selecting an individual in $A_{\geq j}$ becomes 1. Furthermore, the probability of adding 1 to the selected individual is exactly 1/3. Hence, we have $\Pr_{y\sim D(P_t)}(y \in A_{\geq j+1}) \geq 1(1/3)$, and we can satisfy condition (G1) by defining $z_j := 1/3$ for all $j \in [m-1]$.

Step 4: For the parameters we have chosen, it is easy to see by numerical calculation that the population size $\lambda \geq 72(\ln(m) + 9)$ satisfies condition (G3).

Step 5: Using that $\ln([(6\delta\lambda)/(4+\delta\lambda z_j)]) < \ln(6/z_j)$, the expected time to discover the point m is no more than

$$\frac{8}{(1/3)^2}\sum_{j=1}^{m-1}\left(\lambda\ln\left(\frac{6}{1/3}\right)+\frac{1}{1/3}\right) < 216(m-1)(\lambda+1).$$

C. Proof of the Level-Based Theorem

Theorem 1 will be proved using drift analysis [28], [29], a standard tool in theory of randomized search heuristics. We a variant of the additive drift theorem [29], which is proved in the supplementary materials. The lower bound statement will be used in Section VI to evaluate the tightness of the theorem. In what follows, "$(Z_{t+1} - Z_t + \varepsilon)\ ; t < T_a$" denotes "$(Z_{t+1} - Z_t + \varepsilon)\cdot \mathbb{1}_{\{t<T_a\}}$" (see [55, 6.3]).

Theorem 2 (Additive Drift Theorem): Let $(Z_t)_{t\in\mathbb{N}}$ be a discrete-time stochastic process in $[0,\infty)$ adapted to any filtration $(\mathscr{F}_t)_{t\in\mathbb{N}}$. Define $T_a := \min\{t \in \mathbb{N} \mid Z_t \leq a\}$ for any $a \geq 0$. For some $\varepsilon > 0$ and constant $0 < b < \infty$, define the conditions

1.1) $E\left[Z_{t+1} - Z_t + \varepsilon\ ; t < T_a \mid \mathscr{F}_t\right] \leq 0$ for all $t \in \mathbb{N}$;
1.2) $E\left[Z_{t+1} - Z_t + \varepsilon\ ; t < T_a \mid \mathscr{F}_t\right] \geq 0$ for all $t \in \mathbb{N}$;
2) $Z_t < b$ for all $t \in \mathbb{N}$;
3) $E[T_a] < \infty$.

If 1.1), 2), and 3) hold, then $E[T_a \mid \mathscr{F}_0] \leq Z_0/\varepsilon$.
If 1.2), 2), and 3) hold, then $E[T_a \mid \mathscr{F}_0] \geq (Z_0 - a)/\varepsilon$.

In the proofs of Theorems 1 and 8, it will be important that conditions 1.1) and 1.2) apply to the whole past-present (see [28]), unlike the usual additive drift statement [29] that is conditioned on the present state only.

When applying the additive drift theorem to a complex process, Z_t is the result of a (measurable) mapping of the states of the process to a real number. Such a mapping is called the *distance function*, which measures the distance to some target state. Our distance function takes into account both the current level of the population, as well as the distribution of the population around the current level. In particular, let the current level Y_t be the highest level $j \in [m]$ such that there are at least $\gamma_0\lambda$ individuals at level j or higher. Furthermore, for any level $j \in [m]$, let $X_t^{(j)}$ be the number of individuals at level j or higher. Hence, we describe the dynamics of the population by $m+1$ stochastic processes $X_t^{(1)}, \ldots, X_t^{(m)}, Y_t$. Assuming that these processes are adapted to a filtration $\mathscr{F}_t$, we write $E_t[X] := E[X \mid \mathscr{F}_t]$ and $\Pr_t(\mathcal{E}) := E[\mathbb{1}_{\mathcal{E}} \mid \mathscr{F}_t]$. Our approach is to measure the distance of the population at time t by a scalar $g(X_t^{(Y_t+1)}, Y_t)$, where g is a function that satisfies the conditions in Definition 1.

Definition 1: A function $g : (\{0\}\cup[\lambda])\times[m] \to \mathbb{R}$ is called a level function if the following three conditions hold:

1) $\forall x \in \{0\}\cup[\lambda], \forall y \in [m-1]: \quad g(x,y) \geq g(x,y+1)$;
2) $\forall x \in \{0\}\cup[\lambda-1], \forall y \in [m]: \quad g(x,y) \geq g(x+1,y)$;
3) $\forall y \in [m-1]: \quad g(\lambda,y) \geq g(0,y+1)$.

Note that the sum of two level functions is also a level function. Furthermore, the conditions ensure that the distance $g(X_t^{(Y_t+1)}, Y_t)$ of the population decreases monotonically with the current level Y_t. Lemma 1 shows that this monotonicity allows an upper bound on the distance in the next generation which is partly independent of the change in current level.

Lemma 1: If $Y_{t+1} \geq Y_t$, then for any level function g

$$g\left(X_{t+1}^{(Y_{t+1}+1)}, Y_{t+1}\right) \leq g\left(X_{t+1}^{(Y_t+1)}, Y_t\right).$$

Proof: The statement is trivial when $Y_t = Y_{t+1}$. On the other hand, if $Y_{t+1} > Y_t$, then the conditions in Definition 1 imply

$$\begin{aligned} g\left(X_{t+1}^{(Y_{t+1}+1)}, Y_{t+1}\right) &\leq g(0, Y_{t+1}) \leq g(0, Y_t+1) \\ &\leq g(\lambda, Y_t) \leq g\left(X_{t+1}^{(Y_t+1)}, Y_t\right). \end{aligned}$$

■

Proof of Theorem 1: We apply Theorem 2 with respect to the parameter $a = 0$ and the process $Z_t := g(X_t^{(Y_t+1)}, Y_t)$, where g is a level-function, and $(Y_t)_{t\in\mathbb{N}}$ and $(X_t^{(j)})_{t\in\mathbb{N}}$ for $j \in [m]$ are stochastic processes, which will be defined later. $(\mathscr{F}_t)_{t\in\mathbb{N}}$ is the filtration induced by the populations $(P_t)_{t\in\mathbb{N}}$.

We will assume w.l.o.g. that condition (G2) is also satisfied for $j = m-1$, for the following reason. Given Algorithm 1 with a certain mapping D, consider Algorithm 1 with a different mapping $D'(P)$: if $|P \cap A_m| = 0$, then $D'(P) = D(P)$; otherwise $D'(P)$ assigns probability mass 1 to some element x of P that is in A_m, e.g., to the first one among such elements. Note that D' meets conditions (G1) and (G2). Moreover, (G2) holds for $j = m-1$. For the sequence of populations $P'_0, P'_1, \ldots$ of Algorithm 1 with mapping D', we can put $T' := \lambda \cdot \min\{t \mid |P'_t \cap A_m| > 0\}$. Executions of the original algorithm and the modified one before generation T'/λ are identical. On generation T'/λ both algorithms place elements of A_m into the population for the first time. Thus, T' and T are equal in every realization and their expectations are equal.

For any level $j \in [m]$ and time $t \geq 0$, let the random variable $X_t^{(j)} := |P_t \cap A_{\geq j}|$ denote the number of individuals in levels $A_{\geq j}$ at time t. Because $A_{\geq j}$ is partitioned into disjoint sets A_j and $A_{\geq j+1}$, the definition implies

$$\left|P_t \cap A_j\right| = X_t^{(j)} - X_t^{(j+1)}. \tag{1}$$

Algorithm 1 samples all individuals in generation $t+1$ independently from distribution $D(P_t)$. Therefore, given the current population P_t, $X_{t+1}^{(j)}$ is binomially distributed $X_{t+1}^{(j)} \sim \text{Bin}(\lambda, p_{t+1}^{(j)})$, where $p_{t+1}^{(j)} := \Pr_{t,y\sim D(P_t)}(y \in A_{\geq j})$ is the probability of sampling an individual in level j or higher.

The *current level* Y_t of the population at time t is defined as $Y_t := \max\{j \in [m] \mid X_t^{(j)} \geq \gamma_0\lambda\}$. Note that $(X_t^{(j)})_{t\in\mathbb{N}}$ and $(Y_t)_{t\in\mathbb{N}}$ are adapted to the filtration $(\mathscr{F}_t)_{t\in\mathbb{N}}$ because they are defined in terms of the population process $(P_t)_{t\in\mathbb{N}}$.

When $Y_t < m$, there exists a unique $\gamma < \gamma_0$ such that

$$X_t^{(Y_t+1)} = \left|P_t \cap A_{\geq Y_t+1}\right| = \gamma\lambda \tag{2}$$

$$X_t^{(Y_t)} = \left|P_t \cap A_{\geq Y_t}\right| \geq \gamma_0\lambda \text{ and} \tag{3}$$

$$X_t^{(Y_t-1)} = \left|P_t \cap A_{\geq Y_t-1}\right| \geq \gamma_0\lambda. \tag{4}$$

In the case of $X_t^{(Y_t+1)} = 0$, it follows from (1)–(3) that $|P \cap A_j| = X_t^{(Y_t)} \geq \gamma_0\lambda$. Condition (G1) for level $j = Y_t$ then gives

$$p_{t+1}^{(Y_t+1)} = \Pr_{y\sim D(P_t)}\left(y \in A_{\geq Y_t+1}\right) \geq z_{Y_t}. \tag{5}$$

Otherwise if $X_t^{(Y_t+1)} \geq 1$, conditions (G1) and (G2) for level $j = Y_t$ with (2) and (3) imply

$$p_{t+1}^{(Y_t+1)} = \Pr_{y\sim D(P_t)}(y \in A_{\geq Y_t+1}) \tag{6}$$

$$\geq \max\left\{(1+\delta)\frac{X_t^{(Y_t+1)}}{\lambda}, z_j\right\}. \tag{7}$$

Condition (G2) for level $j = Y_t - 1$ with (3) and (4) give

$$p_{t+1}^{(Y_t)} = \Pr_{y\sim D(P_t)}\left(y \in A_{\geq Y_t}\right) \geq (1+\delta)\gamma_0. \tag{8}$$

We now define the process $(Z_t)_{t\in\mathbb{N}}$ as $Z_t := 0$ if $Y_t = m$, and otherwise, if $Y_t < m$, we let $Z_t := g(X_t^{(Y_t+1)}, Y_t)$, where for all k, and for all $1 \leq j < m$, $g(k,j) = g_1(k,j) + g_2(k,j)$ and

$$g_1(k,j) := \ln\left(\frac{1+(\delta/2)\lambda}{1+(\delta/2)\max\{k, z_j\lambda/(1+\delta)\}}\right) + \sum_{i=j+1}^{m-1}\ln\left(\frac{1+(\delta/2)\lambda}{1+(\delta/2)\lambda z_i/(1+\delta)}\right)$$

$$g_2(k,j) := \frac{1}{q_j}\left(1-\frac{\delta^2}{7}\right)^k + \sum_{i=j+1}^{m-1}\frac{1}{q_i}$$

where $q_j := 1-(1-z_j)^\lambda$.

