

Comparability

David S. Johnson

AT&T Labs - Research

<http://www.research.att.com/~dsj/>

Outline

- Classic "Reproducibility" and the Experimental Analysis of Algorithms
- The Traveling Salesman Problem (TSP)
-- our running example
- "Comparability"
- More fun TSP stuff (time permitting)

Experimental Analysis of Algorithms: Measurements of Interest

1. Quality of solution (possibly under multiple criteria)
2. Running Time
3. Memory utilization
4. Communication bandwidth (parallel/distributed algorithms)
5. ...

Experimental Analysis of Algorithms: The Challenge

We can't test algorithms directly.

What we test is

1. Code, supposedly implementing the algorithm
2. Compiled using a particular compiler
3. Run on a particular machine/operating system
4. Using particular instances as input

Classic Reproducibility

Minor changes in "the apparatus" (1-4) should not have a major affect on the results.

Problem

- Changes in the “computational apparatus” can have major effects on the quantities measured.

	Solution Quality	Running Time
Code		
Language/Compiler		
Operating System		
Machine		
Time of Run		
Name of Code		
Test Instances		
Comparison Standard		

Problem

- Changes in the “computational apparatus” can have major effects on the quantities measured.

	Solution Quality	Running Time
Code		Major
Language/Compiler		Significant
Operating System		Significant
Machine		Major
Time of Run		Can be Significant
Name of Code		Can be Significant
Test Instances		Major

The Traveling Salesman Problem

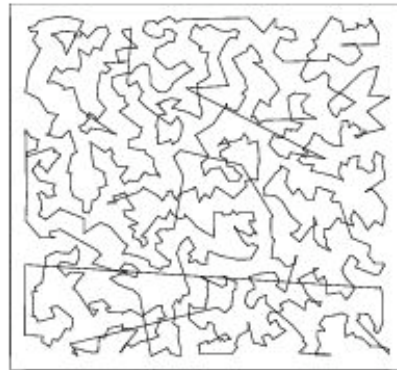
- Given a set C of cities, and for each pair c_i, c_j of cities a distance d_{ij} , find a permutation π of the cities (a tour) that has the minimum possible length

$$\sum_{1 \leq i < n} d_{\pi(i), \pi(i+1)} + d_{\pi(n), \pi(1)}$$

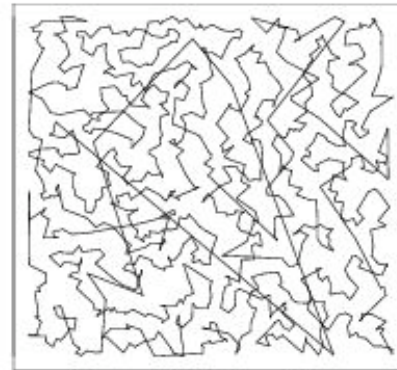
- The classic example of an NP-hard problem
- But not at all typical of such problems
 - Advanced linear-programming-based branch-and-cut procedures have succeeded in finding optimal solutions for real-world instances with as many as 85,900 cities [Applegate, Bixby, Chvatal, & Cook, 2006]
 - Fast implementations of the Lin-Kernighan algorithm can get within 1.5% of optimal on million-city instances in 3 minutes on a modern machine (11 minutes for 3 million cities, 48 minutes for 10 million cities).

8th DIMACS Implementation Challenge:

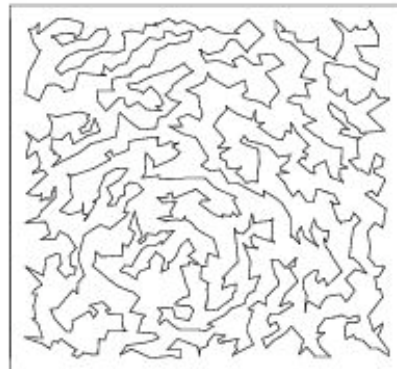
The Traveling Salesman Problem



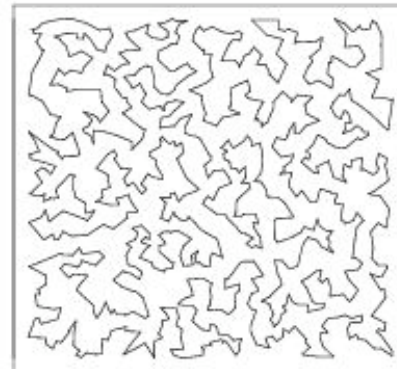
Greedy Tour



Nearest Neighbor Tour



Savings Tour



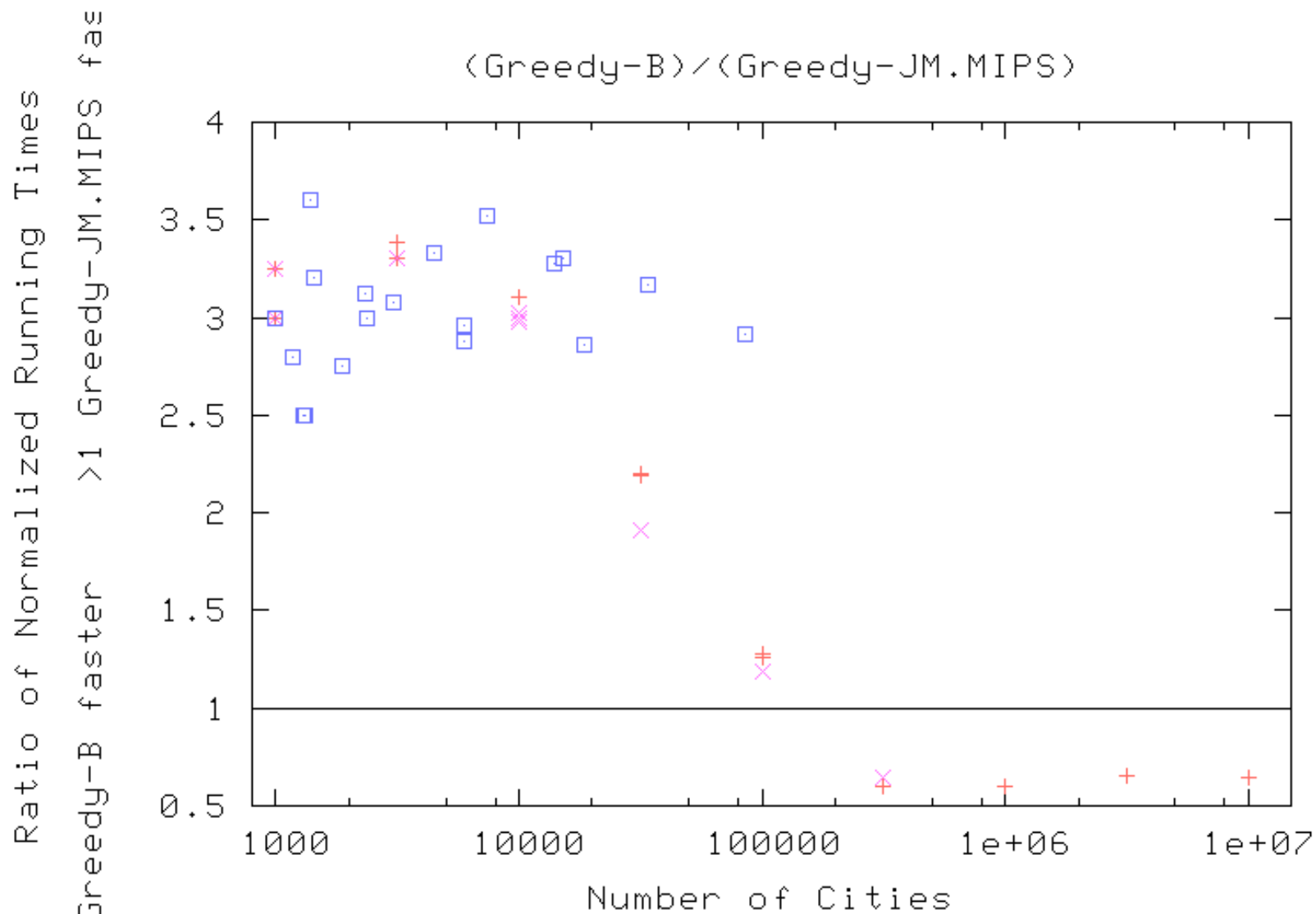
Optimal Tour

Challenge News: Still Open for Business!

<http://www.research.att.com/~dsj/chtsp>

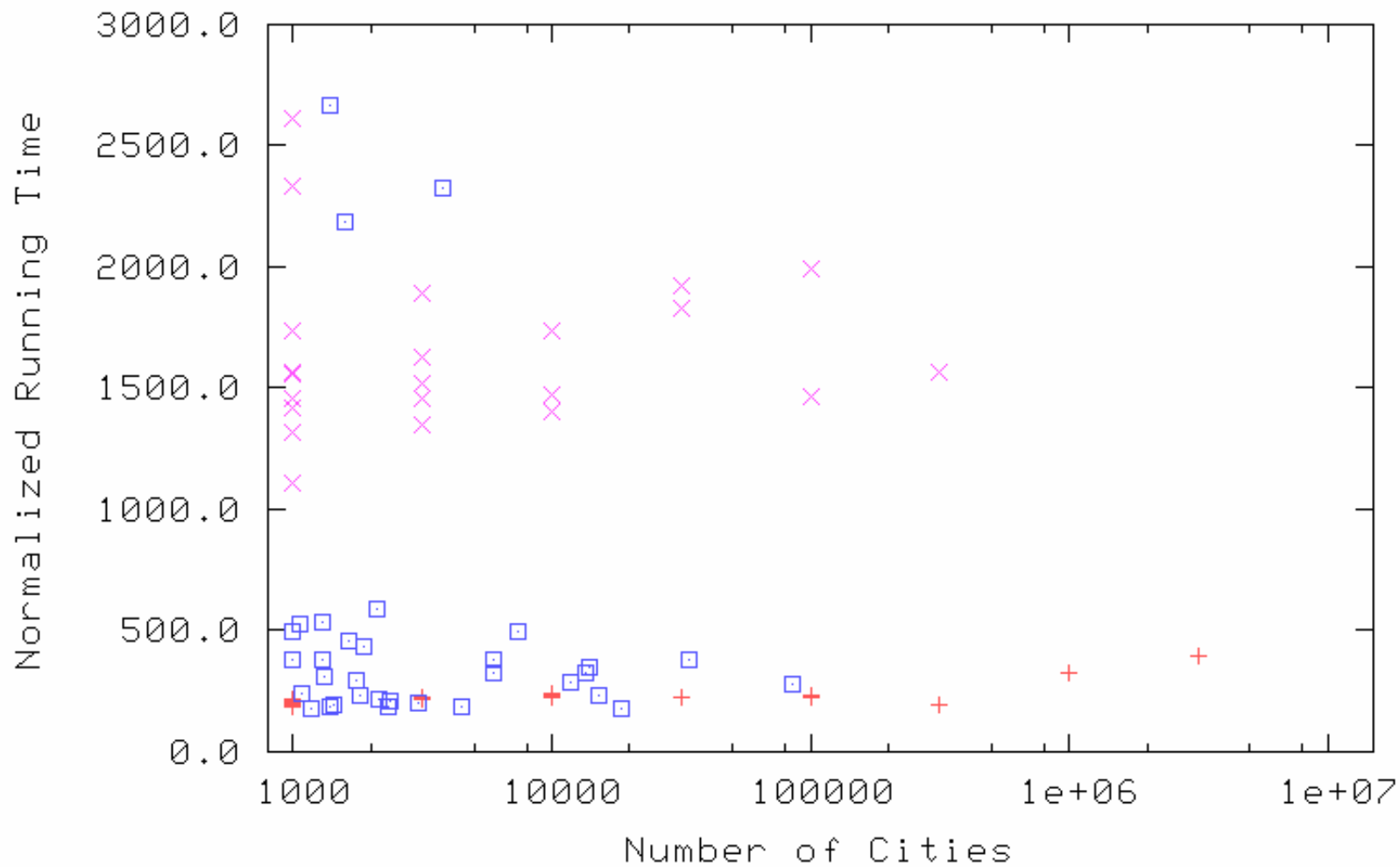
Lin-Kernighan (Johnson-McGeoch Implementation) on a 10,000 City Random Euclidean Instance

Machine	Time in Seconds
Sequent Balance (1984)	~3600
VAX 8550 6Mhz (?) (1990)	340.2
150 Mhz SGI Challenge (1993)	9.8
400 Mhz SGI Challenge (2000)	1.8
2.6 Ghz Intel Xeon (2008)	.57



Uniform Points + TSPLIB Instances □
 Clustered Points ×

LK-JM-20.MIPS : Microseconds/N



Uniform Points + TSPLIB Instances □
 Clustered Points x

Operation Counts: A solution to the non-reproducibility of time?

- For example, number of mallocs, comparisons, calls to key subroutines, etc.
- For the most part unaffected by machine, operating system, language, and code-tuning
- But, especially in complicated algorithms, counts may not be strongly correlated with running time.
- And they may not be useful in comparing algorithms that take different approaches to the same problem.

Reproducibility Issues

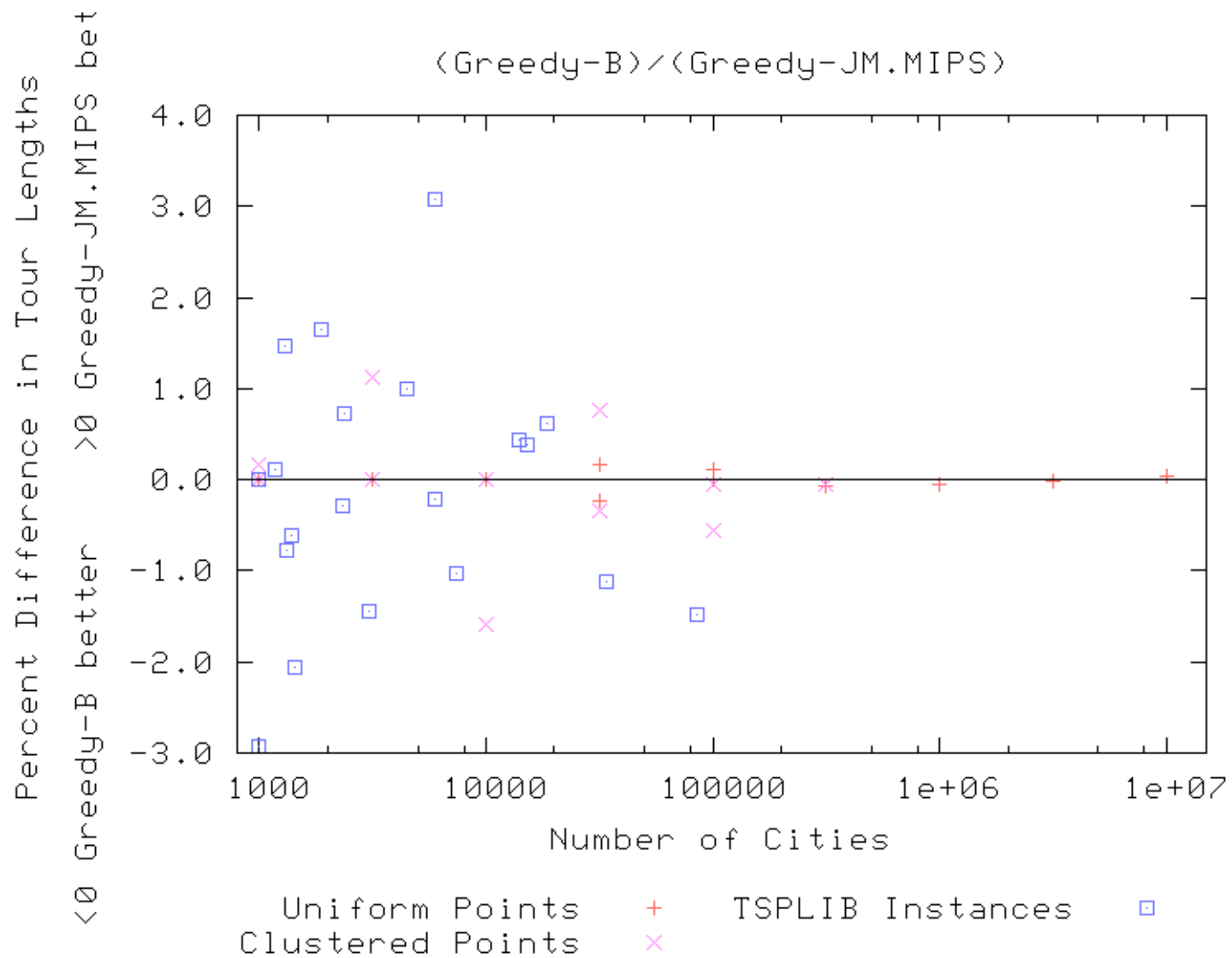
- Changes in the “computational apparatus” can have major effects on the quantities measured.

