City Research Online

City, University of London Institutional Repository

This is the accepted version of the paper.

This version of the publication may differ from the final published version.

Permanent repository link: http://openaccess.city.ac.uk/id/eprint/21328/

Link to published version:

City Research Online: http://openaccess.city.ac.uk/   publications@city.ac.uk

Elitism Levels Traverse Mechanism For The Derivation of Upper Bounds on Unimodal Functions

Aram Ter-Sarkisov

Department of Computer Science
Massey University
Wellington, New Zealand
Email: a.ter-sarkisov@massey.ac.nz

Abstract—In this article we present an Elitism Levels Traverse Mechanism that we designed to find bounds on population-based Evolutionary Algorithms solving unimodal functions. We prove its efficiency theoretically and test it on OneMax function deriving bounds $c\mu n \log n - O(\mu n)$. This analysis can be generalized to any similar algorithm using variants of elitist selection and genetic operators that flip or swap only 1 bit in each string.

Index Terms—Evolutionary computation, Genetic algorithms, Computational complexity

I. INTRODUCTION

We analyze an elitist population-based Evolutionary Algorithm with population size μ and recombination pool size λ, $(\mu + \lambda)$EA using a genetic operator 1-Bit-Swap that recombines information between parents (see [1]).

Most research in theoretical EA community is focused on mutation-based single species algorithms such as $(1 + 1)_{\frac{1}{\mu}}$ EA (see e.g. [2]–[4]) with some sharp bounds on runtime obtained for OneMax function such as $0.982n \log n$ in [4].

Results on population-based algorithms are less abundant, and are restricted to mostly $(\mu + 1)_{\frac{1}{\mu}}$EA (see [5]) with upper bound $O(\mu n + n \log n)$ and $(1 + \lambda)_{\frac{1}{\mu}}$EA (see [6], [7]) with upper bound on OneMax $O(n \log n + n\lambda)$ in [6] and all linear functions $O(\lfloor \frac{n \log n}{\lambda} + \frac{n \log \log \lambda}{\log \lambda} \rfloor)$ in [7].

Although so far $(\mu + \lambda)$ or $(N + N)$ EAs have deserved less attention, they have been the subject of analysis in [8]–[10]. Specifically, in [10] it was derived that for a $(N + N)$ EA with mutation and tournament selection solving OneMax the upper bound is $O(nN \log N + n \log n)$ if measured in the number of function evaluations.

Unfortunately many of these results are not directly comparable due to the difference in selection functions (fitness-proportional, truncation, elitist, tournament, etc) and elitism settings (save 1 best species or some variable proportion).

Even more significantly, it was shown already in [8] that population effect is generally problem-specific, so it is quite hard to generalize findings to other functions. There is ample evidence though (e.g. [5], [6]) that for mutation-based algorithms (incl. Randomized Local Search, RLS) optimizing simple functions such as OneMax population is not beneficial and tends to degrade performance.

II. ALGORITHMS AND PROBLEMS

A. Algorithm

Although the mechanism described in this paper is quite universal, we test it on $(\mu + \lambda)EA_{1BS}$ solving OneMax problem. This problem is well-known in EA community, recent achievements include [4], [11] with some sharp bounds. We selected this problem due to its simplicity and the ability to compare our findings to those available already.

TABLE I
ALGORITHM

$(\mu + \lambda)$ Evolutionary Algorithm using 1-Bit-Swap (1BS)
1   create μ starting species at random
2   $t = 0$
3   **loop**
4   select using a variant of fitness-proportional Tournament selection $\frac{\lambda}{2}$ pairs of parents into the pool
5   swap bits in each pair
6   keep the currently-elite species in the population, replace the rest with the pool, first with new currently-elite, then at random
7      $t = t + 1$
8   **end loop**

B. Selection function

Throughout the article we analyze an elitist recombination-driven $(\mu + \lambda)$ EA using a variant of tournament selection. It is both simple to implement and analyze. But since we recombine information between parents, we are interested in forming *pairs* of species in the recombination pool, and on the construction of these pairs the properties of the algorithm will be derived. This formation occurs in the following way:

TABLE II
SELECTION FUNCTION

Variant of Tournament Selection
1   $k = 0$
2   **loop**
3   select two species from the population at random
4   examine their fitness, the better one enters the pool
5      $k = k + 1$
6   **end loop**

Thus it is obvious that better-fit species have higher chances of entering the pool, so we can expect the proportion of α species to be higher in the pool rather than in the population.

C. 1-Bit-Swap Genetic Operator

We apply the 1-Bit-Swap operator that was found to be useful solving a large number of test problems in [1] and was analyzed extensively in [12], [13] to have outperformed the mainstream RLS algorithm both theoretically and numerically.

Another advantage of 1BS is that we can compare it directly to RLS, since both are local search operators that cannot move too far from the current best search point. The operator works in the following way:

TABLE III
GENETIC OPERATOR

	1-Bit-Swap Operator
1	$j = 0$
2	**loop**
3	select a bit in the first parent uniformly at random
4	select a bit in the second parent uniformly at random
5	swap values in these bits
5	$j = j + 1$
6	**end loop**

III. DEFINITIONS

A. Fitness levels partition

Basic approach to analyzing elitist EA with a simple 1-bit mutation solving unimodal binary-encoded EAs was introduced by Wegener in [14] that is based on fitness partitioning: on a set of binary strings $\{0,1\}^n$ size 2^n a partition into a finite number of nonempty subsets $A_1, A_2, \cdots, A_m$ is defined with ordering $A_1 \precsim A_2 \precsim \cdots \precsim A_m$ s.t. all $a \in A_m$ are the global optimum.

This approach allows definition and derivation of the lower bound of success probability of transition between states, $s_i(a) = P(A_{i+1}|A_i)$ and the upper bound on expected convergence time of the algorithm, expected first hitting time of the best fitness level, $E(X_f) \leq s_1^{-1} + s_2^{-1} \ldots + s_{m-1}^{-1}$. This idea can be extended to the situation when we apply non-negative weights $w(f)$ (see [2], [14]) and to derive lower bounds (by considering the upper bound on $s_i(a)$).

Another tool used extensively in the analysis of EAs are potential (auxiliary) functions that measure progress (see [14]). This is especially useful when working on functions that have fitness plateaus (see e.g. [15]), in which case we make the difference:

1) fitness functions decide whether the new binary input (species) is better than the old one
2) potential function tracks the progress between states of the algorithm (fitness levels)

OneMax (or some simple transformation of it) is used as a potential function for more complicated problems (Royal Roads, Binary Values, Short/Long Path etc).

