Tutorial: Drift Analysis

Benjamin Doerr

Max-Planck-Institut für Informatik
Saarbrücken

GECCO'11, July 12–16, 2011, Dublin, Ireland.
ACM 978-1-4503-0690-4/11/07.

Bio-Sketch

- Benjamin Doerr is a senior researcher at the Max Planck Institute for Computer Science and a professor at Saarland University.
- He received his diploma (1998), PhD (2000) and habilitation (2005) in mathematics from Kiel University.
- Together with Frank Neumann and Ingo Wegener, he founded the theory track at GECCO and served as its co-chair 2007-2009.
- He is a member of the editorial boards of Evolutionary Computation and Information Processing Letters.
- His research area includes theoretical aspects of randomized search heuristics, in particular, run-time analysis and complexity theory.

Agenda

- Motivation and a simple drift result
- Four applications in evolutionary computation theory
  - Coupon collector
  - RLS and (1+1) EA optimize OneMax
  - RLS and (1+1) EA optimize linear functions
  - Finding mininum spanning trees
- More drift methods
- Summary, directions for future research

Objectives of the Tutorial

- This is a tutorial on drift analysis, which is one of the strongest methods in the theory of randomized search heuristics.
- I shall try my best to..
  - tell you on a very elementary level what drift analysis can do for you
  - use a series of examples from easy to advanced to demonstrate how to use drift analysis
  - sketch the other main methods in this area
  - give a one-slide summary of the most important facts
  - sketch a few directions for further research

Drift Analysis: What's the Problem?

- Example from everyday life... ☺
  - Get salary on day 0: X_0 = 1000 €
  - Day 1: Spend half of it in the pub: $X_1 = ½ X_0 = 500$
  - Day 2: Spend half of your money: $X_2 = ½ X_1 = 250$
  - …
  - Day t: Spend half of your money: $X_t = ½ X_{t-1}$
  - Question: When are you broke ($X_T < 1$)?
  - Answer: $T = \lfloor \log_2(X_0) + 1 \rfloor = 10$

Problem: Randomness

- Everyday life is not so regular (lots of randomness)
  - Get salary on day 0: X_0 = 1000 €
  - Day 1: Expect to spend half of it: $E(X_1) = ½ X_0 = 500$
  - Day 2: Expect to spend half of your money: $E(X_2) = ½ X_1$
  - …
  - Day t: Expect to spend half of your money: $E(X_t) = ½ X_{t-1}$
  - Question: When do you expect to be broke?
  - Hope: $E(T) = \lfloor \log_2(X_0) + 1 \rfloor = 10$
  - Warning: You hope for $E(\min\{T|X_T<1\}) = \min\{T|E(X_T)<1\} = 10$
  - "Hope" does not work in theory
  - Solution: Drift theorems

Truth: 10.95 is possible

$E(M_t) = (1/2)^t M_0$

A Drift Theorem

Theorem. Let $X_0, X_1, \ldots$ be non-negative integer random variables. Assume that there is a $\delta > 0$ such that

(1) $\forall t \in \mathbb{N}, x \in \mathbb{N}_0 : E(X_t|X_{t-1} = x) \leq (1-\delta)x.$

Then $T := \min\{t \in \mathbb{N}_0 | X_t = 0\}$ satisfies

$$E(T) \leq (1/\delta)(\ln(X_0)+1).$$

A Drift Theorem

Theorem. Let $X_0, X_1, \ldots$ be non-negative integer random variables. Assume that there is a $\delta > 0$ such that

(1) $\forall t \in \mathbb{N}, x \in \mathbb{N}_0 : E(X_t|X_{t-1} = x) \leq (1-\delta)x.$

Then $T := \min\{t \in \mathbb{N}_0 | X_t = 0\}$ satisfies

$$E(T) \leq (1/\delta)(\ln(X_0)+1).$$

Note. (i) This is clearly not the first drift theorem ever found, but a very useful one.
(ii) This "multiplicative" drift approach was first suggested by Daniel Johannsen. It was first published in BD, Johannsen, Winzen [GECCO'10].

