D 1.3 Integrating and developing methodology

The SAGE Consortium

December 31, 2014

<table>
<thead>
<tr>
<th>Project acronym</th>
<th>SAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project title</td>
<td>Speed of Adaptation in Population Genetics and Evolutionary Computation</td>
</tr>
<tr>
<td>Grant agreement</td>
<td>618091</td>
</tr>
<tr>
<td>Call topic</td>
<td>Future and Emerging Technologies</td>
</tr>
<tr>
<td>Project duration</td>
<td>3 years</td>
</tr>
<tr>
<td>Work Package</td>
<td>WP1 Unified model of evolution</td>
</tr>
<tr>
<td>Deliverable</td>
<td>D1.3 Integrating and developing methodology</td>
</tr>
<tr>
<td>Deliverable due date</td>
<td>Month 12</td>
</tr>
<tr>
<td>Submission date</td>
<td>December 31, 2014</td>
</tr>
<tr>
<td>Dissemination level</td>
<td>PU</td>
</tr>
</tbody>
</table>
Integrating and developing methodology

1 Introduction

The objective of deliverable 1.3 is to integrate methods from both Evolutionary Computation (EC) and Population Genetics (PG), assess which of these have the most potential to be useful in problems from the opposite field and, if possible, develop new methodologies of analysis that could lead to new insights in the analysis of problems from both fields.

We flagged three methods with potential for transfer across fields:

1. The diffusion approximation from PG
2. Drift analysis from EC
3. Level based analysis from EC

In the following sections, we describe these tools and the results of our attempts at applying them to the opposite field, with comments on the difficulties encountered and potential avenues of extension of these tools. Furthermore, we describe work that extended these tools further, extending their usefulness in both fields.

Publications

The deliverable resulted in award-winning publications in two major computer science conferences, while a third paper on applying diffusion approximation in evolutionary computation is being written up for submission to GECCO 2015. The work on drift analysis won the best paper award in ISAAC 2014, one of the main conferences on algorithms and computation. The work on level-based analysis was one of eight papers to be nominated for a best paper award at PPSN 2014, one of the main conferences in evolutionary computation.

   - Link to a publicly available copy of the paper
   - DOI link to publisher’s copy
     http://dx.doi.org/10.1007/978-3-319-13075-0_54

   - Link to a publicly available copy of the paper
   - DOI link to publisher’s copy
     http://dx.doi.org/10.1007/978-3-319-10762-2_90

2 A diffusion approximation for EAs

In PG diffusion approximations are a standard tool by which one can analyse different evolutionary scenarios. In a diffusion approximation, one approximates the discrete nature of the state space of the stochastic process (for examples, number of copies of a particular alleles) by a continuous variable and writes a partial differential equation equation that describes the temporal dynamics (diffusion) of the probability density function over
this space. There are two kinds of diffusion equation: the forward equation, which tracks the forward evolution of the probability density function, and the so-called backwards equation, which tracks the probability that the system will be found at a particular state, conditional on being at a particular state at a previous time. These two kinds of diffusion equation are useful in analysing different aspects of the system: The forward equation is useful in describing long term properties of systems that have a well-behaved steady state distribution, whilst the backwards equation is useful for obtaining results about the reachability of specific states, typically of absorbing states (e.g. probabilities of fixation). A typical scenario is one where only one locus is considered and the frequency of one allele in a population of \( N \) individuals is tracked (\( p \)). For the diffusion approximation to hold, all forces acting on \( p \) (selection, mutation, ...) need to be weak, a situation that is typically not encountered in EAs. In this scenario, the forward equation could be used to obtain the steady state probability distribution of finding the population at a specific allele frequency. For the diffusion approximation to hold, all forces acting on \( p \) (selection, mutation, ...) need to be weak, a situation that is typically not encountered in EAs. In this scenario, the forward equation could be used to obtain the steady state probability distribution of finding the population at a specific allele frequency \( p \), whilst the backward equation could be used to calculate the probability that the population is fixed for this particular allele, given an initial allele frequency \( p_0 \). However, in many cases, the diffusion approximation is not so important at deriving specific analytic results but in identifying which parameters are relevant for the process.