B. Elitism Levels Partition

In this article we extend this approach to a population-based elitist algorithm, but rather than tracking the traverse of levels of fitness, we do the same to the levels of elitism, i.e., number of elite species in the population.

We focus on species that can either evolve to the currently-best over 1 iteration or are already best. Therefore, the population

is broken down into three disjoint subsets:

α : currently best species

β_1 : species with next-best fitness

β_{-1} : the rest of the population that cannot evolve over 1 generation

Since 1BS swaps exactly 1 bit between two parents, this partition in combination with the assumptions made above enables construction of a very precise model, since the value of α cannot 'jump' more than 1 level of fitness and only α-species can breed better population, but only β_1-species may evolve into α and change the probability of evolution.

C. α-levels subpartition

This additional partition is necessary for functions with plateaus for which we use potential functions explained above. The need for it becomes evident in the next section, when probabilities of evolving elite species on two types of functions are compared. In addition to the elitism levels partition, for functions with plateaus we need to subpartition the α-level.

In slight abuse of notation in the rest of this article, we denote A the set of chromosomes in the population with the highest fitness. Also γ is the length of the plateau of fitness. Therefore the set A can be partitioned into

$$A = A_0 \cup A_1 \cup \cdots \cup A_{\gamma-1}$$

where each subset A_m has equal fitness. In order to differentiate between A_m, we assign each elite species an additional auxiliary function, V_m that tracks progress to the next level by counting the number of 1-bits in the fitness level: $A_0 \precsim A_1 \precsim \ldots \precsim A_{\gamma-1}$ with corresponding auxiliary values $V_0 < V_1 \ldots < V_{\gamma-1}$, i.e. OneMax is used as an auxiliary function. Species with both highest fitness and auxiliary values can be viewed as super-elite or α^*.

In the next section we use the notation α, α^* to denote the set of elite or super-elite species, an element of that set and the size of it. This is done to reduce notational clutter.

IV. ELITISM LEVELS TRAVERSE MECHANISM FOR UPPER BOUNDS

In this section we present the main result of the article on a general function that is later confirmed by further application to OneMax Test Function. We are interested in the upper bounds on optimization time (for explanation of Landau notation see e.g. Chapter 9 in [16]).

The working of the Elitism Levels Traverse Mechanism can be illustrated by an example from immunology.

There exists a population of species size N, which is susceptible to M types of infection, which are mutually exclusive, i.e. a species cannot be infected by more than one infection. The size of each set of infected species cannot be larger than m_j. We denote E_j^* an event that there are $1 \leq r \leq m_j$ infected species of type j, of which exactly one spawns an infected offspring that destroys a healthy member of the population. Since the sets of infected species are mutually exclusive, by additivity we obtain the probability that any of the infected species adds exactly one infected offspring:

$$P\left(\bigcup_{j=1}^{M} E_j^*\right) = P\left(\bigcup_{j=1}^{M} \bigcup_{r=1}^{m_j} E_{jr}^*\right) = \sum_{j=1}^{M} \sum_{r=1}^{m_j} P(E_{jr}^*) = \sum_{j=1}^{M} P(E_j^*)$$

This expression is quite complicated for a number of reasons, e.g. the knowledge of m_j. Although we can find bounds on the partial sum of rows of Pascal triangle, it is guaranteed to make the derivation quite messy. Therefore we need to lower-bound this probability. We do this by considering only one infected species of each type rather than r and the event of spawning exactly one infected species by E_j. This gives us the lower bound on the total probability of adding exactly one infected offspring, which is proven in Appendix A:

$$P(E_j^*) \geq P(E_j) \leftrightarrow \sum_{j=1}^{M} P(E_j^*) \geq \sum_{j=1}^{M} P(E_j) \qquad (1)$$

In the notation of EA, $N = \frac{\lambda}{2}$, the number of pairs of parents in the recombination pool with parents that are able to produce exactly one elite offspring. $\delta\mu$ (for $0 < \delta < 1$) is the number of elite individuals in the population that, once it is reached, the probability to generate an offspring with higher fitness is arbitrarily close to one, i.e. $1-o(1)$. We also have $n-1$ levels of fitness. Combining this with the upper bound on the probability of adding elite offsprings to the population, we obtain the upper bound (worst-case) on the optimization time of the algorithm:

$$\mathbf{E}\tau = O\left(\sum_{k=1}^{n} \sum_{\alpha=1}^{\delta\mu} \frac{1}{\sum_{j=1}^{M} P(E_j(\alpha,k))} \right) \qquad (2)$$

Derivation of the upper bound from Equation 2 is rather versatile. We need to identify pairs of possible parents $< p_1, p_2 >$ such that there exists some probability of swapping bits between parents $\varphi_j(k) > 0$ that as a results of applying a genetic operator to this pair either a new α species evolves from lower-ranked ones or an existing α is preserved after the recombination.

Intuitively, for the functions with plateaus both the population size and the number of elite species are more important than for those without plateaus. In the remainder of this section we show that the probability to add a super-elite offspring when solving a function with plateaus is less than the probability to add an elite offspring when solving functions without plateaus.

For the rest of this section we denote f_1 function without plateaus and f_2 function with plateaus. What we show is that $P(E_j^{f_1}(\alpha,k)) \geq P(E_j^{f_2}(\alpha,k))$.

A. Functions without plateaus

For this type of unimodal functions (e.g. OneMax) intuitively it is easier to add an elite offspring and thus reduce the optimization time, but we need to show it rigorously.

The probability to select a pair with an α-parent can be bounded by

$$P_{sel}^{f_1} \geq \frac{\alpha}{\mu} \xi_1$$

where ξ_1 is the probability to select a non-elite species to be paired with the elite one. Also bound the probability to flip the bits $\varphi_j(k) \geq \eta \ \forall \ j, k$. So the probability of an event E_j that includes pairs with elite species is

$$P(E_j^{f_1}) \geq \binom{\frac{\lambda}{2}}{1} \frac{\alpha}{\mu} \eta \xi_1 = \frac{\lambda\alpha\eta\xi_1}{2\mu}$$

The probability to select a pair without the the currently-elite species is lower-bounded by $\frac{\rho_1}{\mu}$. By breaking down the set of parents M_{f_1} in the recombination pool into those including $\alpha-$ parents, $M_{f_1}^*$

and those that do not, $M_{f_1}^{**}$, we can find the lower bound on the probability of adding another elite species:

$$P(S_{\alpha,k}^{f_1}) \geq \frac{M_{f_1}^* \lambda\alpha\xi_1\eta}{2\mu} + \frac{M_{f_1}^{**}\lambda\eta\rho_1}{2\mu}$$

B. Functions with plateaus

As noted in [10], algorithms with well-chosen population size perform similar to, and best individuals evolve along the same path as $(1+1)$EA. The difference between $(\mu + \lambda)$ and $(1 + 1)$ lies in the cost of traversing plateau. For this type of functions the length of plateau $\gamma > 1$. So we have K plateaus w.l.o.g. of the same length $\gamma \in \mathbb{Z}^+$, and $n = \gamma K$.