A Drift Theorem

Theorem. Let $X_0, X_1, \ldots$ be non-negative integer random variables. Assume that there is a $\delta > 0$ such that

(1) $\quad \forall t \in \mathbb{N}, x \in \mathbb{N}_0 : E(X_t | X_{t-1} = x) \le (1-\delta)x.$

Then $T := \min\{t \in \mathbb{N}_0 | X_t = 0\}$ satisfies

$$E(T) \le (1/\delta)(\ln(X_0) + 1).$$

Note. For δ small and X_0 large, the theorem gives an approximate upper bound version of "hope":

$$E(\min\{t \in \mathbb{N}_0 | X_t < 1\}) = E(T) \le (1/\delta)(\ln(X_0)+1)$$
$$\approx \lfloor \log_{1/(1-\delta)}(X_0) + 1 \rfloor = \min\{t | E(X_t) < 1\}$$

Agenda

- Just seen: Motivation—Why drift analysis?
  - Situation: You expect some progress every iteration
  - Drift theorems: "Things are (roughly) as you hoped for"
    - The expected time to reach your goal (roughly) is at most the time needed to collect an expected progress equal to the distance from your goal.
- Next: Four applications from evolutionary computation theory
  - a slightly improved drift theorem
  - Coupon collector
  - OneMax
  - Linear functions
  - Minimum spanning trees

An Improved Drift Theorem

Theorem (BD, L. Goldberg [PPSN'10]). Let $X_0, X_1, \ldots$ be random variables taking values in $\{0\} \cup [1, \infty[$. Assume that there is a $\delta > 0$ such that

(1) $\quad \forall t \in \mathbb{N}, x \in \mathbb{N}_0 : E(X_t | X_{t-1} = x) \le (1-\delta)x.$

Then $T := \min\{t \in \mathbb{N}_0 | X_t = 0\}$ satisfies

(i) $E(T) \le (1/\delta)(\ln(X_0) + 1)$;

(ii) for all $c > 0$, $\Pr(T > (1/\delta)(\ln(X_0) + c) \le e^{-c}$.

Note. Adds the "tail bound" (ii) to what we had before (for free).

Application 1: Coupon Collector

- Coupon Collector Problem:
  - There are n different types of coupons: $T_1, \ldots, T_n$
  - Round 0: You start with no coupon
  - Each round t, you obtain a random coupon C_t
    - $Pr(C_t = T_k) = 1/n$ for all t and k
  - After how many rounds do you have all types of coupons?
- Analysis:
  - X_t := Number of missing coupon types after round t; $X_0 = n$.
  - Question: Smallest T such that $X_T = 0$.
  - If $X_{t-1} = k$, then the chance to get a new coupon in round t is k/n. Drift: $E(X_t | X_{t-1}=k) = (k/n)(k-1) + (1-k/n)k = (1 - 1/n)k$.
  - Drift-Thm gives:
    - $E(T) \le (1/\delta)(\ln(x_0) + 1) = n\,(\ln(n)+1)$
    - For all $\beta>0$, $Pr(T > (\beta+1)\, n \ln(n)) < n^{-\beta}$

Matches the best known bounds, except: the "+1" could be made a "+0.577... + $o(1)$"

Application 2: RLS optimizes OneMax

- One of the most simple randomized search heuristics (RSH): Randomized Local Search (RLS), here used to maximize f: $\{0,1\}^n \to$ R

RLS:
1. Pick $x \in \{0,1\}^n$ uniformly at random % random start-point
2. Pick $i \in \{1, \ldots, n\}$ uniformly at random
3. $y := x$; $y_i := 1 - x_i$ % mutation: flip a random bit
4. if $f(y) \geq f(x)$, then $x := y$ % selection: keep the fitter
5. if not happy, go to 2. % repeat or terminate

- Question: How long does it take to find the maximum of a simple function like OneMax = f: $\{0,1\}^n \to$ R; $x \mapsto x_1 + x_2 + \ldots + x_n$ (number of 'ones' in x)
- Remark: Of course, $x = (1, 1, \ldots, 1)$ is the maximum, and no-one needs an algorithm to find this out.
  Aim: Start understanding RSH via simple examples