	Solution Quality	Running Time
Code		Major
Language/Compiler		Significant
Operating System		Significant
Machine		Major
Time of Run		Can be Significant
Name of Code		Can be Significant
Test Instances		Major

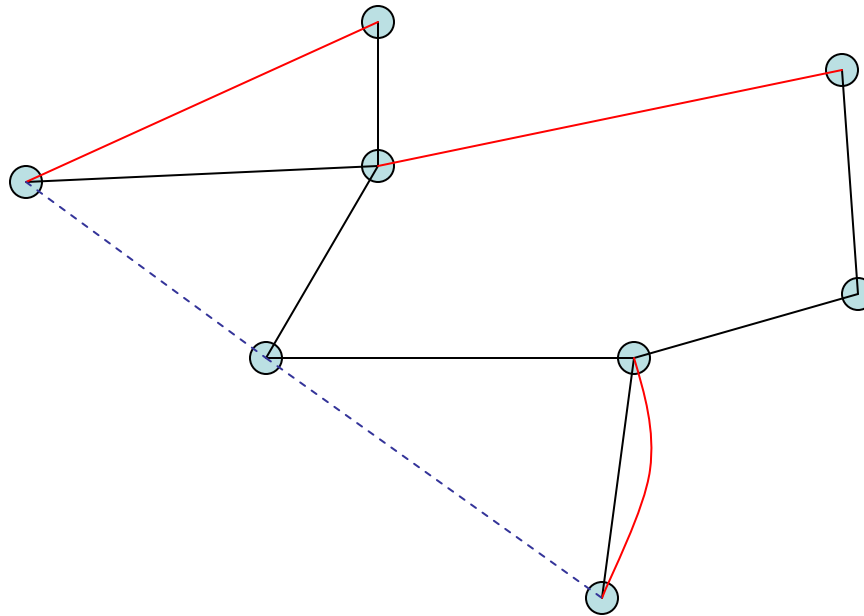
Reproducibility Issues

- Changes in the “computational apparatus” can have major effects on the quantities measured.

	Solution Quality	Running Time
Code	Major	Major
Language/Compiler		Significant
Operating System		Significant
Machine		Major
Time of Run		Can be Significant
Name of Code		Can be Significant
Test Instances		Major



Christofides Algorithm



Add a minimum-weight matching to the spanning tree. Odd degree vertices of M are previously visited cities.
 (No increase in length, assuming Triangle Inequality)
 No longer than $\frac{1}{2}$ Optimal Tour assuming Triangle Inequality

Implementation Detail

(Does not affect worst-case analysis)

- How are Shortcuts performed?
 - Perform shortcuts in sequence as you traverse the Euler Tour
 - At each place where the Euler tour crosses itself, perform the shortcut that reduces the tour length the most

Christofides-with-Standard-Shortcuts Algorithm

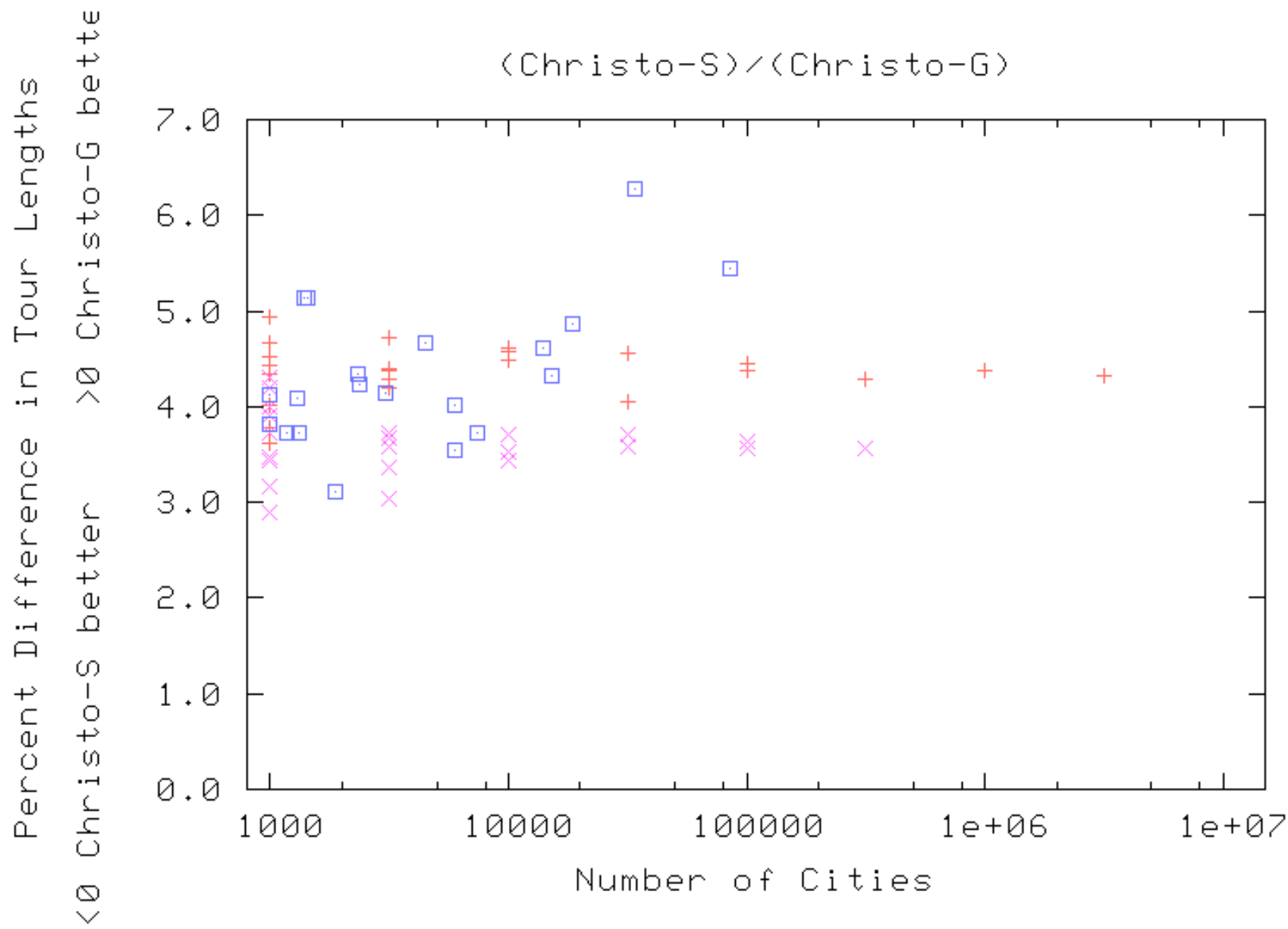
Average Percent Excess over the HK Bound and Normalized Running Times

	1000	3162	10K	31K	100K	316K	1M	3.16M	10M
Average Percent Excess over HK Bound									
Uniform Points	14.48	14.61	14.81	14.67	14.70	14.49	14.59	14.51	--
Clustered Points	12.03	12.79	13.08	13.47	13.50	13.45			
TSPLIB Instances	13.36	14.17	13.41	16.50	15.46				
Random Matrices	--	--	--						

Christofides-with-Smart-Shortcuts Algorithm

Average Percent Excess over the HK Bound and Normalized Running Times

	1000	3162	10K	31K	100K	316K	1M	3.16M	10M
Average Percent Excess over HK Bound									
Uniform Points	9.80	9.79	9.81	9.95	9.85	9.80	9.79	9.75	.
Clustered Points	8.08	9.01	9.21	9.47	9.55	9.55			
TSPLIB Instances	8.72	9.41	8.86	9.62	9.50				
Random Matrices	--	--	--						



Uniform Points + TSPLIB Instances □
 Clustered Points x

Reproducibility *sina qua non*

Algorithms must be described in sufficient detail that any implementation that follows the description will obtain the same solutions (or at least solutions of the same average quality).

Reproducibility Issues

- Changes in the “computational apparatus” can have major effects on the quantities measured.

	Solution Quality	Running Time
Code	Major	Major
Language/Compiler		Significant
Operating System		Significant
Machine		Major
Time of Run		Can be Significant
Name of Code		Can be Significant
Test Instances		Major

Reproducibility Issues

- Changes in the “computational apparatus” can have major effects on the quantities measured.

	Solution Quality	Running Time
Code	Major	Major
Language/Compiler	Possible Effects	Significant
Operating System	Possible Effects	Significant
Machine	Possible Effects	Major
Time of Run	No*	Can be Significant
Name of Code	No*	Can be Significant
Test Instances	Major	Major

Test Instances

1) Real-world structured instances

- Don't provide reproducible results unless you make instances available
- Limits scalability testing

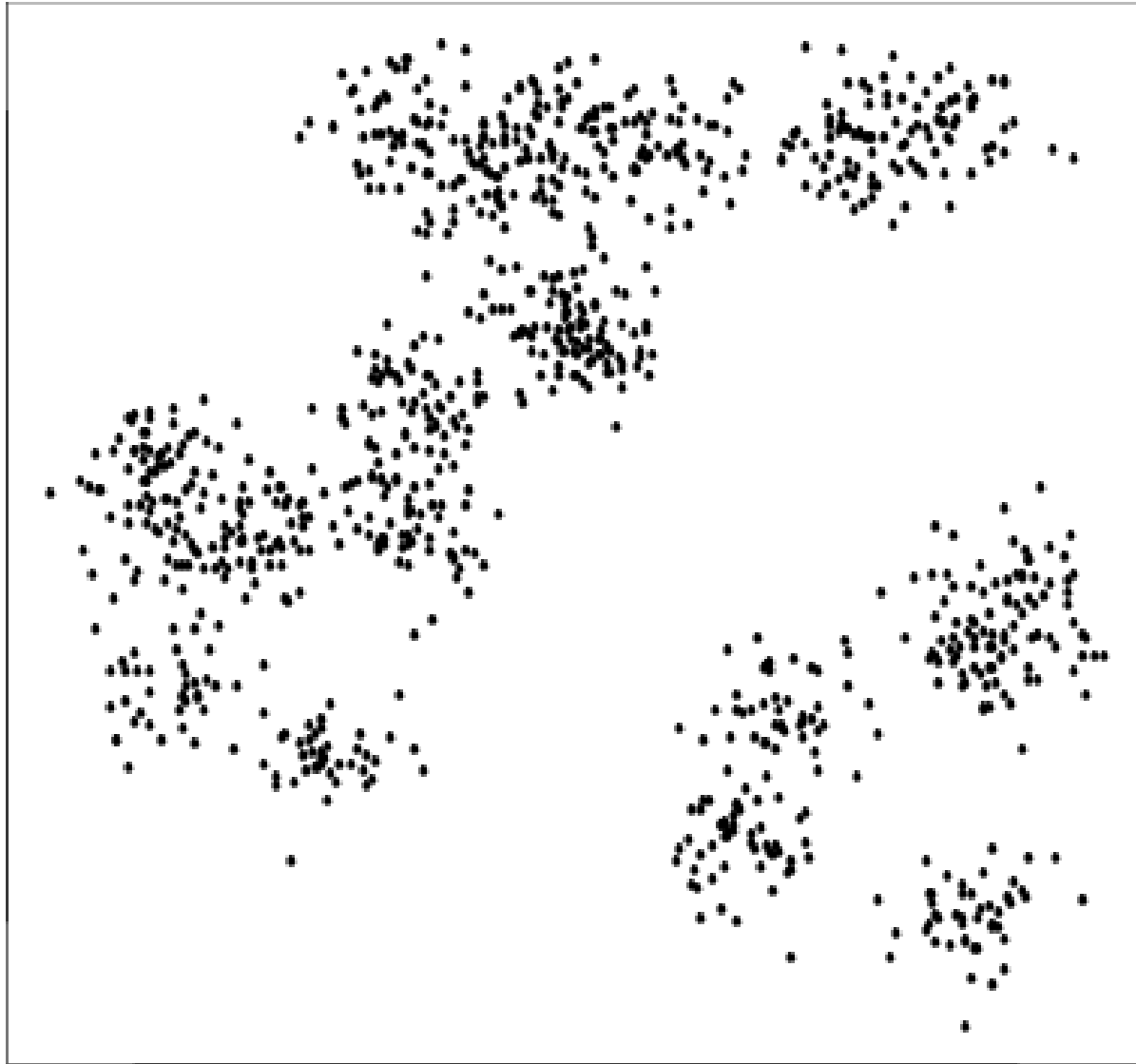
2) Unstructured random instances

- E.g., random distance matrices in the TSP, random 3-SAT instances
- May yield misleading results for practice

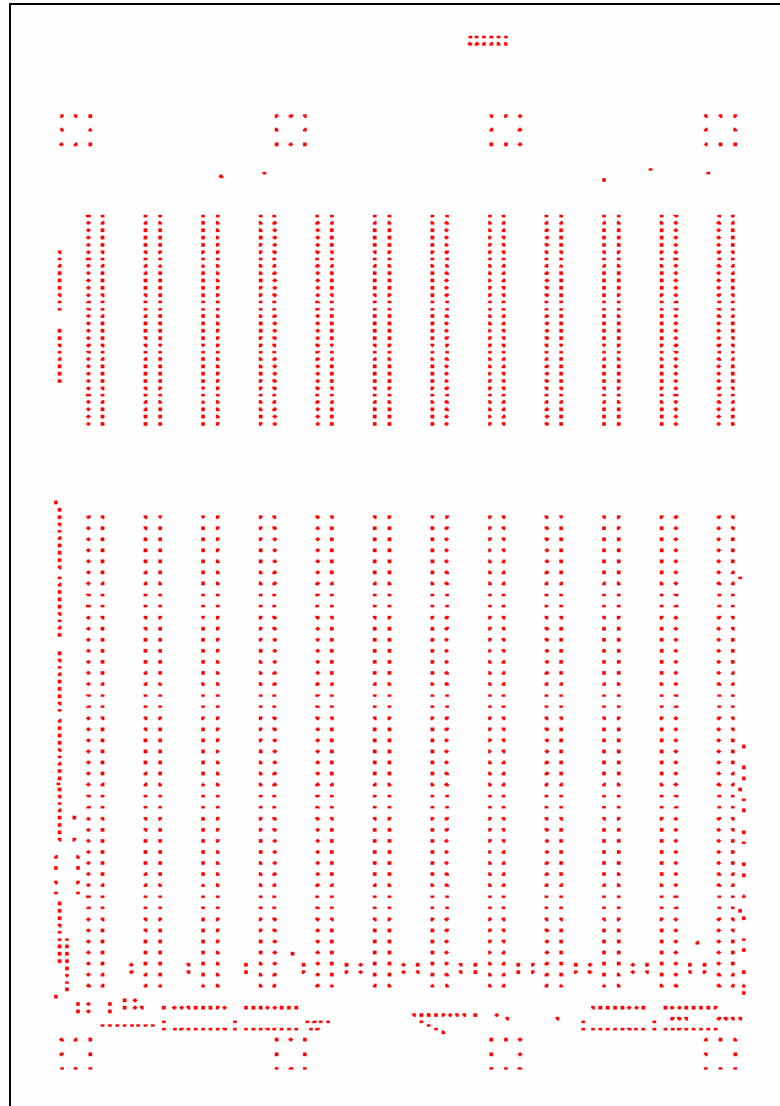
3) Structured random instances

- E.g., points uniform in the unit square under Euclidean metric for the TSP
- Need to be compared to real-world instances as sanity check
- Still better if they can be made available, but even large ones can be compactly distributed via generator codes and seeds.