Also we assume that at the start of the algorithm each 'bin' (plateau) starts with an equal number of 1's and 0's uniformly distributed, therefore fitness of the best species at the beginning of the run is 0. To track progress between jumps in fitness values we use OneMax as an auxiliary function (roughly along the lines of using potential or distance functions, see e.g. [7]) that sums bits in the plateau.

The tricky part in this analysis is that the selection is based on fitness of the string rather than auxiliary function, but the progress towards the next level of fitness plateau depends on the number of parents with highest auxiliary value, V_s rather than $f(s)$. By denoting the subset of α with highest auxiliary function α^*, we notice that $f(\alpha) = f(\alpha^*)$, and $V_\alpha < V_{\alpha^*}$. Also trivially $\alpha^* \leq \alpha$ (for the case of functions without plateaus these functions are identical and last two expressions are equalities).

As shown before, for a unimodal function without plateaus regardless of fitness function, the probability that one of the parents is elite is $\frac{\alpha}{\mu}$, since if two elite species are selected for breeding, parent is chosen randomly. Obviously $\frac{\alpha^*}{\mu} \leq \frac{\alpha}{\mu}$. Additionally,

$$\frac{\alpha^*}{\mu} \cdot \frac{(\alpha - \alpha^*)}{\mu} \cdot \frac{1}{2} \leq \frac{\alpha^*}{\mu} \leq \frac{\alpha}{\mu}$$

Obviously, unlike f_1, for the evolution process on f_2 only a small subset of parents are of use, these having the highest and next-highest auxiliary values. Therefore pairs that do not include at least 1 of these parents can't add an α^* offspring. Similar to f_1, $M_{f_2} = M_{f_2}^* + M_{f_2}^{**}$ and clearly $M_{f_2}^* \leq M_{f_1}^*$ and $M_{f_2}^{**} \leq M_{f_1}^{**}$.

Along the lines of arguments in the previous subsection, $\exists \xi_2$ s.t. probability to select a non super-elite parent in addition to the super-elite one is upper-bounded by it. We get:

$$P_{sel}^{f_2} \leq \frac{\alpha^*(\alpha - \alpha^*)\xi_2}{2\mu^2}$$

so the probability of an event that an α^* parent is added to a pool and a new α^* offspring evolves is upper-bounded by

$$P(E_j^{f_2}) \leq \frac{M_{f_2}^* \lambda\alpha^*(\alpha - \alpha^*)\xi_2\eta}{4\mu^2}$$

where η is the lower bound on the probability of swapping bits. Therefore the probability to add one more α^* species to the population is

$$P(S_{\alpha,k}^{f_2}) \leq \frac{M_{f_2}^* \lambda\alpha^*(\alpha - \alpha^*)\xi_2\eta}{4\mu^2} + \frac{M_{f_2}^{**}\lambda\eta\rho_2}{2\mu}$$

Combining the inequalities above, and taking $M_{f_1}^{**} \geq M_{f_2}^{**} + \frac{\epsilon}{\rho_1}$, $0 < \epsilon < 1$, we compare the values in the first and second fractions

in the expressions for $P(S^{f_1}_{\alpha,k})$ and $P(S^{f_2}_{\alpha,k})$. It is easy to see that $\exists$ two constants $\psi_2 < \psi_1$ s.t.

$$P(S^{f_2}_{\alpha,k}) \leq \psi_2 < \psi_1 \leq P(S^{f_1}_{\alpha,k}) \tag{3}$$

V. Upper Bound on Runtime of $(\mu + \lambda)\mathrm{EA}_{1BS}$ on OneMax test function Using Elitism Levels Traverse Mechanism

In this section we present our findings on the upper bounds on runtime of $(\mu + \lambda)$EA with 1-Bit-Swap operator optimizing OneMax function using the Elitism Levels Traverse Mechanism. We distinguish four pairs of parents that make possible evolution of currently-elite species:

$$
\begin{aligned}
E_1 &:< \alpha, \beta_1 > \\
E_2 &:< \beta_1, \beta_1 > \\
E_3 &:< \alpha, \beta_{-1} > \\
E_4 &:< \beta_1, \beta_{-1} >
\end{aligned}
$$

We do not consider the obvious pair $< \alpha, \alpha >$ as it either adds two elite offsprings, of generates an offspring with higher fitness, something we do not use in the Mechanism.

For the upper bound on optimization time we only consider increase of the number of elite species by at most one. Increase by two or more is ignored, or otherwise transformed into any of the lower-ranked species. Similar approach was used in [10] in bounding the takeover time.

A. Simple upper bound

Of these four cases we start analysis with the first two. Main reason is that the other two cases involve cubic function, which becomes quite complicated to solve (see next subsection). For the cases E_1, E_2 we get the following probabilities of success:

$$
\begin{aligned}
P(E_1) &= 2 \binom{\frac{\lambda}{2}}{1} \varphi_1(k) \cdot \frac{\alpha}{\mu} \cdot \frac{\beta_1}{\mu} \left(1 - \frac{\alpha}{\mu}\right) \\
&= \frac{\alpha \beta_1 \lambda(\mu - \alpha)\varphi_1(k)}{\mu^3}
\end{aligned}
$$

$$
P(E_2) = \binom{\frac{\lambda}{2}}{1} \varphi_2(k) \left(\frac{\beta_1}{\mu}\left(1 - \frac{\alpha}{\mu}\right)\right)^2 = \frac{\lambda \varphi_2(k) \beta_1^2 (\mu - \alpha)^2}{2\mu^4}
$$

The probability of at least 1 of these events is

$$
\begin{aligned}
P(S_{\alpha,k}) &\geq P(E_1) + P(E_2) = \frac{2\varphi_1(k)\alpha\beta_1\lambda(\mu - \alpha)}{\mu^3} \\
&+ \frac{\varphi_2(k)\lambda\beta_1^2\varphi_1(k)(\mu - \alpha)^2}{2\mu^4}
\end{aligned}
$$

and, since $P(S)$ is minimal, the upper bound on expected time to traverse levels of elitism large enough to get a $1 - o(1)$ probability of evolution is

$$\mathbf{E}\tilde{T}_{\alpha,k} \leq \sum_{\alpha=1}^{\delta\mu} \frac{1}{P(S_\alpha)} \tag{4}$$