Application 2: RLS optimizes OneMax

- RLS:
  1. Pick $x \in \{0,1\}^n$ uniformly at random % random start-point
  2. Pick $i \in \{1, \ldots, n\}$ uniformly at random
  3. $y := x$; $y_i := 1 - x_i$ % mutation: flip a random bit
  4. if $f(y) \geq f(x)$, then $x := y$ % selection: keep the fitter
  5. if not happy, go to 2. % repeat or terminate
- Question: How long does it take to find the maximum of a simple function like OneMax = f: $\{0,1\}^n \to$ R; $x \mapsto x_1 + x_2 + \ldots + x_n$ (number of 'ones' in x)
- Analysis:
  - X_t: Number of zero-bits after iteration t (= "$f_{\text{opt}} - f(x)$"). Trivially, $X_0 \leq n$
  - If $X_{t-1} = k$, then with probability k/n, we flip a 'zero' into a 'one', giving $X_t = k - 1$. Otherwise, y is worse than x and thus $X_t = k$
  - As before: $E(X_t | X_{t-1}=k) = (k/n)(k-1) + (1-k/n)k = (1 - 1/n)\, k$ "drift!"
  - Drift Thm gives: Maximum found after $n(\ln(n) + 1)$ iterations (in expect.)

Bottom line: "f-distance to optimum" often good drift measure

Application 2a: (1+1)-EA optimizes OneMax

- One of the most simple evolutionary algorithms (EAs): (1+1)-EA, again used to maximize f: $\{0,1\}^n \to$ R

(1+1)-EA:
1. Pick $x \in \{0,1\}^n$ uniformly at random % random start-point
2. $y := x$
3. For each $i \in \{1, \ldots, n\}$ do % mutation: Flip each bit w.p. $1/n$
   with probability $1/n$ set $y_i := 1 - x_i$
4. if $f(y) \geq f(x)$, then $x := y$ % selection: keep the fitter
5. if not happy, go to 2. % repeat or terminate

- '(1+1)': population size = 1, generate 1 off-spring, perform 'plus'-selection: choose new population from parents and off-springs
- Cannot get stuck in local optima ("always converges").
- Question: Time to maximize OneMax = f: $\{0,1\}^n \to$ R; $x \mapsto x_1 + \ldots + x_n$?

Application 2a: (1+1)-EA optimizes OneMax

- (1+1)-EA:
  1. Pick $x \in \{0,1\}^n$ uniformly at random % random start-point
  2. $y := x$
  3. For each $i \in \{1, \ldots, n\}$ do % mutation: Flip each bit w.p. $1/n$
     with probability $1/n$ set $y_i := 1 - x_i$
  4. if $f(y) \geq f(x)$, then $x := y$ % selection: keep the fitter
  5. if not happy, go to 2. % repeat or terminate
- Question: Time to maximize OneMax = f: $\{0,1\}^n \to$ R; $x \mapsto x_1 + \ldots + x_n$?
- Analysis:
  - X_t: Number of zeroes after iteration t ("f-distance")
  - If $X_{t-1} = k$, then the probability that exactly one of the missing bits is flipped, is $k\,(1/n)\,(1 - 1/n)^{n-1} \geq (1/e)\,(k/n)$. Otherwise, $X_t \leq k$
  - Hence, $E(X_t | X_{t-1}=k) \leq (k - 1)(k/en) + k(1 - k/en) = k\,(1 - 1/en)$
  - Drift Thm: Expected optimization time at most $en(\ln(n) + 1)$