In a paper being readied for publication at a EC venue, we take a slightly different approach. We consider EAs that keep just one genotype acting on functions of unitation (fitness functions that depend only on the number of bits set to 1). Because we are considering only functions of unitation, we can perform the diffusion approximation on this variable: \( p = \frac{|X|}{n} \), where \( X \) is a bitstring, and \( n \) is its length. This corresponds to a different interpretation from the one typically done in PG: In PG, \( p \) corresponds to the number of copies of a specific allele in a population of \( N \) individuals, and the diffusion equation tracks the probability of finding one specific population in a given state. Here, \( p \) corresponds to a definite state of one EA, and the diffusion equation tracks a population of replicate runs of this EA.

We then set up a framework to deal with the analysis of processes of this kind. In general, the probability of finding the EA at a given state for EAs of the kind considered here can be described by summing over the probability of mutating into this state multiplied by the probability of accepting this change (the probability of fixation in PG terms). As a first example, we consider a family of EAs such that mutation only flips one bit per generation. In this case, the expected change in \( p \) is \( E(\Delta p) = (1 - p)\alpha(\Delta w)\delta - p\alpha(-\Delta w)\delta \), where \( \Delta w = w_{\text{new}} - w_{\text{old}} \) is the fitness difference \( w(|X|+1) - w(|X|) \), \( \delta = \frac{1}{n} \) and \( \alpha(\Delta w) \) is the probability of accepting a candidate solution. If we further assume that \( \alpha(-\Delta w) = 1 - \alpha(\Delta w) \), the relevant quantities simplify to:

\[
\begin{align*}
E(\Delta p) &= \delta (\alpha(\Delta w) - p) \\
\text{Var}(\Delta p) &= \delta^2 p(1 - p)
\end{align*}
\]

Note that we haven’t specified neither the problem (encoded by \( w(|X|) \)) nor the probability of fixation \( \alpha(\cdot) \) (beyond the assumption that it is symmetric about 0).

In the manuscript, we exemplify how the diffusion equation can be used to analyse this family of EAs by applying it to deceptive problems and to situations that lead to a mutation-selection balance. We are currently working on analysing other situations where this method can also be used such as the \((1, \lambda)\) EA.

This work is being targeted for publication at the GECCO 2015 conference.

3 Drift analysis with tail bounds

Drift analysis is one of the primary methods used in EC to estimate the running time of evolutionary algorithms (EAs). This method has a similar goal as diffusion approximation: to predict long-term properties of a process given some information about how the process changes per time step.

In drift analysis, the evolutionary process is described by a sequence of random variables \((X_t)_{t \geq 0}\) that describes some aspect of the evolutionary process. Typically, \( X_t \) represents how far the EA is from achieving the goal set by its designers at time time \( t_0 \), such as the number of incorrectly set variables in the current solution. One would like to determine some long-term property of the process, such as

\[
\tau_a = \min \{ t \geq 0 \mid X_t \leq a \},
\]

i.e., the number of iterations (or generations) until the process is within some given distance \( a \) from its goal. As such, the random variable \( \tau_a \) can describe the speed of adaptation, and is therefore highly pertinent to the SAGE project.
To say something about the distribution of $\tau_a$, drift analysis assumes some information about how the process changes during one time step. More formally, it is assumed that one has some information about the distribution of the following random variable

\[ \Delta(x) := (X_t - X_{t+1} \mid X_t = x), \]

i.e., how much the process changed between time $t$ and $t+1$, assuming that the process was in state $x$ at time $t$. The random variable $\Delta(x)$ is called the “drift” of the process, thus giving the method its name.