The expression for the expected first hitting time we obtain as a result of this setup is

$$\mathbf{E}\tilde{T}_{\alpha,k} \leq 2\mu^4 \sum_{\alpha=1}^{\delta\mu} \frac{1}{\beta_1\lambda(\mu - \alpha)(2\alpha\mu\varphi_1(k) - \alpha\beta_1\varphi_2(k) + \beta_1\mu\varphi_2(k))}$$

$$= \frac{2\mu^4}{\lambda}.$$

$$\sum_{\alpha=1}^{\delta\mu} \frac{1}{(\varphi_2(k) - 2\mu\varphi_1(k))\alpha^2 + (2\mu^2\varphi_1(k) - 2\mu\varphi_2(k))\alpha + \mu^2\varphi_2(k)}$$

$$= \frac{2\mu^4}{\lambda(\varphi_2(k) - 2\mu\varphi_1(k))} \sum_{\alpha=1}^{\delta\mu} \frac{1}{\alpha^2 + b_1\alpha + b_0} \tag{5}$$

where

$$
\begin{aligned}
b_0 &= \frac{\mu^2\varphi_2(k)}{\varphi_2(k) - 2\mu\varphi_1(k)} \\
b_1 &= \frac{2\mu(\mu\varphi_1(k) - \varphi_2(k))}{\varphi_2(k) - 2\mu\varphi_1(k)}
\end{aligned}
$$

At this point we set β_1 pessimistically to 1 to simplify the derivation. This a quadratic equation in α. The full solution to Equation 5 is in Appendix B.

The optimization time is

$$\mathbf{E}\tau_{(\mu+\lambda)EA_{1BS}} = O(\mu^{1+\varepsilon_2}n\log n) \tag{6}$$

for some constant $\varepsilon_2(\mu)$. For the second option of $\delta = \frac{c}{\mu}$ the upper bound becomes

$$\mathbf{E}\tau_{(\mu+\lambda)EA_{1BS}} = O(\mu n\log n) \tag{7}$$

or, in the number of function evaluations

$$\mathbf{E}\tau_{(\mu+\lambda)EA_{1BS}} = O(\lambda\mu n\log n) \tag{8}$$

B. Refined upper bound

We add the other two cases to obtain a sharper upper bound on optimization time, we set $\beta_1 = 1$:

$$
\begin{aligned}
P(E_3) &= 2 \binom{\frac{\lambda}{2}}{1} \frac{\alpha}{\mu} \cdot \left(1 - \frac{\alpha + \beta_1}{\mu}\right)^2 \varphi_3(k) \\
&\approx \frac{\lambda\alpha}{\mu} \left(1 - \frac{\alpha}{\mu}\right)^2 \varphi_3(k) \\
P(E_4) &= 2 \binom{\frac{\lambda}{2}}{1} \frac{\beta_1}{\mu} \left(1 - \frac{\alpha}{\mu}\right) \left(1 - \frac{\alpha + \beta_1}{\mu}\right)^2 \varphi_4(k) \\
&\approx \frac{\lambda}{\mu} \left(1 - \frac{\alpha}{\mu}\right)^3 \varphi_4(k)
\end{aligned}
$$

The probability to evolve one more elite species is $(P(E_1), P(E_2)$ are the same as in the previous derivation):

$$P(S_{\alpha,k}) = P(E_1) + P(E_2) + P(E_3) + P(E_4)$$

and the expected time until there are $\delta\mu$ elite strings in the population:

$$
\begin{aligned}
\mathbf{E}\tilde{T}_{\mu,k} &\leq \sum_{\alpha=1}^{\delta\mu} \frac{1}{S_{\alpha,k}} \\
&= 2\mu^4 \sum_{\alpha=1}^{\delta\mu} \frac{1}{b_3\alpha^3 + b_2\alpha^2 + b_1\alpha + b_0} \\
&= \frac{2\mu^4}{b_3} \sum_{\alpha=1}^{\delta\mu} \frac{1}{\alpha^3 + \frac{b_2}{b_3}\alpha^2 + \frac{b_1}{b_3}\alpha + \frac{b_0}{b_3}} \tag{9}
\end{aligned}
$$

where

$$b_0 = \lambda\mu^2(2\mu\varphi_4(k) + \varphi_2(k))$$
$$b_1 = 2\lambda\mu(\mu\varphi_1(k) + \mu^2\varphi_3(k) - 3\mu\varphi_4(k) - \varphi_2(k))$$
$$b_2 = \lambda(\varphi_2(k) - 4\mu^2\varphi_3(k) - 2\mu\varphi_1(k) - 6\mu\varphi_4(k))$$
$$b_3 = 2\lambda(\mu\varphi_3(k) - \varphi_4(k))$$

Full solution of Equation 9 is in Appendix C.

The upper bound on expected optimization time is (for $\delta \neq \frac{c}{\mu}$, c is a constant):

$$\mathbf{E}\tau_{(\mu+\lambda)EA_{1BS}} = \frac{c\mu^{1+\epsilon}n\log n}{\lambda} - O\left(\frac{\mu^{1+\epsilon}n}{\lambda}\right) \qquad (10)$$

for $0 < \epsilon(\mu) < 1$ and if $\delta = \frac{c}{\mu}$, in the number of function evaluations:

$$\mathbf{E}\tau_{(\mu+\lambda)EA_{1BS}} = c\mu n\log n - O(\mu n) \qquad (11)$$

This bound is sharper than the one obtained using simpler arguments earlier in this paper up to the order λ (since more possibilities of adding elite species are considered). It is also comparable to the results in [6], [7], [10] (see below). Such a result likely means that population has positive effect for some relatively small μ, but as it keeps increasing it either levels out (at best) or starts to degrade performance.

C. Generations vs Function evaluations

Tournament selection has a property that you do not need to evaluate every species, but we need to make 2λ evaluations (since two species compete for 1 slot in the recombination pool, so the number of evaluations each generation is $O(\lambda)$. Therefore, in terms of the number of functions evaluations the rough bound becomes $O(\mu\lambda n\log n)$ and the refined one $O(\mu n\log n)$. If $\mu = \lambda = O(1)$ this reduces to the well-known result of $O(n\log n)$ for OneMax function. The λ term in the denominator means that for the algorithm run on parallel computers the increase in the recombination pool size improves the performance.