Bottom line: A weak bound on the drift may suffice

A 3: RLS optimizes Linear Functions

- RLS: 1. Pick $x \in \{0,1\}^n$ uniformly at random % random start-point
  2. Pick $i \in \{1, \dots, n\}$ uniformly at random
  3. $y := x$; $y_i := 1 - x_i$ % mutation: flip a random bit
  4. if $f(y) \geq f(x)$, then $x := y$ % selection: keep the fitter
  5. if not happy, go to 2. % repeat or terminate
- Question: How long does it take to find the maximum of an arbitrary linear function f: $\{0,1\}^n \to$ R; $x \mapsto a_1x_1 + a_2x_2 + \dots + a_nx_n$ (wlog $0<a_1\leq a_2\leq\dots\leq a_n$)
- Analysis:
  - X_t: Number of zeroes after iteration t. Trivially, $X_0 \leq n$
  - As for OneMax: If $X_{t-1} = k$, then with probability k/n, we flip a 'zero' into a 'one' ($X_t = k - 1$). Otherwise, y is worse than x and thus $X_t = k$
  - Message: You can use X_t different from "$f_{\text{opt}} - f(x_t)$"!
  - Why not X_t = "$f_{\text{opt}} - f(x_t)$"? Drift Thm: $E(T) \leq (1/\delta)(\ln(X_0) + 1)$, and X_0 can be large!

Bottom line: Choose the right drift measure!

A 3a: (1+1)-EA optimizes Linear Functions

- (1+1)-EA: 1. Pick $x \in \{0,1\}^n$ uniformly at random % random start-point
  2. $y := x$
  3. For each $i \in \{1, \dots, n\}$ do % mutation: Flip each bit w.p. $1/n$
     with probability $1/n$ set $y_i := 1 - x_i$
  4. if $f(y) \geq f(x)$, then $x := y$ % selection: keep the fitter
  5. if not happy, go to 2. % repeat or terminate
- Maximize f:$\{0,1\}^n \to$ R; $x \mapsto a_1y_1 + a_2x_2 + \dots + a_nx_n$ (wlog $0<a_1\leq a_2\leq\dots\leq a_n$) !
- Classic difficult problem
  - Droste, Jansen, Wegener (2002): Exp. opt. time $E(T) = O(n \log n)$
  - He, Yao (2001-04): $E(T) = O(n \log n)$ via "additive" drift analysis
  - Jägersküpper (2008): $E(T) \lesssim 2.02\, e\, n \ln(n)$ via "average" drift analysis
  - BD, Johannsen, Winzen (2010): $e\, n \ln(n) \lesssim E(T) \lesssim 1.39\, e\, n \ln(n)$
  - BD, L. Goldberg (2010): $O(n \log n)$ whp for any c/n mutation probability

A 3a: (1+1)-EA optimizes Linear Functions

- (1+1)-EA: 1. Pick $x \in \{0,1\}^n$ uniformly at random % random start-point
  2. $y := x$
  3. For each $i \in \{1, \dots, n\}$ do % mutation: Flip each bit w.p. $1/n$
     with probability $1/n$ set $y_i := 1 - x_i$
  4. if $f(y) \geq f(x)$, then $x := y$ % selection: keep the fitter
  5. if not happy, go to 2. % repeat or terminate
- Maximize f:$\{0,1\}^n \to$ R; $x \mapsto a_1x_1 + a_2x_2 + \dots + a_nx_n$ (wlog $0<a_1\leq a_2\leq\dots\leq a_n$) !
- Difficulty: What drift? [E.g., $f(x) = x_1 + 2x_2 + 4x_3 + 8x_4 + \dots + 2^nx_n$]
  - $X_t := f_{\text{opt}} - f$ $\Rightarrow$ X_0 can be too large (as before)
  - X_t := number of zero-bits $\Rightarrow$ drift too small (?)
    - Example: f as above, $x_1 = x_2 = \dots x_{n-1} = 1$, $x_n = 0$, $X_{t-1} = 1$
    - Progress if and only if the n^{th} bit flips (otherwise $X_t = 1$)
    - If n^{th} bit flips, in expectation $(n-1)/n$ other bits flip back (exp. gain: $1/n$)
    - $E(X_t) = 1 - P(n^{\text{th}}$ bit flips$)\, E(X_t | n^{\text{th}}$ bit flips$) = 1 - (1/n)(1/n) = (1-(1/n^2))\, X_{t-1}$

$\delta = 1/n^2$ is not enough, leads to $O(n^2 \log(n))$ time

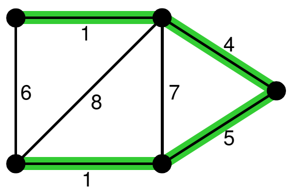
A 3a: (1+1)-EA optimizes Linear Functions

