# Experimental Algorithmics

Conrado Martínez
U. Politècnica de Catalunya

September 27, 2022

1. Experimental Algorithmics

# Experimental Algorithmics: Introduction

- What is Experimental Algorithmics?
- Why doing experiments (with algorithms)?

# Experimental Algorithmics: Introduction

Goals of the analysis of algorithms (and data structures):

1. To predict the amount of computational resources needed by an algorithm, in terms of simple parameters, e.g., *size*.
2. To compare the performance of competing alternative solutions.
3. To help and guide the design of new algorithms or variants of existing ones.
4. To explain the observed performance of algorithms.

# Experimental Algorithmics: Introduction

Goals of the analysis of algorithms (and data structures):

1. To predict the amount of computational resources needed by an algorithm, in terms of simple parameters, e.g., *size*.
2. To compare the performance of competing alternative solutions.
3. To help and guide the design of new algorithms or variants of existing ones.
4. To explain the observed performance of algorithms.

# Experimental Algorithmics: Introduction

Goals of the analysis of algorithms (and data structures):

1. To predict the amount of computational resources needed by an algorithm, in terms of simple parameters, e.g., *size*.
2. To compare the performance of competing alternative solutions.
3. To help and guide the design of new algorithms or variants of existing ones.
4. To explain the observed performance of algorithms.

# Experimental Algorithmics: Introduction

Goals of the analysis of algorithms (and data structures):

1. To predict the amount of computational resources needed by an algorithm, in terms of simple parameters, e.g., *size*.
2. To compare the performance of competing alternative solutions.
3. To help and guide the design of new algorithms or variants of existing ones.
4. To explain the observed performance of algorithms.

# Experimental Algorithmics: Introduction

- Goals #1 and #4 are “scientific” goals.
- Goals #2 and #3 are “engineering” goals.

# Experimental Algorithmics: Introduction

- Goals #1 and #4 are “scientific” goals.
- Goals #2 and #3 are “engineering” goals.

# Experimental Algorithmics: Introduction

At a deep level:

- Goals #1 (predict) and #4 (explain) are identical to the main goals of any other science
- We look for quantitative predictions, the use of mathematical models that provide measurable and precise descriptions of the behavior of a system (which ultimately explain it)
- The rôle of experiments in Computer Science is the rôle they play in the scientific method.

# Experimental Algorithmics: Introduction

At a deep level:

- Goals #1 (predict) and #4 (explain) are identical to the main goals of any other science
- We look for quantitative predictions, the use of mathematical models that provide measurable and precise descriptions of the behavior of a system (which ultimately explain it)
- The rôle of experiments in Computer Science is the rôle they play in the scientific method.

# Experimental Algorithmics: Introduction

At a deep level:

- Goals #1 (predict) and #4 (explain) are identical to the main goals of any other science
- We look for quantitative predictions, the use of mathematical models that provide measurable and precise descriptions of the behavior of a system (which ultimately explain it)
- The rôle of experiments in Computer Science is the rôle they play in the scientific method.

# Intro to EA

1. Experiments are the source of (controlled) observations, a very fruitful starting point for the scientific endeavour
2. They help us develop hypotheses and intuitions about the behavior of algorithms (pilot studies)
3. They serve to test the hypotheses and refine them
4. They are the ultimate yardstick for the utility of our computational and mathematical models
5. Exhaustive experiments provide compelling evidence to support the hypotheses and validate mathematical models (workhorse studies)

# Intro to EA

1. Experiments are the source of (controlled) observations, a very fruitful starting point for the scientific endeavour
2. They help us develop hypotheses and intuitions about the behavior of algorithms (pilot studies)
3. They serve to test the hypotheses and refine them
4. They are the ultimate yardstick for the utility of our computational and mathematical models
5. Exhaustive experiments provide compelling evidence to support the hypotheses and validate mathematical models (workhorse studies)