It follows from Lemma 5 that $g(k,j)$ is a level function. Furthermore, $g(k,j) \geq 0$ for all $k \in \{0\} \cup [\lambda]$ and all $j \in [m]$. Due to properties 1 and 2 of level functions, and [14, Lemma 31], the distance is always bounded from above by

$$g(0,1) \leq \sum_{i=1}^{m-1}\left(\ln\left(\frac{1+(\delta/2)\lambda}{1+(\delta/2)z_i\lambda/(1+\delta)}\right)+\frac{1}{q_i}\right) < \sum_{i=1}^{m-1}\left(\ln\left(\frac{4+2\delta\lambda}{4+\delta z_i\lambda}\right)+1+\frac{1}{\lambda z_i}\right). \tag{9}$$

Using that $z_i \leq 1$, this can be bounded further by

$$< \sum_{i=1}^{m-1}\left(\ln\left(\frac{4+2\delta\lambda}{z_i(4+\delta\lambda)}\right)+1+\frac{1}{\lambda z_i}\right) = \sum_{i=1}^{m-1}\left(\ln\left(\frac{1}{z_i}\left(1+\frac{\delta\lambda}{4+\delta\lambda}\right)\right)+1+\frac{1}{\lambda z_i}\right). \tag{10}$$

We then exploit that $\ln(x) \leq x-1$ for all $x > 0$ so

$$< \sum_{i=1}^{m-1}\left(\frac{2}{z_i}+\frac{1}{\lambda z_i}\right).$$

Finally, since $z_i \geq z_*$ and $\lambda \geq \lceil 4\ln(128)\rceil = 20$ by (G3)

$$< \frac{m}{z_*}\left(2+\frac{1}{\lambda}\right) \leq \frac{41m}{20z_*}. \tag{11}$$

Hence, we have $0 \leq Z_t < g(0,1) < \infty$ for all $t \in \mathbb{N}$ which implies that condition 2 of the drift theorem is satisfied.

The drift of the process at time t is $E_t\left[\Delta_{t+1}\right]$, where

$$\Delta_{t+1} := g\left(X_t^{(Y_t+1)}, Y_t\right) - g\left(X_{t+1}^{(Y_{t+1}+1)}, Y_{t+1}\right).$$

We bound the drift by the law of total probability as

$$E_t\left[\Delta_{t+1}\right] = (1-\Pr_t(Y_{t+1} < Y_t))E_t\left[\Delta_{t+1} \mid Y_{t+1} \geq Y_t\right] + \Pr_t(Y_{t+1} < Y_t)E_t\left[\Delta_{t+1} \mid Y_{t+1} < Y_t\right]. \tag{12}$$

The event $Y_{t+1} < Y_t$ holds if and only if $X_{t+1}^{(Y_t)} < \gamma_0\lambda$. Due to (8), we obtain the following by a Chernoff bound

$$\Pr_t(Y_{t+1} < Y_t) = \Pr_t\left(X_{t+1}^{(Y_t)} < \left(1-\frac{\delta}{1+\delta}\right)(1+\delta)\gamma_0\lambda\right) \leq \exp\left(-\frac{\delta^2\gamma_0\lambda}{2(1+\delta)}\right) \leq \frac{\delta^2 z_*}{128m} \tag{13}$$

where the last inequality takes into account the population size required by condition (G3). Given the low probability of the event $Y_{t+1} < Y_t$, it suffices to use the pessimistic bound (11)

$$E_t\left[\Delta_{t+1} \mid Y_{t+1} < Y_t\right] \geq -g(0,1) \geq -\frac{41m}{20z_*}. \tag{14}$$

If $Y_{t+1} \geq Y_t$, we can apply Lemma 1

$$E_t\left[\Delta_{t+1} \mid Y_{t+1} \geq Y_t\right] \geq E_t\left[g\left(X_t^{(Y_t+1)}, Y_t\right) - g\left(X_{t+1}^{(Y_t+1)}, Y_t\right) \mid Y_{t+1} \geq Y_t\right].$$

Note that event $Y_{t+1} \geq Y_t$ is equivalent to having $X_{t+1}^{(Y_t)} \geq \gamma_0\lambda$, then due to Lemma 7, in the following we can skip the condition on the event when needed.

If $X_t^{(Y_t+1)} = 0$, then $X_t^{(Y_t+1)} \leq X_{t+1}^{(Y_t+1)}$ and

$$E_t\left[g_1\left(X_t^{(Y_t+1)}, Y_t\right) - g_1\left(X_{t+1}^{(Y_t+1)}, Y_t\right) \mid Y_{t+1} \geq Y_t\right] \geq 0$$

because the function g_1 satisfies property 2 in Definition 1. Furthermore, we have the lower bound

$$E_t\left[g_2\left(X_t^{(Y_t+1)}, Y_t\right) - g_2\left(X_{t+1}^{(Y_t+1)}, Y_t\right) \mid Y_{t+1} \geq Y_t\right] > \Pr_t\left(X_{t+1}^{(Y_t+1)} \geq 1\right)(g_2(0,Y_t) - g_2(1,Y_t)) \geq \frac{\delta^2}{7}$$

where the last inequality follows because of (5) and $\Pr_t(X_{t+1}^{(Y_t+1)} \geq 1) \geq 1-(1-p_{t+1}^{(Y_t+1)})^\lambda \geq 1-(1-z_{Y_t})^\lambda = q_{Y_t}$, and $g_2(0, Y_t) - g_2(1, Y_t) = (1/q_{Y_t})(\delta^2/7)$.

In the other case, where $X_t^{(Y_t+1)} \geq 1$, we obtain

$$E_t\Big[g_1\Big(X_t^{(Y_t+1)}, Y_t\Big) - g_1\Big(X_{t+1}^{(Y_t+1)}, Y_t\Big) \mid Y_{t+1} \geq Y_t\Big] \geq E_t\left[\ln\left(\frac{1+\frac{\delta}{2}X_{t+1}^{(Y_t+1)}}{1+\frac{\delta}{2}\max\left\{X_t^{(Y_t+1)}, \frac{z_{Y_t}\lambda}{1+\delta}\right\}}\right)\right] \geq \frac{\delta^2}{7}$$

where the last inequality follows from Lemma 6 for the parameters $X \coloneqq X_{t+1}^{(Y_t+1)}$ and $p \coloneqq p_{t+1}^{(Y_t+1)}$ as given by (7). For the function g_2, we get

$$E_t\Big[g_2\Big(X_t^{(Y_t+1)}, Y_t\Big) - g_2\Big(X_{t+1}^{(Y_t+1)}, Y_t\Big) \mid Y_{t+1} \geq Y_t\Big] = \frac{1}{q_{Y_t}}\left(\left(1-\frac{\delta^2}{7}\right)^{X_t^{(Y_t)}} - E_t\left[\left(1-\frac{\delta^2}{7}\right)^{X_{t+1}^{(Y_t+1)}}\right]\right) > 0$$

where the last inequality is due to [14, Lemma 6] (see also the supplementary materials), applied to $X_{t+1}^{(Y_t+1)} \sim \mathrm{Bin}(\lambda, p_{t+1}^{(Y_t+1)})$ with $p_{t+1}^{(Y_t+1)} \geq (1+\delta)X_t^{(Y_t+1)}/\lambda$ [see (7)] and the parameter $\kappa = -\ln(1-\delta^2/7) < \delta$.

Taking into account all cases, we have

$$E_t\big[\Delta_{t+1} \mid Y_{t+1} \geq Y_t\big] \geq \frac{\delta^2}{7}. \tag{15}$$

We now have bounds for all the quantities in (12) with (13)–(15), and we get

$$\begin{aligned} E_t\big[\Delta_{t+1}\big] &= (1-\Pr_t(Y_{t+1} < Y_t))E_t\big[\Delta_{t+1} \mid Y_{t+1} \geq Y_t\big] \\ &\quad + \Pr_t(Y_{t+1} < Y_t)E_t\big[\Delta_{t+1} \mid Y_{t+1} < Y_t\big] \\ &\geq \left(1-\frac{\delta^2 z_*}{128m}\right)\frac{\delta^2}{7} - \left(\frac{\delta^2 z_*}{128m}\right)\left(\frac{41m}{20z_*}\right) > \frac{\delta^2}{8}. \end{aligned}$$

We now verify condition 3 of Theorem 2, i.e., that T has finite expectation. Let $p_* \coloneqq \min\{(1+\delta)/\lambda, z_*\}$, and note by conditions (G1) and (G2) that the current level increases by at least one with probability $\Pr_t(Y_{t+1} > Y_t) \geq (p_*)^{\gamma_0\lambda}$. Due to the definition of the modified process D', if $Y_t = m$, then $Y_{t+1} = m$. Hence, the probability of reaching $Y_t = m$ is lower bounded by the probability of the event that the current level increases in all of at most m consecutive generations, i.e., $\Pr_t(Y_{t+m} = m) \geq (p_*)^{\gamma_0\lambda m} > 0$. It follows that $E[T] < \infty$.