dsj1000 from TSPLIB



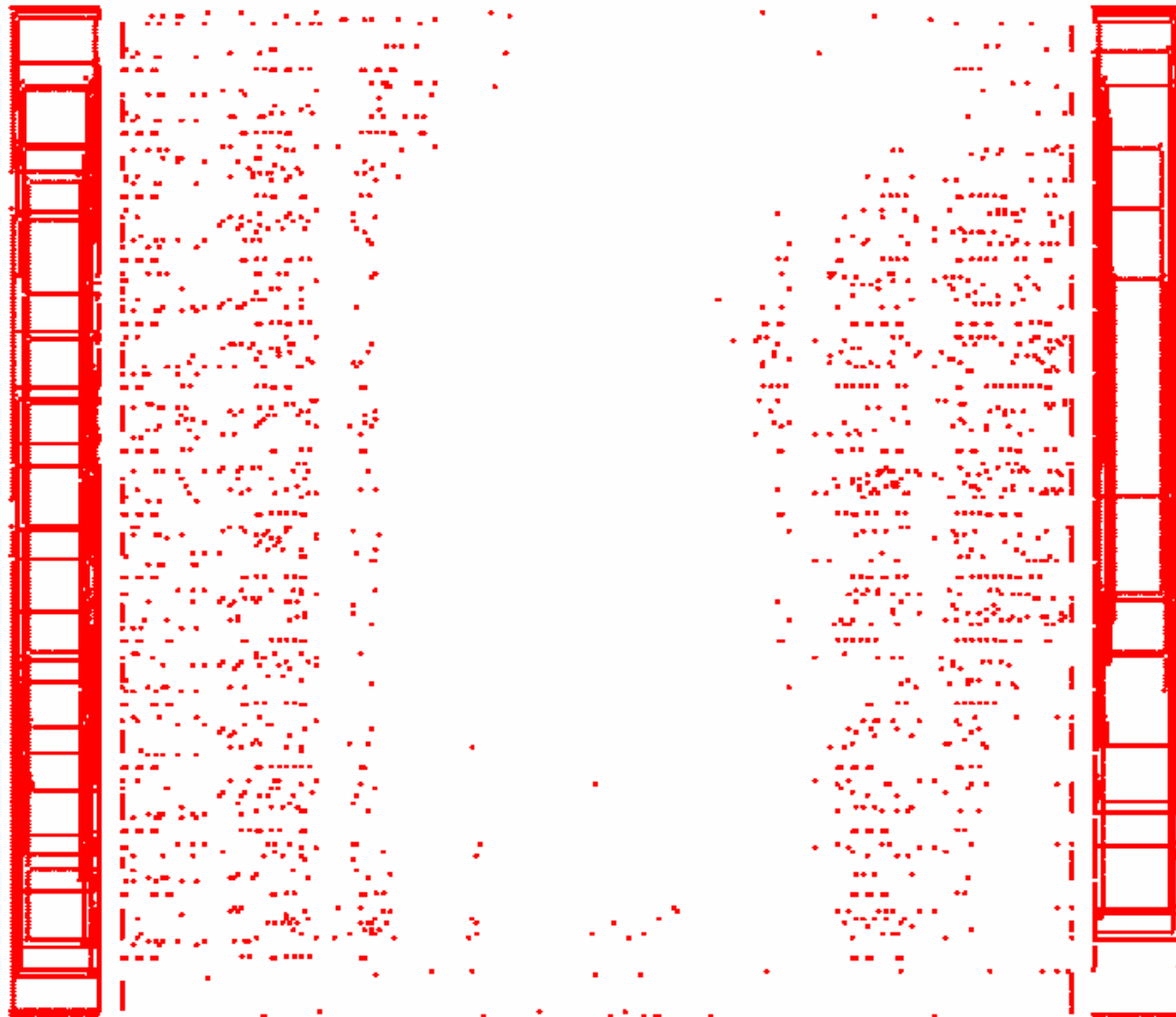
d2103 from TSPLIB



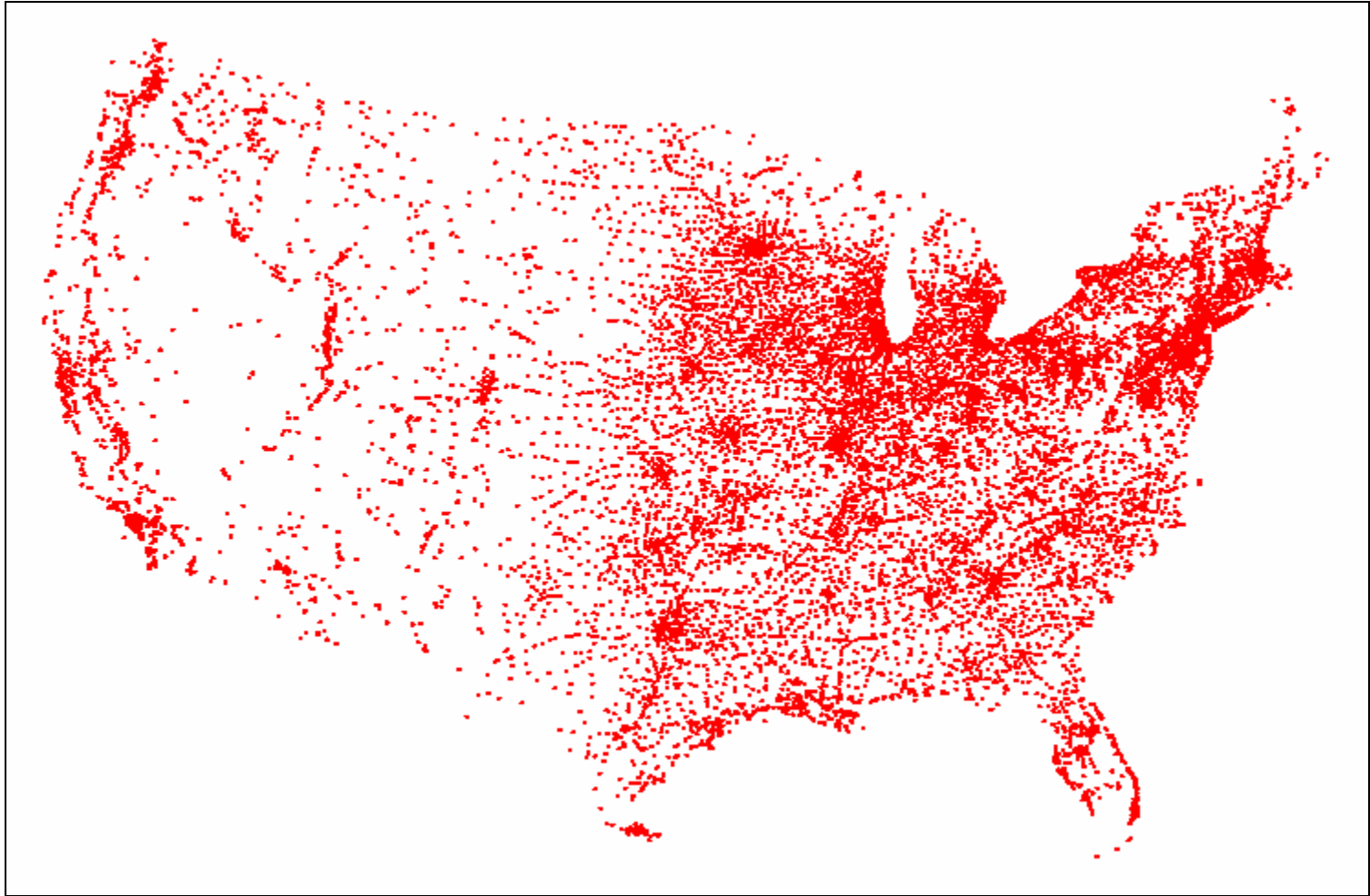
rl5915 from TSPLIB



pla7397 from TSPLIB (from AT&T)



usa13509 from TSPLIB



The red plots all taken from the Traveling Salesman Problem page

<http://www.tsp.gatech.edu/>

Reproducibility Issues

- Changes in the “computational apparatus” can have major effects on the quantities measured.

	Solution Quality	Running Time
Code	Major	Major
Language/Compiler	Possible Effects	Significant
Operating System	Possible Effects	Significant
Machine	Possible Effects	Major
Time of Run	No*	Can be Significant
Name of Code	No*	Can be Significant
Test Instances	Major	Major

Reproducibility Issues

- Changes in the “computational apparatus” can have major effects on the quantities measured.

	Solution Quality	Running Time
Code	Major	Major
Language/Compiler	Possible Effects	Significant
Operating System	Possible Effects	Significant
Machine	Possible Effects	Major
Time of Run	No*	Can be Significant
Name of Code	No*	Can be Significant
Test Instances	Major	Major
Comparison Standard	Major	----

Evaluating Solution Quality: Options

- 1) Simply report value of solution metric (cost, length, ...)
 - Valuable only if instance is also provided
 - Doesn't tell us how *good* the solution is.
- 2) Report ratio to optimal solution value
 - Requires that optimum be known
 - Can limit size and quantity of test instances
- 3) Report ratio to expected optimum (random instances)
 - Don't often know the expected optimum, may be asymptotic
- 4) Report ratio to best solution currently known
 - Moving target, but plausible if combined with (1).
- 5) Report ratio to computable bound on optimum
 - Example: Held-Karp bound with TSP
 - Don't have such bounds for many problems
- 6) Report ratio to result of well-defined algorithm.

Outline

- Classic "Reproducibility" and the Experimental Analysis of Algorithms
- The Traveling Salesman Problem (TSP) -- our running example
- "Comparability"
- More fun TSP stuff (time permitting)

- **Methodological Problem:** If we specify the computational apparatus sufficiently tightly to obtain reproducible results, these results will likely be far too narrow to be of interest.
- **One Solution:** Our hypotheses/conclusions should be about how the apparatus affects the measurements for the algorithm(s) in question, not about the measurements themselves, so we can predict performance in various scenarios.
- **Drawback:** Few of us have the time and effort to test multiple codes on multiple machines, large sets of instances in many classes, etc.
- **Challenge:** How do we write experimental papers today that will be of use to future researchers?

What does the future hold?

- Is the algorithm A you studied worth implementing (code worth finding and compiling) on my (presumably faster) machine to solve my (possibly much larger) instances?
- Is my new algorithm/code "better" than your algorithm/code A ?
- Is my implementation of algorithm A consistent with that in your paper?

Preliminary Judgment

Based on surveying the TSP literature for **The Traveling Salesman Problem: A Case Study in Local Optimization**, D. S. Johnson and L. A. McGeoch, in *Local Search in Combinatorial Optimization*, E. H. L. Aarts and J. K. Lenstra (editors), John-Wiley and Sons, Ltd., 1997, pp. 215-310, and continuing to follow the literature since then.

We still have a long way to go...

What is Needed

- Describe your algorithm in sufficient detail.
- Describe compilers, O.S., and machine in detail, and provide benchmark results for their composition that will aid in extrapolation.
- Run on a variety of instances, including as large ones as possible to help assess scalability of your results.
- Use reproducible instances and standards of comparison.
- Report running times, memory usage, and key operation counts.

Baby Steps: The DIMACS TSP Challenge

- Large range of instance types, sizes
 - Random Euclidean:
 - ten 1000-city, five 3162-city, three 10K-city, two 31K-city, two 100K-city, one 316K-, 1M-, 3M- and 10M-city
 - Random Clustered Euclidean:
 - ten 1000-city, five 3162-city, three 10K-city, two 31K-city, two 100K-city, one 316K-city
 - TSPLIB
 - 34 instances from 1000 to 85,900 cities
 - Random Distance Matrices
 - Four 1000-city, two 3162-city, and one 10K-city
- Data requested on tour lengths, running times, memory usage, machine, language, OS, etc.
- TSP-specific machine benchmarking

ALGORITHM: Benchmark Greedy**MACHINE: Silicon Graphics 196 Mhz MIPS R10000 [mips196]****RUN: 1****SUBMITTER: Johnson-McGeoch**

Raw Data			
Instance	Tour Length	Time in Seconds	Memory Usage
E1k.0	27494385	.04	--
E1k.1	26316209	.04	--
E1k.2	26530429	.04	--
E1k.3	27033547	.04	--
E1k.4	26467511	.04	--
E1k.5	26840234	.04	--
E1k.6	28297348	.04	--
E1k.7	27637102	.04	--
E1k.8	26645986	.04	--
E1k.9	27394231	.04	--
E3k.0	47391812	.13	--
E3k.1	47622395	.13	--
E3k.2	46051871	.13	--
E3k.3	47777886	.13	--
E3k.4	47134097	.13	--
E10k.0	83537327	.46	--
E10k.1	83168578	.46	--
E10k.2	82758628	.46	--
E31k.0	146526854	2.5	--
E31k.1	145422817	2.5	--
E100k.0	257807924	18	--
E100k.1	256821944	18	--

s.mips196

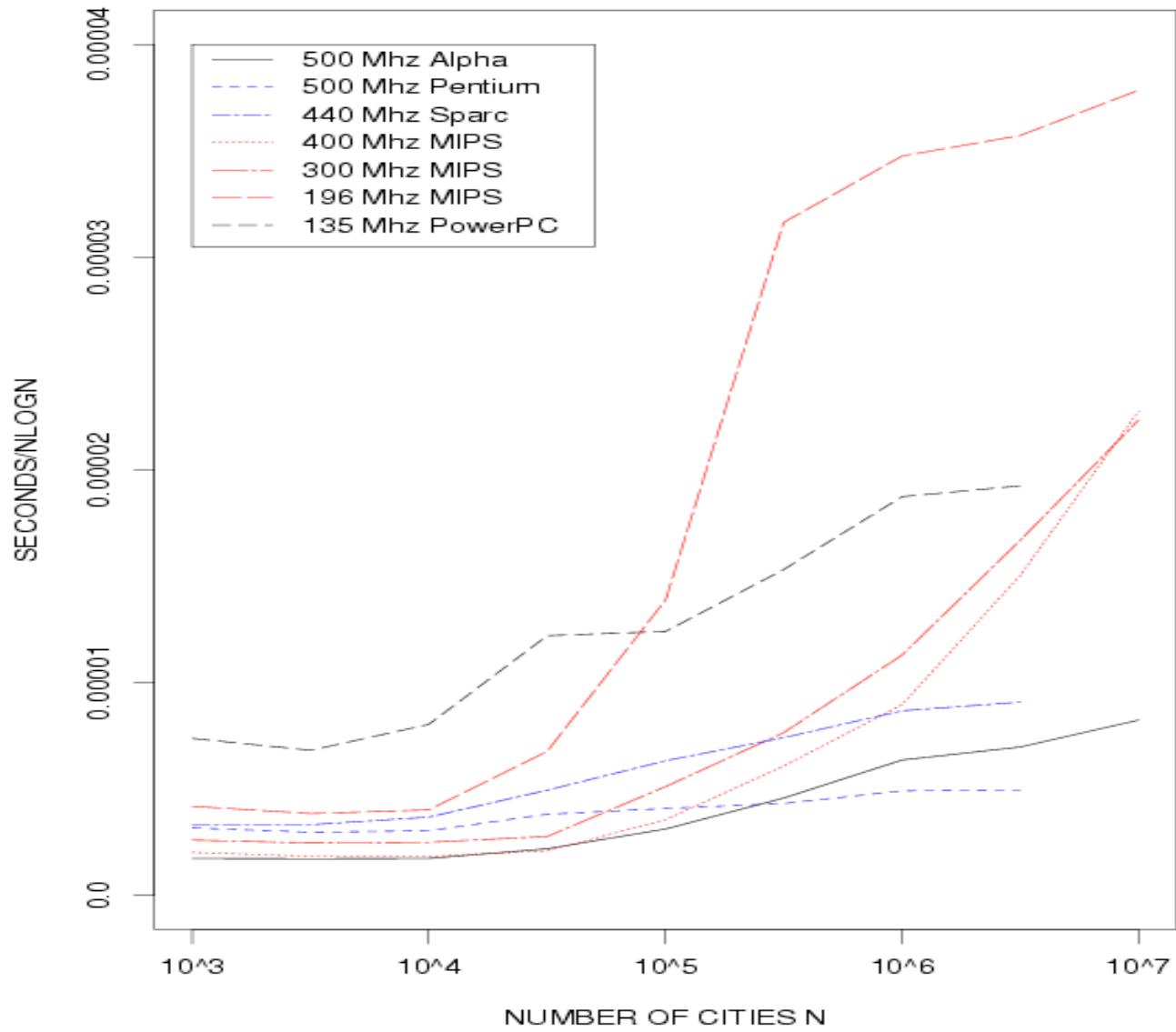
Cities

Runs

Total Seconds

1,000	1000	29
3,162	316	31
10,000	100	37
31,623	32	71
100,000	10	160
316,228	3	380
1,000,000	1	480
3,162,278	1	1690
10,000,000	1	6100

MACHINE SPEEDS



Normalization

- For given number of cities n , interpolate values u_n and a_n from the benchmark results for both your machine and the standard "target" DEC Alpha machine.
- Normalize your time for an n -city instance by dividing it by u_n/a_n

ALGORITHM: Benchmark Greedy

MACHINE: Silicon Graphics 196 Mhz MIPS R10000 [mips196]

RUN: 1

SUBMITTER: Johnson-McGeoch

Instance	Tour Length	Percent over OPT	Percent over HK	Running Time (Seconds)	Normalized Time	Memory Usage (Megabytes)
E1k.0	27494385	17.70	18.60	0.04	0.02	--
E1k.1	26316209	14.49	15.22	0.04	0.02	--
E1k.2	26530429	15.23	16.06	0.04	0.02	--
E1k.3	27033547	16.81	17.53	0.04	0.02	--
E1k.4	26467511	16.60	17.41	0.04	0.02	--
E1k.5	26840234	15.73	16.41	0.04	0.02	--
E1k.6	28297348	21.19	22.15	0.04	0.02	--
E1k.7	27637102	20.80	21.93	0.04	0.02	--
E1k.8	26645986	15.72	16.89	0.04	0.02	--
E1k.9	27394231	17.29	18.00	0.04	0.02	--
E3k.0	47391812	16.63	17.46	0.13	0.06	--
E3k.1	47622395	18.12	18.92	0.13	0.06	--
E3k.2	46051871	14.26	15.11	0.13	0.06	--
E3k.3	47777886	17.71	18.50	0.13	0.06	--
E3k.4	47134097	15.65	16.49	0.13	0.06	--
E10k.0	83537327	?	17.06	0.46	0.20	--
E10k.1	83168578	?	16.21	0.46	0.20	--
E10k.2	82758628	?	15.99	0.46	0.20	--
E31k.0	146526854	?	15.85	2.50	0.81	--
E31k.1	145422817	?	14.83	2.50	0.81	--
E100k.0	257807924	?	14.92	18.00	4.05	--



Algorithm Comparison Page

Algorithm 1

Algorithm 2

Spacefilling Curve



Instance Reading Only (196 Mhz MIPS)



Comparisons between Algorithm 1 and Algorithm 2

- Normalized Data
- Summary Table
- Charts

Include Results for Distance Matrix Instances in Chart? Yes



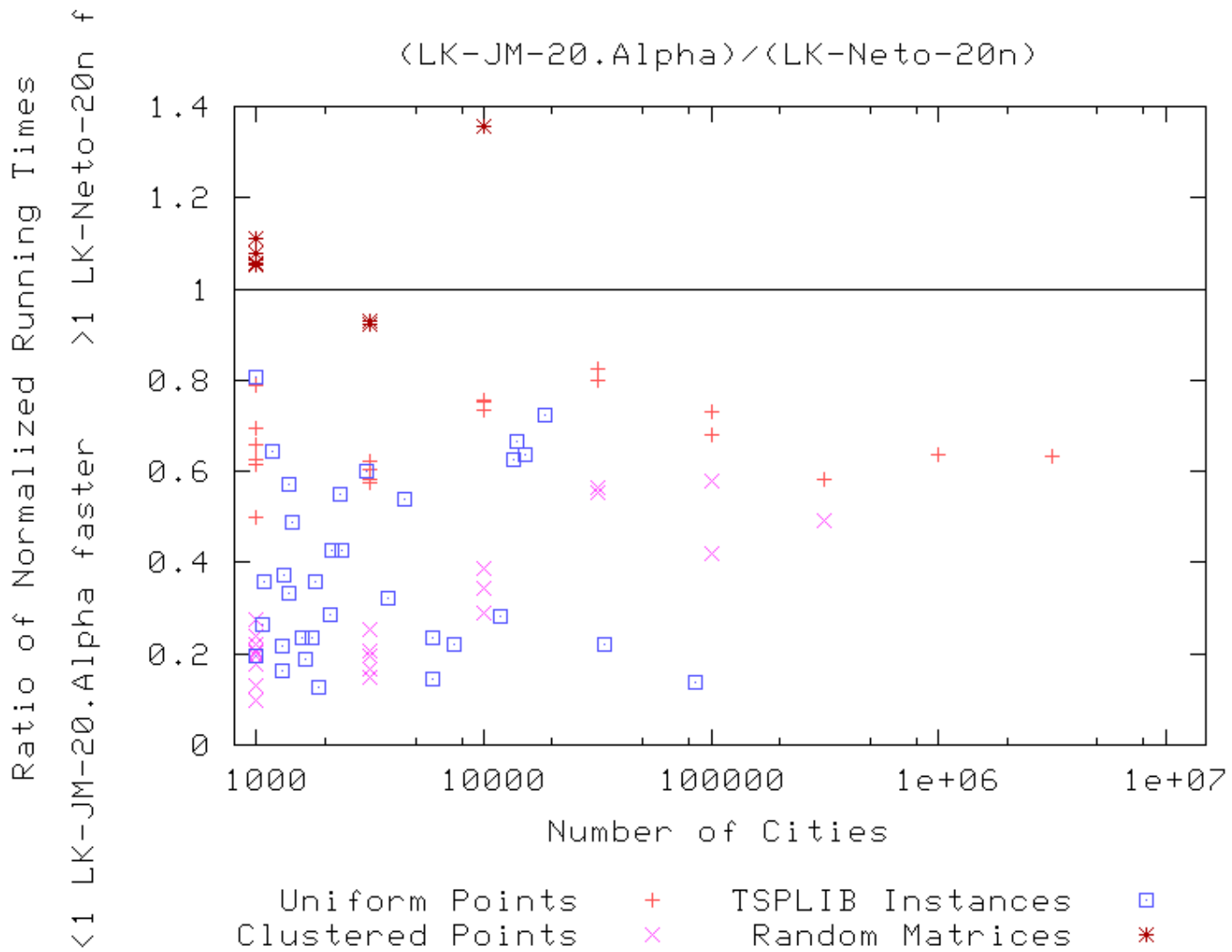
Results for

- Raw Data
- Normalized Data
- Summary Table
- Charts: Normalize Using Running Time Divisor Only Show Time Chart? No

