Heuristic Optimization

Lecture 5

Algorithm Engineering Group
Hasso Plattner Institute, University of Potsdam

12 May 2015

Why is theory important?

We want to understand how an algorithm behaves over certain inputs.

Idea: run the algorithm over a large set of instances and observe its behavior.

Problem: sometimes evidence can be deceiving! Even when we think a process is well-behaved, it may not behave as we expect for all inputs.

Why is theory important?

At least half of the natural numbers less than any given number have an odd number of prime factors. — *George Pólya (1919)*

factor parity $m < n = 20$	
odd	**even**
19	$16 = 2^4$
$18 = 2 \cdot 3^2$	$15 = 3 \cdot 5$
17	$14 = 2 \cdot 7$
13	$10 = 2 \cdot 5$
$12 = 2^2 \cdot 3$	$9 = 3^2$
11	$6 = 2 \cdot 3$
$8 = 2^3$	$4 = 2^2$
7	
5	
3	
2	

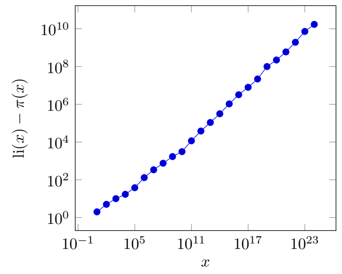
Resolved (false) by C. Brian Haselgrove (1958).

Smallest n for which the conjecture fails: $n = 906\,150\,257$ found by Minura Tanaka (1980).

Why is theory important?

Let $\pi(x)$ be the prime counting function and $\mathrm{li}(x) = \int_0^k \frac{dt}{\ln t}$.

All numerical evidence (1900s): $\pi(x) < \mathrm{li}(x)$



Skewes (1955): there must exist a value of x below

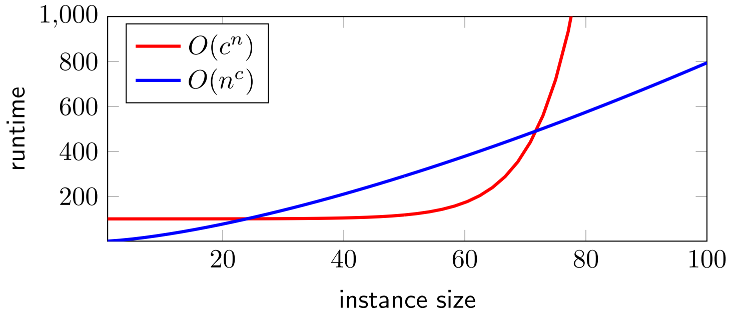
$$e^{e^{e^{7.705}}} < 10^{10^{10^{963}}}$$

for which $\pi(x) > \mathrm{li}(x)$. Currently, explicit x is unknown, but the bounds are

$$10^{14} < x < e^{727.951346801}.$$

Furthermore, this occurs infinitely often!

Why is theory important?



We want to make rigorous, indisputable arguments about the behavior of algorithms.

We want to understand how the behavior generalizes to any problem size.

Design and analysis of algorithms

Correctness

"does the algorithm always output the correct solution?"

Complexity

"how many computational resources are required?"

Design and analysis of algorithms

Randomized search heuristics

- Random local search
- Metropolis algorithm, simulated annealing
- Evolutionary algorithms, genetic algorithms
- Ant colony optimization

General-purpose: can be applied to any optimization problem

Challenges:

- Unlike classical algorithms, they are not designed with their analysis in mind
- Behavior depends on a random number generator

Convergence

First question: does the algorithm even find the solution?

Definition.

Let $f\colon S \to \mathbb{R}$ for a finite set S. Let $S \supseteq S^\star := \{x \in S : f(x) \text{ is optimal}\}$. We say an algorithm converges if it finds an element of $S^\star$ with probability 1 and holds it forever after.

Two conditions for convergence (Rudolph, 1998)

1. There is a positive probability to reach any point in the search space from any other point
2. The best solution is never lost (elitism)

Does the (1+1) EA converge on every function $f\colon \{0,1\}^n \to \mathbb{R}$?

Does RLS converge on every function $f\colon \{0,1\}^n \to \mathbb{R}$?

Can you think of how to modify RLS so that it converges?

Runtime analysis

In most cases, randomized search heuristics visit the global optimum in finite time (or can be easily modified to do so)

A far more important question: how long does it take?

To characterize this unambiguously: count the number of "primitive steps" until a solution is visited for the first time (typically a function growing with the input size)

We typically use asymptotic notation to classify the growth of such functions.

Runtime analysis

Randomized search heuristics

- time to evaluate fitness function evaluation is much higher than the rest
- do not perform the same operations even if the input is the same
- do not output the same result if run twice

Given a function $f\colon S \to \mathbb{R}$, the runtime of some RSH A applied to f is a random variable T_f that counts the number of calls A makes to f until an optimal solution is first generated.

We are interested in

- Estimating $E(T_f)$, the expected runtime of A on f
- Estimating $\Pr(T_f \le t)$, the success probability of A after t steps on f

Runtime analysis

RandomSearch

Choose x uniformly at random from S;
while stopping criterion not met **do**
  Choose y uniformly at random from S;
  if $f(y) \ge f(x)$ **then** $x \leftarrow y$;
end

We already have the tools to analyze this!

Suppose w.l.o.g., there is a unique maximum solution $x^\star \in S$ (if there are more, it can only be faster).

Consider a run of the algorithm $(x^{(0)}, x^{(1)}, \ldots)$ where $x^{(t)}$ is the solution generated in the t-th iteration.

Runtime analysis

Define the random variable X_t for $t \in \mathbb{N}_0$ as

$$X_t = \begin{cases} 1 & \text{if } x^{(t)} = x^\star, \\ 0 & \text{otherwise;} \end{cases}$$

So X_t has a Bernoulli distribution with parameter $p = 1/|S|$ (see Lecture 3).

