International Journal of Automation and Computing 04(3), July 2007, 281-293
DOI: 10.1007/s11633-007-0281-3

Time Complexity of Evolutionary Algorithms for Combinatorial Optimization: A Decade of Results

Pietro S. Oliveto Jun He* Xin Yao

The Center of Excellence for Research in Computational Intelligence and Applications (CERCIA), School of Computer Science, University of Birmingham, Edgbaston, Birmingham B15 2TT, UK

Abstract: Computational time complexity analyzes of evolutionary algorithms (EAs) have been performed since the mid-nineties. The first results were related to very simple algorithms, such as the (1+1)-EA, on toy problems. These efforts produced a deeper understanding of how EAs perform on different kinds of fitness landscapes and general mathematical tools that may be extended to the analysis of more complicated EAs on more realistic problems. In fact, in recent years, it has been possible to analyze the (1+1)-EA on combinatorial optimization problems with practical applications and more realistic population-based EAs on structured toy problems. This paper presents a survey of the results obtained in the last decade along these two research lines. The most common mathematical techniques are introduced, the basic ideas behind them are discussed and their elective applications are highlighted. Solved problems that were still open are enumerated as are those still awaiting for a solution. New questions and problems arisen in the meantime are also considered.

Keywords: Evolutionary algorithms, computational complexity, combinatorial optimization, evolutionary computation theory.

1 Introduction

Evolutionary algorithms (EAs) are a wide class of randomized problem solvers based on principles of biological evolution. They have been used, often successfully, in many computational areas such as Optimization, Learning, Adaptation and others[1]. Hence, there are lots of available experimental results concerning this class of algorithms, but compared to that amount, the theoretical knowledge of how they perform lags way behind. Indeed, the largest part of the available research studies are of empirical nature. However, since the eighties when EAs started to become popular, there have always been theoretical studies of this class of algorithms, albeit few.

Early studies were concerned with explaining the behavior of EAs rather than analyzing their performance. The Schema theory[2] is probably the most popular tool used in these early attempts. In particular it was first proposed to analyze the behavior of the simple genetic algorithm (sGA)[3]. As Eiben and Rudolph stated when analyzing these initial tools[4], although the schema theory was considered fundamental for understanding GAs up to the early nineties, it "cannot explain the dynamical or limit behavior of EAs".

In the nineties, with the advent of Markov Chain theory in the analysis of EAs, the first convergence results appeared related to their time-limit behavior for optimization problems. Since the state transitions of an EA are of probabilistic nature, the deterministic definition of convergence was obviously not appropriate. Hence definitions of stochastic convergence had to be used[5]. Ideally the EA should be able to find the solution to the problem it is tackling with probability 1 after a finite number of steps, regardless of its initialization. In such a case the algorithm is said to visit the global optimum in finite time. If the algorithm holds the solution in the population forever after, then it is said to converge to the optimum. Using Markov Chains, Rudolph proved that canonical GAs using mutation, crossover and proportional selection do not converge to the global optimum, while elitist variants do[6]. Then he extended his analysis by defining general conditions that, if satisfied by an EA, guarantee its convergence[7]. The work was motivated by the fact that it was "not necessary to build a quantitatively exact Markov model for each variant of an evolutionary algorithm in order to investigate the limit behavior". Conditions for non-convergence were also given. Hence, only when an EA does not satisfy the given conditions is a specific analysis necessary. Another novelty was that the intensive work concerning the convergence of EAs had finally "led to simple proofs which do not require Markov chain theory anymore".

However, if an algorithm does converge, the analysis of the time limit behavior does not give any hints about the expected time for the solution to be found, far less any precise statement. Aytug and Kohler performed an analysis of the number of generations which are sufficient to guarantee convergence with a fixed level of confidence (i.e. under a certain probability δ) independent of the optimization function[8]. The obtained results have been improved by Greenhalgh and Marshall[9]. Nevertheless, the best upper bound the analysis can guarantee is the same as that of a random algorithm (RA) choosing random individuals independently in each generation[10]. The failure of these approaches, to obtain useful bounds, confirmed the

Manuscript received date March 5, 2007; revised date May 31, 2007
This work was supported by an EPSRC grant (No. EP/C520696/1).
*Corresponding author. E-mail address: J.He@cs.bham.ac.uk

idea that when analyzing the time complexity of problem-independent search heuristics the function to be optimized needs to be considered. Further confirmation of such an idea came when Droste proved the existence of functions for which the (1+1)-EA evolution strategy (ES) finds the global optimum in time $\Theta(n^n)$[11]. Hence, the need of measuring the performance of EAs on specific problems became evident in the late nineties. In an overview of the field of Evolutionary Computation in 1995, Yao states that to his best knowledge a 1991 paper by Hart and Belew is the only paper on the topic[12]. In particular, it appeared necessary to use "standard" measures such as a relationship between the size of the problem being tackled and the expected time needed for the solution to be found. Beyer confirm stating that "there were almost no results, before the mid-nineties, estimating the worst-case expected optimization time of an evolutionary algorithm working on some problem or estimating the probability of obtaining within t(n) steps a solution fulfilling a minimum demand of quality"[13].

As a consequence, an attempt was finally made towards a systematical time complexity analysis that turns the theory of EAs into "a legal part of the theory of efficient algorithms"[13].

Due to their stochastic nature, the time complexity analysis of EAs is not an easy task. The first attempts are related to basic EAs (such as the (1+1)-EA) on simple functions with some structural properties. The goal of this approach was that of understanding what properties of a function make its optimization easy or hard for an EA. Choosing simple pseudo-Boolean problems with "nice structures" it was possible to[14]:

1. describe the behaviour of an EA on typical issues of functions;
2. show some extreme behaviours of EAs;
3. disprove widely accepted conjectures;
4. compare different variants of EAs.

Following the above motivations, in 1998 Droste, Jansen and Wegener, analyzed the (1+1)-EA on pseudo-Boolean functions such as OneMax and bin[15]. Then, they extended their results to linear functions by proving an upper bound of $\Theta(n \log n)$[11]. In the same year Droste also analyzed, again, the (1+1)-EA on unimodal functions disproving the widely spread conjecture that EAs are efficient on all such functions by producing an example for which the algorithm takes exponential expected time[16]. Wegener and Witt, in the the year 2000 proved that there are also quadratic functions for which the expected optimization time of the (1+1)-EA is exponential in the problem size n. The same year, Jansen and Wegener analysed the algorithm on plateaus of constant fitness values (i.e. short path problems(SPPs)) and proved that the (1+1)-EA can efficiently optimize such functions in time $O(n^3)$[17]. Furthermore they showed how a (1+1)-EA, not accepting individuals with the same fitness values, is not efficient on plateaus.

A further objective of these studies was that of obtaining mathematical methods and tools that may prove to be useful in the analysis of general EAs on more sophisticated and realistic problems. This was necessary because it soon turned out the use of Markov chains on their own was not sufficient for the analysis. Some tools had to be imported from the general field analyzing randomized algorithms[18] and others were especially defined. An overview of the most common tools used for the analysis of the described problems can be found in [14].

As a consequence of these preliminary efforts, nowadays it is possible to analyze the (1+1)-EA on combinatorial optimization problems with "practical" applications. Furthermore some results about more realistic EAs having populations and crossover have also appeared. Such works have become possible by using tools obtained in previous analyzes directly, or by extending them to the new tackled problems. However, in some cases, brand new tools seem to be necessary.

A detailed overview of the first runtime results available up to the late nineties can be found in [7]. Various open questions in the nineties can also be found in [12]. Some of the questions have been answered in the mean time. On the other hand some are still open and new questions have arisen. The aim of this paper is to extend the above reviews by covering another decade in the time complexity analysis of EAs. The most important achievements will be given together with the ideas laying behind the most common mathematical tools used in the analyzes. The reader may obtain a wider picture of how such tools are used by studying the quoted references.