By Theorem 2 and the upper bound on $g(0,1)$ in (9)

$$E[T] \leq \lambda \cdot \frac{g(0,1)}{\delta^2/8} < \left(\frac{8\lambda}{\delta^2}\right)\sum_{i=1}^{m-1}\ln\left(\frac{4+2\delta\lambda}{4+z_i\delta\lambda}\right) + 1 + \frac{1}{\lambda z_i}$$

then using that $4 \leq \delta\lambda/5$ from (G3) and $(1/5+2)e < 6$

$$< \left(\frac{8\lambda}{\delta^2}\right)\sum_{i=1}^{m-1}\left(\ln\left(\frac{6\delta\lambda}{4+z_i\delta\lambda}\right) + \frac{1}{\lambda z_i}\right). \qquad \blacksquare$$

It can be seen from the proof of Theorem 1 that it easily extends to algorithms where the mapping D is time-dependent, provided that (G1), (G2), and (G3) hold for any t for some fixed (time independent) values $z_1, \ldots, z_{m-1}$, δ, and γ_0.

III. Tools for Analyzing Genetic Algorithms

We now derive two corollaries of Theorem 1 tailored to Algorithm 2 and give conditions on tunable parameters of selection mechanisms making the corollaries applicable.

A fitness function is not defined explicitly in Algorithm 2, so no assumptions on an f-based partition will be needed in the corollaries. Here, we generalize the *cumulative selection probability* of `select`, denoted $\beta(\gamma, P)$, which was defined relative to the fitness function f in [14], to the one that is relative to the partition $(A_1, \ldots, A_m)$. To define $\beta(\gamma, P)$ of `select` with respect to f for a population P of λ search points, we first assume $(f_1, \ldots, f_\lambda)$ to be the vector of sorted fitness values of P, i.e., $f_i \geq f_{i+1}$ for each $i \in [\lambda-1]$. Then

$$\beta(\gamma, P) \coloneqq \sum_{i=1}^{\lambda} p_{\mathrm{sel}}(i \mid P) \cdot \big[f(P(i)) \geq f_{\lceil\gamma\lambda\rceil}\big]$$

for any $\gamma \in (0, 1]$. Here and below, $[\cdot]$ is the Iverson bracket.

Similarly, given a partition $(A_1, \ldots, A_m)$, if we use $(\ell_1, \ldots, \ell_\lambda)$ to denote the sorted levels of search points in P, i.e., $\ell_i \geq \ell_{i+1}$ for each $i \in [\lambda-1]$, then the *cumulative selection probability* of `select` with respect to $(A_1, \ldots, A_m)$ is

$$\zeta(\gamma, P) \coloneqq \sum_{i=1}^{\lambda} p_{\mathrm{sel}}(i \mid P) \cdot \big[P(i) \in A_{\geq \ell_{\lceil\gamma\lambda\rceil}}\big].$$

These definitions are related by the following lemma.

Lemma 2: For any f-based partition of $\mathcal{X}$ and $\lambda \in \mathbb{N}$

$$\forall P \in \mathcal{X}^\lambda, \forall \gamma \in (0, 1] \quad \zeta(\gamma, P) \geq \beta(\gamma, P). \tag{16}$$

A. Analysis of Nonpermanent Use of Crossover

We first derive from Theorem 1 a corollary that is adapted to Algorithm 2 with $p_c < 1$. This setting covers the case $p_c = 0$, i.e., only unary variation operators are used. This specific case is the main subject of [14]. As we will see later on, stronger and more general results can be claimed with the corollary.

Corollary 2: Given a partition $(A_1, \ldots, A_m)$ of $\mathcal{X}$, define $T \coloneqq \min\{t\lambda \mid |P \cap A_m| > 0\}$ where for all $t \in \mathbb{N}$, $P_t \in \mathcal{X}^\lambda$ is the population of Algorithm 2 in generation t. If $p_c < 1$ and there exist $s_1, \ldots, s_{m-1}, s_*, p_0, \delta \in (0, 1]$, and a constant $\gamma_0 \in (0, 1)$ such that for any population $P \in \mathcal{X}^\lambda$

(M1): for each level $j \in [m-1]$

$$p_{\mathrm{mut}}\big(y \in A_{\geq j+1} \mid x \in A_j\big) \geq s_j$$

(M2): for each level $j \in [m-1]$

$$p_{\mathrm{mut}}\big(y \in A_{\geq j} \mid x \in A_j\big) \geq p_0$$

(M3): for any population $P \in (\mathcal{X} \setminus A_m)^\lambda$ and $\gamma \in (0, \gamma_0]$

$$\zeta(\gamma, P) \geq \frac{(1+\delta)\gamma}{p_0(1-p_c)}$$

(M4): the population size λ satisfies

$$\lambda \geq \left(\frac{4}{\gamma_0\delta^2}\right)\ln\left(\frac{128m}{\gamma_0 s_*\delta^2}\right) \text{ where } s_* \coloneqq \min_{j\in[m-1]}\{s_j\}$$

then

$$E[T] < \left(\frac{8}{\delta^2}\right)\sum_{j=1}^{m-1}\left(\lambda\ln\left(\frac{6\delta\lambda}{4+\gamma_0 s_j\delta\lambda}\right) + \frac{1}{\gamma_0 s_j}\right).$$

Proof: We apply Theorem 1 following the guidelines.

Step 1: It is skipped because we already have the partition.

Step 2: Assume that $|P \cap A_{\geq j}| \geq \gamma_0\lambda$ and $|P \cap A_{\geq j+1}| \geq \gamma\lambda > 0$ for some $\gamma \leq \gamma_0$. To create an individual in $A_{\geq j+1}$, it suffices to pick an $x \in |P \cap A_k|$ for any $k \geq j+1$ and mutate it to an individual in $A_{\geq k}$, the probability of such an event, according to (M2) and (M3), is at least $(1-p_c)\zeta(\gamma, P)p_0 \geq (1+\delta)\gamma$. So (G2) holds with p_0, δ and γ_0 from (M3).

Step 3: We are given $|P \cap A_j| \geq \gamma_0\lambda$. Thus, with probability $\zeta(\gamma_0, P)$, the selection mechanism chooses an individual x in either A_j or $A_{\geq j+1}$. If $x \in A_j$, then the mutation operator will by (M1) upgrade x to $A_{\geq j+1}$ with probability s_j. If $x \in A_{\geq j+1}$, then by (M2), the mutation operator leaves the individual in $A_{\geq j+1}$ with probability p_0. Finally, no crossover occurs with probability $1 - p_c$, so the probability of producing an individual in $A_{\geq j+1}$ is at least $(1-p_c)\zeta(\gamma_0, P)\min\{s_j, p_0\} \geq (1-p_c)\zeta(\gamma_0, P)s_j p_0 > \gamma_0 s_j$ and (G1) holds with $z_j = \gamma_0 s_j$, $z_* = \gamma_0 s_*$.

Step 4: Given $z_* = \gamma_0 s_*$, condition (M4) yields (G3).

Step 5: Conditions (G1)–(G3) are satisfied and Theorem 1 gives

$$E[T] \leq \frac{8\lambda}{\delta^2}\sum_{j=1}^{m-1}\left(\ln\left(\frac{6\delta\lambda}{4+z_j\delta\lambda}\right)+\frac{1}{z_j\lambda}\right) = \frac{8}{\delta^2}\sum_{j=1}^{m-1}\left(\lambda\ln\left(\frac{6\delta\lambda}{4+\gamma_0 s_j\delta\lambda}\right)+\frac{1}{\gamma_0 s_j}\right). \quad \blacksquare$$

The proof implies that any operator can stand for crossover in line 4 of Algorithm 2, and the result will still hold.

B. Analysis of Permanent Use of Crossover

We now adapt Theorem 1 to Algorithm 2 with $p_c = 1$.

Corollary 3: Given a partition $(A_1, \ldots, A_m)$ of $\mathcal{X}$, define $T := \min\{t\lambda \mid |P_t \cap A_m| > 0\}$, where for all $t \in \mathbb{N}$, $P_t \in \mathcal{X}^\lambda$ is the population of Algorithm 2. If $p_c = 1$ and there exist $s_1, \ldots, s_{m-1}, s_*, p_0, \varepsilon, \delta \in (0, 1]$, and a constant $\gamma_0 \in (0, 1)$ such that

(C1): for each level $j \in [m-1]$

$$p_{\text{mut}}\big(y \in A_{\geq j+1} \mid x \in A_j\big) \geq s_j$$

(C2): for each level $j \in [m]$

$$p_{\text{mut}}\big(y \in A_{\geq j} \mid x \in A_j\big) \geq p_0$$

(C3): for each level $j \in [m-2]$

$$p_{\text{xor}}\big(x \in A_{\geq j+1} \mid u \in A_{\geq j}, v \in A_{\geq j+1}\big) \geq \varepsilon$$

(C4): for any population $P \in (\mathcal{X} \setminus A_m)^\lambda$ and $\gamma \in (0, \gamma_0]$

$$\zeta(\gamma, P) \geq \gamma\sqrt{\frac{1+\delta}{p_0\gamma_0\varepsilon}}$$

(C5): the population size satisfies

$$\lambda \geq \left(\frac{4}{\gamma_0\delta^2}\right)\ln\left(\frac{128m}{\gamma_0\delta^2 s_*}\right), \text{where} s_* := \min_{j\in[m-1]}\{s_j\}$$

then

$$E[T] < \left(\frac{8}{\delta^2}\right)\sum_{j=1}^{m-1}\left(\lambda\ln\left(\frac{6\delta\lambda}{4+\gamma_0 s_j\delta\lambda}\right)+\frac{1}{\gamma_0 s_j}\right).$$

Proof: We apply Theorem 1 following the guidelines.

Step 1: It is skipped because the partition is already defined.

Step 2: We are given $|P \cap A_{\geq j}| \geq \gamma_0\lambda$ and $|P \cap A_{\geq j+1}| \geq \gamma\lambda > 0$. To create an individual in $A_{\geq j+1}$, it suffices to pick the individual u in $A_{\geq j}$ and v in $A_{\geq j+1}$, then to produce an individual in A_k for any $k \geq j+1$ by crossover and not destroy the produced individual by mutation. The probability of such an event according to (C2), (C3), and (C4) is bounded from below by $\zeta(\gamma_0, P)\zeta(\gamma, P)\varepsilon p_0 \geq (1+\delta)\gamma$. Condition (G2) is then satisfied with the same γ_0 and δ as in (C4).