Include Results for Distance Matrix Instances in Chart? Yes

[Instance Comparisons](#)[Challenge Homepage](#)[About the Challenge](#)[Download Page](#)[Results Page](#)

Johnson-McGeoch-LK (500 Mhz Dec Alpha) vs Neto-LK (1 Ghz Pentium III)



Average Percent Excess **LK-Neto-20q** Over **LK-JM-20.Alpha** and Ratio of Running Times **LK-Neto-20q / LK-JM-20.Alpha**

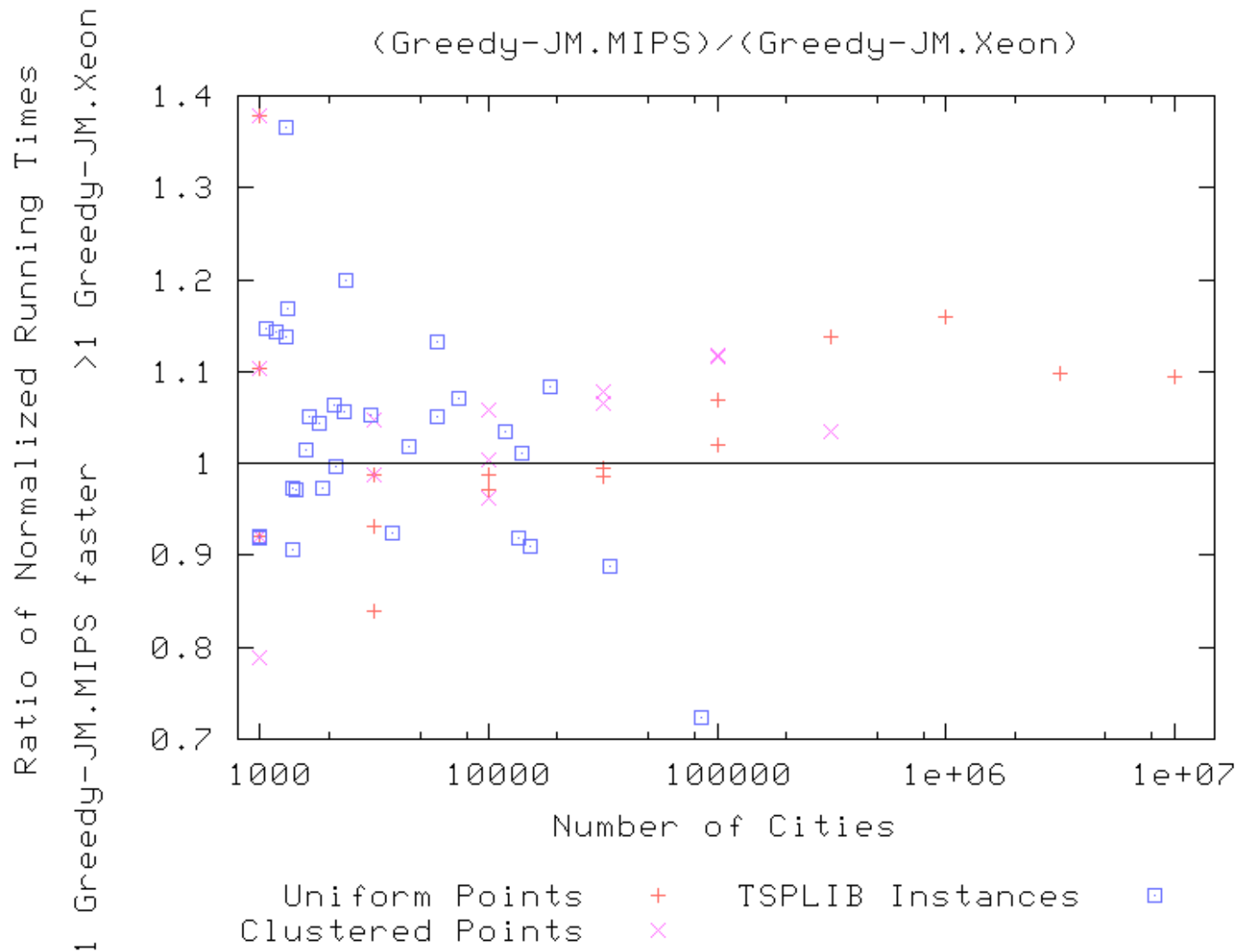
(Excess Entry > 0 Means **LK-JM-20.Alpha** Better. Running Time Entry > 1 Means **LK-JM-20.Alpha** is Faster)

	1000	3162	10K	31K	100K	316K	1M	3.16M	10M
Average Percent Excess									
Uniform Points	0.05	-0.07	-0.10	-0.12	-0.01	0.02	-0.04	-0.04	--
Clustered Points	0.56	0.93	1.42	0.77	0.47	0.41			
TSPLIB Instances	-0.04	0.08	-0.13	0.34	0.37				
Random Matrices	--	--	--						
Average Ratio of Normalized Running Times									
Uniform Points	1.8	2.1	1.9	2.3	3.5	5.8	7.6	9.2	--
Clustered Points	5.2	5.6	3.7	2.2	2.6	2.9			
TSPLIB Instances	2.3	2.5	3.6	3.0	7.1				
Random Matrices	--	--	--						

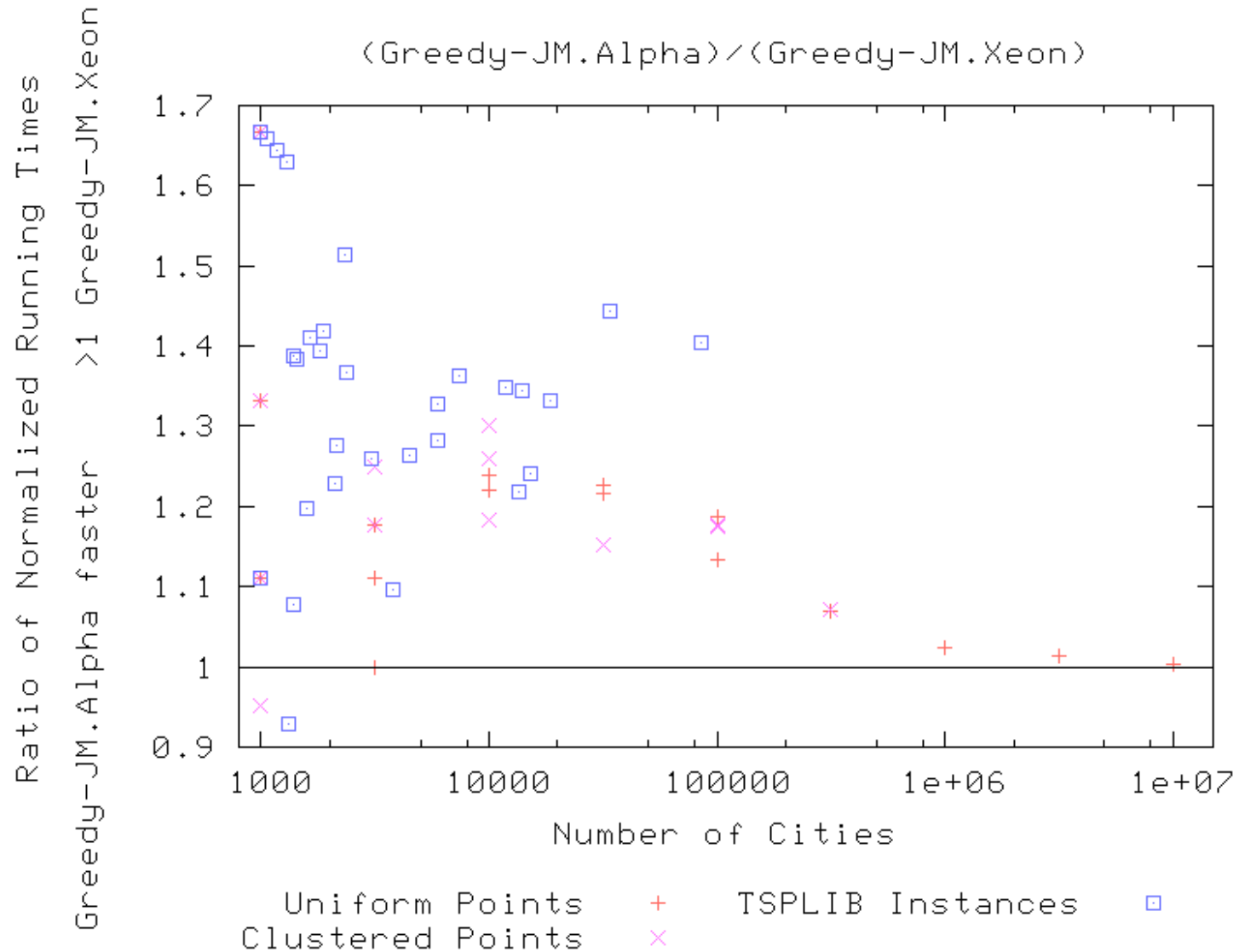
TSPLIB entries are averages for the following instances:

N=1000	pr1002, pcb1173, rl1304, nrw1379
N=3162	pr2392, pcb3038, fnl4461
N=10k	pla7397, brd14051
N=31k	pla33810
N=100k	pla85900

Benchmark vs Benchmark: 196 Mhz MIPS vs 2.6 Ghz Intel Xeon



Benchmark vs Benchmark: 500 Mhz Alpha vs 2.6 Ghz Intel Xeon

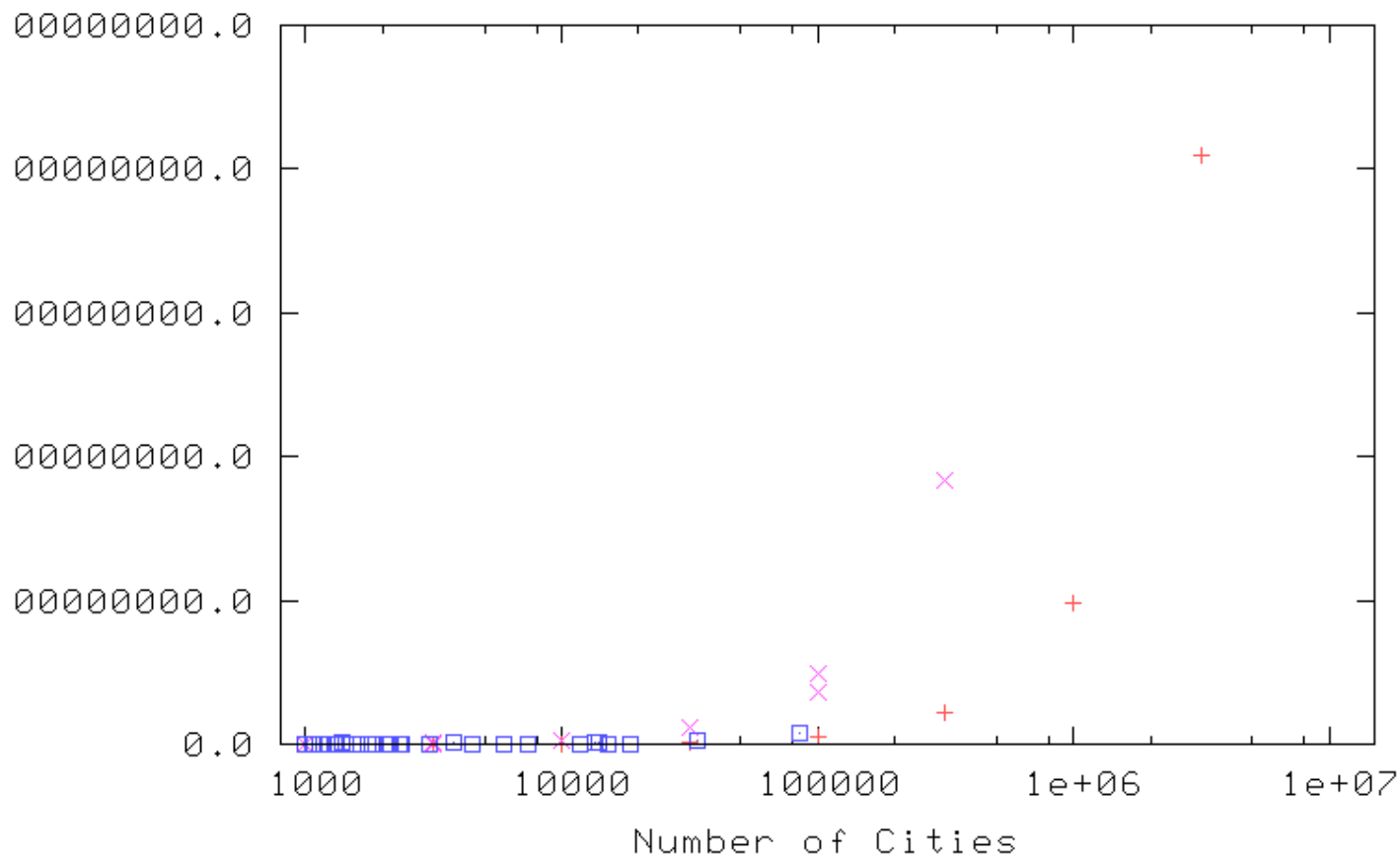


Results for

- Raw Data
- Normalized Data
- Summary Table
- Charts: Normalize Using Running Time Divisor Only Show Time Chart?

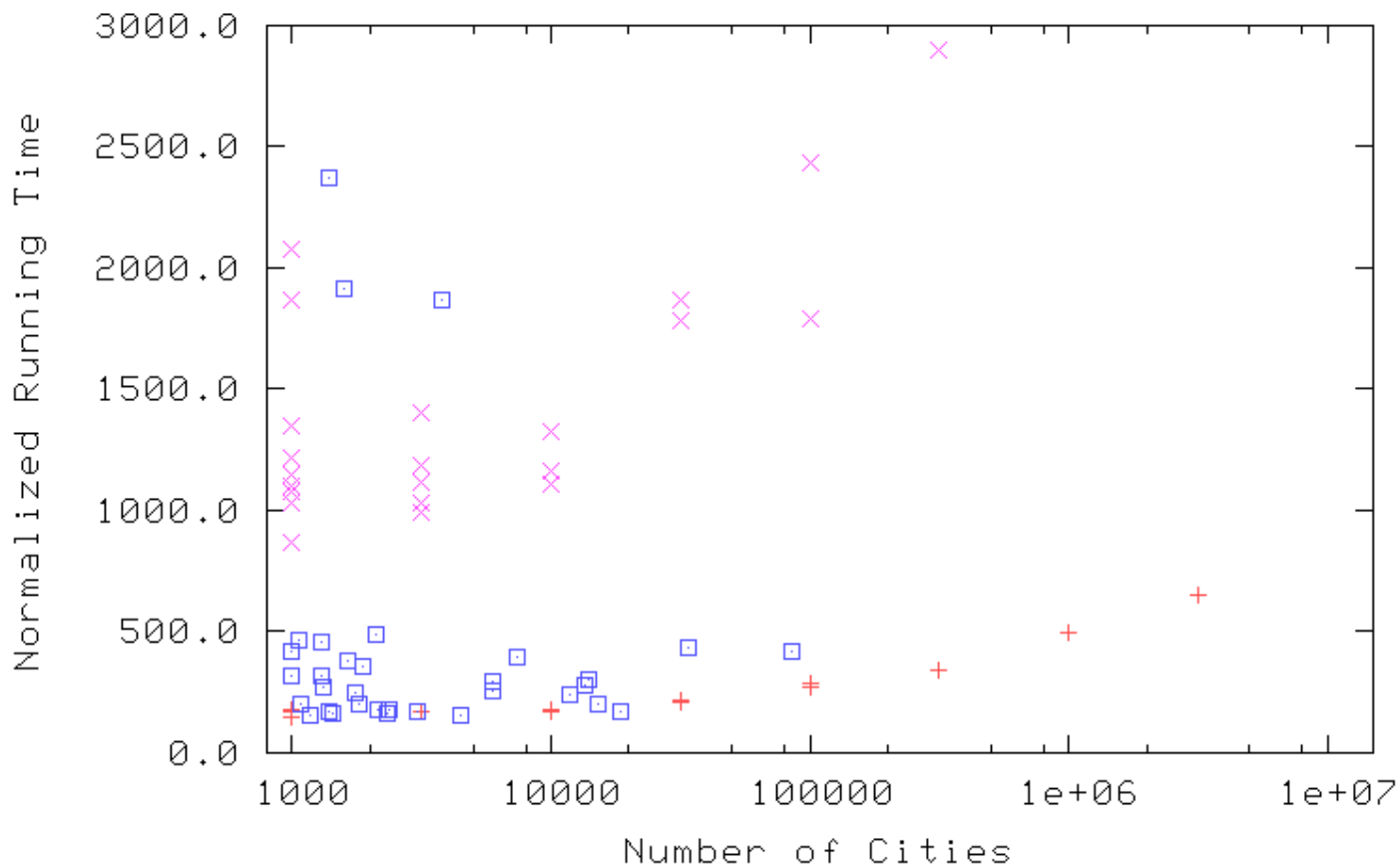
Include Results for Distance Matrix Instances in Chart?

