The rôle of experiments in Analysis of Algorithms

Conrado Martínez

Univ. Politècnica de Catalunya, Spain

April 2008

Disclaimer

- Maybe I'll make a few "controversial" statements
- And there'll be little math ... sorry!
- I express here my own opinions; my goal is NOT to convince you BUT to make you think about it!

Disclaimer

- Maybe I'll make a few "controversial" statements
- And there'll be little math ... sorry!
- I express here my own opinions; my goal is NOT to convince you BUT to make you think about it!

Disclaimer

- Maybe I'll make a few "controversial" statements
- And there'll be little math ... sorry!
- I express here my own opinions; my goal is NOT to convince you BUT to make you think about it!

Contents

Contents

- "Philosophy"
- Example(s)

The goals of Analysis of Algorithms

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

The Goals of Analysis of Algorithms

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

The goals of Analysis of Algorithms

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

The Goals of Analysis of Algorithms

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

The Goals of Analysis of Algorithms

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

The Goals of Analysis of Algorithms

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

"Philosophical" issues

Claim:
Analysis of Algorithms $\approx$ Theoretical Physics

A pun

$E = \Theta(m)$ (Einstein's Special Relativity)

$F = O(md^{-2})$ (Newton's Gravity Law)

$I = O(V/\sqrt{R})$ (Ohm's Resistance Law)

$\Delta x = \Omega(\Delta p)$ (Heisenberg's Uncertainty Principle)

$I(\nu, T) = O(T\nu^2)$ (Planck's Law for Black Body Radiation);

A pun

$$E = mc^2$$ (Einstein's Special Relativity)

$$F = G\frac{Mm}{d^2}$$ (Newton's Gravity Law)

$$I = \frac{V}{R}$$ (Ohm's Resistance Law)

$$I(\nu, T) = \frac{2h\nu^3}{c^2}\frac{1}{e^{\frac{h\nu}{kT}} - 1}$$

Methodological issues

At a formal level:
We share many mathematical techniques and tools

- Complex analysis
- Differential equations
- Probability
- Linear algebra
- Generating functions (a.k.a. partition functions)

Methodological issues

Other areas of Computer Science rely more heavily in logic, abstract algebra, geometry, . . .

Epistemological issues

At a deep level:

- Goals #1 and #4 of AofA are identical to the main goals of any other Science
- We share with Theoretical Physics the quest for quantitative predictions, the use of mathematical models that provide measurable and precise descriptions of the behavior of a system (which ultimately explain it)
- Theoretical Physics is interested in observed natural phenomena; we (AofA) are interested in the behavior of artificial systems: algorithms

Epistemological issues

At a deep level:

- Goals #1 and #4 of AofA are identical to the main goals of any other Science
- We share with Theoretical Physics the quest for quantitative predictions, the use of mathematical models that provide measurable and precise descriptions of the behavior of a system (which ultimately explain it)
- Theoretical Physics is interested in observed natural phenomena; we (AofA) are interested in the behavior of artificial systems: algorithms

Epistemological issues

At a deep level:

- Goals #1 and #4 of AofA are identical to the main goals of any other Science
- We share with Theoretical Physics the quest for quantitative predictions, the use of mathematical models that provide measurable and precise descriptions of the behavior of a system (which ultimately explain it)
- Theoretical Physics is interested in observed natural phenomena; we (AofA) are interested in the behavior of artificial systems: algorithms

The rôle of experiments

My answer:
The rôle of experiments in AofA is the rôle they play in the scientific method.

The rôle of experiments

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

The rôle of experiments

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

The rôle of experiments

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

The rôle of experiments

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

The rôle of experiments

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

The rôle of experiments

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

The rôle of experiments

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

The rôle of experiments

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

The rôle of experiments

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

The rôle of experiments

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

Do's and Don'ts

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

Do's and Don'ts

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

Do's and Don'ts

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

Do's and Don'ts

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

Do's and Don'ts

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

Do's and Don'ts

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

Do's and Don'ts

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

Do's and Don'ts

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

Do's and Don'ts

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

Do's and Don'ts

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

Do's and Don'ts

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

Do's and Don'ts

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

Do's and Don'ts

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

Do's and Don'ts

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

Do's and Don'ts

- We (the AofA community) need not do the experiments ourselves; but we always should strive for someone to do them:
  - Beforehand, they provide data on which we can build our theories.
  - Afterwards, they should be used to validate our theories.

Do's and Don'ts

- We (the AofA community) need not do the experiments ourselves; but we always should strive for someone to do them:
  - Beforehand, they provide data on which we can build our theories.
  - Afterwards, they should be used to validate our theories.

Do's and Don'ts

- We (the AofA community) need not do the experiments ourselves; but we always should strive for someone to do them:
  - Beforehand, they provide data on which we can build our theories.
  - Afterwards, they should be used to validate our theories.

Do's and Don'ts

- If you want to do experiments yourself: Read the literature on experimental algorithmics, e.g. the papers by D. Johnson, by B. Moret or by C.C. McGeoch $\Longrightarrow$ better experiments, more useful data, ...
- If you do not want to do experiments yourself: Read the literature on experimental algorithmics $\Longrightarrow$ check that the experimental setup is correct and well designed, the results can be useful for you

Do's and Don'ts

- If you *want to do experiments yourself*: Read the literature on experimental algorithmics, e.g. the papers by D. Johnson, by B. Moret or by C.C. McGeoch $\Longrightarrow$ better experiments, more useful data, ...
- If you *do not want to do experiments yourself*: Read the literature on experimental algorithmics $\Longrightarrow$ check that the experimental setup is correct and well designed, the results can be useful for you

A very small sample of (un)explained experiments

- J. Eppinger's award winning "An empirical study of insertion and deletion in binary search trees" (1983); see the paper by J. Culberson and I. Munro for some models and simulations providing partial answers.
- R. Sedgewick's experiments on depth-first search to locate a path from u to v in a (grid) graph (AofA 2005).
- R. Sedgewick's experiments on the height of left-leaning red-black trees (AofA 2008).
- J.-F. Marckert's simulations of DLAs, animals, and other combinatorial structures (AofA 2008).