Step 3: We assume $|P \cap A_j| \geq \gamma_0\lambda$. Note that condition (C3) written for level $j-1$ is $p_{\text{xor}}(x \in A_{\geq j} \mid u \in A_{\geq j-1}, v \in A_{\geq j}) \geq \varepsilon$, and because $A_{\geq j} \subset A_{\geq j-1}$ then $p_{\text{xor}}(x \in A_{\geq j} \mid u \in A_{\geq j}, v \in A_{\geq j}) \geq \varepsilon$. To create an individual in levels $A_{\geq j+1}$, it then suffices to pick both parents u and v from levels $A_{\geq j}$ in line 3, produce an intermediary offspring in A_k for any $k \geq j$ via crossover, and from this an individual in $A_{\geq j+1}$ via mutation. If $k = j$, we need to improve the produced individual by mutation, relying on (C1). Otherwise if $k > j$ it suffices not to destroy the produced individual by mutation, relying on (C2). It follows from (C4) that the probability of producing an individual in $A_{\geq j+1}$ is at least $\zeta(\gamma_0, P)^2\varepsilon\min\{s_j, p_0\} \geq \zeta(\gamma_0, P)^2\varepsilon s_j p_0 > \gamma_0 s_j$. Condition (G1) then holds for $z_j = \gamma_0 s_j$ and $z_* = \gamma_0 s_*$.

Step 4: Given that $z_* = \gamma_0 s_*$, (C5) implies (G3).

Step 5: Conditions (G1–3) are satisfied, so Theorem 1 gives

$$E[T] \leq \frac{8}{\delta^2}\sum_{j=1}^{m-1}\left(\lambda\ln\left(\frac{6\delta\lambda}{4+\gamma_0 s_j\delta\lambda}\right)+\frac{1}{\gamma_0 s_j}\right).$$

$\blacksquare$

Corollary 3 is similar to Corollary 2, except that condition (C2) has to additionally hold for level A_m, that (C3) is a new condition on the `crossover` operator, and that condition (C4) on the `select` operator differs from (M3).

C. Analysis of Selection Mechanisms

We show how to parameterize the following selection mechanisms such that condition (M3) of Corollary 2 and (C4) of Corollary 3 are satisfied. In *k-tournament selection*, k individuals are sampled uniformly at random with replacement from the population, and the fittest of these individuals is returned. In (μ, λ)-*selection*, parents are sampled uniformly at random among the fittest μ individuals in the population. A function $\alpha : \mathbb{R} \to \mathbb{R}$ is a ranking function [26] if $\alpha(x) \geq 0$ for all $x \in [0, 1]$, and $\int_0^1 \alpha(x)dx = 1$. In ranking selection with ranking function α, the probability of selecting an individual among $\gamma\lambda$ best individuals is $\int_0^\gamma \alpha(x)dx$.

In *linear ranking selection*, parameterized by $\eta \in (1, 2]$, the ranking function is $\alpha(\gamma) := \eta(1 - 2\gamma) + 2\gamma$. We define *exponential ranking selection* parameterized by $\eta > 0$ with $\alpha(\gamma) := \eta e^{\eta(1-\gamma)}/(e^\eta - 1)$.

Lemma 3: Assuming that $(A_1, \ldots, A_m)$ is a partition of $\mathcal{X}$ with $(A_1, \ldots, A_{m-1})$ being an f-based partition of $\mathcal{X} \setminus A_m$, for any constants $\delta' > 0$, $p_0 \in (0, 1)$, $\varepsilon \in (0, 1)$, and for any non-negative parameter $p_c = 1 - \Omega(1)$, there exists a constant $\gamma_0 \in (0, 1)$ such that all the following selection mechanisms:

1) k-tournament selection;
2) (μ, λ)-selection;
3) linear ranking selection;
4) exponential ranking selection

with their parameters k, λ/μ, and η being set to no less than

$$\frac{1 + \delta'}{(1 - p_c)p_0}$$

satisfy (M3), that is

$$\zeta(\gamma, P) \geq \frac{\left(1 + \delta''\right)\gamma}{p_0(1 - p_c)}$$

for any $\gamma \in (0, \gamma_0]$, any $P \in (\mathcal{X}\setminus A_m)^\lambda$, and some constant $\delta'' > 0$.

Lemma 4: Given a partition $(A_1, \ldots, A_m)$ of $\mathcal{X}$ with $(A_1, \ldots, A_{m-1})$ being an f-based partition of $\mathcal{X} \setminus A_m$, for any constants $\delta' > 0$, $p_0 \in (0, 1)$, and $\varepsilon \in (0, 1)$, there exists a constant $\gamma_0 \in (0, 1)$ such that the following selection mechanisms:

1) k-tournament selection with $k \geq 4(1 + \delta')/(\varepsilon p_0)$;
2) (μ, λ)-selection with $\lambda/\mu \geq (1 + \delta')/(\varepsilon p_0)$;
3) exponential ranking selection with $\eta \geq 4(1 + \delta')/(\varepsilon p_0)$

satisfy (C4), that is

$$\zeta(\gamma, P) \geq \gamma\sqrt{\frac{1 + \delta'}{p_0 \varepsilon \gamma_0}}$$

for any $\gamma \in (0, \gamma_0]$, and any $P \in (\mathcal{X}\setminus A_m)^\lambda$.

Lemmas 3 and 4 are proved in the supplementary materials.

IV. Applications to Genetic Algorithms

We now apply the results from Section III. Proofs have been moved to the supplementary materials due to space limitations. Given a bitstring x a *bitwse mutation operator*, returns a bitstring y, where for each $i \in [n]$, bit y_i, is set independently to $1 - x_i$ with probability p_m and is otherwise kept equal to x_i. The parameter $p_m \in [0, 1]$ is called the *mutation rate*.

A. Optimization of Pseudo-Boolean Functions

In this section, we consider the expected runtime of nonelitist GAs in Algorithm 2 on the following functions: $\textsc{OneMax}(x) := \sum_{i=1}^n x_i = |x|_1 = \textsc{Om}(x)$, $\textsc{LeadingOnes}(x) := \sum_{i=1}^n \prod_{j=1}^i x_i = \textsc{Lo}(x)$

$$\textsc{Jump}_r(x) := \begin{cases} n + 1 & \text{if } |x|_1 = n \\ r + |x|_1 & \text{if } |x|_1 \leq n - r \\ n - |x|_1 & \text{otherwise.} \end{cases}$$

$\textsc{RoyalRoad}_r(x) := \sum_{i=0}^{n/r-1} \prod_{j=1}^r x_{ir+j}$, $\textsc{Linear}(x) := \sum_{i=1}^n c_i x_i$, where each $c_i \in \mathbb{R}$. For Linear, w.l.o.g. we can assume $c_1 \geq \cdots \geq c_n > 0$ [14]. We also consider the class of ℓ-Unimodal functions, where each function has exactly ℓ distinctive fitness values $f_1 < \cdots < f_\ell$, and each bitstring x of the search space is either optimal or it has a Hamming-neighbor y with $f(y) > f(x)$.

Several results about the runtime of EAs with parent or offspring population size greater than one can be found in the literature. For the illustrative purpose, we cite just some of results. In [48], it is shown that $(1, \lambda)$ EA on OneMax has the runtime $\mathcal{O}(n \log n + n\lambda)$, provided that $\lambda \geq \log_{(e/e-1)} n$, and on the ℓ-Unimodal functions this algorithm has the runtime $O(\ell n + \ell\lambda)$, given that $\lambda \geq \log_{(e/e-1)}(\ell n)$. A nonelitist GA using bitwise mutation and tournament selection with $k = \Omega(\lambda)$ and $\lambda = \Omega(n \log n)$ has runtime $\mathcal{O}(n\lambda)$ on LeadingOnes [22]. The $(1 + \lambda)$ EA on any linear function has runtime $O(n \log n + n\lambda)$, see [18]. A $(\mu + 1)$ GA where the uniform crossover is applied with probability $p_c = \mathcal{O}(1/(nr))$ and μ is chosen appropriately is shown to have $\mathcal{O}\left(\mu n^2 r^3 + 4^r/p_c\right)$ runtime on $\textsc{Jump}_r$ function [31]. The case of $p_c = 1$ is treated in [10].

Our results presented below apply only to nonelitist GAs with bitwise mutation. For a moderate use of crossover, i.e., $p_c = 1 - \Omega(1)$, Corollary 2 and Lemma 3 yield the following.

Theorem 3: The expected runtime of the GA in Algorithm 2, with $p_c = 1 - \Omega(1)$, using any crossover operator, a bitwise mutation with mutation rate χ/n for any fixed constant $\chi > 0$ and one of the selection mechanisms: k-tournament selection, (μ, λ)-selection, linear or exponential ranking selection, with their parameters k, λ/μ, and η being set to no less than $(1 + \delta)e^\chi/(1 - p_c)$, where $\delta \in (0, 1]$ being any constant, is:

1) $\mathcal{O}(n\lambda)$ on OneMax if $\lambda \geq c \ln n$;
2) $\mathcal{O}\left(n^2 + n\lambda \ln \lambda\right)$ on LeadingOnes if $\lambda \geq c \ln n$;
3) $\mathcal{O}(n\ell + \ell\lambda \ln \lambda)$ on ℓ-Unimodal if $\lambda \geq c \ln(\ell n)$;
4) $\mathcal{O}\left(n^2 + n\lambda \ln \lambda\right)$ on Linear if $\lambda \geq c \ln n$;
5) $\mathcal{O}((n/\chi)^r + n\lambda + \lambda \ln \lambda)$ on $\textsc{Jump}_r$ if $\lambda \geq cr \ln n$;
6) $\mathcal{O}((n/\chi)^r \ln(n/r) + [(n\lambda \ln \lambda)/r])$ on $\textsc{RoyalRoad}_{r\geq 2}$ if $\lambda \geq cr \ln n$;

for some sufficiently large constant c.

The proof is in the supplementary materials.