LK-JM-20.Alpha : Microseconds/1



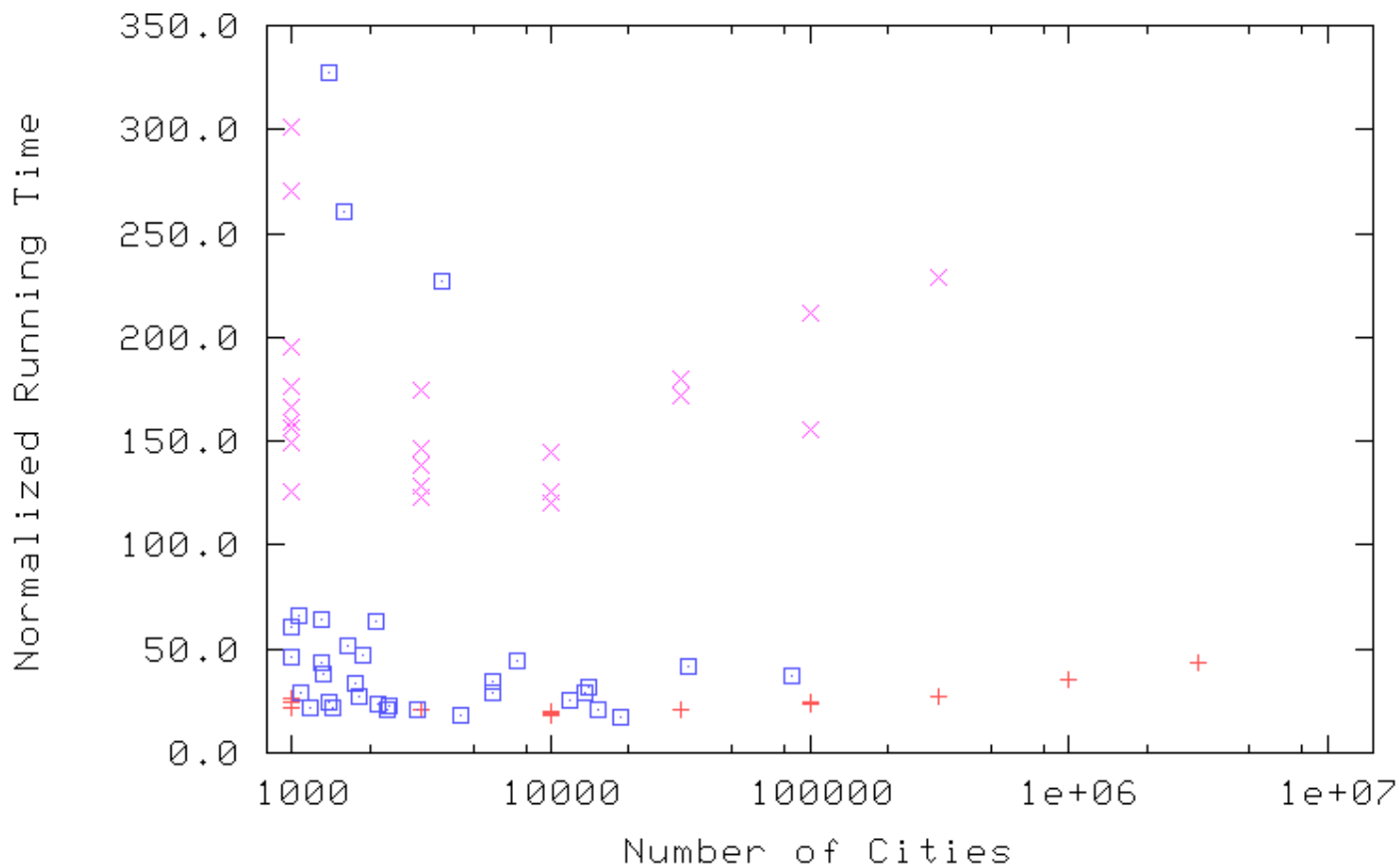
Uniform Points + TSPLIB Instances □
Clustered Points x

LK-JM-20.Alpha : Microseconds/N



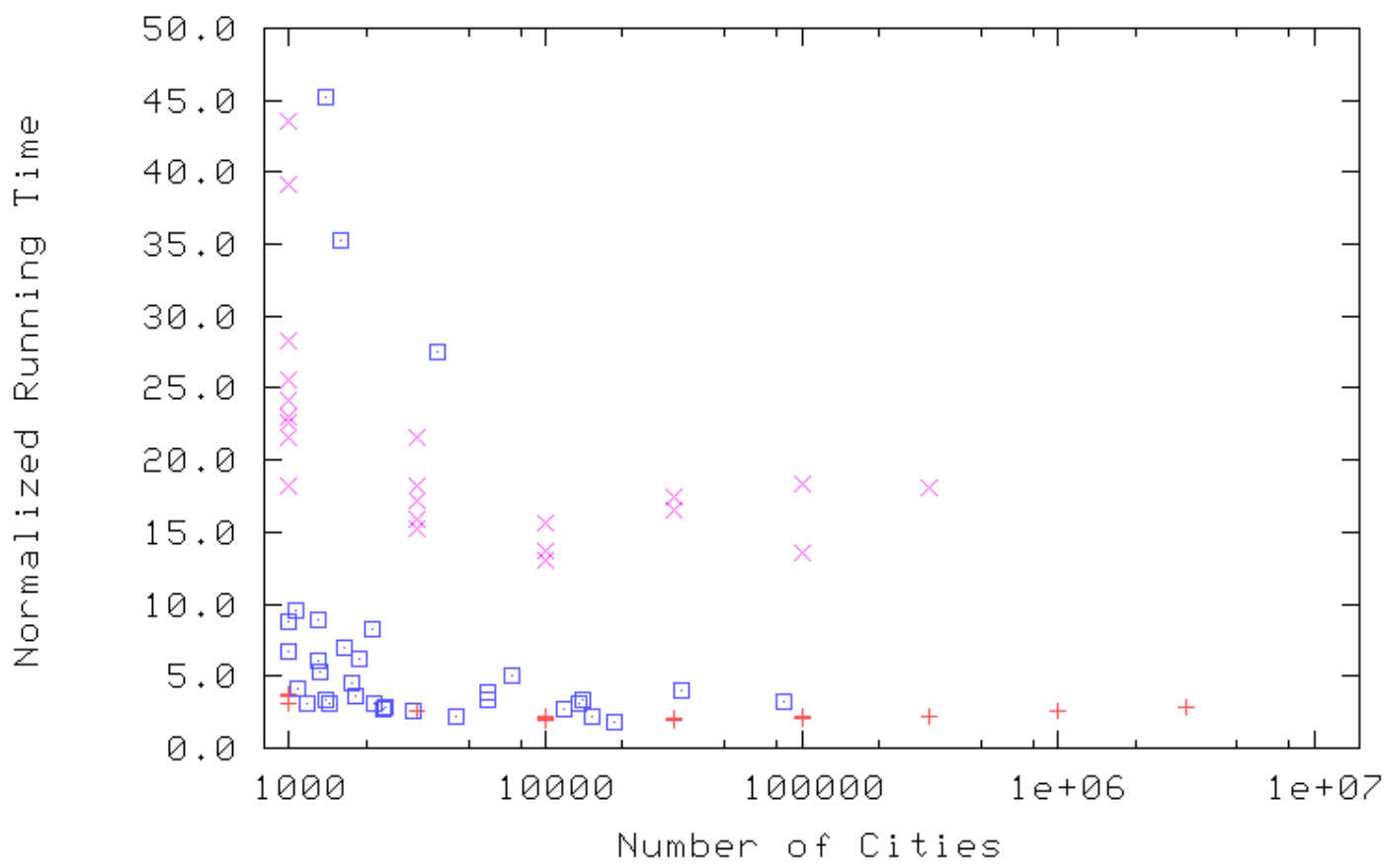
Uniform Points + TSPLIB Instances □
Clustered Points ×

LK-JM-20.Alpha : Microseconds/NlogN



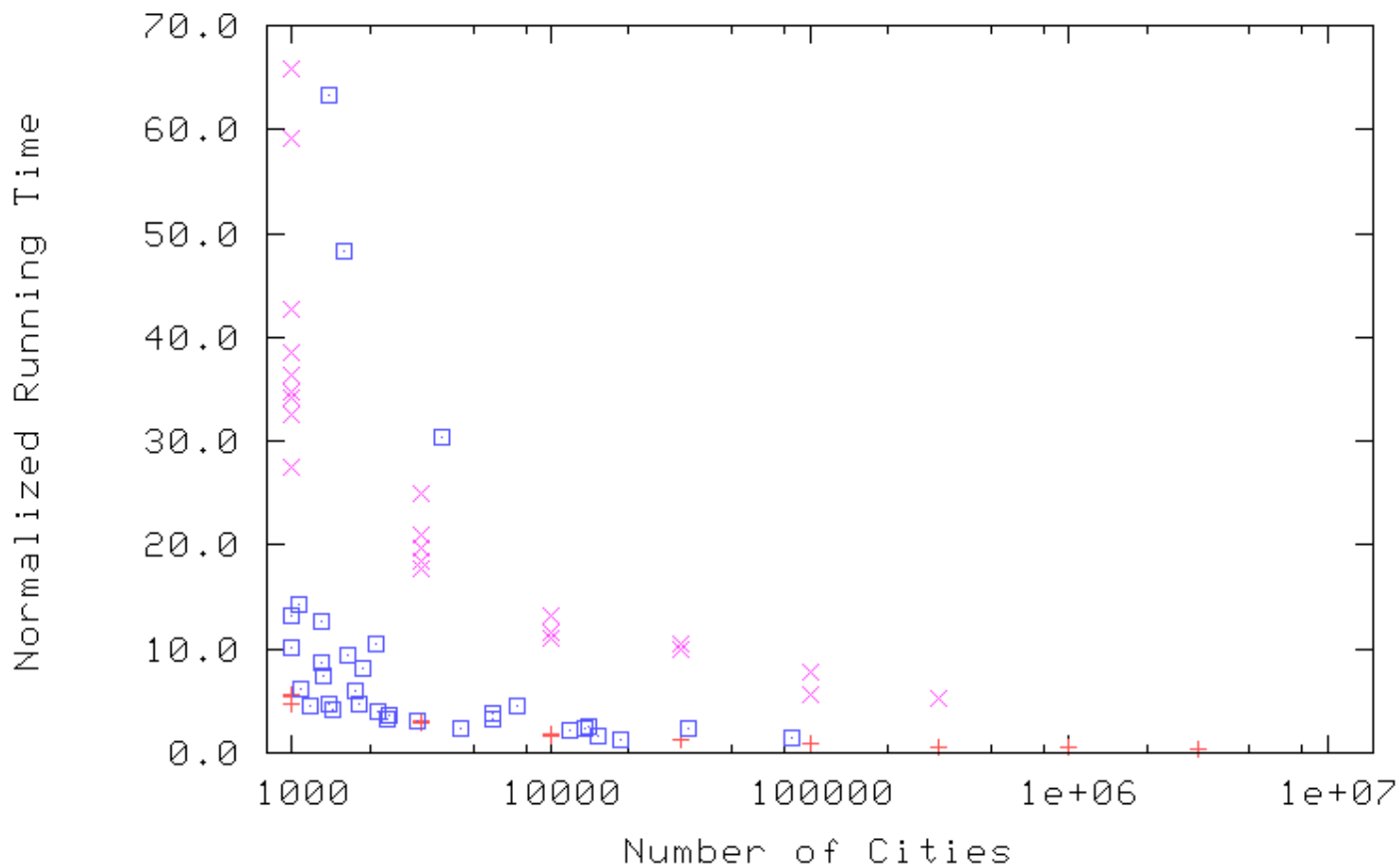
Uniform Points + TSPLIB Instances □
Clustered Points ×

LK-JM-20.Alpha : Microseconds/Nlog²N



Uniform Points + TSPLIB Instances □
Clustered Points x

LK-JM-20.Alpha : Microseconds/ $N^{1.5}$



Uniform Points + TSPLIB Instances \square
Clustered Points \times

Outline

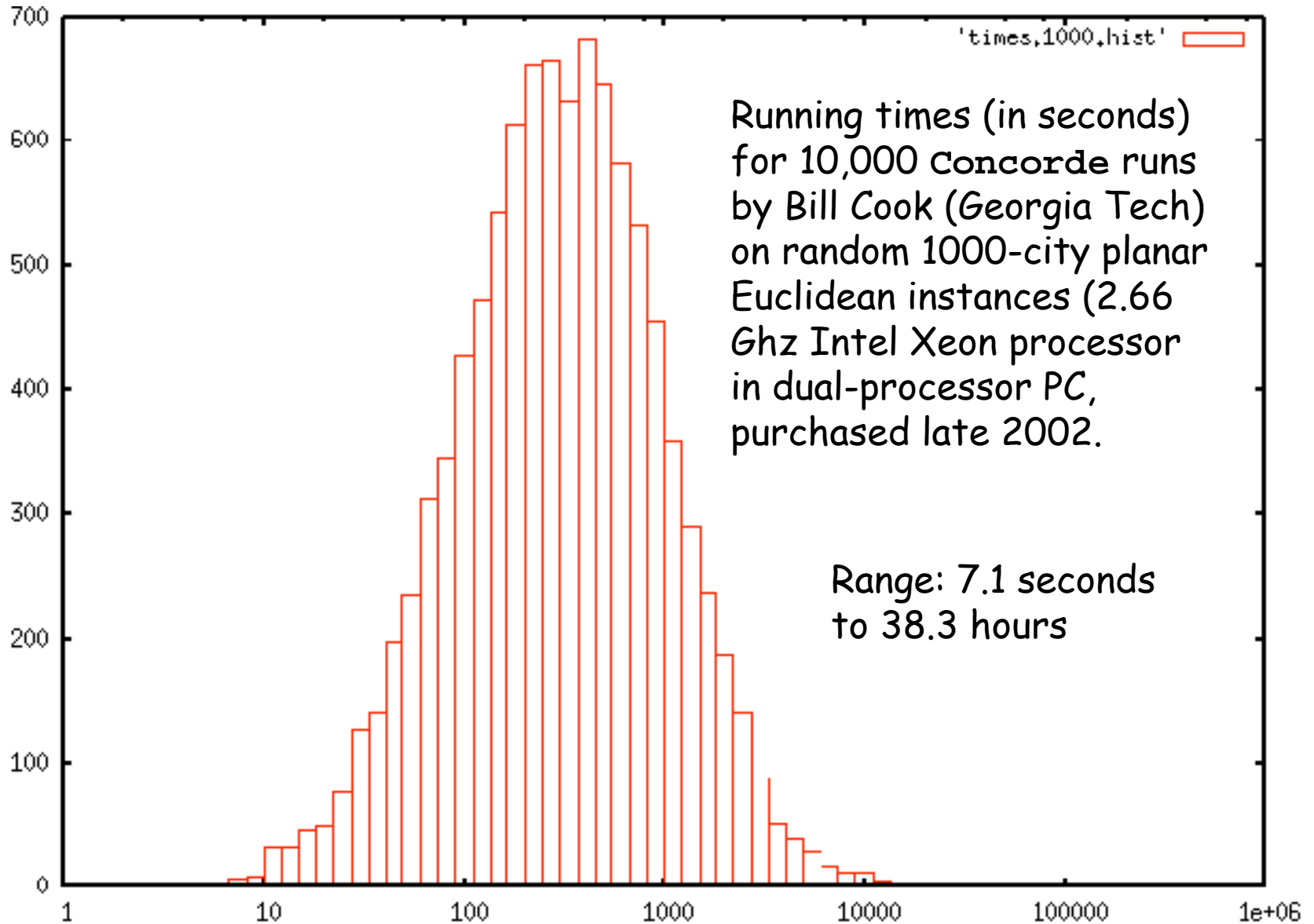
- Classic "Reproducibility" and the Experimental Analysis of Algorithms
- The Traveling Salesman Problem (TSP) -- our running example
- "Comparability"
- More fun TSP stuff (time appears to permit)

