合肥大學
HEFEI UNIVERSITY

Comparing Optimization Algorithms

Thomas Weise (汤卫思)
tweise@hfuu.edu.cn

Institute of Applied Optimization (IAO) | 应用优化研究所
School of Artificial Intelligence and Big Data | 人工智能与大数据学院
Hefei University | 合肥大学
Hefei, Anhui, China | 中国安徽省合肥市

Version: 2025-08-11

Outline

1. Introduction
2. Views on Performance and Time
3. Statistical Measures
4. Statistical Comparisons
5. Testing is Not Enough
6. Other Stuff
7. Summary
8. Advertisement

Introduction

Introduction

- There are many optimization algorithms.

Introduction

- There are many optimization algorithms.
- For solving an optimization problem, we want to use the algorithm most suitable for it.

Introduction

- There are many optimization algorithms.
- For solving an optimization problem, we want to use the algorithm most suitable for it.
- What does this mean?

Introduction

- There are many optimization algorithms.
- For solving an optimization problem, we want to use the algorithm most suitable for it.
- What does this mean?
- And how do we find this algorithm?

Introduction

- There are many optimization algorithms.
- For solving an optimization problem, we want to use the algorithm most suitable for it.
- What does this mean?
- And how do we find this algorithm?
- Hopefully this lesson will help answering these questions.

Introduction

- There are many optimization algorithms.
- For solving an optimization problem, we want to use the algorithm most suitable for it.
- What does this mean?
- And how do we find this algorithm?
- Hopefully this lesson will help answering these questions.
- As a complement to this lesson, I suggest the report *"Benchmarking in Optimization: Best Practice and Open Issues"*[6] on arXiv.

Exact vs. Heuristic Algorithms

- In optimization, there exist exact and heuristic algorithms.

Exact vs. Heuristic Algorithms

- In optimization, there exist exact and heuristic algorithms.
- Let's look at the classical Traveling Salesperson Problem (TSP)[2,27,39,61].

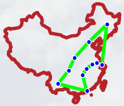
Exact vs. Heuristic Algorithms

- In optimization, there exist exact and heuristic algorithms.
- Let's look at the classical Traveling Salesperson Problem (TSP)[2,27,39,61].
    - Clearly, there is (at least) one shortest tour.

Exact vs. Heuristic Algorithms

- In optimization, there exist exact and heuristic algorithms.
- Let's look at the classical Traveling Salesperson Problem (TSP)[2,27,39,61].
    - Clearly, there is (at least) one shortest tour.

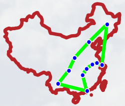
getting the optimal solution
for a TSP



Exact vs. Heuristic Algorithms

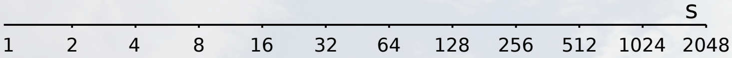
- In optimization, there exist exact and heuristic algorithms.
- Let's look at the classical Traveling Salesperson Problem (TSP)[2,27,39,61].
  - Clearly, there is (at least) one shortest tour.
  - Theory proofs that the time needed to find this tour may grow exponentially with the number s of cities we want to visit in the worst case.[1,14,15,35,38]

getting the optimal solution
for a TSP



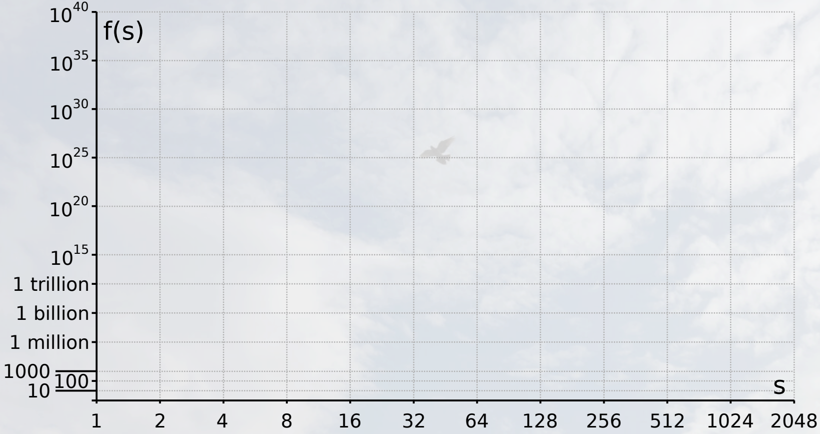
Exact vs. Heuristic Algorithms

- In optimization, there exist exact and heuristic algorithms.
- Let's look at the classical Traveling Salesperson Problem (TSP)[2,27,39,61].
- What does exponential growth mean?
- Let's say we have a number of cities s.



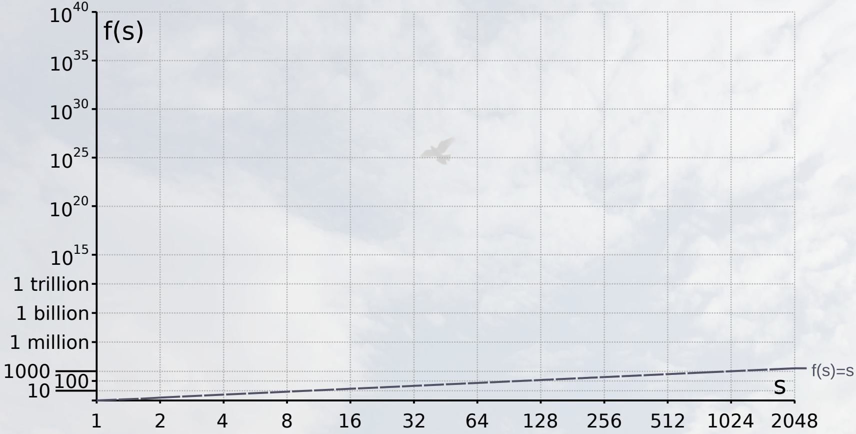
Exact vs. Heuristic Algorithms

- In optimization, there exist exact and heuristic algorithms.
- Let's look at the classical Traveling Salesperson Problem (TSP)[2,27,39,61].
- Let's say we have a number of cities s and a runtime as a function $f(s)$ in this log-log plot.



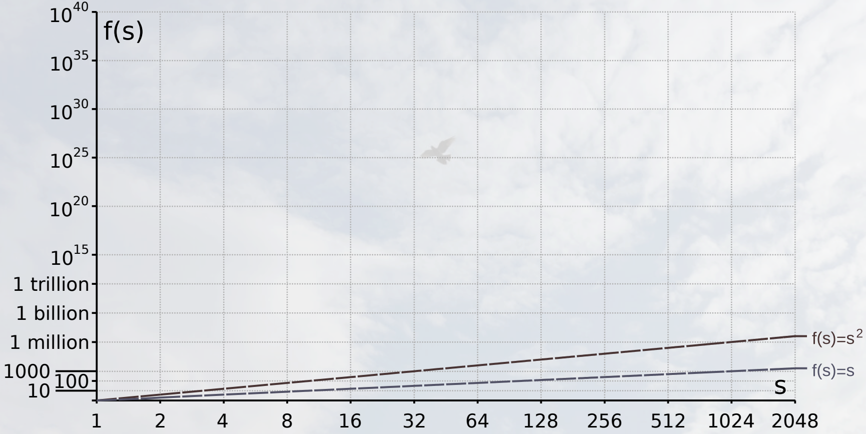
Exact vs. Heuristic Algorithms

- In optimization, there exist exact and heuristic algorithms.
- Let's look at the classical Traveling Salesperson Problem (TSP)[2,27,39,61].
- A linear function means that the runtime $f(s)$ grows slowly with s.



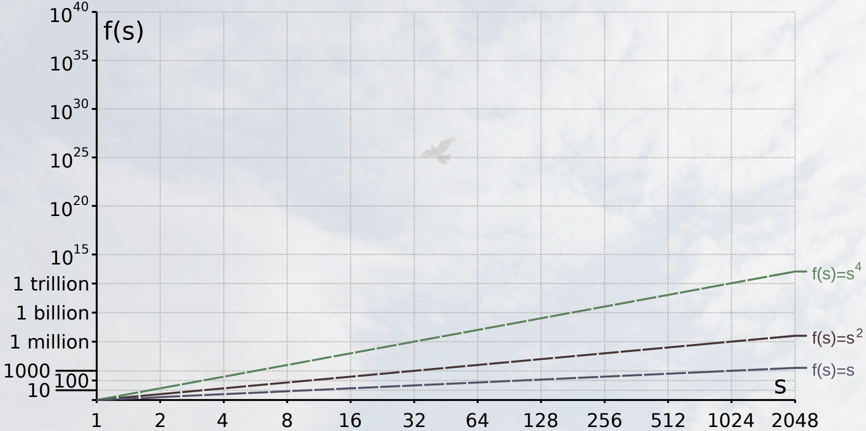
Exact vs. Heuristic Algorithms

- In optimization, there exist exact and heuristic algorithms.
- Let's look at the classical Traveling Salesperson Problem (TSP)[2,27,39,61].
- A quadratic function (a straight line in log-log plots) is also OK.



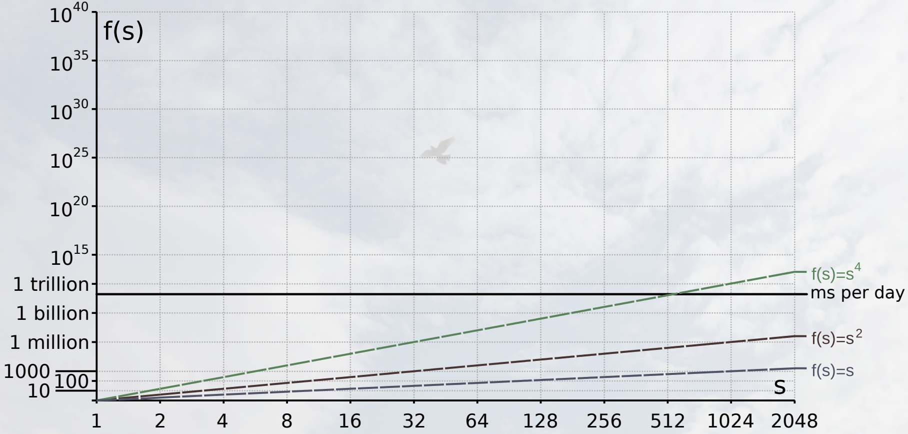
Exact vs. Heuristic Algorithms

- In optimization, there exist exact and heuristic algorithms.
- Let's look at the classical Traveling Salesperson Problem (TSP)[2,27,39,61].
- A quartic function $f(s) = s^4$ gets quite large for growing s.



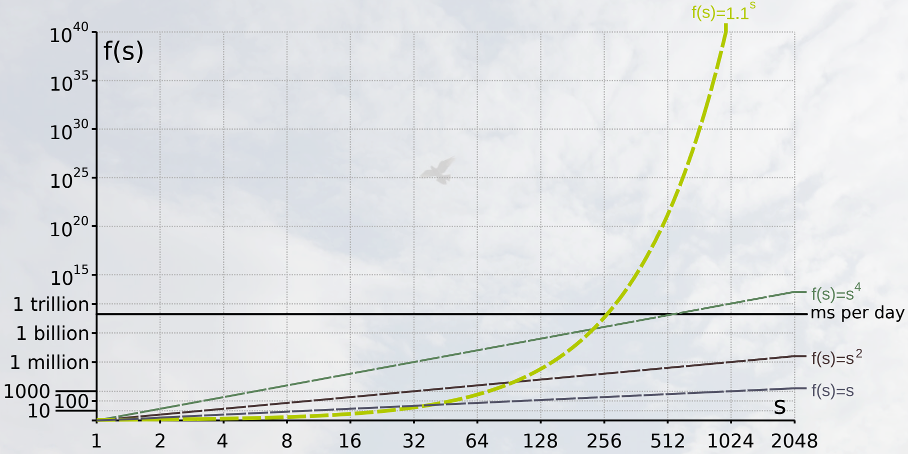
Exact vs. Heuristic Algorithms

- In optimization, there exist exact and heuristic algorithms.
- Let's look at the classical Traveling Salesperson Problem (TSP)[2,27,39,61].
- A quartic function exceeds the number of milliseconds per day at $s \approx 512$.



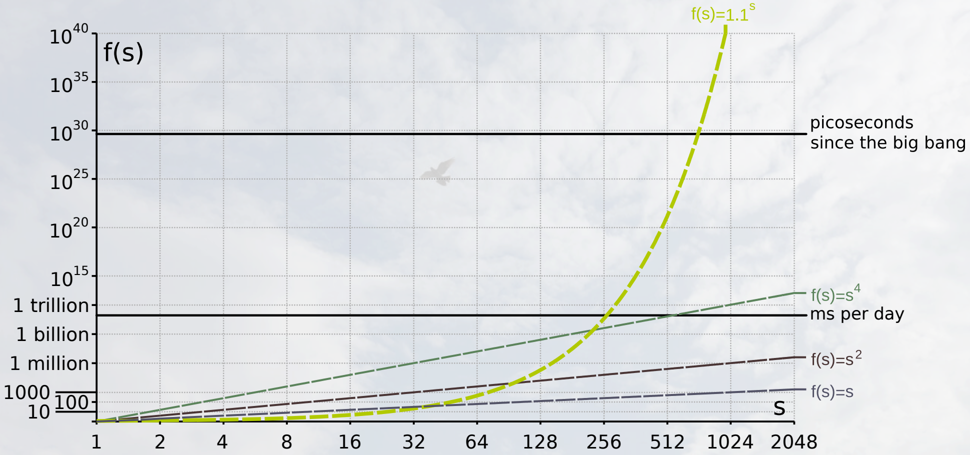
Exact vs. Heuristic Algorithms

- In optimization, there exist exact and heuristic algorithms.
- Let's look at the classical Traveling Salesperson Problem (TSP)[2,27,39,61].
- But this is nothing compared to the exponential function $f(s) = 1.1^s \dots$



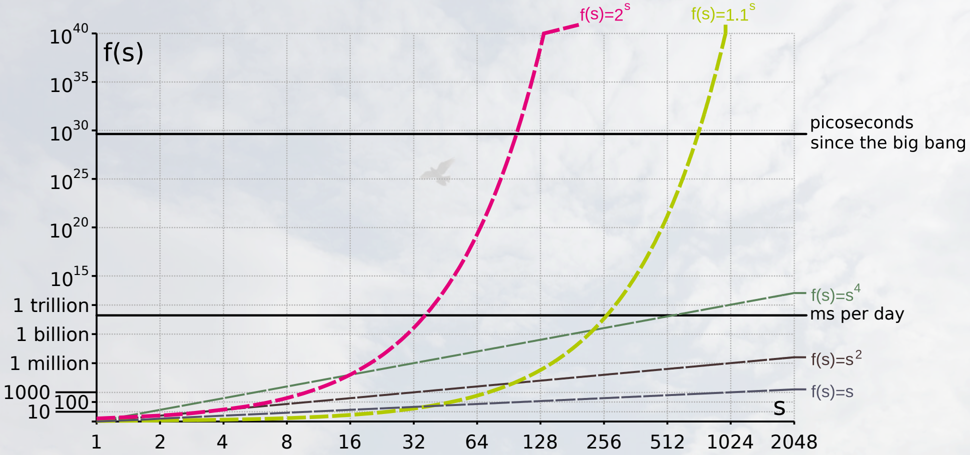
Exact vs. Heuristic Algorithms

- In optimization, there exist exact and heuristic algorithms.
- Let's look at the classical Traveling Salesperson Problem (TSP)[2,27,39,61].
- A runtime of 1.1^s becomes infeasible for $s > 512$.



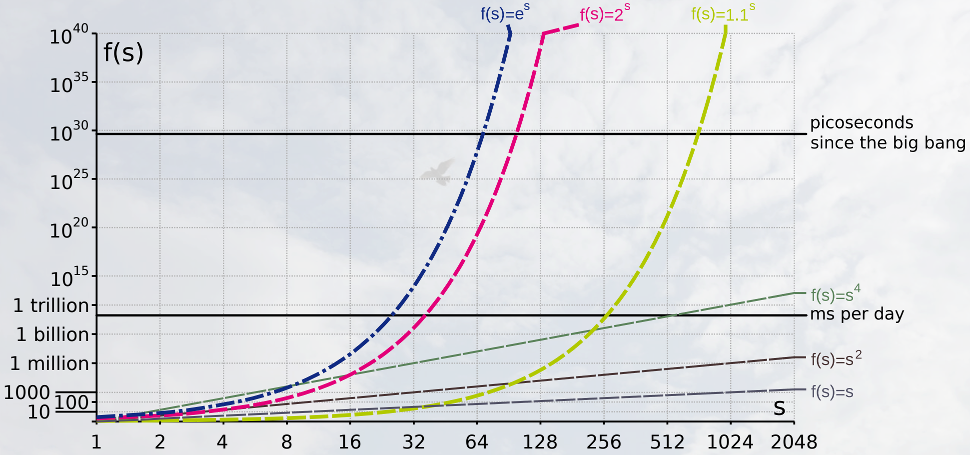
Exact vs. Heuristic Algorithms

- In optimization, there exist exact and heuristic algorithms.
- Let's look at the classical Traveling Salesperson Problem (TSP)[2,27,39,61].
- For larger bases, the runtime grows even faster.



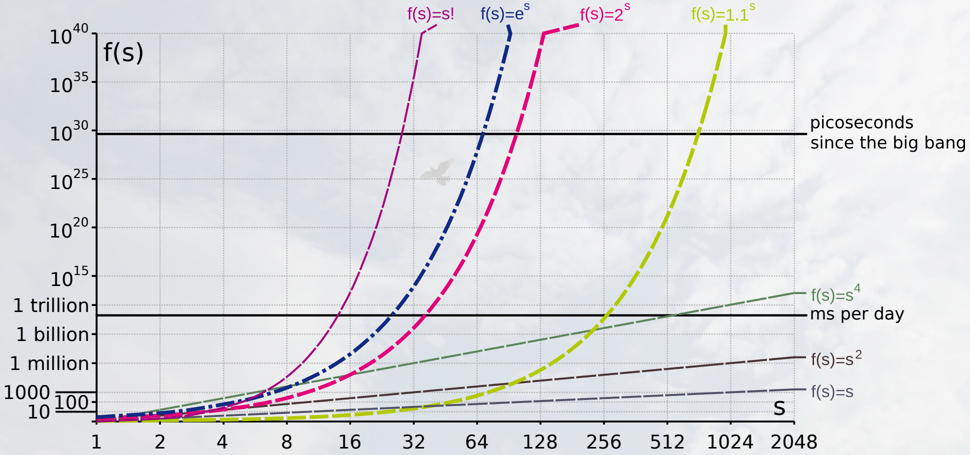
Exact vs. Heuristic Algorithms

- In optimization, there exist exact and heuristic algorithms.
- Let's look at the classical Traveling Salesperson Problem (TSP)[2,27,39,61].
- For larger bases, the runtime grows even faster.



Exact vs. Heuristic Algorithms

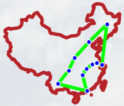
- In optimization, there exist exact and heuristic algorithms.
- Let's look at the classical Traveling Salesperson Problem (TSP)[2,27,39,61].
- If we would enumerate all possible tours of s cities in a TSP, that would be $s!$.



Exact vs. Heuristic Algorithms

- In optimization, there exist exact and heuristic algorithms.
- Let's look at the classical Traveling Salesperson Problem (TSP)[2,27,39,61].

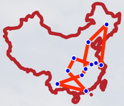
getting the optimal solution
for a TSP may take too long



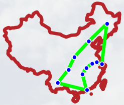
consumed runtime: very much / too (?) long

Exact vs. Heuristic Algorithms

- In optimization, there exist exact and heuristic algorithms.
- Let's look at the classical Traveling Salesperson Problem (TSP)[2,27,39,61].
    - But we can find just some tour very quickly.



getting the optimal solution
for a TSP may take too long



very little / fast — consumed runtime — very much / too (?) long

Exact vs. Heuristic Algorithms

- In optimization, there exist exact and heuristic algorithms.
- Let's look at the classical Traveling Salesperson Problem (TSP)[2,27,39,61].
  - But we can find just some tour very quickly.
  - Of course the quality of that tour will be lower.

Exact vs. Heuristic Algorithms

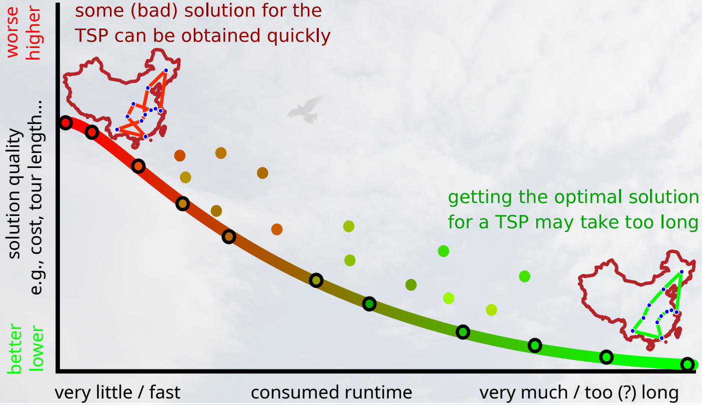
- In optimization, there exist exact and heuristic algorithms.
- Let's look at the classical Traveling Salesperson Problem (TSP)[2,27,39,61].
    - Of course the quality of that tour will be lower: the tour will be longer than the best one.

Exact vs. Heuristic Algorithms

- In optimization, there exist exact and heuristic algorithms.
- Let's look at the classical Traveling Salesperson Problem (TSP)[2,27,39,61].
  - Is there something in between?

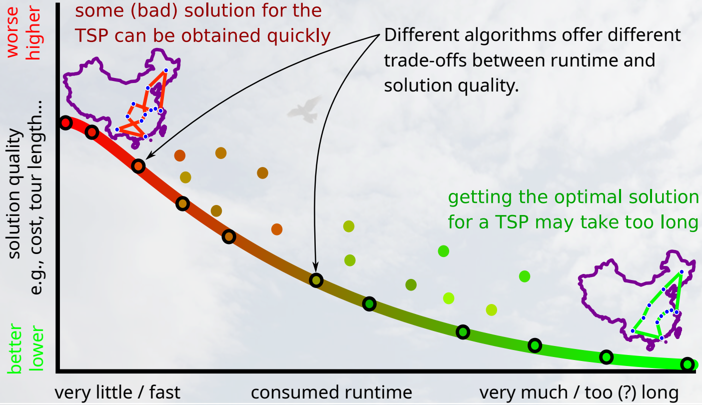
Exact vs. Heuristic Algorithms

- In optimization, there exist exact and heuristic algorithms.
- Let's look at the classical Traveling Salesperson Problem (TSP)[2,27,39,61].
    - Is there something in between?
    - (Meta-)Heuristic optimization algorithms try to find solutions which are as good as possible as fast as possible.



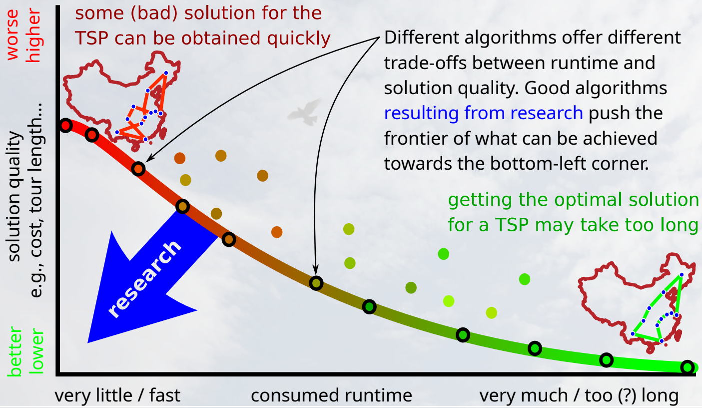
Exact vs. Heuristic Algorithms

- In optimization, there exist exact and heuristic algorithms.
- Let's look at the classical Traveling Salesperson Problem (TSP)[2,27,39,61].
  - (Meta-)Heuristic optimization algorithms try to find solutions which are as good as possible as fast as possible.
  - Optimization often means to make a trade-off between solution quality and runtime.



Exact vs. Heuristic Algorithms

- In optimization, there exist exact and heuristic algorithms.
- Let's look at the classical Traveling Salesperson Problem (TSP)[2,27,39,61].
    - Optimization often means to make a trade-off between solution quality and runtime.



Views on Performance and Time

Views on Performance

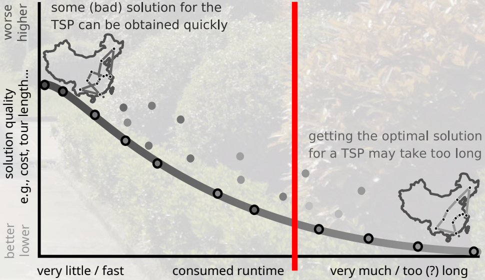
- Runtime and solution quality in optimization are intertwined and should never be considered separately.

Views on Performance

- Runtime and solution quality in optimization are intertwined and should never be considered separately.
- There are two main views on what performance is[29,30,62,63].

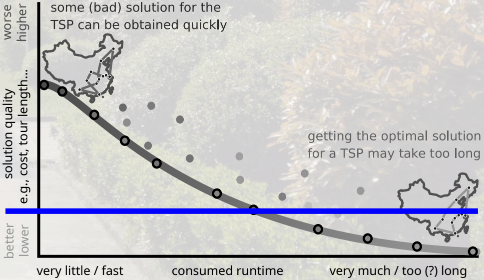
Views on Performance

- Runtime and solution quality in optimization are intertwined and should never be considered separately.
- There are two main views on what performance is[29,30,62,63]:
    1. Solution quality reached after a certain runtime



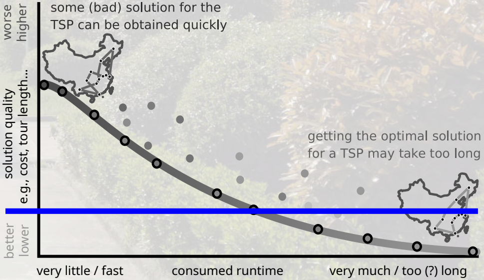
Views on Performance

- Runtime and solution quality in optimization are intertwined and should never be considered separately.
- There are two main views on what performance is[29,30,62,63]:
    1. Solution quality reached after a certain runtime
    2. Runtime to reach a certain solution quality



Views on Performance

- Runtime and solution quality in optimization are intertwined and should never be considered separately.
- There are two main views on what performance is[29,30,62,63]:
  1. Solution quality reached after a certain runtime
  2. Runtime to reach a certain solution quality



What is Runtime?

- What actually is runtime?

Clock Time as Absolute Runtime

We can measure the (absolute) consumed runtime of the algorithm in ms.

Clock Time as Absolute Runtime

We can measure the (absolute) consumed runtime of the algorithm implementation in ms.

Clock Time as Absolute Runtime

We can measure the (absolute) consumed runtime of the algorithm implementation in ms.

- Advantages

Clock Time as Absolute Runtime

We can measure the (absolute) consumed runtime of the algorithm implementation in ms.

- Advantages:
    - Results in many works reported in this format

Clock Time as Absolute Runtime

We can measure the (absolute) consumed runtime of the algorithm implementation in ms.

- Advantages:
    - Results in many works reported in this format
    - A quantity that makes physical sense

Clock Time as Absolute Runtime

We can measure the (absolute) consumed runtime of the algorithm implementation in ms.

- Advantages:
    - Results in many works reported in this format
    - A quantity that makes physical sense
    - Includes all "hidden complexities" of an algorithm implementation (memory management, matrix operations, data structures, ...)

Clock Time as Absolute Runtime

We can measure the (absolute) consumed runtime of the algorithm implementation in ms.

- Advantages:
    - Results in many works reported in this format
    - A quantity that makes physical sense
    - Includes all "hidden complexities" of an algorithm implementation (memory management, matrix operations, data structures, ...)
- Disadvantages

Clock Time as Absolute Runtime

We can measure the (absolute) consumed runtime of the algorithm implementation in ms.

- Advantages:
  - Results in many works reported in this format
  - A quantity that makes physical sense
  - Includes all "hidden complexities" of an algorithm implementation (memory management, matrix operations, data structures, . . .)
- Disadvantages:
  - Strongly machine dependent and inherently incomparable over different machines

Clock Time as Absolute Runtime

We can measure the (absolute) consumed runtime of the algorithm implementation in ms.

- Advantages:
  - Results in many works reported in this format
  - A quantity that makes physical sense
  - Includes all "hidden complexities" of an algorithm implementation (memory management, matrix operations, data structures, ...)
- Disadvantages:
  - Strongly machine dependent and inherently incomparable over different machines
  - Measurements are only valuable for a few years

Clock Time as Absolute Runtime

We can measure the (absolute) consumed runtime of the algorithm implementation in ms.

- Advantages:
  - Results in many works reported in this format
  - A quantity that makes physical sense
  - Includes all "hidden complexities" of an algorithm implementation (memory management, matrix operations, data structures, ...)
- Disadvantages:
  - Strongly machine dependent and inherently incomparable over different machines
  - Measurements are only valuable for a few years
  - Can be biased by "outside effects," e.g., OS, scheduling, other processes, I/O, swapping, ...

Clock Time as Absolute Runtime

We can measure the (absolute) consumed runtime of the algorithm implementation in ms.

- Advantages:
  - Results in many works reported in this format
  - A quantity that makes physical sense
  - Includes all "hidden complexities" of an algorithm implementation (memory management, matrix operations, data structures, ...)
- Disadvantages:
  - Strongly machine dependent and inherently incomparable over different machines
  - Measurements are only valuable for a few years
  - Can be biased by "outside effects," e.g., OS, scheduling, other processes, I/O, swapping, ...
- Hardware, software, OS, programming language, etc. all have nothing to do with the optimization algorithm itself and are relevant only in a specific application...

Clock Time as Absolute Runtime

We can measure the (absolute) consumed runtime of the algorithm implementation in ms.

- Advantages:
  - Results in many works reported in this format
  - A quantity that makes physical sense
  - Includes all "hidden complexities" of an algorithm implementation (memory management, matrix operations, data structures, ...)
- Disadvantages:
  - Strongly machine dependent and inherently incomparable over different machines
  - Measurements are only valuable for a few years
  - Can be biased by "outside effects," e.g., OS, scheduling, other processes, I/O, swapping, ...
- Hardware, software, OS, programming language, etc. all have nothing to do with the optimization algorithm itself and are relevant only in a specific application...
- ...for research they may be less interesting, while for a specific application they do matter.

Objective Function Evaluations: FEs

We can measure (count) the objective function evaluations (FEs), i.e., the number of tested candidate solutions.

Objective Function Evaluations: FEs

We can measure (count) the objective function evaluations (FEs), i.e., the number of tested candidate solutions.

- Advantages

Objective Function Evaluations: FEs

We can measure (count) the objective function evaluations (FEs), i.e., the number of tested candidate solutions.

- Advantages:
    - Results in many works reported in this format (or FEs can be deduced)

Objective Function Evaluations: FEs

We can measure (count) the objective function evaluations (FEs), i.e., the number of tested candidate solutions.

- Advantages:
  - Results in many works reported in this format (or FEs can be deduced)
  - Machine-independent, theory-related measure

Objective Function Evaluations: FEs

We can measure (count) the objective function evaluations (FEs), i.e., the number of tested candidate solutions.

- Advantages:
  - Results in many works reported in this format (or FEs can be deduced)
  - Machine-independent, theory-related measure
  - Cannot be influenced by "outside effects"

Objective Function Evaluations: FEs

We can measure (count) the objective function evaluations (FEs), i.e., the number of tested candidate solutions.