A very small sample of (un)explained experiments

- J. Eppinger's award winning "An empirical study of insertion and deletion in binary search trees" (1983); see the paper by J. Culberson and I. Munro for some models and simulations providing partial answers.
- R. Sedgewick's experiments on depth-first search to locate a path from u to v in a (grid) graph (AofA 2005).
- R. Sedgewick's experiments on the height of left-leaning red-black trees (AofA 2008).
- J.-F. Marckert's simulations of DLAs, animals, and other combinatorial structures (AofA 2008).

A very small sample of (un)explained experiments

- J. Eppinger's award winning "An empirical study of insertion and deletion in binary search trees" (1983); see the paper by J. Culberson and I. Munro for some models and simulations providing partial answers.
- R. Sedgewick's experiments on depth-first search to locate a path from u to v in a (grid) graph (AofA 2005).
- R. Sedgewick's experiments on the height of left-leaning red-black trees (AofA 2008).
- J.-F. Marckert's simulations of DLAs, animals, and other combinatorial structures (AofA 2008).

A very small sample of (un)explained experiments

- J. Eppinger's award winning "An empirical study of insertion and deletion in binary search trees" (1983); see the paper by J. Culberson and I. Munro for some models and simulations providing partial answers.
- R. Sedgewick's experiments on depth-first search to locate a path from u to v in a (grid) graph (AofA 2005).
- R. Sedgewick's experiments on the height of left-leaning red-black trees (AofA 2008).
- J.-F. Marckert's simulations of DLAs, animals, and other combinatorial structures (AofA 2008).

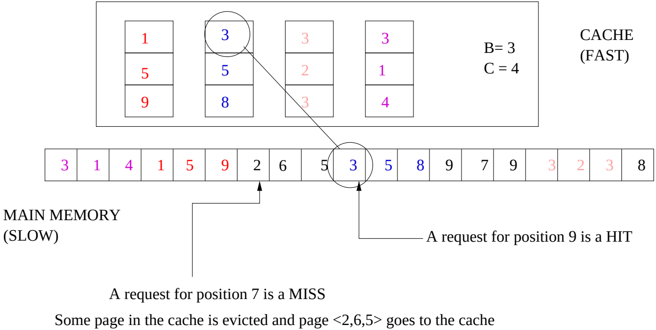
A very small sample of (un)explained experiments

- The cache performance of quicksort
- Other experiments (time permits)

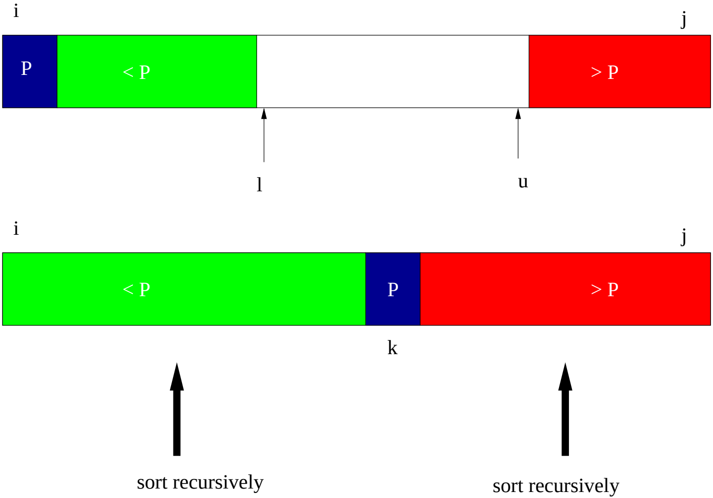
A very small sample of (un)explained experiments

- The cache performance of quicksort
- Other experiments (time permits)

The cache performance of quicksort



The cache performance of quicksort



The cache performance of quicksort

Q: How many cache misses do we expect when sorting a file of size n with quicksort?

The cache performance of quicksort

Q: How many cache misses do we expect when sorting a file of size n with quicksort?

A: $\Theta(\frac{n}{B}\ln n)$ ...But let's try being more precise!

The cache performance of quicksort

The model: a fully associative cache (with LRU replacement) with C pages of size B each (i.e., each page can store up to B elements)

Suppose that quicksort empties the cache after each partitioning stage

The cache performance of quicksort

The model: a fully associative cache (with LRU replacement) with C pages of size B each (i.e., each page can store up to B elements)

Suppose that quicksort empties the cache after each partitioning stage $\Longrightarrow$

$$\mathbb{E}[M_n] = \left\lceil \frac{n}{B} \right\rceil + 1 + \frac{2}{n} \sum_{k=1}^{n} \mathbb{E}[M_{k-1}], \qquad n > B$$

The cache performance of quicksort

But things are actually MUCH more complicated!

- After partitioning $A[i..j]$ some elements are in the cache, so the recursive calls on $A[i..k-1]$ and $A[k+1..j]$ may take advantage
- Observation: elements in the cache after partitioning are $\approx A[i..i+B-1]$ and $A[k-p..k+q]$ with $q-p+1 = C(B-1)$

The cache performance of quicksort

But things are actually MUCH more complicated!

- After partitioning $A[i..j]$ some elements are in the cache, so the recursive calls on $A[i..k-1]$ and $A[k+1..j]$ may take advantage
- Observation: elements in the cache after partitioning are $\approx A[i..i+B-1]$ and $A[k-p..k+q]$ with $q-p+1 = C(B-1)$

The cache performance of quicksort

But things are actually MUCH more complicated!