Without claiming to be complete we restrict our attention to combinatorial optimization problems with single objectives. Although such a restriction cuts out a lot of related theoretical work, it helps in keeping focus on one general topic without risking too much dispersion.

The paper is structured as follows. In section 2 the algorithms considered in the paper are introduced. Section 3 describes the ideas behind the most widely used tools in the runtime analysis of EAs. Section 4 is a survey of the achievements that have been obtained in approximately a decade of of computational complexity analyzes of EAs. The section emphasizes the improvements since the late nineties results of the (1+1)-EA for toy problems. The paper ends with some conclusions and directions for further research.

2 Evolutionary algorithms

There are several different types of EAs. The three most popular branches are probably genetic algorithms (GAs), evolutionary programming (EP) and evolution strategies (ESs)[5]. Each type has numerous variants due to different parameter settings and implementations. In this section the EAs considered in this paper are presented.

Sometimes, in empirical work comparing different search heuristics, it seems that an attempt is being made towards

deciding which algorithm class is generally better than the other. "The answer to the question which EA is the best, is problem dependent. There is no universally best algorithm which can achieve the best result for all problems. One of the challenges to researchers is to identify and characterize which EA is best suited for which problem"[12].

A general EA derived from ESs is the $(\mu + \lambda)$-EA, that creates λ new individuals in each generation and chooses the best μ from the offspring and the parent population to survive for the next generation. The standard algorithm, with candidate solutions represented as bit-strings, works as follows:

- Initialization: choose μ individuals $x \in \{0,1\}^n$ randomly as the initial population;
- Mutation: create λ new individuals. To create each one choose x randomly from the parent population and flip each bit in x with probability p;
- Selection: create the new population by choosing the best μ individuals out of the $\mu + \lambda$;
- Repeat the last two steps until a stopping criterion is fulfilled.

The most simple and theoretically analyzed EA is the (1+1)-EA which is obtained from the above algorithm with $\mu = \lambda = 1$. Sometimes a distinction is made between a purely elitist (1+1)-EA and a (1+1)-EA accepting equal valued individuals. The former algorithm accepts only individuals with strictly higher fitness values. The latter, instead, also accepts individuals with the same fitness as its predecessor. The mutation probability of $p = 1/n$ is generally considered the best choice[19]. A similar algorithm that just flips one individual per generation is called a random local search (RLS) algorithm.

Common population-based evolution strategies are the (μ+1)-EA and the (1+λ)-EA. The first has a parent population of μ individuals and generates a new individual per generation. The best μ individuals are chosen to survive. Evolutionary algorithms that create one new individual per generation are called steady state EAs. The second algorithm creates an offspring population of λ individuals but only the best, out of the λ+1, survives. A variant of this algorithm is the (1, λ)-EA where the new individual is chosen only from the offspring population. A (1, 1)-EA does not make much sense as no selection mechanism is applied at all, hence a random walk on the fitness landscape is obtained.

Algorithms which are more similar to those used in practice are the $(\mu+\lambda)$-EA and the (μ, λ)-EA or the (N+N)-EA and the (N, N)-EA. It is not uncommon to use diversity mechanisms with population-based EAs, to avoid premature convergence.

Other population-based EAs may use different selection schemes, based on probability measures such as fitness proportional selection or tournament selection[2]. If a crossover operator is also applied to the parent population before the mutation operator, then a genetic algorithm (GA) is obtained. Popular crossover operators are one point crossover an uniform crossover.

The one point crossover operator chooses a point i randomly between 1 and the string length n. The bits from positions i to n of the two bit-strings are swapped.

Given two individuals of the population x and x', the uniform crossover operator creates a new individual y by setting each bit $y_i = x_i$ with probability 1/2, and otherwise it sets the bit to $y_i = x'_i$.

The algorithms described in this section are those that are considered in the paper. Other branches of evolutionary computation such as genetic programming (GP) and immunological algorithms (IAs)[20] are not covered because, to our best knowledge, no time complexity results are available yet. Other meta-heuristics such as ant colony optimization (ACO) or particle swarm optimization (PSO) are beyond the scope of this paper.

3 Mathematical tools

In general, if X_f is the random variable measuring the time for the solution to be found by an EA for a certain function f, then the run time analysis of a randomized algorithm consists of[14]:

1. The estimation of $E(X_f)$ in the best, the average and the worst case;
2. The calculation of the success probability $Pr(X_f \leq t)$ in the best, the average and the worst case.

Since EAs are stochastic processes with each state usually depending only on the value of the previous one, the most straightforward and common method used to model an EA is Markov chains. However it is not always easy to derive explicit expressions, or time bounds, for the estimation of the random variable X_f, directly from the transition matrix of a Markov Chain. This section first describes a general Markov chain framework introduced by He and Yao[21] together with its limitations. Next, tail inequalities for the calculation of overwhelming success (or failure) probabilities are discussed. At last the ideas behind the most common methods used to simplify the analysis are introduced.

3.1 A Markov chain framework

Let $(X_t; t = 0, 1, ..)$ be an homogeneous absorbing Markov chain, defined on state-space S. Then the matrix of the probabilities of each state i to reach each state j in one step, i.e. the transition matrix, can be written in the following canonical form[22]

$$P = \begin{pmatrix} I & 0 \\ R & T \end{pmatrix}. \quad (1)$$

Here T denotes the set of transient states, $S-T$ the set of absorbing states, and R the sub-matrix of the probabilities

of going in one step from each state i, with $i \in T$ to each state j with $j \in S - T$.

The matrix $N = (I - T)^{-1}$ is called the fundamental matrix of the absorbing Markov chain, and $m = N1$ is the vector of the mean absorption times where each entry m_i is the expected time to reach the recurrent states when the chain starts from state i[23].

Unfortunately for most transition matrices P, it is difficult if not impossible to invert the matrix $(I - T)$ to obtain the fundamental matrix, N.

An attempt of building a general framework for analyzing the average hitting times of EAs by using the theory of Markov chains was made by He and Yao[21]. In the paper it is shown that when the matrix T is in a simple form (i.e. a tridiagonal matrix or a lower triangular c) it is possible to obtain an explicit expression for the vector m. For when it is not possible or practical to derive bounds from the explicit equations, conditions are given for deriving the hitting time without the need of solving the linear system.

For a (1+1)-EA using elitist selection and any "proper" mutation optimizing functions defined in the binary space $S = \{0, 1\}^n$, it is shown that the sub-matrix T of the canonical transition matrix P is a lower triangular matrix. So the first hitting time of the EA (i.e.the time to reach the absorbing states) is given by $m = (I - T)^{-1}1$, and explicit expressions for each m_i are derived.

By changing the selection scheme to elitist selection accepting equal-valued individuals, the sub-matrix T is not to necessarily lower triangular, and it is shown that in some special cases the matrix may be tridiagonal. In such cases explicit expressions for m can be derived again. Unfortunately, the case is related to an algorithm that just flips one bit per generation. So the EA is reduced to a random local search (RLS) algorithm.

A similar discussion can be applied to (N+N)-EAs with any elitist selection scheme which always keeps the best individual in the population. The transition matrix P associated with the EA is[21]

$$P = \begin{pmatrix} I & 0 & 0 & 0 & \dots & 0 \\ R_{10} & T_{11} & 0 & 0 & \dots & 0 \\ R_{20} & T_{21} & T_{22} & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ R_{L0} & T_{L1} & T_{L2} & T_{L3} & \dots & T_{LL} \end{pmatrix} \quad (2)$$

and the first hitting times are given by

$$\begin{aligned} &m_1 = (I - T_{11})^{-1}1, \\ &m_k = (I - T_{kk})^{-1}(T_{k1}m_1 + \ldots + T_{kk-1}m_{k-1}), \\ &\quad k = 2, \ldots, L. \end{aligned} \quad (3)$$

"Since the matrix $T_{kk}(k = 1, \ldots, L)$ usually is not in simple form, we cannot get an explicit solution to the system"[21].