- Advantages:
  - Results in many works reported in this format (or FEs can be deduced)
  - Machine-independent, theory-related measure
  - Cannot be influenced by “outside effects”
  - In many optimization problems, computing the objective value is the most time consuming task

Objective Function Evaluations: FEs

We can measure (count) the objective function evaluations (FEs), i.e., the number of tested candidate solutions.

- Advantages:
    - Results in many works reported in this format (or FEs can be deduced)
    - Machine-independent, theory-related measure
    - Cannot be influenced by "outside effects"
    - In many optimization problems, computing the objective value is the most time consuming task
- Disadvantages

Objective Function Evaluations: FEs

We can measure (count) the objective function evaluations (FEs), i.e., the number of tested candidate solutions.

- Advantages:
  - Results in many works reported in this format (or FEs can be deduced)
  - Machine-independent, theory-related measure
  - Cannot be influenced by “outside effects”
  - In many optimization problems, computing the objective value is the most time consuming task
- Disadvantages:
  - No clear relationship to real runtime

Objective Function Evaluations: FEs

We can measure (count) the objective function evaluations (FEs), i.e., the number of tested candidate solutions.

- Advantages:
  - Results in many works reported in this format (or FEs can be deduced)
  - Machine-independent, theory-related measure
  - Cannot be influenced by “outside effects”
  - In many optimization problems, computing the objective value is the most time consuming task
- Disadvantages:
  - No clear relationship to real runtime
  - Does not contain “hidden complexities” of algorithm

Objective Function Evaluations: FEs

We can measure (count) the objective function evaluations (FEs), i.e., the number of tested candidate solutions.

- Advantages:
  - Results in many works reported in this format (or FEs can be deduced)
  - Machine-independent, theory-related measure
  - Cannot be influenced by "outside effects"
  - In many optimization problems, computing the objective value is the most time consuming task
- Disadvantages:
  - No clear relationship to real runtime
  - Does not contain "hidden complexities" of algorithm
  - 1 FE: very different costs in different situations![61]

Objective Function Evaluations: FEs

We can measure (count) the objective function evaluations (FEs), i.e., the number of tested candidate solutions.

- Advantages:
  - Results in many works reported in this format (or FEs can be deduced)
  - Machine-independent, theory-related measure
  - Cannot be influenced by "outside effects"
  - In many optimization problems, computing the objective value is the most time consuming task
- Disadvantages:
  - No clear relationship to real runtime
  - Does not contain "hidden complexities" of algorithm
  - 1 FE: very different costs in different situations![61]
    - When applying a local search that swaps two cities in each move to the TSP, 1 FE can be done in $\mathcal{O}(1)$[61].

Objective Function Evaluations: FEs

We can measure (count) the objective function evaluations (FEs), i.e., the number of tested candidate solutions.

- Advantages:
  - Results in many works reported in this format (or FEs can be deduced)
  - Machine-independent, theory-related measure
  - Cannot be influenced by "outside effects"
  - In many optimization problems, computing the objective value is the most time consuming task
- Disadvantages:
  - No clear relationship to real runtime
  - Does not contain "hidden complexities" of algorithm
  - 1 FE: very different costs in different situations![61]
    - When applying a local search that swaps two cities in each move to the TSP, 1 FE can be done in $\mathcal{O}(1)$[61].
    - When applying Ant Colony Optimization (ACO) instead, each FE takes $\mathcal{O}(s^2)$[61].

Objective Function Evaluations: FEs

We can measure (count) the objective function evaluations (FEs), i.e., the number of tested candidate solutions.

- Advantages:
  - Results in many works reported in this format (or FEs can be deduced)
  - Machine-independent, theory-related measure
  - Cannot be influenced by "outside effects"
  - In many optimization problems, computing the objective value is the most time consuming task
- Disadvantages:
  - No clear relationship to real runtime
  - Does not contain "hidden complexities" of algorithm
  - 1 FE: very different costs in different situations![61]
    - When applying a local search that swaps two cities in each move to the TSP, 1 FE can be done in $\mathcal{O}(1)$[61].
    - When applying Ant Colony Optimization (ACO) instead, each FE takes $\mathcal{O}(s^2)$[61].
- Relevant for comparing algorithms, but not so much for the practical application or comparing implementations.

Do not count generations

- In an evolutionary algorithm (EA)[5,59], in each generation (= iteration), a set of new solution is created and evaluated.

Do not count generations

- In an evolutionary algorithm (EA)[5,59], in each generation (= iteration), a set of new solution is created and evaluated.
- Traditionally, the number of generations passed until some goal was reached was used in the EA community.

Do not count generations

- In an evolutionary algorithm (EA)[5,59], in each generation (= iteration), a set of new solution is created and evaluated.
- Traditionally, the number of generations passed until some goal was reached was used in the EA community.
- Do not use the number of *generations* in your EA as time measure! Instead count the FEs.

Do not count generations

- In an evolutionary algorithm (EA)[5,59], in each generation (= iteration), a set of new solution is created and evaluated.
- Traditionally, the number of generations passed until some goal was reached was used in the EA community.
- Do not use the number of *generations* in your EA as time measure! Instead count the FEs, because:
- The "number of generations" are not really comparable for different population sizes or with algorithms that do not use populations.

Do not count generations

- In an evolutionary algorithm (EA)[5,59], in each generation (= iteration), a set of new solution is created and evaluated.
- Traditionally, the number of generations passed until some goal was reached was used in the EA community.
- Do not use the number of *generations* in your EA as time measure! Instead count the FEs, because:
- The "number of generations" are not really comparable for different population sizes or with algorithms that do not use populations.
- Often, the mapping between generations and FEs is not clear

Do not count generations

- In an evolutionary algorithm (EA)[5,59], in each generation (= iteration), a set of new solution is created and evaluated.
- Traditionally, the number of generations passed until some goal was reached was used in the EA community.
- Do not use the number of *generations* in your EA as time measure! Instead count the FEs, because:
- The "number of generations" are not really comparable for different population sizes or with algorithms that do not use populations.
- Often, the mapping between generations and FEs is not clear, for example
    - Do you evaluate offspring solutions that are identical to their parents?

Do not count generations

- In an evolutionary algorithm (EA)[5,59], in each generation (= iteration), a set of new solution is created and evaluated.
- Traditionally, the number of generations passed until some goal was reached was used in the EA community.
- Do not use the number of *generations* in your EA as time measure! Instead count the FEs, because:
- The "number of generations" are not really comparable for different population sizes or with algorithms that do not use populations.
- Often, the mapping between generations and FEs is not clear, for example
  - Do you evaluate offspring solutions that are identical to their parents?
  - Is a local search involved that refines some or all solutions in the population?

Do not count generations

- In an evolutionary algorithm (EA)[5,59], in each generation (= iteration), a set of new solution is created and evaluated.
- Traditionally, the number of generations passed until some goal was reached was used in the EA community.
- Do not use the number of *generations* in your EA as time measure! Instead count the FEs, because:
- The "number of generations" are not really comparable for different population sizes or with algorithms that do not use populations.
- Often, the mapping between generations and FEs is not clear, for example
  - Do you evaluate offspring solutions that are identical to their parents?
  - Is a local search involved that refines some or all solutions in the population?
  - In a $(\mu + \lambda)$-EA, is the first population of size $\mu + \lambda$, λ, or μ?

Do not count generations

- In an evolutionary algorithm (EA)[5,59], in each generation (= iteration), a set of new solution is created and evaluated.
- Traditionally, the number of generations passed until some goal was reached was used in the EA community.
- Do not use the number of *generations* in your EA as time measure! Instead count the FEs, because:
- The "number of generations" are not really comparable for different population sizes or with algorithms that do not use populations.
- Often, the mapping between generations and FEs is not clear, for example
  - Do you evaluate offspring solutions that are identical to their parents?
  - Is a local search involved that refines some or all solutions in the population?
  - In a $(\mu + \lambda)$-EA, is the first population of size $\mu + \lambda$, λ, or μ?
  - What if the population size changes adaptively?

Do not count generations

- In an evolutionary algorithm (EA)[5,59], in each generation (= iteration), a set of new solution is created and evaluated.
- Traditionally, the number of generations passed until some goal was reached was used in the EA community.
- Do not use the number of *generations* in your EA as time measure! Instead count the FEs, because:
- The "number of generations" are not really comparable for different population sizes or with algorithms that do not use populations.
- Often, the mapping between generations and FEs is not clear, for example
  - Do you evaluate offspring solutions that are identical to their parents?
  - Is a local search involved that refines some or all solutions in the population?
  - In a $(\mu + \lambda)$-EA, is the first population of size $\mu + \lambda$, λ, or μ?
  - What if the population size changes adaptively?
- I suggest to prefer FEs over generations if you want to count algorithm steps.

Runtime

- I suggest to always measure both the consumed FEs and the runtime in milliseconds.

Runtime

- I suggest to always measure both the consumed FEs and the runtime in milliseconds.
- Anyway, with what we have learned, we can rewrite the two views by choosing a time measure[29,62]

Runtime

- I suggest to always measure both the consumed FEs and the runtime in milliseconds.
- Anyway, with what we have learned, we can rewrite the two views by choosing a time measure[29,62], e.g.:
    1. Solution quality reached after a certain number of FEs

Runtime

- I suggest to always measure both the consumed FEs and the runtime in milliseconds.
- Anyway, with what we have learned, we can rewrite the two views by choosing a time measure[29,62], e.g.:
  1. Solution quality reached after a certain number of FEs
  2. Milliseconds needed to reach a certain solution quality

Solution Quality

- Common measure of solution quality: Objective function value of best solution discovered.

Solution Quality

- Common measure of solution quality: Objective function value of best solution discovered.
- Rewrite the two views[29,62]

Solution Quality

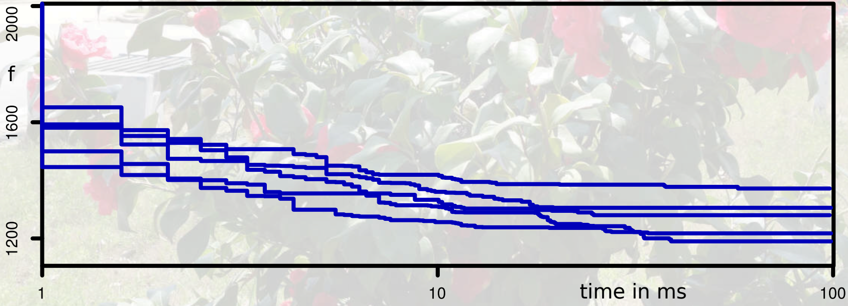
- Common measure of solution quality: Objective function value of best solution discovered.
- Rewrite the two views[29,62]:
    1. Best objective function value reached after a certain number of milliseconds

Solution Quality

- Common measure of solution quality: Objective function value of best solution discovered.
- Rewrite the two views[29,62]:
  1. Best objective function value reached after a certain number of milliseconds
  2. Number of FEs needed to reach a certain objective function value

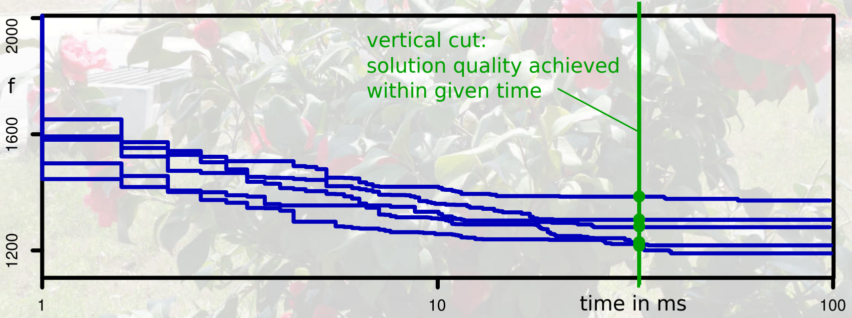
Views on Performance

- Which one is the “better” view on performance?



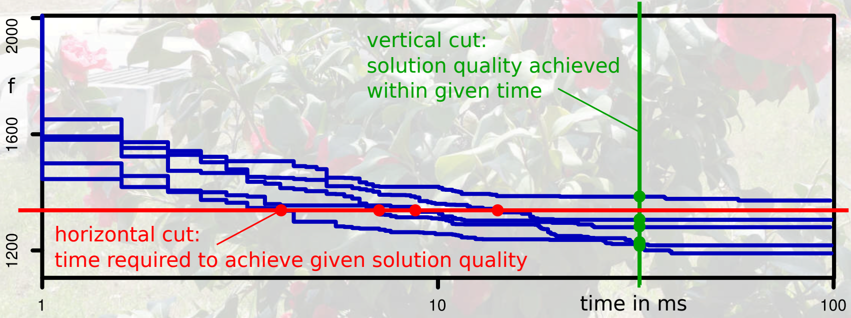
Views on Performance

- Which one is the “better” view on performance?
  1. Best objective function value reached after a certain number of FEs



Views on Performance

- Which one is the "better" view on performance?
  1. Best objective function value reached after a certain number of FEs
  2. Number of FEs needed to reach a certain objective function value



Views on Performance

- Which one is the "better" view on performance?
    1. Best objective function value reached after a certain number of FEs
    2. Number of FEs needed to reach a certain objective function value
- This question is still debated in research...

Which view is better?

- Number of FEs needed to reach a certain objective function value
- Preferred by, e.g., the BBOB/COCO benchmark suite[29]

Which view is better?

- Number of FEs needed to reach a certain objective function value
- Preferred by, e.g., the BBOB/COCO benchmark suite[29]:
  - Measuring the time needed to reach a target function value allows meaningful statements such as "Algorithm A is two/ten/hundred times faster than Algorithm B in solving this problem."

Which view is better?

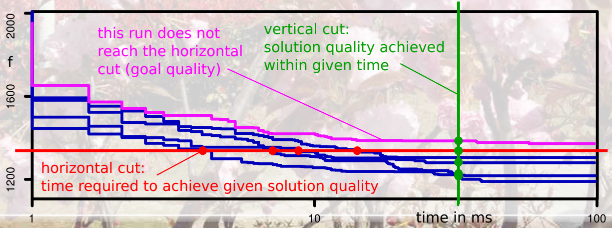
- Number of FEs needed to reach a certain objective function value
- Preferred by, e.g., the BBOB/COCO benchmark suite[29]:
  - Measuring the time needed to reach a target function value allows meaningful statements such as "Algorithm A is two/ten/hundred times faster than Algorithm B in solving this problem."
  - However, there is no interpretable meaning to the fact that Algorithm A reaches a function value that is two/ten/hundred times smaller than the one reached by Algorithm B.

Which view is better?

- Number of FEs needed to reach a certain objective function value
- Preferred by, e.g., the BBOB/COCO benchmark suite[29]:
  - Measuring the time needed to reach a target function value allows meaningful statements such as "Algorithm A is two/ten/hundred times faster than Algorithm B in solving this problem."
  - However, there is no interpretable meaning to the fact that Algorithm A reaches a function value that is two/ten/hundred times smaller than the one reached by Algorithm B.
  - "Benchmarking Theory Perspective"

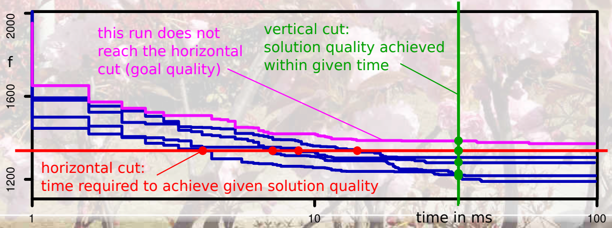
Which view is better?

- Number of FEs needed to reach a certain objective function value
- Preferred by, e.g., the BBOB/COCO benchmark suite[29]:
  - Measuring the time needed to reach a target function value allows meaningful statements such as “Algorithm A is two/ten/hundred times faster than Algorithm B in solving this problem.”
  - However, there is no interpretable meaning to the fact that Algorithm A reaches a function value that is two/ten/hundred times smaller than the one reached by Algorithm B.
  - “Benchmarking Theory Perspective”
- Sometimes problematic: What if one run does not reach the goal quality?



Which view is better?

- Number of FEs needed to reach a certain objective function value
- Preferred by, e.g., the BBOB/COCO benchmark suite[29]:
  - Measuring the time needed to reach a target function value allows meaningful statements such as "Algorithm A is two/ten/hundred times faster than Algorithm B in solving this problem."
  - However, there is no interpretable meaning to the fact that Algorithm A reaches a function value that is two/ten/hundred times smaller than the one reached by Algorithm B.
  - "Benchmarking Theory Perspective"
- Sometimes problematic: What if one run does not reach the goal quality?
- Then, alternative measures need to be computed, such as the ERT[3,47] or PAR2 and PAR10[8,36].



Which view is better?

- Best objective function value reached after a certain number of FEs

Which view is better?

- Best objective function value reached after a certain number of FEs
- Preferred by many benchmark suites such as[55].

Which view is better?

- Best objective function value reached after a certain number of FEs
- Preferred by many benchmark suites such as[55].
- Practice Perspective: Best results achievable with given time budget wins.

Which view is better?

- Best objective function value reached after a certain number of FEs
- Preferred by many benchmark suites such as[55].
- Practice Perspective: Best results achievable with given time budget wins.
- This perspective maybe less suitable for scientific benchmarking, but surely is useful in practice.

Which view is better?

- Best objective function value reached after a certain number of FEs
- Preferred by many benchmark suites such as[55].
- Practice Perspective: Best results achievable with given time budget wins.
- This perspective maybe less suitable for scientific benchmarking, but surely is useful in practice.
- “How good is the tour for the TSP that we can find in 5 minutes with our algorithm?”

Which view is better?

- Best objective function value reached after a certain number of FEs
- Preferred by many benchmark suites such as[55].
- Practice Perspective: Best results achievable with given time budget wins.
- This perspective maybe less suitable for scientific benchmarking, but surely is useful in practice.
- "How good is the tour for the TSP that we can find in 5 minutes with our algorithm?"
- Always well-defined, because vertical cuts can always be reached.

Views on Performance

- No official consensus on which view is “better.”

Views on Performance

- No official consensus on which view is “better.”
- This also strongly depends on the situation.

Views on Performance

- No official consensus on which view is "better."
- This also strongly depends on the situation.
- If we can actually always solve the problem to a "natural" goal quality (e.g., to optimality), then we should prefer the horizontal cut (time-to-target) method.

Views on Performance

- No official consensus on which view is “better.”
- This also strongly depends on the situation.
- If we can actually always solve the problem to a “natural” goal quality (e.g., to optimality), then we should prefer the horizontal cut (time-to-target) method.
- If we have clear application requirements specifying a fixed budget, then we should prefer the fixed-budget approach.

Views on Performance

- No official consensus on which view is "better."
- This also strongly depends on the situation.
- If we can actually always solve the problem to a "natural" goal quality (e.g., to optimality), then we should prefer the horizontal cut (time-to-target) method.
- If we have clear application requirements specifying a fixed budget, then we should prefer the fixed-budget approach.
- Otherwise, the best approach may be: Evaluate algorithm according to both methods.[61–63]

Views on Performance

- No official consensus on which view is “better.”
- This also strongly depends on the situation.
- If we can actually always solve the problem to a “natural” goal quality (e.g., to optimality), then we should prefer the horizontal cut (time-to-target) method.
- If we have clear application requirements specifying a fixed budget, then we should prefer the fixed-budget approach.
- Otherwise, the best approach may be: Evaluate algorithm according to both methods.[61–63]
- Maybe cast a net of several horizontal and vertical cuts, to get a better picture. . .

Determining Target Values

- How to determine the right maximum FEs or target function values?

Determining Target Values

- How to determine the right maximum FEs or target function values?
    1. from the constraints of a practical application

Determining Target Values

- How to determine the right maximum FEs or target function values?
  1. from the constraints of a practical application
  2. from studies in literature regarding similar or the same problem

Determining Target Values

- How to determine the right maximum FEs or target function values?
    1. from the constraints of a practical application
    2. from studies in literature regarding similar or the same problem
    3. from simple or well-known algorithms

Determining Target Values

- How to determine the right maximum FEs or target function values?
  1. from the constraints of a practical application
  2. from studies in literature regarding similar or the same problem
  3. from simple or well-known algorithms
  4. from experience

Determining Target Values

- How to determine the right maximum FEs or target function values?
  1. from the constraints of a practical application
  2. from studies in literature regarding similar or the same problem
  3. from simple or well-known algorithms
  4. from experience
  5. from prior, small-scale experiments

Determining Target Values

- How to determine the right maximum FEs or target function values?
    1. from the constraints of a practical application
    2. from studies in literature regarding similar or the same problem
    3. from simple or well-known algorithms
    4. from experience
    5. from prior, small-scale experiments
    6. based on known results or well-accepted bounds

Statistical Measures

Problem Instances and Randomized Algorithms

- For each optimization problem (like the TSP) there are several instances (e.g., different sets of cities that need to be visited).

Problem Instances and Randomized Algorithms

- For each optimization problem (like the TSP) there are several instances (e.g., different sets of cities that need to be visited).
  - Some instances will be easy, some will be hard.

Problem Instances and Randomized Algorithms

- For each optimization problem (like the TSP) there are several instances (e.g., different sets of cities that need to be visited).
  - Some instances will be easy, some will be hard.
  - We always must use multiple different problem instances to get reliable results.

Problem Instances and Randomized Algorithms

- For each optimization problem (like the TSP) there are several instances (e.g., different sets of cities that need to be visited).
  - Some instances will be easy, some will be hard.
  - We always must use multiple different problem instances to get reliable results.
  - Performance indicators need to be computed for each instance and also summarized over several instances.

Problem Instances and Randomized Algorithms

- For each optimization problem (like the TSP) there are several instances (e.g., different sets of cities that need to be visited).
  - Some instances will be easy, some will be hard.
  - We always must use multiple different problem instances to get reliable results.
  - Performance indicators need to be computed for each instance and also summarized over several instances.
- Special situation: Randomized Algorithms

Problem Instances and Randomized Algorithms

- For each optimization problem (like the TSP) there are several instances (e.g., different sets of cities that need to be visited).
  - Some instances will be easy, some will be hard.
  - We always must use multiple different problem instances to get reliable results.
  - Performance indicators need to be computed for each instance and also summarized over several instances.
- Special situation: Randomized Algorithms:
  - Performance values cannot be given as an “absolute” value!

Problem Instances and Randomized Algorithms

- For each optimization problem (like the TSP) there are several instances (e.g., different sets of cities that need to be visited).
  - Some instances will be easy, some will be hard.
  - We always must use multiple different problem instances to get reliable results.
  - Performance indicators need to be computed for each instance and also summarized over several instances.
- Special situation: Randomized Algorithms:
  - Performance values cannot be given as an "absolute" value!
  - 1 run = 1 application of an optimization algorithm to a problem, runs are independent from all prior runs.

Problem Instances and Randomized Algorithms

- For each optimization problem (like the TSP) there are several instances (e.g., different sets of cities that need to be visited).
  - Some instances will be easy, some will be hard.
  - We always must use multiple different problem instances to get reliable results.
  - Performance indicators need to be computed for each instance and also summarized over several instances.
- Special situation: Randomized Algorithms:
  - Performance values cannot be given as an "absolute" value!
  - 1 run = 1 application of an optimization algorithm to a problem, runs are independent from all prior runs.
  - Results can be different for each run!

Problem Instances and Randomized Algorithms

- For each optimization problem (like the TSP) there are several instances (e.g., different sets of cities that need to be visited).
  - Some instances will be easy, some will be hard.
  - We always must use multiple different problem instances to get reliable results.
  - Performance indicators need to be computed for each instance and also summarized over several instances.
- Special situation: Randomized Algorithms:
  - Performance values cannot be given as an “absolute” value!
  - 1 run = 1 application of an optimization algorithm to a problem, runs are independent from all prior runs.
  - Results can be different for each run!
  - Executing a randomized algorithm one time does not give reliable information.

Problem Instances and Randomized Algorithms

- For each optimization problem (like the TSP) there are several instances (e.g., different sets of cities that need to be visited).
  - Some instances will be easy, some will be hard.
  - We always must use multiple different problem instances to get reliable results.
  - Performance indicators need to be computed for each instance and also summarized over several instances.
- Special situation: Randomized Algorithms:
  - Performance values cannot be given as an "absolute" value!
  - 1 run = 1 application of an optimization algorithm to a problem, runs are independent from all prior runs.
  - Results can be different for each run!
  - Executing a randomized algorithm one time does not give reliable information.
  - Statistical evaluation over sets of runs necessary.

Important Distinction

- Crucial Difference: distribution and sample

Important Distinction

- Crucial Difference: distribution and sample
- A sample is what we *measure*.

Important Distinction

- Crucial Difference: distribution and sample
- A sample is what we *measure*.
- A distribution is the asymptotic result of the ideal process.

Important Distinction

- Crucial Difference: distribution and sample
- A sample is what we *measure*.
- A distribution is the asymptotic result of the ideal process.
- Statistical parameters of the distribution can be estimated from a sample.

Important Distinction

- Crucial Difference: distribution and sample
- A sample is what we *measure*.
- A distribution is the asymptotic result of the ideal process.
- Statistical parameters of the distribution can be estimated from a sample.
- Example: Dice Throw

Important Distinction

- Crucial Difference: distribution and sample
- A sample is what we *measure*.
- A distribution is the asymptotic result of the ideal process.
- Statistical parameters of the distribution can be estimated from a sample.
- Example: Dice Throw
- How likely is it to roll a 1, 2, 3, 4, 5, or 6?

Important Distinction

# throws	number	f(1)	f(2)	f(3)	f(4)	f(5)	f(6)
1	5	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000

Important Distinction

# throws	number	f(1)	f(2)	f(3)	f(4)	f(5)	f(6)
1	5	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000
2	4	0.0000	0.0000	0.0000	0.5000	0.5000	0.0000

Important Distinction

# throws	number	f(1)	f(2)	f(3)	f(4)	f(5)	f(6)
1	5	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000
2	4	0.0000	0.0000	0.0000	0.5000	0.5000	0.0000
3	1	0.3333	0.0000	0.0000	0.3333	0.3333	0.0000

Important Distinction

# throws	number	f(1)	f(2)	f(3)	f(4)	f(5)	f(6)
1	5	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000
2	4	0.0000	0.0000	0.0000	0.5000	0.5000	0.0000
3	1	0.3333	0.0000	0.0000	0.3333	0.3333	0.0000
4	4	0.2500	0.0000	0.0000	0.5000	0.2500	0.0000

Important Distinction

# throws	number	f(1)	f(2)	f(3)	f(4)	f(5)	f(6)
1	5	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000
2	4	0.0000	0.0000	0.0000	0.5000	0.5000	0.0000
3	1	0.3333	0.0000	0.0000	0.3333	0.3333	0.0000
4	4	0.2500	0.0000	0.0000	0.5000	0.2500	0.0000
5	3	0.2000	0.0000	0.2000	0.4000	0.2000	0.0000

Important Distinction

# throws	number	f(1)	f(2)	f(3)	f(4)	f(5)	f(6)
1	5	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000
2	4	0.0000	0.0000	0.0000	0.5000	0.5000	0.0000
3	1	0.3333	0.0000	0.0000	0.3333	0.3333	0.0000
4	4	0.2500	0.0000	0.0000	0.5000	0.2500	0.0000
5	3	0.2000	0.0000	0.2000	0.4000	0.2000	0.0000
6	3	0.1667	0.0000	0.3333	0.3333	0.1667	0.0000

Important Distinction

# throws	number	f(1)	f(2)	f(3)	f(4)	f(5)	f(6)
1	5	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000
2	4	0.0000	0.0000	0.0000	0.5000	0.5000	0.0000
3	1	0.3333	0.0000	0.0000	0.3333	0.3333	0.0000
4	4	0.2500	0.0000	0.0000	0.5000	0.2500	0.0000
5	3	0.2000	0.0000	0.2000	0.4000	0.2000	0.0000
6	3	0.1667	0.0000	0.3333	0.3333	0.1667	0.0000
7	2	0.1429	0.1429	0.2857	0.2857	0.1429	0.0000
8	1	0.2500	0.1250	0.2500	0.2500	0.1250	0.0000
9	4	0.2222	0.1111	0.2222	0.3333	0.1111	0.0000
10	2	0.2000	0.2000	0.2000	0.3000	0.1000	0.0000

Important Distinction

# throws	number	f(1)	f(2)	f(3)	f(4)	f(5)	f(6)
1	5	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000
2	4	0.0000	0.0000	0.0000	0.5000	0.5000	0.0000
3	1	0.3333	0.0000	0.0000	0.3333	0.3333	0.0000
4	4	0.2500	0.0000	0.0000	0.5000	0.2500	0.0000
5	3	0.2000	0.0000	0.2000	0.4000	0.2000	0.0000
6	3	0.1667	0.0000	0.3333	0.3333	0.1667	0.0000
7	2	0.1429	0.1429	0.2857	0.2857	0.1429	0.0000
8	1	0.2500	0.1250	0.2500	0.2500	0.1250	0.0000
9	4	0.2222	0.1111	0.2222	0.3333	0.1111	0.0000
10	2	0.2000	0.2000	0.2000	0.3000	0.1000	0.0000
11	6	0.1818	0.1818	0.1818	0.2727	0.0909	0.0909
12	3	0.1667	0.1667	0.2500	0.2500	0.0833	0.0833

Important Distinction

# throws	number	f(1)	f(2)	f(3)	f(4)	f(5)	f(6)
1	5	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000
2	4	0.0000	0.0000	0.0000	0.5000	0.5000	0.0000
3	1	0.3333	0.0000	0.0000	0.3333	0.3333	0.0000
4	4	0.2500	0.0000	0.0000	0.5000	0.2500	0.0000
5	3	0.2000	0.0000	0.2000	0.4000	0.2000	0.0000
6	3	0.1667	0.0000	0.3333	0.3333	0.1667	0.0000
7	2	0.1429	0.1429	0.2857	0.2857	0.1429	0.0000
8	1	0.2500	0.1250	0.2500	0.2500	0.1250	0.0000
9	4	0.2222	0.1111	0.2222	0.3333	0.1111	0.0000
10	2	0.2000	0.2000	0.2000	0.3000	0.1000	0.0000
11	6	0.1818	0.1818	0.1818	0.2727	0.0909	0.0909
12	3	0.1667	0.1667	0.2500	0.2500	0.0833	0.0833
100	...	0.1900	0.2100	0.1500	0.1600	0.1200	0.1700
1'000	...	0.1700	0.1670	0.1620	0.1670	0.1570	0.1770
10'000	...	0.1682	0.1699	0.1680	0.1661	0.1655	0.1623
100'000	...	0.1671	0.1649	0.1664	0.1676	0.1668	0.1672
1'000'000	...	0.1673	0.1663	0.1662	0.1673	0.1666	0.1664

Important Distinction

# throws	number	f(1)	f(2)	f(3)	f(4)	f(5)	f(6)
1	5	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000
2	4	0.0000	0.0000	0.0000	0.5000	0.5000	0.0000
3	1	0.3333	0.0000	0.0000	0.3333	0.3333	0.0000
4	4	0.2500	0.0000	0.0000	0.5000	0.2500	0.0000
5	3	0.2000	0.0000	0.2000	0.4000	0.2000	0.0000
6	3	0.1667	0.0000	0.3333	0.3333	0.1667	0.0000
7	2	0.1429	0.1429	0.2857	0.2857	0.1429	0.0000
8	1	0.2500	0.1250	0.2500	0.2500	0.1250	0.0000
9	4	0.2222	0.1111	0.2222	0.3333	0.1111	0.0000
10	2	0.2000	0.2000	0.2000	0.3000	0.1000	0.0000
11	6	0.1818	0.1818	0.1818	0.2727	0.0909	0.0909
12	3	0.1667	0.1667	0.2500	0.2500	0.0833	0.0833
100	...	0.1900	0.2100	0.1500	0.1600	0.1200	0.1700
1'000	...	0.1700	0.1670	0.1620	0.1670	0.1570	0.1770
10'000	...	0.1682	0.1699	0.1680	0.1661	0.1655	0.1623
100'000	...	0.1671	0.1649	0.1664	0.1676	0.1668	0.1672
1'000'000	...	0.1673	0.1663	0.1662	0.1673	0.1666	0.1664
10'000'000	...	0.1667	0.1667	0.1666	0.1668	0.1667	0.1665
100'000'000	...	0.1667	0.1666	0.1666	0.1667	0.1667	0.1667
1'000'000'000	...	0.1667	0.1667	0.1667	0.1667	0.1667	0.1667

Important Distinction

- Crucial Difference: distribution and sample
- A sample is what we *measure*.
- A distribution is the asymptotic result of the ideal process.
- Statistical parameters of the distribution can be estimated from a sample.
- Example: Dice Throw
- How likely is it to roll a 1, 2, 3, 4, 5, or 6?
- All statistically determined parameters are just estimates based on measurements.