Drift theorems can be classified according to what information they require about the distribution of $\Delta(x)$, and secondly according to how much information they provide about the distribution of the time $\tau_a$. There is a clear trade-off here. The less assumptions one makes about the distribution of $\Delta(x)$, the less one can say about the distribution of $\tau_a$. Most drift theorems assume some knowledge about the expectation of $\Delta(x)$. For example, the additive drift theorem applies when there is some constant $\varepsilon > 0$ such that for all $x \geq a$

\[ E[\Delta(x)] \geq \varepsilon \]  

However, this condition does not always hold, or only hold for small values of $\varepsilon$. In many applications, the expected change is proportional to the current position $x$. The so-called multiplicative drift theorem take this into account, assuming that there exists a constant $\delta \in (0, 1)$ such that for all $x \geq a$

\[ E[\Delta(x)] \geq \delta x \]  

More generally, one can consider the so-called variable drift condition where for some well-behaved function $h : \mathbb{R} \to \mathbb{R}$

\[ E[\Delta(x)] \geq h(x). \]  

Either of the three conditions above translate into some bound on the expectation of the time $\tau_a$ expressed in terms of $\varepsilon, \delta$ or $h$. However, more information about the distribution of $\tau_a$ than its expectation is often desirable. Ideally, one would like to estimate its cumulative distribution function $F_{\tau_a}(t) = \Pr(\tau_a \leq t)$.

In our contribution published in the ISAAC 2014 conference, we presented bounds on the cumulative distribution function of $\tau_a$ given that the process satisfies the variable drift condition (5) above. This result requires additional information about $\Delta(x)$, either in the form of bounds on the moment-generating function of $\Delta(x)$, or in the form of additional constraints on the function $h$. A plethora of different drift theorems have been developed in the literature over the last years. Our theorem generalises virtually all of these theorems. The paper won the best paper award at the ISAAC conference.

## 4 Applying runtime analysis to natural populations

Runtime analysis is the study of the time complexity of algorithms. In EAs, the tools used for this are typically based on drift analysis, which provides bounds on the expected time for an EA to reach a certain target state or states, typically the maximum of the fitness function. These tools typically look at the one-step increase of the algorithm and maximise it or minimise it to produce upper or lower bounds on the total number of iterations required to reach the target state.

We investigated the applicability of this tool to the analysis of models describing natural populations. In particular, we tried to apply it to the SSWM evolutionary regime. This regime was previously identified as very similar to the (1+1) EA, one of the simplest EAs, and the target of much work on runtime analysis in a multitude of problems. The major features of the SSWM regime are that mutations are rare and they sweep to fixation quickly enough that the dynamics can be described by a process of “jumping” between genotypes, in a similar way to the (1+1) EA. The major distinction to the (1+1) EA is that candidate genotypes are “accepted” with probability proportional to the fitness difference to the current genotype, in contrast with the (1+1) EA in which any fitter solution is accepted.

The SSWM regime has been previously analysed, albeit mostly in random fitness landscapes. The promise of these EC-inspired techniques is that they will allow for the analysis of more realistic fitness landscapes. With this in mind, we calculated the one step increase for the SSWM and compared it with the (1+1) EA. As a stepping stone towards more complicated situations, we first consider the case of uniformly accessible multiple alleles (corresponding to a mutational landscape represented by a complete graph). We found that the SSWM regime takes on average smaller steps than the (1+1) EA which is to be expected since many fitter mutations
are rejected in this regime. However, if we condition on the steps that are accepted, the step size of the SSWM is larger than in the (1+1) EA, since the acceptance probability in this regime is biased towards mutations of larger effect.

In particular, the expected step size of all steps \((s \geq 0)\) in the (1+1) EA:

\[
\text{Step}_{((1+1) \text{ EA})} = \int_{\min_x}^{\max_x} s \rho_{A((1+1) \text{ EA})} \rho_O ds = \frac{\rho_O}{2} y^2.
\]

where \(y\) is the fitness difference of the current genotype to the fittest maximum. On the other hand, for the SSWM regime:

\[
\text{Step}_{(\text{SSWM})} = \int_{\min_x}^{\max_x} s \rho_{A(\text{SSWM})} \rho_O ds = \frac{\rho_O}{3} y^2.
\]

When looking at all steps, SSWM steps are smaller on average than those by the (1+1) EA. However, this is only because we also consider steps that are rejected. To find the size of an expected accepted step, we have to divide it by the fraction of accepted steps:

In the (1+1) EA:

\[
F_{((1+1) \text{ EA})} = \int_{\min_x}^{\max_x} \rho_{A((1+1) \text{ EA})} \rho_O ds = \rho_O y.
\]

In SSWM:

\[
F_{\text{SSWM}} = \int_{0}^{\max_x} \rho_{A(\text{SSWM})} \rho_O ds = \frac{\rho_O y}{2}.
\]

Therefore, the expected size of accepted steps in the (1+1) EA is

\[
\text{Step}_{A((1+1) \text{ EA})} = \frac{1}{2} y
\]

and in SSWM it is

\[
\text{Step}_{A(\text{SSWM})} = \frac{2}{3} y.
\]

This last quantity is of considerable interested for evolutionary biology since we typically only have information about the mutations that fixed, i.e., the accepted steps. It should be noted that this derivation does not depend on assumptions about the distribution of the fitness values in this landscape, in contrast to previous analyses in which extreme value theory was assumed.

We then moved on to a more complex, albeit still one of the simplest, mutational landscape: the so-called OneMax. In this setting we consider a bitstring, or a genotype of \(n\) diallelic loci, and a fitness function that assigns to each genotype a fitness value equal to the number of 1s in this bitstring.

5 Level based analysis

The fitness-level technique is one of the oldest and simplest techniques to analyse the runtime of evolutionary algorithms [7]. The technique was originally developed for the (1+1) EA and other simple evolutionary algorithms which have negligible population sizes. The idea is to partition the set of genotypes into levels which are ordered according to fitness values. The maximum level should contain only those genotypes that the algorithm has been set out to find, i.e. optimal solutions or solutions of satisfactory approximation quality. Bounds on the
runtime can then be obtained from estimates of the transition probabilities between the levels. The reciprocal of a lower bound on the probability that the algorithm transitions from a given level to a higher one is an upper bound on the expected time that the algorithm stays on that level. The sum of these reciprocals is therefore an upper bound on the total expected runtime. This approach often provides good upper bounds on the expected runtime. SAGE member Sudholt proposed a further enhancement and extension of the fitness level technique based on analysis of average jumps [6]. His refined technique yields both lower and upper bounds on the expected runtime.

In a sequence of papers [2, 3, 4], we have generalised the fitness level technique to the setting of evolutionary algorithms with realistic population sizes. Initially, our results were restricted to evolutionary algorithms using unary variation operators, e.g. mutation but not recombination. From these studies, we concluded that the expected time to optimise a function depends critically on the balance between the selective pressure imposed by the selection mechanisms and the amount of variation introduced by variation operators. When the selective pressure is too low compared to a threshold which is defined by the amount of variation, the algorithm needs exponential expected time to travel through the levels to optimise the function [4]. Conversely, when the selective pressure is above the threshold, it is possible to show an upper bound on the runtime [3]. In particular, we have found that the degree of selective pressure only needs to be marginally above the threshold in order to guarantee polynomial expected runtime. This observation enabled us to identify specific settings of non-elitist EAs that guarantee in expected polynomial optimisation time in the scenarios of optimisation under uncertainty [2]. The settings often imply a sufficiently large population and a small enough mutation rate. One can argue that these settings share many similarities with biological populations.

Following up from these papers, we have generalised the method to a much broader class of evolutionary processes [1]. Our general result is stated for evolutionary processes described at a higher level of abstraction, the so-called distribution one (also discussed in deliverable 1.1, unified model of evolution): at a given time \( t \), each new individual \( i \) of the next population \( P_{t+1} \) is generated by sampling independently from a distribution \( D(P_t) \) which is parameterised by the current population \( P_t \). Our theorem requires three properties that \( D(P_t) \) should satisfy for any \( t \): (i) a sufficient pressure (or probability) to create new individuals at levels strictly higher than \( j \) when a constant fraction of the population is already at \( j \); (ii) the more individuals at levels strictly higher than \( j \) are created, the more they reproduce, e.g. with some multiplicative factor \((1 + \delta)\) for some \(\delta > 0\); and (iii) a sufficiently large population. As an illustration for the power of such general statement, an analysis on the runtime of Simple Genetic Algorithms with various recombination operators and selection mechanisms on standard functions like \textsc{OneMax} and \textsc{LeadingOnes} were performed. Note that such analysis had not been possible in our previous work [3] because of the restriction to unary variation operators. The paper describing the new level-based technique was one of eight papers to be nominated for a best paper award at the PPSN 2014 conference.