For non-elitist selection schemes only in special cases will it be possible to easily invert the fundamental matrix, but no general rules may be given. Again advantages can be obtained if one-bit flipping mutation is considered. Such a mutation scheme leads to a tridiagonal matrix. He and Yao took advantage of this situation by analyzing all sorts of different selection schemes in a comparison between EAs using only one individual against population-based EAs on different pseudo-Boolean functions[24]. However, the explicit application of Markov chains was possible because only 1-bit flips were used leading to tridiagonal matrices which are not too difficult to invert. Nevertheless, the calculations are not simple.

The general limitations, described above, in deriving expected time expressions from the transition matrix highlight the necessity of introducing other randomized algorithm analysis tools that may be used as "tricks" to gather information about the Markov process without having to build the exact Markov chain model.

3.2 Tail inequalities

As described in the previous section, an important issue in the analysis of a randomized algorithm is the calculation of the success probability. Given a random variable X_f (for example the expected time for an event to occur such as the global optimum having been found), it can frequently occur that it assumes values that are quite higher than its expectation. For such a reason it is more interesting to find expected run times that occur with high probability rather than just their expectation which may be smaller. Given a deviation, tail inequalities are tools that estimate the probability that a random variable deviates from its expected value of the amount defined by the deviation[18]. In the analysis of EAs, tail inequalities "are useful to turn expected run times into upper time bounds which hold with overwhelming probability"[14]. Moreover, they can be used for deriving bounds of intermediate "important" events in the analysis of the optimization process that occur with high probability since "for many intermediate results, expected values are useless"[14].

Markov inequality. The fundamental tail inequality, which is also the basis for the others is the Markov inequality[18]: Let X be a random variable assuming only non-negative values, and $E[X]$ its expectation. Then for all $t \in R^+$,

$$Pr[X \geq t] \leq \frac{E[X]}{t}. \quad (4)$$

The Markov inequality is usually not strong enough to lead to useful results by itself, so other tail inequalities are derived from this one if extra information on the expectation is considered.

Chebyshev's inequality. Chebyshev's inequality derives from the Markov inequality by using the property that, with a small variance, large deviations from the expectation are unlikely[25]: Let X be a random variable, $E[X]$ its ex-

pectation, and $V[X]$ its variance. Then for all $t > 0$,

$$Pr(|X - E[X]| \geq t) \leq \frac{V[X]}{t^2}. \quad (5)$$

Chernoff's inequalities. Let $X_1, X_2, \ldots X_n$ be independent Poisson trials (i.e. taking values in {0,1}) such that, for $1 \leq i \leq n, Pr(X_i = 1) = p_i$. Then, for $X = \sum_{i=1}^{n} X_i$ the expectation $E(x) = \sum_{i=1}^{n} p_i$ and

1. for $0 \leq \delta \leq 1$

$$Pr(X \leq (1-\delta)E[X]) \leq \mathrm{e}^{\frac{-E[X]\delta^2}{2}}. \quad (6)$$

2. for $\delta > 0$

$$Pr(X > (1+\delta)E[X]) \leq \left[\frac{\mathrm{e}^{\delta}}{(1+\delta)^{1+\delta}}\right]^{E[X]}. \quad (7)$$

Chernoff's inequalities or "exponential" inequalities, are used for obtaining considerably sharper bounds on the tail probabilities[18]. They are the most used inequalities in the time complexity analysis of EAs, as they show how a binomially distributed variable is very close to its expected value and they "permit to produce an estimate with expected values as "true" values"[14].

3.3 Artificial fitness levels

When, in the mid-nineties, the first runtime analyzes of EAs were performed, the study of the (1+1)-EA was a natural choice. Although it is different compared to EAs often used in practice, its analysis is important for the following reasons[26]:

1. The (1+1)-EA is very efficient for a large number of functions;
2. The (1+1)-EA can be interpreted as a randomized hill-climber that can not get stuck forever on a local optimum;
3. The analysis of the (1+1)-EA reveals tools that can be used in the analysis of more complex EAs.

The most natural method spawned in the first runtime analyzes is artificial fitness levels. On pseudo-Boolean functions of unitation, the fitness value depends on the number of ones in the bit-string representing an individual, but not on their position. In analyzing the algorithm on such functions, it was natural to partition the space in sets depending on the number of ones of a bit-string. Later, the method was extended to be used in more general settings rather than just with unitation functions. A description of the general method follows.

Artificial fitness levels consist of creating a partition of the search space based on the fitness function. Rather than considering the whole search space S as a set of $|S| = n$ different states, it is divided into $m < n$ states $A_1, \ldots, A_m$, such that for all points $a \in A_i$ and $b \in A_j$ it happens that $f(a) < f(b)$ if $i < j$. Thus, A_m contains only optimal search points.

In such a way, the Markov chain can be constructed considering only m different states rather than n. Apart from providing a smaller Markov chain transition matrix, artificial fitness levels lead to simplified calculations.

Just by using artificial fitness levels the $O(n \log n)$ upper bound of the (1+1)-EA for OneMax can be obtained[11]. The proof of the lower bound is less straightforward. In order to prove a lower bound of $\Omega(n \log n)$, it is necessary to show that with overwhelming probability the algorithm takes at least $cn \log n$ steps to reach the optimum with c a positive constant. By using Chernoff bounds it is proved that with exponentially high probability the starting point has less than $2n/3$ ones. The rest of the proof shows that $\Omega(n \log n)$ steps are necessary to flip the remaining $n/3$ zero bits into one bits at least once[11]. The latter part of the proof is inspired by the coupon collector′s problem.

3.4 The coupon collector′s problem

The coupon collector′s problem is used very often in the analysis of EAs when it is necessary to derive a lower bound for a given number of bits to be chosen at least once. The problem is defined in the following way[18]:

"There are n types of coupons and at each trial one coupon is chosen at random. Each coupon has the same probability of being extracted. The goal is to find the exact number of trials before the collector has obtained all the n coupons".

Let us consider a Markov process with states $S_0, S_1, \ldots, S_n$ and let the process be in state S_i when i different coupons have been collected. If one coupon is obtained in each step, then the probability at state S_i of reaching state S_{i+1} is $P_{i+1} = i/n$. Then, since the expectation is $E = 1/P$, the expected time for all the coupons to be collected is[18]

$$E(X) = \sum_{i=0}^{n-1} \frac{n}{n-i} = n \sum_{i=0}^{n-1} \frac{1}{i} = n \log n + O(n). \quad (8)$$

The coupon collector′s theorem states that the value of X is sharply concentrated around its expected value with overwhelming probability:

The coupon collector′s theorem. Let X be the expected time for all the coupons to be collected. Then[18]

$$\lim_{n\to\infty} Pr[(n \log n - cn) \leq X \leq (n \log n + cn)] = \mathrm{e}^{-\mathrm{e}^{-c}} - \mathrm{e}^{-\mathrm{e}^{c}}. \quad (9)$$

As the value of c increases the probability reaches rapidly 1, centering the coupon collector′s expected time in a very small interval around the value of $n \log n$. Furthermore, the bound for not too small n are very close to the bounds in the limit[14], validating proofs for large enough n.

So by considering zero bits as missing coupons and one bits as collected ones, the idea has been applied to the analysis of the (1+1)-EA for OneMax. However the algorithm

may flip more than one bit in each generation, meaning that in each trial more than one coupon is obtained, hence the coupon collector results may be applied, but not in their original form. Since their first appearance, artificial fitness levels and the coupon collector′s problem have frequently been applied to the analysis of EAs[14].

