Theory of Randomised Search Heuristics
Part I: Introduction to the Analysis of Randomised Search Heuristics

Dirk Sudholt

University of Sheffield

Midlands Graduate School 2014

This project has received funding from the European Union’s Seventh Framework Programme for research, technological development and demonstration under grant agreement no 618091 (SAGE).
Randomised Search Heuristics

Randomised Search Heuristics (RSH)
- evolutionary algorithms
- simulated annealing
- swarm intelligence
- artificial immune systems
- ...

Benefits
- applicable where exact approaches are intractable
- applicable when problem is not well understood (black-box setting)
- lack of time, money, or expertise to design a tailored algorithm
- usually easy to implement and easy to apply
- robust and often surprisingly successful
Scheme of an Evolutionary Algorithm (EA)

Select parents for reproduction

Mutation/Recombination

Selection for new population
Why Do We Need Theory of RSH?

Survey amongst users of EAs [Hornby and Yu, 2006]

Primary obstacle for acceptance of EAs in industry is that they are “poorly understood” (39.7% of respondents).

Questions

- Which RSH is the best for my problem?
- How to tune parameters efficiently?
- How to make design choices?
- What performance can I expect from a RSH?
- How to design better RSH?
Runtime Analysis of RSH

Goals
- understand how RSH work
- get to know their capabilities and limitations
- solid theoretical foundation
- aid in the design of better RSH

What we are looking for
Bounds on the (expected) time until a metaheuristic finds a satisfactory solution for a given problem.
- global optimum
- good approximation

Notion of “time”
- number of evaluations of the objective function
- number of iterations / generations
Approach

Tools from the analysis of randomized algorithms
- tail inequalities (Markov, Chernoff, ...)
- Markov chain theory
- random walks, stochastic processes
- asymptotic notation
- amortized analysis
- ...

Challenge
- RSH often not designed to support an analysis

Perspective
- Classical algorithms theory: problem $\rightarrow$ algorithms
- RSH: algorithm (paradigm) $\rightarrow$ problems
Roadmap for this Course

- **Lecture 1**: Introduction to the analysis of RSH (Dirk Sudholt)
  - simple algorithms on simple problems
  - fitness-level method

- **Lecture 2**: Drift Analysis: a Tool for Analysing RSH (Per Kristian Lehre)
  - bounds on hitting times for stochastic processes
  - how to analyse populations in EAs

- **Lecture 3**: Theory of Evolutionary Algorithms for Combinatorial Optimisation (Pietro Oliveto)
  - performance of EAs on easy and NP-hard combinatorial problems

- **Lecture 4**: Mutation in Evolutionary Algorithms and Artificial Immune Systems (Christine Zarges)
  - extension to artificial immune systems
  - case study comparing different types of mutation
Randomised Local Search

RLS for maximization of $f : \{0, 1\}^n \to \mathbb{R}$

Choose $x \in \{0, 1\}^n$ uniformly at random.

repeat forever

Create $y$ by flipping a single bit in $x$ chosen uniformly at random.

if $f(y) \geq f(x)$ then $x := y$.

Properties:

- simple hill climber
- can find local optima
- cannot escape from local optima
A Simple Test Problem

**Task**
Find a hidden target string.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>solution</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fitness: number of correct bits.

**Task**
Find the all-ones string.

<table>
<thead>
<tr>
<th>target</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>solution</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fitness: number of 1-bits (OneMax).
Collecting Coupons

RLS on OneMax is like trying to collect \( n \) coupons.

Worst case: assume we start with no coupons, i.e. \( 0^n \).

If we have collected \( i \) coupons, the probability of getting a new one is
\[
\frac{n - i}{n}.
\]

The expected waiting time (# draws) for this is
\[
\frac{n}{n - i}.
\]

Summing up all these times gives an upper bound of
\[
\sum_{i=0}^{n-1} \frac{n}{n - i} = n \cdot \sum_{i=1}^{n} \frac{1}{i} \leq n(\ln(n) + 1).
\]

\textbf{Theorem}

The expected running time of RLS on OneMax is \( n \ln n + O(n) \).
The (1+1) Evolutionary Algorithm

(1+1) EA for maximization of $f : \{0, 1\}^n \rightarrow \mathbb{R}$

Choose $x \in \{0, 1\}^n$ uniformly at random.

repeat forever

Create $y$ by flipping each bit in $x$ independently with probability $1/n$.

if $f(y) \geq f(x)$ then $x := y$.

Properties:

• reflects basic principle of mutation and selection
• stochastic hill climber
• flips one bit in expectation
• can mimic one step of RLS
• can escape from local optima by flipping many bits
Fitness-level Method for the (1+1) EA

\[ \text{Pr}((1+1) \text{ EA leaves } A_i) \geq s_i \]

Expected optimization time of (1+1) EA at most \( \sum_{i=1}^{m-1} \frac{1}{s_i} \).
Fitness-Level Method: \( (1+1) \) EA on OneMax

\[
\text{OneMax} (x) := \sum_{i=1}^{n} x_i
\]

Fitness-level partition: \( A_i := \{ x \mid \text{OneMax}(x) = i \} \).