- If we call first on $A[i..k-1]$ then the elements $A[k+1..k+q]$ won't likely remain in the cache when the recursive sort of $A[i..k-1]$ finishes! $\Longrightarrow$ the # of misses of the two recursive calls are not independent
- And we are not taking alignment issues into account here!

The cache performance of quicksort

But things are actually MUCH more complicated!

- If we call first on $A[i..k-1]$ then the elements $A[k+1..k+q]$ won't likely remain in the cache when the recursive sort of $A[i..k-1]$ finishes! $\Longrightarrow$ the # of misses of the two recursive calls are not independent
- And we are not taking alignment issues into account here!

The cache performance of quicksort

LaMarca, Ladner: "The Influence of Caches on the Performance of Sorting" (1997)

- "Memory-tuned quicksort": small subarrays are immediately sorted with insertion sort rather than left unsorted until a final single insertion sort step
- "Multiquicksort": partition the array into several chunks using multiple pivots, so that each chunk fits the cache
- Both variants present modest performance gains

...

The cache performance of quicksort

LaMarca, Ladner: "The Influence of Caches on the Performance of Sorting" (1997)

- "Memory-tuned quicksort": small subarrays are immediately sorted with insertion sort rather than left unsorted until a final single insertion sort step
- "Multiquicksort": partition the array into several chunks using multiple pivots, so that each chunk fits the cache
- Both variants present modest performance gains

...

The cache performance of quicksort

LaMarca, Ladner: "The Influence of Caches on the Performance of Sorting" (1997)

- "Memory-tuned quicksort": small subarrays are immediately sorted with insertion sort rather than left unsorted until a final single insertion sort step
- "Multiquicksort": partition the array into several chunks using multiple pivots, so that each chunk fits the cache
- Both variants present modest performance gains

  ...

Small first

- In order to guarantee $O(\log n)$ stack size, the two recursive calls are reordered so that we first sort the smallest subarray
- This might be good for cache misses: we sort first a subarray that fits "better" in the cache ... right?

Small first

- In order to guarantee $O(\log n)$ stack size, the two recursive calls are reordered so that we first sort the smallest subarray
- This might be good for cache misses: we sort first a subarray that fits "better" in the cache ... right?

Small first

```
void quicksort(Vector& A, int i, int j) {
    int k;
    if (j - i + 1 <= n0) {
      easysort(A, i, j); return;
    }
    int pp = get_pivot(A, i, j);
    swap(A[i], A[pp]);
    partition(A, i, j, k);
    if (k - i <= j - k) {
      quicksort(A, i, k - 1);
      quicksort(A, k + 1, j);
    } else {
      quicksort(A, k + 1, j);
      quicksort(A, i, k - 1);
    }
};
```

Bidirectional partition

If we partition $A[i..j]$ and then call quicksort on $A[i..k-1]$, partition from right to left; if you call quicksort on $A[k..j]$ partition from left to right (usual partition)

Bidirectional partition

```
void sort_lr(Vector& A, int i, int j) {
    int k;
    if (j - i + 1 <= n0) {
      easysort(A, i, j); return;
    }
    int pp = get_pivot(A, i, j);
    swap(A[i], A[pp]);
    partition_lr(A, i, j, k);
    sort_rl(A, i, k - 1);
    sort_lr(A, k + 1, j);
}
```

Bidirectional partition

```
void sort_rl(Vector& A, int i, int j) {
    int k;
    if (j - i + 1 <= n0) {
      easysort(A, i, j); return;
    int pp = get_pivot(A, i, j);
    swap(A[j], A[pp]);
    partition_rl(A, i, j, k);
    sort_lr(A, k + 1, j);
    sort_rl(A, i, k - 1);
}
```

Experiments

- We generate $s = 500$ random permutations of each size $n \in \{1000, 2000, \ldots, 50000\}$
- We count the number of misses for each variant to process each input in the sample (the same input is fed to each quicksort)
- First set: $B := 100$, $C := 10$; second set: $B := 4$, $C = 25$
- Ratios (cache_size)/(array_size) ranging from 0.02 to 1 and from 0.002 to 0.1
- Simple pivot selection scheme

Experiments

- We generate $s = 500$ random permutations of each size $n \in \{1000, 2000, \dots, 50000\}$
- We count the number of misses for each variant to process each input in the sample (the same input is fed to each quicksort)
- First set: $B := 100$, $C := 10$; second set: $B := 4$, $C = 25$
- Ratios (cache_size)/(array_size) ranging from 0.02 to 1 and from 0.002 to 0.1
- Simple pivot selection scheme

Experiments

- We generate $s = 500$ random permutations of each size $n \in \{1000, 2000, \ldots, 50000\}$
- We count the number of misses for each variant to process each input in the sample (the same input is fed to each quicksort)
- First set: $B := 100$, $C := 10$; second set: $B := 4$, $C = 25$
- Ratios (cache_size)/(array_size) ranging from 0.02 to 1 and from 0.002 to 0.1
- Simple pivot selection scheme

Experiments

- We generate $s = 500$ random permutations of each size $n \in \{1000, 2000, \ldots, 50000\}$
- We count the number of misses for each variant to process each input in the sample (the same input is fed to each quicksort)
- First set: $B := 100$, $C := 10$; second set: $B := 4$, $C = 25$
- Ratios (cache_size)/(array_size) ranging from 0.02 to 1 and from 0.002 to 0.1
- Simple pivot selection scheme