Important Distinction

- Crucial Difference: distribution and sample
- A sample is what we *measure*.
- A distribution is the asymptotic result of the ideal process.
- Statistical parameters of the distribution can be estimated from a sample.
- Example: Dice Throw
- How likely is it to roll a 1, 2, 3, 4, 5, or 6?
- All statistically determined parameters are just estimates based on measurements.
- The parameters of a random process cannot be measured directly, but only be estimated from multiple measures.

Measures of the Average

- Assume that we have obtained a sample $A = (a_0, a_1, \ldots, a_{n-1})$ of n observations from an experiment.

Measures of the Average

- Assume that we have obtained a sample $A = (a_0, a_1, \dots, a_{n-1})$ of n observations from an experiment, e.g., we have measured the qualities a_i of the best discovered solutions of $n = 101$ independent runs of an optimization algorithm.

Measures of the Average

- Assume that we have obtained a sample $A = (a_0, a_1, \dots, a_{n-1})$ of n observations from an experiment, e.g., we have measured the qualities a_i of the best discovered solutions of $n = 101$ independent runs of an optimization algorithm.
- We usually want to reduce this set of numbers to a single value which can give us an impression of what the "average outcome" (or result quality is).

Measures of the Average

- Assume that we have obtained a sample $A = (a_0, a_1, \dots, a_{n-1})$ of n observations from an experiment, e.g., we have measured the qualities a_i of the best discovered solutions of $n = 101$ independent runs of an optimization algorithm.
- We usually want to reduce this set of numbers to a single value which can give us an impression of what the "average outcome" (or result quality is).
- Three of the most common options for doing so, for estimating the "center" of a distribution, are the arithmetic mean, the median, and the geometric mean.

Arithmetic Mean

Definition: Arithmetic Mean

The arithmetic mean $\mathrm{mean}(A)$ is an estimate of the expected value of a distribution from which a dataset was sampled.

Arithmetic Mean

Definition: Arithmetic Mean

The arithmetic mean $\mathrm{mean}(A)$ is an estimate of the expected value of a distribution from which a dataset was sampled. It is computed on data sample $A = (a_0, a_1, \ldots, a_{n-1})$ as the sum of all n elements a_i in the sample data A divided by the total number n of values.

Arithmetic Mean

Definition: Arithmetic Mean

The arithmetic mean $\text{mean}(A)$ is an estimate of the expected value of a distribution from which a dataset was sampled. It is computed on data sample $A = (a_0, a_1, \dots, a_{n-1})$ as the sum of all n elements a_i in the sample data A divided by the total number n of values.

$$\text{mean}(A) = \frac{1}{n} \sum_{i=0}^{n-1} a_i \tag{1}$$

Sample Median

Definition: Median

The median $\text{median}(A)$ is the value separating the bigger-valued half from the smaller-valued half of a data sample.

Sample Median

Definition: Median

The median $\text{median}(A)$ is the value separating the bigger-valued half from the smaller-valued half of a data sample. Its estimate is the value right in the middle of a *sorted* data sample $A = (a_0, a_1, \ldots, a_{n-1})$ where $a_{i-1} \leq a_i \ \forall i \in 1 \ldots (n-1)$.

Sample Median

Definition: Median

The median $\mathrm{median}(A)$ is the value separating the bigger-valued half from the smaller-valued half of a data sample. Its estimate is the value right in the middle of a *sorted* data sample $A = (a_0, a_1, \dots, a_{n-1})$ where $a_{i-1} \leq a_i \; \forall i \in 1 \dots (n-1)$.

$$\mathrm{median}(A) = \begin{cases} a_{\frac{n-1}{2}} & \text{if } n \text{ is odd} \\ \frac{1}{2}\left(a_{\frac{n}{2}-1} + a_{\frac{n}{2}}\right) & \text{otherwise} \end{cases} \quad \text{if } a_{i-1} \leq a_i \; \forall i \in 1 \dots (n-1) \qquad (2)$$

Outliers

- Sometimes the data contains outliers[26,43].

Outliers

- Sometimes the data contains outliers[26,43], i.e., observations which are much different from the other measurements.

Outliers

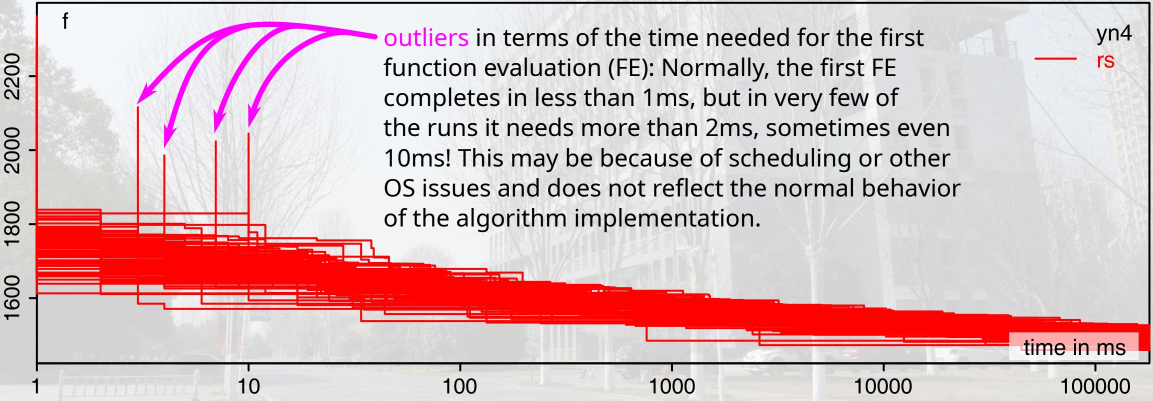
- Sometimes the data contains outliers[26,43], i.e., observations which are much different from the other measurements.
- They may represent measurement errors or observations which have been been disturbed by unusual effects.

Outliers

- Sometimes the data contains outliers[26,43], i.e., observations which are much different from the other measurements.
- They may represent measurement errors or observations which have been been disturbed by unusual effects.
- For example, maybe the operating system was updating itself during a run of one of our algorithms and, thus, took away some of the computation budget.

Outliers

- For example, maybe the operating system was updating itself during a run of one of our algorithms and, thus, took away some of the computation budget.
- In my experiments here, there are sometimes outliers in the time that it takes to create and evaluate the first candidate solution.



Outliers

- For example, maybe the operating system was updating itself during a run of one of our algorithms and, thus, took away some of the computation budget.
- In my experiments here, there are sometimes outliers in the time that it takes to create and evaluate the first candidate solution.
- But outliers are actually important. So I say this right now. I will also say it again later. But I am afraid that you may tune out during the following example. So remember: Outliers are important. Anyway. . .

Example for Data Samples w/o Outlier

- Two sets of data samples A and B with $n_a = n_b = 19$ values.

$$\begin{aligned} A &= (1, 3, 4, 4, 4, 5, 6, 6, 6, 6, 7, 7, 9, 9, 9, 10, 11, 12, 14) \\ B &= (1, 3, 4, 4, 4, 5, 6, 6, 6, 6, 7, 7, 9, 9, 9, 10, 11, 12, 10'008) \end{aligned}$$

Example for Data Samples w/o Outlier

- Two sets of data samples A and B with $n_a = n_b = 19$ values.

$$A = (1, 3, 4, 4, 4, 5, 6, 6, 6, 6, 7, 7, 9, 9, 9, 10, 11, 12, 14)$$
$$B = (1, 3, 4, 4, 4, 5, 6, 6, 6, 6, 7, 7, 9, 9, 9, 10, 11, 12, 10'008)$$

- We find that

Example for Data Samples w/o Outlier

- Two sets of data samples A and B with $n_a = n_b = 19$ values.

$$\begin{aligned} A &= (1, 3, 4, 4, 4, 5, 6, 6, 6, 6, 7, 7, 9, 9, 9, 10, 11, 12, 14) \\ B &= (1, 3, 4, 4, 4, 5, 6, 6, 6, 6, 7, 7, 9, 9, 9, 10, 11, 12, 10'008) \end{aligned}$$

- We find that
  - $\text{mean}(A) = \frac{1}{19}\sum_{i=0}^{18} a_i = \frac{133}{19} = 7$

Example for Data Samples w/o Outlier

- Two sets of data samples A and B with $n_a = n_b = 19$ values.

$$\begin{aligned} A &= (1, 3, 4, 4, 4, 5, 6, 6, 6, 6, 7, 7, 9, 9, 9, 10, 11, 12, 14) \\ B &= (1, 3, 4, 4, 4, 5, 6, 6, 6, 6, 7, 7, 9, 9, 9, 10, 11, 12, 10'008) \end{aligned}$$

- We find that
  - $\mathrm{mean}(A) = \frac{1}{19}\sum_{i=0}^{18} a_i = \frac{133}{19} = 7$ and
  - $\mathrm{mean}(B) = \frac{1}{19}\sum_{i=0}^{18} b_i = \frac{10'127}{19} = 553$

Example for Data Samples w/o Outlier

- Two sets of data samples A and B with $n_a = n_b = 19$ values.

$$\begin{aligned} A &= (1, 3, 4, 4, 4, 5, 6, 6, 6, 6, 7, 7, 9, 9, 9, 10, 11, 12, 14) \\ B &= (1, 3, 4, 4, 4, 5, 6, 6, 6, 6, 7, 7, 9, 9, 9, 10, 11, 12, 10'008) \end{aligned}$$

- We find that
  - $\text{mean}(A) = \frac{1}{19}\sum_{i=0}^{18} a_i = \frac{133}{19} = 7$ and
  - $\text{mean}(B) = \frac{1}{19}\sum_{i=0}^{18} b_i = \frac{10'127}{19} = 553$, while
  - $\text{median}(A) = a_9 = 6$

Example for Data Samples w/o Outlier

- Two sets of data samples A and B with $n_a = n_b = 19$ values.

$$\begin{aligned} A &= (1, 3, 4, 4, 4, 5, 6, 6, 6, 6, 7, 7, 9, 9, 9, 10, 11, 12, 14) \\ B &= (1, 3, 4, 4, 4, 5, 6, 6, 6, 6, 7, 7, 9, 9, 9, 10, 11, 12, 10'008) \end{aligned}$$

- We find that
  - $\mathrm{mean}(A) = \frac{1}{19}\sum_{i=0}^{18} a_i = \frac{133}{19} = 7$ and
  - $\mathrm{mean}(B) = \frac{1}{19}\sum_{i=0}^{18} b_i = \frac{10'127}{19} = 553$, while
  - $\mathrm{median}(A) = a_9 = 6$ and
  - $\mathrm{median}(B) = b_9 = 6$.

Example for Data Samples w/o Outlier

- Two sets of data samples A and B with $n_a = n_b = 19$ values.

$$\begin{aligned} A &= (1, 3, 4, 4, 4, 5, 6, 6, 6, 6, 7, 7, 9, 9, 9, 10, 11, 12, 14) \\ B &= (1, 3, 4, 4, 4, 5, 6, 6, 6, 6, 7, 7, 9, 9, 9, 10, 11, 12, 10'008) \end{aligned}$$

- We find that
  - $\mathrm{mean}(A) = \frac{1}{19}\sum_{i=0}^{18} a_i = \frac{133}{19} = 7$ and
  - $\mathrm{mean}(B) = \frac{1}{19}\sum_{i=0}^{18} b_i = \frac{10'127}{19} = 553$, while
  - $\mathrm{median}(A) = a_9 = 6$ and
  - $\mathrm{median}(B) = b_9 = 6$.
- The median is not affected by the outliers.

Example for Data Samples w/o Outlier

- Two sets of data samples A and B with $n_a = n_b = 19$ values.

$$\begin{aligned} A &= (1, 3, 4, 4, 4, 5, 6, 6, 6, 6, 7, 7, 9, 9, 9, 10, 11, 12, 14) \\ B &= (1, 3, 4, 4, 4, 5, 6, 6, 6, 6, 7, 7, 9, 9, 9, 10, 11, 12, 10'008) \end{aligned}$$

- We find that
  - $\text{mean}(A) = \frac{1}{19}\sum_{i=0}^{18} a_i = \frac{133}{19} = 7$ and
  - $\text{mean}(B) = \frac{1}{19}\sum_{i=0}^{18} b_i = \frac{10'127}{19} = 553$, while
  - $\text{median}(A) = a_9 = 6$ and
  - $\text{median}(B) = b_9 = 6$.
- The median is not affected by the outliers.
- $\text{mean}(B) = 553$ is a value completely different from anything that actually occurs in B...
  ...it gives us a completely wrong impression.

Outliers can be important!

- If you think about it, where could outliers in our experiments come from?

Outliers can be important!

- If you think about it, where could outliers in our experiments come from?
    1. The operating systems scheduling or other strange effects could mess with our timing.

Outliers can be important!

- If you think about it, where could outliers in our experiments come from?
  1. The operating systems scheduling or other strange effects could mess with our timing. This could cause worse results.

Outliers can be important!

- If you think about it, where could outliers in our experiments come from?
    1. The operating systems scheduling or other strange effects could mess with our timing. This could cause worse results. But usually this is already it.

Outliers can be important!

- If you think about it, where could outliers in our experiments come from?
  1. The operating systems scheduling or other strange effects could mess with our timing. This could cause worse results. But usually this is already it.
  2. Unless your objective function is noisy, e.g., if you measure some physical quantity, or the objective function involves randomized simulations, there are hardly any other "outside" effects that could mess up our results!

Outliers can be important!

- If you think about it, where could outliers in our experiments come from?
    1. The operating systems scheduling or other strange effects could mess with our timing. This could cause worse results. But usually this is already it.
    2. Unless your objective function is noisy, e.g., if you measure some physical quantity, or the objective function involves randomized simulations, there are hardly any other "outside" effects that could mess up our results!
- Instead, most likely there could be
    - bugs in our code!

Outliers can be important!

- If you think about it, where could outliers in our experiments come from?
  1. The operating systems scheduling or other strange effects could mess with our timing. This could cause worse results. But usually this is already it.
  2. Unless your objective function is noisy, e.g., if you measure some physical quantity, or the objective function involves randomized simulations, there are hardly any other "outside" effects that could mess up our results!
- Instead, most likely there could be
  - bugs in our code!
    - Bugs in our code are *the* most important number one reason for outliers!

Outliers can be important!

- If you think about it, where could outliers in our experiments come from?
    1. The operating systems scheduling or other strange effects could mess with our timing. This could cause worse results. But usually this is already it.
    2. Unless your objective function is noisy, e.g., if you measure some physical quantity, or the objective function involves randomized simulations, there are hardly any other "outside" effects that could mess up our results!
- Instead, most likely there could be
    - bugs in our code!
        - Bugs in our code are *the* most important number one reason for outliers!
        - Yes, also in your code! (Btw: Please use unit test[45,51,56].)

Outliers can be important!

- If you think about it, where could outliers in our experiments come from?
    1. The operating systems scheduling or other strange effects could mess with our timing. This could cause worse results. But usually this is already it.
    2. Unless your objective function is noisy, e.g., if you measure some physical quantity, or the objective function involves randomized simulations, there are hardly any other "outside" effects that could mess up our results!
- Instead, most likely there could be
    - bugs in our code!
        - Bugs in our code are *the* most important number one reason for outliers!
        - Yes, also in your code! (Btw: Please use unit test[45,51,56].)
    - Or: bad (but rare) worst-case behaviors of our algorithm!

Outliers can be important!

- If you think about it, where could outliers in our experiments come from?
  1. The operating systems scheduling or other strange effects could mess with our timing. This could cause worse results. But usually this is already it.
  2. Unless your objective function is noisy, e.g., if you measure some physical quantity, or the objective function involves randomized simulations, there are hardly any other "outside" effects that could mess up our results!
- Instead, most likely there could be
  - bugs in our code!
    - Bugs in our code are *the* most important number one reason for outliers!
    - Yes, also in your code! (Btw: Please use unit test[45,51,56].)
  - Or: bad (but rare) worst-case behaviors of our algorithm!

    Imagine that:

Outliers can be important!

- If you think about it, where could outliers in our experiments come from?
    1. The operating systems scheduling or other strange effects could mess with our timing. This could cause worse results. But usually this is already it.
    2. Unless your objective function is noisy, e.g., if you measure some physical quantity, or the objective function involves randomized simulations, there are hardly any other “outside” effects that could mess up our results!
- Instead, most likely there could be
    - bugs in our code!
        - Bugs in our code are *the* most important number one reason for outliers!
        - Yes, also in your code! (Btw: Please use unit test[45,51,56].)
    - Or: bad (but rare) worst-case behaviors of our algorithm!

      Imagine that: Your algorithm can actually solve the TSP or Maximum Satisfiability (MaxSAT) problem in polynomial time on 90% of all instances. . .

Outliers can be important!

- If you think about it, where could outliers in our experiments come from?
    1. The operating systems scheduling or other strange effects could mess with our timing. This could cause worse results. But usually this is already it.
    2. Unless your objective function is noisy, e.g., if you measure some physical quantity, or the objective function involves randomized simulations, there are hardly any other "outside" effects that could mess up our results!
- Instead, most likely there could be
    - bugs in our code!
        - Bugs in our code are *the* most important number one reason for outliers!
        - Yes, also in your code! (Btw: Please use unit test[45,51,56].)
    - Or: bad (but rare) worst-case behaviors of our algorithm!

      Imagine that: Your algorithm can actually solve the TSP or Maximum Satisfiability (MaxSAT) problem in polynomial time on 90% of all instances... ...but on 10%, it needs exponential time.

Outliers can be important!

- If you think about it, where could outliers in our experiments come from?
    1. The operating systems scheduling or other strange effects could mess with our timing. This could cause worse results. But usually this is already it.
    2. Unless your objective function is noisy, e.g., if you measure some physical quantity, or the objective function involves randomized simulations, there are hardly any other "outside" effects that could mess up our results!
- Instead, most likely there could be
    - bugs in our code!
        - Bugs in our code are *the* most important number one reason for outliers!
        - Yes, also in your code! (Btw: Please use unit test[45,51,56].)
    - Or: bad (but rare) worst-case behaviors of our algorithm!

      Imagine that: Your algorithm can actually solve the TSP or Maximum Satisfiability (MaxSAT) problem in polynomial time on 90% of all instances. . . . . . but on 10%, it needs exponential time. If you just look at the median runtime, you may think you discovered something awesome.

Outliers can be important!

- If you think about it, where could outliers in our experiments come from?
    1. The operating systems scheduling or other strange effects could mess with our timing. This could cause worse results. But usually this is already it.
    2. Unless your objective function is noisy, e.g., if you measure some physical quantity, or the objective function involves randomized simulations, there are hardly any other "outside" effects that could mess up our results!
- Instead, most likely there could be
    - bugs in our code!
        - Bugs in our code are *the* most important number one reason for outliers!
        - Yes, also in your code! (Btw: Please use unit test[45,51,56].)
    - Or: bad (but rare) worst-case behaviors of our algorithm!

      Imagine that: Your algorithm can actually solve the TSP or Maximum Satisfiability (MaxSAT) problem in polynomial time on 90% of all instances... ...but on 10%, it needs exponential time. If you just look at the median runtime, you may think you discovered something awesome. Actually, this is quite common...

Outliers can be important!

- If you t
  1. Th … ing. This co
  2. Ur … , or the ob … tside” eff
- Instead
  - bu
  - Or … ility I … , it (… you r … c

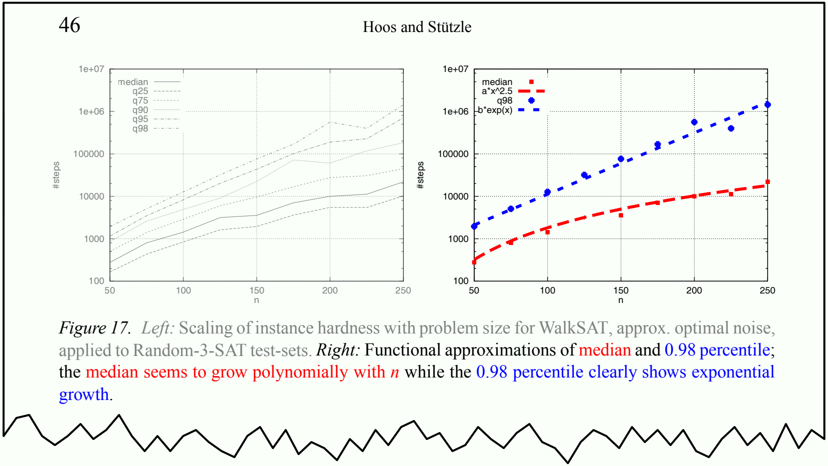

46 Hoos and Stützle

Figure 17. *Left:* Scaling of instance hardness with problem size for WalkSAT, approx. optimal noise, applied to Random-3-SAT test-sets. *Right:* Functional approximations of median and 0.98 percentile; the median seems to grow polynomially with *n* while the 0.98 percentile clearly shows exponential growth.

(Taken from the paper *“Local Search Algorithms for SAT: An Empirical Evaluation”* by Hoos and Stützle, coloring added manually[32].)

Outliers can be important!

- If you think about it, where could outliers in our experiments come from?
    1. The operating systems scheduling or other strange effects could mess with our timing. This could cause worse results. But usually this is already it.
    2. Unless your objective function is noisy, e.g., if you measure some physical quantity, or the objective function involves randomized simulations, there are hardly any other "outside" effects that could mess up our results!
- Instead, most likely there could be
    - bugs in our code!
        - Bugs in our code are *the* most important number one reason for outliers!
        - Yes, also in your code! (Btw: Please use unit test[45,51,56].)
    - Or: bad (but rare) worst-case behaviors of our algorithm!

      Imagine that: Your algorithm can actually solve the TSP or Maximum Satisfiability (MaxSAT) problem in polynomial time on 90% of all instances... ...but on 10%, it needs exponential time. If you just look at the median runtime, you may think you discovered something awesome. Actually, this is quite common...
- Thus, we may actually want that outliers influence our statistics...

Geometric Mean

OK, arithmetic mean, median . . . but what about the geometric mean?

Geometric Mean

OK, arithmetic mean, median ... but what about the geometric mean?

Definition: Geometric Mean

The geometric mean $\operatorname{geom}(A)$ is the n^{th} root of the product of n positive values.

Geometric Mean

OK, arithmetic mean, median ... but what about the geometric mean?

Definition: Geometric Mean

The geometric mean $\mathrm{geom}(A)$ is the n^{th} root of the product of n positive values.

$$\mathrm{geom}(A) = \sqrt[n]{\prod_{i=0}^{n-1} a_i} \tag{3}$$

$$\tag{4}$$

Geometric Mean

OK, arithmetic mean, median ... but what about the geometric mean?

Definition: Geometric Mean

The geometric mean $\mathrm{geom}(A)$ is the n^{th} root of the product of n positive values.

$$\mathrm{geom}(A) = \sqrt[n]{\prod_{i=0}^{n-1} a_i} \quad (3)$$

$$\mathrm{geom}(A) = \exp\left(\frac{1}{n}\sum_{i=0}^{n-1} \log a_i\right) \quad (4)$$

Normalized Data

- Often, our data is somehow normalized.

Normalized Data

- Often, our data is somehow normalized.
- Let's say we solve the problem instances I_1 to I_3 with the different algorithms A_1 to A_3.

Normalized Data

- Often, our data is somehow normalized.
- Let's say we solve the problem instances I_1 to I_3 with the different algorithms A_1 to A_3.
- We measure the required runtimes as follows:

Normalized Data

- Often, our data is somehow normalized.
- Let's say we solve the problem instances I_1 to I_3 with the different algorithms A_1 to A_3.
- We measure the required runtimes as follows:

	A_1	A_2	A_3
I_1			
I_2			
I_3			

Normalized Data

- Often, our data is somehow normalized.
- Let's say we solve the problem instances I_1 to I_3 with the different algorithms A_1 to A_3.
- We measure the required runtimes as follows:

	A_1	A_2	A_3
I_1	10 s		
I_2			
I_3			

Normalized Data

- Often, our data is somehow normalized.
- Let's say we solve the problem instances I_1 to I_3 with the different algorithms A_1 to A_3.
- We measure the required runtimes as follows:

	A_1	A_2	A_3
I_1	10 s		
I_2	20 s		
I_3			

Normalized Data

- Often, our data is somehow normalized.
- Let's say we solve the problem instances I_1 to I_3 with the different algorithms A_1 to A_3.
- We measure the required runtimes as follows:

	A_1	A_2	A_3
I_1	10 s		
I_2	20 s		
I_3	40 s		

Normalized Data

- Often, our data is somehow normalized.
- Let's say we solve the problem instances I_1 to I_3 with the different algorithms A_1 to A_3.
- We measure the required runtimes as follows:

	A_1	A_2	A_3
I_1	10 s	20 s	
I_2	20 s		
I_3	40 s		

Normalized Data

- Often, our data is somehow normalized.
- Let's say we solve the problem instances I_1 to I_3 with the different algorithms A_1 to A_3.
- We measure the required runtimes as follows:

	A_1	A_2	A_3
I_1	10 s	20 s	
I_2	20 s	40 s	
I_3	40 s		

Normalized Data

- Often, our data is somehow normalized.
- Let's say we solve the problem instances I_1 to I_3 with the different algorithms A_1 to A_3.
- We measure the required runtimes as follows:

	A_1	A_2	A_3
I_1	10 s	20 s	
I_2	20 s	40 s	
I_3	40 s	10 s	

Normalized Data

- Often, our data is somehow normalized.
- Let's say we solve the problem instances I_1 to I_3 with the different algorithms A_1 to A_3.
- We measure the required runtimes as follows:

	A_1	A_2	A_3
I_1	10 s	20 s	40 s
I_2	20 s	40 s	
I_3	40 s	10 s	

Normalized Data

- Often, our data is somehow normalized.
- Let's say we solve the problem instances I_1 to I_3 with the different algorithms A_1 to A_3.
- We measure the required runtimes as follows:

	A_1	A_2	A_3
I_1	10 s	20 s	40 s
I_2	20 s	40 s	10 s
I_3	40 s	10 s	

Normalized Data

- Often, our data is somehow normalized.
- Let's say we solve the problem instances I_1 to I_3 with the different algorithms A_1 to A_3.
- We measure the required runtimes as follows:

	A_1	A_2	A_3
I_1	10 s	20 s	40 s
I_2	20 s	40 s	10 s
I_3	40 s	10 s	20 s

Normalized Data

- Often, our data is somehow normalized.
- We measure the required runtimes as follows:
- The arithmetic mean values are the same.

	A_1	A_2	A_3
I_1	10 s	20 s	40 s
I_2	20 s	40 s	10 s
I_3	40 s	10 s	20 s
mean:	23.33 s	23.33 s	23.33 s

Normalized Data

- Often, our data is somehow normalized.
- The arithmetic mean and the median values are the same.

	A_1	A_2	A_3
I_1	10 s	20 s	40 s
I_2	20 s	40 s	10 s
I_3	40 s	10 s	20 s
mean:	23.33 s	23.33 s	23.33 s
median:	20.00 s	20.00 s	20.00 s

Normalized Data

- Often, our data is somehow normalized.
- The arithmetic mean, the median, and the geometric mean values are the same.

	A_1	A_2	A_3
I_1	10 s	20 s	40 s
I_2	20 s	40 s	10 s
I_3	40 s	10 s	20 s
mean:	23.33 s	23.33 s	23.33 s
median:	20.00 s	20.00 s	20.00 s
geom:	20.00 s	20.00 s	20.00 s

Normalized Data

- Often, our data is somehow normalized.
- The arithmetic mean, the median, and the geometric mean values are the same.
- We can conclude that the three algorithms offer the same performance in average over these benchmark instances.

	A_1	A_2	A_3
I_1	10 s	20 s	40 s
I_2	20 s	40 s	10 s
I_3	40 s	10 s	20 s
mean:	23.33 s	23.33 s	23.33 s
median:	20.00 s	20.00 s	20.00 s
geom:	20.00 s	20.00 s	20.00 s

Normalized Data

- Often, our data is somehow normalized.
- We can conclude that the three algorithms offer the same performance in average over these benchmark instances.
- But often the measured numbers "look messier" and are harder to compare at first glance.

	A_1	A_2	A_3
I_1	10 s	20 s	40 s
I_2	20 s	40 s	10 s
I_3	40 s	10 s	20 s
mean:	23.33 s	23.33 s	23.33 s
median:	20.00 s	20.00 s	20.00 s
geom:	20.00 s	20.00 s	20.00 s

Normalized Data

- Often, our data is somehow normalized.
- But often the measured numbers "look messier" and are harder to compare at first glance.
- So often we want to normalize them by picking one algorithm as "standard" and dividing them by its measurements.

	A_1	A_2	A_3
I_1	10 s	20 s	40 s
I_2	20 s	40 s	10 s
I_3	40 s	10 s	20 s
mean:	23.33 s	23.33 s	23.33 s
median:	20.00 s	20.00 s	20.00 s
geom:	20.00 s	20.00 s	20.00 s

Normalized Data

- Often, our data is somehow normalized.
- But often the measured numbers "look messier" and are harder to compare at first glance.
- So often we want to normalize them by picking one algorithm as "standard" and dividing them by its measurements.
- Let's say A_1 was a well-known heuristic, maybe we even took its results from a paper, and we want to use it as baseline for comparison and normalize our data by it.

	A_1	A_2	A_3
I_1	10 s	20 s	40 s
I_2	20 s	40 s	10 s
I_3	40 s	10 s	20 s
mean:	23.33 s	23.33 s	23.33 s
median:	20.00 s	20.00 s	20.00 s
geom:	20.00 s	20.00 s	20.00 s

Normalized Data

- Often, our data is somehow normalized.
- So often we want to normalize them by picking one algorithm as "standard" and dividing them by its measurements.
- Let's say A_1 was a well-known heuristic, maybe we even took its results from a paper, and we want to use it as baseline for comparison and normalize our data by it.

	A_1	A_2	A_3
I_1	10 s	20 s	40 s
I_2	20 s	40 s	10 s
I_3	40 s	10 s	20 s
mean:	23.33 s	23.33 s	23.33 s
median:	20.00 s	20.00 s	20.00 s
geom:	20.00 s	20.00 s	20.00 s

Normalized Data

- Often, our data is somehow normalized.
- Let's say A_1 was a well-known heuristic, maybe we even took its results from a paper, and we want to use it as baseline for comparison and normalize our data by it.
- OK, so we get this table with normalized values, which allow us to make sense of the data at first glance.