- (1+1)-EA: 1. Pick $x \in \{0,1\}^n$ uniformly at random % random start-point
  2. $y := x$
  3. For each $i \in \{1, \dots, n\}$ do % mutation: Flip each bit w.p. $1/n$
     with probability $1/n$ set $y_i := 1 - x_i$
  4. if $f(y) \geq f(x)$, then $x := y$ % selection: keep the fitter
  5. if not happy, go to 2. % repeat or terminate
- Maximize f:$\{0,1\}^n \to$ R; $x \mapsto a_1x_1 + a_2x_2 + \dots + a_nx_n$ (wlog $0<a_1\leq a_2\leq\dots\leq a_n$) !
- Solution (sketched, using ideas from [DJW02], [HY02], [DJW10]):
  - X_t: $x_1 + \dots + x_{\lfloor n/2 \rfloor} + (5/4)\, x_{\lfloor n/2+1 \rfloor} + \dots + (5/4)\, x_n$ for the x after iteration t
  - Compute: If $X_{t-1} = k$, then $E(X_t) \leq (1 - 0.01/n)\, k$. [less than 1 page]
  - Drift Thm: Optimization time is $O(n \log n)$ with high probability.
  - Note: (i) These X_t work for all linear functions ☺
    (ii) Alternative: "Average drift" argument Jägersküpper [PPSN'08]

Bottom line: Choose a clever drift measure!

Application 4: (1+1)-EA optimizes MST

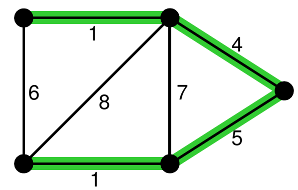
- Minimum Spanning Tree (MST) problem:
  - Input: Undirected connected graph $G = (V, E)$, edge weights (w_e) in N
  - Task: Compute a connected spanning subgraph $T = (V, E')$ of G with minimal weight $w(T) = \sum_{e \in E'} w_e$



- RSH for combinatorial optimization problems – new aspects
  - How to represent the solutions? E.g. bit-strings, permutations, …
  - What is a good mutation operator for this representation?
  - Possibly: Use a clever fitness function f.

Application 4: (1+1)-EA optimizes MST

- Minimum Spanning Tree (MST) problem:
  - Input: Undirected connected graph $G = (V, E)$, edge weights (w_e) in N
  - Task: Compute a connected spanning subgraph $T = (V, E')$ of G with minimal weight $w(T) = \sum_{e \in E'} w_e$



$f(T) = 1+4+5+1 = 11$

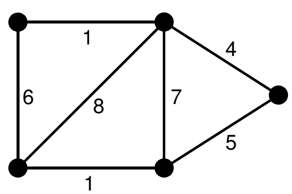
$f(T) = 1+6+8+5+c_{\text{penalty}} =$ HUGE

- Here: Mostly standard
  - Representation: Bitstring x of length $m = |E|$, $x_e = 1$ if $e \in T$
  - Mutation: Standard bit mutation (flip each bit w.p. $1/m$)
  - Fitness function (to be minimized): $w(T) + c_{\text{penalty}}$(#components of $T - 1$)

Application 4: (1+1)-EA optimizes MST

- (1+1)-EA:
  1. Pick $x \in \{0,1\}^m$ uniformly at random % random start-point
  2. $y := x$
  3. For each $i \in \{1, …, m\}$ do % mutation: Flip each bit w.p. $1/m$
     with probability $1/m$ set $y_i := 1 - x_i$
  4. if $f(y) \geq f(x)$, then $x := y$ % $f(x) = w(T) + c_{\text{penalty}}$(#comp-1)
  5. if not happy, go to 2. % repeat or terminate

- Theorem [Neumann, Wegener (2004)]: The expected optimization time of the (1+1) EA searching for an MST is $O(m^2 \log(m w_{\max}))$

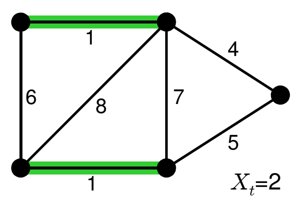


- Proof: Expected weight decrease method
- Next: Drift theorem (plus many arguments of [NW04]) yields same bound, plus tail bounds, with simpler proof