Experiments

- We generate $s = 500$ random permutations of each size $n \in \{1000, 2000, \ldots, 50000\}$
- We count the number of misses for each variant to process each input in the sample (the same input is fed to each quicksort)
- First set: $B := 100$, $C := 10$; second set: $B := 4$, $C = 25$
- Ratios (cache_size)/(array_size) ranging from 0.02 to 1 and from 0.002 to 0.1
- Simple pivot selection scheme

Experiments

The baseline for comparison is quicksort with maximal "waste": $\mathbb{E}\left[M_n^{(0)}\right] = 2\frac{n}{B}\ln n + \Theta(n)$.
We look at the following quantities

- $\rho_n = \overline{M}_n / \mathbb{E}\left[M_n^{(0)}\right]$
- $\sigma_n = (\mathbb{E}\left[M_n^{(0)}\right] - \overline{M}_n)/n$
- μ_n = average # of misses per access = $\frac{\overline{M}_n}{2n\ln n + \Theta(n)} \approx \frac{1}{B}$

Experiments

The baseline for comparison is quicksort with maximal "waste": $\mathbb{E}\left[M_n^{(0)}\right] = 2\frac{n}{B}\ln n + \Theta(n)$.
We look at the following quantities

- $\rho_n = \overline{M}_n / \mathbb{E}\left[M_n^{(0)}\right]$
- $\sigma_n = (\mathbb{E}\left[M_n^{(0)}\right] - \overline{M}_n)/n$
- μ_n = average # of misses per access = $\frac{\overline{M}_n}{2n\ln n + \Theta(n)} \approx \frac{1}{B}$

Experiments

The baseline for comparison is quicksort with maximal "waste": $\mathbb{E}\left[M_n^{(0)}\right] = 2\frac{n}{B}\ln n + \Theta(n)$.

We look at the following quantities

- $\rho_n = \overline{M}_n / \mathbb{E}\left[M_n^{(0)}\right]$
- $\sigma_n = (\mathbb{E}\left[M_n^{(0)}\right] - \overline{M}_n)/n$
- μ_n = average # of misses per access = $\frac{\overline{M}_n}{2n\ln n + \Theta(n)} \approx \frac{1}{B}$

Experiments

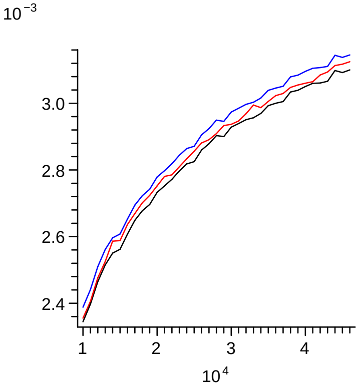
- The hypothesis: $\mathbb{E}[M_n] = \mathbb{E}\left[M_n^{(0)}\right] - k \cdot n$
- Different variants would have different values of k

Experiments

- The hypothesis: $\mathbb{E}[M_n] = \mathbb{E}\left[M_n^{(0)}\right] - k \cdot n$
- Different variants would have different values of k

Experiments

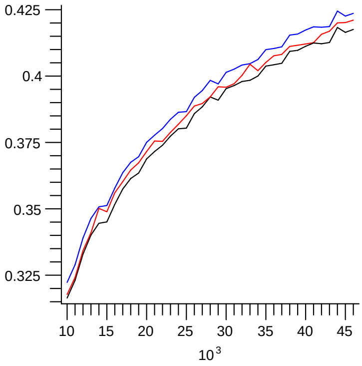
The number u_m of misses per access ($B = 100, C = 10$)



Black = Std; Red = Bidirectional; Blue = Small First

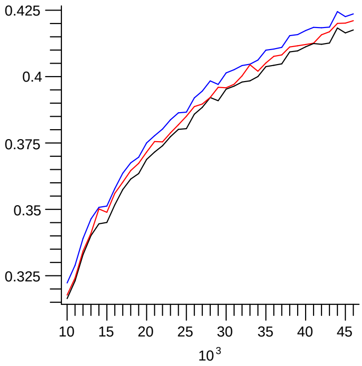
Experiments

The ratio $\rho_n = M_n / M_n^{(0)}$ ($B = 100, C = 10$)



Experiments

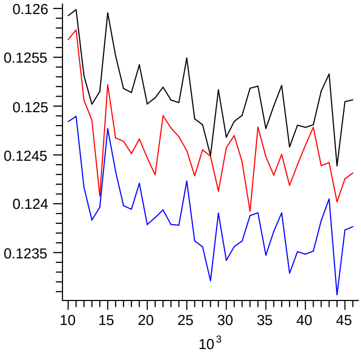
The ratio $\rho_n = M_n / M_n^{(0)}$ ($B = 100, C = 10$)



But ρ_n should tend to 1, right?

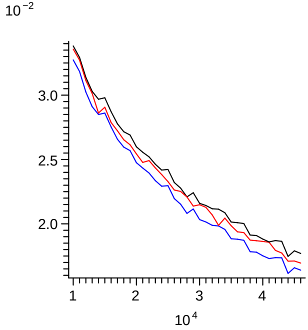
Experiments

The coefficient of n: $k \approx \frac{M_n^{(0)} - M_n}{n}$ ($B = 100, C = 10$)



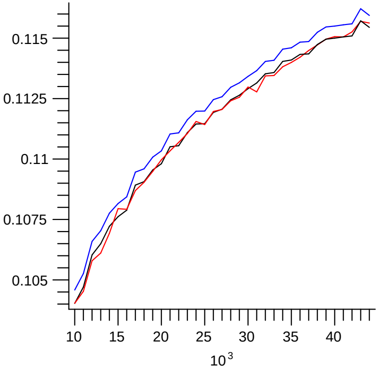
Experiments

The coefficient $k' \approx \frac{(n \ln n)/B - M_n}{n}$ $(B = 100, C = 10)$



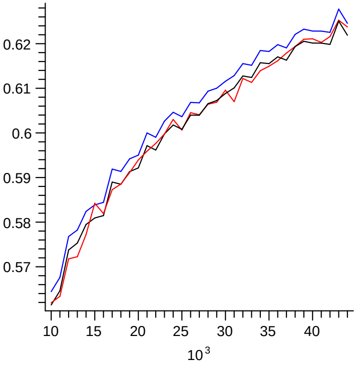
Experiments

The number μ_n of misses per access ($B = 4, C = 25$)