	A_1	A_2	A_3
I_1	10 s	20 s	40 s
I_2	20 s	40 s	10 s
I_3	40 s	10 s	20 s
mean:	23.33 s	23.33 s	23.33 s
median:	20.00 s	20.00 s	20.00 s
geom:	20.00 s	20.00 s	20.00 s

	A_1	A_2	A_3
I_1	1.00	2.00	4.00
I_2	1.00	2.00	0.50
I_3	1.00	0.25	0.50

Normalized Data

- Often, our data is somehow normalized.
- OK, so we get this table with normalized values, which allow us to make sense of the data at first glance.
- If we now compute the arithmetic mean

	A_1	A_2	A_3
I_1	10 s	20 s	40 s
I_2	20 s	40 s	10 s
I_3	40 s	10 s	20 s
mean:	23.33 s	23.33 s	23.33 s
median:	20.00 s	20.00 s	20.00 s
geom:	20.00 s	20.00 s	20.00 s

	A_1	A_2	A_3
I_1	1.00	2.00	4.00
I_2	1.00	2.00	0.50
I_3	1.00	0.25	0.50
mean:	1.00	1.42	1.67

Normalized Data

- Often, our data is somehow normalized.
- OK, so we get this table with normalized values, which allow us to make sense of the data at first glance.
- If we now compute the arithmetic mean, then A_1 is best

	A_1	A_2	A_3
I_1	10 s	20 s	40 s
I_2	20 s	40 s	10 s
I_3	40 s	10 s	20 s
mean:	23.33 s	23.33 s	23.33 s
median:	20.00 s	20.00 s	20.00 s
geom:	20.00 s	20.00 s	20.00 s

	A_1	A_2	A_3
I_1	1.00	2.00	4.00
I_2	1.00	2.00	0.50
I_3	1.00	0.25	0.50
mean:	1.00	1.42	1.67

Normalized Data

- Often, our data is somehow normalized.
- OK, so we get this table with normalized values, which allow us to make sense of the data at first glance.
- If we now compute the arithmetic mean, then A_1 is best and A_3 looks worst.

	A_1	A_2	A_3
I_1	10 s	20 s	40 s
I_2	20 s	40 s	10 s
I_3	40 s	10 s	20 s
mean:	23.33 s	23.33 s	23.33 s
median:	20.00 s	20.00 s	20.00 s
geom:	20.00 s	20.00 s	20.00 s

	A_1	A_2	A_3
I_1	1.00	2.00	4.00
I_2	1.00	2.00	0.50
I_3	1.00	0.25	0.50
mean:	1.00	1.42	1.67

Normalized Data

- Often, our data is somehow normalized.
- OK, so we get this table with normalized values, which allow us to make sense of the data at first glance.
- If we now compute the arithmetic mean, then A_1 is best and A_3 looks worst.
- According to the median

	A_1	A_2	A_3
I_1	10 s	20 s	40 s
I_2	20 s	40 s	10 s
I_3	40 s	10 s	20 s
mean:	23.33 s	23.33 s	23.33 s
median:	20.00 s	20.00 s	20.00 s
geom:	20.00 s	20.00 s	20.00 s

	A_1	A_2	A_3
I_1	1.00	2.00	4.00
I_2	1.00	2.00	0.50
I_3	1.00	0.25	0.50
mean:	1.00	1.42	1.67
median:	1.00	2.00	0.50

Normalized Data

- Often, our data is somehow normalized.
- If we now compute the arithmetic mean, then A_1 is best and A_3 looks worst.
- According to the median, A_3 is best

	A_1	A_2	A_3
I_1	10 s	20 s	40 s
I_2	20 s	40 s	10 s
I_3	40 s	10 s	20 s
mean:	23.33 s	23.33 s	23.33 s
median:	20.00 s	20.00 s	20.00 s
geom:	20.00 s	20.00 s	20.00 s

	A_1	A_2	A_3
I_1	1.00	2.00	4.00
I_2	1.00	2.00	0.50
I_3	1.00	0.25	0.50
mean:	1.00	1.42	1.67
median:	1.00	2.00	0.50

Normalized Data

- Often, our data is somehow normalized.
- If we now compute the arithmetic mean, then A_1 is best and A_3 looks worst.
- According to the median, A_3 is best and A_2 is worst!

	A_1	A_2	A_3
I_1	10 s	20 s	40 s
I_2	20 s	40 s	10 s
I_3	40 s	10 s	20 s
mean:	23.33 s	23.33 s	23.33 s
median:	20.00 s	20.00 s	20.00 s
geom:	20.00 s	20.00 s	20.00 s

	A_1	A_2	A_3
I_1	1.00	2.00	4.00
I_2	1.00	2.00	0.50
I_3	1.00	0.25	0.50
mean:	1.00	1.42	1.67
median:	1.00	2.00	0.50

Normalized Data

- Often, our data is somehow normalized.
- If we now compute the arithmetic mean, then A_1 is best and A_3 looks worst.
- According to the median, A_3 is best and A_2 is worst!
- Only the geometric mean still indicates that the algorithms perform the same...

	A_1	A_2	A_3
I_1	10 s	20 s	40 s
I_2	20 s	40 s	10 s
I_3	40 s	10 s	20 s
mean:	23.33 s	23.33 s	23.33 s
median:	20.00 s	20.00 s	20.00 s
geom:	20.00 s	20.00 s	20.00 s

	A_1	A_2	A_3
I_1	1.00	2.00	4.00
I_2	1.00	2.00	0.50
I_3	1.00	0.25	0.50
mean:	1.00	1.42	1.67
median:	1.00	2.00	0.50
geom:	1.00	1.00	1.00

Normalized Data

- Often, our data is somehow normalized.
- Hm.

	A_1	A_2	A_3
I_1	10 s	20 s	40 s
I_2	20 s	40 s	10 s
I_3	40 s	10 s	20 s
mean:	23.33 s	23.33 s	23.33 s
median:	20.00 s	20.00 s	20.00 s
geom:	20.00 s	20.00 s	20.00 s

	A_1	A_2	A_3
I_1	1.00	2.00	4.00
I_2	1.00	2.00	0.50
I_3	1.00	0.25	0.50
mean:	1.00	1.42	1.67
median:	1.00	2.00	0.50
geom:	1.00	1.00	1.00

Normalized Data

- Often, our data is somehow normalized.
- Hm. OK, then let's normalize using the results of A_2 instead.

	A_1	A_2	A_3
I_1	10 s	20 s	40 s
I_2	20 s	40 s	10 s
I_3	40 s	10 s	20 s
mean:	23.33 s	23.33 s	23.33 s
median:	20.00 s	20.00 s	20.00 s
geom:	20.00 s	20.00 s	20.00 s

Normalized Data

- Often, our data is somehow normalized.
- Hm. OK, then let's normalize using the results of A_2 instead.
- OK, so we get this table with normalized values.

	A_1	A_2	A_3
I_1	10 s	20 s	40 s
I_2	20 s	40 s	10 s
I_3	40 s	10 s	20 s
mean:	23.33 s	23.33 s	23.33 s
median:	20.00 s	20.00 s	20.00 s
geom:	20.00 s	20.00 s	20.00 s

	A_1	A_2	A_3
I_1	0.50	1.00	2.00
I_2	0.50	1.00	0.25
I_3	4.00	1.00	2.00

Normalized Data

- Often, our data is somehow normalized.
- OK, so we get this table with normalized values.
- If we now compute the arithmetic mean

	A_1	A_2	A_3
I_1	10 s	20 s	40 s
I_2	20 s	40 s	10 s
I_3	40 s	10 s	20 s
mean:	23.33 s	23.33 s	23.33 s
median:	20.00 s	20.00 s	20.00 s
geom:	20.00 s	20.00 s	20.00 s

	A_1	A_2	A_3
I_1	0.50	1.00	2.00
I_2	0.50	1.00	0.25
I_3	4.00	1.00	2.00
mean:	1.67	1.00	1.42

Normalized Data

- Often, our data is somehow normalized.
- OK, so we get this table with normalized values.
- If we now compute the arithmetic mean, then A_2 is best

	A_1	A_2	A_3
I_1	10 s	20 s	40 s
I_2	20 s	40 s	10 s
I_3	40 s	10 s	20 s
mean:	23.33 s	23.33 s	23.33 s
median:	20.00 s	20.00 s	20.00 s
geom:	20.00 s	20.00 s	20.00 s

	A_1	A_2	A_3
I_1	0.50	1.00	2.00
I_2	0.50	1.00	0.25
I_3	4.00	1.00	2.00
mean:	1.67	1.00	1.42

Normalized Data

- Often, our data is somehow normalized.
- OK, so we get this table with normalized values.
- If we now compute the arithmetic mean, then A_2 is best and A_1 looks worst.

	A_1	A_2	A_3
I_1	10 s	20 s	40 s
I_2	20 s	40 s	10 s
I_3	40 s	10 s	20 s
mean:	23.33 s	23.33 s	23.33 s
median:	20.00 s	20.00 s	20.00 s
geom:	20.00 s	20.00 s	20.00 s

	A_1	A_2	A_3
I_1	0.50	1.00	2.00
I_2	0.50	1.00	0.25
I_3	4.00	1.00	2.00
mean:	1.67	1.00	1.42

Normalized Data

- Often, our data is somehow normalized.
- OK, so we get this table with normalized values.
- If we now compute the arithmetic mean, then A_2 is best and A_1 looks worst.
- According to the median

	A_1	A_2	A_3
I_1	10 s	20 s	40 s
I_2	20 s	40 s	10 s
I_3	40 s	10 s	20 s
mean:	23.33 s	23.33 s	23.33 s
median:	20.00 s	20.00 s	20.00 s
geom:	20.00 s	20.00 s	20.00 s

	A_1	A_2	A_3
I_1	0.50	1.00	2.00
I_2	0.50	1.00	0.25
I_3	4.00	1.00	2.00
mean:	1.67	1.00	1.42
median:	0.50	1.00	2.00

Normalized Data

- Often, our data is somehow normalized.
- If we now compute the arithmetic mean, then A_2 is best and A_1 looks worst.
- According to the median, A_1 is best

	A_1	A_2	A_3
I_1	10 s	20 s	40 s
I_2	20 s	40 s	10 s
I_3	40 s	10 s	20 s
mean:	23.33 s	23.33 s	23.33 s
median:	20.00 s	20.00 s	20.00 s
geom:	20.00 s	20.00 s	20.00 s

	A_1	A_2	A_3
I_1	0.50	1.00	2.00
I_2	0.50	1.00	0.25
I_3	4.00	1.00	2.00
mean:	1.67	1.00	1.42
median:	0.50	1.00	2.00

Normalized Data

- Often, our data is somehow normalized.
- If we now compute the arithmetic mean, then A_2 is best and A_1 looks worst.
- According to the median, A_1 is best and A_3 is worst!

	A_1	A_2	A_3
I_1	10 s	20 s	40 s
I_2	20 s	40 s	10 s
I_3	40 s	10 s	20 s
mean:	23.33 s	23.33 s	23.33 s
median:	20.00 s	20.00 s	20.00 s
geom:	20.00 s	20.00 s	20.00 s

	A_1	A_2	A_3
I_1	0.50	1.00	2.00
I_2	0.50	1.00	0.25
I_3	4.00	1.00	2.00
mean:	1.67	1.00	1.42
median:	0.50	1.00	2.00

Normalized Data

- Often, our data is somehow normalized.
- If we now compute the arithmetic mean, then A_2 is best and A_1 looks worst.
- According to the median, A_1 is best and A_3 is worst!
- Only the geometric mean still indicates that the algorithms perform the same...

	A_1	A_2	A_3
I_1	10 s	20 s	40 s
I_2	20 s	40 s	10 s
I_3	40 s	10 s	20 s
mean:	23.33 s	23.33 s	23.33 s
median:	20.00 s	20.00 s	20.00 s
geom:	20.00 s	20.00 s	20.00 s

	A_1	A_2	A_3
I_1	0.50	1.00	2.00
I_2	0.50	1.00	0.25
I_3	4.00	1.00	2.00
mean:	1.67	1.00	1.42
median:	0.50	1.00	2.00
geom:	1.00	1.00	1.00

Normalized Data

- Often, our data is somehow normalized.
- Only the geometric mean still indicates that the algorithms perform the same. . .
- The geometric mean is the only meaningful average if we have normalized data![24]

Normalized Data

- Often, our data is somehow normalized.
- The geometric mean is the only meaningful average if we have normalized data![24]
- And we very often have normalized data.

Normalized Data

- Often, our data is somehow normalized.
- The geometric mean is the only meaningful average if we have normalized data![24]
- And we very often have normalized data.
- For example, at least half of the papers on the Job Shop Scheduling Problem (JSSP) normalize the result qualities they obtain on benchmark instances with the Best Known Solutions (BKSes).

Normalized Data

- Often, our data is somehow normalized.
- The geometric mean is the only meaningful average if we have normalized data![24]
- And we very often have normalized data.
- For example, at least half of the papers on the Job Shop Scheduling Problem (JSSP) normalize the result qualities they obtain on benchmark instances with the Best Known Solutions (BKSes) and then compute the arithmetic mean.

Arithmetic Mean vs. Median vs. Geometric Mean

- Most publications report arithmetic mean results, many report median results, almost none report geometric means.

Arithmetic Mean vs. Median vs. Geometric Mean

- Most publications report arithmetic mean results, many report median results, almost none report geometric means.
- The median is more robust against outliers compared to the arithmetic mean.

Arithmetic Mean vs. Median vs. Geometric Mean

- Most publications report arithmetic mean results, many report median results, almost none report geometric means.
- The median is more robust against outliers compared to the arithmetic mean, however, in normal application scenarios, there are very few acceptable reasons for outliers.

Arithmetic Mean vs. Median vs. Geometric Mean

- Most publications report arithmetic mean results, many report median results, almost none report geometric means.
- The median is more robust against outliers compared to the arithmetic mean, however, in normal application scenarios, there are very few acceptable reasons for outliers.
- We therefore want to know both the arithmetic mean and the median.

Arithmetic Mean vs. Median vs. Geometric Mean

- Most publications report arithmetic mean results, many report median results, almost none report geometric means.
- The median is more robust against outliers compared to the arithmetic mean, however, in normal application scenarios, there are very few acceptable reasons for outliers.
- We therefore want to know both the arithmetic mean and the median:
  - If the arithmetic mean is much worse than the median, then

Arithmetic Mean vs. Median vs. Geometric Mean

- Most publications report arithmetic mean results, many report median results, almost none report geometric means.
- The median is more robust against outliers compared to the arithmetic mean, however, in normal application scenarios, there are very few acceptable reasons for outliers.
- We therefore want to know both the arithmetic mean and the median:
  - If the arithmetic mean is much worse than the median, then
    - maybe we have a bug in our code that only sometimes has an impact.

Arithmetic Mean vs. Median vs. Geometric Mean

- Most publications report arithmetic mean results, many report median results, almost none report geometric means.
- The median is more robust against outliers compared to the arithmetic mean, however, in normal application scenarios, there are very few acceptable reasons for outliers.
- We therefore want to know both the arithmetic mean and the median:
  - If the arithmetic mean is much worse than the median, then
    - maybe we have a bug in our code that only sometimes has an impact or
    - our algorithm has a bad worst-case behavior (which is also good to know).

Arithmetic Mean vs. Median vs. Geometric Mean

- Most publications report arithmetic mean results, many report median results, almost none report geometric means.
- The median is more robust against outliers compared to the arithmetic mean, however, in normal application scenarios, there are very few acceptable reasons for outliers.
- We therefore want to know both the arithmetic mean and the median:
  - If the arithmetic mean is much worse than the median, then
    - maybe we have a bug in our code that only sometimes has an impact or
    - our algorithm has a bad worst-case behavior (which is also good to know).
  - If the median is much worse than the mean, then the mean is too optimistic, i.e., most of the time we should expect worse results.

Arithmetic Mean vs. Median vs. Geometric Mean

- Most publications report arithmetic mean results, many report median results, almost none report geometric means.
- The median is more robust against outliers compared to the arithmetic mean, however, in normal application scenarios, there are very few acceptable reasons for outliers.
- We therefore want to know both the arithmetic mean and the median:
  - If the arithmetic mean is much worse than the median, then
    - maybe we have a bug in our code that only sometimes has an impact or
    - our algorithm has a bad worst-case behavior (which is also good to know).
  - If the median is much worse than the mean, then the mean is too optimistic, i.e., most of the time we should expect worse results.
- If there are outliers, the value of the arithmetic mean itself may be very different from any actually observed value, while the median is (almost always) similar to some actual measurements.

Arithmetic Mean vs. Median vs. Geometric Mean

- Most publications report arithmetic mean results, many report median results, almost none report geometric means.
- The median is more robust against outliers compared to the arithmetic mean, however, in normal application scenarios, there are very few acceptable reasons for outliers.
- We therefore want to know both the arithmetic mean and the median.

Arithmetic Mean vs. Median vs. Geometric Mean

- Most publications report arithmetic mean results, many report median results, almost none report geometric means.
- The median is more robust against outliers compared to the arithmetic mean, however, in normal application scenarios, there are very few acceptable reasons for outliers.
- We therefore want to know both the arithmetic mean and the median.
- Often, our data is implicitly or explicitly normalized.

Arithmetic Mean vs. Median vs. Geometric Mean

- Most publications report arithmetic mean results, many report median results, almost none report geometric means.
- The median is more robust against outliers compared to the arithmetic mean, however, in normal application scenarios, there are very few acceptable reasons for outliers.
- We therefore want to know both the arithmetic mean and the median.
- Often, our data is implicitly or explicitly normalized, e.g.,
  - if we divide result qualities by results of well-known heuristics or BKSes or

Arithmetic Mean vs. Median vs. Geometric Mean

- Most publications report arithmetic mean results, many report median results, almost none report geometric means.
- The median is more robust against outliers compared to the arithmetic mean, however, in normal application scenarios, there are very few acceptable reasons for outliers.
- We therefore want to know both the arithmetic mean and the median.
- Often, our data is implicitly or explicitly normalized, e.g.,
  - if we divide result qualities by results of well-known heuristics or BKSes or
  - if we normalize the runtime using another algorithm as standard.

Arithmetic Mean vs. Median vs. Geometric Mean

- Most publications report arithmetic mean results, many report median results, almost none report geometric means.
- The median is more robust against outliers compared to the arithmetic mean, however, in normal application scenarios, there are very few acceptable reasons for outliers.
- We therefore want to know both the arithmetic mean and the median.
- Often, our data is implicitly or explicitly normalized, e.g.,
  - if we divide result qualities by results of well-known heuristics or BKSes or
  - if we normalize the runtime using another algorithm as standard.
- Then, the arithmetic mean and median can be very misleading and the geometric mean must be computed.

Arithmetic Mean vs. Median vs. Geometric Mean

- Most publications report arithmetic mean results, many report median results, almost none report geometric means.
- The median is more robust against outliers compared to the arithmetic mean, however, in normal application scenarios, there are very few acceptable reasons for outliers.
- We therefore want to know both the arithmetic mean and the median.
- Often, our data is implicitly or explicitly normalized, e.g.,
  - if we divide result qualities by results of well-known heuristics or BKSes or
  - if we normalize the runtime using another algorithm as standard.
- Then, the arithmetic mean and median can be very misleading and the geometric mean must be computed.
- I think: On raw data, compute all three measures of average, and pay special attention to the one looking the worst.

Arithmetic Mean vs. Median vs. Geometric Mean

- Most publications report arithmetic mean results, many report median results, almost none report geometric means.
- The median is more robust against outliers compared to the arithmetic mean, however, in normal application scenarios, there are very few acceptable reasons for outliers.
- We therefore want to know both the arithmetic mean and the median.
- Often, our data is implicitly or explicitly normalized, e.g.,
  - if we divide result qualities by results of well-known heuristics or BKSes or
  - if we normalize the runtime using another algorithm as standard.
- Then, the arithmetic mean and median can be very misleading and the geometric mean must be computed.
- I think: On raw data, compute all three measures of average, and pay special attention to the one looking the worst. On normalized data, compute the geometric mean.

Arithmetic Mean vs. Median vs. Geometric Mean

- Most publications report arithmetic mean results, many report median results, almost none report geometric means.
- The median is more robust against outliers compared to the arithmetic mean, however, in normal application scenarios, there are very few acceptable reasons for outliers.
- We therefore want to know both the arithmetic mean and the median.
- Often, our data is implicitly or explicitly normalized, e.g.,
  - if we divide result qualities by results of well-known heuristics or BKSes or
  - if we normalize the runtime using another algorithm as standard.
- Then, the arithmetic mean and median can be very misleading and the geometric mean must be computed.
- I think: On raw data, compute all three measures of average, and pay special attention to the one looking the worst. On normalized data, compute the geometric mean, but also consider the arithmetic mean and median *if and only if they make **your** algorithm look worse.*

Measures of the Spread

- The average gives us a good impression about the central value or location of a distribution.

Measures of the Spread

- The average gives us a good impression about the central value or location of a distribution.
- It does not tell us much about the range of the data.

Measures of the Spread

- The average gives us a good impression about the central value or location of a distribution.
- It does not tell us much about the range of the data.
- We do not know whether the data we have measured is very similar to the median or whether it may differ very much from the mean.

Measures of the Spread

- The average gives us a good impression about the central value or location of a distribution.
- It does not tell us much about the range of the data.
- We do not know whether the data we have measured is very similar to the median or whether it may differ very much from the mean.
- An average alone is not very meaningful – if we known nothing about the range of the data.

Measures of the Spread

- The average gives us a good impression about the central value or location of a distribution.
- It does not tell us much about the range of the data.
- We do not know whether the data we have measured is very similar to the median or whether it may differ very much from the mean.
- An average alone is not very meaningful – if we known nothing about the range of the data.
- We can therefore compute a measure of dispersion, i.e., a value that tells us whether the observations are stretched and spread far or squeezed tight around the center.

Sample Variance

Definition: Variance

The variance of a distribution is the expectation of the squared deviation of the underlying random variable from its expected value.

Sample Variance

Definition: Variance

The variance of a distribution is the expectation of the squared deviation of the underlying random variable from its expected value.

Definition: Sample Variance

The variance $\text{var}(A)$ of a data sample $A = (a_0, a_1, \ldots, a_{n-1})$ with n observations can be estimated as:

$$\text{var}(A) = \frac{1}{n-1}\sum_{i=0}^{n-1}(a_i - \text{mean}(A))^2 = \frac{1}{n-1}\left[\left(\sum_{i=0}^{n-1} a_i^2\right) - \frac{1}{n}\left(\sum_{i=0}^{n-1} a_i\right)^2\right]$$

Standard Deviation

Definition: Sample Standard Deviation

The standard deviation $\mathrm{sd}(A)$ of a data sample $A = (a_0, a_1, \ldots, a_{n-1})$ with n observations is the square root of the estimated variance $\mathrm{var}(A)$.

$$\mathrm{sd}(A) = \sqrt{\mathrm{var}(A)}$$

Standard Deviation

- Small standard deviations indicate that the observations tend to be similar to the mean.

Standard Deviation

- Small standard deviations indicate that the observations tend to be similar to the mean.
- Large standard deviations indicate that they tend to be far from the mean.

Standard Deviation

- Small standard deviations indicate that the observations tend to be similar to the mean.
- Large standard deviations indicate that they tend to be far from the mean.
- Small standard deviations in optimization results and runtime indicate that the algorithm is reliable.

Standard Deviation

- Small standard deviations indicate that the observations tend to be similar to the mean.
- Large standard deviations indicate that they tend to be far from the mean.
- Small standard deviations in optimization results and runtime indicate that the algorithm is reliable.
- Large standard deviations indicate unreliable algorithms.

Standard Deviation

- Small standard deviations indicate that the observations tend to be similar to the mean.
- Large standard deviations indicate that they tend to be far from the mean.
- Small standard deviations in optimization results and runtime indicate that the algorithm is reliable.
- Large standard deviations indicate unreliable algorithms, but may also offer a potential that could be exploited.

Standard Deviation

- Small standard deviations indicate that the observations tend to be similar to the mean.
- Large standard deviations indicate that they tend to be far from the mean.
- Small standard deviations in optimization results and runtime indicate that the algorithm is reliable.
- Large standard deviations indicate unreliable algorithms, but may also offer a potential that could be exploited: Given enough time, we can restart algorithms several times and expect to get different (and thus sometimes better) solutions.

Quantiles

Definition: Sample Quantile

The q-quantiles are the cut points that divide a sorted data sample $A = (a_0, a_1, \dots, a_{n-1})$ where $a_{i-1} \leq a_i \; \forall i \in 1 \dots (n-1)$ into q equally-sized parts.

Quantiles

Definition: Sample Quantile

The q-quantiles are the cut points that divide a sorted data sample $A = (a_0, a_1, \ldots, a_{n-1})$ where $a_{i-1} \leq a_i \; \forall i \in 1 \ldots (n-1)$ into q equally-sized parts. $\mathrm{quantile}_q^k(A)$ be the k^{th} q-quantile, with $k \in 1 \ldots (q-1)$, i.e., there are $q-1$ of the q-quantiles.

$$
\begin{aligned}
h &= (n-1)\tfrac{k}{q} \\
\mathrm{quantile}_q^k(A) &= \begin{cases} a_h & \text{if } h \text{ is integer} \\ a_{\lfloor h \rfloor} + (h - \lfloor h \rfloor) * \left(a_{\lfloor h \rfloor + 1} - a_{\lfloor h \rfloor}\right) & \text{otherwise} \end{cases}
\end{aligned}
$$

Quantiles

Definition: Sample Quantile

The q-quantiles are the cut points that divide a sorted data sample $A = (a_0, a_1, \ldots, a_{n-1})$ where $a_{i-1} \leq a_i \; \forall i \in 1 \ldots (n-1)$ into q equally-sized parts. $\text{quantile}_q^k(A)$ be the k^{th} q-quantile, with $k \in 1 \ldots (q-1)$, i.e., there are $q-1$ of the q-quantiles.

$$
\begin{aligned}
h &= (n-1)\frac{k}{q} \\
\text{quantile}_q^k(A) &= \begin{cases} a_h & \text{if } h \text{ is integer} \\ a_{\lfloor h \rfloor} + (h - \lfloor h \rfloor) * \left(a_{\lfloor h \rfloor + 1} - a_{\lfloor h \rfloor}\right) & \text{otherwise} \end{cases}
\end{aligned}
$$

- Quantiles are a generalized form of the median.

Quantiles

Definition: Sample Quantile

The q-quantiles are the cut points that divide a sorted data sample $A = (a_0, a_1, \ldots, a_{n-1})$ where $a_{i-1} \leq a_i \; \forall i \in 1 \ldots (n-1)$ into q equally-sized parts. $\text{quantile}_q^k(A)$ be the k^{th} q-quantile, with $k \in 1 \ldots (q-1)$, i.e., there are $q-1$ of the q-quantiles.

$$
\begin{aligned}
h &= (n-1)\frac{k}{q} \\
\text{quantile}_q^k(A) &= \begin{cases} a_h & \text{if } h \text{ is integer} \\ a_{\lfloor h \rfloor} + (h - \lfloor h \rfloor) * \left(a_{\lfloor h \rfloor + 1} - a_{\lfloor h \rfloor}\right) & \text{otherwise} \end{cases}
\end{aligned}
$$

- Quantiles are a generalized form of the median.
- The $\text{quantile}_1^2(A)$ is the median of A

Quantiles

Definition: Sample Quantile

The q-quantiles are the cut points that divide a sorted data sample $A = (a_0, a_1, \ldots, a_{n-1})$ where $a_{i-1} \leq a_i \; \forall i \in 1 \ldots (n-1)$ into q equally-sized parts. $\text{quantile}_q^k(A)$ be the k^{th} q-quantile, with $k \in 1 \ldots (q-1)$, i.e., there are $q-1$ of the q-quantiles.

$$\begin{aligned} h &= (n-1)\frac{k}{q} \\ \text{quantile}_q^k(A) &= \begin{cases} a_h & \text{if } h \text{ is integer} \\ a_{\lfloor h \rfloor} + (h - \lfloor h \rfloor) * \left(a_{\lfloor h \rfloor + 1} - a_{\lfloor h \rfloor}\right) & \text{otherwise} \end{cases} \end{aligned}$$

- Quantiles are a generalized form of the median.
- The $\text{quantile}_1^2(A)$ is the median of A
- 4-quantiles are called *quartiles*.

Quantiles

Definition: Sample Quantile

The q-quantiles are the cut points that divide a sorted data sample $A = (a_0, a_1, \dots, a_{n-1})$ where $a_{i-1} \leq a_i \; \forall i \in 1 \dots (n-1)$ into q equally-sized parts. $\text{quantile}_q^k(A)$ be the k^{th} q-quantile, with $k \in 1 \dots (q-1)$, i.e., there are $q-1$ of the q-quantiles.

$$
\begin{aligned}
h &= (n-1)\frac{k}{q} \\
\text{quantile}_q^k(A) &= \begin{cases} a_h & \text{if } h \text{ is integer} \\ a_{\lfloor h \rfloor} + (h - \lfloor h \rfloor) * \left(a_{\lfloor h \rfloor + 1} - a_{\lfloor h \rfloor}\right) & \text{otherwise} \end{cases}
\end{aligned}
$$

- Quantiles are a generalized form of the median.
- The $\text{quantile}_1^2(A)$ is the median of A
- 4-quantiles are called *quartiles*.
- We often consider *percentiles* or write things like "98% quantile" or "0.98 percentile" or "98% percentile" meaning $\text{quantile}_{100}^{98}$.

Standard Deviation: Example

- Two data samples A and B with $n_a = n_b = 19$ values.