# Intro to EA

1. Experiments are the source of (controlled) observations, a very fruitful starting point for the scientific endeavour
2. They help us develop hypotheses and intuitions about the behavior of algorithms (pilot studies)
3. They serve to test the hypotheses and refine them
4. They are the ultimate yardstick for the utility of our computational and mathematical models
5. Exhaustive experiments provide compelling evidence to support the hypotheses and validate mathematical models (workhorse studies)

# Intro to EA

1. Experiments are the source of (controlled) observations, a very fruitful starting point for the scientific endeavour
2. They help us develop hypotheses and intuitions about the behavior of algorithms (pilot studies)
3. They serve to test the hypotheses and refine them
4. They are the ultimate yardstick for the utility of our computational and mathematical models
5. Exhaustive experiments provide compelling evidence to support the hypotheses and validate mathematical models (workhorse studies)

# Intro to EA

1. Experiments are the source of (controlled) observations, a very fruitful starting point for the scientific endeavour
2. They help us develop hypotheses and intuitions about the behavior of algorithms (pilot studies)
3. They serve to test the hypotheses and refine them
4. They are the ultimate yardstick for the utility of our computational and mathematical models
5. Exhaustive experiments provide compelling evidence to support the hypotheses and validate mathematical models (workhorse studies)

# Intro to EA

1. Experiments can be used to check to what extent the conclusions drawn from the models apply in (real life?) situations where some of our assumptions do not hold, e.g., randomness of the input, independence, etc.
2. If your model does not apply, try to find an explanation for failure
3. Correctness is not an absolute concept in natural sciences: the claim that the Earth is an sphere is not correct, but it more “correct” than the claim it is plane! Use experiments to quantify the “correctness” of your model

# Intro to EA

1. Experiments can be used to check to what extent the conclusions drawn from the models apply in (real life?) situations where some of our assumptions do not hold, e.g., randomness of the input, independence, etc.
2. If your model does not apply, try to find an explanation for failure
3. Correctness is not an absolute concept in natural sciences: the claim that the Earth is an sphere is not correct, but it more “correct” than the claim it is plane! Use experiments to quantify the “correctness” of your model

# Intro to EA

1. Experiments can be used to check to what extent the conclusions drawn from the models apply in (real life?) situations where some of our assumptions do not hold, e.g., randomness of the input, independence, etc.
2. If your model does not apply, try to find an explanation for failure
3. Correctness is not an absolute concept in natural sciences: the claim that the Earth is an sphere is not correct, but it more "correct" than the claim it is plane! Use experiments to quantify the "correctness" of your model

# Intro to EA

1. Simulations are closely related to experiments but a simulation produces (numerical) data according to a theoretical model
2. Simulations are very useful to investigate when the asymptotic regime starts, estimate the magnitude of hidden constants, etc.
3. For us it is often difficult to draw a line between experiments and simulations: the algorithms ($\equiv$ nature) might be seen as models themselves!

# Intro to EA

1. Simulations are closely related to experiments but a simulation produces (numerical) data according to a theoretical model
2. Simulations are very useful to investigate when the asymptotic regime starts, estimate the magnitude of hidden constants, etc.
3. For us it is often difficult to draw a line between experiments and simulations: the algorithms ($\equiv$ nature) might be seen as models themselves!

# Intro to EA

1. Simulations are closely related to experiments but a simulation produces (numerical) data according to a theoretical model
2. Simulations are very useful to investigate when the asymptotic regime starts, estimate the magnitude of hidden constants, etc.
3. For us it is often difficult to draw a line between experiments and simulations: the algorithms ($\equiv$ nature) might be seen as models themselves!

# Intro to EA: Do's and Don'ts

1. Experiments should be set up with a falsifiable hypothesis (that's what the scientific method requires!)
2. If not, they're fine for the exploratory phase, but not much more . . . They might be good to illustrate your point, but be aware of their limited value
3. Experiments must be reproducible: better report artifact-independent measures!