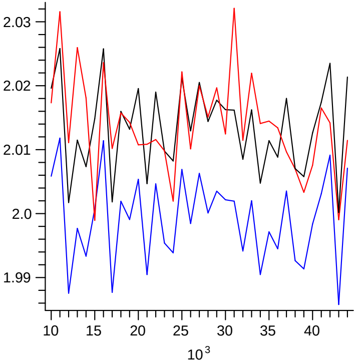
Experiments

The ratio $\rho_n = M_n / M_n^{(0)}$ ($B = 4, C = 25$)



Experiments

The coefficient of n: $k \approx \frac{M_n^{(0)} - M_n}{n}$ $(B = 4, C = 25)$



Experiments

- The experiments either do not support the hypothesis or k is very, very small for all considered strategies :(
- There are very small differences among the different strategies ... but why?
- For example, Small-First makes the same number of accesses as standard quicksort but apparently $\mathbb{E}\left[M_n^{(\text{std})}\right] < \mathbb{E}\left[M_n^{(\text{small-first})}\right]$
- Bidirectional partitioning is slightly better than Small-First, but still slightly worse than standard

Jump to end

Experiments

- The experiments either do not support the hypothesis or k is very, very small for all considered strategies :(
- There are very small differences among the different strategies … but why?
- For example, Small-First makes the same number of accesses as standard quicksort but apparently $\mathbb{E}\left[M_n^{(\mathrm{std})}\right] < \mathbb{E}\left[M_n^{(\mathrm{small\text{-}first})}\right]$
- Bidirectional partitioning is slightly better than Small-First, but still slightly worse than standard

Jump to end

Experiments

- The experiments either do not support the hypothesis or k is very, very small for all considered strategies :(
- There are very small differences among the different strategies ... but why?
- For example, Small-First makes the same number of accesses as standard quicksort but apparently $\mathbb{E}\left[M_n^{(\text{std})}\right] < \mathbb{E}\left[M_n^{(\text{small-first})}\right]$
- Bidirectional partitioning is slightly better than Small-First, but still slightly worse than standard

Jump to end

Experiments

- The experiments either do not support the hypothesis or k is very, very small for all considered strategies :(
- There are very small differences among the different strategies ... but why?
- For example, Small-First makes the same number of accesses as standard quicksort but apparently $\mathbb{E}\left[M_n^{(\text{std})}\right] < \mathbb{E}\left[M_n^{(\text{small-first})}\right]$
- Bidirectional partitioning is slightly better than Small-First, but still slightly worse than standard

Jump to end

Filtering DNA sequences

- Given two long sequences of DNA, a common task in Bioinformatics is to find regions of similarity among them (cf. G. Gonnet, AofA 2008)
- Finding the similarities is computationally expensive
- Quite often, simple algorithms called filters are run on the two sequences to focus the similarity computation on promising regions

Joint work with F. Bassino and J. Clément

Filtering DNA sequences

- Given two long sequences of DNA, a common task in Bioinformatics is to find regions of similarity among them (cf. G. Gonnet, AofA 2008)
- Finding the similarities is computationally expensive
- Quite often, simple algorithms called filters are run on the two sequences to focus the similarity computation on promising regions

Joint work with F. Bassino and J. Clément

Filtering DNA sequences

- Given two long sequences of DNA, a common task in Bioinformatics is to find regions of similarity among them (cf. G. Gonnet, AofA 2008)
- Finding the similarities is computationally expensive
- Quite often, simple algorithms called filters are run on the two sequences to focus the similarity computation on promising regions

Joint work with F. Bassino and J. Clément

Filtering DNA sequences

- The filter processes "windows" u and v of length L and $2L$ from the given sequences U and V and either keeps them for future inspection or discards them
- The filters we consider are reliable: if u and v are contain significant similaraties then the filter will keep (u, v)
- A pair (u, v) is "good" if there is a substring v' of v such that the edit distance between u and v' is $\leq d$ —we allow for up to d errors; symbollicaly: $\Delta^*(u, v) \leq d$
- Typical values: $L = 100$, $d = 0.1L$
- The problem is that a filter might keep (u, v) which are not "good"; those are called false positives

Filtering DNA sequences

- The filter processes "windows" u and v of length L and $2L$ from the given sequences U and V and either keeps them for future inspection or discards them
- The filters we consider are reliable: if u and v are contain significant similaraties then the filter will keep (u, v)
- A pair (u, v) is "good" if there is a substring v' of v such that the edit distance between u and v' is $\leq d$ —we allow for up to d errors; symbollicaly: $\Delta^*(u, v) \leq d$
- Typical values: $L = 100$, $d = 0.1L$
- The problem is that a filter might keep (u, v) which are not "good"; those are called false positives

Filtering DNA sequences

- The filter processes "windows" u and v of length L and $2L$ from the given sequences U and V and either keeps them for future inspection or discards them
- The filters we consider are reliable: if u and v are contain significant similaraties then the filter will keep (u, v)
- A pair (u, v) is "good" if there is a substring v' of v such that the edit distance between u and v' is $\leq d$ —we allow for up to d errors; symbollicaly: $\Delta^*(u, v) \leq d$
- Typical values: $L = 100$, $d = 0.1L$
- The problem is that a filter might keep (u, v) which are not "good"; those are called false positives