Standard Deviation: Example

- Two data samples A and B with $n_a = n_b = 19$ values.

$$\begin{aligned} A &= (1, 3, 4, 4, 4, 5, 6, 6, 6, 6, 7, 7, 9, 9, 9, 10, 11, 12, 14) \\ \mathrm{mean}(A) &= 7 \\ B &= (1, 3, 4, 4, 4, 5, 6, 6, 6, 6, 7, 7, 9, 9, 9, 10, 11, 12, 10'008) \\ \mathrm{mean}(B) &= 533 \end{aligned}$$

Standard Deviation: Example

- Two data samples A and B with $n_a = n_b = 19$ values.

$$
\begin{aligned}
A &= (1,3,4,4,4,5,6,6,6,6,7,7,9,9,9,10,11,12,14) \\
\mathrm{mean}(A) &= 7 \\
B &= (1,3,4,4,4,5,6,6,6,6,7,7,9,9,9,10,11,12,10'008) \\
\mathrm{mean}(B) &= 533 \\
\mathrm{var}(A) &= \frac{1}{19-1}\sum_{i=1}^{19}(a_i - \mathrm{mean}(A))^2 = \frac{198}{18} = 11 \\
\mathrm{var}(B) &= \frac{1}{19-1}\sum_{i=1}^{19}(b_i - \mathrm{mean}(B))^2 = \frac{94'763'306}{18} \approx 5'264'628
\end{aligned}
$$

Standard Deviation: Example

- Two data samples A and B with $n_a = n_b = 19$ values.

$$\begin{aligned}
A &= (1, 3, 4, 4, 4, 5, 6, 6, 6, 6, 7, 7, 9, 9, 9, 10, 11, 12, 14) \\
\text{mean}(A) &= 7 \\
B &= (1, 3, 4, 4, 4, 5, 6, 6, 6, 6, 7, 7, 9, 9, 9, 10, 11, 12, 10'008) \\
\text{mean}(B) &= 533 \\
\text{var}(A) &= \frac{1}{19-1}\sum_{i=1}^{19}(a_i - \text{mean}(A))^2 = \frac{198}{18} = 11 \\
\text{var}(B) &= \frac{1}{19-1}\sum_{i=1}^{19}(b_i - \text{mean}(B))^2 = \frac{94'763'306}{18} \approx 5'264'628 \\
\text{sd}(A) &= \sqrt{\text{var}\, A} = \sqrt{11} \approx 3.3 \\
\text{sd}(B) &= \sqrt{\text{var}\, B} = \sqrt{\frac{94'763'306}{18}} \approx 2294
\end{aligned}$$

Standard Deviation: Example

- Two data samples A and B with $n_a = n_b = 19$ values.

$$\begin{aligned}
A &= (1,3,4,4,4,5,6,6,6,6,7,7,9,9,9,10,11,12,14)\\
B &= (1,3,4,4,4,5,6,6,6,6,7,7,9,9,9,10,11,12,10'008)\\
\operatorname{var}(A) &= \frac{1}{19-1}\sum_{i=1}^{19}(a_i - \operatorname{mean}(A))^2 = \frac{198}{18} = 11\\
\operatorname{var}(B) &= \frac{1}{19-1}\sum_{i=1}^{19}(b_i - \operatorname{mean}(B))^2 = \frac{94'763'306}{18} \approx 5'264'628\\
\operatorname{sd}(A) &= \sqrt{\operatorname{var} A} = \sqrt{11} \approx 3.3\\
\operatorname{sd}(B) &= \sqrt{\operatorname{var} B} = \sqrt{\frac{94'763'306}{18}} \approx 2294
\end{aligned}$$

- Being based on the arithmetic mean, the variance and standard deviation are heavily influenced by outliers – with all pros and cons coming with that...

Quantiles: Example

- Two data samples A and B with $n_a = n_b = 19$ values.

Quantiles: Example

- Two data samples A and B with $n_a = n_b = 19$ values.

$$
\begin{aligned}
A &= (1, 3, 4, 4, 4, 5, 6, 6, 6, 6, 7, 7, 9, 9, 9, 10, 11, 12, 14) \\
B &= (1, 3, 4, 4, 4, 5, 6, 6, 6, 6, 7, 7, 9, 9, 9, 10, 11, 12, 10'008)
\end{aligned}
$$

Quantiles: Example

- Two data samples A and B with $n_a = n_b = 19$ values.

$$
\begin{aligned}
A &= (1, 3, 4, 4, 4, 5, 6, 6, 6, 6, 7, 7, 9, 9, 9, 10, 11, 12, 14) \\
B &= (1, 3, 4, 4, 4, 5, 6, 6, 6, 6, 7, 7, 9, 9, 9, 10, 11, 12, \textcolor{red}{10'008}) \\
\text{quantile}^1_4(A) &= \text{quantile}^1_4(B) = 4.5 \\
\text{quantile}^3_4(A) &= \text{quantile}^3_4(B) = 9
\end{aligned}
$$

Quantiles: Example

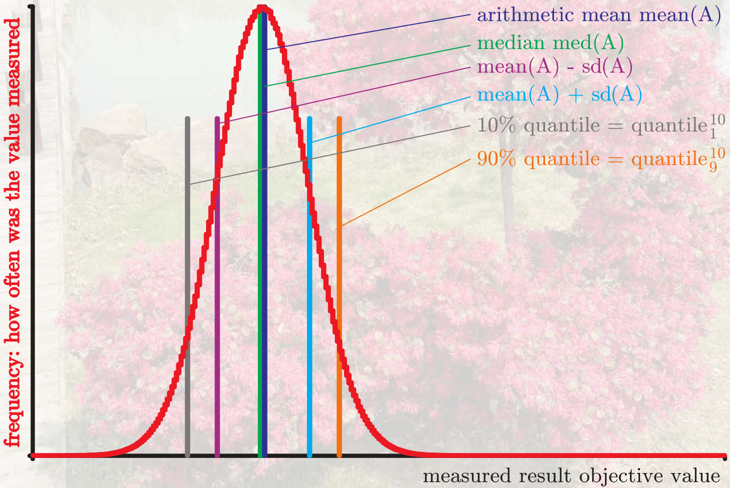
- Two data samples A and B with $n_a = n_b = 19$ values.

$$
\begin{aligned}
A &= (1, 3, 4, 4, 4, 5, 6, 6, 6, 6, 7, 7, 9, 9, 9, 10, 11, 12, 14) \\
B &= (1, 3, 4, 4, 4, 5, 6, 6, 6, 6, 7, 7, 9, 9, 9, 10, 11, 12, \textcolor{red}{10'008}) \\
\text{quantile}_4^1(A) &= \text{quantile}_4^1(B) = 4.5 \\
\text{quantile}_4^3(A) &= \text{quantile}_4^3(B) = 9
\end{aligned}
$$

- Being generalizations of the median, the quantiles are little influenced by outliers – with all pros and cons coming with that. . .

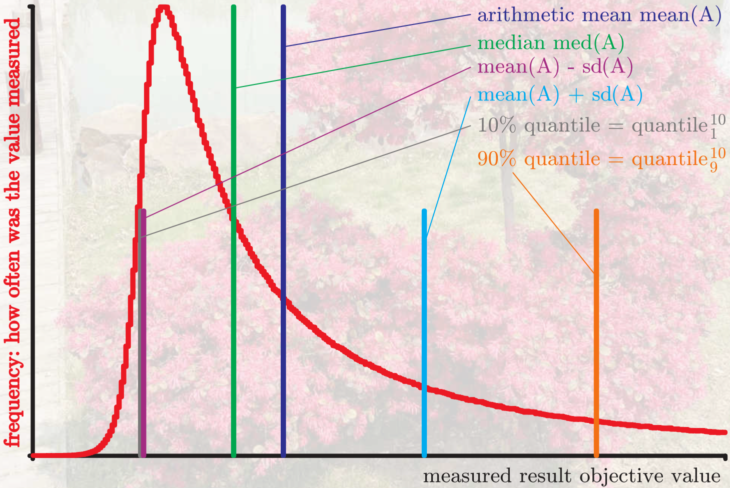
Further Example

- The implicit assumption that $\text{mean} \pm \text{sd}$ is a meaningful range is not always true!



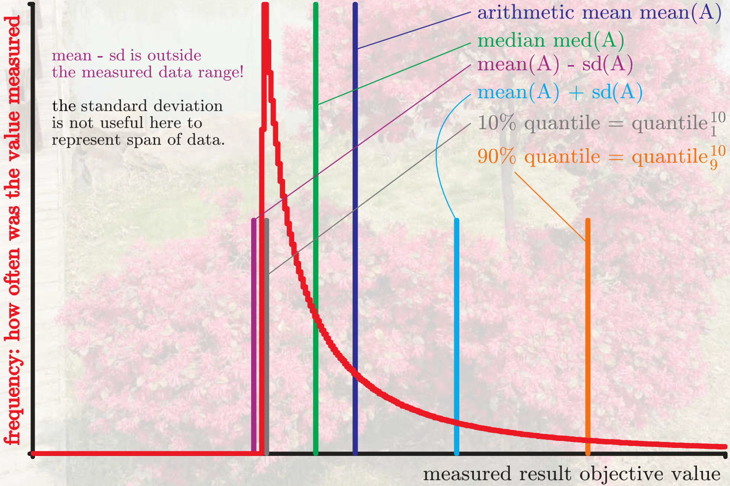
Further Example

- The implicit assumption that $\text{mean} \pm \text{sd}$ is a meaningful range is not always true!



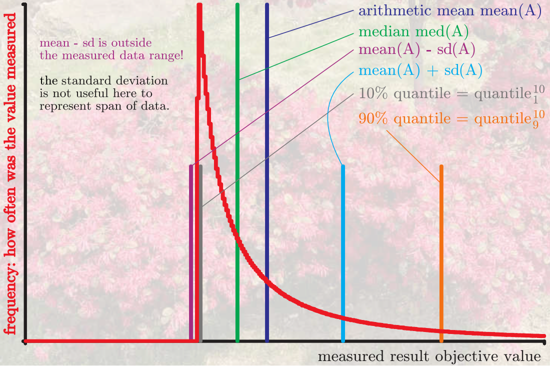
Further Example

- The implicit assumption that $\text{mean} \pm \text{sd}$ is a meaningful range is not always true!



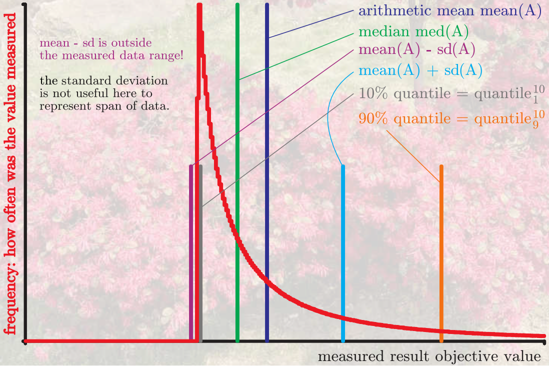
Further Example

- The implicit assumption that $\text{mean} \pm \text{sd}$ is a meaningful range is not always true!
- Such a shape is possible in optimization!



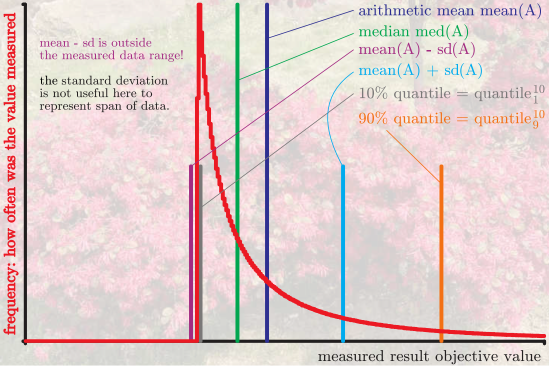
Further Example

- The implicit assumption that $\text{mean} \pm \text{sd}$ is a meaningful range is not always true!
- Such a shape is possible in optimization:
  - The global optimum marks a lower bound for the possible objective values.



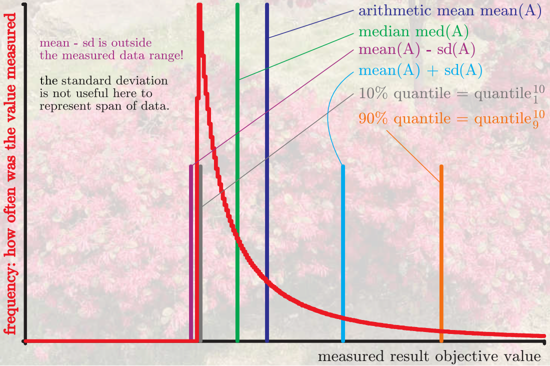
Further Example

- The implicit assumption that $\text{mean} \pm \text{sd}$ is a meaningful range is not always true!
- Such a shape is possible in optimization:
  - The global optimum marks a lower bound for the possible objective values.
  - A good algorithm often returns results which are close-to-optimal.



Further Example

- The implicit assumption that $\text{mean} \pm \text{sd}$ is a meaningful range is not always true!
- Such a shape is possible in optimization:
  - The global optimum marks a lower bound for the possible objective values.
  - A good algorithm often returns results which are close-to-optimal.
  - There may be a long tail of few but significantly worse runs.



Further Example

- The implicit assumption that $\mathrm{mean} \pm \mathrm{sd}$ is a meaningful range is not always true!
- Such a shape is possible in optimization:
  - The global optimum marks a lower bound for the possible objective values.
  - A good algorithm often returns results which are close-to-optimal.
  - There may be a long tail of few but significantly worse runs.
  - A statement such as *"For this TSP instance, our algorithm can find tours with a length of* 100 ± 120 *km."* makes little sense...

Statistical Comparisons

Introduction

- We can now, e.g., perform 20 runs each with two different optimization algorithms on one problem instance and compute the medians of a performance indicator.

Introduction

- We can now, e.g., perform 20 runs each with two different optimization algorithms on one problem instance and compute the medians of a performance indicator.
- Likely, they will be different.

Introduction

- We can now, e.g., perform 20 runs each with two different optimization algorithms on one problem instance and compute the medians of a performance indicator.
- Likely, they will be different.
- For one of the two algorithms, the results will be better.

Introduction

- We can now, e.g., perform 20 runs each with two different optimization algorithms on one problem instance and compute the medians of a performance indicator.
- Likely, they will be different.
- For one of the two algorithms, the results will be better.
- What does this mean?

Introduction

- We can now, e.g., perform 20 runs each with two different optimization algorithms on one problem instance and compute the medians of a performance indicator.
- Likely, they will be different.
- For one of the two algorithms, the results will be better.
- What does this mean?
- It means that one of the two algorithms is better.

Introduction

- We can now, e.g., perform 20 runs each with two different optimization algorithms on one problem instance and compute the medians of a performance indicator.
- Likely, they will be different.
- For one of the two algorithms, the results will be better.
- What does this mean?
- It means that one of the two algorithms is better with a certain probability.

Introduction

- We can now, e.g., perform 20 runs each with two different optimization algorithms on one problem instance and compute the medians of a performance indicator.
- Likely, they will be different.
- For one of the two algorithms, the results will be better.
- What does this mean?
- It means that one of the two algorithms is better with a certain probability.
- If we say *"A is better than B,"* we have a certain probability p to be wrong.

Introduction

- We can now, e.g., perform 20 runs each with two different optimization algorithms on one problem instance and compute the medians of a performance indicator.
- Likely, they will be different.
- For one of the two algorithms, the results will be better.
- What does this mean?
- It means that one of the two algorithms is better with a certain probability.
- If we say *"A is better than B,"* we have a certain probability p to be wrong.
- The statement *"A is better than B"* makes only sense after we have decided about an upper bound α for the acceptable error probability p! (and if $p < \alpha$, obviously)

Statistical Tests

- Compare two data samples $A = (a_1, a_2, \dots)$ and $B = (b_1, b_2, \dots)$ and

Statistical Tests

- Compare two data samples $A = (a_1, a_2, \dots)$ and $B = (b_1, b_2, \dots)$ and
- get a result (e.g., *"The median of A is bigger than the median of B"*) together with an error probability p that the conclusion is wrong.

Statistical Tests

- Compare two data samples $A = (a_1, a_2, \ldots)$ and $B = (b_1, b_2, \ldots)$ and
- get a result (e.g., *"The median of A is bigger than the median of B"*) together with an error probability p that the conclusion is wrong.
- If p is less than a previously chosen significance level (upper bound) α, we can accept the conclusion.

Statistical Tests

- Compare two data samples $A = (a_1, a_2, \dots)$ and $B = (b_1, b_2, \dots)$ and
- get a result (e.g., *"The median of A is bigger than the median of B"*) together with an error probability p that the conclusion is wrong.
- If p is less than a previously chosen significance level (upper bound) α, we can accept the conclusion.
- Otherwise, the observation is not significant.

Statistical Tests

- Compare two data samples $A = (a_1, a_2, \dots)$ and $B = (b_1, b_2, \dots)$ and
- get a result (e.g., *"The median of A is bigger than the median of B"*) together with an error probability p that the conclusion is wrong.
- If p is less than a previously chosen significance level (upper bound) α, we can accept the conclusion.
- Otherwise, the observation is not significant and must be ignored.

Statistical Tests

- Compare two data samples $A = (a_1, a_2, \dots)$ and $B = (b_1, b_2, \dots)$ and
- get a result (e.g., *"The median of A is bigger than the median of B"*) together with an error probability p that the conclusion is wrong.
- If p is less than a previously chosen significance level (upper bound) α, we can accept the conclusion.
- Otherwise, the observation is not significant and must be ignored.
- But how can we arrive at such statements? How can we even estimate a probability to be wrong?

Statistical Tests

- Compare two data samples $A = (a_1, a_2, \dots)$ and $B = (b_1, b_2, \dots)$ and
- get a result (e.g., *"The median of A is bigger than the median of B"*) together with an error probability p that the conclusion is wrong.
- If p is less than a previously chosen significance level (upper bound) α, we can accept the conclusion.
- Otherwise, the observation is not significant and must be ignored.
- But how can we arrive at such statements? How can we even estimate a probability to be wrong?
- Disclaimer: I am not a mathematician. What follows are simplified explanations of concepts.

Example for Underlying Idea

- Coin flip game: We flip a coin. If it is heads, I give you 1 RMB, if it is tails, you give me 1 RMB.

Example for Underlying Idea

- Coin flip game: We flip a coin. If it is heads, I give you 1 RMB, if it is tails, you give me 1 RMB.

Example for Underlying Idea

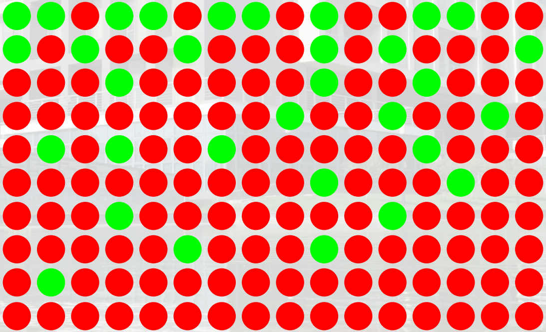
- Coin flip game: We flip a coin. If it is heads, I give you 1 RMB, if it is tails, you give me 1 RMB.
- We play 160 times.

Example for Underlying Idea

- Coin flip game: We flip a coin. If it is heads, I give you 1 RMB, if it is tails, you give me 1 RMB.
- We play 160 times.
- I win 128 times. You win 32 times.

Example for Underlying Idea

- Coin flip game: We flip a coin. If it is heads, I give you 1 RMB, if it is tails, you give me 1 RMB.
- We play 160 times.
- I win 128 times. You win 32 times.



Example for Underlying Idea

- Coin flip game: We flip a coin. If it is heads, I give you 1 RMB, if it is tails, you give me 1 RMB.
- We play 160 times.
- I win 128 times. You win 32 times.
- Did I cheat? Is my coin "fixed?" (i.e., is your chance to win $\neq 0.5$)

Example for Underlying Idea

- Coin flip game: We flip a coin. If it is heads, I give you 1 RMB, if it is tails, you give me 1 RMB.
- We play 160 times.
- I win 128 times. You win 32 times.
- Did I cheat? Is my coin "fixed?" (i.e., is your chance to win $\neq 0.5$)
- Assumption: I cheat. (alternative hypothesis H_1)

Example for Underlying Idea

- Coin flip game: We flip a coin. If it is heads, I give you 1 RMB, if it is tails, you give me 1 RMB.
- We play 160 times.
- I win 128 times. You win 32 times.
- Did I cheat? Is my coin "fixed?" (i.e., is your chance to win $\neq 0.5$)
- Assumption: I cheat. (alternative hypothesis H_1)
- It is impossible to compute my winning probability if I cheated...

Example for Underlying Idea

- Coin flip game: We flip a coin. If it is heads, I give you 1 RMB, if it is tails, you give me 1 RMB.
- We play 160 times.
- I win 128 times. You win 32 times.
- Did I cheat? Is my coin "fixed?" (i.e., is your chance to win $\neq 0.5$)
- Assumption: I cheat. (alternative hypothesis H_1)
- It is impossible to compute my winning probability if I cheated...
- Counter-Assumption: I did not cheat. (null hypothesis H_0)

Example for Underlying Idea

- Coin flip game: We flip a coin. If it is heads, I give you 1 RMB, if it is tails, you give me 1 RMB.
- We play 160 times.
- I win 128 times. You win 32 times.
- Did I cheat? Is my coin "fixed?" (i.e., is your chance to win $\neq 0.5$)
- Assumption: I cheat. (alternative hypothesis H_1)
- It is impossible to compute my winning probability if I cheated...
- Counter-Assumption: I did not cheat. (null hypothesis H_0)
- How likely is it that I win at least 128 times if I did not cheat?

Example for Underlying Idea

- Coin flip game: We flip a coin. If it is heads, I give you 1 RMB, if it is tails, you give me 1 RMB.
- We play 160 times.
- I win 128 times. You win 32 times.
- Did I cheat? Is my coin "fixed?" (i.e., is your chance to win $\neq 0.5$)
- Assumption: I cheat. (alternative hypothesis H_1)
- It is impossible to compute my winning probability if I cheated...
- Counter-Assumption: I did not cheat. (null hypothesis H_0)
- How likely is it that I win at least 128 times if I did not cheat?
- (What we will do right now is called *binomial test*.)

Example for Underlying Idea

- How likely is it that I win at least 128 times if I did not cheat?

Example for Underlying Idea

- How likely is it that I win at least 128 times if I did not cheat?
- Then, the probabilities for heads and tails are $q = P(\texttt{head}) = P(\texttt{tail}) = 0.5$.

Example for Underlying Idea

- How likely is it that I win at least 128 times if I did not cheat?
- Then, the probabilities for heads and tails are $q = P(\texttt{head}) = P(\texttt{tail}) = 0.5$.
- Flipping a coin n times is a Bernoulli Process

Example for Underlying Idea

- How likely is it that I win at least 128 times if I did not cheat?
- Then, the probabilities for heads and tails are $q = P(\texttt{head}) = P(\texttt{tail}) = 0.5$.
- Flipping a coin n times is a Bernoulli Process
- The probability $P(k|n)$ to flip $k \in 0..n$ times heads (or tails) is thus:

$$P(k|n) = \binom{n}{k} 0.5^k * (1 - 0.5)^{n-k} = \binom{n}{k} 0.5^k * 0.5^{n-k} = \binom{n}{k} \frac{1}{2^n}$$

Example for Underlying Idea

- How likely is it that I win at least 128 times if I did not cheat?
- Then, the probabilities for heads and tails are $q = P(\texttt{head}) = P(\texttt{tail}) = 0.5$.
- Flipping a coin n times is a Bernoulli Process
- The probability $P(k|n)$ to flip $k \in 0..n$ times heads (or tails) is thus:

$$P(k|n) = \binom{n}{k} 0.5^k * (1-0.5)^{n-k} = \binom{n}{k} 0.5^k * 0.5^{n-k} = \binom{n}{k} \frac{1}{2^n}$$

- For winning at least $z = 128$ times, we need to compute:

$$P(k \geq z|n) = \sum_{i=z}^{n} P(i|n)$$

Example for Underlying Idea

- How likely is it that I win at least 128 times if I did not cheat?
- Then, the probabilities for heads and tails are $q = P(\texttt{head}) = P(\texttt{tail}) = 0.5$.
- Flipping a coin n times is a Bernoulli Process
- The probability $P(k|n)$ to flip $k \in 0..n$ times heads (or tails) is thus:

$$P(k|n) = \binom{n}{k} 0.5^k * (1-0.5)^{n-k} = \binom{n}{k} 0.5^k * 0.5^{n-k} = \binom{n}{k} \frac{1}{2^n}$$

- For winning at least $z = 128$ times, we need to compute:

$$P(k \geq z|n) = \sum_{i=z}^{n} P(i|n) = \sum_{i=128}^{160} P(i|160) = \sum_{i=128}^{160} \left[\binom{160}{i} \frac{1}{2^{160}} \right]$$

Example for Underlying Idea

- How likely is it that I win at least 128 times if I did not cheat?
- Then, the probabilities for heads and tails are $q = P(\texttt{head}) = P(\texttt{tail}) = 0.5$.
- Flipping a coin n times is a Bernoulli Process
- The probability $P(k|n)$ to flip $k \in 0..n$ times heads (or tails) is thus:

$$P(k|n) = \binom{n}{k} 0.5^k * (1-0.5)^{n-k} = \binom{n}{k} 0.5^k * 0.5^{n-k} = \binom{n}{k} \frac{1}{2^n}$$

- For winning at least $z = 128$ times, we need to compute:

$$P(k \geq z|n) = \sum_{i=z}^{n} P(i|n) = \frac{1}{2^{160}} \sum_{i=128}^{160} \binom{160}{i}$$

Example for Underlying Idea

- How likely is it that I win at least 128 times if I did not cheat?
- Then, the probabilities for heads and tails are $q = P(\texttt{head}) = P(\texttt{tail}) = 0.5$.
- Flipping a coin n times is a Bernoulli Process
- The probability $P(k|n)$ to flip $k \in 0..n$ times heads (or tails) is thus:

$$P(k|n) = \binom{n}{k} 0.5^k * (1-0.5)^{n-k} = \binom{n}{k} 0.5^k * 0.5^{n-k} = \binom{n}{k} \frac{1}{2^n}$$

- For winning at least $z = 128$ times, we need to compute:

$$\begin{aligned} P(k \geq z|n) &= \sum_{i=z}^{n} P(i|n) = \frac{1}{2^{160}} \sum_{i=128}^{160} \binom{160}{i} \\ &= \frac{1'538'590'628'148'134'280'316'221'828'039'113}{365'375'409'332'725'729'550'921'208'179'070'754'913'983'135'744} \end{aligned}$$

Example for Underlying Idea

- How likely is it that I win at least 128 times if I did not cheat?
- Then, the probabilities for heads and tails are $q = P(\texttt{head}) = P(\texttt{tail}) = 0.5$.
- Flipping a coin n times is a Bernoulli Process
- The probability $P(k|n)$ to flip $k \in 0..n$ times heads (or tails) is thus:

$$P(k|n) = \binom{n}{k} 0.5^k * (1-0.5)^{n-k} = \binom{n}{k} 0.5^k * 0.5^{n-k} = \binom{n}{k} \frac{1}{2^n}$$

- For winning at least $z = 128$ times, we need to compute:

$$P(k \geq z|n) = \sum_{i=z}^{n} P(i|n) = \frac{1}{2^{160}} \sum_{i=128}^{160} \binom{160}{i} \approx \frac{1.539 * 10^{33}}{3.654 * 10^{47}}$$

Example for Underlying Idea

- How likely is it that I win at least 128 times if I did not cheat?
- Then, the probabilities for heads and tails are $q = P(\texttt{head}) = P(\texttt{tail}) = 0.5$.
- Flipping a coin n times is a Bernoulli Process
- The probability $P(k|n)$ to flip $k \in 0..n$ times heads (or tails) is thus:

$$P(k|n) = \binom{n}{k} 0.5^k * (1 - 0.5)^{n-k} = \binom{n}{k} 0.5^k * 0.5^{n-k} = \binom{n}{k} \frac{1}{2^n}$$

- For winning at least $z = 128$ times, we need to compute:

$$\begin{aligned} P(k \geq z|n) &= \sum_{i=z}^{n} P(i|n) = \frac{1}{2^{160}} \sum_{i=128}^{160} \binom{160}{i} \approx \frac{1.539 * 10^{33}}{3.654 * 10^{47}} \\ &\approx 0.0000000000000421098571 \end{aligned}$$

Example for Underlying Idea

- How likely is it that I win at least 128 times if I did not cheat?
- Then, the probabilities for heads and tails are $q = P(\texttt{head}) = P(\texttt{tail}) = 0.5$.
- Flipping a coin n times is a Bernoulli Process
- The probability $P(k|n)$ to flip $k \in 0..n$ times heads (or tails) is thus:

$$P(k|n) = \binom{n}{k} 0.5^k * (1-0.5)^{n-k} = \binom{n}{k} 0.5^k * 0.5^{n-k} = \binom{n}{k} \frac{1}{2^n}$$

- For winning at least $z = 128$ times, we need to compute:

$$\begin{aligned} P(k \geq z|n) &= \sum_{i=z}^{n} P(i|n) = \frac{1}{2^{160}} \sum_{i=128}^{160} \binom{160}{i} \approx \frac{1.539 * 10^{33}}{3.654 * 10^{47}} \\ &\approx 4.211 \cdot 10^{-15} \end{aligned}$$

Example for Underlying Idea

- Question: How likely is it that I win at least 128 times if I did not cheat?

Example for Underlying Idea

- Question: How likely is it that I win at least 128 times if I did not cheat?
- If the coin was an ideal coin, the chance that I win at least 128 out of 160 times is about $4 \cdot 10^{-15}$.

Example for Underlying Idea

- Question: How likely is it that I win at least 128 times if I did not cheat?
- If the coin was an ideal coin, the chance that I win at least 128 out of 160 times is about $4 \cdot 10^{-15}$.
- If you claim that I cheat, your chance to be wrong is about $4 \cdot 10^{-15}$.

Example for Underlying Idea

- Question: How likely is it that I win at least 128 times if I did not cheat?
- If the coin was an ideal coin, the chance that I win at least 128 out of 160 times is about $4 \cdot 10^{-15}$.
- If you claim that I cheat, your chance to be wrong is about $4 \cdot 10^{-15}$.
- Thus, if we cannot accept a chance p to be wrong higher than a significance level $\alpha = 1\%$, we can still say:

The observation is significant, I did likely cheat.

A More Specific Example for Tests

- We want to compare two algorithms $\mathcal{A}$ and $\mathcal{B}$ on a given problem instance.

A More Specific Example for Tests

- We want to compare two algorithms $\mathcal{A}$ and $\mathcal{B}$ on a given problem instance.
- We have conducted a small experiment and measured objective values of their final results in a few runs in form of the two data sets A and B, respectively:

$$
\begin{aligned}
A &= (2, 5, 6, 7, 9, 10) \\
B &= (1, 3, 4, 8)
\end{aligned}
$$

A More Specific Example for Tests

- We want to compare two algorithms $\mathcal{A}$ and $\mathcal{B}$ on a given problem instance.
- We have conducted a small experiment and measured objective values of their final results in a few runs in form of the two data sets A and B, respectively:

$$\begin{aligned} A &= (2, 5, 6, 7, 9, 10) \\ B &= (1, 3, 4, 8) \end{aligned}$$

- From this, we can estimate the arithmetic means:

A More Specific Example for Tests

- We want to compare two algorithms $\mathcal{A}$ and $\mathcal{B}$ on a given problem instance.
- We have conducted a small experiment and measured objective values of their final results in a few runs in form of the two data sets A and B, respectively:

$$\begin{aligned} A &= (2, 5, 6, 7, 9, 10) \\ B &= (1, 3, 4, 8) \end{aligned}$$

- From this, we can estimate the arithmetic means:

$$\begin{aligned} \mathrm{mean}(A) &= \frac{39}{6} = 6.5 \\ \mathrm{mean}(B) &= \frac{16}{4} = 4 \end{aligned}$$

A More Specific Example

$$\text{mean}(A) = \frac{39}{6} = 6.5$$
$$\text{mean}(B) = \frac{16}{4} = 4$$

- It looks like algorithm $\mathcal{B}$ may produce the smaller objective values.

A More Specific Example

$$\text{mean}(A) = \frac{39}{6} = 6.5$$
$$\text{mean}(B) = \frac{16}{4} = 4$$

- It looks like algorithm $\mathcal{B}$ may produce the smaller objective values.
- But is this assumption justified based on the data we have?

A More Specific Example

$$\begin{aligned}\mathrm{mean}(A) &= \frac{39}{6} = 6.5\\ \mathrm{mean}(B) &= \frac{16}{4} = 4\end{aligned}$$

- It looks like algorithm $\mathcal{B}$ may produce the smaller objective values.
- But is this assumption justified based on the data we have?
- Is the difference between $\mathrm{mean}(A)$ and $\mathrm{mean}(B)$ significant at a threshold of $\alpha = 2\%$?

A More Specific Example

- If $\mathcal{B}$ is truly better than $\mathcal{A}$, which is our hypothesis H_1, then we cannot calculate anything.