# Intro to EA: Do's and Don'ts

1. Experiments should be set up with a falsifiable hypothesis (that's what the scientific method requires!)
2. If not, they're fine for the exploratory phase, but not much more … They might be good to illustrate your point, but be aware of their limited value
3. Experiments must be reproducible: better report artifact-independent measures!

# Intro to EA: Do's and Don'ts

1. Experiments should be set up with a falsifiable hypothesis (that's what the scientific method requires!)
2. If not, they're fine for the exploratory phase, but not much more . . . They might be good to illustrate your point, but be aware of their limited value
3. Experiments must be reproducible: better report artifact-independent measures!

# Intro to EA: Do's and Don'ts

- Empirical comparative studies ("running races") can raise your adrenaline but have little or no explicative power . . .
- They are OK from the engineering perspective
- Comparing two variants $A'$ and $A''$ of an algorithm is useful from the scientific point of view; the differences in performance could hopefully be explained in terms of the (small) differences between $A'$ and $A''$

# Intro to EA: Do's and Don'ts

- Empirical comparative studies ("running races") can raise your adrenaline but have little or no explicative power ...
- They are OK from the engineering perspective
- Comparing two variants $A'$ and $A''$ of an algorithm is useful from the scientific point of view; the differences in performance could hopefully be explained in terms of the (small) differences between $A'$ and $A''$

# Intro to EA: Do's and Don'ts

- Empirical comparative studies ("running races") can raise your adrenaline but have little or no explicative power ...
- They are OK from the engineering perspective
- Comparing two variants $A'$ and $A''$ of an algorithm is useful from the scientific point of view; the differences in performance could hopefully be explained in terms of the (small) differences between $A'$ and $A''$

# Intro to EA: Do's and Don'ts

- CPU time depends on many factors, including the instrument of measure (the computer)!
- A serious scientific study of CPU time and other machine-dependent features must analyze and explain each of the factors involved: run the experiments for different architectures, programming languages, ...
- Studies of CPU time versus input size alone are more often than not useless; we need experiments to reveal the dependence of CPU time on several parameters
- Experiments at that level of instantation (see later) are difficult to reproduce

# Intro to EA: Do's and Don'ts

- CPU time depends on many factors, including the instrument of measure (the computer)!
- A serious scientific study of CPU time and other machine-dependent features must analyze and explain each of the factors involved: run the experiments for different architectures, programming languages, ...
- Studies of CPU time versus input size alone are more often than not useless; we need experiments to reveal the dependence of CPU time on several parameters
- Experiments at that level of instantation (see later) are difficult to reproduce

# Intro to EA: Do's and Don'ts

- CPU time depends on many factors, including the instrument of measure (the computer)!
- A serious scientific study of CPU time and other machine-dependent features must analyze and explain each of the factors involved: run the experiments for different architectures, programming languages, ...
- Studies of CPU time versus input size alone are more often than not useless; we need experiments to reveal the dependence of CPU time on several parameters
- Experiments at that level of instantation (see later) are difficult to reproduce

# Intro to EA: Do's and Don'ts

- CPU time depends on many factors, including the instrument of measure (the computer)!
- A serious scientific study of CPU time and other machine-dependent features must analyze and explain each of the factors involved: run the experiments for different architectures, programming languages, ...
- Studies of CPU time versus input size alone are more often than not useless; we need experiments to reveal the dependence of CPU time on several parameters
- Experiments at that level of instantation (see later) are difficult to reproduce

# Intro to EA: Do's and Don'ts

- Benchmarks are not experiments; they are observations, much like observing the behavior of animals in the wild.
- They are a nice source of observational data, to be explained and complemented by experiments (under controlled conditions).
- They are also good (although far from complete!) to check the utility of our mathematical models; of course it's very nice when your predictions explain well "real-life" phenomena
- Good predictions for real-life data $\Longrightarrow$ understanding what is the "structure" of these instances $\Longrightarrow$ we can generate synthetic instances that mimick well real-life instances but that we can control at will