3.5 The gambler′s ruin problem

When, the function′s landscape contains a plateau of constant fitness, and the EA needs to cross it to reach the global optimum, then it is forced to do a random walk on the plateau. This occurs because the algorithm cannot be driven towards the optimum by increasing fitness values. Big plateaus, such as those of the Needle-in-a-haystack problem, lead to exponential run times[27]. Surprisingly, if the plateau is not too large the (1+1)-EA accepting equal valued individuals has a good performance. This result has been obtained by applying the gambler's ruin theory[25] to the analysis of EAs.

The gambler′s ruin problem, derived from classical probability theory, was introduced in the analysis of EAs by Jansen and Wegener when analyzing the (1+1)-EA on plateaus of constant fitness of short path problems (SPPs)[17]. By following similar ideas they were able to derive a polynomial upper bound for the expected time to optimize the SPP. Subsequently it was again found useful, for similar goals, by Giel and Wegener when analyzing the same algorithm on instances of the maximum matching problem[28] and by Oliveto for vertex cover instances[29].

The problem is formalized as follows[25]:

"Consider a gambler who wins or loses a dollar with probability p and q respectively. Let his initial capital be z and his adversary's capital be $a - z$. The game continues until the gambler or his adversary is ruined. The goal of the ruin problem is to find the probability of the gambler's ruin and the probability distribution of the game".

If q_z is the probability of the gambler′s ruin, hence $p_z = 1 - q_z$ the probability of his adversary′s ruin, then[25]

1. if $p \neq q$

$$q_z = \frac{(q/p)^a - (q/p)^z}{(q/p)^a - 1} \tag{10}$$

2. if $p = q = 1/2$

$$q_z = 1 - \frac{z}{a}. \tag{11}$$

In[25] the following, formulae for the expected duration of the game are also derived. Let D_z be the expected duration of the game. Then

1. if $p \neq q$

$$D_z = \frac{z}{q-p} - \frac{a}{q-p}\frac{1-(q/p)^z}{1-(q/p)^a} \tag{12}$$

2. if $p = q = 1/2$

$$D_z = z(a - z).$$

However these expectations are seldom used in the analysis of EAs. Usually it is preferred to estimate the duration of the game by which at least the number of trials guaranteeing the adversary′s ruin have been accomplished with sufficient probability (i.e. constant).

The (1+1)-EA is modeled on plateaus by considering the conditional probabilities of steps moving towards the global optimum as the gambler winning a bet and those of moving away from the optimum as the gambler losing a bet. The ruin problem considers that in each step one dollar may be won or lost. Attention has to be paid when applying the ruin ideas to the step length which for the (1+1)-EA varies according to the number of bits that are flipped, hence more than one dollar may be won or lost in each step.

3.6 Potential functions

Sometimes, partitioning the space (i.e. artificial fitness levels) is not sufficient to obtain useful time bounds. Potential functions are a "straightforward generalization of the proof technique of artificial fitness layers"[30].

Potential functions consider a different distance heuristic to measure the distance of the current solution from the optimum rather than using the fitness function values. In other words, the EA uses the fitness function to decide whether to accept the new solution at the next step, while the analysis uses the potential function to measure the algorithm′s progress. A Markov process that is slower than the actual one is analysed in order to obtain an upper bound for the optimization time. Obviously if the upper bound is correct for a slower process it is still correct for a faster one.

Originally the potential functions method was introduced in the theory of EAs by Droste to obtain an $O(n \log n)$ upper bound for the (1+1)-EA in the optimization of linear functions[15]. The method also turned useful for upper bounds of the (μ+1)-EA for pseudo-Boolean functions such as leading ones and OneMax[30]. For instance, in the proof of the upper bound for leading ones, the potential function at time t returns the maximum leading ones value out of all the individuals of the population at time step t. Then the probabilities that the potential is increased at each step and the number of times the potential needs to be increased, before it is guaranteed the optimum has been found, have to be estimated. Further details may be found in [14].

3.7 Typical run investigations

Typical runs have been introduced in the analysis of EAs following the consideration that the "global behavior of a process" is predictable with high probability contrary to the local behavior which, instead, is quite unpredictable. For instance, the result of just one coin toss is not easy to state (i.e. if it will be a head rather than a tail), but it is possible to get very tight bounds on the number of heads that have appeared after a large number of coin tosses, allowing for an exponentially small error probability.

The idea behind typical runs is that of dividing the pro-

cess into phases which are long enough to assure that some event happens with very high probability. Hence, each phase has an exponentially low probability of failing (the failure probability), meaning that with very low probability the considered event does not happen in the considered phase. Following such a sequence of events, the last phase should lead the EA to the global optimum with exponentially small failure probability. As a result, it can be proved that in a typical run the algorithm finds the global optimum in a time that is lower than the sum of each phase time.

Typical run investigations occur very often in run time analyzes of EAs, combined with Chernoff bounds used to obtain the failure probabilities. Examples are the exponential time bound for the (1+1)-EA on the trap function[27] and the proof that the use of crossover can, at least sometimes, reduce the run time from super-polynomial to polynomial[31].

3.8 Drift analysis

Drift analysis, deriving from the theory of Martingales, was introduced in 1997[32] as a model for estimating the computational time of EAs, and published in 2001 by He and Yao[33].

Martingales have been used in the nineties to prove the convergence of EAs using non-elitist selection strategies[7].

Let S_{opt} be the set of populations containing the optimal solution of an optimization problem, S the set of all possible populations, and $d(X_t)$ a function for measuring the distance from population X to S_{opt}, at time-step t. Then the one-step mean drift at generation t is[33]:

$$E[d(X_t) - d(X_{t+1})] := \sum_{X \in S} \Big(d(X_t) - d(X_{t+1})\Big) \cdot P(X_t, X_{t+1}; t). \tag{13}$$

If $E[d(X_t) - d(X_{t+1})] \geq 0 \quad \forall X \in S$ then the distance function $d(X_t)$ is a super-martingale.

The idea behind drift analysis is that if the distance of the current solution from the optimal one is d and "the drift towards the optimal solution is greater than Δ at each time step, we would need at most d/Δ time steps to find the optimal solution. Hence the key issue is to estimate Δ and d"[33].

The main motivation in the concept of drift analysis is that it may often be easier to estimate the drift of a random sequence rather than the first hitting times directly from the Markov chain. Obviously it is still necessary to model the EA as a random sequence, for instance with Markov chains.

In [33] and [34] drift conditions are given to determine whether the average time complexity of an EA to solve a given problem is polynomial or exponential in the problem size. The most general ones, for polynomial time, are described in the following[33].

Let $\{d(X_k); k = 1, 2, \}$ be a distance function giving the distance of population X at time step k from the global optimum.

Condition 1. For any given population X, there exists a polynomial of the problem size n, $h_0(n) > 0$ such that $d(X) \leq h_0(n)$.

Condition 2. For any time $k \geq 0$, if the population X_k satisfies $d(X_k) > 0$, then there exists a polynomial of problem size n, $h_1(n) > 0$ such that

$$E[d(X_k) - d(X_{k+1})] \geq \frac{1}{h_1(n)}. \tag{14}$$

If $d(x)$ satisfies conditions 1 and 2 then the expected first hitting time to reach the optimal solution is bounded from above by a polynomial function of the problem size.

Similarly the lower bound of the first hitting time is estimated by bounding the one-step mean drift from above.

The main advantage of drift analysis over Markov chains is that the former overcomes the difficulties in deriving the time expressions from complex transition matrices and that, being a more general tool, there are less restrictions to the algorithms and problems it can be applied to. The two key issues in applying drift analysis are those of defining a good distance function and estimating the mean drift. On the other hand, the generality of these two steps may require a higher mathematical confidence than that needed in the application of Markov chains.

He and Yao have shown how results of the analysis of EAs, previously proved by using other methods, could have been obtained with the drift analysis technique and they have extended the analysis of functions such as linear functions and long path problems to population-based EAs rather than just analyzing the (1+1)-EA. They have also shown how by using drift analysis it is possible to obtain bounds for an EA with crossover on instance classes of the NP-hard subset sum problem[33].