Filtering DNA sequences

- The filter processes "windows" u and v of length L and $2L$ from the given sequences U and V and either keeps them for future inspection or discards them
- The filters we consider are reliable: if u and v are contain significant similaraties then the filter will keep (u, v)
- A pair (u, v) is "good" if there is a substring v' of v such that the edit distance between u and v' is $\leq d$ —we allow for up to d errors; symbollicaly: $\Delta^*(u, v) \leq d$
- Typical values: $L = 100$, $d = 0.1L$
- The problem is that a filter might keep (u, v) which are not "good"; those are called false positives

Filtering DNA sequences

- The filter processes "windows" u and v of length L and $2L$ from the given sequences U and V and either keeps them for future inspection or discards them
- The filters we consider are reliable: if u and v are contain significant similaraties then the filter will keep (u, v)
- A pair (u, v) is "good" if there is a substring v' of v such that the edit distance between u and v' is $\leq d$ —we allow for up to d errors; symbollicaly: $\Delta^*(u, v) \leq d$
- Typical values: $L = 100$, $d = 0.1L$
- The problem is that a filter might keep (u, v) which are not "good"; those are called false positives

Filtering DNA sequences

- A k-match in (u, v) is a pair (i, j) such that $u[i..i+k-1] = v[j..j+k-1]$
- A folklore result states that

$$\#\text{ of } k\text{-matches} \geq L - k(d+1) + 1$$

- This is the basis for the filter analyzed by Sutinen and Szpankowski (Fun with Algorithms, 1998)
- There is an optimal choice for k: if too small then you get too many matches by pure chance; if too large then the lower bound above is not useful
- We consider a filter used in the system Nimbus which counts the number of order preserving matches

Filtering DNA sequences

- A k-match in (u, v) is a pair (i, j) such that $u[i..i+k-1] = v[j..j+k-1]$
- A folklore result states that

$$\#\text{ of } k\text{-matches} \geq L - k(d+1) + 1$$

- This is the basis for the filter analyzed by Sutinen and Szpankowski (Fun with Algorithms, 1998)
- There is an optimal choice for k: if too small then you get too many matches by pure chance; if too large then the lower bound above is not useful
- We consider a filter used in the system Nimbus which counts the number of order preserving matches

Filtering DNA sequences

- A k-match in (u, v) is a pair (i, j) such that $u[i..i + k - 1] = v[j..j + k - 1]$
- A folklore result states that

$$\#\text{ of } k\text{-matches} \geq L - k(d + 1) + 1$$

- This is the basis for the filter analyzed by Sutinen and Szpankowski (Fun with Algorithms, 1998)
- There is an optimal choice for k: if too small then you get too many matches by pure chance; if too large then the lower bound above is not useful
- We consider a filter used in the system Nimbus which counts the number of order preserving matches

Filtering DNA sequences

- A k-match in (u, v) is a pair (i, j) such that $u[i..i+k-1] = v[j..j+k-1]$
- A folklore result states that

$$\#\ \text{of}\ k\text{-matches} \geq L - k(d+1) + 1$$

- This is the basis for the filter analyzed by Sutinen and Szpankowski (Fun with Algorithms, 1998)
- There is an optimal choice for k: if too small then you get too many matches by pure chance; if too large then the lower bound above is not useful
- We consider a filter used in the system Nimbus which counts the number of order preserving matches

Filtering DNA sequences

- A k-match in (u,v) is a pair (i,j) such that $u[i..i+k-1] = v[j..j+k-1]$
- A folklore result states that

$$\#\text{ of } k\text{-matches} \geq L - k(d+1) + 1$$

- This is the basis for the filter analyzed by Sutinen and Szpankowski (Fun with Algorithms, 1998)
- There is an optimal choice for k: if too small then you get too many matches by pure chance; if too large then the lower bound above is not useful
- We consider a filter used in the system Nimbus which counts the number of order preserving matches

Filtering DNA sequences

- The relevant parameter is the efficiency of the filter

$$f = \Pr\{(u,v) \text{ is good} \,|\, (u,v) \text{ is kept}\}$$

- The efficiency f can be calculated from

$$\nu_\delta = \Pr\{(u,v) \text{ is kept} \,|\, \Delta^*(u,v) = \delta\}, \qquad \delta > d$$

- We have an approximate model to compute ν_δ (there's still some work to be done!)
- But now I'll show you a few results from experiments...

Filtering DNA sequences

- The relevant parameter is the efficiency of the filter

$$f = \Pr\{(u, v) \text{ is good} \mid (u, v) \text{ is kept}\}$$

- The efficiency f can be calculated from

$$\nu_\delta = \Pr\{(u, v) \text{ is kept} \mid \Delta^*(u, v) = \delta\}, \qquad \delta > d$$

- We have an approximate model to compute ν_δ (there's still some work to be done!)
- But now I'll show you a few results from experiments...

Filtering DNA sequences

- The relevant parameter is the efficiency of the filter

$$f = \Pr\{(u,v) \text{ is good} \mid (u,v) \text{ is kept}\}$$

- The efficiency f can be calculated from

$$\nu_\delta = \Pr\{(u,v) \text{ is kept} \mid \Delta^*(u,v) = \delta\}, \qquad \delta > d$$

- We have an approximate model to compute ν_δ (there's still some work to be done!)
- But now I'll show you a few results from experiments...