Application 4: (1+1)-EA optimizes MST

- (1+1)-EA:
  1. Pick $x \in \{0,1\}^m$ uniformly at random % random start-point
  2. $y := x$
  3. For each $i \in \{1, …, m\}$ do % mutation: Flip each bit w.p. $1/m$
     with probability $1/m$ set $y_i := 1 - x_i$
  4. if $f(y) \geq f(x)$, then $x := y$ % $f(x) = w(T) + c_{\text{penalty}}$(#comp-1)
  5. if not happy, go to 2. % repeat or terminate

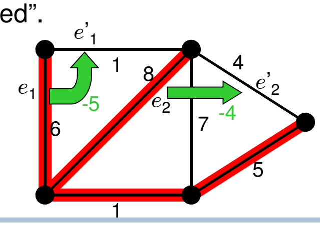
- Analysis (1): After $O(m \log m)$ iterations, T is connected w.h.p.:
  - X_t = #comp – 1 after iteration t
  - If $X_{t-1} = k > 0$, then there are at least k edges that
    - are all not in T
    - adding each one decreases X_t
  - $E(X_t) = (1 - 1/em)\, k$ as before. Done with Drift Thm, since $X_0 \leq m$.



X_t=2

Application 4: (1+1)-EA optimizes MST

- Analysis (2): Let T be already connected. Then it stays connected. And after $O(m^2 \log(mw_{\max}))$ iterations, w.h.p. $w(T)$ is minimal.
  - $X_t = w(T) - w_{\text{opt}}$ for the T after iteration t
  - If $X_{t-1} = D > 0$, then there are $e_1, \ldots, e_k$ in T and $e'_1, \ldots, e'_k$ in $E \setminus T$ s.t.
    - $T' = T - \{e_1, \ldots, e_k\} + \{e'_1, \ldots, e'_k\}$ is an MST,
    - hence $x = \sum_i (w(e_i) - w(e'_i))$, and
    - for all i, $T_i = T - e_i + e'_i$ is a spanning tree with $w(T_i) < w(T)$
  - With prob. $\geq 1/em^2$, one iteration flips exactly the edges e_i and e'_i. These are disjoint events that are "accepted".
  - $E(X_t) \leq D - \sum_i (1/em^2)\,(w(e_i) - w(e'_i))$
    $= (1 - 1/em^2)\,D$
  - Done with drift theorem, since $X_0 \leq \sum_{e \in E} w(e) \leq m\, w_{\max}$.



$X_t = 6+8+1+5 - 11 = 9$

Agenda

- Drift analysis: From expected progress to expected run-time.
- Just seen: Four applications from evolutionary computation theory
  - Coupon collector, OneMax, linear functions, minimum spanning trees
- Next: More drift methods and related stuff
  - Roots
  - Artificial fitness functions
  - Additive drift analysis
  - Lower bounds & negative drift
  - Average drift
  - Adaptive drift
- Summary, directions for future research

The Roots, Artificial Fitness Functions

- While natural, drift analysis ("expected progress ⇒ expected run-time") builds on substantial maths developed, e.g., by Wald (1944), Doob (1956), Tweedie (1976), Hajek (1982) and many others. See, e.g.,
  - Dyer, M., Greenhill, C.: Random walks on combinatorial objects. In: Surveys in Combinatorics 1999, University Press (1999) 101-136
- "Artificial fitness functions"
  - Analyze the progress of an EA by looking at the progress with respect to a potential function different from the fitness
  - First done without drift analysis in by Droste&Jansen&Wegener (2002)
  - Works often well with drift arguments ("choose a drift measure different from the fitness")

Additive Drift

- He&Yao (2001-04): First explicit use of drift analysis in EA theory.
  - Used to give a simpler and more insightful proof of the $O(n \log n)$ run-time of the (1+1) EA optimizing linear functions.
- Additive Drift: Transform an additive expected progress into a run-time
  - Start with 1000 Euros, spend at least 10 Euros each night on beer, and you're broke after at most 100 nights.
  - Start with 1000 Euros, spend in expectation at least10 Euros each night (until you're broke). When do you expect this to happen?
    - Yipiieh, the hoped for at most 100 nights are true due to complicated maths