The importance of carefully choosing the distance function is highlighted in [34]. He and Yao show that considerably different bounds are obtained with different distance functions and the most intuitive choice is not always the best.

Furthermore, drift analysis has proved to be a very powerful and useful tool for proving exponential lower bounds in all sorts of recent work related to classical combinatorial optimization problems. An example is the analysis performed by Giel on instances of the maximum matching problem[28].

3.9 Family trees

Family trees have been found particularly useful for deriving lower bounds (but not only) for population-based EAs, in particular the (μ+1)-EA[30]. A family tree $T_t(x)$ contains an individual x as the root and all the descendants of x at time t. Obviously $T_0(x)$ contains only the root x. Although all the nodes in a family tree are different individuals, they may represent the same string. The idea behind the method when used for the obtainment of lower bounds is that, with overwhelming probability, the tree has to have at least a certain depth for the optimum to be found. Then the proof consists of calculating the

time needed for a tree of such depth to be created. Family trees are a powerful tool introduced in the field analyzing EAs especially for examining population-based ones. The technique is not limited to the analysis of the (μ+1)-EA. "The most interesting direction seems to be an extension to (μ+λ) strategies by a combination of existing theory on the (1+λ)-EA"[30],

4 A decade of results

This section discusses the state-of-the art concerning the computational complexity of EAs. Subsection 4.1 considers the improvements of population-based EAs analyzes and describes the reasons the recent research work in this area and points out the limitations of the current knowledge. Subsection 4.2 focuses on the improvements achieved considering EAs for classical combinatorial optimization problems.

4.1 From an individual to a population

In a similar manner to the analysis of the (1+1)-EA, the first theoretical results for population-based EAs were obtained for functions with "nice" structures. The idea was to understand when a population or the crossover operator could be beneficial. The first analyzes were motivated from the conjecture that GAs were supposed to outperform (1+1)-EAs but no theoretical proofs were available. In particular there was a necessity of proving the existence of a family of functions on which GAs perform better. Another direction in the analysis of population-based EAs has been to consider different evolution strategies (ESs) rather than approaching GAs directly. In the following subsections these two research lines will be considered separately.

4.1.1 When populations are beneficial

When comparing single-individual EAs against population based ones, the analysis has to take into account that the former algorithms may use restart strategies. Apart from fairness in the comparison this is also necessary because in practice EAs may be restarted after some time. In 1998 Rudolph stated that "it is easy to see that an EA with larger population size cannot be worse than the (1+1)-EA with respect to the number of iterations"[7]. However, the number of generations is not a fair performance measure for the comparison, while the number of fitness evaluations is an appropriate one.

In 1999 Jansen and Wegener presented, for the first time, an analysis of a function (i.e. $JUMP_m$) that can be optimized in a more efficient way with the use of uniform crossover compared to an EA that does not use it, if replications of individuals in the population are avoided[31]. Typical run investigations and results of the coupon collector′s problem are useful in the analysis. The problem of whether populations alone could be beneficial remained open.

In 2001, Jansen and Wegener extended their results by producing a class of functions for which a population-based EA, without the use of crossover, takes polynomial time for its optimization. On the other hand, the (1+1)-EA takes super-polynomial optimization time[35]. In order to obtain their results, they have to avoid that the population does not converge too rapidly. The algorithm satisfies the condition if a diversity measure is applied. There are two possibilities: avoidance of duplicates and avoidance of replications. The latter diversity mechanism only assures that every child has at least one bit mutated in each generation. This is the mechanism used by the algorithm. Again, it is interesting to point out that typical runs are often useful in these analyzes.

Classes of royal road functions for which an EA (i.e. a steady-state GA), with the use of uniform crossover and one point crossover, outperforms any evolution strategy (i.e. polynomial time against exponential time) are also presented by Jansen and Wegener the same year[36]. Witt introduced functions for which, by increasing the population size, the run time improves from exponential to polynomial without the use of diversity mechanisms or of crossover (The best gap proved previously was super-polynomial vs polynomial)[37]. However, examples with reverse run-times are also given. In 2003 Storch and Wegener introduced new royal road functions for which the same previous results can be obtained but with the minimum population size of 2 individuals[38]. He and Yao, in 2002, presented another paper proving how a population may be beneficial over single individuals on all sorts of pseudo boolean toy problems[24]. This time the results are obtained through careful selection strategies, although just 1-bit flips are used.

The above described results prove that sometimes populations and/or crossover may be useful by showing function classes for which a combination of a certain population size and a well chosen crossover operator provide a better performance. But it is still not very clear when and how to choose the population size or the crossover operator. Also, there are numerous examples of when they are not useful at all. Hence, the question of whether it is "normal" to encounter the former landscapes in real-world applications remains open. However empirical results seem to suggest that they should be preferred.

4.1.2 Population-based evolution strategies

Another direction in the analysis of population-based EAs has been that of considering different kinds of evolution strategies (ESs) rather than approaching directly the GA. Jansen analyzed the (1+ λ)-EA on leading ones and OneMax and extended their results empirically to other more complicated benchmark functions[39]. "Unfortunately the analyzes presented here are not capable of making strong a priori predictions regarding the appropriate value of λ for an arbitrary landscape". Still considering offspring population ESs, Storch and Jägerskupper, instead, compared the (1+λ) and the (1,λ) strategies and proved that not only are there some values of λ for which the comma strategy performs at least as well as the plus strategy, but may also outperform it on some functions[40]. An extension, of the artificial fitness levels method is introduced for the analysis.

On the other hand, Witt analyzed the (μ+1)-EA on well known pseudo-Boolean functions (i.e. OneMax, Leading

Ones, and SPPs) and produced an example for which the algorithm is more efficient than the (1+1)-EA[30]. However on the "classic" pseudo boolean functions no benefit was found either for the "offspring" population ES or for the "parent" population one. At least the function for which populations give a drastic improvement is more complicated than the other analyzed ones, partially confirming the empirical conjecture. While the proofs of the upper bounds usually use the potential functions technique, the family trees approach is introduced for the obtainment of the lower bounds.

The above results reveal a necessity of analyzing population-based EAs on problems that may be more similar to real-world applications. Straightforward examples are classical combinatorial optimization problems having practical applications. If a population-based EA performs better than a single-individual EA on "difficult" combinatorial optimization problems, then there may be a good chance that they perform better on difficult real life applications.

However, except for He and Yao′s analysis of a (2N+2N)-EA with crossover on some instances of the subset sum problem[33], the few available results are related to problems in P. Such results will be discussed in the next section.

4.2 From toy problems to classical problems

Randomised search heuristics are robust problem-independent search algorithms. They are designed for the following practical reasons[16]:

1. some problems in realistic applications need to be solved quickly, and often there are not enough resources in terms of money, time or knowledge to construct a problem-dependent algorithm.

2. the function that needs to be optimized may not be known, and only by sampling the search space may some knowledge about the problem be gained.

The above goals reveal the necessity of designing problem-independent algorithms even though an algorithm especially constructed for solving a problem should work better. Hence the focus is on designing heuristics which should work well on large classes of problems. Concerning their theoretical analysis, it is not possible to study the algorithms′ performance on "unknown" functions. So, their behavior has to be analyzed on large classes of functions and on well-known combinatorial optimization problems with practical applications. Both positive and negative results can help understanding on what kind of problems an algorithm may perform better or worse, hence help practitioners in choosing the heuristic and in setting the parameters. Furthermore, EAs are often used for solving combinatorial optimization problems because they are easy to use and empirical results suggest their performance is often successful.

The analysis of the (1+1)-EA on pseudo-Boolean functions may be criticized because the problems tackled by the algorithm are not realistic enough. Neither is the algorithm, not having populations far less a crossover operator. The main justification for such analyzes was that the "structured" problems gave insights on the behavior of the algorithm and that useful tools for more complex analyzes would be obtained. In fact, nowadays, some results on "classical" combinatorial optimization problems have appeared.