In the case of regular use of crossover, i.e., $p_c = 1$, we limit our consideration to *mask-based* crossovers. Given two parent genotypes u and v, such operator consists in first choosing (deterministically or randomly) a binary string $\tilde{m} = (m_1, \ldots, m_n)$ to produce two offspring vectors x', x'' as

$$x'_i = \begin{cases} u_i, & \text{if } m_i = 1 \\ v_i, & \text{otherwise,} \end{cases} \quad x''_i = \begin{cases} v_i, & \text{if } m_i = 1 \\ u_i, & \text{otherwise.} \end{cases}$$

Then one element of $\{x', x''\}$ chosen uniformly at random is returned. The well-known uniform crossover and k-point crossover are examples of mask-based crossover operators.

For a frequent use of crossover, i.e., $p_c = 1$, [9, Lemma 2], Corollary 3, and Lemma 4 yield the following.

Theorem 4: Assume that the GA in Algorithm 2 with $p_c = 1$ uses any mask-based crossover operator, a bitwise mutation with mutation rate χ/n for any fixed constant $\chi > 0$, and one of the following selection mechanisms:

1) k-tournament selection with $k \geq 8(1 + \delta)e^\chi$;
2) (μ, λ)-selection with $\lambda/\mu \geq 2(1 + \delta)e^\chi$;
3) exponential ranking selection with $\eta \geq 8(1 + \delta)e^\chi$;

for any constant $\delta > 0$. Then there exists a constant $c > 0$, such that the expected runtime of the GA is:

1) $\mathcal{O}(n\lambda)$ on ONEMAX if $\lambda \geq c \ln n$;
2) $\mathcal{O}\left(n^2 + n\lambda \ln \lambda\right)$ on LEADINGONES if $\lambda \geq c \ln n$.

The proof can be found in the supplementary materials.

In the next sections, we further demonstrate the generality of Theorem 1 through Corollary 2 by deriving bounds on the expected runtime of GAs with $p_c = 1 - \Omega(1)$ to optimize or to approximate the optimal solutions.

B. Optimization on Permutation Space

Given n distinct elements from a totally ordered set, we consider the problem of ordering them so that some *measure of sortedness* is maximized. In [49], the (1+1) EA was analyzed on several measures of sortedness, including $\text{INV}(\pi)$ which is defined to be the number of pairs (i, j) such that $1 \leq i < j \leq n$, $\pi(i) < \pi(j)$ (i.e., pairs in correct order). We show that with the method introduced in this paper, analyzing GAs on Sorting problem with INV measure, denoted by $\text{SORTING}_{\text{Inv}}$, is not much harder than analyzing the (1+1) EA.

For the mutation, we use the Exchange(π) operator [49], which consecutively applies N pairwise exchanges between uniformly selected pairs of indices, where N is a random number drawn from a Poisson distribution with parameter 1.

Theorem 5: If the GA in Algorithm 2 with $p_c = 1 - \Omega(1)$ uses any crossover operator, the Exchange mutation operator, one of the selection mechanisms k-tournament selection, (μ, λ)-selection, and linear or exponential ranking selection, with their parameters k, λ/μ and η being set to no less than $(1+\delta)e/(1-p_c)$, then there exists a constant $c > 0$ such that if the population size is $\lambda \geq c \ln n$, the expected time to obtain the optimum of $\text{SORTING}_{\text{Inv}}$ is $\mathcal{O}\left(n^2\lambda\right)$.

The proof can be found in the supplementary materials.

C. Search for Local Optima

A great interest in the area of combinatorial optimization is to find approximate solutions to NP-hard problems, because exact solutions for such problems are unlikely be computable in polynomial time under the so-called P $\neq$ NP hypothesis. In the case of maximization problems, a feasible solution is called a *ρ-approximate solution* if its objective function value is at least ρ times the optimum for some $\rho \in (0, 1]$. Local search is one method among others to approximate solutions for combinatorial optimization problems through finding local optima (a formal definition is given below). For a number of well-known problems, it was shown [1] that any local optimum is guaranteed to be a ρ-approximate solution with a constant ρ.

Suppose that a *neighborhood* $\mathcal{N}(x) \subseteq \mathcal{X}$ is defined for every $x \in \mathcal{X}$. The mapping $\mathcal{N} : \mathcal{X} \to 2^{\mathcal{X}}$ is called the *neighborhood mapping* and all elements of $\mathcal{N}(x)$ are called *neighbors* of x. For example, a frequently used neighborhood mapping in the case of binary search space $\mathcal{X} = \{0, 1\}^n$ is defined by the Hamming distance $H(\cdot, \cdot)$ and a radius r as $\mathcal{N}(x) = \{y \mid H(x, y) \leq r\}$. If $f(y) \leq f(x)$ holds for all neighbors y of $x \in \mathcal{X}$, then x is called a *local optimum* with respect to $\mathcal{N}$. The set of all local optima is denoted by $\mathcal{LO}$ (note that global optima also belong to $\mathcal{LO}$).

A local search method starts from some initial solution y_0. Each iteration of the algorithm consists of moving from the current solution x to a new solution in its neighborhood, so that the value of the fitness function is increased. The algorithm continues until a local optimum is reached. Let m be the number of different fitness values attained by solutions from $\mathcal{X} \backslash \mathcal{LO}$ plus 1. Then starting from any point, the local search method finds a local optimum within at most $m - 1$ steps, each step requiring at most $\mathcal{N}(x)$ fitness evaluations.

The following result provides sufficient conditions that ensure the GA finds (at least) a local optimum with a runtime not much greater than that of the local search.

Corollary 4: Given some positive constants p_0, ε_0, and δ, define the following conditions:

(X1) $p_{\text{mut}}(y \mid x) \geq s$ for any $x \in \mathcal{X}$, $y \in \mathcal{N}(x)$;
(X2) $p_{\text{mut}}(x \mid x) \geq p_0$ for all $x \in \mathcal{X}$;
(X3) $p_{\text{xor}}(f(x') \geq \max\{f(u), f(v)\} \mid u, v) \geq \varepsilon_0$ for any $u, v \in \mathcal{X}$;
(X4.1) The nonelitist GA in Algorithm 2 is set with $p_c = 1$, and it uses one of the following selection mechanisms:
  1) k-tournament selection with $k \geq ([4(1+\delta)]/[\varepsilon_0 p_0])$;
  2) (μ, λ)-selection with $(\lambda/\mu) \geq ([1+\delta]/[\varepsilon_0 p_0])$;
  3) exponential ranking selection with $\eta \geq [(4(1+\delta))/(\varepsilon_0 p_0)]$;
(X4.2) The nonelitist GA is set with $p_c = 1 - \Omega(1)$, and it uses one of the following selection mechanisms: k-tournament selection, (μ, λ)-selection, linear or exponential ranking selection, with their parameters k, λ/μ, and η being set to no less than $([1+\delta]/[(1-p_c)p_0])$.

If (X1)–(X3) and (X4.1) hold, or exclusively (X1), (X2) and (X4.2) hold, then there exists a constant c, such that for $\lambda \geq c \ln(m/s)$, a local optimum is reached for the first time after $\mathcal{O}(m\lambda \ln \lambda + (m/s))$ fitness evaluations in expectation.

Condition (X4.1) or (X4.2) characterizes the setting of selection mechanisms, while (X1)–(X3) bear the properties of the variation operators over the neighborhood structure $\mathcal{N}$. Particularly, (X1) assumes a lower bound s on the probability that the mutation operator transforms an input solution into a specific neighbor. Note that in most of the local search algorithms, the neighborhood $\mathcal{N}(x)$ may be enumerated in polynomial time of the problem input size. For such neighborhood mappings, a mutation operator that generates the uniform distribution over $\mathcal{N}(x)$ will satisfy (X1) with $1/s$ polynomially bounded in the problem input size.

If crossover is frequently used, i.e., $p_c = 1$, we also need to satisfy condition (X3) on the crossover operator. It requires that the fitness of solution x on the output of crossover is not less than the fitness of parents with probability at least ε_0. Note that such a requirement is satisfied with $\varepsilon_0 = 1$ for the optimized crossover operators, where the offspring is computed as a solution to the *optimal recombination problem* (see [23]). This supplementary problem is known to be polynomially solvable for maximum clique, set packing, set partition and some other NP-hard problems (see [23]).

The proof of Corollary 4 directly follows from Corollaries 3 and 2 of Theorem 1, see supplementary materials.

Consider the binary search space $\{0, 1\}^n$ with Hamming neighborhood of a constant radius r, a fitness function f such that $m \in \text{poly}(n)$, and assume that the GA uses the bitwise mutation operator and $p_c = 1 - \Omega(1)$. This operator

outputs a string y, given a string x, with probability $p_{\mathrm{m}}^{H(x,y)}(1-p_{\mathrm{m}})^{n-H(x,y)}$. Note that probability $p_{\mathrm{m}}^{j}(1-p_{\mathrm{m}})^{n-j}$, as a function of p_{m} attains its maximum at $p_{\mathrm{m}} = j/n$. It is easy to show (see [22]) that for any $x \in \mathcal{X}$ and $y \in \mathcal{N}(x)$, the bitwise mutation operator with $p_{\mathrm{m}} = r/n$ satisfies the condition $p_{\mathrm{mut}}(y \mid x) = \mathcal{O}(1/n^r)$. For a sufficiently large n and any $x \in \mathcal{X}$ holds $p_{\mathrm{mut}}(x \mid x) \geq e^{-r}/2 = \Omega(1)$. By Corollary 4, a GA with the above mentioned operators, given appropriate λ, p_{m} and p_c, first visits a local optimum with respect to a constant radius Hamming neighborhood after a polynomially bounded number of fitness evaluations in expectation.

Consider the following two unconstrained (and unweighted) problems.

1) MAX-SAT: Given a CNF formula in n logical variables which is represented by m' clauses $\mathbf{c}_1, \ldots, \mathbf{c}_{m'}$ and each clause is a disjunction of logical variables or their negations, it is required to find an assignment of the variables so that the number of satisfied clauses is maximized.
2) MAX-CUT: Given an undirected graph $G = (V, E)$, it is required to find a partition of V into two sets $(S, V \setminus S)$, so that $\delta(S) := |\{(u, v) \mid (u, v) \in E, u \in S, v \notin S\}|$, is maximized.

Both problems are NP-hard, and their solutions can be naturally represented by bitstrings. Particularly, any local optimum with respect to the neighborhood defined by Hamming distance 1 has at least half the optimal fitness [1].