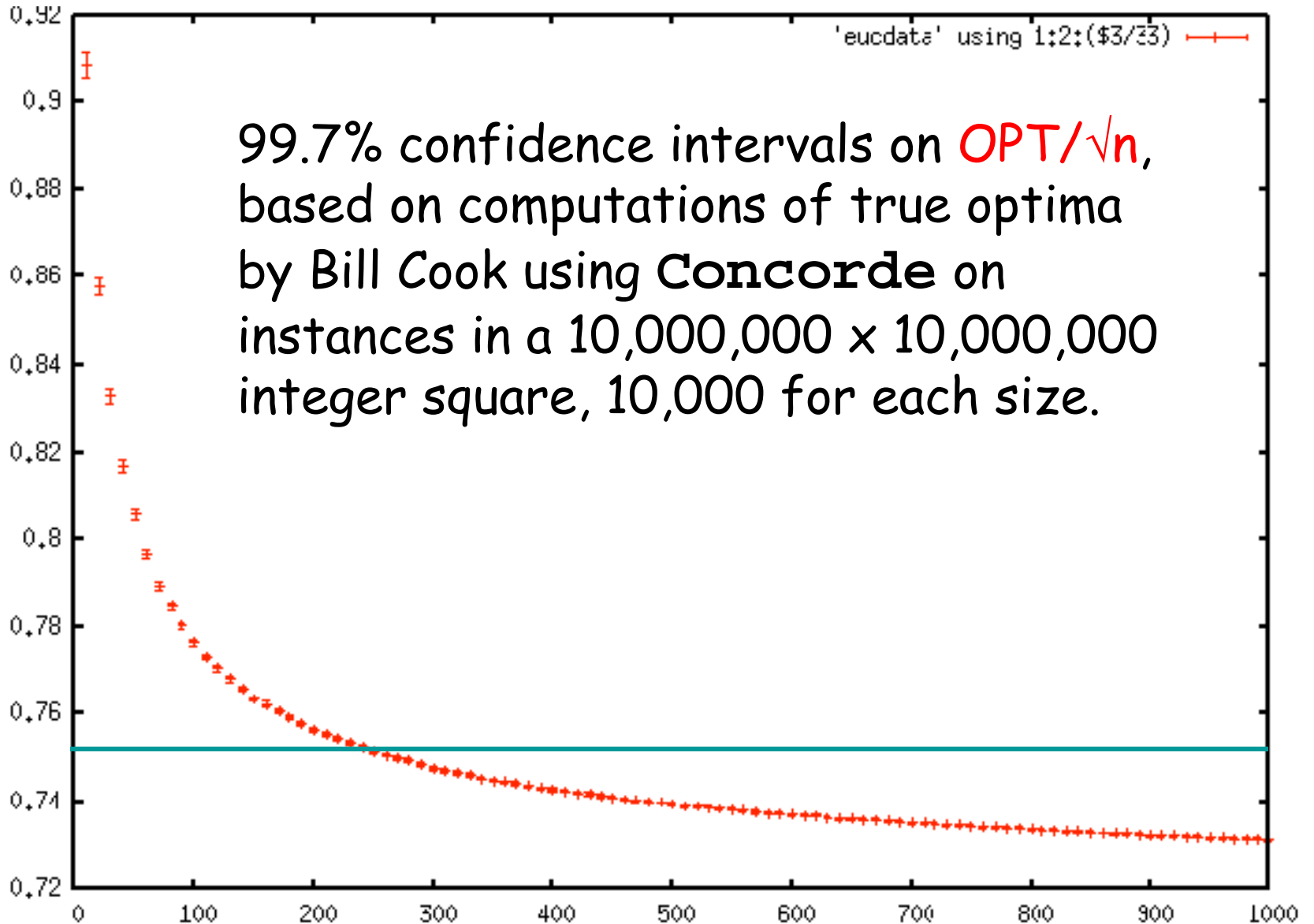
Standards of Comparison, Revisited

- Points uniform in the unit square, expected optima
- Beardwood, Halton, and Hammersley [1959]:
 - Optimal tour length for an n -city instances approaches $C\sqrt{n}$ for some constant C as $n \rightarrow \infty$
 - Estimated $C \approx .75$, based on hand solutions for a 202-city and a 400-city instance.
 - Subsequent supposedly more-accurate estimate due to Stein [1977] based on computer simulations claimed $C \approx .765$
- Many researchers subsequently used $(.765)\sqrt{n}$ as a surrogate for the optimal tour length.
- Is this realistic?

What does today's data tell us?

- Much faster machines, much better algorithms
- The Concorde optimization code and its relatives can now find optimal solutions for 1,000-city random Euclidean instances with (relative) ease.
- Bill Cook (Georgia Tech), one of the developers of Concorde, has generated lots of data for us to play with.
- See <http://www2.isye.gatech.edu/~wcook/beta/>

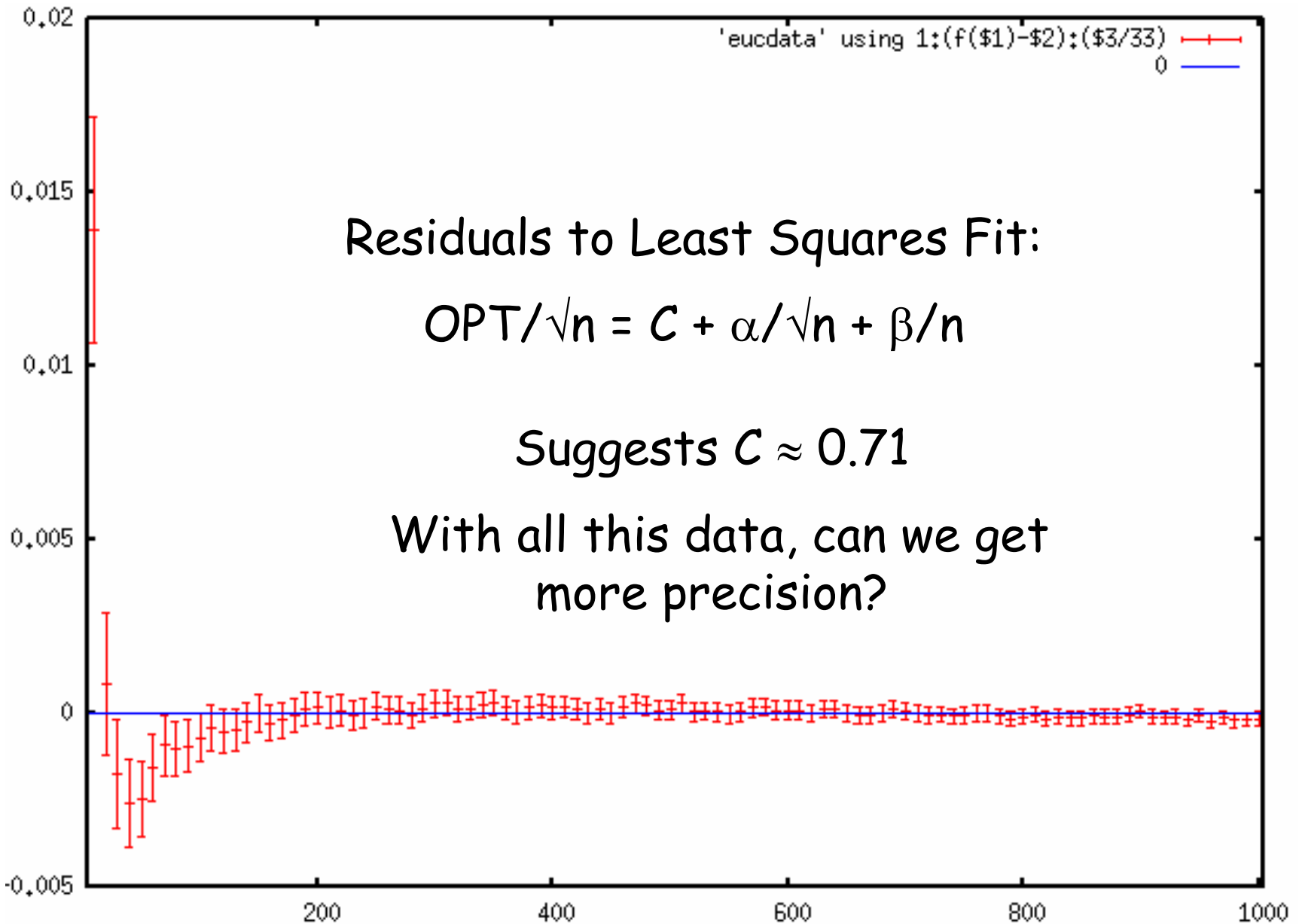




Fits what function?

- Expected distance to nearest neighbor proportional to $1/\sqrt{n}$, times n cities yields $\Theta(\sqrt{n})$
- $O(\sqrt{n})$ cities close to the boundary are missing some neighbors, for an added contribution proportional to $(\sqrt{n})(1/\sqrt{n})$, or $\Theta(1)$
- A constant number of cities are close to two boundaries (at the corners of the square), which may add an additional $\Theta(1/\sqrt{n})$
- This yields target function

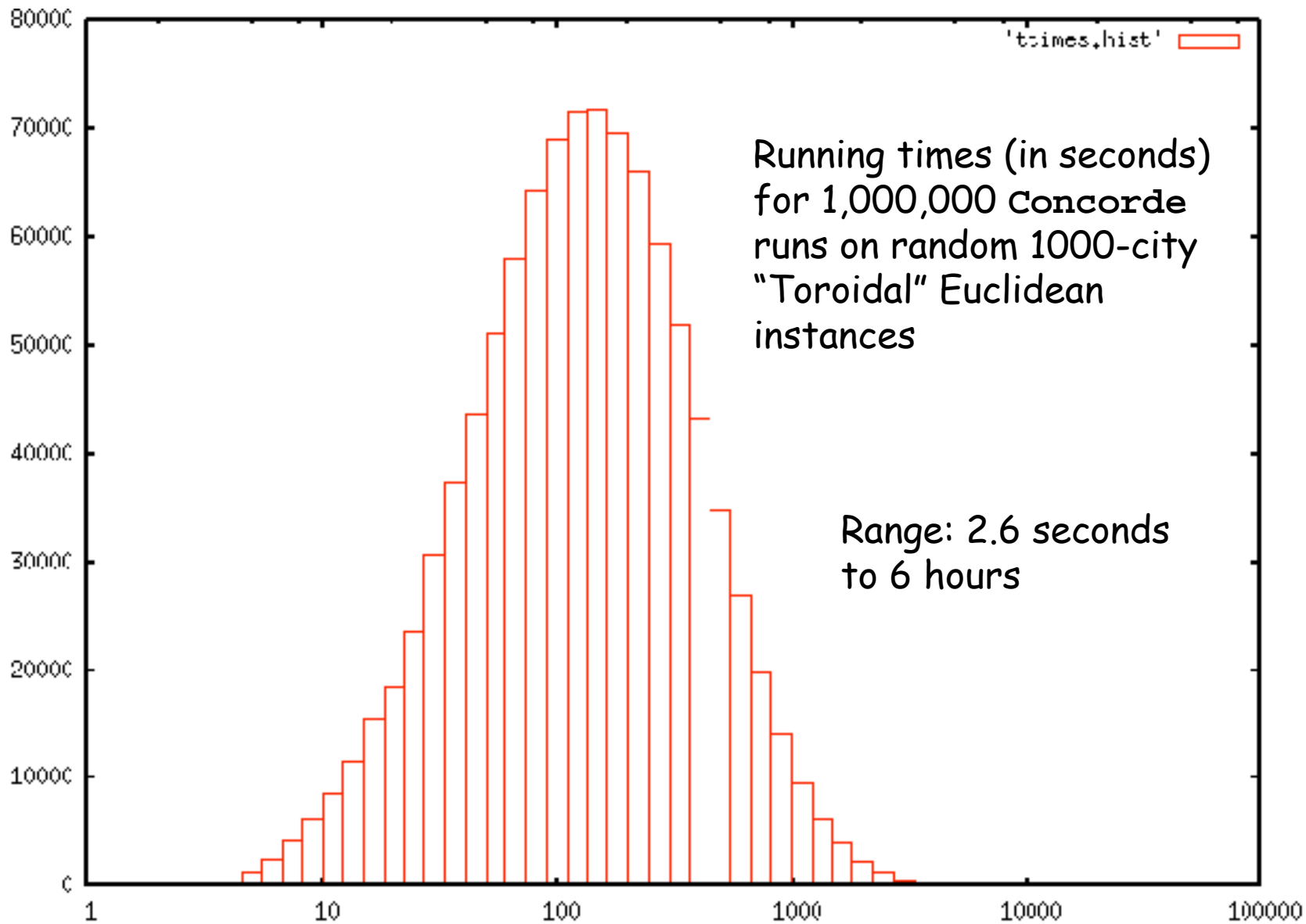
$$OPT/\sqrt{n} = C + \beta/\sqrt{n} + \gamma/n$$

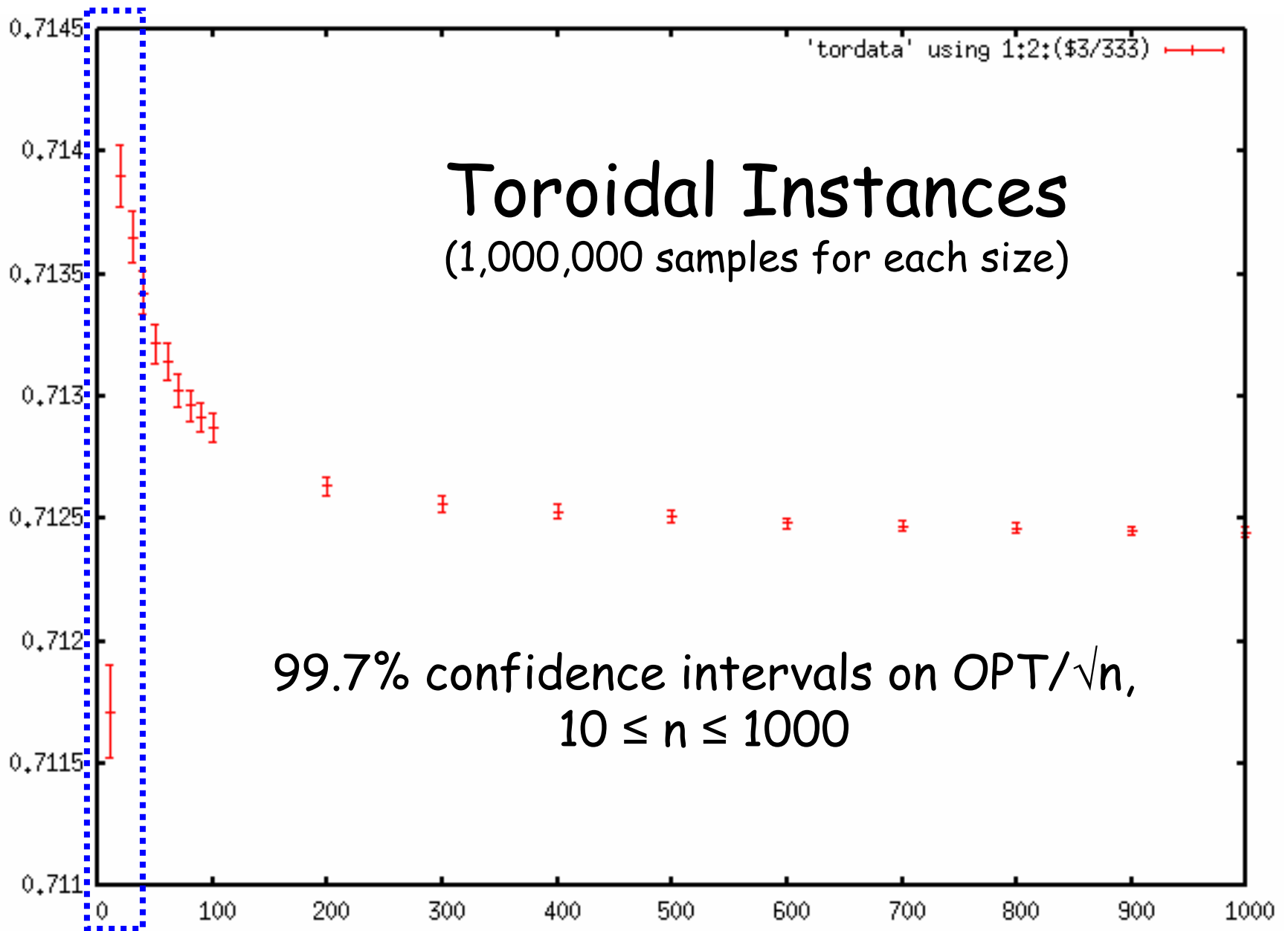


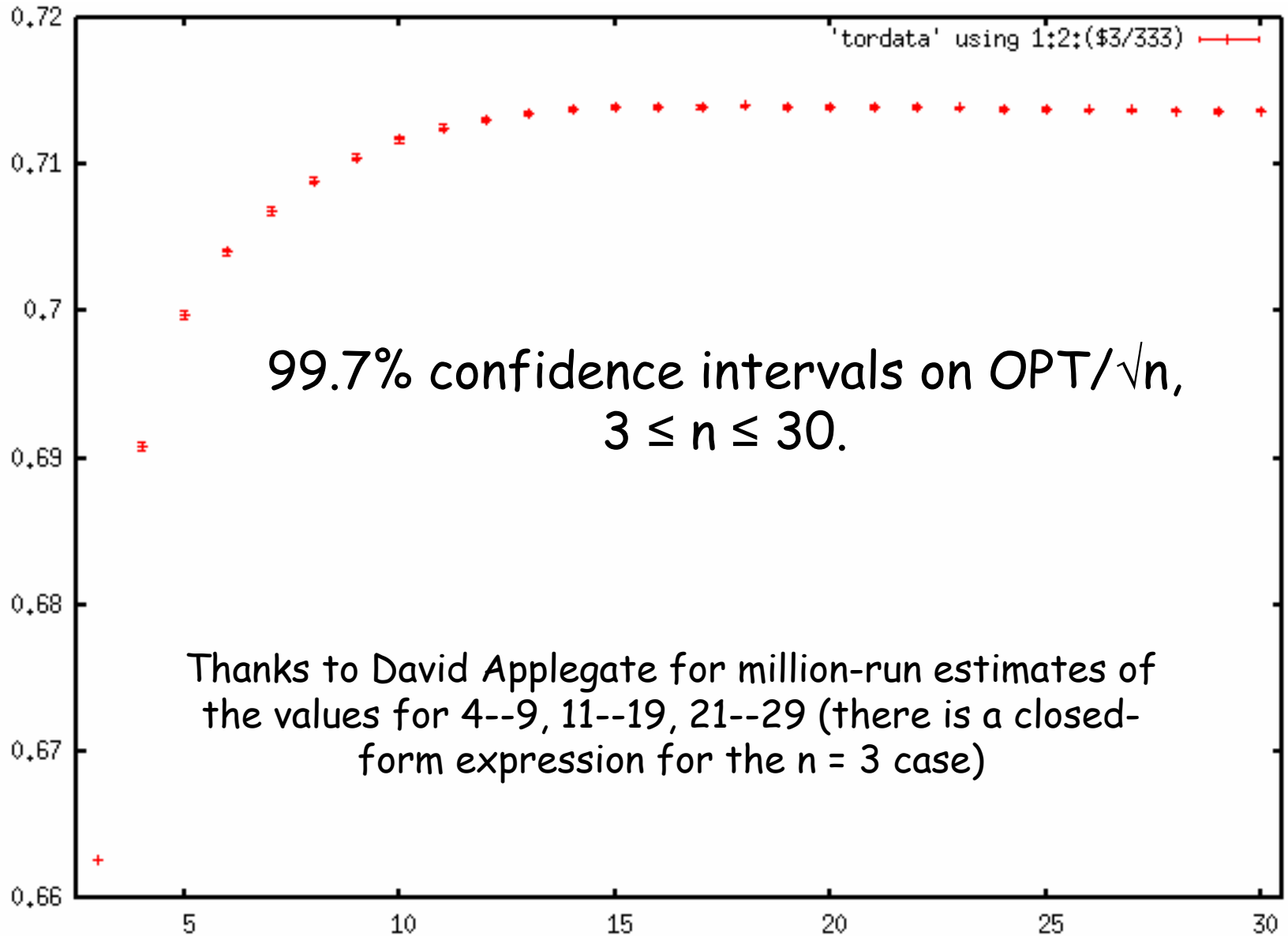
How to highlight correlations with n that aren't caused by the boundary:

"Toroidal" Instances

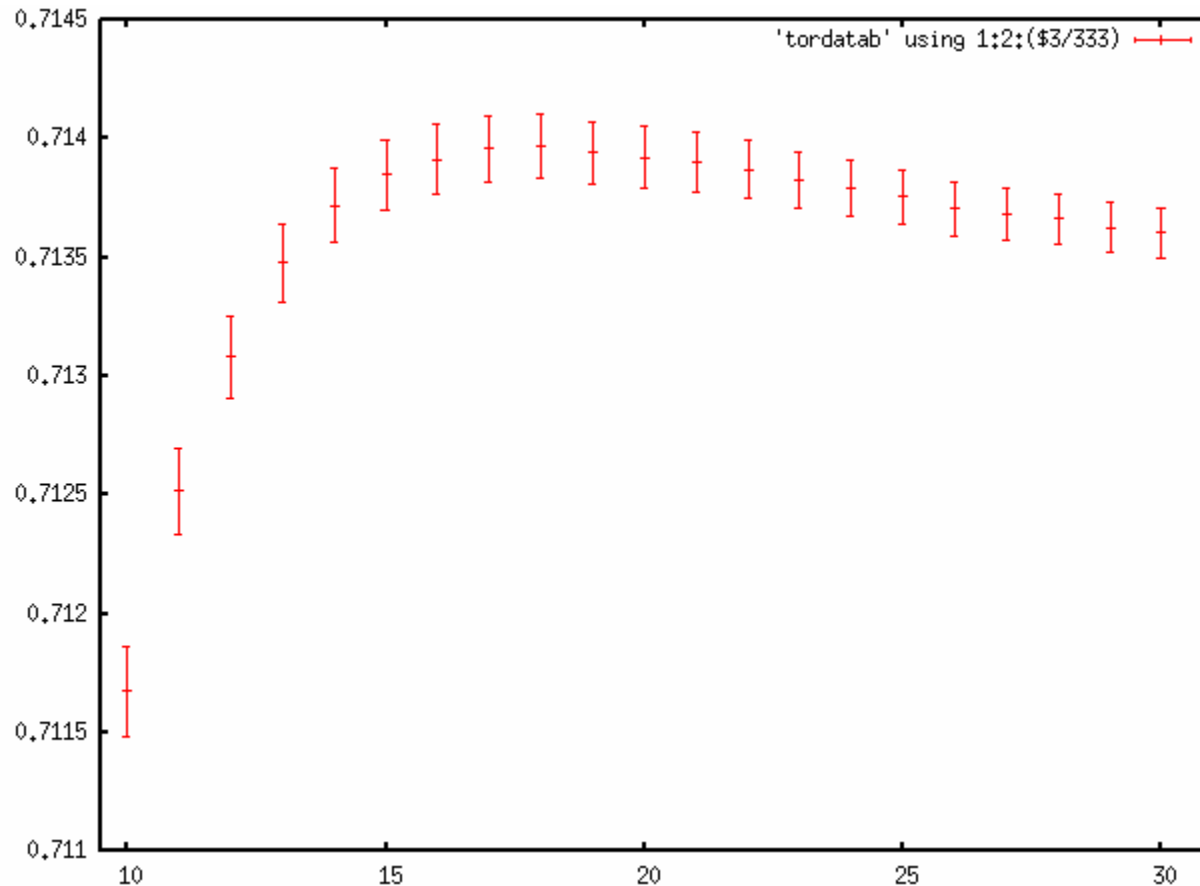
- Join left boundary of the unit square to the right boundary, top to the bottom
- Same asymptotic constant for $E[\text{OPT}/\sqrt{n}]$ as for planar instances [Jaillet, 1992]
- Somewhat easier to solve - Bill Cook solved 1,000,000 random instances for each n





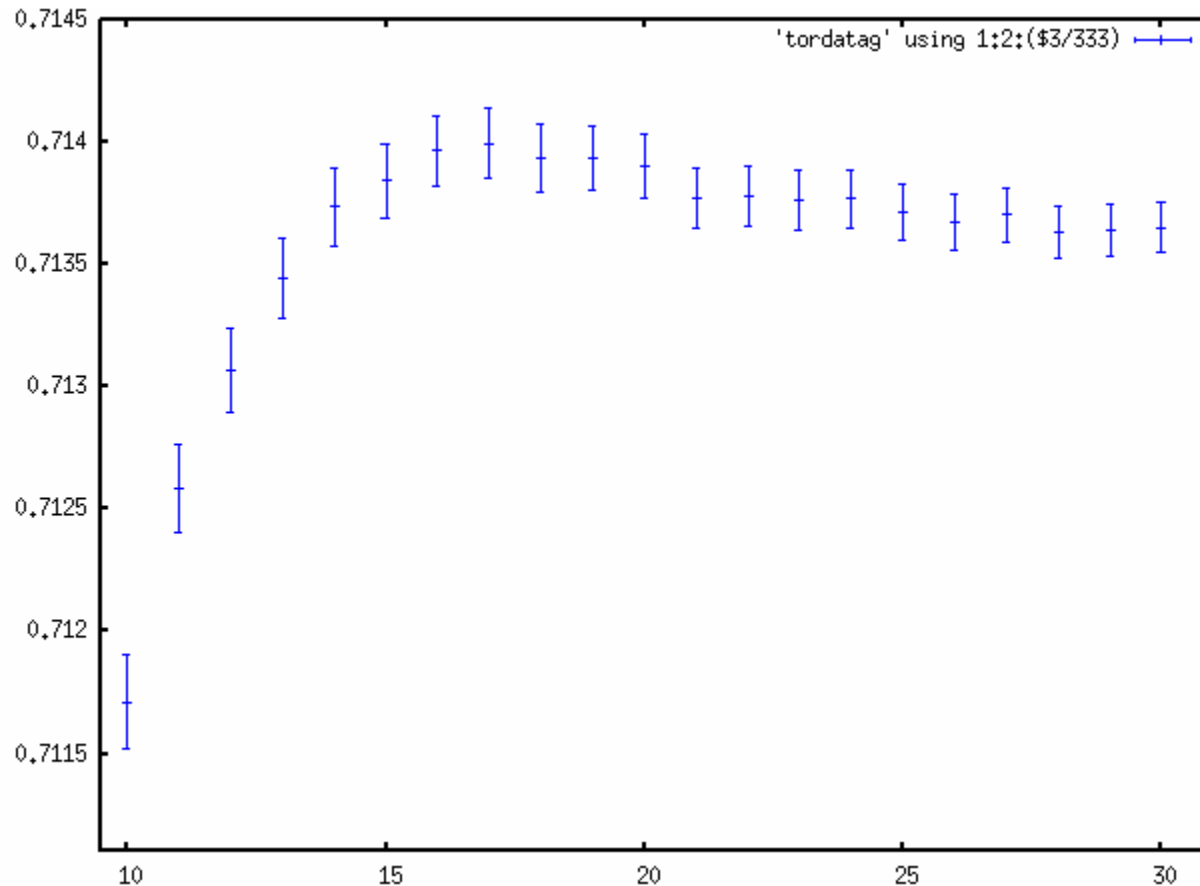


1st Random Instance Generation Scheme

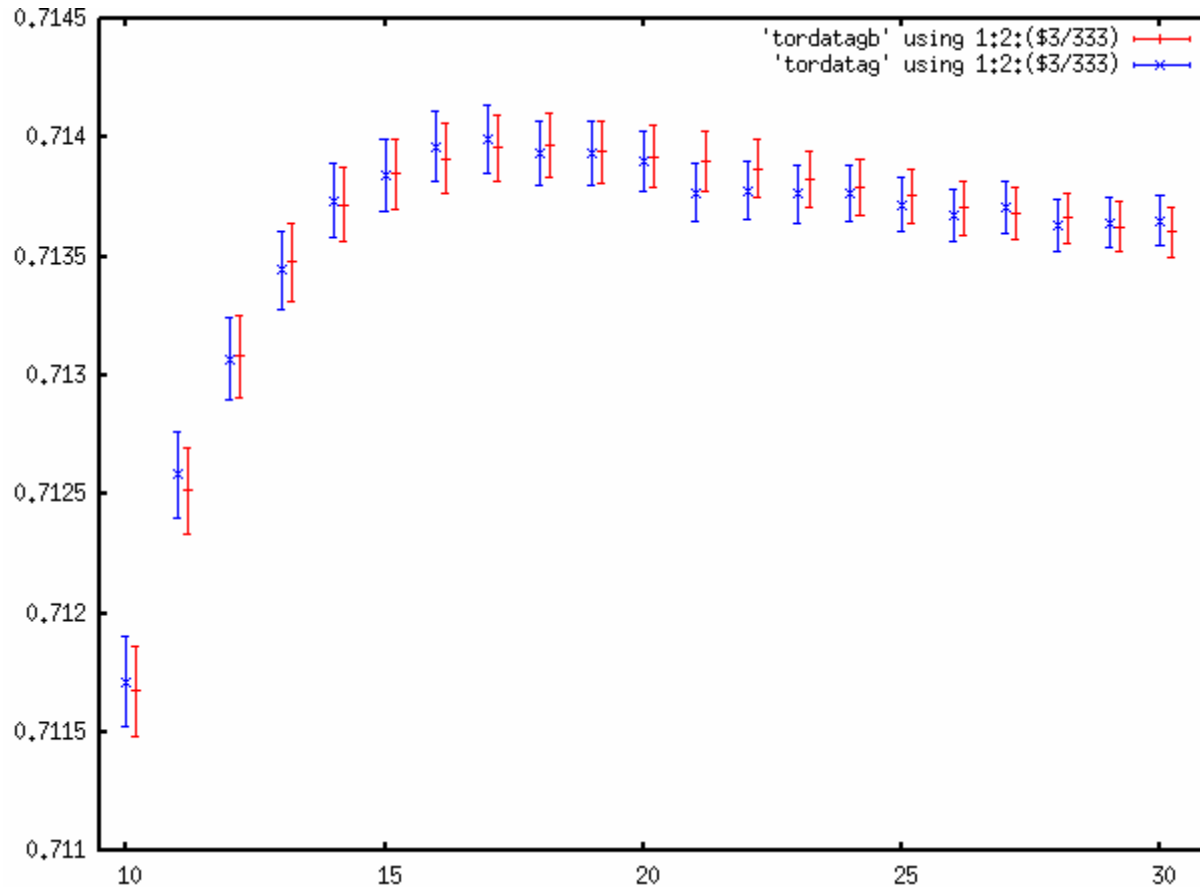


99.7% confidence intervals on OPT/\sqrt{n} ,
 $10 \leq n \leq 30$.