It is generally agreed that an algorithm is "efficient" if its expected time is bounded above by a polynomial function of the problem size. When tackling a problem with an algorithm the first question requiring an answer is whether the algorithm can efficiently find the solution. If a problem is in the P class[41], then there exists an algorithm that solves it efficiently. If it cannot be expected that an EA may outperform the problem-specific algorithm, it should be at least efficient on "easy" problems.

On the other hand, if a problem belongs to the NP-Hard class[41], then it is not expected that an algorithm can find the solution of every instance of the problem efficiently, unless $P = NP$ which is unlikely. In this case the interest is turned to the quality of the approximate solution the EA can guarantee on the problem given polynomial time. In other terms, given any instance of the problem, how worse will the solution the EA delivers be compared to the global optimum? If it is not expected that an algorithm performs well on some instances of an NP-Hard problem, this does not exclude the possibility of it performing well on the rest. Hence, average case analyzes are also of great interest.

Once the above questions are answered comparison with problem specific algorithms and with other heuristics may be attempted.

4.2.1 Problems in P

The problems of sorting a sequence of elements and that of finding shortest paths in graphs are basic but important computer science problems. The (1+1)-EA can efficiently sort a sequence of n elements in expected time $\Theta(n^2 \log n)$[42]. The algorithm mutates individuals by exchanging the positions of two randomly chosen elements (exchange operator). The algorithm is also efficient if it mutates an individual by moving an element to a different position and shifting the ones after that place (jump operator). On this problem the importance of choosing the fitness function carefully is highlighted. Scharnow[42] analyze the performance of the algorithm by using different measures of presortedness and prove that a bad choice leads to exponential run-times. As previously conjectured the algorithm performs worse than Quicksort or Mergesort but is efficient anyway for sorting. Matters become worse when considering the problem of finding shortest paths between a source and the other vertices of an undirected graph. It is pointed out by Scharnow that the (1+1)-EA can get trapped in a landscape similar to that of a needle-in-a-haystack, hence the expected time, in such a case, is exponential. The authors prove that by modeling the problem as a multi-objective optimization problem the time can be bounded by $O(n^3)$. The $n-1$ objectives are to minimize the lengths

of each of the $n-1$ paths. Modeling single-objective problems as multi-objective ones has also been considered for spanning trees[43] and for vertex covers[44].

So the (1+1)-EA is not always efficient for "easy" problems, although by changing the algorithm slightly, the runtimes may vary considerably. This can be noticed again for the Eulerian cycle problem. By using the canonical *exchange* operator, the (1+1)-EA has exponential expected runtime, while with the *jump* operator, it is $O(m^5)$, with m being the number of graph edges[45]. Doerr reduce the bound to $O(m^3)$ by modifying the jump operator[46]. They also prove the existence of classes for which the bound is tight. Recently, two different kinds of graph representation have been studied. In this way, firstly, the expected runtime has been lowered to $\Theta(m^2 \log m)$[47]. Secondly, by considering adjacency lists representations, the expected runtime has been further reduced to $\Theta(m \log m)$[48].

Another problem in P for which the (1+1)-EA is not efficient is the maximum matching problem as proved by Giel and Wegener [28]. They prove that the (1+1)-EA is a PRAS (i.e. a $(1+\epsilon)$-Approximation algorithm for the problem). In particular, the expected time to guarantee the approximation is $O(m^{2\lceil 1/\epsilon \rceil})$ for $\epsilon > 0$. They use augmenting path ideas from the theory of maximum matchings to obtain the above result. Furthermore, they prove the existence of instances for which the expected exact optimization time is exponential in the problem size. A crucial tool for the exponential time result is drift analysis, while the gambler ruin problem proves to be useful when analyzing other instance classes.

Expecting polynomial upper bounds for "easy" problems does not seem to be exaggerated even when the problem is considered one of the most difficult in P. It would be interesting to understand whether populations and crossover may lead to improved time bounds, although the authors conjecture they will not.

Matters get better when considering the spanning tree problem. Neumann and Wegener prove a bound of $\Theta(m^2 \log n)$ for the (1+1)-EA as long as the edge weights are polynomially bounded[49]. Again, notions from the theory of spanning trees are useful for the results as are typical runs and ideas derived from the coupon collector′s problem. It is also proved that with $\lambda = \lceil m^2/n \rceil$ a $(1+\lambda)-EA$ has an upper bound of $O(n \log n)$. This, again, holds if the weights are polynomially bounded. However no improvement due to populations can be seen from the result. Considering recombination, "only problem-specific crossover operators seem to be useful". However, "It is difficult to analyze heuristics with these crossover operators"[49].

It is striking that for most of the described problems there is no difference in terms of asymptotic performance or approximation quality between the (1+1)-EA and the random local search (RLS) algorithm. It would be expected that a global search algorithm such as the (1+1)-EA would exploit its advantage of being able to perform larger steps than a RLS algorithm. However this is not evident in the results discussed above.

4.2.2 NP-hard problems

Considering "difficult" problems (i.e. NP-Hard problems) there may be higher chances that EAs may be competitive. On the NP-Hard partition problem Witt has proved that both the (1+1)-EA and an RLS algorithm find at least a 4/3-approximate solution in an expected number of $O(n^2)$ steps. Furthermore, the algorithms are both PRAS if multiple runs are used. An average case analysis, also shows that the (1+1)-EA creates solutions that "converge to optimality in expectation"[50]. Although the partition problem has a fully polynomial approximation scheme (FPAS)[51], not much better can be expected for NP-hard problems and "this is also as strong as the result for the LPT rule"[50].

Sometimes general search randomized heuristics can do nearly as well as problem specific algorithms. Storch has proved that a Metropolis algorithm can find maximum cliques on random planar graphs in time $\Theta(n)$ with overwhelming probability and in expectation[52]. The (1+1)-EA, although being "efficient", needs $\Theta(n^6)$ generations. Similar results also hold for semi-random planar graphs. The work was extended proving that on semi-random sparse graphs, populations (i.e. a $(\mu+1)$-EA with distinct individuals) are beneficial compared to single individuals[53]. In particular, the existence of random classes of graphs where even a super-polynomial decrease of the expected number of steps is proved, by applying a large enough population.

For vertex cover, a similar problem to Clique[41], Friedrich have proved that there exist instances for which the (1+1)-EA can produce arbitrarily bad approximations[44]. They have further shown that a simple evolutionary multi-objective optimizer (SEMO) does not suffer of the same problem. Oliveto have also shown that the same holds if the (1+1)-EA uses multi-starts[29]. However, if for both the Clique and the Vertex Cover problems the existence of exponential time instance classes has been proved, the quality of the approximate solutions that can be guaranteed by an EA remains to be ascertained.

5 Conclusion and future work

The early results of the time complexity theory of the (1+1)-EA on toy problems have led to the development of several tools and tricks that, being general, can be applied to the analysis of the same algorithm on more sophisticated problems.

The time complexity analysis of evolutionary algorithms seems to be developing along two different but converging lines. On one hand the current research is directed towards analyzing the (1+1)-EA on famous classical combinatorial optimization problems which are NP-Hard. For such classes of problems it is not expected that an algorithm, especially designed for the problem, is able to find the solution in polynomial time, far less a general purpose algorithm like an EA. As a consequence the attention is focused on the quality of approximate solutions the EA is able to produce indepen-

dent from the instance class being tackled. Although not on a NP-Hard problem, Giel and Wegener have proved that the (1+1)-EA is a PRAS for the Maximum Matching problem [28]. The achievement of this result (i.e. analyzing EAs as approximation algorithms for maximum matching),"similar to that for simulated annealing"[54], was considered "very valuable" in the mid-nineties[12].