Theorem 6: Suppose the GA in Algorithm 2 is applied to MAX-SAT or to MAX-CUT using a bitwise mutation with $p_{\mathrm{m}} = \chi/n$, where $\chi > 0$ is a constant, a crossover with $p_c = 1 - \Omega(1)$ and one of the selection mechanisms: k-tournament selection, (μ, λ)-selection, linear or exponential ranking selection, with their parameters k, λ/μ and η being set to no less than $([(1+\delta)e^{\chi}]/[1-p_c])$, where $\delta > 0$ is any constant. Then there exists a constant c, such that for $\lambda \geq c\ln(nm')$, a 1/2-approximate solution is reached for the first time after $\mathcal{O}\big(m'\lambda\ln\lambda + nm'\big)$ fitness evaluations in expectation for MAX-SAT, and after $\mathcal{O}(|E|\lambda\ln\lambda + |V|\,|E|)$ for MAX-CUT.

The proof is analogous to the analysis of ℓ-UNIMODAL function in Theorem 3, combined with Corollary 4, where $m \leq m' + 1$ for MAX-SAT and $m \leq |E| + 1$ for MAX-CUT.

V. ESTIMATION OF DISTRIBUTION ALGORITHMS

There are few rigorous runtime results for UMDA and other EDAs. The techniques used in previous analyses of EDAs were often complex, e.g., relying on Markov chains theory. Surprisingly, even apparently simple problems, such as the expected runtime of UMDA on ONEMAX, were open until recently. Indeed, much more is known about classical EAs, e.g., the (1+1) EA solves ONEMAX in expected time $\Theta(n\ln n)$, and this is optimal for the class of unary unbiased black-box algorithms [38].

Algorithm 1 matches closely the typical behavior of EDAs: given a current distribution over the search space, sample a finite number of search points, and update the probability distribution. We demonstrate the ease at which the expected runtime of UMDA with margins and truncation selection on the ONEMAX function can be obtained using the level-based theorem without making any simplifying assumptions about the optimization process.

Algorithm 4 UMDA

Require:
A pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$, and "margins" $m' \in [0, \mu/2)$.

1: Initialise the vector $p_0 := (1/2, \ldots, 1/2)$.
2: **for** $t = 1, 2, 3, \ldots$ **do**
3: **for** $x = 1$ to λ **do**
4: Sample the x-th individual $P_t(x, \cdot)$ according to
$$P_t(x, i) \sim \text{Bernoulli}(p_{t-1}(i)) \text{ for all } i \in [n].$$
5: Sort the population P_t according to f.
6: For all $i \in [n]$, with $X_i := \sum_{k=1}^{\mu} P_t(k, i)$, define
$$p_t(i) := \max\left\{\min\left\{\frac{X_i}{\mu}, 1 - \frac{m'}{\mu}\right\}, \frac{m'}{\mu}\right\}.$$

To start, we give a formal description of UMDA. If $P \in \mathcal{X}^{\lambda}$ is a population of λ solutions, let $P(k, i)$ denote the value in the ith bit position of the kth solution in P. The UMDA with (μ, λ)-truncation selection is defined in Algorithm 4.

The algorithm has three parameters, the parent population size μ, the offspring population size λ, and a parameter $m' < \mu$ controlling the size of the margins. It is necessary to set $m' > 0$ to prevent a premature convergence, e.g., without this margin, $p_t(i)$ can go to a nonoptimal fixation, this prevents further exploration and causes an infinite runtime. Based on insights about optimal mutation rates in the (1+1) EA, we will use the parameter setting $m' = \mu/n$ in the rest of this section.

It is immediately clear that the UMDA in Algorithm 4 is a special case of Algorithm 1. The probability distribution $D(P_t)$ of y is computed in step 7, and is $\Pr(y = x) = \prod_{j=1}^{n} p_t(j)^{x_j}(1-p_t(j))^{1-x_j}$ for any bitstring $x \in \{0,1\}^n$.

In some other randomized search heuristics, such as ant colony optimization and cGA, the sampling distribution D_t does not only depend on the current population but also on additional information, such as pheromone values. The level-based theorem does not apply to such algorithms.

Theorem 7: Given any positive constants $\delta \in (0, 1)$, and $\gamma_0 \leq [1/((1+\delta)13e)]$, the UMDA with offspring population size λ with $b\ln(n) \leq \lambda \leq n/\gamma_0$ for some constant $b > 0$, parent population size $\mu = \gamma_0\lambda$, and margins $m' = \mu/n$, has expected optimization time $\mathcal{O}(n\lambda\ln\lambda)$ on ONEMAX.

The proof which is available in our supplementary materials follows our guidelines of level-based analysis, a preliminary version of it appeared in [13]. To obtain lower bounds on the tail of the level distribution, we make use of the Feige inequality [24] (or see [13, Corollary 3]).

A similar analysis for LEADINGONES [13] yields a runtime bound $\mathcal{O}\big(n\lambda\ln\lambda + n^2\big)$ with offspring population size $\lambda \geq b\ln(n)$ for some constant $b > 0$ without use of Feige's inequality. The previous result [8] on LEADINGONES requires a larger population size and gives a longer runtime bound.

Table I summarizes the runtime bounds for the example applications of the tools presented in this paper and the above mentioned result for UMDA on LEADINGONES.

TABLE I
SUMMARY OF RESULTS FOR THE GA [ALGORITHM 2, $p_c = 1 - \Omega(1)$], GA1 (ALGORITHM 2, $p_c = 1$) AND UMDA (ALGORITHM 4, $m' = \mu/n$)

RUNTIME RESULT

Problem	Algorithm	Min. λ	Runtime
ONEMAX	GA, GA1	$c \ln n$	$\mathcal{O}(n\lambda)$
	UMDA	$c \ln n$	$\mathcal{O}(n\lambda \ln \lambda)$
LEADINGONES	GA, GA1, UMDA	$c \ln n$	$\mathcal{O}\left(n^2 + n\lambda \ln \lambda\right)$
ℓ-UNIMODAL	GA	$c \ln(n\ell)$	$\mathcal{O}(n\ell + \ell\lambda \ln \lambda)$
LINEAR	GA	$c \ln n$	$\mathcal{O}\left(n^2 + n\lambda \ln \lambda\right)$
JUMP_r	GA	$cr \ln n$	$\mathcal{O}\left(\left(\frac{n}{\chi}\right)^r + n\lambda + \lambda \ln \lambda\right)$
$\text{ROYALROAD}_{r \geq 2}$	GA	$cr \ln n$	$\mathcal{O}\left(\left(\frac{n}{\chi}\right)^r \ln\left(\frac{n}{r}\right) + \frac{n\lambda \ln \lambda}{r}\right)$
$\text{SORTING}_{\text{INV}}$	GA	$c \ln n$	$\mathcal{O}\left(n^2\lambda\right)$
$\frac{1}{2}$-approx. MAX-SAT	GA	$c \ln(nm')$	$\mathcal{O}(nm' + m'\lambda \ln \lambda)$
$\frac{1}{2}$-approx. MAX-CUT	GA	$c \ln(\lvert V\rvert \lvert E\rvert)$	$\mathcal{O}(\lvert V\rvert \lvert E\rvert + \lvert E\rvert\lambda \ln \lambda)$

CONFIGURATION

Alg.	Recomb.	Selection	Setting
GA	any	k-tournament	$k \geq \frac{(1+\delta)e^{\chi}}{1-p_c}$
	any	(μ, λ)-selection	$\frac{\lambda}{\mu} \geq \frac{(1+\delta)e^{\chi}}{1-p_c}$
	any	linear ranking	$\eta \geq \frac{(1+\delta)e^{\chi}}{1-p_c}$
	any	exp. ranking	$\eta \geq \frac{(1+\delta)e^{\chi}}{1-p_c}$
GA1	mask-based	k-tournament	$k \geq 8(1+\delta)e^{\chi}$
	mask-based	(μ, λ)-selection	$\frac{\lambda}{\mu} \geq 2(1+\delta)e^{\chi}$
	mask-based	exp. ranking	$\eta \geq 8(1+\delta)e^{\chi}$
UMDA	n/a	(μ, λ)-selection	$\frac{\lambda}{\mu} \geq 13(1+\delta)e$

On $\{0,1\}^n$, GA and GA1 use bitwise mutation operator with rate χ/n, where χ is any constant. On permutation search space, i.e., Sorting, GA uses Exchange mutation and its setting assumes $\chi = 1$. In the case of MAX-SAT, n is the number of logical variables and m' is the number of clauses. Parameter δ is any positive constant, and c is some constant.

VI. LEVEL-BASED THEOREM IS ALMOST TIGHT

How accurate are the time bounds provided by the level-based theorem? To answer this question, we first interpret the theorem as a universally quantified statement over the operators D satisfying the conditions of the theorem. More formally, given a choice of level-partition and set of parameters $z_1, \ldots, z_{m-1}, \delta, \gamma_0$, which we collectively denote by Θ, the theorem can be expressed in the form of $\forall D \in \mathcal{D}_\Theta \colon E[T_D] \leq t_\Theta$, where $\mathcal{D}_\Theta$ is the set of operators D in Algorithm 1 that satisfy the conditions of the level-based theorem with parameterizations Θ, $E[T_D]$ is the expected runtime of Algorithm 1 with a given operator D, and t_Θ is the upper time bound provided by the level-based theorem which depends on Θ.

In order to obtain an accurate bound for a specific operator D, e.g., the (μ, λ) EA applied to the ONEMAX function, it is necessary to choose a parametrization Θ that reflects this process as tightly as possible. If the bounds on the upgrade probabilities z_j for the (μ, λ) EA are too small, then the class $\mathcal{D}_\Theta$ includes other processes which are slower than the (μ, λ) EA, and the corresponding bound t_Θ cannot be accurate. Hence, the theorem is limited by the accuracy at which one can describe the process by some class $\mathcal{D}_\Theta$.

Assuming a fixed parameterizations Θ, it is possible to make a precise statement about the tightness of the upper bound t_Θ. Theorem 8 below is an existential statement on the form $\exists D \in \mathcal{D}_\Theta \colon E[T_D] \geq t'_\Theta$, where the lower bound t'_Θ is close to the upper bound t_Θ. Hence, given the information the theorem has about the process through Θ, the runtime bound is close to optimal. More information about the process would be required to obtain a more accurate runtime bound.