Let T be the smallest t for which $X_t = 1$.

Then T is a geometrically distributed random variable (see Lecture 3).

Expected runtime: $E(T) = 1/p = |S|$

Runtime analysis

Success probability: $\Pr(T \le k) = 1 - (1-p)^k$

For example,

$$\Pr(T \le |S|) = 1 - (1 - 1/|S|)^{|S|} \ge 1 - 1/e \approx 0.6321$$

Constant chance that it takes $|S|$ steps to find the solution.

Let $S = \{0,1\}^n$. Let's bound the success probability before $2^{\epsilon n}$ for some constant $0 < \epsilon < 1$.

$$\Pr(T \le 2^{\epsilon n}) = 1 - (1 - 2^{-n})^{2^{\epsilon n}} \le 1 - \underbrace{(1 - 2^{-n} 2^{\epsilon n})}_{\text{see HW 2, Exercise 2a}}$$

$$= 2^{-n(1-\epsilon)} = 2^{-\Theta(n)}$$

So the probability that random search is successful before $2^{\Theta(n)}$ steps is vanishing quickly (faster than every polynomial) as n grows.

Runtime analysis

Let's consider more interesting cases. . .

Recall from Project 1:

(1+1) EA

Choose x uniformly at random from $\{0,1\}^n$;
while stopping criterion not met **do**
  $y \leftarrow x$;
  foreach $i \in \{1, \ldots, n\}$ **do**
    With probability $1/n$, $y_i \leftarrow (1 - y_i)$;
  end
  if $f(y) \ge f(x)$ **then** $x \leftarrow y$;
end

In each iterations, how many bits flip in expectation?
What is the probability exactly one bit flips?
What is the probability exactly two bits flip?
What is the probability that no bits flip?

Runtime Analysis

Theorem (Droste et al., 2002)

The expected runtime of the (1+1) EA for an arbitrary function $f\colon \{0,1\}^n \to \mathbb{R}$ is $O(n^n)$.

Proof.
Without loss of generality, suppose $x^\star$ is the unique optimum and x is the current solution.

Let $k = |\{i : x_i \neq x_i^\star\}|$.

Each bit flips (resp., does not flip) with probability $1/n$ (resp., with probability $1 - 1/n$).

Runtime Analysis

In order to reach the global optimum in the next step the algorithm has to mutate the k bits and leave the $n-k$ bits alone.

The probability to create the global optimum in the next step is

$$\left(\frac{1}{n}\right)^k \left(1 - \frac{1}{n}\right)^{n-k} \ge \left(\frac{1}{n}\right)^n = n^{-n}.$$

Assuming the process has not already generated the optimal solution, in expectation we wait $O(n^n)$ steps until this happens. □

Note: we are simply overestimating the time to find the optimal for any arbitrary pseudo-Boolean function.

Note: The upper bound is worse than for RandomSearch. In fact, there are functions where RandomSearch is guaranteed to perform better than the (1+1) EA.

Initialization

Recall from Project 1: $\textsc{OneMax}\colon \{0,1\}^n \to \mathbb{R}, x \mapsto |x|$;

How good is the initial solution?

Let X count the number of 1-bits in the initial solution. $E(X) = n/2$.

How likely to get exactly $n/2$?

$$\Pr(X = n/2) = \binom{n}{n/2} \frac{1}{2^{n/2}} \left(1 - \frac{1}{n}\right)^{n/2}$$

For $n = 100$, $\Pr(X = 50) \approx 0.0796$

Initialization (Tail Inequalities)

How likely is the initial solution no worse than $(3/4)n$?

Markov's Inequality

Let X be a random variable with $P(X < 0) = 0$. For all $a > 0$ we have

$$\Pr(X \geq a) \leq \frac{E(X)}{a}.$$

$$E(X) = n/2;\ \text{then}\ \Pr(X \geq (3/4)n) \leq \frac{E(X)}{(3/4)n} \leq 2/3$$

Initialization (Tail Inequalities)

Let $X_1, X_2, \ldots X_n$ be independent Poisson trials each with probability p_i;
For $X = \sum_{i=1}^{n} X_i$, the expectation is $E(X) = \sum_{i=1}^{p_i}$.

Chernoff Bounds

- for $0 \leq \delta \leq 1$, $\Pr(X \leq (1-\delta)E(X)) \leq e^{\frac{-E(x)\delta^2}{2}}$.
- for $\delta > 0$, $\Pr(X > (1+\delta)E(X)) \leq \left(\frac{e^\delta}{(1+\delta)^{(1+\delta)}}\right)^{E(X)}$.

E.g., $p_i = 1/2$, $E(X) = n/2$, fix $\delta = 1/2 \to (1+\delta)E(X) = (3/4)n$,

$$\Pr(X > (3/4)n) \leq \left(\frac{e^{1/2}}{(3/2)^{(3/2)}}\right)^{n/2} = c^{-n/2}.$$

Initialization (Tail Inequalities):

A simple example

Let $n = 100$. How likely is the initial solution no worse than $\textsc{OneMax}(x) = 75$?

$\Pr(X_i) = 1/2$ and $E(X) = 100/2 = 50$.

Markov: $\Pr(X \geq 75) \leq \frac{50}{75} = \frac{2}{3}$.

Chernoff: $\Pr(X \geq (1+1/2)50) \leq \left(\frac{\sqrt{e}}{(3/2)^{(3/2)}}\right)^{50} < 0.0054$.

In reality, $\Pr(X \geq 75) = \sum_{i=75}^{100} \binom{100}{i} 2^{-100} \approx 0.0000002818141$.