D. Comparison to earlier results

The closest comparison we can draw is to $(N+N)$EA with mutation and tournament selection function in [10], $O(nN\log N + n\log n)$ if measured in the number of function evaluations (Proposition 4). By setting $N = O(1) = c \geq 1$ this bound becomes $n\log n + O(n)$, which is larger than just $O(n\log n)$. If instead we set $\mu = N = O(\sqrt{\log n})$ or $O(\frac{\log n}{\log\log n})$ the result in [10] is sharper than in this paper. For populations $\Omega(\frac{\sqrt{n}}{\log n})$ though the bound in this article becomes sharper again, e.g., for $\mu = N = O(\sqrt{n})$ it is $cn^{\frac{3}{2}}\log n - n^{\frac{3}{2}}$, and in [10] it is $\frac{n^{\frac{3}{2}}\log n}{2} + O(n\log n)$.

VI. DISCUSSION

We presented a new tool to analyze population-based elitist EAs, Elitism Levels Traverse Mechanism, which we used to derive a new upper bound on $(\mu + \lambda)$ EAs with a recombination operator and a variant of tournament selection solving OneMax problem.

We derived and proved the lower bound on the probability of evolving exactly 1 new currently-elite species, which helped us obtain the upper bound on the expected optimization time.

We showed that for a function with fitness plateaus it is harder to add a super-elite offspring to the population than an elite offspring

for a function without plateaus. This means that the very number of super-elite species in the population is more important in the former case than the number of elite species in the latter.

It may seem from the derived equations that population generally degrades performance (since μ is in the numerator), but for small size of population, when the cost of functions evaluations is not much different from 1, population brings about some positive effect.

As it keeps increasing, the effect levels out, at the same time the costs of evaluating functions grows and population loses its benefit. For other algorithms, s.a. RLS the effect even of small-sized population is usually negative, which makes EA+1BS (and, possibly, other recombination-based algorithms) stand out.

At the same time the recombination pool improves performance (at least when measured in terms of the number of generations), since λ is in the denominator. This means there is a benefit from increasing recombination pool size when the algorithm is run on parallel computers.

The Mechanism we have designed in this article proved to be quite efficient in deriving upper bounds for OneMax function and we are confident it can also yield tight upper bounds on other population-driven algorithms and more complicated problems.

VII. CONCLUSIONS AND FUTURE WORK

There are many reasons to use population in evolutionary computing rather than just $(1+1)$ or $(1,1)$ algorithms, that includes higher diversity and shorter evolutionary path (see [10]). We intend to expand the results in this article by considering the following extensions to the upper bound tool:

1) Analysis of functions with fitness plateaus. Apparently for functions with fitness plateaus (e.g. Royal roads) both large populations and large number of elite parents are crucial compared to functions without one, so we will extend our findings to these functions as well.
2) Typical runtime analysis. It is fairly obvious that the actual number of elite species grows every generation at some rate that realistically lies between the upper and lower bounds. We need to find an approximation on the expected number of α added to the population every generation and thus estimate the typical runtime.
3) Elitism rates analysis. In this article we never really considered the rate of elitism, i.e. the actual number of species saved in the population each generation, although numerical computation shows that it has a strong effect on the runtime. So far we only said that all the elite species are saved each generation, thus accumulating over time till $\delta\mu$. It would be interesting to compare elitism level 1 to 50%, i.e. if there is any difference if only 1 species is saved compared to half of the population.
4) Derivation of $\delta\mu$ to find the proportion of elite species that yields a high enough probability of evolution. Quite obviously it is different for functions with plateaus and without.
5) Derivation of the optimal population size. We will do this by comparing the number of functions evaluations necessary of $(1+1)$ and $(\mu + \lambda)$ algorithms.

REFERENCES

[1] A. Ter-Sarkisov, S. Marsland, and B. Holland, "The k-Bit-Swap: A New Genetic Algorithm Operator," in *Genetic and Evolutionary Computing Conference (GECCO) 2010*, 2010, pp. 815–816.

[2] S. Droste, T. Jansen, and I. Wegener, "On the analysis of the (1+1) evolutionary algorithm," *Theoretical Computer Science*, vol. 276, pp. 51–81, 2002.

[3] B. Doerr, D. Johannsen, and C. Winzen, "Multiplicative Drift Analysis," in *Genetic and Evolutionary Computing Conference (GECCO) 2010*, 2010, pp. 1449–1456.

[4] B. Doerr, M. Fouz, and C. Witt, "Sharp Bounds by Probability-Generating Functions and Variable Drift," in *Genetic and Evolutionary Computing Conference (GECCO) 2011*, 2011, pp. 2083–2090.

[5] C. Witt, "An Analysis of the $(\mu+1)$EA on Simple Pseudo-Boolean Functions," in *Genetic and Evolutionary Computing Conference (GECCO) 2004*, 2004, pp. 761–773.

[6] T. Jansen, K. A. De Jong, and I. Wegener, "On the Choice of the Offspring Population Size in Evolutionary Algorithm," *Evolutionary Computation*, vol. 13(4), pp. 413–440, 2005.

[7] J. He, "A Note on the First Hitting Time of $(1 + \lambda)$ Evolutionary Algorithm for Linear Functions with Boolean Inputs," in *Conference on Evolutionary Computation (CEC)*, 2010, pp. 1–6.

[8] J. He and X. Yao, "From an Individual to a Population: An Analysis of the First Hitting Time of Population-Based Evolutionary Algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 6-5, October 2002, pp. 495–511, 2002.

[9] ——, "A study of drift analysis for estimating computation time of evolutionary algorithm," *Natural Computing*, vol. 3(2004), pp. 21–35, 2004.

[10] 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 Transactions on Systems, Man, and Cybernetics*, vol. 39(5), pp. 1092–1106, 2009.

[11] B. Doerr, M. Fouz, and C. Witt, "Quasirandom Evolutionary Algorithm," in *Genetic and Evolutionary Computing Conference (GECCO) 2010*, 2010, pp. 1457–1464.

[12] A. Ter-Sarkisov and S. Marsland, "Convergence Properties of Two $(\mu + \lambda)$ Evolutionary Algorithms on OneMax and Royal Roads Test Functions," in *International Conference on Evolutionary Computation Theorey and Applications (ECTA)*, 2011, pp. 196–202.