In some concrete cases, one can prove that the level-based theorem is close to optimal, using parallel black-box complexity theory [2]. From Corollary 2 with $p_c = 0$, which specializes the level-based theorem to algorithms with unary mutation operators, one can obtain the bounds $\mathcal{O}(n\lambda + n \ln n)$ for ONEMAX, and $\mathcal{O}\left(n\lambda \ln \lambda + n^2\right)$ for LEADINGONES for appropriately parameterized EAs. These bounds are within a $\mathcal{O}(\ln \lambda)$-factor of the lower bounds that hold for any parallel unbiased black-box algorithm [2]. For population sizes $\lambda = \mathcal{O}(n/\ln n)$ and $\lambda = \Omega(\ln n)$, the resulting $\mathcal{O}\left(n^2\right)$ bound on LEADINGONES is asymptotically tight, because it matches the lower bound that holds for all black-box algorithms with unary unbiased variation operators [38].

Theorem 8: Given any partition of $\mathcal{X}$ into m nonempty subsets $(A_1, \ldots, A_m)$, for any $z_1, \ldots, z_{m-1}, \delta, \gamma_0 \in (0, 1)$, where $z_j \in (0, \gamma_0)$ for all $j \in [m-1]$, and $\lambda \in \mathbb{N}$, there exists a mapping D which satisfies conditions (G1), (G2), and (G3) of Theorem 1, such that Algorithm 1 with mapping D has expected hitting time

$$E[T] \geq \left(\frac{2}{3\delta} \sum_{j=1}^{m-2} \lambda \ln\left(\frac{\gamma_0 \lambda}{1 + 2\lambda z_j + 1/\delta^2}\right)\right) + \sum_{j=1}^{m-1} \frac{1}{z_j}$$

where $T := \min\{\lambda t \in \mathbb{N} \mid |P_t \cap A_m| > 0\}$.

The proof can be found in the supplementary materials.

VII. CONCLUSION

This paper introduces a new technique, the so-called level-based analysis, that easily yields upper bounds on the expected runtime of complex, nonelitist search processes. The technique was first illustrated on GAs. We have shown that GAs efficiently optimize standard benchmark functions and some combinatorial optimization problems. As long as the population size is not overly large, the population does not incur an asymptotic slowdown on these functions compared to standard EAs that do not use populations. So, speedups can be achieved by parallelizing fitness evaluations. Furthermore, previous work using a weaker form of the level-based analysis indicates that nonelitist, population-based EAs have an advantage on more complex problems, including those with noisy [12], dynamic [11], and peaked [15] fitness landscapes. To demonstrate the generality of the theorem, we also provided runtime results for the UMDA algorithm, an estimation of distribution algorithm, for which few theoretical results exist. Finally, we have shown via lower bounds on the runtime of a concrete process that, given the information the theorem requires about the process, the upper bounds are close to tight.

The conditions of the level-based theorem yield settings for algorithmic parameters, such as population size, mutation and crossover rates, selection pressure etc., that are sufficient to guarantee a time complexity bound. This opens up the possibility of theory-led design of EAs with guaranteed runtimes,

where the algorithm is designed to satisfy the conditions of the level-based theorem. This paper also opens several new directions for future work. An important open problem is to develop techniques for proving lower bounds on the runtime of Algorithm 1. Rowe and Sudholt [48] showed that the nonelitist $(1, \lambda)$ EA becomes inefficient when the population size is too small. While condition (G3) in this paper gives a sufficient condition for the population size, it would be interesting to determine a necessary condition on the population size to efficiently reach the last level A_m.

APPENDIX

We state the lemmas used in the proof of Theorem 1. Their proofs are provided in the supplementary materials.

Lemma 5: The functions g_1 and g_2 defined below are level functions for any $c > 0$, $\kappa \in (0, 1)$, $x \in [\lambda]$, $y \in [m]$ and $\gamma_j, q_j \in (0, 1]$ for each $j \in [m-1]$

$$g_1(x, y) \coloneqq \ln\left(\frac{1+c\lambda}{1+c\max\{x, \gamma_y\lambda\}}\right) + \sum_{i=y+1}^{m-1} \ln\left(\frac{1+c\lambda}{1+c\gamma_i\lambda}\right)$$

$$g_2(x, y) \coloneqq \frac{(1-\kappa)^x}{q_y} + \sum_{i=y+1}^{m-1} \frac{1}{q_i}$$

for all $y \in [m-1]$, and $g_1(x, y) \coloneqq g_2(x, y) \coloneqq 0$ for $y = m$.

Lemma 6 (Improved Version of [14, Lemma 5]: If $X \sim \mathrm{Bin}(\lambda, p)$ with $p \geq (i/\lambda)(1+\delta)$ and $i \geq 1$ for some $\delta \in (0, 1]$, then

$$E\left[\ln\left(\frac{1+\delta X/2}{1+\delta i/2}\right)\right] \geq \frac{\delta^2}{7}.$$

Lemma 7: Let $\{X_i\}_{i\in[\lambda]}$ be independent identically distributed random variables, define $Y(j) \coloneqq \sum_{i=1}^{\lambda} \mathbb{1}_{\{X_i \geq j\}}$ for any $j \in \mathbb{R}$. It holds for any $a, b, c, j \in \mathbb{R}$ with $c \geq 0$ and $b \leq \lambda$ that

$$(i) \quad \Pr(Y(j+c) \geq a \mid Y(j) \geq b) \geq \Pr(Y(j+c) \geq a)$$

and for any nondecreasing function f

$$(ii) \quad E\big[f(Y(j+c)) \mid Y(j) \geq b\big] \geq E\big[f(Y(j+c))\big]$$

provided that both expectations are well-defined.

ACKNOWLEDGMENT

Early ideas were discussed at Dagstuhl Seminars 13271 and 15211 "Theory of Evolutionary Algorithms."

REFERENCES

[1] G. Ausiello and M. Protasi, "Local search, reducibility and approximability of NP-optimization problems," *Inf. Process. Lett.*, vol. 54, no. 2, pp. 73–79, 1995.

[2] G. Badkobeh, P. K. Lehre, and D. Sudholt, "Unbiased black-box complexity of parallel search," in *Proc. PPSN XIII*, Ljubljana, Slovenia, 2014, pp. 892–901.

[3] H.-G. Beyer, H.-P. Schwefel, and I. Wegener, "How to analyse evolutionary algorithms," *Theor. Comput. Sci.*, vol. 287, no. 1, pp. 101–130, 2002.

[4] T. Chen, J. He, G. Sun, G. Chen, and X. Yao, "A new approach for analyzing average time complexity of population-based evolutionary algorithms on unimodal problems," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 5, pp. 1092–1106, Oct. 2009.

[5] T. Chen, P. K. Lehre, K. Tang, and X. Yao, "When is an estimation of distribution algorithm better than an evolutionary algorithm?" in *Proc. CEC*, Trondheim, Norway, 2009, pp. 1470–1477.

[6] T. Chen, K. Tang, G. Chen, and X. Yao, "On the analysis of average time complexity of estimation of distribution algorithms," in *Proc. CEC*, Singapore, 2007, pp. 453–460.

[7] T. Chen, K. Tang, G. Chen, and X. Yao, "Rigorous time complexity analysis of univariate marginal distribution algorithm with margins," in *Proc. CEC*, Trondheim, Norway, 2009, pp. 2157–2164.

[8] T. Chen, K. Tang, G. Chen, and X. Yao, "Analysis of computational time of simple estimation of distribution algorithms," *IEEE Trans. Evol. Comput.*, vol. 14, no. 1, pp. 1–22, Feb. 2010.

[9] D. Corus, D.-C. Dang, A. V. Eremeev, and P. K. Lehre, "Level-based analysis of genetic algorithms and other search processes," in *Proc. PPSN XIII*, Ljubljana, Slovenia, 2014, pp. 912–921.

[10] D.-C. Dang *et al.*, "Emergence of diversity and its benefits for crossover in genetic algorithms," in *Proc. PPSN XIV*, Edinburgh, U.K., 2016, pp. 890–900.

[11] D.-C. Dang, T. Jansen, and P. K. Lehre, "Populations can be essential in tracking dynamic optima," *Algorithmica*, vol. 78, no. 2, pp. 660–680, 2017.

[12] D.-C. Dang and P. K. Lehre, "Efficient optimisation of noisy fitness functions with population-based evolutionary algorithms," in *Proc. FOGA XIII*, Aberystwyth, U.K., 2015, pp. 62–68.

[13] D.-C. Dang and P. K. Lehre, "Simplified runtime analysis of estimation of distribution algorithms," in *Proc. GECCO*, Madrid, Spain, 2015, pp. 513–518.

[14] D.-C. Dang and P. K. Lehre, "Runtime analysis of non-elitist populations: From classical optimisation to partial information," *Algorithmica*, vol. 75, no. 3, pp. 428–461, 2016.

[15] D.-C. Dang and P. K. Lehre, "Self-adaptation of mutation rates in non-elitist populations," in *Proc. PPSN XIV*, Edinburgh, U.K., 2016, pp. 803–813.

[16] B. Doerr and C. Doerr, "Optimal parameter choices through self-adjustment: Applying the 1/5-th rule in discrete settings," in *Proc. GECCO*, Madrid, Spain, 2015, pp. 1335–1342.

[17] B. Doerr and C. Doerr, "A tight runtime analysis of the (1+(λ, λ)) genetic algorithm on OneMax," in *Proc. GECCO*, Madrid, Spain, 2015, pp. 1423–1430.

[18] B. Doerr and M. Künnemann, "How the (1+λ) evolutionary algorithm optimizes linear functions," in *Proc. GECCO*, Amsterdam, The Netherlands, 2013, pp. 1589–1596.

[19] S. Droste, "A rigorous analysis of the compact genetic algorithm for linear functions," *Nat. Comput.*, vol. 5, no. 3, pp. 257–283, 2006.

[20] S. Droste, T. Jansen, and I. Wegener, "On the analysis of the (1+1) evolutionary algorithm," *Theor. Comput. Sci.*, vol. 276, nos. 1–2, pp. 51–81, 2002.