A More Specific Example

- If $\mathcal{B}$ is truly better than $\mathcal{A}$, which is our hypothesis H_1, then we cannot calculate anything.
- Let us therefore assume as null hypothesis H_0 the observed difference did just happen by chance and, well, $\mathcal{A} \equiv \mathcal{B}$.

A More Specific Example

- If $\mathcal{B}$ is truly better than $\mathcal{A}$, which is our hypothesis H_1, then we cannot calculate anything.
- Let us therefore assume as null hypothesis H_0 the observed difference did just happen by chance and, well, $\mathcal{A} \equiv \mathcal{B}$.
- Then, this would mean that the data samples A and B stem from the same algorithm (as $\mathcal{A} \equiv \mathcal{B}$).

A More Specific Example

- If $\mathcal{B}$ is truly better than $\mathcal{A}$, which is our hypothesis H_1, then we cannot calculate anything.
- Let us therefore assume as null hypothesis H_0 the observed difference did just happen by chance and, well, $\mathcal{A} \equiv \mathcal{B}$.
- Then, this would mean that the data samples A and B stem from the same algorithm (as $\mathcal{A} \equiv \mathcal{B}$).
- The division into the two sets would only be artificial, an artifact of our experimental design.

A More Specific Example

- If $\mathcal{B}$ is truly better than $\mathcal{A}$, which is our hypothesis H_1, then we cannot calculate anything.
- Let us therefore assume as null hypothesis H_0 the observed difference did just happen by chance and, well, $\mathcal{A} \equiv \mathcal{B}$.
- Then, this would mean that the data samples A and B stem from the same algorithm (as $\mathcal{A} \equiv \mathcal{B}$).
- The division into the two sets would only be artificial, an artifact of our experimental design.
- Instead of having two data samples, we only have one, namely the union set O with 10 elements:

A More Specific Example

- If $\mathcal{B}$ is truly better than $\mathcal{A}$, which is our hypothesis H_1, then we cannot calculate anything.
- Let us therefore assume as null hypothesis H_0 the observed difference did just happen by chance and, well, $\mathcal{A} \equiv \mathcal{B}$.
- Then, this would mean that the data samples A and B stem from the same algorithm (as $\mathcal{A} \equiv \mathcal{B}$).
- The division into the two sets would only be artificial, an artifact of our experimental design.
- Instead of having two data samples, we only have one, namely the union set O with 10 elements:

$$O = A \cup B = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$$

A More Specific Example

$$O = A \cup B = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$$

- Any division C into two sets with 4 and 6 elements has the same probability

A More Specific Example

$$O = A \cup B = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$$

- Any division C into two sets with 4 and 6 elements has the same probability
- $|O| = 10$

A More Specific Example

$$O = A \cup B = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$$

- Any division C into two sets with 4 and 6 elements has the same probability
- $|O| = 10$
- There are $\binom{10}{4} = 210$ different ways to draw 4 (or 6) elements from O

A More Specific Example

$$O = A \cup B = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$$

- Any division C into two sets with 4 and 6 elements has the same probability
- $|O| = 10$
- There are $\binom{10}{4} = 210$ different ways to draw 4 (or 6) elements from O
- If H_0 holds, all have the same probability

A More Specific Example

$$O = A \cup B = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$$

- Any division C into two sets with 4 and 6 elements has the same probability
- $|O| = 10$
- There are $\binom{10}{4} = 210$ different ways to draw 4 (or 6) elements from O
- If H_0 holds, all have the same probability
- Let's use a Python[60] program to test the combinations

A More Specific Example

```python
"""Enumerate all combinations of numbers 1 to 10."""
mean_lower_or_equal_to_4 = 0  # how often did we find a mean <= 4
total_combinations       = 0  # total number of tested combinations

for i in range(1, 11):                  # i goes from 1 to 10
    for j in range(1, i):               # j goes from 1 to i - 1
        for k in range(1, j):           # k goes from 1 to j - 1
            for l in range(1, k):       # l goes from 1 to k - 1
                if ((i + j + k + l) / 4) <= 4:  # check for extreme case
                    mean_lower_or_equal_to_4 += 1  # count extreme case
                total_combinations += 1    # count all combinations

print(f" combinations with mean <= 4: {mean_lower_or_equal_to_4}")
print(f"total number of combinations: {total_combinations}")
```

A More Specific Example

```python
"""Enumerate all combinations of numbers 1 to 10."""
mean_lower_or_equal_to_4 = 0  # how often did we find a mean <= 4
total_combinations       = 0  # total number of tested combinations

for i in range(1, 11):                # i goes from 1 to 10
    for j in range(1, i):             # j goes from 1 to i - 1
        for k in range(1, j):         # k goes from 1 to j - 1
            for l in range(1, k):     # l goes from 1 to k - 1
                if ((i + j + k + l) / 4) <= 4:  # check for extreme case
                    mean_lower_or_equal_to_4 += 1  # count extreme case
                total_combinations += 1    # count all combinations

print(f" combinations with mean <= 4: {mean_lower_or_equal_to_4}")
print(f"total number of combinations: {total_combinations}")
```

```
 combinations with mean <= 4: 27
total number of combinations: 210
```

A More Specific Example

$$O = A \cup B = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$$

- Any division C into two sets with 4 and 6 elements has the same probability
- $|O| = 10$
- There are $\binom{10}{4} = 210$ different ways to draw 4 (or 6) elements from O
- If H_0 holds, all have the same probability
- There are 27 such combinations with a mean of less or equal 4.

A More Specific Example

$$O = A \cup B = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$$

- Any division C into two sets with 4 and 6 elements has the same probability
- $|O| = 10$
- There are $\binom{10}{4} = 210$ different ways to draw 4 (or 6) elements from O
- If H_0 holds, all have the same probability
- There are 27 such combinations with a mean of less or equal 4.
- The probability p to observe a situation at least as extreme as A and B under H_0 is thus:

A More Specific Example

$$O = A \cup B = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$$

- Any division C into two sets with 4 and 6 elements has the same probability
- $|O| = 10$
- There are $\binom{10}{4} = 210$ different ways to draw 4 (or 6) elements from O
- If H_0 holds, all have the same probability
- There are 27 such combinations with a mean of less or equal 4.
- The probability p to observe a situation at least as extreme as A and B under H_0 is thus:

$$p = \frac{\#\text{cases } C:\ \mathrm{mean}(C) \leq \mathrm{mean}(B)}{\#\text{all cases}} = \frac{27}{210} = \frac{9}{70} \approx 0.1286$$

A More Specific Example

- Extreme cases into the other direction are the same, because if $\mathrm{mean}(B) \leq 4$ then $\mathrm{mean}(A) \geq 6.5$ for any division $A \cup B = O$ and vice versa.

$$O = A \cup B = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$$

A More Specific Example

- Extreme cases into the other direction are the same, because if $\mathrm{mean}(B) \leq 4$ then $\mathrm{mean}(A) \geq 6.5$ for any division $A \cup B = O$ and vice versa:

$$\begin{aligned} O &= A \cup B = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10) \\ \sum_{\forall o \in O} o &= \sum_{o=1}^{10} o = \frac{10(10+1)}{2} = 55 \end{aligned}$$

A More Specific Example

- Extreme cases into the other direction are the same, because if $\text{mean}(B) \leq 4$ then $\text{mean}(A) \geq 6.5$ for any division $A \cup B = O$ and vice versa:

$$\begin{aligned} O &= A \cup B = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10) \\ \sum_{\forall o \in O} o &= \sum_{o=1}^{10} o = \frac{10(10+1)}{2} = 55 \end{aligned}$$

$$\text{mean}(B) = \left(\frac{1}{4} \sum_{\forall b \in B} b \right) \leq 4 \quad \Longrightarrow \quad \left(\sum_{\forall b \in B} b \right) \leq 4 * 4 \leq 16$$

A More Specific Example

- Extreme cases into the other direction are the same, because if $\text{mean}(B) \leq 4$ then $\text{mean}(A) \geq 6.5$ for any division $A \cup B = O$ and vice versa:

$$
\begin{aligned}
O &= A \cup B = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10) \\
\sum_{\forall o \in O} o &= \sum_{o=1}^{10} o = \frac{10(10+1)}{2} = 55 \\
\text{mean}(B) = \left(\frac{1}{4} \sum_{\forall b \in B} b\right) \leq 4 &\implies \left(\sum_{\forall b \in B} b\right) \leq 4 * 4 \leq 16 \\
O = A \cup B &\implies \sum_{\forall a \in A} a = \left(\sum_{\forall o \in O} o\right) - \left(\sum_{\forall b \in B} b\right)
\end{aligned}
$$

A More Specific Example

- Extreme cases into the other direction are the same, because if $\text{mean}(B) \leq 4$ then $\text{mean}(A) \geq 6.5$ for any division $A \cup B = O$ and vice versa:

$$\begin{aligned}
O &= A \cup B = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10) \\
\sum_{\forall o \in O} o &= \sum_{o=1}^{10} o = \frac{10(10+1)}{2} = 55 \\
\text{mean}(B) = \left(\frac{1}{4} \sum_{\forall b \in B} b\right) \leq 4 &\implies \left(\sum_{\forall b \in B} b\right) \leq 4 * 4 \leq 16 \\
O = A \cup B &\implies \sum_{\forall a \in A} a = \left(\sum_{\forall o \in O} o\right) - \left(\sum_{\forall b \in B} b\right) \\
\sum_{\forall b \in B} b \leq 16 &\implies \left(\sum_{\forall a \in A} a\right) \geq 55 - 16 \geq 39
\end{aligned}$$

A More Specific Example

- Extreme cases into the other direction are the same, because if $\text{mean}(B) \leq 4$ then $\text{mean}(A) \geq 6.5$ for any division $A \cup B = O$ and vice versa:

$$\begin{aligned}
\sum_{\forall o \in O} o &= \sum_{o=1}^{10} o = \frac{10(10+1)}{2} = 55 \\
\text{mean}(B) = \left(\frac{1}{4}\sum_{\forall b \in B} b\right) \leq 4 &\implies \left(\sum_{\forall b \in B} b\right) \leq 4 * 4 \leq 16 \\
O = A \cup B &\implies \sum_{\forall a \in A} a = \left(\sum_{\forall o \in O} o\right) - \left(\sum_{\forall b \in B} b\right) \\
\sum_{\forall b \in B} b \leq 16 &\implies \left(\sum_{\forall a \in A} a\right) \geq 55 - 16 \geq 39 \\
\text{mean}(A) &= \frac{1}{6}\left(\sum_{\forall a \in A} a\right)
\end{aligned}$$

A More Specific Example

- Extreme cases into the other direction are the same, because if $\text{mean}(B) \leq 4$ then $\text{mean}(A) \geq 6.5$ for any division $A \cup B = O$ and vice versa:

$$\text{mean}(B) = \left(\frac{1}{4} \sum_{\forall b \in B} b\right) \leq 4 \quad \Longrightarrow \quad \left(\sum_{\forall b \in B} b\right) \leq 4 * 4 \leq 16$$

$$O = A \cup B \quad \Longrightarrow \quad \sum_{\forall a \in A} a = \left(\sum_{\forall o \in O} o\right) - \left(\sum_{\forall b \in B} b\right)$$

$$\sum_{\forall b \in B} b \leq 16 \quad \Longrightarrow \quad \left(\sum_{\forall a \in A} a\right) \geq 55 - 16 \geq 39$$

$$\text{mean}(A) \quad = \quad \frac{1}{6} \left(\sum_{\forall a \in A} a\right)$$

$$\text{mean}(B) \leq 4 \quad \Rightarrow \quad \text{mean}(A) \geq \frac{39}{6} \geq 6.5$$

A More Specific Example

- Extreme cases into the other direction are the same, because if $\text{mean}(B) \leq 4$ then $\text{mean}(A) \geq 6.5$ for any division $A \cup B = O$ and vice versa:

$$\text{mean}(B) = \left(\frac{1}{4}\sum_{\forall b \in B} b\right) \leq 4 \implies \left(\sum_{\forall b \in B} b\right) \leq 4 * 4 \leq 16$$

$$O = A \cup B \implies \sum_{\forall a \in A} a = \left(\sum_{\forall o \in O} o\right) - \left(\sum_{\forall b \in B} b\right)$$

$$\sum_{\forall b \in B} b \leq 16 \implies \left(\sum_{\forall a \in A} a\right) \geq 55 - 16 \geq 39$$

$$\text{mean}(A) = \frac{1}{6}\left(\sum_{\forall a \in A} a\right)$$

$$\text{mean}(B) \leq 4 \Rightarrow \text{mean}(A) \geq \frac{39}{6} \geq 6.5$$

- So – of course – we could have also done the test the other way around with the same result!

A More Specific Example

- The probability p to observe a constallation at least as extreme as A or B under H_0 is thus:

$$p = \frac{\#\text{cases } C : \text{mean}(C) \leq \text{mean}(B)}{\#\text{all cases}} = \frac{27}{210} = \frac{9}{70} \approx 0.1286$$

A More Specific Example

- The probability p to observe a constallation at least as extreme as A or B under H_0 is thus:

$$p = \frac{\text{\#cases } C : \text{mean}(C) \leq \text{mean}(B)}{\text{\#all cases}} = \frac{27}{210} = \frac{9}{70} \approx 0.1286$$

- If we claim that A and B are from distributions with means as different as observed...

A More Specific Example

- The probability p to observe a constallation at least as extreme as A or B under H_0 is thus:

$$p = \frac{\#\text{cases } C:\ \mathrm{mean}(C) \leq \ \mathrm{mean}(B)}{\#\text{all cases}} = \frac{27}{210} = \frac{9}{70} \approx 0.1286$$

- If we claim that A and B are from distributions with means as different as observed. . .
- . . . we are wrong with probability $p \approx 0.13$

A More Specific Example

- The probability p to observe a constallation at least as extreme as A or B under H_0 is thus:

$$p = \frac{\text{\#cases } C : \text{mean}(C) \leq \text{mean}(B)}{\text{\#all cases}} = \frac{27}{210} = \frac{9}{70} \approx 0.1286$$

- If we claim that A and B are from distributions with means as different as observed…
- …we are wrong with probability $p \approx 0.13$
- At a significance level of $\alpha = 2\%$, the means of A and B are not significantly different! $(2\% < 0.13)$

A More Specific Example

- The probability p to observe a constallation at least as extreme as A or B under H_0 is thus:

$$p = \frac{\#\text{cases } C: \; \text{mean}(C) \leq \; \text{mean}(B)}{\#\text{all cases}} = \frac{27}{210} = \frac{9}{70} \approx 0.1286$$

- If we claim that A and B are from distributions with means as different as observed. . .
- . . . we are wrong with probability $p \approx 0.13$
- At a significance level of $\alpha = 2\%$, the means of A and B are not significantly different! ($2\% < 0.13$)
- Actually: This here is an example for an *Randomization Test*[11,22].

A More Specific Example

- The probability p to observe a constallation at least as extreme as A or B under H_0 is thus:

$$p = \frac{\#\text{cases } C:\ \mathrm{mean}(C) \leq \mathrm{mean}(B)}{\#\text{all cases}} = \frac{27}{210} = \frac{9}{70} \approx 0.1286$$

- If we claim that A and B are from distributions with means as different as observed...
- ...we are wrong with probability $p \approx 0.13$
- At a significance level of $\alpha = 2\%$, the means of A and B are not significantly different! $(2\% < 0.13)$
- Actually: This here is an example for an *Randomization Test*[11,22].
- The method here is only feasible for small sample sets, real tests are more sophisticated

Statistical Tests: Types

- Statistical tests are more elegant mathematical approaches than the example shown before.

Statistical Tests: Types

- Statistical tests are more elegant mathematical approaches than the example shown before. In order to work, they have preconditions, they make certain assumptions.

Statistical Tests: Types

- Statistical tests are more elegant mathematical approaches than the example shown before. In order to work, they have preconditions, they make certain assumptions.
- There are two types of tests:

Statistical Tests: Types

- Statistical tests are more elegant mathematical approaches than the example shown before. In order to work, they have preconditions, they make certain assumptions.
- There are two types of tests:
  1. Parametric Tests

Statistical Tests: Types

- Statistical tests are more elegant mathematical approaches than the example shown before. In order to work, they have preconditions, they make certain assumptions.
- There are two types of tests:
    1. Parametric Tests
        - Assume that the data samples follow a certain distribution

Statistical Tests: Types

- Statistical tests are more elegant mathematical approaches than the example shown before. In order to work, they have preconditions, they make certain assumptions.
- There are two types of tests:
  1. Parametric Tests
     - Assume that the data samples follow a certain distribution
     - Examples[12]: t-test (assumes normal distribution)

Statistical Tests: Types

- Statistical tests are more elegant mathematical approaches than the example shown before. In order to work, they have preconditions, they make certain assumptions.
- There are two types of tests:
  1. Parametric Tests
     - Assume that the data samples follow a certain distribution
     - Examples[12]: t-test (assumes normal distribution)
     - The distribution of the data we measure is unknown…

Statistical Tests: Types

- Statistical tests are more elegant mathematical approaches than the example shown before. In order to work, they have preconditions, they make certain assumptions.
- There are two types of tests:
  1. Parametric Tests
     - Assume that the data samples follow a certain distribution
     - Examples[12]: t-test (assumes normal distribution)
     - The distribution of the data we measure is unknown. . .
     - . . . and usually not normal nor symmetric (see the further quantiles/stddev plot example).

Statistical Tests: Types

- Statistical tests are more elegant mathematical approaches than the example shown before. In order to work, they have preconditions, they make certain assumptions.
- There are two types of tests:
  1. Parametric Tests
     - Assume that the data samples follow a certain distribution
     - Examples[12]: t-test (assumes normal distribution)
     - The distribution of the data we measure is unknown…
     - …and usually not normal nor symmetric (see the further quantiles/stddev plot example).
     - The condition for using such tests often cannot be met (known distribution)

Statistical Tests: Types

- Statistical tests are more elegant mathematical approaches than the example shown before. In order to work, they have preconditions, they make certain assumptions.
- There are two types of tests:
  1. Parametric Tests
     - Assume that the data samples follow a certain distribution
     - Examples[12]: t-test (assumes normal distribution)
     - The distribution of the data we measure is unknown…
     - …and usually not normal nor symmetric (see the further quantiles/stddev plot example).
     - The condition for using such tests often cannot be met (known distribution)
     - Parametric tests should usually not be used here!

Statistical Tests: Types

- Statistical tests are more elegant mathematical approaches than the example shown before. In order to work, they have preconditions, they make certain assumptions.
- There are two types of tests:
  1. Parametric Tests
  2. Non-Parametric Tests

Statistical Tests: Types

- Statistical tests are more elegant mathematical approaches than the example shown before. In order to work, they have preconditions, they make certain assumptions.
- There are two types of tests:
  1. Parametric Tests
  2. Non-Parametric Tests
     - Make few assumption about the distribution from which the data was sampled.

Statistical Tests: Types

- Statistical tests are more elegant mathematical approaches than the example shown before. In order to work, they have preconditions, they make certain assumptions.
- There are two types of tests:
  1. Parametric Tests
  2. Non-Parametric Tests
     - Make few assumption about the distribution from which the data was sampled.
     - Examples[59]: the Wilcoxon rank sum test with continuity correction (also called Mann-Whitney U test[7,31,44,53]), Fisher's Exact Test[23], the Sign Test[28,53], the Randomization Test[11,22], and Wilcoxon's Signed Rank Test[65].

Statistical Tests: Types

- Statistical tests are more elegant mathematical approaches than the example shown before. In order to work, they have preconditions, they make certain assumptions.
- There are two types of tests:
  1. Parametric Tests
  2. Non-Parametric Tests
     - Make few assumption about the distribution from which the data was sampled.
     - Examples[59]: the Wilcoxon rank sum test with continuity correction (also called Mann-Whitney U test[7,31,44,53]), Fisher's Exact Test[23], the Sign Test[28,53], the Randomization Test[11,22], and Wilcoxon's Signed Rank Test[65].
     - These tests are more robust (less assumptions)

Statistical Tests: Types

- Statistical tests are more elegant mathematical approaches than the example shown before. In order to work, they have preconditions, they make certain assumptions.
- There are two types of tests:
  1. Parametric Tests
  2. Non-Parametric Tests
     - Make few assumption about the distribution from which the data was sampled.
     - Examples[59]: the Wilcoxon rank sum test with continuity correction (also called Mann-Whitney U test[7,31,44,53]), Fisher's Exact Test[23], the Sign Test[28,53], the Randomization Test[11,22], and Wilcoxon's Signed Rank Test[65].
     - These tests are more robust (less assumptions)
     - This usually is the kind of test we want to use.

Statistical Tests: Types

- Statistical tests are more elegant mathematical approaches than the example shown before. In order to work, they have preconditions, they make certain assumptions.
- There are two types of tests:
  1. Parametric Tests
  2. Non-Parametric Tests
     - Make few assumption about the distribution from which the data was sampled.
     - Examples[59]: the Wilcoxon rank sum test with continuity correction (also called Mann-Whitney U test[7,31,44,53]), Fisher's Exact Test[23], the Sign Test[28,53], the Randomization Test[11,22], and Wilcoxon's Signed Rank Test[65].
     - These tests are more robust (less assumptions)
     - This usually is the kind of test we want to use.
     - They work similar to the previous test example, but with larger sample sizes

Statistical Tests: Types

- Statistical tests are more elegant mathematical approaches than the example shown before. In order to work, they have preconditions, they make certain assumptions.
- There are two types of tests:
  1. Parametric Tests
  2. Non-Parametric Tests
     - Make few assumption about the distribution from which the data was sampled.
     - Examples[59]: the Wilcoxon rank sum test with continuity correction (also called Mann-Whitney U test[7,31,44,53]), Fisher's Exact Test[23], the Sign Test[28,53], the Randomization Test[11,22], and Wilcoxon's Signed Rank Test[65].
     - These tests are more robust (less assumptions)
     - This usually is the kind of test we want to use.
     - They work similar to the previous test example, but with larger sample sizes
     - Often, the most suitable test is the Mann-Whitney U test.

A fair warning

- There are many algorithms and even more configuration parameters.

A fair warning

- There are many algorithms and even more configuration parameters.
- All kinds of algorithm modules and parameters have some kind of impact on the performance.

A fair warning

- There are many algorithms and even more configuration parameters.
- All kinds of algorithm modules and parameters have some kind of impact on the performance.
- If I have two different algorithms $\mathcal{A}$ and $\mathcal{B}$, logic dictates that their performance is also different.

A fair warning

- There are many algorithms and even more configuration parameters.
- All kinds of algorithm modules and parameters have some kind of impact on the performance.
- If I have two different algorithms $\mathcal{A}$ and $\mathcal{B}$, logic dictates that their performance is also different.
- But is this difference usually significant?

A fair warning

- There are many algorithms and even more configuration parameters.
- All kinds of algorithm modules and parameters have some kind of impact on the performance.
- If I have two different algorithms $\mathcal{A}$ and $\mathcal{B}$, logic dictates that their performance is also different.
- But is this difference usually significant?
- From the viewpoint of statistics: Probably yes.

A fair warning

- There are many algorithms and even more configuration parameters.
- All kinds of algorithm modules and parameters have some kind of impact on the performance.
- If I have two different algorithms $\mathcal{A}$ and $\mathcal{B}$, logic dictates that their performance is also different.
- But is this difference usually significant?
- From the viewpoint of statistics: Probably yes.
- If I just conduct enough runs, maybe thousands, or millions, than even a difference of 0.001% in performance will pass a test as significant.

A fair warning

- There are many algorithms and even more configuration parameters.
- All kinds of algorithm modules and parameters have some kind of impact on the performance.
- If I have two different algorithms $\mathcal{A}$ and $\mathcal{B}$, logic dictates that their performance is also different.
- But is this difference usually significant?
- From the viewpoint of statistics: Probably yes.
- If I just conduct enough runs, maybe thousands, or millions, than even a difference of 0.001% in performance will pass a test as significant.
- To be practically significant, the measured difference of results should be large enough and statistically significant already with few runs, say, 11 or 21, not just with ≥ 100 runs.

Compare $N \geq 2$ Algorithms

- For comparing $N \geq 2$ algorithms, we can compare any two algorithms with each other

Compare $N \geq 2$ Algorithms

- For comparing $N \geq 2$ algorithms, we can compare any two algorithms with each other
- N Algorithms $\Rightarrow k = N(N-1)/2$ statistical tests (e.g., Mann-Whitney U)

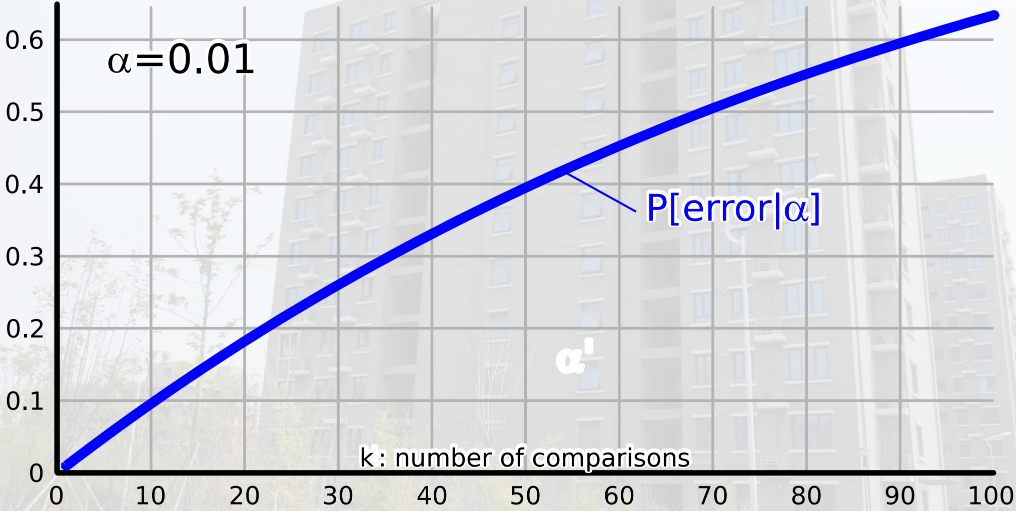
Compare $N \geq 2$ Algorithms

- For comparing $N \geq 2$ algorithms, we can compare any two algorithms with each other
- N Algorithms $\Rightarrow k = N(N-1)/2$ statistical tests (e.g., Mann-Whitney U)
- k tests and each with error proability α

Compare $N \geq 2$ Algorithms

- For comparing $N \geq 2$ algorithms, we can compare any two algorithms with each other
- N Algorithms $\Rightarrow k = N(N-1)/2$ statistical tests (e.g., Mann-Whitney U)
- k tests and each with error proability $\alpha \Longrightarrow$ total probability E to make at least one error $E = 1 - ((1-\alpha)^k)$

Compare $N \geq 2$ Algorithms



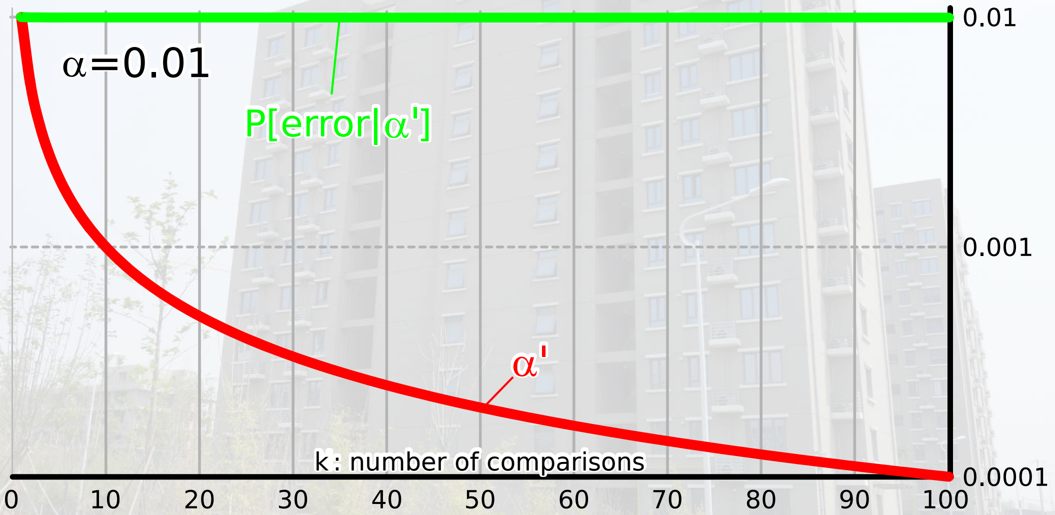
Compare $N \geq 2$ Algorithms

- For comparing $N \geq 2$ algorithms, we can compare any two algorithms with each other
- N Algorithms $\Rightarrow k = N(N-1)/2$ statistical tests
- k tests and each with error proability $\alpha \Longrightarrow$ total probability E to make at least one error $E = 1 - ((1-\alpha)^k)$
- Correction needed

Compare $N \geq 2$ Algorithms

- For comparing $N \geq 2$ algorithms, we can compare any two algorithms with each other
- N Algorithms $\Rightarrow k = N(N-1)/2$ statistical tests
- k tests and each with error proability $\alpha \Longrightarrow$ total probability E to make at least one error $E = 1 - ((1-\alpha)^k)$
- Correction needed: Bonferroni correction[20]: Use $\alpha' = \alpha/k$ as significance level instead of α, then the overall probability E to make an error will remain $E \leq \alpha$.