Filtering DNA sequences

- The relevant parameter is the efficiency of the filter

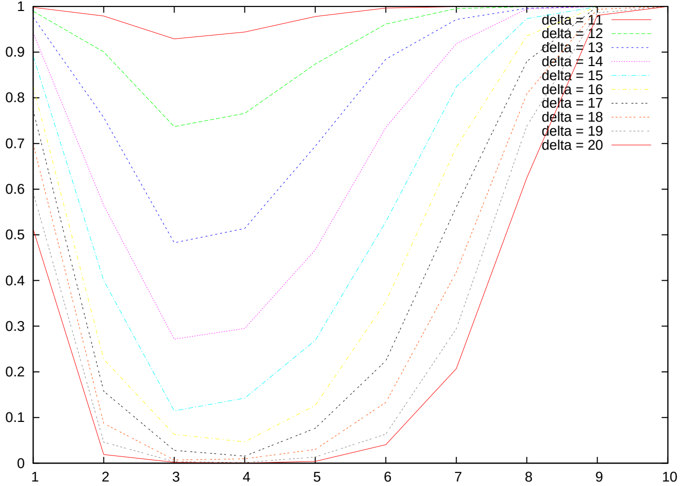
$$f = \Pr\{(u,v) \text{ is good} \mid (u,v) \text{ is kept}\}$$

- The efficiency f can be calculated from

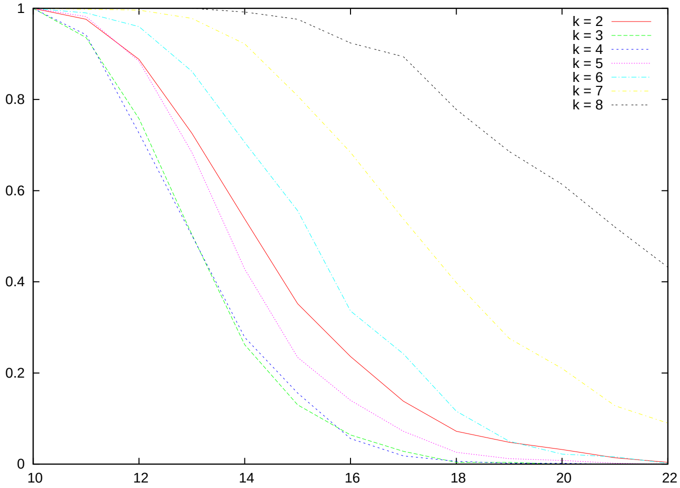
$$\nu_\delta = \Pr\{(u,v) \text{ is kept} \mid \Delta^*(u,v) = \delta\}, \qquad \delta > d$$

- We have an approximate model to compute ν_δ (there's still some work to be done!)
- But now I'll show you a few results from experiments...

Filtering DNA sequences



Filtering DNA sequences



▸ Jump to end

The hiring problem

- Originally introduced by Broder et al. (SODA 2008)
- A (potentially infinite) sequence of i.i.d. random variables Q_i uniformly distributed in $[0, 1]$
- At step i you either hire or discard candidate i with score Q_i (cf. T. Bruss, AofA 2008)
- Decisions are irrevocable
- Goals: hire candidates at some reasonable rate, improve the "mean" quality of the company's staff

Joint work with M. Archibald

The hiring problem

- Originally introduced by Broder et al. (SODA 2008)
- A (potentially infinite) sequence of i.i.d. random variables Q_i uniformly distributed in $[0, 1]$
- At step i you either hire or discard candidate i with score Q_i (cf. T. Bruss, AofA 2008)
- Decisions are irrevocable
- Goals: hire candidates at some reasonable rate, improve the "mean" quality of the company's staff

Joint work with M. Archibald

The hiring problem

- Originally introduced by Broder et al. (SODA 2008)
- A (potentially infinite) sequence of i.i.d. random variables Q_i uniformly distributed in $[0, 1]$
- At step i you either hire or discard candidate i with score Q_i (cf. T. Bruss, AofA 2008)
- Decisions are irrevocable
- Goals: hire candidates at some reasonable rate, improve the "mean" quality of the company's staff

Joint work with M. Archibald

The hiring problem

- Originally introduced by Broder et al. (SODA 2008)
- A (potentially infinite) sequence of i.i.d. random variables Q_i uniformly distributed in $[0, 1]$
- At step i you either hire or discard candidate i with score Q_i (cf. T. Bruss, AofA 2008)
- Decisions are irrevocable
- Goals: hire candidates at some reasonable rate, improve the "mean" quality of the company's staff

Joint work with M. Archibald

The hiring problem

- Originally introduced by Broder et al. (SODA 2008)
- A (potentially infinite) sequence of i.i.d. random variables Q_i uniformly distributed in $[0, 1]$
- At step i you either hire or discard candidate i with score Q_i (cf. T. Bruss, AofA 2008)
- Decisions are irrevocable
- Goals: hire candidates at some reasonable rate, improve the "mean" quality of the company's staff

Joint work with M. Archibald

The hiring problem

- Here: a permutation π of length n, candidate i has score $\pi(i)$
- The model is equivalent after "normalization", but is amenable to techniques from analytic combinatorics
- We call a hiring strategy rank-based if and only if it only depends on the relative ranks of the candidates seen so far
- $\mathcal{H}(\pi)$ = the set of candidates hired in permutation π; $h(\pi) = \#\mathcal{H}(\pi)$

The hiring problem

- Here: a permutation π of length n, candidate i has score $\pi(i)$
- The model is equivalent after "normalization", but is amenable to techniques from analytic combinatorics
- We call a hiring strategy rank-based if and only if it only depends on the relative ranks of the candidates seen so far
- $\mathcal{H}(\pi)$ = the set of candidates hired in permutation π; $h(\pi) = \#\mathcal{H}(\pi)$