Another open issue was to understand "whether an EA offers any advantage over the classical optimization algorithms for TSPs in terms of average case complexity. This is a very difficult problem as the average case analysis is already difficult to conduct for a deterministic algorithm, let alone for a population-based stochastic algorithm"[12]. While the problem concerning TSP is still open, an average case analysis of the (1+1)-EA for the NP-Complete Partition problem has been performed by Witt and the result is that the algorithm "converges to optimality in expectation"[50]. Witt also proves that the algorithm produces at least a 4/3-approximate solution in expected time $O(n^2)$ on any instance of the problem, and that it is a PRAS if multiple runs are used.

However, if some past open questions have been answered, there still is a need of understanding how well EAs work on other "famous" problems, especially NP-Hard ones. In particular, average-case analyzes still lack and results related to EAs as approximation algorithms are still very few. It still is a mystery whether there is an important problem where an EA can outperform the best algorithm known for solving it.

On the other hand, the research concerning more realistic EAs, using at least populations if not crossover, is still way behind. The current research seems to be following a similar strategy as that applied to the analysis of the (1+1)-EA, which starting from "nice structured" problems, has enabled its extension to more realistic ones in just a few years. These attempts will hopefully lead to mathematical tools which will simplify the analyzes of (N+N)-EAs first and GAs later. Empirical work suggests that population-based EAs should perform well on various "difficult" combinatorial optimization problems. However there are no available theoretical results confirming the conjecture. Even though there has been a consistent improvement in the last decade it still is not clear how long it will take before it will be possible to make precise statements about GAs for important problems.

Another issue that needs great consideration is understanding if the results concerning EAs for classical combinatorial optimization problems actually do help practitioners in using them for optimizing "unknown" functions. The results reported throughout the paper are certainly very valuable concerning the application of EAs for classical combinatorial optimization problems. However, the main real-world application domain for randomized search heuristics, such as EAs, is "unknown" functions. It is not clear whether the available and desirable results, in such a setting, help in choosing a randomized search heuristic rather than another one, or if they help the choice of the parameter values.

References

[1] R. Sarker, M. Mohammadian, X. Yao. *Evolutionary Optimization*, Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[2] D. E. Goldberg. *Genetic Algorithms for Search, Optimization, and Machine Learning*, Addison-Wesley, USA, 1989.

[3] J. H. Holland. *Adaptation in Natural and Artificial Systems*, 2nd ed., MIT Press, Cambridge, MA, 1992.

[4] A. E. Eiben, G. Rudolph. Theory of Evolutionary Algorithms: A Bird′s Eye View. *Theoretical Computer Science*, vol. 229, no.1–2, pp. 3–9, 1999.

[5] T. Bäck, D. B. Fogel, Z. Michalewicz. *Handbook of Evolutionary Computation*, IOP Publishing Ltd, Bristol, UK, 1997.

[6] G. Rudolph. Convergence Analysis of Canonical Genetic Algorithms. *IEEE Transactions on Neural Networks*, vol. 5, no.1, pp. 96–101, 1994.

[7] G. Rudolph. Finite Markov Chain Results in Evolutionary Computation: A Tour d′Horizon. *Fundamenta Informaticae*, vol. 35, no.1–4, pp. 67–89, 1998.

[8] H. Aytug, G. J. Koehler. Stopping Criteria for Finite Length Genetic Algorithms. *ORSA Journal on Computing*, vol. 8, no.2, pp. 183–191, 1996.

[9] D. Greenhalgh, S. Marshall. Convergence Criteria for Genetic Algorithms. *SIAM Journal on Computing*, vol. 30, no.1, pp. 269–282, 2000.

[10] M. Safe, J. A. Carballido, I. Ponzoni, N. B. Brignole. On Stopping Criteria for Genetic Algorithms, *In 17th Brazilian Symposium on Artificial Intelligence*, *Lecture Notes in Artificial Intelligence*, Springe-Verlag, vol. 3171, pp. 405–413, 2004.

[11] S. Droste, T. Jansen, I. Wegener. On the Analysis of the (1+1) Evolutionary Algorithm. *Theoretical Computer Science*, vol. 276, no. 1–2, pp. 51–81, 2002.

[12] X. Yao. An Overview of Evolutionary Computation. *Chinese Journal of Advanced Software Research*, vol. 3, no. 1, pp. 12–29, 1996.

[13] H. G. Beyer, H. G. Schwefel, I. Wegener. How to Analyse Evolutionary Algorithms. *Theoretical Computer Science*, vol. 287, no. 1, pp. 101–130, 2002.

[14] I. Wegener. Methods for the Analysis of Evolutionary Algorithms on Pseudo-Boolean Functions. In *Evolutionary Optimization*, R. Sarker, M. Mohammadian, X. Yao (eds.), Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[15] S. Droste, T. Jansen, I. Wegener. A Rigorous Complexity Analysis of the (1+1) Evolutionary Algorithm for Separable Functions with Boolean Inputs. *Evolutionary Computation*, vol. 6, no. 2, pp. 185–196, 1998.

[16] I. Wegener. On the Design and Analysis of Evolutionary Algorithms. In *Proceedings of the Workshop on Algorithm Engineering as a New Paradigm*, Kyoto University, Japan, pp. 37–47, 2000.

[17] T. Jansen, 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, no. 6, pp. 589–599, 2000.

[18] M. Motwani, P. Raghavan. *Randomized Algorithms*, Cambridge University Press, UK, 1995.

[19] T. Bäck. Optimal Mutation Rates in Genetic Search. In *Proceedings of the 5th International Conference on Genetic Algorithms*, Morgan-Kaufman, pp. 2-8, 1993.

[20] V. Cutello, G. Nicosia, P. S. Oliveto, M. Romeo. On the Convergence of Immune Algorithms. In *Proceedings of the 1st IEEE Symposium on Foundations of Computational Intelligence*, Honolulu, Hawaii, USA, 2007.

[21] J. He, X. Yao. Towards an Analytic Framework for Analyzing the Computation Time of Evolutionary Algorithms. *Artificial Intelligence*, vol. 145, no. 1–2 pp. 59–97, 2003.

[22] M. Iosifescu. *Finite Markov Processes and Their Applications*, John Wiley & Sons, 1980.

[23] D. L. Isaacson, R. W. Madsen. *Markov Chains: Theory and Applications*, John Wiley & Sons, 1976.

[24] J. He, X. Yao. From an Individual to a Population: An Analysis of the 1st Hitting Time of Population-based Evolutionary Algorithms. *IEEE Transactions Evolutionary Computation*, vol. 6, no. 5, pp. 495–511, 2002.

[25] W. Feller. *An Introduction to Probability Theory and Its Applications*, vol. 1, John Wiley & Sons, 1968.

[26] I. Wegener. Theoretical Aspects of Evolutionary Algorithms. In *Proceedings of the 28th International Colloquium on Automata, Languages and Programming*, London, UK, pp. 64–78, 2001.

[27] S. Droste, T. Jansen, I. Wegener. On the Optimization of Unimodal Functions with the $(1 + 1)$ Evolutionary Algorithm. In *Proceedings of the Parallel Problem Solving from Nature Conference*, *Lecture Notes in Computer Science*, Springer-Verlag, vol. 1498, pp. 13–22, 1998.

[28] O. Giel, I. Wegener. Evolutionary Algorithms and the Maximum Matching Problem. In *Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science*, Speringer-Verlag, pp. 415–426, 2003.

[29] P. S. Oliveto, J. He, X. Yao. Evolutionary Algorithms and the Vertex Cover Problem. In *Proceedings of the Congress on Evolutionary Computation*, to be published.

[30] C. Witt. Runtime Analysis of the (μ+1) EA on Simple pseudo-Boolean Functions. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, Seattle, Washington, USA, pp. 651-658, 2006.

[31] T. Jansen, I. Wegener. On the Analysis of Evolutionary Algorithms - A Proof that Crossover Really Can Help. In *Proceedings of the 7th Annual European Symposium on Algorithms*, Springer-Verlag, pp.184–193, 1999.