[13] ——, "Convergence of a Recombination-Based Elitist Evolutionary Algorithm on the Royal Roads Test Function," in *24th Australasian Joint Conference on Artificial Intelligence*, 2011, pp. 361–371.

[14] I. Wegener, "Methods for the analysis of EAs on pseudo-boolean functions," *International Series in Operations Research and Management Science*, vol. 48, VII, pp. 349–369, 2002.

[15] T. Jansen and I. Wegener, "Evolutionary Algorithms-How to Cope With Plateaus of Constant Fitness and When to Reject Strings of The Same Fitness," *IEEE Transactions on Evolutionary Computation*, vol. 5(6), pp. 589–599, 2001.

[16] R. L. Graham, D. E. Knuth, and O. Patashnik, *Concrete Mathematics:A Foundation for Computer Science*. Addison-Wesley Publishing Company, 1995.

[17] M. Mitchell, *Introduction to Genetic Algorithm*. Kluwer Academic Publishers, 1996.

APPENDIX A
PROOF OF EQUATIONS 1-2

Main idea and logic of the lower bound on the probability of adding an elite offspring and the upper bound on runtime following from this is presented in Section IV. Here we present the derivation of this bound.

We prove this lower bound inequality for an arbitrary subset (it is not to be confused with at trivial one of the form

$\sum_{k \geq r} \binom{n}{k} p^k (1-p)^k > \binom{n}{r} p^r (1-p)^{n-r}$):

$$P(E_j) = \binom{\frac{\lambda}{2}}{1} P_{sel} P_{swap} = \binom{\frac{\lambda}{2}}{1} P_{sel} \varphi_j(k)$$

$$P(E_j^*) = P\left(\bigcup_{r=1}^{m_j} E_{jr}^* \right)$$

$$= \sum_{r=1}^{m_j} \binom{\frac{\lambda}{2}}{r} P_{sel}^r (1 - P_{sel})^{\frac{\lambda}{2}-r} \binom{r}{1} \varphi_j(k)(1-\varphi_j(k))^{r-1}$$

In this expression P_{swap} is not necessarily the probability to swap bits $< 0, 1 >$. It is the probability to swap bits such that an elite offspring evolves. Since all the terms in the sum are positive, we use the lower bound on this expression:

$$P(E_j^*) \geq \sum_{r=1}^{m_j} \binom{\frac{\lambda}{2}}{r} P_{sel}^r (1-P_{sel})^{\frac{\lambda}{2}-r} \varphi_j(k)(1-\varphi_j(k))^{r-1}$$

$$\geq \frac{\varphi_j(k)}{(1-\varphi_j(k))} \left(\sum_{r=0}^{m_j} \binom{\frac{\lambda}{2}}{r} P_{sel}^r (1-\varphi_j(k))^r (1-P_{sel})^{\frac{\lambda}{2}-r} \right.$$

$$\left. - (1-P_{sel})^{\frac{\lambda}{2}} \right)$$

$$\geq \varphi_j(k) \left(\sum_{r=0}^{m_j} \binom{\frac{\lambda}{2}}{r} P_{sel}^r (1-\varphi_j(k))^r (1-P_{sel})^{\frac{\lambda}{2}-r} - \right.$$

$$\left. (1-P_{sel})^{\frac{\lambda}{2}} \right)$$

Canceling out $\varphi_j(k)$ and moving the term $e^{-1} \leq (1-P_{sel})^{\frac{\lambda}{2}} \leq \frac{1}{\sqrt{e}} < 1$ on the other side, LHS of the inequality becomes

$$P(E_j^*) \geq \sum_{r=0}^{m_j} \binom{\frac{\lambda}{2}}{r} P_{sel}^r (1-\varphi_j(k))^r (1-P_{sel})^{\frac{\lambda}{2}-r}$$

$$\geq (P_{sel}(1-\varphi_j(k)) + 1 - P_{sel})^{\frac{\lambda}{2}}$$

$$= (1 - P_{sel}\varphi_j(k))^{\frac{\lambda}{2}}$$

and the RHS is upper-bounded by

$$\frac{1}{\sqrt{e}} + \frac{\lambda P_{sel}}{2} = \frac{1}{\sqrt{e}} + o(\lambda^{c-1}) \text{by the argument below}$$

LHS is lower-bounded by (using Bernoulli inequality for $\frac{\lambda}{2} \geq 1$):

$$P(E_j^*) \geq (1 - P_{sel}\varphi_j(k))^{\frac{\lambda}{2}} \geq 1 - \frac{\lambda P_{sel}\varphi_j(k)}{2}$$

Since we can select $P_{sel} = o(\lambda^c)$ and $\varphi_j(k) = O(\frac{1}{n^c}), c \in \mathbb{Z}$, the expression is

$$P(E_j^*) = 1 - o(1) > \frac{1}{\sqrt{e}} + o(1) = P(E_j) \qquad (12)$$

thus proving the upper bound on the probability of evolving 1 more elite species for an arbitrary subset. This logic applies for each of the M subsets (types of pairs) of the recombination pool, and the inequality becomes

$$P\left(\bigcup_{j=1}^{M} E_j^* \right) > \sum_{j=1}^{M} P(E_j) \qquad (13)$$

The upper bound in Equation 2 follows directly.

We have a quadratic equation

$$S(\mu, k) = \sum_{\alpha} P_2(\alpha) = \sum_{\alpha=1}^{\delta\mu} \frac{1}{\alpha^2 + b_1\alpha + b_0}$$

with

$$b_0 = \frac{\mu^2\varphi_2(k)}{\varphi_2(k) - 2\mu\varphi_1(k)}$$
$$b_1 = \frac{2\mu(\mu\varphi_1(k) - \varphi_2(k))}{\varphi_2(k) - 2\mu\varphi_1(k)}$$

In order to simplify the already complicated derivation, we want the expression above in the form

$$P_2(\alpha) = \frac{1}{(\alpha + r)^2} = \frac{1}{(\alpha^2 + 2r\alpha + r^2)}$$

for some r, not necessarily rational. From equating coefficients it becomes clear that

$$r = \sqrt{b_0} \vee r = \frac{b_1}{2}$$

and so, using the first root

$$S(\mu, k) = \sum_{\alpha=1}^{\delta\mu} \frac{1}{(\alpha + r)^2} = \psi_1(\sqrt{b_0}) - \psi_1(\sqrt{b_0} + \delta\mu + 1)$$