# Intro to EA: Do's and Don'ts

- Benchmarks are not experiments; they are observations, much like observing the behavior of animals in the wild.
- They are a nice source of observational data, to be explained and complemented by experiments (under controlled conditions).
- They are also good (although far from complete!) to check the utility of our mathematical models; of course it's very nice when your predictions explain well "real-life" phenomena
- Good predictions for real-life data $\Longrightarrow$ understanding what is the "structure" of these instances $\Longrightarrow$ we can generate synthetic instances that mimick well real-life instances but that we can control at will

# Intro to EA: Do's and Don'ts

- Benchmarks are not experiments; they are observations, much like observing the behavior of animals in the wild.
- They are a nice source of observational data, to be explained and complemented by experiments (under controlled conditions).
- They are also good (although far from complete!) to check the utility of our mathematical models; of course it's very nice when your predictions explain well "real-life" phenomena
- Good predictions for real-life data $\Longrightarrow$ understanding what is the "structure" of these instances $\Longrightarrow$ we can generate synthetic instances that mimick well real-life instances but that we can control at will

# Intro to EA: Do's and Don'ts

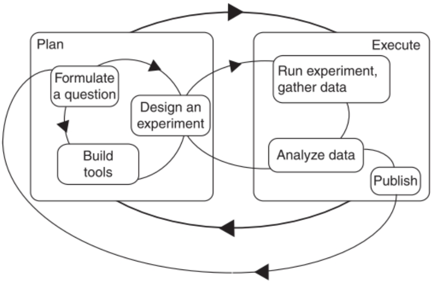
- Benchmarks are not experiments; they are observations, much like observing the behavior of animals in the wild.
- They are a nice source of observational data, to be explained and complemented by experiments (under controlled conditions).
- They are also good (although far from complete!) to check the utility of our mathematical models; of course it's very nice when your predictions explain well "real-life" phenomena
- Good predictions for real-life data $\Longrightarrow$ understanding what is the "structure" of these instances $\Longrightarrow$ we can generate synthetic instances that mimick well real-life instances but that we can control at will

# Intro to EA

- Levels / scales of instantiation
  - Algorithmic paradigms, algorithmic schemes, metaheuristics
  - Algorithms (e.g., # of comparisons)
  - Source code/programs (e.g., # of instructions)
  - Processes (e.g., CPU time elapsed)
- Test subject vs. test program—often different!

# Intro to EA

- The experimental process



# Intro to EA

Experimental goals

- reproducibility / replication
- efficiency
- relevance / utility

# Intro to EA

- Pilot studies vs. the *workhorse*
- Spurious results & artifacts: bugs, external factors, biased "randomness", round-up & fixed-point arithmetic, ...

# Experimental Setup

- Input generation ($\leftarrow$ reproducibility, efficiency, relevance!)
- Measures: prefer abstract, machine-independent quantities, conduct a separate study to relate abstract measures to machine-dependent measures (e.g., CPU time)
- Instrumentation: does measuring change results? where and when to measure?
- Carrying out the experiments: test subject vs. test program

# Experimental Setup

- Input generation ($\leftarrow$ reproducibility, efficiency, relevance!)
- Measures: prefer abstract, machine-independent quantities, conduct a separate study to relate abstract measures to machine-dependent measures (e.g., CPU time)
- Instrumentation: does measuring change results? where and when to measure?
- Carrying out the experiments: test subject vs. test program

# Experimental Setup

- Input generation ($\leftarrow$ reproducibility, efficiency, relevance!)
- Measures: prefer abstract, machine-independent quantities, conduct a separate study to relate abstract measures to machine-dependent measures (e.g., CPU time)
- Instrumentation: does measuring change results? where and when to measure?
- Carrying out the experiments: test subject vs. test program

# Experimental Setup

- Input generation ($\leftarrow$ reproducibility, efficiency, relevance!)
- Measures: prefer abstract, machine-independent quantities, conduct a separate study to relate abstract measures to machine-dependent measures (e.g., CPU time)
- Instrumentation: does measuring change results? where and when to measure?
- Carrying out the experiments: test subject vs. test program