[32] J. He. Study on the Foundation of Evolutionary Computation. Technical Report, Department of Computer Science, Harbin Institute of Technology, China, 1998 (in Chinese).

[33] J. He, X. Yao. Drift Analysis and Average Time Complexity of Evolutionary Algorithms. *Artificial Intelligence*, vol. 127, no. 1, pp. 57–85, 2001.

[34] J. He, X. Yao. A Study of Drift Analysis for Estimating Computation Time of Evolutionary Algorithms. *Natural Computing: An International Journal*, vol. 3, no. 1, pp. 21–35, 2004.

[35] T. Jansen, I.Wegener. On the Utility of Populations in Evolutionary Algorithms. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, IEEE, pp. 1034–1041, 2001.

[36] T. Jansen, I. Wegener. Real Royal Road Functions: Where Crossover Provably is Essential. *Discrete Applied Mathematics*, vol. 149, no. 1–3, pp. 111–125, 2005.

[37] C. Witt. Population Size vs. Runtime of a Simple EA. In *Proceedings of the Congress on Evolutionary Computation*, Canberra, pp. 1996–2003, 2003.

[38] T. Storch. Real Royal Road Functions for Constant Population Size. *Theoretical Computer Science*, vol. 320, no. 1, pp. 123–134, 2004.

[39] T. Jansen, K. A. De Jong, I. Wegener. On the Choice of the Offspring Population Size in Evolutionary Algorithms. *Evolutionary Computation*, vol. 13, no. 4, pp. 413–440, 2005.

[40] J. Jägerskupper, T. Storch. When the Plus Strategy Outperforms the Comma Strategy and When Not. In *Proceedings of the 1st IEEE Symposium on Foundations of Computational Intelligence*, Hawaii, USA, pp. 25–32, 2007.

[41] M. R. Garey, D. S. Johnson. *Computers and Intractability a Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, New York, 1979.

[42] J. Scharnow, K. Tinnefeld, I. Wegener. Fitness Landscapes Based on Sorting and Shortest Path Problems. In *Proceedings of 7th Conference on Parallel Problem Solving from Nature Conference*, pp. 54–63, 2002.

[43] F. Neumann, I. Wegener. Minimum Spanning Trees Made Easier Via Multi-objective Optimization. *Natural Computing*, vol. 5, no. 3, pp. 305–319, 2006.

[44] T. Friedrich, J. He, N. Hebbinghaus, F. Neumann, C. Witt. Approximating Covering Problems by Randomized Search Heuristics Using Multi-objective Models. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, 2007*, to be published.

[45] F. Neumann. Expected Run-times of Evolutionary Algorithms for the Eulerian Cycle Problem. In *Proceedings of the Congress on Evolutionary Computation*, IEEE, pp. 904–910, 2004.

[46] B. Doerr, N. Hebbinghaus, F. Neumann. Speeding Up Evolutionary Algorithms through Restricted Mutation Operators. In *Proceedings of the 9th Parallel Problem Solving from Nature Conference*, *Leture Notes in Computer Science*, Springer-Verlag, Reykjavik, Iceland, vol. 4193, pp. 978–987, 2006.

[47] B. Doerr, C. Klein, T. Storch. Faster Evolutionary Algorithms by Superior Graph Representation. In *Proceedings of the 1st IEEE Symposium on Foundations of Computational Intelligence*, Hawaii, USA, pp. 245–250, 2007.

[48] B. Doerr, D. Johannsen. Adjacency List Matchings - An Ideal Genotype for Cycle Covers. In *Proceedings of the Annual Conference on Genetic and Evolutionary Computation, 2007*, to be published.

[49] F. Neumann, I. Wegener. Randomized Local Search, Evolutionary Algorithms, and the Minimum Spanning Tree Problem. In *Proceedings of Genetic and Evolutionary Computation Conference*, *Leture Notes in Computer Science*, Springer-Verlag, Seattle, Washington, USA, vol. 3102, pp. 713–724, 2004.

[50] C. Witt. Worst-case and Average-case Approximations by Simple Randomized Search Heuristics. In *Proceedings of the 22nd Annual Symposium on Theoretical Aspects of Computer Science*, *Lectures Notes in Computer Science*, Springer-Verlag, Stuttgart, Germany, vol. 3404, pp. 44–56, 2005.

[51] D. S. Hochbaum. *Approximation Algorithms for NP-Hard Problems*, Wadsworth Publishing Company, USA, 1996.

[52] T. Storch. How randomized search heuristics find maximum cliques in planar graphs. In *Proceedings of the Annual Conference on Genetic and Evolutionary Computation*, Seattle, Washington, USA, pp. 567–574, 2006.

[53] T. Storch. Finding Large Cliques in Sparse Semi-random Graphs by Simple Randomised Search Heuristics, Technical Report No. CI-211/06, Fachbereich Informatik, Universitat Dortmund, 2006.

[54] G. H. Sasaki, B. Hajek. The Time Complexity of Maximum Matching by Simulated Annealing. *Journal of the Association of Computing Machinery*, vol. 35, no. 2, pp. 387–403, 1988.

Pietro S. Oliveto received the Laurea degree in computer science from the University of Catania, Italy, in 2005. Since 2006 he has been a Ph.D. candidate at the School of Computer Science, University of Birmingham, UK.

His Ph.D. topic is the computational complexity analysis of evolutionary algorithms which is part of an EPSRC funded project. His main research interest is the time complexity analysis of randomized algorithms for combinatorial optimization problems. He is currently considering local search, evolutionary, and artificial immune system algorithms.

Jun He received his Ph.D. degree in computer science from Wuhan University, China in 1995. Currently he is a research fellow at the School of Computer Science, University of Birmingham, England.

His research interests include evolutionary computation, data mining and network security.

Xin Yao obtained his B.Sc. from the University of Science and Technology of China (USTC) in Hefei, China, in 1982, M.Sc. from the North China Institute of Computing Technology in Beijing, China, in 1985, and Ph.D. from USTC in Hefei, China, in 1990.

He was an associate lecturer and lecturer between 1985 and 1990 at USTC while working on his Ph.D.. His Ph.D. work on simulated annealing and evolutionary algorithms was awarded the President′s Award for Outstanding Thesis by the Chinese Academy of Sciences. He took up a postdoctoral fellowship in the Computer Sciences Laboratory at the Australian National University (ANU) in Canberra in 1990, and continued his work on simulated annealing and evolutionary algorithms. He joined the Knowledge-Based Systems Group at CSIRO Division of Building, Construction and Engineering in Melbourne in 1991, working primarily on an industrial project on automatic inspection of sewage pipes. He returned to Canberra in 1992 to take up a lectureship in the School of Computer Science, University College, the University of New South Wales (UNSW), the Australian Defence Force Academy (ADFA), where he was later promoted to a senior lecturer and associate professor. Attracted by the English weather, he moved to the University of Birmingham, England, as a professor (chair) of computer science on 1 April 1999. Currently he is the director of CERCIA (the Center of Excellence for Research in Computational Intelligence and Applications) at Birmingham, UK, a distinguished visiting professor of the University of Science and Technology of China in Hefei, China, and a visiting professor of three other universities.

He has more than 200 refereed research publications. In his spare time, he does the voluntary work as the editor-in-chief of *IEEE Transactions on Evolutionary Computation*, an associate editor or editorial board member of several other journals, and the editor of the World Scientific book series on "Advances in Natural Computation". He has been invited to give more than 45 invited keynote and plenary speeches at conferences and workshops world-wide. His major research interests include evolutionary computation, neural network ensembles, and their applications.

Prof. Yao is an IEEE Fellow and a distinguished lecturer of IEEE Computational Intelligence Society. He won the 2001 IEEE Donald G. Fink Prize Paper Award for his work on evolutionary artificial neural networks.