Additive Drift: Details

Theorem (Hajek 1982, He&Yao 2001). Let $X_0, X_1, \dots$ be random variables describing a Markov process over a finite state space $\mathcal{S} \subseteq \mathbb{R}$. Let T be the random variable that denotes the earliest point in time $t \in \mathbb{N}_0$ such that $X_t \leq 0$. If there exist $\delta > 0$ and $c > 0$ such that

(i) $E(X_t - X_{t+1} \mid X_t) \geq \delta$ for all $t < T$,

(ii) $X_0 \leq c$.

Then $E(T) \leq \frac{c}{\delta}$.

Additive vs. Multiplicative Drift

- Additive drift is strongest when you expect a steady (uniform) progress
  - "spend 10 Euros each night"
  - (1+1) EA optimizes LeadingOnes, single-source shortest paths
- Multiplicative drift is strongest when the progress is proportional to the distance from the optimum
  - "spend half your money each night"
  - natural: progress is easier when further away from optimum
  - (1+1) EA optimizes OneMax, MST, Eulerian cycles, …
- The expected-time bound in the multiplicative setting can be derived from the additive drift theorem
- Tail bounds do not hold in the additive setting

Lower Bounds (sketch)

- Additive drift theorem also works for lower bounds:
  - "Start with 1000 Euros. If you expect to spend at most 10 Euros a night, then the expected time you're broke is in at least 100 nights"
  - Details: Exchange "≤" and "≥" in (i), (ii) and the conclusion of the additive drift theorem.
- Negative (additive) drift (He&Yao, Giel&Lehre, Happ&Johannsen&Klein &Neumann, Oliveto&Witt)
  - "Start with 1000 Euros. If you expect to earn 10 Euros a night, how unlikely is it that you're broke within the next 100 years?"
  - Needs some extra assumptions that "big losses are very unlikely"
- Currently no such results for multiplicative drift

Point-wise vs. Average Drift (sketch)

- In the language of EA: Let x_0, x_1, x_2… be a sequence of search points computed by some RSH. Let g be a potential function.
- All drift theorems shown above…
  - only need something like that at all times t, the random search points x_t and x_{t+1} satisfy $E(g(x_t) - g(x_{t+1}) \mid g(x_t) > 0) \geq \delta$
  - but have only been applied using the stronger assumption of "point-wise drift":
    - for all search points x, $E(g(x_t) - g(x_{t+1}) \mid x_t = x) \geq \delta$
  - Advantage: If you can show point-wise drift, you don't have to care about the distribution of the random search point x_t
  - Problem: You need to show good drift for every search point, even those occurring rarely

Point-wise vs. Average Drift (2)

- The first to use less than point-wise drift was Jägersküpper (PPSN'08).
- Technical result: You can take the number of wrong bits as drift measure in the linear functions problem!
  - Let $x_0, x_1, x_2, \ldots$ be the sequence of search points stored by the (1+1) EA optimizing a linear function after each iteration. Let g = OneMax.
  - Then $E(g(x_t) - g(x_{t+1}) \mid g(x_t)) \geq c/n$, c some explicit constant.
- Advantages:
  - More natural proof
  - First reasonable constant for the total run-time:
    - 2.02 $e\, n$ ln(n) (1+o(1))
    - constant improved to 1.39 by DJW10 using J's drift estimate together with multiplicative drift

Adaptive Drift

- Problem: If the mutation rate p is higher than 7/n, then there are not drift measures that work for all linear functions [DJW10]
  - Consequence: Not clear if the run-time is still $O(n \log n)$
- Solution: For each mutation rate p and each linear function f take a custom-tailored drift measure [DG10]
  - Result: The (1+1) EA with mutation rate $p = c/n$, c any constant, finds the optimum of any linear function in time $O(n \log n)$.
  - Warning: Custom-tailors are not cheap…
  - Bonus result: Same approach shows that the classic (1+1) EA with p = 1/n finds the optimum of BinaryValue in time $e\, n$ ln(n) (1±o(1))
    - the same time as for OneMax ☺