The hiring problem

- Here: a permutation π of length n, candidate i has score $\pi(i)$
- The model is equivalent after "normalization", but is amenable to techniques from analytic combinatorics
- We call a hiring strategy rank-based if and only if it only depends on the relative ranks of the candidates seen so far
- $\mathcal{H}(\pi)$ = the set of candidates hired in permutation π; $h(\pi) = \#\mathcal{H}(\pi)$

The hiring problem

- Here: a permutation π of length n, candidate i has score $\pi(i)$
- The model is equivalent after "normalization", but is amenable to techniques from analytic combinatorics
- We call a hiring strategy rank-based if and only if it only depends on the relative ranks of the candidates seen so far
- $\mathcal{H}(\pi)$ = the set of candidates hired in permutation π; $h(\pi) = \#\mathcal{H}(\pi)$

The hiring problem

Let $\pi \circ j$ denote the permutation one gets after relabelling $j, j+1, \ldots, n = |\pi|$ to $j+1, j+2, \ldots, n+1$ and appending j at the end.
Example: $32451 \circ 3 = 425613$
Let $X_j(\pi) = 1$ if candidate with score j is hired after π and $X_j(\pi) = 0$ otherwise.

$$h(\pi \circ j) = h(\pi) + X_j(\pi)$$

Let $X(\pi)$ the number of j such that $X_j(\pi) = 1$.

The hiring problem

Theorem

Let $H(z,u) = \sum_{\pi\in\mathcal{P}} \frac{z^{|\pi|}}{|\pi|!} u^{h(\pi)}$. Then

$$(1-z)\frac{\partial}{\partial z}H(z,u) - H(z,u) = (u-1)\sum_{\pi\in\mathcal{P}} X(\pi)\frac{z^{|\pi|}}{|\pi|!}u^{h(\pi)}.$$

Hiring above the maximum

Candidate i is hired if and only if her score is above the score of the best currently hired candidate.

- $X(\pi) = 1$
- $\mathcal{H}(\pi) = \{i : i \text{ is a left-to-right maximum}\}$
- $\mathbb{E}[h_n] = [z^n] \left.\frac{\partial H}{\partial u}\right|_{u=1} = \ln n + O(1)$
- Variance of h_n is also $\ln n + O(1)$ and after proper normalization h_n^* converges to $\mathcal{N}(0,1)$

Hiring above the maximum

Candidate i is hired if and only if her score is above the score of the best currently hired candidate.

- $X(\pi) = 1$
- $\mathcal{H}(\pi) = \{i : i \text{ is a left-to-right maximum}\}$
- $\mathbb{E}[h_n] = [z^n] \left.\frac{\partial H}{\partial u}\right|_{u=1} = \ln n + O(1)$
- Variance of h_n is also $\ln n + O(1)$ and after proper normalization h_n^* converges to $\mathcal{N}(0,1)$

Hiring above the maximum

Candidate i is hired if and only if her score is above the score of the best currently hired candidate.

- $X(\pi) = 1$
- $\mathcal{H}(\pi) = \{i : i \text{ is a left-to-right maximum}\}$
- $\mathbb{E}[h_n] = [z^n] \left.\frac{\partial H}{\partial u}\right|_{u=1} = \ln n + O(1)$
- Variance of h_n is also $\ln n + O(1)$ and after proper normalization h_n^* converges to $\mathcal{N}(0,1)$

Hiring above the maximum

Candidate i is hired if and only if her score is above the score of the best currently hired candidate.

- $X(\pi) = 1$
- $\mathcal{H}(\pi) = \{i : i \text{ is a left-to-right maximum}\}$
- $\mathbb{E}[h_n] = [z^n] \left.\frac{\partial H}{\partial u}\right|_{u=1} = \ln n + O(1)$
- Variance of h_n is also $\ln n + O(1)$ and after proper normalization h_n^* converges to $\mathcal{N}(0,1)$

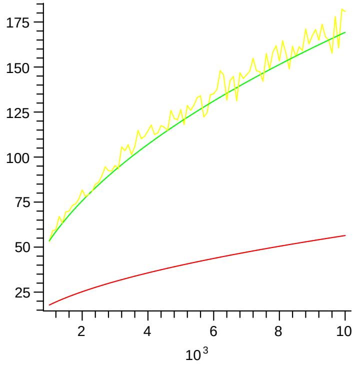
Hiring above the median

Candidate i is hired if and only if her score is above the score of the median of the scores of currently hired candidates.

- $X(\pi) = \lceil (h(\pi)+1)/2 \rceil$
- $\sqrt{\frac{n}{\pi}}(1+O(n^{-1})) \leq \mathbb{E}[h_n] \leq 3\sqrt{\frac{n}{\pi}}(1+O(n^{-1}))$
- This result follows easily by using previous theorem with $X_L(\pi) = (h(\pi)+1)/2$ and $X_U(\pi) = (h(\pi)+3)/2$ to lower and upper bound

Hiring above the median

$n \in \{1000, \ldots, 10000\}$, $M = 100$ random permutations for each n



Remarks on the hiring problem

- Experiments hint at $X_U(\pi)$ close to giving the right answer. But why?
- We have many other results on the two previous strategies and other reasonable strategies (e.g., hire above the best P%)
- There is a lot of work that remains to be done

Conclusions

- This talk is only my personal view of this important(?) issue
- ...and a taste of a few problems where experimentation can play its part
- I'll be happy to answer questions (if I can), but I prefer you express your comments!

Conclusions

- This talk is only my personal view of this important(?) issue
- ...and a taste of a few problems where experimentation can play its part
- I'll be happy to answer questions (if I can), but I prefer you express your comments!

Conclusions

- This talk is only my personal view of this important(?) issue
- ...and a taste of a few problems where experimentation can play its part
- I'll be happy to answer questions (if I can), but I prefer you express your comments!

Thank you for your attention