[21] S. Droste, T. Jansen, and I. Wegener, "Upper and lower bounds for randomized search heuristics in black-box optimization," *Theor. Comput. Syst.*, vol. 39, no. 4, pp. 525–544, 2006.

[22] A. Eremeev, "Hitting times of local and global optima in genetic algorithms with very high selection pressure," *Yugoslav J. Oper. Res.*, vol. 27, no. 3, pp. 323–339, 2017.

[23] A. V. Eremeev and J. V. Kovalenko, "Optimal recombination in genetic algorithms for combinatorial optimization problems—Part I," *Yugoslav J. Oper. Res.*, vol. 24, no. 1, pp. 1–20, 2014.

[24] U. Feige, "On sums of independent random variables with unbounded variance, and estimating the average degree in a graph," in *Proc. 36th STOC*, 2004, pp. 594–603.

[25] T. Friedrich, T. Kötzing, M. S. Krejca, and A. M. Sutton, "The benefit of recombination in noisy evolutionary search," in *Proc. 26th ISAAC*, Nagoya, Japan, 2015, pp. 140–150.

[26] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Boston, MA, USA: Addison-Wesley, 1989.

[27] C. González, J. A. Lozano, and P. Larrañaga, "Analyzing the PBIL algorithm by means of discrete dynamical systems," *Complex Syst.*, vol. 12, no. 4, pp. 465–479, 2000.

[28] B. Hajek, "Hitting-time and occupation-time bounds implied by drift analysis with applications," *Adv. Appl. Probab.*, vol. 14, no. 3, pp. 502–525, 1982.

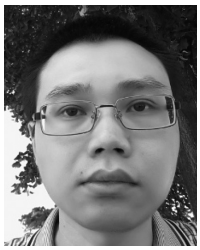
[29] J. He and X. Yao, "Drift analysis and average time complexity of evolutionary algorithms," *Artif. Intell.*, vol. 127, no. 1, pp. 57–85, 2001.

[30] J. He and X. Yao, "From an individual to a population: An analysis of the first hitting time of population-based evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 6, no. 5, pp. 495–511, Oct. 2002.

[31] T. Jansen and I. Wegener, "The analysis of evolutionary algorithms—A proof that crossover really can help," *Algorithmica*, vol. 34, no. 1, pp. 47–66, 2002.
[32] T. Kötzing, D. Sudholt, and M. Theile, "How crossover helps in pseudo-Boolean optimization," in *Proc. GECCO*, 2011, pp. 989–996.
[33] M. S. Krejca and C. Witt, "Lower bounds on the run time of the univariate marginal distribution algorithm on OneMax," in *Proc. Found. Genet. Algorithms (FOGA)*, 2017, pp. 65–79.
[34] P. Larrañaga and J. A. Lozano, Eds., *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation* (Genetic Algorithms and Evolutionary Computation), vol. 2. New York, NY, USA: Springer, 2002.
[35] J. Lässig and D. Sudholt, "General upper bounds on the runtime of parallel evolutionary algorithms," *Evol. Comput.*, vol. 22, no. 3, pp. 405–437, 2014.
[36] P. K. Lehre, "Fitness-levels for non-elitist populations," in *Proc. GECCO*, Dublin, Ireland, 2011, pp. 2075–2082.
[37] P. K. Lehre and P. T. H. Nguyen, "Improved runtime bounds for the univariate marginal distribution algorithm via anti-concentration," in *Proc. GECCO*, Berlin, Germany, 2017, pp. 1383–1390.
[38] P. K. Lehre and C. Witt, "Black-box search by unbiased variation," *Algorithmica*, vol. 64, no. 4, pp. 623–642, 2012.
[39] P. K. Lehre and X. Yao, "Crossover can be constructive when computing unique input–output sequences," *Soft Comput.*, vol. 15, no. 9, pp. 1675–1687, 2011.
[40] P. K. Lehre and X. Yao, "On the impact of mutation-selection balance on the runtime of evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 16, no. 2, pp. 225–241, Apr. 2012.
[41] A. Moraglio and D. Sudholt, "Principled design and runtime analysis of abstract convex evolutionary search," *Evol. Comput.*, vol. 25, no. 2, pp. 205–236, 2017.
[42] H. Mühlenbein and G. Paaß, "From recombination of genes to the estimation of distributions I. binary parameters," in *Proc. PPSN IV*, Berlin, Germany, 1996, pp. 178–187.
[43] H. Mühlenbein, "The equation for response to selection and its use for prediction," *Evol. Comput.*, vol. 5, no. 3, pp. 303–346, 1997.
[44] F. Neumann, P. S. Oliveto, and C. Witt, "Theoretical analysis of fitness-proportional selection: Landscapes and efficiency," in *Proc. GECCO*, Montreal, QC, Canada, 2009, pp. 835–842.
[45] P. S. Oliveto and C. Witt, "Improved time complexity analysis of the simple genetic algorithm," *Theor. Comput. Sci.*, vol. 605, pp. 21–41, Nov. 2015.
[46] M. Pelikan, K. Sastry, and D. E. Goldberg, "Scalability of the Bayesian optimization algorithm," *Int. J. Approx. Reason.*, vol. 31, no. 3, pp. 221–258, 2002.
[47] A. Prügel-Bennett, J. E. Rowe, and J. Shapiro, "Run-time analysis of population-based evolutionary algorithm in noisy environments," in *Proc. FOGA XIII*, Aberystwyth, U.K., 2015, pp. 69–75.
[48] J. E. Rowe and D. Sudholt, "The choice of the offspring population size in the $(1, \lambda)$ evolutionary algorithm," *Theor. Comput. Sci.*, vol. 545, pp. 20–38, Aug. 2014.
[49] J. Scharnow, K. Tinnefeld, and I. Wegener, "The analysis of evolutionary algorithms on sorting and shortest paths problems," *J. Math. Model. Algorithms*, vol. 3, no. 4, pp. 349–366, 2004.
[50] J. L. Shapiro, "Drift and scaling in estimation of distribution algorithms," *Evol. Comput.*, vol. 13, no. 1, pp. 99–123, 2005.
[51] D. Sudholt, "How crossover speeds up building block assembly in genetic algorithms," *Evol. Comput.*, vol. 25, no. 2, pp. 237–274, 2017.
[52] D. Sudholt and C. Witt, "Update strength in EDAs and ACO: How to avoid genetic drift," in *Proc. GECCO*, Denver, CO, USA, 2016, pp. 61–68.
[53] M. D. Vose, *The Simple Genetic Algorithm: Foundations and Theory*. Cambridge, MA, USA: MIT Press, 1999.
[54] M. D. Vose and A. H. Wright, "Stability of vertex fixed points and applications," in *Proc. FOGA III*, Estes Park, CO, USA, 1995, pp. 103–113.
[55] D. Williams, *Probability With Martingales*. Cambridge, U.K.: Cambridge Univ. Press, 1991.
[56] C. Witt, "Runtime analysis of the $(\mu+1)$ EA on simple pseudo-Boolean functions," *Evol. Comput.*, vol. 14, no. 1, pp. 65–86, 2006.
[57] C. Witt, "Upper bounds on the runtime of the univariate marginal distribution algorithm on OneMax," in *Proc. GECCO*, Berlin, Germany, 2017, pp. 1415–1422.
[58] Y. Yu, C. Qian, and Z.-H. Zhou, "Switch analysis for running time analysis of evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 19, no. 6, pp. 777–792, Dec. 2015.
[59] Q. Zhang and H. Mühlenbein, "On the convergence of a class of estimation of distribution algorithms," *IEEE Trans. Evol. Comput.*, vol. 8, no. 2, pp. 127–136, Apr. 2004.

Dogan Corus received the B.S. and M.S. degrees in industrial engineering from Koc University, Istanbul, Turkey, and the Ph.D. degree in computer science from the University of Nottingham, Nottingham, U.K., under the supervision of P. K. Lehre.

He joined the University of Sheffield, Sheffield, U.K., as a Research Associate in 2015. His current research interests include runtime analysis of bio-inspired algorithms, in particular, population-based algorithms and genetic algorithms.



Duc-Cuong Dang received the Engineer's degree in computer and network, the M.Sc. degree in computer science, and the Ph.D. degree in computer science and operational research from the Université de Technologie de Compiègne, Compiègne, France, in 2008 and 2011, respectively.

From 2011 to 2012, he was a Lecturer with the Université de Technologie de Compiègne, and from 2013 to 2016, he held a Research Fellow position with the University of Nottingham, Nottingham, U.K. He is interested in applying Mathematics and Computer Science to solve real-world optimization problems. His current research interest includes the theoretical aspects of heuristic and metaheuristic search.

Anton V. Eremeev received the graduation degree in applied mathematics and the Ph.D. degree in application of computers, mathematical modeling, and mathematical methods in scientific research from Omsk State University, Omsk, Russia, in 1996 and 2000, respectively.

He is a Deputy Director and the Lead Researcher with the Omsk Department of Sobolev Institute of Mathematics, Siberian Branch of Russian Academy of Sciences, Novosibirsk, Russia, a Professor with Omsk State University. His current research interests include theory of evolutionary algorithms, heuristics and software for scheduling and production line optimization, approximation algorithms for NP-hard problems, and optimization problems in electricity markets.

Per Kristian Lehre received the M.Sc. and Ph.D. degrees in computer science from the Norwegian University of Science and Technology, Trondheim, Norway, in 2006.

He is a Senior Lecturer with the University of Birmingham, Birmingham, U.K. He held Post-Doctoral positions with the School of Computer Science, University of Birmingham, Birmingham, U.K. and the Technical University of Denmark, Kgs. Lyngby, Denmark. From 2011, he was a Lecturer with the School of Computer Science, University of Nottingham, Nottingham, U.K., until 2017, when he returned to Birmingham. His current research interest includes theoretical aspects of nature-inspired search heuristics, in particular, runtime analysis of population-based evolutionary algorithms.

Dr. Lehre was a recipient of several best paper awards, including GECCO in 2006, 2009, 2010, and 2013, ICSTW in 2008, and ISAAC in 2014. He is an Editorial Board Member of Evolutionary Computation, and an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION. He was the Coordinator of the successful 2M euro EU-funded project SAGE which brought together the theory of evolutionary computation and population genetics.