# Experimental Setup

- Tune the code to generate inputs and to do the experiments
- Some items of advice for a good experimental study
  - More trials, larger samples
  - Larger inputs to show long-term/asymptotic regime
  - Do not summarize prematurely: Report "raw" data. Data analysis is a different task to be done by a different tool
  - Find a trade-off bewteen reporting too much or not enough data points (efficiency, utility)
  - Don't use "lossy" performance measures; you can use them in the data analysis & visualization phase. E.g., you are interested in $R = X/Y$, your experiment should report $X$ and $Y$ values.
  - Prefer focused, "narrow" indicators, e.g., counts for a specific kind of elementary operation vs. a count of executed instructions.

# Experimental Setup

- Tune the code to generate inputs and to do the experiments
- Some items of advice for a good experimental study
  - More trials, larger samples
  - Larger inputs to show long-term/asymptotic regime
  - Do not summarize prematurely. Report "raw" data. Data analysis is a different task to be done by a different tool
  - Find a trade-off bewteen reporting too much or not enough data points (efficiency, utility)
  - Don't use "lossy" performance measures; you can use them in the data analysis & visualization phase. E.g., you are interested in $R = X/Y$, your experiment should report $X$ and $Y$ values.
  - Prefer focused, "narrow" indicators, e.g., counts for a specific kind of elementary operation vs. a count of executed instructions.

# Experimental Setup

- Tune the code to generate inputs and to do the experiments
- Some items of advice for a good experimental study
  - More trials, larger samples
  - Larger inputs to show long-term/asymptotic regime
  - Do not summarize prematurely. Report "raw" data. Data analysis is a different task to be done by a different tool
  - Find a trade-off bewteen reporting too much or not enough data points (efficiency, utility)
  - Don't use "lossy" performance measures; you can use them in the data analysis & visualization phase. E.g., you are interested in $R = X/Y$, your experiment should report $X$ and $Y$ values.
  - Prefer focused, "narrow" indicators, e.g., counts for a specific kind of elementary operation vs. a count of executed instructions.

# Experimental Setup

- Tune the code to generate inputs and to do the experiments
- Some items of advice for a good experimental study
  - More trials, larger samples
  - Larger inputs to show long-term/asymptotic regime
  - Do not summarize prematurely. Report “raw” data. Data analysis is a different task to be done by a different tool
  - Find a trade-off bewteen reporting too much or not enough data points (efficiency, utility)
  - Don’t use “lossy” performance measures; you can use them in the data analysis & visualization phase. E.g., you are interested in $R = X/Y$, your experiment should report $X$ and $Y$ values.
  - Prefer focused, “narrow” indicators, e.g., counts for a specific kind of elementary operation vs. a count of executed instructions.

# Experimental Setup

- Tune the code to generate inputs and to do the experiments
- Some items of advice for a good experimental study
  - More trials, larger samples
  - Larger inputs to show long-term/asymptotic regime
  - Do not summarize prematurely. Report "raw" data. Data analysis is a different task to be done by a different tool
  - Find a trade-off bewteen reporting too much or not enough data points (efficiency, utility)
  - Don't use "lossy" performance measures; you can use them in the data analysis & visualization phase. E.g., you are interested in $R = X/Y$, your experiment should report $X$ and $Y$ values.
  - Prefer focused, "narrow" indicators, e.g., counts for a specific kind of elementary operation vs. a count of executed instructions.

# Experimental Setup

- Tune the code to generate inputs and to do the experiments
- Some items of advice for a good experimental study
  - More trials, larger samples
  - Larger inputs to show long-term/asymptotic regime
  - Do not summarize prematurely. Report “raw” data. Data analysis is a different task to be done by a different tool
  - Find a trade-off bewteen reporting too much or not enough data points (efficiency, utility)
  - Don’t use “lossy” performance measures; you can use them in the data analysis & visualization phase. E.g., you are interested in $R = X/Y$, your experiment should report $X$ and $Y$ values.
  - Prefer focused, “narrow” indicators, e.g., counts for a specific kind of elementary operation vs. a count of executed instructions.