Compare $N \geq 2$ Algorithms



Testing is Not Enough

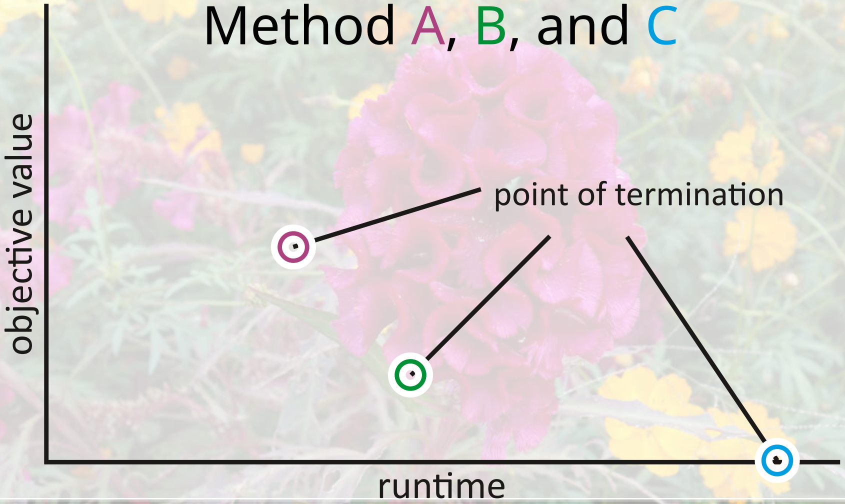
The question of termination

- Literature usually reports tuples "(instance, result, runtime)"

The question of termination

- Literature usually reports tuples "(instance, result, runtime)"
- Papers often use different termination criteria

The question of termination



The question of termination

- Literature usually reports tuples "(instance, result, runtime)"
- Problem: Papers often use different termination criteria

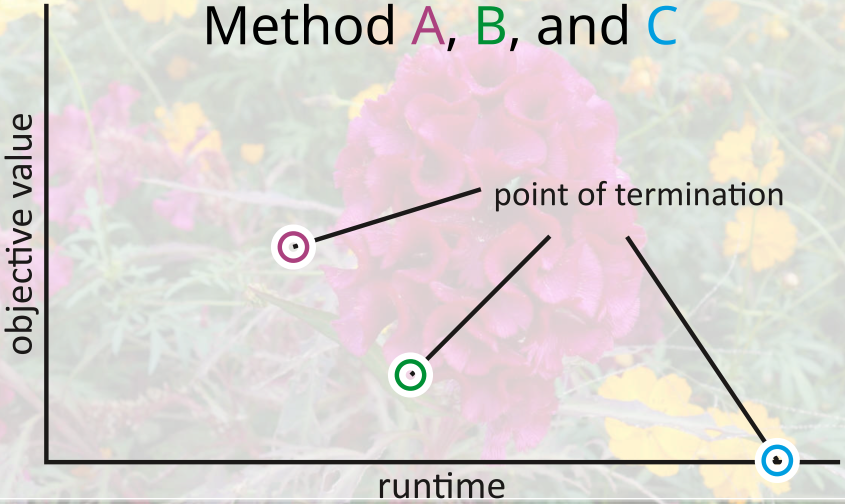
The question of termination

- Literature usually reports tuples "(instance, result, runtime)"
- Problem: Papers often use different termination criteria
- Anytime Algorithms[10]

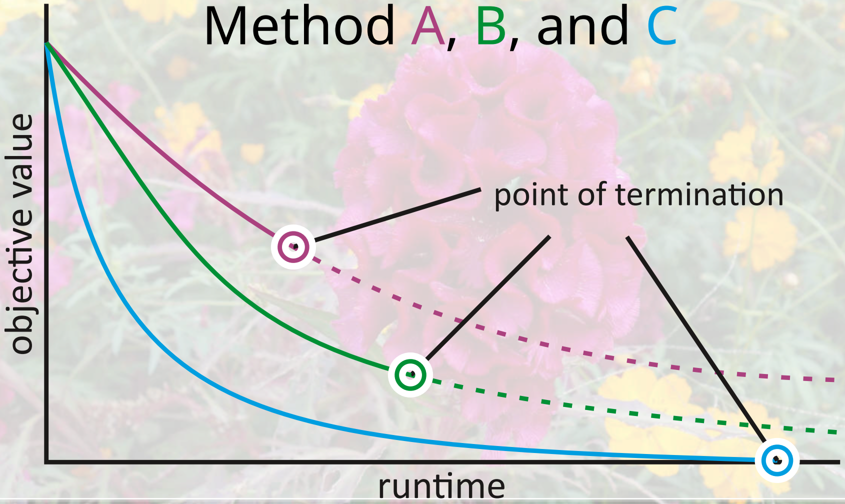
The question of termination

- Literature usually reports tuples "(instance, result, runtime)"
- Problem: Papers often use different termination criteria
- Anytime Algorithms[10]: Always have approximate solution, refine it iteratively

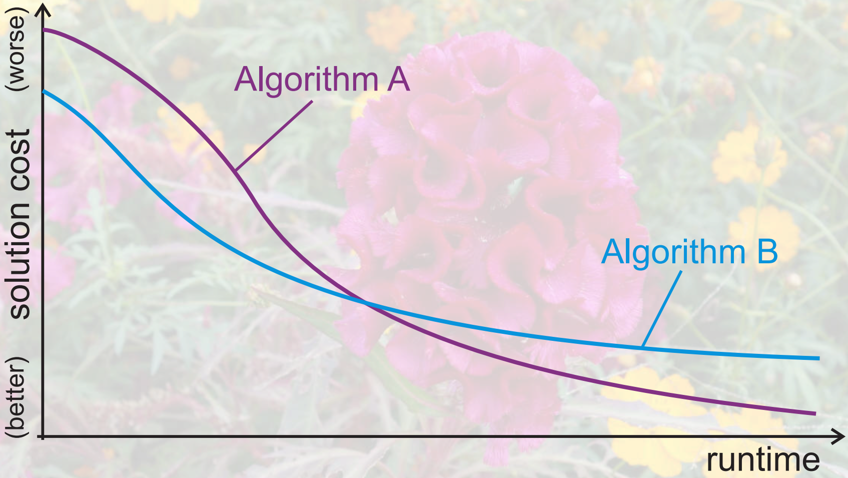
The question of termination



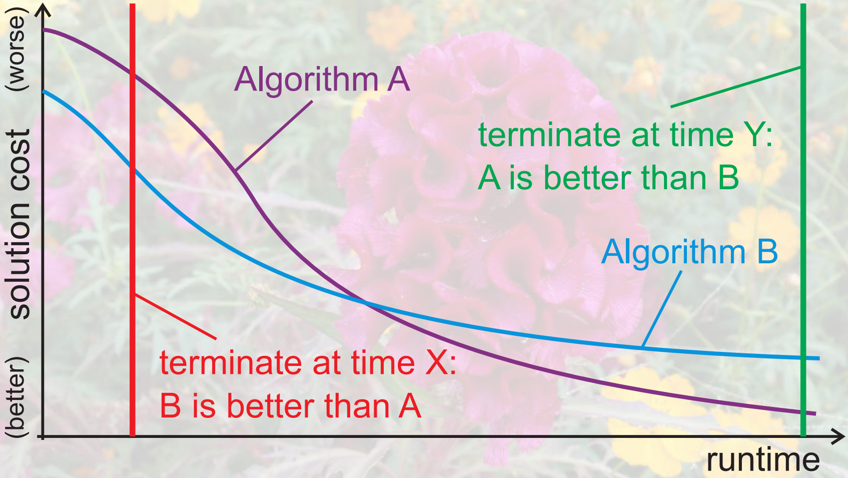
The question of termination



The question of termination



The question of termination



The question of termination

- Literature usually reports tuples "(instance, result, runtime)"
- Problem: Papers often use different termination criteria
- Anytime Algorithms[10]: Always have approximate solution, refine it iteratively
- One measure point per run or instance does not tell the whole story!

The question of termination

- Literature usually reports tuples "(instance, result, runtime)"
- Problem: Papers often use different termination criteria
- Anytime Algorithms[10]: Always have approximate solution, refine it iteratively
- One measure point per run or instance does not tell the whole story!
- Using statistical tests cannot solve this issue (still: at one point in time).

The question of termination

- Literature usually reports tuples "(instance, result, runtime)"
- Problem: Papers often use different termination criteria
- Anytime Algorithms[10]: Always have approximate solution, refine it iteratively
- One measure point per run or instance does not tell the whole story!
- Using statistical tests cannot solve this issue (still: at one point in time).
- We should have the "whole performance curves!"

The question of termination

- Literature usually reports tuples "(instance, result, runtime)"
- Problem: Papers often use different termination criteria
- Anytime Algorithms[10]: Always have approximate solution, refine it iteratively
- One measure point per run or instance does not tell the whole story!
- Using statistical tests cannot solve this issue (still: at one point in time).
- We should have the "whole performance curves!" . . . ideally mean or median curves over several runs!

Other Stuff

New Algorithms and Problems

- There are many papers that introduce two things at the same time: a new optimization problem and a new algorithm.

New Algorithms and Problems

- There are many papers that introduce two things at the same time: a new optimization problem and a new algorithm. I think this is not a good idea.

New Algorithms and Problems

- There are many papers that introduce two things at the same time: a new optimization problem and a new algorithm. I think this is not a good idea.
- If we introduce a new optimization algorithm, we should test it on well-known, well-established benchmark problems.

New Algorithms and Problems

- There are many papers that introduce two things at the same time: a new optimization problem and a new algorithm. I think this is not a good idea.
- If we introduce a new optimization algorithm, we should test it on well-known, well-established benchmark problems. For such problems, results from other well-known and well-established algorithms exist – so we can compare our algorithm to them and investigate its performance objectively.

New Algorithms and Problems

- There are many papers that introduce two things at the same time: a new optimization problem and a new algorithm. I think this is not a good idea.
- If we introduce a new optimization algorithm, we should test it on well-known, well-established benchmark problems. For such problems, results from other well-known and well-established algorithms exist – so we can compare our algorithm to them and investigate its performance objectively.
- If we introduce a new optimization problem, we should apply well-known and well-established algorithms to it.

New Algorithms and Problems

- There are many papers that introduce two things at the same time: a new optimization problem and a new algorithm. I think this is not a good idea.
- If we introduce a new optimization algorithm, we should test it on well-known, well-established benchmark problems. For such problems, results from other well-known and well-established algorithms exist – so we can compare our algorithm to them and investigate its performance objectively.
- If we introduce a new optimization problem, we should apply well-known and well-established algorithms to it. This way we get proper baseline results and can understand whether the problem is hard for the state-of-the-art and/or how far this state-of-the-art allows us to go.

New Algorithms and Problems

- There are many papers that introduce two things at the same time: a new optimization problem and a new algorithm. I think this is not a good idea.
- If we introduce a new optimization algorithm, we should test it on well-known, well-established benchmark problems. For such problems, results from other well-known and well-established algorithms exist – so we can compare our algorithm to them and investigate its performance objectively.
- If we introduce a new optimization problem, we should apply well-known and well-established algorithms to it. This way we get proper baseline results and can understand whether the problem is hard for the state-of-the-art and/or how far this state-of-the-art allows us to go.
- If we have two "moving parts" at the same time, it is hard to understand whether an algorithm is good and whether a problem is hard.

New Algorithms and Problems

- There are many papers that introduce two things at the same time: a new optimization problem and a new algorithm. I think this is not a good idea.
- If we introduce a new optimization algorithm, we should test it on well-known, well-established benchmark problems. For such problems, results from other well-known and well-established algorithms exist – so we can compare our algorithm to them and investigate its performance objectively.
- If we introduce a new optimization problem, we should apply well-known and well-established algorithms to it. This way we get proper baseline results and can understand whether the problem is hard for the state-of-the-art and/or how far this state-of-the-art allows us to go.
- If we have two "moving parts" at the same time, it is hard to understand whether an algorithm is good and whether a problem is hard.
- If you have an own new algorithm on a new problem and use other algorithms for comparision, you might be tempted to just use the most basic configurations of these algorithms.

New Algorithms and Problems

- There are many papers that introduce two things at the same time: a new optimization problem and a new algorithm. I think this is not a good idea.
- If we introduce a new optimization algorithm, we should test it on well-known, well-established benchmark problems. For such problems, results from other well-known and well-established algorithms exist – so we can compare our algorithm to them and investigate its performance objectively.
- If we introduce a new optimization problem, we should apply well-known and well-established algorithms to it. This way we get proper baseline results and can understand whether the problem is hard for the state-of-the-art and/or how far this state-of-the-art allows us to go.
- If we have two "moving parts" at the same time, it is hard to understand whether an algorithm is good and whether a problem is hard.
- If you have an own new algorithm on a new problem and use other algorithms for comparision, you might be tempted to just use the most basic configurations of these algorithms. Then your algorithm might look good, while it actually is not.

New Algorithms and Problems

- There are many papers that introduce two things at the same time: a new optimization problem and a new algorithm. I think this is not a good idea.
- If we introduce a new optimization algorithm, we should test it on well-known, well-established benchmark problems. For such problems, results from other well-known and well-established algorithms exist – so we can compare our algorithm to them and investigate its performance objectively.
- If we introduce a new optimization problem, we should apply well-known and well-established algorithms to it. This way we get proper baseline results and can understand whether the problem is hard for the state-of-the-art and/or how far this state-of-the-art allows us to go.
- If we have two "moving parts" at the same time, it is hard to understand whether an algorithm is good and whether a problem is hard.
- If you have an own new algorithm on a new problem and use other algorithms for comparision, you might be tempted to just use the most basic configurations of these algorithms. Then your algorithm might look good, while it actually is not.
- Know the standard benchmark instances for your field!

Reproducibility

- Your experiments should be well-documented and reproducible.

Reproducibility

- Your experiments should be well-documented and reproducible.
- In the ideal case, someone else can run your code and get the same results.

Reproducibility

- Your experiments should be well-documented and reproducible.
- In the ideal case, someone else can run your code and get the same results.
- For this purpose, you should make your code available, e.g., put it on GitHub or zenodo.org.

Reproducibility

- Your experiments should be well-documented and reproducible.
- In the ideal case, someone else can run your code and get the same results.
- For this purpose, you should make your code available, e.g., put it on GitHub or zenodo.org.
- If your experiments are time-consuming, also make sure to properly store all your results in human- and machine-readable form, ideally in a comma-separated values (CSV) format.

Reproducibility

- Your experiments should be well-documented and reproducible.
- In the ideal case, someone else can run your code and get the same results.
- For this purpose, you should make your code available, e.g., put it on GitHub or zenodo.org.
- If your experiments are time-consuming, also make sure to properly store all your results in human- and machine-readable form, ideally in a comma-separated values (CSV) format.
- You should make an archive such that a) I can directly run the same experiments that you did and b) also have all the data and tools to create the same statistics and figures.

Reproducibility

- Your experiments should be well-documented and reproducible.
- In the ideal case, someone else can run your code and get the same results.
- For this purpose, you should make your code available, e.g., put it on GitHub or zenodo.org.
- If your experiments are time-consuming, also make sure to properly store all your results in human- and machine-readable form, ideally in a comma-separated values (CSV) format.
- You should make an archive such that a) I can directly run the same experiments that you did and b) also have all the data and tools to create the same statistics and figures.
- But what if someone finds an error in work?

Reproducibility

- Your experiments should be well-documented and reproducible.
- In the ideal case, someone else can run your code and get the same results.
- For this purpose, you should make your code available, e.g., put it on GitHub or zenodo.org.
- If your experiments are time-consuming, also make sure to properly store all your results in human- and machine-readable form, ideally in a comma-separated values (CSV) format.
- You should make an archive such that a) I can directly run the same experiments that you did and b) also have all the data and tools to create the same statistics and figures.
- But what if someone finds an error in work?
- That is OK.

Reproducibility

- Your experiments should be well-documented and reproducible.
- In the ideal case, someone else can run your code and get the same results.
- For this purpose, you should make your code available, e.g., put it on GitHub or zenodo.org.
- If your experiments are time-consuming, also make sure to properly store all your results in human- and machine-readable form, ideally in a comma-separated values (CSV) format.
- You should make an archive such that a) I can directly run the same experiments that you did and b) also have all the data and tools to create the same statistics and figures.
- But what if someone finds an error in work?
- That is OK.
- Better they find it in your code that you voluntarily provided than after going through significant re-implementation effort. . .

Cheating

What are typical bad / cheating behavior in research on optimization?

Cheating

What are typical bad / cheating behavior in research on optimization?

- Cherry-Picking

Cheating

What are typical bad / cheating behavior in research on optimization?

- Cherry-Picking:
  - On a benchmark instance, many runs are conducted with different random seeds.

Cheating

What are typical bad / cheating behavior in research on optimization?

- Cherry-Picking:
  - On a benchmark instance, many runs are conducted with different random seeds. But only the 10 with the best results are reported.

Cheating

What are typical bad / cheating behavior in research on optimization?

- Cherry-Picking:
  - On a benchmark instance, many runs are conducted with different random seeds. But only the 10 with the best results are reported. This can be prevented by generating the sequence of random seeds with a deterministic algorithm and reporting both[64].

Cheating

What are typical bad / cheating behavior in research on optimization?

- Cherry-Picking:
  - On a benchmark instance, many runs are conducted with different random seeds. But only the 10 with the best results are reported. This can be prevented by generating the sequence of random seeds with a deterministic algorithm and reporting both[64].
  - Only the benchmark instances where the algorithm performs well are chosen.

Cheating

What are typical bad / cheating behavior in research on optimization?

- Cherry-Picking:
  - On a benchmark instance, many runs are conducted with different random seeds. But only the 10 with the best results are reported. This can be prevented by generating the sequence of random seeds with a deterministic algorithm and reporting both[64].
  - Only the benchmark instances where the algorithm performs well are chosen. Be wary of statements such as "We now present the results of our algorithm on 10 of the TSPLib instances." (TSPLib has more than 100...)

Cheating

What are typical bad / cheating behavior in research on optimization?

- Cherry-Picking:
  - On a benchmark instance, many runs are conducted with different random seeds. But only the 10 with the best results are reported. This can be prevented by generating the sequence of random seeds with a deterministic algorithm and reporting both[64].
  - Only the benchmark instances where the algorithm performs well are chosen. Be wary of statements such as "We now present the results of our algorithm on 10 of the TSPLib instances." (TSPLib has more than 100...)
  - Weak algorithms are chosen for comparison.

Cheating

What are typical bad / cheating behavior in research on optimization?

- Cherry-Picking:
  - On a benchmark instance, many runs are conducted with different random seeds. But only the 10 with the best results are reported. This can be prevented by generating the sequence of random seeds with a deterministic algorithm and reporting both[64].
  - Only the benchmark instances where the algorithm performs well are chosen. Be wary of statements such as "We now present the results of our algorithm on 10 of the TSPLib instances." (TSPLib has more than 100...)
  - Weak algorithms are chosen for comparison. Comparison must always be done with the state-of-the-art on the specific problem at hand.

Cheating

What are typical bad / cheating behavior in research on optimization?

- Cherry-Picking:
  - On a benchmark instance, many runs are conducted with different random seeds. But only the 10 with the best results are reported. This can be prevented by generating the sequence of random seeds with a deterministic algorithm and reporting both[64].
  - Only the benchmark instances where the algorithm performs well are chosen. Be wary of statements such as "We now present the results of our algorithm on 10 of the TSPLib instances." (TSPLib has more than 100...)
  - Weak algorithms are chosen for comparison. Comparison must always be done with the state-of-the-art on the specific problem at hand. Be wary of statements such as "We compare our algorithm with the standard Genetic Algorithm." (because the SGA is usually not the state-of-the-art)

Cheating

What are typical bad / cheating behavior in research on optimization?

- Cherry-Picking
- Sometimes, results may be straight up fabricated.

Cheating

What are typical bad / cheating behavior in research on optimization?

- Cherry-Picking
- Sometimes, results may be straight up fabricated. Algorithm must be clearly specified and ideally the source code is available to prevent this.

Cheating

What are typical bad / cheating behavior in research on optimization?

- Cherry-Picking
- Sometimes, results may be straight up fabricated.
- Misleading statistics are reported (see our discussion on normalization).

Cheating

What are typical bad / cheating behavior in research on optimization?

- Cherry-Picking
- Sometimes, results may be straight up fabricated.
- Misleading statistics are reported
- Uneven configuration effort: Much effort is spent on configuring the own algorithm, the algorithms used for comparison are used with bad settings.

Cheating

What are typical bad / cheating behavior in research on optimization?

- Cherry-Picking
- Sometimes, results may be straight up fabricated.
- Misleading statistics are reported
- Uneven configuration effort.
- Incomparable results are reported (see our discussion on why testing is not enough).

Cheating

What are typical bad / cheating behavior in research on optimization?

- Cherry-Picking
- Sometimes, results may be straight up fabricated.
- Misleading statistics are reported
- Uneven configuration effort.
- Incomparable results are reported.
- Misleading significance in test results (high α, many runs, no corrections).

Cheating

What are typical bad / cheating behavior in research on optimization?

- Cherry-Picking
- Sometimes, results may be straight up fabricated.
- Misleading statistics are reported
- Uneven configuration effort.
- Incomparable results are reported.
- Misleading significance in test results (high α, many runs, no corrections).

Reproducibility prevents cheating and misunderstandings!

Summary

Summary

- The optimization algorithms we consider in this lecture are randomized.

Summary

- The optimization algorithms we consider in this lecture are randomized.
- Comparing them must be done in a statistical way using data from multiple runs

Summary

- The optimization algorithms we consider in this lecture are randomized.
- Comparing them must be done in a statistical way using data from multiple runs
- Two views on performance

Summary

- The optimization algorithms we consider in this lecture are randomized.
- Comparing them must be done in a statistical way using data from multiple runs
- Two views on performance:
    1. best result after fixed number of FEs/runtime

Summary

- The optimization algorithms we consider in this lecture are randomized.
- Comparing them must be done in a statistical way using data from multiple runs
- Two views on performance:
    1. best result after fixed number of FEs/runtime
    2. number of FEs/runtime needed to get certain result

Summary

- The optimization algorithms we consider in this lecture are randomized.
- Comparing them must be done in a statistical way using data from multiple runs
- Two views on performance:
  1. best result after fixed number of FEs/runtime
  2. number of FEs/runtime needed to get certain result
- For every single algorithm/configuration, compute

Summary

- The optimization algorithms we consider in this lecture are randomized.
- Comparing them must be done in a statistical way using data from multiple runs
- Two views on performance:
  1. best result after fixed number of FEs/runtime
  2. number of FEs/runtime needed to get certain result
- For every single algorithm/configuration, compute:
  1. arithmetic and geometric mean and median of key performance indicators

Summary

- The optimization algorithms we consider in this lecture are randomized.
- Comparing them must be done in a statistical way using data from multiple runs
- Two views on performance:
  1. best result after fixed number of FEs/runtime
  2. number of FEs/runtime needed to get certain result
- For every single algorithm/configuration, compute:
  1. arithmetic and geometric mean and median of key performance indicators
  2. quartiles or top/bottom 1% quantile to get a feeling for the usual range of values

Summary

- The optimization algorithms we consider in this lecture are randomized.
- Comparing them must be done in a statistical way using data from multiple runs
- Two views on performance:
  1. best result after fixed number of FEs/runtime
  2. number of FEs/runtime needed to get certain result
- For every single algorithm/configuration, compute:
  1. arithmetic and geometric mean and median of key performance indicators
  2. quartiles or top/bottom 1% quantile to get a feeling for the usual range of values
  3. don't trust just arithmetic mean or standard deviation alone

Summary

- The optimization algorithms we consider in this lecture are randomized.
- Comparing them must be done in a statistical way using data from multiple runs
- Two views on performance:
  1. best result after fixed number of FEs/runtime
  2. number of FEs/runtime needed to get certain result
- For every single algorithm/configuration, compute:
  1. arithmetic and geometric mean and median of key performance indicators
  2. quartiles or top/bottom 1% quantile to get a feeling for the usual range of values
  3. don't trust just arithmetic mean or standard deviation alone
  4. geometric mean if the data is normalized

Summary

- The optimization algorithms we consider in this lecture are randomized.
- Comparing them must be done in a statistical way using data from multiple runs
- Two views on performance:
  1. best result after fixed number of FEs/runtime
  2. number of FEs/runtime needed to get certain result
- For every single algorithm/configuration, compute:
  1. arithmetic and geometric mean and median of key performance indicators
  2. quartiles or top/bottom 1% quantile to get a feeling for the usual range of values
  3. don't trust just arithmetic mean or standard deviation alone
  4. geometric mean if the data is normalized
- Use non-parametric statistical tests with corrections for multiple comparisons.

Summary

- The optimization algorithms we consider in this lecture are randomized.
- Comparing them must be done in a statistical way using data from multiple runs
- Two views on performance:
  1. best result after fixed number of FEs/runtime
  2. number of FEs/runtime needed to get certain result
- For every single algorithm/configuration, compute:
  1. arithmetic and geometric mean and median of key performance indicators
  2. quartiles or top/bottom 1% quantile to get a feeling for the usual range of values
  3. don't trust just arithmetic mean or standard deviation alone
  4. geometric mean if the data is normalized
- Use non-parametric statistical tests with corrections for multiple comparisons.
- Do not only collect one data sample per run, try to plot progress curves.

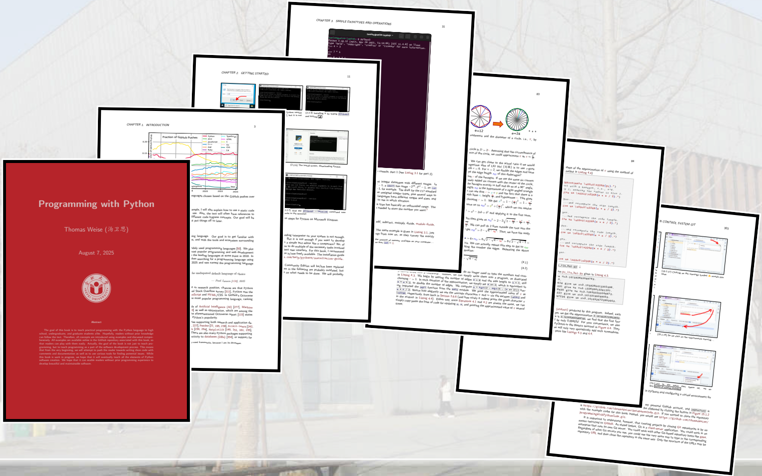
Summary

- The optimization algorithms we consider in this lecture are randomized.
- Comparing them must be done in a statistical way using data from multiple runs
- Two views on performance:
  1. best result after fixed number of FEs/runtime
  2. number of FEs/runtime needed to get certain result
- For every single algorithm/configuration, compute:
  1. arithmetic and geometric mean and median of key performance indicators
  2. quartiles or top/bottom 1% quantile to get a feeling for the usual range of values
  3. don't trust just arithmetic mean or standard deviation alone
  4. geometric mean if the data is normalized
- Use non-parametric statistical tests with corrections for multiple comparisons.
- Do not only collect one data sample per run, try to plot progress curves.
- Use well-known benchmarks, provide your source code!

Advertisement

Programming with Python

We have a freely available course book on *Programming with Python* at https://thomasweise.github.io/programmingWithPython, with focus on practical software development using the Python ecosystem of tools[60].

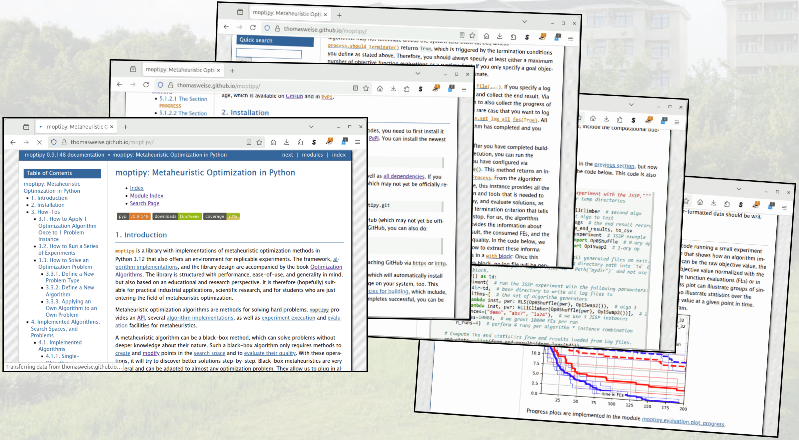


Databases

We have a freely available course book on *Databases* at https://thomasweise.github.io/databases, with actual practical examples using a real database management system (DBMS)[58].

Metaheuristic Optimization in Python: moptipy

We offer moptipy[64] a mature open source Python package for metaheuristic optimization, which implements several algorithms, can run self-documenting experiments in parallel and in a distributed fashion, and offers statistical evaluation tools.



谢谢您门！

Thank you!

Vielen Dank!

References I

[1] Scott Aaronson. "The Limits of Quantum Computers". *Scientific American* 298(3):62–69, Mar. 2008. New York, NY, USA: Scientific American, Inc., a division of Springer Nature Limited. ISSN: 1946-7087. doi:10.1038/scientificamerican0308-62. URL: http://www.cs.virginia.edu/~robins/The_Limits_of_Quantum_Computers.pdf (visited on 2021-08-10) (cit. on pp. 12–14).

[2] David Lee Applegate, Robert E. Bixby, Vašek Chvátal, and William John Cook. *The Traveling Salesman Problem: A Computational Study*. 2nd ed. Vol. 17. Princeton Series in Applied Mathematics. Princeton, NJ, USA: Princeton University Press, 2007. ISBN: 978-0-691-12993-8 (cit. on pp. 10–33, 474).

[3] Anne Auger and Nikolaus Hansen. "Performance Evaluation of an Advanced Local Search Evolutionary Algorithm". In: Sept. 2–4, 2005, Edinburgh, Scotland, UK. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE), 2005, pp. 1777–1784. ISBN: 978-0-7803-9363-9. doi:10.1109/CEC.2005.1554903. URL: http://www.cmap.polytechnique.fr/~nikolaus.hansen/cec2005localcmaes.pdf (visited on 2025-07-25) (cit. on pp. 88–93).

[4] Paul Gustav Heinrich Bachmann. *Die Analytische Zahlentheorie / Dargestellt von Paul Bachmann*. Vol. Zweiter Theil of Zahlentheorie: Versuch einer Gesamtdarstellung dieser Wissenschaft in ihren Haupttheilen. Leipzig, Sachsen, Germany: B. G. Teubner, 1894. ISBN: 978-1-4181-6963-3. URL: http://gallica.bnf.fr/ark:/12148/bpt6k994750 (visited on 2023-12-13) (cit. on p. 476).

[5] Thomas Bäck, David B. Fogel, and Zbigniew "Zbyszek" Michalewicz, eds. *Handbook of Evolutionary Computation*. Bristol, England, UK: IOP Publishing Ltd and Oxford, Oxfordshire, England, UK: Oxford University Press, 1997. ISBN: 978-0-7503-0392-7 (cit. on pp. 66–75, 473).

[6] Thomas Bartz-Beielstein, Carola Doerr, Daan van den Berg, Jakob Bossek, Sowmya Chandrasekaran, Tome Eftimov, Andreas Fischbach, Pascal Kerschke, William La Cava, Manuel López-Ibáñez, Katherine Mary Malan, Jason Hall Moore, Boris Naujoks, Patryk Orzechowski, Vanessa Volz, Markus Wagner, and Thomas Weise (汤卫思). "Benchmarking in Optimization: Best Practice and Open Issues". (abs/2007.03488), Dec. 18, 2020. doi:10.48550/arXiv.2007.03488. URL: https://arxiv.org/abs/2007.03488 (visited on 2025-07-25). arXiv:2007.03488v2 [cs.NE] 16 Dec 2020 (cit. on pp. 4–9).

[7] Daniel F. Bauer. "Constructing Confidence Sets Using Rank Statistics". *Journal of the American Statistical Association (J AM STAT ASSOC)* 67(339):687–690, Sept. 1972. London, England, UK: Taylor and Francis Ltd. for Alexandria, VA, USA: American Statistical Association (ASA). ISSN: 0162-1459. doi:10.1080/01621459.1972.10481279 (cit. on pp. 373–378).

References II

[8] Bernd Bischl, Pascal Kerschke, Lars Kotthoff, Thomas Marius Lindauer, Yuri Malitsky, Alexandre Fréchette, Holger H. Hoos, Frank Hutter, Kevin Leyton-Brown, Kevin Tierney, and Joaquin Vanschoren. "ASlib: A Benchmark Library for Algorithm Selection". 237:41–58, Aug. 2016. doi:10.1016/j.artint.2016.04.003 (cit. on pp. 88–93).

[9] Jacek Błażewicz, Wolfgang Domschke, and Erwin Pesch. "The Job Shop Scheduling Problem: Conventional and New Solution Techniques". *European Journal of Operational Research* 93(1):1–33, Aug. 1996. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 0377-2217. doi:10.1016/0377-2217(95)00362-2 (cit. on p. 474).

[10] Mark S. Boddy and Thomas L. Dean. *Solving Time-Dependent Planning Problems*. Tech. rep. CS-89-03. Providence, RI, USA: Brown University, Department of Computer Science, Feb. 1989. URL: ftp://ftp.cs.brown.edu/pub/techreports/89/cs89-03.pdf (visited on 2025-07-25) (cit. on pp. 395, 396, 398–400, 405–408).

[11] Jürgen Bortz, Gustav Adolf Lienert, and Klaus Boehnke. *Verteilungsfreie Methoden in der Biostatistik*. 3rd ed. Springer-Lehrbuch. Berlin/Heidelberg, Germany: Springer-Verlag GmbH Germany, 2008. ISSN: 0937-7433. ISBN: 978-3-445-11034-3. doi:10.1007/978-3-540-74707-9 (cit. on pp. 356–361, 373–378).