For large b_0 these expressions involving digamma function can be expanded asymptotically in Taylor series (we use only the first two terms):

$$S(\mu, k) \approx \left(\frac{1}{b_0} - \frac{1}{2b_0}\right) - \left(\frac{1}{b_0} - \frac{\delta\mu}{b_0} - \frac{1}{2b_0}\right) = \frac{\delta\mu}{b_0}$$
$$= \frac{\delta\mu(\varphi_2(k) - 2\mu\varphi_1(k))}{\mu^2\varphi_2(k)} = \frac{\delta(\varphi_2(k) - 2\mu\varphi_1(k))}{\mu\varphi_2(k)}$$

and therefore the expected time to traverse enough levels of elitism to improve 1 bit of the string is (plugging this expression into Equation 5)

$$\mathbf{E}\tilde{T}_{\mu,k} = \frac{2\mu^4}{\lambda(\varphi_2(k) - 2\mu\varphi_1(k))} \cdot \frac{\delta(\varphi_2(k) - 2\mu\varphi_1(k))}{\mu\varphi_2(k)}$$
$$= \frac{2\mu^3\delta}{\lambda\varphi_2(k)}$$

To improve the pair $< \beta_1, \beta_1 >$ we need to either swap 1 from the first parent and 0 from the second, or the other way around (any other outcome just keeps the current number of bits in each parent):

$$\varphi_2(k) = 2 \cdot \frac{k-1}{n} \cdot \frac{n-k+1}{n} = \frac{2(k-1)(n-k+1)}{n^2}$$

Plugging this into the expression for $\mathbf{E}\tilde{T}_k$, we obtain the expected optimization time of the algorithm, pessimistically assuming that at the beginning of the run the best species has only 2 1-bit and finishes at $n-2$, since if the fitness of $\beta_1 = n-1$ implies the fitness of $\alpha = n$.

$$\mathbf{E}\tau_{(\mu+\lambda)EA_{1BS}} \leq \frac{\mu^3 n^2 \delta}{\lambda} \sum_{k=2}^{n-2} \frac{1}{(k-1)(n-k+1)}$$
$$= \frac{\delta\mu^3 n^2}{\lambda} \cdot \frac{1}{n}\left(\sum_{k=2}^{n-2}\frac{1}{k-1} + \sum_{k=2}^{n-2}\frac{1}{n-k+1}\right)$$
$$= \frac{\delta\mu^3 n}{\lambda}\left(\log(n-1) + O(1)\right)$$

The second step is due to partial fraction expansion. Although this seems quite a loose bound given cubic in μ, we take $\mu = O(\lambda)$ so all we need to establish is δ to reduce the power.

Obviously $0 < \delta < 1$, but we need to select it s.t. summation over α makes sense. We set $\delta = \mu^{-\varepsilon_1}$ for an arbitrary $\varepsilon_1 > 0$ s.t. $\delta\mu = \mu^{1-\varepsilon_1} > 1$. Then $\delta\mu^2 = \mu^{2-\varepsilon_1} = \mu^{1+\varepsilon_2}$. For example, $\varepsilon = \frac{1}{2}$ yields $\delta\mu = \sqrt{\mu}$ and $\delta\mu^2 = \sqrt{\mu^3}$. Therefore, the upper bound on the expected convergence time is

$$\mathbf{E}\tau_{(\mu+\lambda)EA_{1BS}} = O(\mu^{1+\varepsilon_2}n\log n) \qquad (14)$$

In fact if (similar to [10]) we set $\delta = \frac{c}{\mu}$ for $c \in \mathbb{Z}^+$, we get $\delta\mu = c$ and $\delta\mu^2 = c\mu = O(\mu)$, so the expectation becomes linear in μ:

$$\mathbf{E}\tau_{(\mu+\lambda)EA_{1BS}} = O(\mu n\log n) \qquad (15)$$

We need a solution to the cubic equation of the form

$$S(\mu, k) = \sum_{\alpha} P_3(\alpha) = \sum_{\alpha=1}^{\delta\mu} \frac{1}{\alpha^3 + b_2'\alpha_2 + b_1'\alpha + b_0}$$

where

$$b_2' = \frac{\varphi_2(k) - 4\mu^2\varphi_3(k) - 2\mu\varphi_1(k) - 6\mu\varphi_4(k)}{2(\mu\varphi_3(k) - \varphi_4(k))}$$
$$b_1' = \frac{\mu(\mu\varphi_1(k) + \mu^2\varphi_3(k) - 3\mu\varphi_4(k) - \varphi_2(k))}{2(\mu\varphi_3(k) - \varphi_4(k))}$$
$$b_0' = \frac{\mu^2(2\mu\varphi_4(k) + \varphi_2(k))}{2(\mu\varphi_3(k) - \varphi_4(k))}$$

Solution to $S(\mu, k)$ is of the form

$$S(\mu, k) = \sum_{\alpha} \frac{1}{(\alpha + \rho)^3} = \sum_{\alpha} \frac{1}{\alpha^3 + 3\alpha^2\rho + 3\alpha\rho^2 + \rho^3}$$

Equating the coefficients we obtain three roots ρ:

$$\rho = \frac{b_2'}{3}$$
$$\rho = \pm\frac{\sqrt{b_1'}}{3}$$
$$\rho = \sqrt[3]{b_0'}$$

To simplify the increasingly hard notation, we select only the last root:

$$S(\mu, k) = \sum_{\alpha=1}^{\delta\mu} \frac{1}{(\alpha + \sqrt[3]{b_0'})^3} = \frac{\psi_2(\sqrt[3]{b_0'} + \delta\mu + 1) - \psi_2(\sqrt[3]{b_0'} + 1)}{2}$$
$$= \frac{1}{2}\left(\left(-\frac{1}{\sqrt[3]{b_0'}^2} + \frac{2\delta\mu + 1}{b_0'}\right) - \left(-\frac{1}{\sqrt[3]{b_0'}^2} + \frac{1}{b_0'}\right)\right)$$
$$= \frac{1}{2} \cdot \frac{2\delta\mu}{b_0'} = \frac{\delta\mu}{b_0'}$$

The second line in the derivation was obtained by expanding both second-order polygamma functions in Taylor series as $b_0' \to \infty$ and taking two first terms of each function. We now combine the front term in Equation 9 with this derivation to obtain the expression on

the upper bound on achieving the number of elite species in the population $\delta\mu$:

$$\mathbf{E}\tilde{T}_{\mu,k} \leq \frac{2\mu^5\delta}{\lambda b_3 b_0'} = \frac{2\delta\mu^3}{\lambda(2\mu\varphi_3(k) + \varphi_4(k))}$$

since

$$b_3 b_0' = 2(\mu\varphi_3(k) - \varphi_4(k)) \cdot \frac{\mu^2(2\mu\varphi_3(k) + \varphi_4(k))}{2(\mu\varphi_3(k) - \varphi_4(k))}$$
$$= \mu^2(2\mu\varphi_3(k) + \varphi_4(k))$$