Sufficient condition for leaving \( A_i \): just flip one 0-bit.

\[
s_i \geq \frac{n - i}{n} \cdot \left( 1 - \frac{1}{n} \right)^{n-1} \geq \frac{n - i}{en}
\]

Bound on the expected optimization time of \( (1+1) \) EA

\[
\sum_{i=0}^{n-1} \frac{1}{s_i} = \sum_{i=0}^{n-1} \frac{en}{n-i} = en \sum_{i=1}^{n} \frac{1}{i} \leq en \ln n + O(n)
\]

Lower bound: \( en \ln n - O(n) \) [Doerr, Fouz, Witt, 2011].
Fitness-Level Method: \((1+1)\) EA on LeadingOnes

\[ \text{LO} (x) := \sum_{i=1}^{n} \prod_{j=1}^{i} x_j \] counts the number of leading ones.

Fitness-level partition: \(A_i := \{x \mid \text{LO}(x) = i\}\).

Sufficient condition for leaving \(A_i\): flip bit \(i + 1\).

\[ s_i \geq \frac{1}{n} \cdot \left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{1}{en} \]

Bound on the expected optimization time of \((1+1)\) EA

\[ \sum_{i=0}^{n-1} \frac{1}{s_i} = \sum_{i=0}^{n-1} en = en^2. \]
(1+1) EA Always Finds an Optimum

**Theorem**

(1+1) EA *finds a global optimum on every function in expected time* $n^n$.

Same for all RSH that use standard mutation operators.

Fitness-level partition:

$$A_0 = \{0, 1\}^n \setminus \text{OPT}$$
$$A_1 = \text{OPT}$$

Worst case for $A_0$: all $n$ bits have to flip. So

$$s_0 \geq \left(\frac{1}{n}\right)^n$$

and we get an upper bound of

$$\sum_{i=0}^{0} \frac{1}{s_0} = \frac{1}{s_0} \leq n^n.$$
**Jump**

\[ \text{Jump}_k \text{ [Jansen and Wegener, 2002]} : \text{ “jump” of } k \text{ bits required.} \]

![Graph showing fitness versus number of 1-bits]

Take \( s_0, \ldots, s_{n-1} \) as for OneMax and
\[
s_n = \left(\frac{1}{n}\right)^k \left(1 - \frac{1}{n}\right)^{n-k} \geq \frac{1}{(en)^k}.
\]

Expected optimisation time of \((1+1)\) EA on \( \text{Jump}_k \) is \( O(n \log n + n^k) \).
Extensions of the Fitness-Level Method

Upper bounds for population/swarm-based search heuristics

Idea: add time for search to focus on current best level.

- elitist populations [Witt, 2006]
- ant colony optimisation [Gutjahr and Sebastiani, 2008]
- binary particle swarm optimisation [Sudholt and Witt, 2010]
- parallel evolutionary algorithms [Lässig and Sudholt, 2013]
- non-elitist populations [Lehre, 2011]

Further Applications

- lower bounds [Sudholt, 2013]
- tail bounds [Zhou, Luo, Lu, Han, 2012 and Witt, 2013]
Crossover on Jump\textsubscript{$k$}

Success probability for mutation: $\sim n^{-k}$.

\[
\begin{array}{cccccccc}
0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & 1 & 1 & 0 & 1
\end{array}
\]

Success probability for uniform crossover: 0.

\[
\begin{array}{cccccccc}
0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & 1 & 1 & 0 & 1
\end{array}
\]

Success prob. for uniform crossover of complementary pair: $2^{-2k} = 4^{-k}$.

Superpolynomial gap for $k = \log n$

<table>
<thead>
<tr>
<th></th>
<th>$4^k$</th>
<th>$n^k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n = 10$</td>
<td>100</td>
<td>2099</td>
</tr>
<tr>
<td>$n = 100$</td>
<td>10,000</td>
<td>$2 \cdot 10^{13}$</td>
</tr>
<tr>
<td>$n = 1000$</td>
<td>1,000,000</td>
<td>$8 \cdot 10^{29}$</td>
</tr>
</tbody>
</table>

Superpolynomial gap for simple EAs with uniform crossover

[Jansen and Wegener, 2002 and Kötzing, Sudholt, Theile, 2011].
Further Reading

- Bioinspired Computation in Combinatorial Optimization
- Theory of Randomized Search Heuristics
- Analyzing Evolutionary Algorithms
Conclusions

Evolutionary algorithms can be analysed!

Insights for evolutionary algorithms

- Simple EAs can do hill climbing efficiently (as well as RLS)
- Global mutations can optimise any problem in finite time
- $\text{Jump}_k$: example where crossover is beneficial.

Fitness-level method

- Simple yet powerful method for analysing RSH
- Examples for (1+1) EA: improvements through mutation
- Extensions to populations, parallel EAs, non-elitist populations, etc.

Thank you!