[12] Shaun Burke. "Missing Values, Outliers, Robust Statistics & Non-Parametric Methods". *LC·GC Europe* 1(2):19–24, Jan. 2001. Cranbury, NJ, USA: MJH Life Sciences. ISSN: 1471-6577. URL: https://www.researchgate.net/profile/Marcel-Severijnen/post/Source-apportionment-of-atmospheric-Hg/attachment/5c3cbf323843b0067550831f/AS%3A715017976557570%401547484978416/download/Missing+Values%2C+Outliers%2C+Robust+Statistics+%26+Non-parametric+Methods+LCGC+Eur+Burke+2001+-+4+de+4.pdf (visited on 2025-07-25). Online Supplement (cit. on pp. 366–371).

[13] Noureddine Chabini and Rachid Beguenane. "FPGA-Based Designs of the Factorial Function". In: *IEEE Canadian Conference on Electrical and Computer Engineering (CCECE'2022)*. Sept. 18–20, 2022, Halifax, NS, Canada. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE), 2022, pp. 16–20. ISBN: 978-1-6654-8432-9. doi:10.1109/CCECE49351.2022.9918302 (cit. on p. 475).

[14] Bo Chen, Chris N. Potts, and Gerhard J. Woeginger. "A Review of Machine Scheduling: Complexity, Algorithms and Approximability". In: *Handbook of Combinatorial Optimization*. Ed. by Panos Miltiades Pardalos, Ding-Zhu Du, and Ronald Lewis Graham. 1st ed. Boston, MA, USA: Springer, 1998, pp. 1493–1641. ISBN: 978-1-4613-7987-4. doi:10.1007/978-1-4613-0303-9_25. See also pages 21–169 in volume 3/3 by Norwell, MA, USA: Kluwer Academic Publishers. (Cit. on pp. 12–14, 474, 476).

References III

[15] Stephen Arthur Cook. "The Complexity of Theorem-Proving Procedures". In: *Third Annual ACM Symposium on Theory of Computing (STOC'1971)*. May 3–5, 1971, Shaker Heights, OH, USA. Ed. by Michael A. Harrison, Ranan B. Banerji, and Jeffrey D. Ullman. New York, NY, USA: Association for Computing Machinery (ACM), 1971, pp. 151–158. ISBN: 978-1-4503-7464-4. doi:10.1145/800157.805047 (cit. on pp. 12–14, 474, 476).

[16] "`csv` – CSV File Reading and Writing". In: *Python 3 Documentation. The Python Standard Library*. Beaverton, OR, USA: Python Software Foundation (PSF), 2001–2025. URL: https://docs.python.org/3/library/csv.html (visited on 2024-11-14) (cit. on p. 473).

[17] Marco Dorigo, Mauro Birattari, and Thomas Stützle. "Ant Colony Optimization". *IEEE Computational Intelligence Magazine (CIM)* 1:28–39, Nov. 2006. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE). ISSN: 1556-603X. doi:10.1109/MCI.2006.329691. URL: https://www.researchgate.net/publication/308953674 (visited on 2025-07-25) (cit. on p. 473).

[18] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. "Ant System: Optimization by a Colony of Cooperating Agents". *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics* 26(1):29–41, Feb. 1996. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE). ISSN: 1083-4419. doi:10.1109/3477.484436. URL: https://www.researchgate.net/publication/5589170 (visited on 2025-07-26) (cit. on p. 473).

[19] Marco Dorigo and Thomas Stützle. *Ant Colony Optimization*. Cambridge, MA, USA: MIT Press, 2004. ISBN: 978-0-262-04219-2 (cit. on p. 473).

[20] Olive Jean Dunn. "Multiple Comparisons Among Means". *Journal of the American Statistical Association (J AM STAT ASSOC)* 56(293):52–64, Mar. 1961. London, England, UK: Taylor and Francis Ltd. for Alexandria, VA, USA: American Statistical Association (ASA). ISSN: 0162-1459. doi:10.1080/01621459.1961.10482090 (cit. on pp. 389, 391, 392).

[21] Jacques Dutka. "The Early History of the Factorial Function". *Archive for History of Exact Sciences* 43(3):225–249, Sept. 1991. Berlin/Heidelberg, Germany: Springer-Verlag GmbH Germany. ISSN: 0003-9519. doi:10.1007/BF00389433. Communicated by Umberto Bottazzini (cit. on p. 475).

[22] Eugene S. Edgington. *Randomization Tests*. 3rd ed. Boca Raton, FL, USA: CRC Press, Inc., 1995. ISBN: 978-0-8247-9669-3 (cit. on pp. 356–361, 373–378).

References IV

[23] Sir Ronald Aylmer Fisher. "On the Interpretation of χ^2 from Contingency Tables, and the Calculation of P". *Journal of the Royal Statistical Society* 85:87–94, 1922. Chichester, West Sussex, England, UK: Blackwell Publishing Ltd for London, England, UK: Royal Statistical Society (RSS). ISSN: 0035-9238. URL: http://hdl.handle.net/2440/15173 (visited on 2021-08-10) (cit. on pp. 373–378).

[24] Philip J. Fleming and John J. Wallace. "How Not to Lie with Statistics: The Correct Way to Summarize Benchmark Results". *Communications of the ACM (CACM)* 29(3):218–221, Mar. 1986. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 0001-0782. doi:10.1145/5666.5673 (cit. on pp. 186–228).

[25] Michael T. Goodrich. *A Gentle Introduction to NP-Completeness*. Irvine, CA, USA: University of California, Irvine, Apr. 2022. URL: https://ics.uci.edu/~goodrich/teach/cs165/notes/NPComplete.pdf (visited on 2025-08-01) (cit. on p. 476).

[26] Frank E. Grubbs. "Procedures for Detecting Outlying Observations in Samples". *Technometrics* 11(1):1–21, 1969. London, England, UK: Taylor and Francis Ltd. ISSN: 0040-1706. doi:10.1080/00401706.1969.10490657 (cit. on pp. 152–155).

[27] Gregory Z. Gutin and Abraham P. Punnen, eds. *The Traveling Salesman Problem and its Variations*. Vol. 12. Combinatorial Optimization (COOP). New York, NY, USA: Springer New York, May 2002. ISSN: 1388-3011. doi:10.1007/b101971 (cit. on pp. 10–33, 474).

[28] Lorenz Gygax. *Statistik für Nutztierethologen – Einführung in die statistische Denkweise: Was ist, was macht ein statistischer Test?* 1.1. Zürich, Switzerland: Swiss Federal Veterinary Office (FVO), Centre for Proper Housing of Ruminants and Pigs, June 2003. URL: http://www.proximate-biology.ch/documents/introEtho.pdf (visited on 2025-07-25) (cit. on pp. 373–378).

[29] Nikolaus Hansen, Anne Auger, Steffen Finck, and Raymond Ros. *Real-Parameter Black-Box Optimization Benchmarking 2010: Experimental Setup*. Rapport de Recherche RR-7215. inria-00462481. Institut National de Recherche en Informatique et en Automatique (INRIA), Mar. 9, 2010. URL: http://hal.inria.fr/inria-00462481 (visited on 2025-07-25) (cit. on pp. 35–39, 76–83, 88–93).

[30] Nikolaus Hansen, Steffen Finck, and Raymond Ros. *COCO – COmparing Continuous Optimizers: The Documentation*. Tech. rep. RR-0409. inria-00597334. May 2011. URL: http://hal.inria.fr/inria-00597334 (visited on 2025-07-25) (cit. on pp. 35–39).

[31] Myles Hollander and Douglas Alan Wolfe. *Nonparametric Statistical Methods*. Chichester, West Sussex, England, UK: John Wiley and Sons Ltd., 1973. ISBN: 978-0-471-40635-8 (cit. on pp. 373–378).

References V

[32] Holger H. Hoos and Thomas Stützle. "Local Search Algorithms for SAT: An Empirical Evaluation". *Journal of Automated Reasoning* 24(4):421–481, May 2000. London, England, UK: Springer Nature Limited. ISSN: 0168-7433. doi:10.1023/A:1006350622830. URL: https://www.cs.ubc.ca/~hoos/Publ/jar00.pdf (visited on 2025-07-25) (cit. on p. 180).

[33] Holger H. Hoos and Thomas Stützle. *Stochastic Local Search: Foundations & Applications*. Elsevier B.V., 2004. ISBN: 978-1-55860-872-6 (cit. on p. 474).

[34] John Hunt. *A Beginners Guide to Python 3 Programming*. 2nd ed. Undergraduate Topics in Computer Science (UTICS). Cham, Switzerland: Springer, 2023. ISBN: 978-3-031-35121-1. doi:10.1007/978-3-031-35122-8 (cit. on p. 474).

[35] Richard "Dick" Manning Karp. "Reducibility Among Combinatorial Problems". In: *Complexity of Computer Computations. Proceedings of a symposium on the Complexity of Computer Computations, held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department*. Mar. 20–22, 1972. Ed. by Raymond E. "Ray" Miller and James W. "Jim" Thatcher. New York, NY, USA: Springer New York, 1972, pp. 85–103. ISBN: 978-1-4684-2003-6. doi:10.1007/978-1-4684-2001-2_9 (cit. on pp. 12–14).

[36] Pascal Kerschke, Jakob Bossek, and Heike Trautmann. "Parameterization of State-of-the-Art Performance Indicators: A Robustness Study based on Inexact TSP Solvers". In: *Genetic and Evolutionary Computation Conference Companion (GECCO'2018)*. Kyoto, Japan. Ed. by Hernán E. Aguirre and Keiki Takadama. New York, NY, USA: Association for Computing Machinery (ACM), 2018, pp. 1737–1744. ISBN: 978-1-4503-5764-7. doi:10.1145/3205651.3208233 (cit. on pp. 88–93).

[37] Edmund Landau. *Handbuch der Lehre von der Verteilung der Primzahlen*. Leipzig, Sachsen, Germany: B. G. Teubner, 1909. ISBN: 978-0-8218-2650-8 (cit. on p. 476).

[38] Eugene Leighton Lawler, Jan Karel Lenstra, Alexander Hendrik George Rinnooy Kan, and David B. Shmoys. "Sequencing and Scheduling: Algorithms and Complexity". In: *Production Planning and Inventory*. Ed. by Stephen C. Graves, Alexander Hendrik George Rinnooy Kan, and Paul H. Zipkin. Vol. IV of Handbooks of Operations Research and Management Science. Amsterdam, The Netherlands: Elsevier B.V., 1993. Chap. 9, pp. 445–522. ISSN: 0927-0507. ISBN: 978-0-444-87472-6. doi:10.1016/S0927-0507(05)80189-6. URL: http://alexandria.tue.nl/repository/books/339776.pdf (visited on 2023-12-06) (cit. on pp. 12–14, 474, 476).

References VI

[39] Eugene Leighton Lawler, Jan Karel Lenstra, Alexander Hendrik George Rinnooy Kan, and David B. Shmoys. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Estimation, Simulation, and Control – Wiley-Interscience Series in Discrete Mathematics and Optimization. Chichester, West Sussex, England, UK: Wiley Interscience, Sept. 1985. ISSN: 0277-2698. ISBN: 978-0-471-90413-7 (cit. on pp. 10–33, 474).

[40] Kent D. Lee and Steve Hubbard. *Data Structures and Algorithms with Python*. Undergraduate Topics in Computer Science (UTICS). Cham, Switzerland: Springer, 2015. ISBN: 978-3-319-13071-2. doi:10.1007/978-3-319-13072-9 (cit. on p. 474).

[41] Peter Luschny. *A New Kind of Factorial Function*. Highland Park, NJ, USA: The OEIS Foundation Inc., Oct. 4, 2015. URL: https://oeis.org/A000142/a000142.pdf (visited on 2024-09-29) (cit. on p. 475).

[42] Mark Lutz. *Learning Python*. 6th ed. Sebastopol, CA, USA: O'Reilly Media, Inc., Mar. 2025. ISBN: 978-1-0981-7130-8 (cit. on p. 474).

[43] Gangadharrao Soundalyarao Maddala. *Introduction to Econometrics*. 2nd ed. New York, NY, USA: MacMillan, 1992. ISBN: 978-0-02-374545-4 (cit. on pp. 152–155).

[44] Henry B. Mann and Donald R. Whitney. "On a Test of whether One of Two Random Variables is Stochastically Larger than the Other". *The Annals of Mathematical Statistics (AOMS)* 18(1):50–60, Mar. 1947. Beachwood, OH, USA: Institute of Mathematical Statistics. ISSN: 0003-4851. doi:10.1214/aoms/1177730491. URL: http://projecteuclid.org/euclid.aoms/1177730491 (visited on 2025-07-25) (cit. on pp. 373–378).

[45] Ashwin Pajankar. *Python Unit Test Automation: Automate, Organize, and Execute Unit Tests in Python*. New York, NY, USA: Apress Media, LLC, Dec. 2021. ISBN: 978-1-4842-7854-3 (cit. on pp. 166–181, 475).

[46] Yasset Pérez-Riverol, Laurent Gatto, Rui Wang, Timo Sachsenberg, Julian Uszkoreit, Felipe da Veiga Leprevost, Christian Fufezan, Tobias Ternent, Stephen J. Eglen, Daniel S. Katz, Tom J. Pollard, Alexander Konovalov, Robert M. Flight, Kai Blin, and Juan Antonio Vizcaíno. "Ten Simple Rules for Taking Advantage of Git and GitHub". *PLOS Computational Biology* 12(7), July 14, 2016. San Francisco, CA, USA: Public Library of Science (PLOS). ISSN: 1553-7358. doi:10.1371/JOURNAL.PCBI.1004947 (cit. on p. 474).

References VII

[47] Kenneth V. Price. "Differential Evolution vs. The Functions of the 2nd ICEO". In: *IEEE International Conference on Evolutionary Computation*. Indianapolis, IN, USA. Ed. by Russell C. "Russ" Eberhart, Peter J. Angeline, Thomas Bäck, Zbigniew "Zbyszek" Michalewicz, and Xin Yao (姚新). Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE), 1997, pp. 153–157. ISBN: 978-0-7803-3949-1. doi:10.1109/ICEC.1997.592287 (cit. on pp. 88–93).

[48] Sanatan Rai and George Vairaktarakis. "NP-Complete Problems and Proof Methodology". In: *Encyclopedia of Optimization*. Ed. by Christodoulos A. Floudas and Panos Miltiades Pardalos. 2nd ed. Boston, MA, USA: Springer, Sept. 2008, pp. 2675–2682. ISBN: 978-0-387-74758-3. doi:10.1007/978-0-387-74759-0_462 (cit. on p. 476).

[49] Gerhard Reinelt. "TSPLIB – A Traveling Salesman Problem Library". *ORSA Journal on Computing* 3(4):376–384, Aut. 1991. Catonsville, MD, USA: The Institute for Operations Research and the Management Sciences (INFORMS). ISSN: 0899-1499. doi:10.1287/ijoc.3.4.376 (cit. on p. 475).

[50] Gerhard Reinelt. *TSPLIB95*. Tech. rep. Heidelberg, Baden-Württemberg, Germany: Universität Heidelberg, Institut für Angewandte Mathematik, 1995–Apr. 28, 2016. URL: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp95.pdf (visited on 2025-08-01) (cit. on p. 475).

[51] Per Runeson. "A Survey of Unit Testing Practices". *IEEE Software* 23(4):22–29, July–Aug. 2006. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE). ISSN: 0740-7459. doi:10.1109/MS.2006.91 (cit. on pp. 166–181, 475).

[52] Yakov Shafranovich. *Common Format and MIME Type for Comma-Separated Values (CSV) Files*. Request for Comments (RFC) 4180. Wilmington, DE, USA: Internet Engineering Task Force (IETF), Oct. 2005. URL: https://www.ietf.org/rfc/rfc4180.txt (visited on 2025-02-05) (cit. on p. 473).

[53] Sidney Siegel and N. John Castellan Jr. *Nonparametric Statistics for The Behavioral Sciences*. Humanities, Social Sciences, & Language. New York, NY, USA: McGraw-Hill, 1988. ISBN: 978-0-07-057357-4 (cit. on pp. 373–378).

[54] Anna Skoulikari. *Learning Git*. Sebastopol, CA, USA: O'Reilly Media, Inc., May 2023. ISBN: 978-1-0981-3391-7 (cit. on p. 474).

References VIII

[55] Ke Tang (唐珂), Xiaodong Li, Ponnuthurai Nagaratnam Suganthan, Zhenyu Yang (杨振宇), and Thomas Weise (汤卫思). *Benchmark Functions for the CEC'2010 Special Session and Competition on Large-Scale Global Optimization*. Tech. rep. Hefei, Anhui, China (中国安徽省合肥市): University of Science and Technology of China (USTC, 中国科学技术大学), School of Computer Science and Technology (计算机科学与技术学院), Nature Inspired Computation and Applications Laboratory (NICAL), Jan. 8, 2010 (cit. on pp. 94–99).

[56] George K. Thiruvathukal, Konstantin Läufer, and Benjamin Gonzalez. "Unit Testing Considered Useful". *Computing in Science & Engineering* 8(6):76–87, Nov.–Dec. 2006. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE). ISSN: 1521-9615. doi:10.1109/MCSE.2006.124. URL: https://www.researchgate.net/publication/220094077 (visited on 2024-10-01) (cit. on pp. 166–181, 475).

[57] Mariot Tsitoara. *Beginning Git and GitHub: Version Control, Project Management and Teamwork for the New Developer*. New York, NY, USA: Apress Media, LLC, Mar. 2024. ISBN: 979-8-8688-0215-7 (cit. on pp. 474, 475).

[58] Thomas Weise (汤卫思). *Databases*. Hefei, Anhui, China (中国安徽省合肥市): Hefei University (合肥大学), School of Artificial Intelligence and Big Data (人工智能与大数据学院), Institute of Applied Optimization (应用优化研究所, IAO), 2025. URL: https://thomasweise.github.io/databases (visited on 2025-01-05) (cit. on pp. 461, 473).

[59] Thomas Weise (汤卫思). *Global Optimization Algorithms – Theory and Application*. self-published, 2009. URL: https://www.researchgate.net/publication/200622167 (visited on 2025-07-25) (cit. on pp. 66–75, 373–378, 473).

[60] Thomas Weise (汤卫思). *Programming with Python*. Hefei, Anhui, China (中国安徽省合肥市): Hefei University (合肥大学), School of Artificial Intelligence and Big Data (人工智能与大数据学院), Institute of Applied Optimization (应用优化研究所, IAO), 2024–2025. URL: https://thomasweise.github.io/programmingWithPython (visited on 2025-01-05) (cit. on pp. 342, 460, 474).

[61] Thomas Weise (汤卫思), Raymond Chiong, Jörg Lässig, Ke Tang (唐珂), Shigeyoshi Tsutsui, Wenxiang Chen (陈文祥), Zbigniew "Zbyszek" Michalewicz, and Xin Yao (姚新). "Benchmarking Optimization Algorithms: An Open Source Framework for the Traveling Salesman Problem". *IEEE Computational Intelligence Magazine (CIM)* 9(3):40–52, Aug. 2014. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE). ISSN: 1556-603X. doi:10.1109/MCI.2014.2326101 (cit. on pp. 10–33, 53–65, 100–105, 473, 474).

References IX

[62] Thomas Weise (汤卫思), Li Niu (牛力), and Ke Tang (唐珂). "AOAB – Automated Optimization Algorithm Benchmarking". In: *12th Annual Conference Companion on Genetic and Evolutionary Computation (GECCO'2010)*. July 7–11, 2010, Portland, OR, USA. Ed. by Martin Pelikan and Jürgen Branke. New York, NY, USA: Association for Computing Machinery (ACM), 2010, pp. 1479–1486. ISBN: 978-1-4503-0073-5. doi:10.1145/1830761.1830763 (cit. on pp. 35–39, 76–83, 100–105).

[63] Thomas Weise (汤卫思), Xiaofeng Wang (王晓峰), Qi Qi (齐琪), Bin Li (李斌), and Ke Tang (唐珂). "Automatically Discovering Clusters of Algorithm and Problem Instance Behaviors as well as their Causes from Experimental Data, Algorithm Setups, and Instance Features". *Applied Soft Computing (ASOC)* 73:366–382, Dec. 2018. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 1568-4946. doi:10.1016/J.ASOC.2018.08.030 (cit. on pp. 35–39, 100–105).

[64] Thomas Weise (汤卫思) and Zhize Wu (吴志泽). "Replicable Self-Documenting Experiments with Arbitrary Search Spaces and Algorithms". In: *Conference on Genetic and Evolutionary Computation (GECCO'2023), Companion Volume*. July 15–19, 2023, Lisbon, Portugal. Ed. by Sara Silva and Luís Paquete. New York, NY, USA: Association for Computing Machinery (ACM), 2023, pp. 1891–1899. ISBN: 979-8-4007-0120-7. doi:10.1145/3583133.3596306 (cit. on pp. 430–437, 462, 474).

[65] Frank Wilcoxon. "Individual Comparisons by Ranking Methods". *Biometrics Bulletin* 1(6):80–83, Dec. 1945. Washington, D.C., USA: International Biometric Society. ISSN: 0099-4987 (cit. on pp. 373–378).

[66] Kinza Yasar and Craig S. Mullins. *Definition: Database Management System (DBMS)*. Newton, MA, USA: TechTarget, Inc., June 2024. URL: https://www.techtarget.com/searchdatamanagement/definition/database-management-system (visited on 2025-01-11) (cit. on p. 473).

Glossary I

EA An *evolutionary algorithm* is a metaheuristic optimization method that maintains a population of candidate solutions, which undergo selection (where better solutions are chosen with higher probability) and reproduction (where mutation and recombination create a new candidate solution from one or two existing ones, repectively)[5,59].

ACO *Ant Colony Optimization* is a nature-inspired optimization method for combinatorial problems where solutions are generated by "ants" that move from node to node in a graph choosing edges based on (1) the simulated pheromone on the edges and (2) a per-edge heuristic value[17–19]. If an ant produced a good solution, "pheromone" is distributed over the edges it visited, making it more likely to be re-visited by other ants.

BKS The *Best Known Solution* for an instance of an optimization problem is the best solution (measured based on the objective values) that has ever been reported in literature. BKSes are not necessarily globally optimal, as in many instances of $\mathcal{NP}$-hard problems, the true optima are unknown.

CSV *Comma-Separated Values* is a very common and simple text format for exchanging tabular or matrix data[52]. Each row in the text file represents one row in the table or matrix. The elements in the row are separated by a fixed delimiter, usually a comma (","), sometimes a semicolon (";"). Python offers some out-of-the-box CSV support in the `csv` module[16].

DB A *database* is an organized collection of structured information or data, typically stored electronically in a computer system. Databases are discussed in our book *Databases*[58].

DBMS A *database management system* is the software layer located between the user or application and the database (DB). The DBMS allows the user/application to create, read, write, update, delete, and otherwise manipulate the data in the DB[66].

FE *Objective function evaluations* are an implementation-independent measure of runtime for optimization algorithms[61]. 1 FE equals to one evaluated candidate solution during the optimization process.

Glossary II

Git is a distributed Version Control Systems (VCS) which allows multiple users to work on the same code while preserving the history of the code changes[54,57]. Learn more at https://git-scm.com.

GitHub is a website where software projects can be hosted and managed via the Git VCS[46,57]. Learn more at https://github.com.

JSSP The *Job Shop Scheduling Problem*[9,38] is one of the most prominent and well-studied scheduling tasks. In a JSSP instance, there are k machines and m jobs. Each job must be processed once by each machine in a job-specific sequence and has a job-specific processing time on each machine. The goal is to find an assignment of jobs to machines that results in an overall shortest makespan, i.e., the schedule which can complete all the jobs in the shortest time. The JSSP is $\mathcal{NP}$-complete[14,38].

MaxSAT The goal of satisfiablity problems is to find an assignment for n Boolean variables that make a given Boolean formula $F : \{0, 1\}^n \mapsto \{0, 1\}$ become true. In the *Maximum Satisfiability (MaxSAT)* problem[33], F is given in conjunctive normal form, i.e., the variables appear as literals either directly or negated in m "or" clauses, which are all combined into one "and." The objective function $f(x)$, subject to minimization, computes the number of clauses which are false under the variable setting x. If $f(x) = 0$, then all clauses of F are true, which solves the problem. The MaxSat problem is $\mathcal{NP}$-complete[15].

moptipy is the *Metaheuristic Optimization in Python* library[64]. Learn more at https://thomasweise.github.io/moptipy.

Python The Python programming language[34,40,42,60], i.e., what you will learn about in our book[60]. Learn more at https://python.org.

TSP In an instance of the *Traveling Salesperson Problem*, also known as *Traveling Salesman Problem*, a set of n cities or locations as well as the distances between them are defined[2,27,39,61]. The goal is to find the shortest round-trip tour that starts at one city, visits all the other cities one time each, and returns to the origin. The TSP is one of the most well-known $\mathcal{NP}$-hard combinatorial optimization problems.

Glossary III

TSPLib is a library of benchmark instances for the Traveling Salesperson Problem (TSP) available at http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95[49,50].

unit test Software development is centered around creating the program code of an application, library, or otherwise useful system. A *unit test* is an *additional* code fragment that is not part of that productive code. It exists to execute (a part of) the productive code in a certain scenario (e.g., with specific parameters), to observe the behavior of that code, and to compare whether this behavior meets the specification[45,51,56]. If not, the unit test fails. The use of unit tests is at least threefold: First, they help us to detect errors in the code. Second, program code is usually not developed only once and, from then on, used without change indefinitely. Instead, programs are often updated, improved, extended, and maintained over a long time. Unit tests can help us to detect whether such changes in the program code, maybe after years, violate the specification or, maybe, cause another, depending, module of the program to violate its specification. Third, they are part of the documentation or even specification of a program.

VCS A *Version Control System* is a software which allows you to manage and preserve the historical development of your program code[57]. A distributed VCS allows multiple users to work on the same code and upload their changes to the server, which then preserves the change history. The most popular distributed VCS is Git.

$i!$ The factorial $a!$ of a natural number $a \in \mathbb{N}_1$ is the product of all positive natural numbers less than or equal to a, i.e., $a! = 1 * 2 * 3 * 4 * \cdots * (a-1) * a$[13,21,41].

$i..j$ with $i, j \in \mathbb{Z}$ and $i \leq j$ is the set that contains all integer numbers in the inclusive range from i to j. For example, 5..9 is equivalent to $\{5, 6, 7, 8, 9\}$

$\operatorname{geom}(A)$ The *geometric mean* $\operatorname{geom}(A)$ is the n^{th} root of the product of n *positive* values in a dataset $A = (a_0, a_1, \dots, a_{n-1})$ with $a_i > 0$ for all $i \in 0..n$, i.e., $\operatorname{geom}(A) = \sqrt[n]{\prod_{i=0}^{n-1} a_i} = \exp\left(\frac{1}{n}\sum_{i=0}^{n-1} \log a_i\right)$.

Glossary IV

$\text{mean}(A)$ The *arithmetic mean* $\text{mean}(A)$ is an estimate of the expected value of a distribution from which a data sample was, well, sampled. Its is computed on data sample $A = (a_0, a_1, \ldots, a_{n-1})$ as the sum of all n elements a_i in the sample data A divided by the total number n of values, i.e., $\text{mean}(A) = \frac{1}{n}\sum_{i=0}^{n-1} a_i$.

$\text{median}(A)$ The *median* $\text{median}(A)$ is the value separating the bigger-valued half from the smaller-valued half of a data sample or distribution. Its estimate is the value right in the middle of a *sorted* data sample $A = (a_0, a_1, \ldots, a_{n-1})$ where $a_{i-1} \leq a_i \; \forall i \in 1 \ldots (n-1)$ with an odd number of elements and the mean of the two values in the middle if n is even. In other words, $\text{median}(A) = a_{\frac{n-1}{2}}$ if n is odd and $\frac{1}{2}\left(a_{\frac{n}{2}-1} + a_{\frac{n}{2}}\right)$ otherwise, i.e., if n is even.

$\mathbb{N}_1$ the set of the natural numbers *excluding* 0, i.e., 1, 2, 3, 4, and so on. It holds that $\mathbb{N}_1 \subset \mathbb{Z}$.

$\mathcal{NP}$ $\mathcal{NP}$ is the class of computational problems that can be solved in polynomial time by a non-deterministic machine and can be verified in polynomial time by a deterministic machine (such as a normal computer)[25].

$\mathcal{NP}$-complete A decision problem is $\mathcal{NP}$-complete if it is in $\mathcal{NP}$ and all problems in $\mathcal{NP}$ are reducible to it in polynomial time[25,48]. A problem is $\mathcal{NP}$-complete if it is $\mathcal{NP}$-hard and if it is in $\mathcal{NP}$.

$\mathcal{NP}$-hard Algorithms that guarantee to find the correct solutions of $\mathcal{NP}$-hard problems[14,15,38] need a runtime that is exponential in the problem scale in the worst case. A problem is $\mathcal{NP}$-hard if all problems in $\mathcal{NP}$ are reducible to it in polynomial time[25].

$\mathcal{O}(g(x))$ If $f(x) = \mathcal{O}(g(x))$, then there exist positive numbers $x_0 \in \mathbb{R}^+$ and $c \in \mathbb{R}^+$ such that $f(x) \leq c * g(x) \forall x \geq x_0$[4,37]. In other words, $\mathcal{O}(g(x))$ describes an upper bound for function growth.

Glossary V

$\mathrm{quantile}_q^k(A)$ The q-*quantiles* are the cut points that divide a sorted data sample $A = (a_0, a_1, \dots, a_{n-1})$ where $a_{i-1} \leq a_i \; \forall i \in 1 \dots (n-1)$ into q equally-sized parts. $\mathrm{quantile}_q^k(A)$ be the k^{th} q-quantile, with $k \in 1 \dots (q-1)$, i.e., there are $q-1$ of the q-quantiles. In the context of this book, define $h = (n-1)\frac{k}{q}$. $\mathrm{quantile}_q^k(A)$ then can be computed as a_h if h is integer, i.e., $h \in \mathbb{Z}$, and as $a_{\lfloor h \rfloor} + (h - \lfloor h \rfloor) * \left(a_{\lfloor h \rfloor + 1} - a_{\lfloor h \rfloor}\right)$ otherwise. It holds that $\mathrm{quantile}_1^2(A) = \mathrm{median}(A)$

$\mathbb{R}$ the set of the real numbers.

$\mathbb{R}^+$ the set of the positive real numbers, i.e., $\mathbb{R}^+ = \{x \in \mathbb{R} : x > 0\}$.

$\mathrm{sd}(A)$ The statistical estimate $\mathrm{sd}(A)$ of the *standard deviation* of a data sample $A = (a_0, a_1, \dots, a_{n-1})$ with n observations is the square root of the estimated variance $\mathrm{var}(A)$, i.e., $\mathrm{sd}\, A = \sqrt{\mathrm{var}(A)}$.

$\mathrm{var}(A)$ The *variance* of a distribution is the expectation of the squared deviation of the underlying random variable from its mean. The variance $\mathrm{var}(A)$ of a data sample $A = (a_0, a_1, \dots, a_{n-1})$ with n observations can be estimated as $\mathrm{var}(A) = \frac{1}{n-1} \sum_{i=0}^{n-1} (a_i - \mathrm{mean}(A))^2$.

$\mathbb{Z}$ the set of the integers numbers including positive and negative numbers and 0, i.e., ..., -3, -2, -1, 0, 1, 2, 3, ..., and so on. It holds that $\mathbb{Z} \subset \mathbb{R}$.