We are now ready to find the upper bound on the expected optimization time of the algorithm:

$$\mathbf{E}\tau_{(\mu+\lambda)EA_{1BS}} \leq \sum_{k=3}^{n-3} \mathbf{E}\tilde{T}_{\mu,k} = \frac{2\delta\mu^3}{\lambda} \sum_{k=3}^{n-1} \frac{1}{2\mu\varphi_3(k) + \varphi_4(k)}$$
(16)

Here again we pessimistically assume that the best species at the start of the run has fitness 3, since in such case fitness of β_{-1} has minimal fitness of 1, otherwise we obtain inconsistencies s.a. $\frac{1}{0}$. We have two probabilities to consider for the two new types of pairs:

$$<\alpha, \beta_{-1}>: \varphi_3(k) = \frac{k}{n} \cdot \frac{k-2}{n} + \frac{n-k}{n} \cdot \frac{n-k+2}{n}$$
$$= \frac{k(k-2) + (n-k)(n-k+2)}{n^2}$$

We need to preserve the better parent in order to get it added to the population, so need to either select 1-bits in each parent or 0-bits in each parent. for the last swap probability, $\varphi_4(k)$, we need only to select a 0 in the β_1 parent and 1 in β_{-1} parent, other options either degrade the better parent or leave the current fitness.

$$<\beta_1, \beta_{-1}>: \varphi_4(k) = \frac{(n-k+1)(k-2)}{n^2}$$

We continue with manipulating with the summand over k:

$$S(n,k) = \frac{1}{2\mu\varphi_3(k) + \varphi_4(k)}$$
$$= \frac{n^2}{(4\mu-1)k^2 + (n - 8\mu - 4\mu n + 3)k + 4\mu n - 2n + 2\mu n^2 - 2}$$
$$\leq \frac{n^2}{\mu} \cdot \frac{1}{k^2 - 4(n+2)k + 2n(n+1)}$$

We leave out the first fraction, and factor the denominator in the form $(k-s)(k-r)$, s.t. s, r are solutions to the set of equations:

$$\begin{cases} s+r &= 4(n+2) \\ sr &= 2n(n+1) \end{cases}$$

The resulting solution (we only use one of the roots, which are symmetric) is:

$$\begin{cases} s &= 2n + \sqrt{2}\sqrt{n^2 + 7n + 8} + 4 \\ r &= 2n - \sqrt{2}\sqrt{n^2 + 7n + 8} + 4 \end{cases}$$

The value under the root can be bounded by

$$n + 2 \leq \sqrt{n^2 + 7n + 8} \leq n + 4$$

So the expression becomes upper-bounded by

$$\frac{1}{(k - (2n + \sqrt{2}(n+2)))(k - (2n - \sqrt{2}(n+4)))}$$

Expanding this in partial fractions, we obtain

$$\frac{1}{2\sqrt{2}(n+3)} \cdot \left(\frac{1}{k - (2n + \sqrt{2}(n+2))} \right.$$
$$\left. - \frac{1}{k - (2n - \sqrt{2}(n+2))} \right)$$

We obtain two sums over k:

$$S_1(n,k) = \sum_{k=3}^{n-3} \frac{1}{k - (2n + \sqrt{2}(n+2))}$$
$$\approx \psi_0(n - 2n - \sqrt{2}n) - \psi_0(3 - 2n - \sqrt{2}n)$$
$$= \psi_0(-(1-\sqrt{2})n) - \psi_0(3 - (2+\sqrt{2})n)$$
$$= O(1) - O(1) = -O(1)$$

The result of $-O(1)$ is due to the fact that we can select any n, for which the values of digamma function are small negative constants (see Appendix D for details on Taylor series expansion of $\psi_0(n)$ for $n < 0$). For the second sum, we notice the upper bound on the value in the denominator, since $2 - \sqrt{2} \approx 0.58 < 1$:

$$S_2(n,k) = \sum_{k=3}^{n-3} \frac{1}{k - (2n - \sqrt{2}(n+2))}$$
$$\leq \sum_{k=3}^{n-3} \frac{1}{k-n} = -\sum_{k=3}^{n-3} \frac{1}{n-k} \approx -\log(n-3) + O(1)$$

the minus sign in front of the expression cancels out and we obtain the upper bound for $S(\mu, n)$:

$$S(\mu, n) \leq \frac{n^2(\log(n-3) - O(1)}{\mu n}$$

and the upper bound on the expected first hitting time:

$$\mathbf{E}\tau_{(\mu+\lambda)EA_{1BS}} \leq \frac{2\delta\mu^2 n(\log(n-3) - O(1))}{\lambda}$$
(17)

with $\delta = \frac{c}{\mu}$ the expression becomes (measured in the number of generations, for $c > 0$)

$$\mathbf{E}\tau_{(\mu+\lambda)EA_{1BS}} = \frac{c\mu n \log n}{\lambda} - O\left(\frac{\mu n}{\lambda}\right)$$
(18)

or, in the number of function evaluations,

$$\mathbf{E}\tau_{(\mu+\lambda)EA_{1BS}} = c\mu n \log n - O(\mu n)$$
(19)

APPENDIX D
MATHEMATICAL EXPRESSIONS

There is a number of important mathematical expression used throughout the article, we present some of them here:

$$H(n) = \sum_{k=0}^{n-1} \frac{1}{n-k} = \sum_{k=0}^{n-1} \frac{1}{k} \approx \int_0^{n-1} \frac{dx}{n-x} = \log n < \log n + 1$$

Digamma function:

$$\psi_0(n) = \log n + O\left(\frac{1}{n}\right)$$

For $\psi_0(n)$ with $n \to -\infty$ we use the largest term of Taylor series for asymptotic expansion:

$$\psi_0(n) \approx \pi \cot(\pi n) + O\left(\frac{1}{n}\right)$$

The values for $\cot(\pi n)$ for integer n, such as in this article, are infinity. Therefore for expressions for $S_1(n, k)$ and $S_2(n, k)$ we have selected some constants, e.g. $-(1 - \sqrt{2}), 2 - \sqrt{2}$, s.t. the resulting values are constants. Since n is arbitrarily large, we can find such n that the difference between them is negative, hence we obtain term $-O(1)$.