# Experimental Setup

- Tune the code to generate inputs and to do the experiments
- Some items of advice for a good experimental study
  - More trials, larger samples
  - Larger inputs to show long-term/asymptotic regime
  - Do not summarize prematurely. Report “raw” data. Data analysis is a different task to be done by a different tool
  - Find a trade-off bewteen reporting too much or not enough data points (efficiency, utility)
  - Don’t use “lossy” performance measures; you can use them in the data analysis & visualization phase. E.g., you are interested in $R = X/Y$, your experiment should report $X$ and $Y$ values.
  - Prefer focused, “narrow” indicators, e.g., counts for a specific kind of elementary operation vs. a count of executed instructions.

# Experimental Setup

- Tune the code to generate inputs and to do the experiments
- Some items of advice for a good experimental study
  - More trials, larger samples
  - Larger inputs to show long-term/asymptotic regime
  - Do not summarize prematurely. Report “raw” data. Data analysis is a different task to be done by a different tool
  - Find a trade-off bewteen reporting too much or not enough data points (efficiency, utility)
  - Don’t use “lossy” performance measures; you can use them in the data analysis & visualization phase. E.g., you are interested in $R = X/Y$, your experiment should report $X$ and $Y$ values.
  - Prefer focused, “narrow” indicators, e.g., counts for a specific kind of elementary operation vs. a count of executed instructions.

# Experimental Setup

Apply variance-reduction techniques (VRT):

- Use the same inputs for variants of an algorithm to be compared (reduces variance and saves times!), measure $\Delta = X - X'$

$$\mathbb{V}[\Delta] = \mathbb{V}[X] + \mathbb{V}[X'] - 2\text{Cov}(X, X')$$

- Use a control variate $X'$, another measure with known expected value $\mu'$ and positively correlated with the measure of your interest $X$:

$$Y = X - c(X' - \mu), \mathbb{E}[Y] = \mathbb{E}[X]$$

  Optimal value for $c = \text{Cov}(X, Y')/\,\mathbb{V}[X']$; estimate a good value for $c$ with some small pilot study

# Experimental Setup

Apply variance-reduction techniques (VRT):

- Conditional expectation VRT (conditional Monte Carlo): split state generation from cost measurement and make many (all?) measurements on the same state.
  Very usual in empirical studies of data structures
  1. build the random data structure
  2. measure some unique parameter that conveys info about the costs (IPL/EPL in search trees, path length/list length in hash tables, . . . )

# Data Analysis

- Curve fitting: don't overfit, use theory to provide a reasonable guess / ground model
- Visualization of data: prefer plots to table to convey information, but don't put too much information; be watchful with misleading plots
- Testing hypotheses: use statistical tools, go beyond averages; collect as much data from the experiments as possible, to be processed later

# Examples

- Deletions in binary search trees $\Longrightarrow$ Eppinger (1983)
- Percolation in random graphs $\Longrightarrow$ Sedgewick's slides for *Algorithms*
- Cache performance of quicksort $\Longrightarrow$ LaMarca & Ladner (1999)
- Some further examples $\Longrightarrow$ check the shared documentation folder *Examples*

# Experimental Algorithmics

To learn more:

Catherine C. McGeoch
A Guide to Experimental Algorithmics.
Cambridge Univ. Press, 2012

# Experimental Algorithmics

To learn more:


J. L. Eppinger
An Empirical Study of Insertion and Deletion in Binary Search Trees.
*Comm. ACM* 26(9):663-669, 1983.

A. LaMarca and R. E. Ladner
The Influence of Caches on the Performance of Sorting.
*J. Algorithms* 31(1):66–104, 1999.