Summary

- Drift analysis: Show an expected progress and gain an expected run-time!
- Several drift theorems:
  - additive: good when uniform progress
    - also yields lower bounds
  - multiplicative: good when progress proportional to distance from goal
    - also tail bounds: "with probability at least 1-exp(-…)…"
- Crucial: How to measure "progress"?
  - simple & good: fitness
  - using structural properties, e.g., "number of wrong bits"
  - clever, e.g., important half of bits counts 5/4, others only 1.
  - average drift: avoid problems with rare exceptions
  - adaptive: custom-tailored measure for each instance

Open Problems (1)

- Tight bounds for combinatorial problems
  - Minimum spanning tree
    - Using fitness as progress measure, above I showed that the (1+1) EA finds an MST in time $O(m^2 \log(m w_{max}))$
    - Does a better measure show $O(m^2 \log(m))$, which is the current best lower bound?
  - Same question for the single-criterion formulation of the single-source shortest path problem
    - With fitness as progress measure: $O(n^3 \log(n w_{max})$
    - Best known lower bound $O(n^3 \log(n))$

Open Problems (2)

- Drift techniques:
  - Multiplicative drift & lower bounds
    - Not true: $E(X_{t+1}|X_t) \geq (1\text{-}\delta)\, X_t \Rightarrow E(T) \geq (1/\delta)\, (\ln(X_0)+1)$
    - Something like this should be true if the X_t behave nicely, i.e., tend to be close to their expectation
  - Additive drift & tail bounds
    - Additive drift allows bounds on expected hitting times, but no good tail bounds ("with high probability…")
    - Tail bounds should hold if the X_t behave nicely
  - Note: In all non-artificial problems, progress behaves nicely

Thanks a lot!

References

- Benjamin Doerr and Leslie Goldberg. Adaptive drift analysis. In Proceedings of Parallel Problem Solving from Nature (PPSN XI), LNCS 6238, pages 32-41. Springer, 2010.
- Benjamin Doerr and Leslie A. Goldberg. Drift analysis with tail bounds. In Proceedings of Parallel Problem Solving from Nature (PPSN XI), LNCS 6238, pages 174-183. Springer, 2010.
- Stefan Droste, Thomas Jansen, and Ingo Wegener. On the analysis of the (1+1) evolutionary algorithm. Theoretical Computer Science, 276(1-2):51-81, 2002.
- Benjamin Doerr, Daniel Johannsen, and Carola Winzen. Drift analysis and linear functions revisited. In Proceedings of IEEE Congress on Evolutionary Computation (CEC 2010), pages 1967-1974. IEEE, 2010.
- Benjamin Doerr, Daniel Johannsen, and Carola Winzen. Multiplicative drift analysis. In Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2010), pages 1449-1456. ACM, 2010.
- Oliver Giel and Per Kristian Lehre. On the effect of populations in evolutionary multi-objective optimization. In Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2006), pages 651-658. ACM, 2006.
- Bruce Hajek. Hitting-time and occupation-time bounds implied by drift analysis with applications. Advances in Applied Probability, 14(3):502-525, 1982.
- Edda Happ, Daniel Johannsen, Christian Klein, and Frank Neumann. Rigorous analyses of fitness-proportional selection for optimizing linear functions. In Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2008), pages 953-960. ACM, 2008.
- Jun He and Xin Yao. Drift analysis and average time complexity of evolutionary algorithms. Artificial Intelligence, 127(1):57-85, 2001.
- Jun He and Xin Yao. A study of drift analysis for estimating computation time of evolutionary algorithms. Natural Computing, 3(1):21-35, 2004.
- Jens Jägersküpper. A blend of Markov-chain and drift analysis. In Proceedings of Parallel Problem Solving from Nature (PPSN X), LNCS 5199, pages 41-51. Springer, 2008.
- Frank Neumann and Ingo Wegener. Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. Theoretical Computer Science, 378(1):32-40, 2007.
- Pietro S. Oliveto and Carsten Witt. Simplified drift analysis for proving lower bounds in evolutionary computation. Algorithmica, 2011. In press.