In ongoing work, we are applying the new level-based methods to population genetics models of evolution in rugged fitness landscapes. Also, we will further analyse speed of adaptation with recombination operators in WP3.
Bibliography


Introduction

- Runtime Analysis of Evolutionary Algorithms
  - Rigorous analysis of how optimisation time depends on
    - structure of the fitness landscape,
    - configuration of the algorithm (parameter settings)
  - makes extensive use of a small set of analytical tools
  - analysis of how optimisation time depends on
- Runtime analysis traditionally limited to simple EAs
  - populations and crossover most often not considered
    - typically (1+1) EA
  - tools have been lacking for more complex algorithms
- More recently non-elitist population-based EAs
  - fitness levels, drift analysis, measure concentration...
- Limited to unary variation operators
  - which handles large populations
  - such as crossover operators
- This Paper: Level-based Runtime Analysis
  - a new analytical tool that makes runtime analysis of complex evolutionary algorithms easy to see
  - handles complex high-order variation operators, such as crossover operators
  - includes non-trivial problems

Toy Example

- Search space \( X = \{1, \ldots, n\} \) and fitness \( f(x) = x \).
- \((\mu, \lambda)\) EA with uniform mutation and \( \lambda = \mu/2 \)

For each generation \( t \geq 0 \) do

for \( z \) in \( 1, \ldots, \lambda \) do

Sample \( y \) from the best \( \lambda/2 \) individuals in \( P_t \).
Set \( P_{t+1}(i) = y \) if \( U(0,1) < \lambda/(\mu+1) \).

Runtime with Crossover

- In Genetic Algorithms, \( D \) is typically
  - recombin two selected individuals and mutate
- Sufficient to determine when \( D \) satisfies (G1)-(G3)
  - properties of mutation and crossover operator
  - and strength of selection mechanism

Corollary 1

If conditions (C1)-(C5) hold, then (G1)-(G3) are satisfied.

Results on Example Problems

<table>
<thead>
<tr>
<th>Selection Mechanism</th>
<th>w/o crossover</th>
<th>w/ crossover</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tournament</td>
<td>( k &lt; e^x )</td>
<td>( k &lt; e^x )</td>
</tr>
<tr>
<td>(mu)-Selection</td>
<td>( 1 &lt; e^x )</td>
<td>( 1 &lt; e^x )</td>
</tr>
<tr>
<td>Offspring</td>
<td>( 1 &lt; e^x )</td>
<td>( 1 &lt; e^x )</td>
</tr>
<tr>
<td>LeadingOnes</td>
<td>( \Omega(n \ln \lambda + n^2) )</td>
<td>( \Omega(n \ln (n + \lambda)) )</td>
</tr>
<tr>
<td>Sorting (RIV)</td>
<td>( \Omega(n^2 \ln n) )</td>
<td>( \Omega(n^2 \ln n) )</td>
</tr>
</tbody>
</table>

Note: Sorting problem requires larger selection pressure with a multiplier of \( \Omega(n) \) where \( p_0 \) is the constant probability of applying crossover in each generation.

Future Work

- analysis of broad problem classes
- problem-specific crossover operators
- estimation of distribution algorithms (EDAs), genetic programming (GP)
- more precise analysis of population dynamics
- eg population diversity

Conclusion

- A Tool for Runtime Analysis of Complex Processes
  - combines high level of generality...
  - \( D \) can represent most selection and variation processes...
  - with simplicity of application

- Runtime Analysis of EAs with Crossover
  - the crossover operator beneficial in
    - combining best and majority individuals
    - compared to mutation-only EAs, the theorem imposes
      a stronger selective pressure, and
    - a larger population size
    - similar asymptotic runtime on simple problems

- Future Work

Figure 1: Poster of \( \Pi \) presented at PPSN 2014.