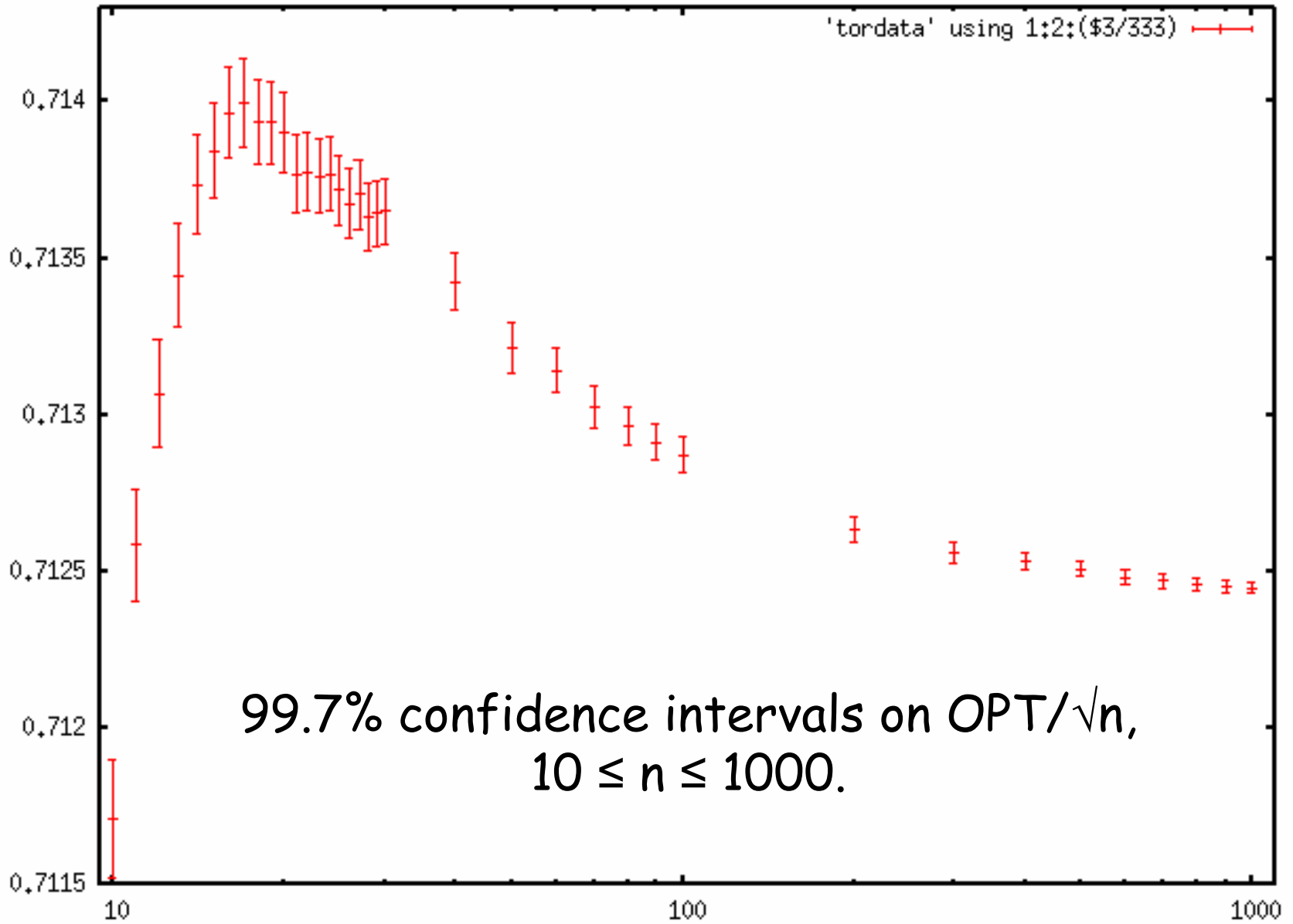
2nd Random Instance Generation Scheme

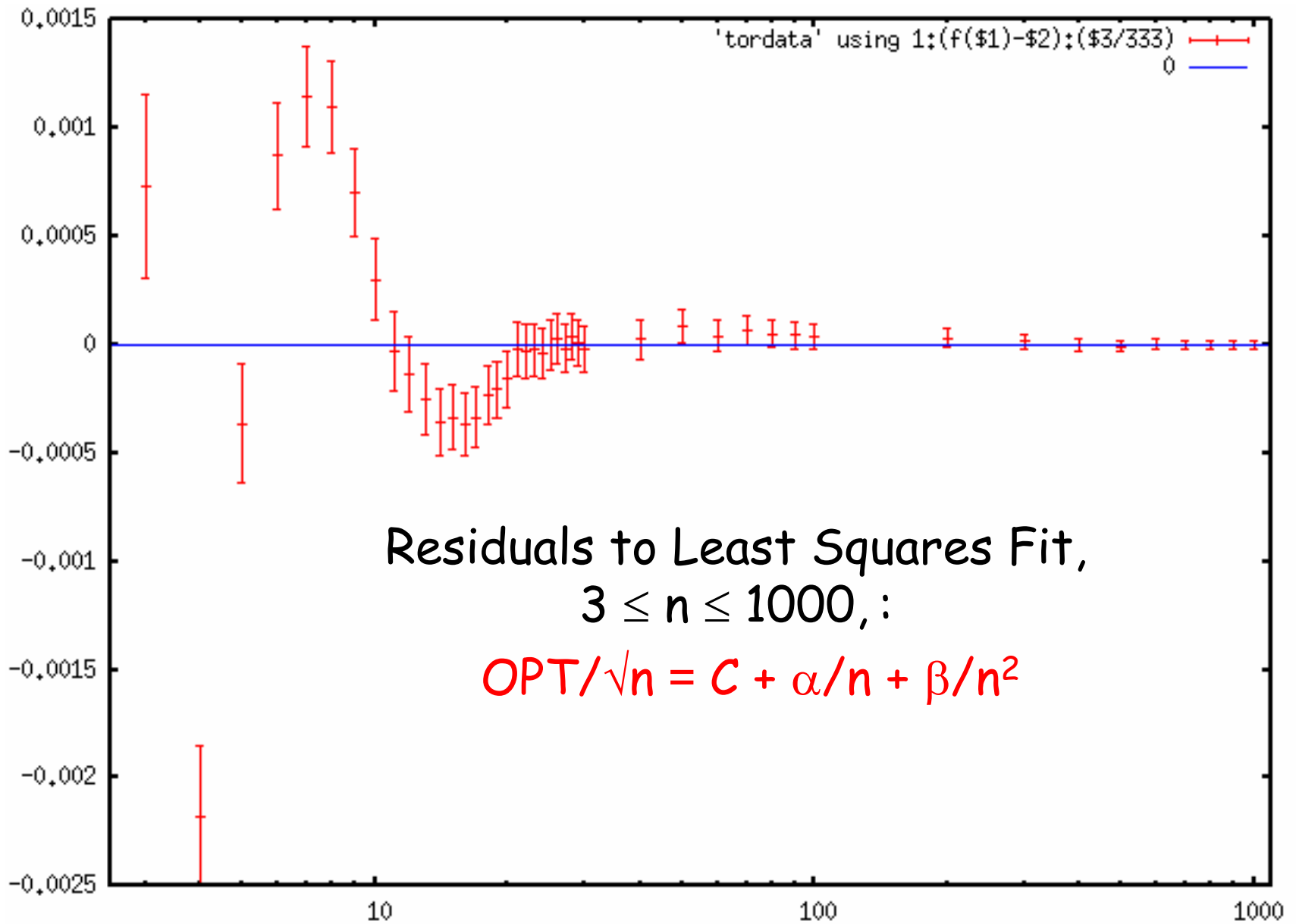


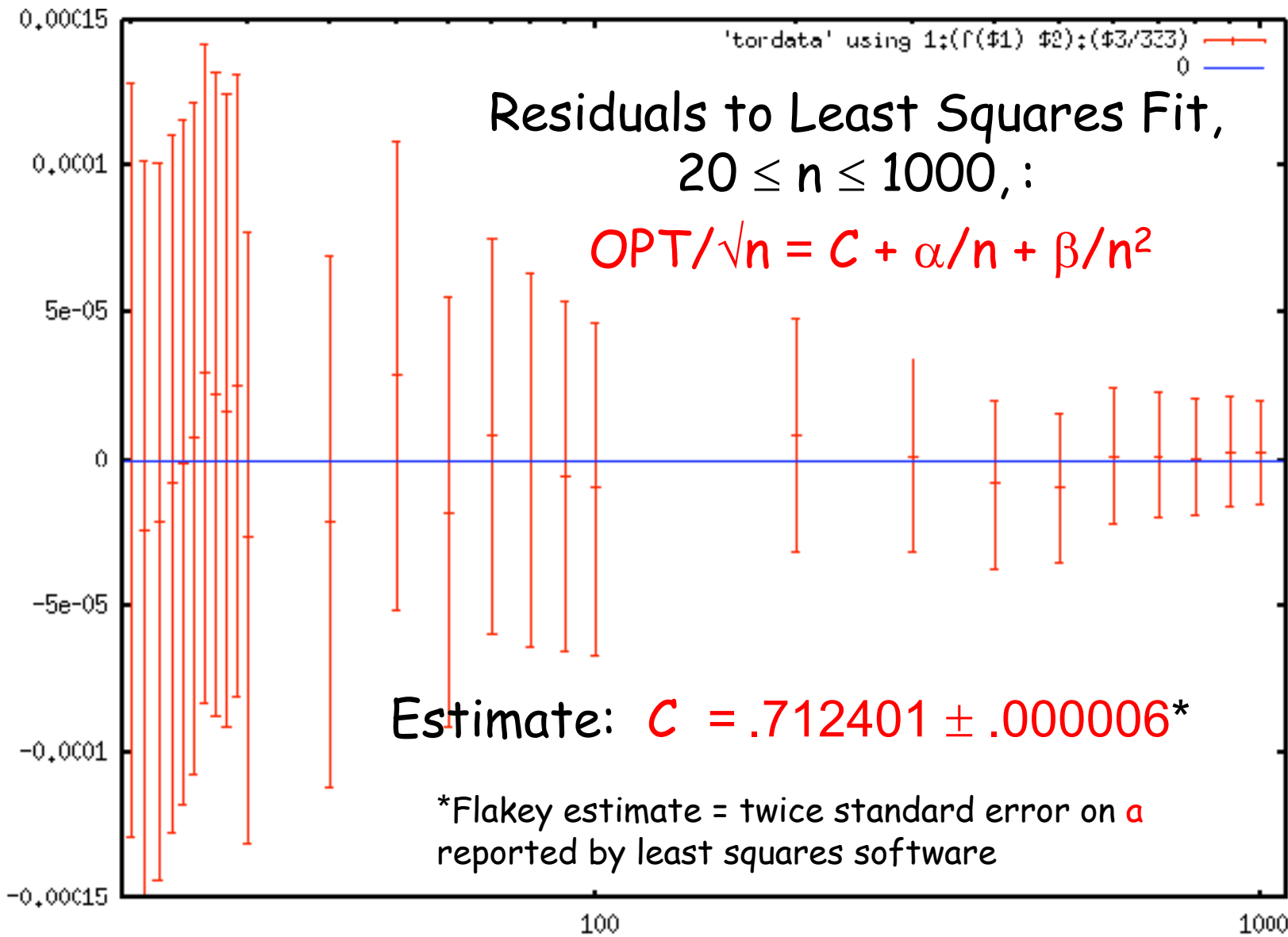
99.7% confidence intervals on OPT/\sqrt{n} ,
 $10 \leq n \leq 30$.

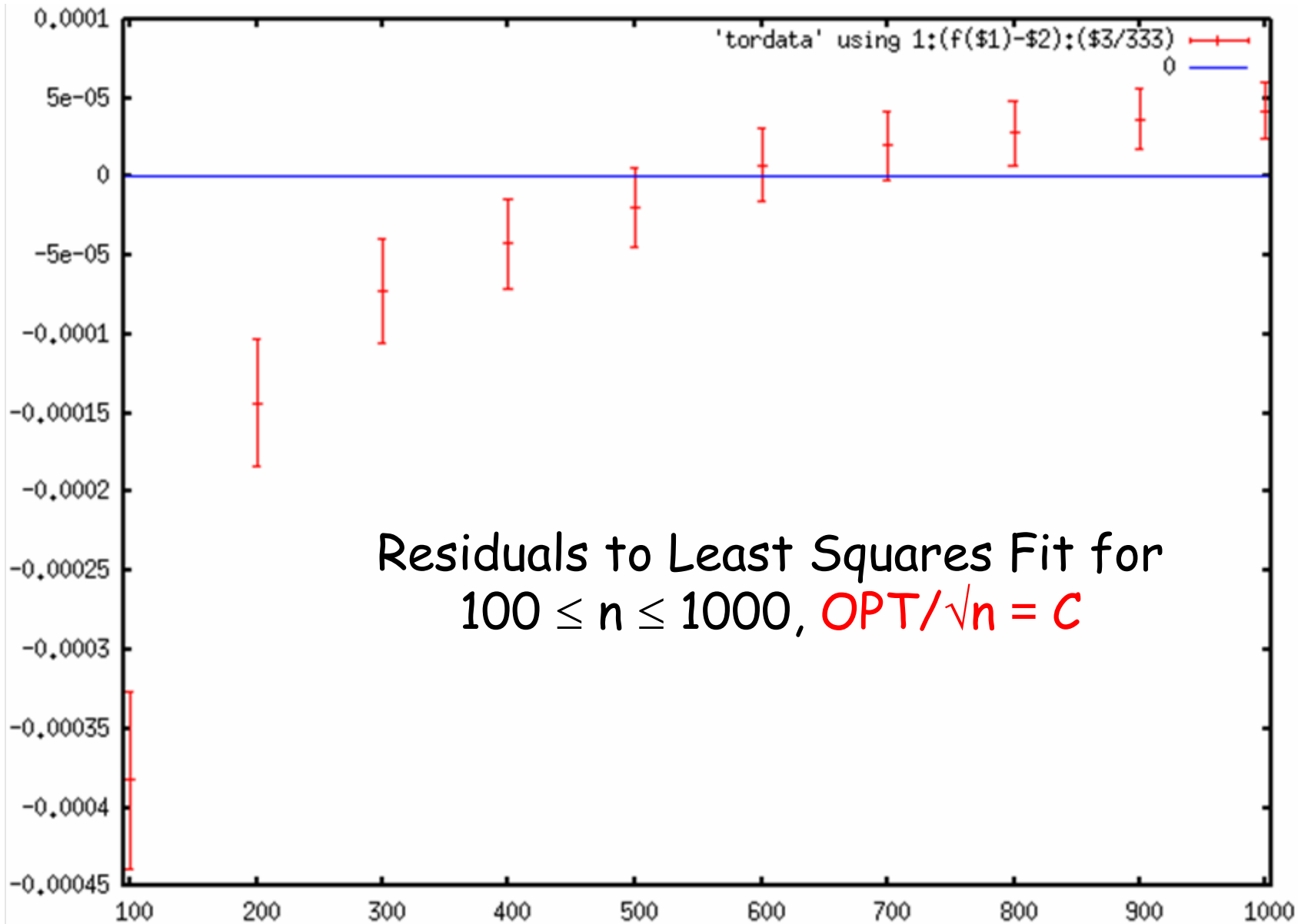


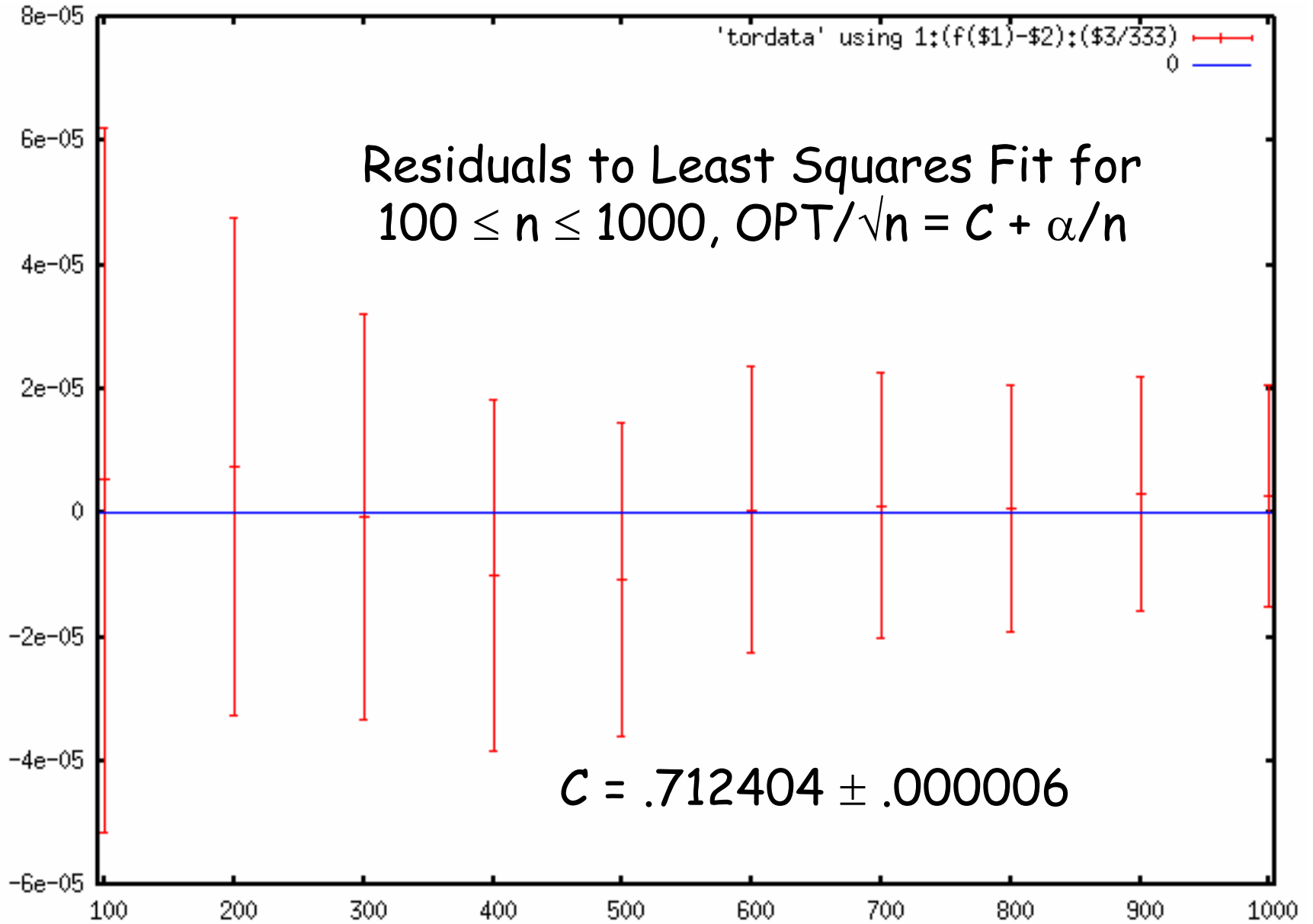
99.7% confidence intervals on OPT/\sqrt{n} ,
 $10 \leq n \leq 30$.







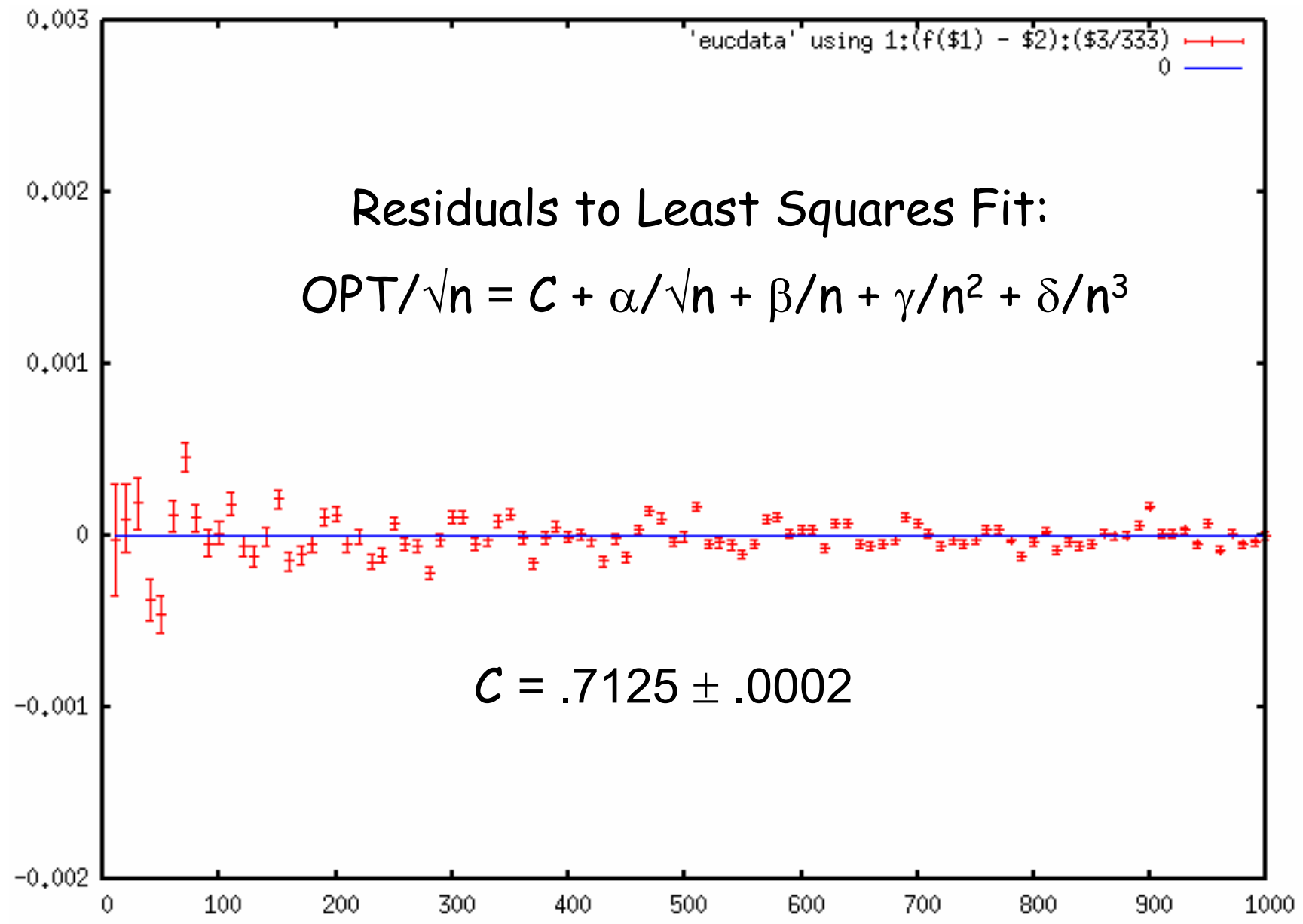






'eucdata' using 1:(f(\$1) - \$2):(\$3/333) 0

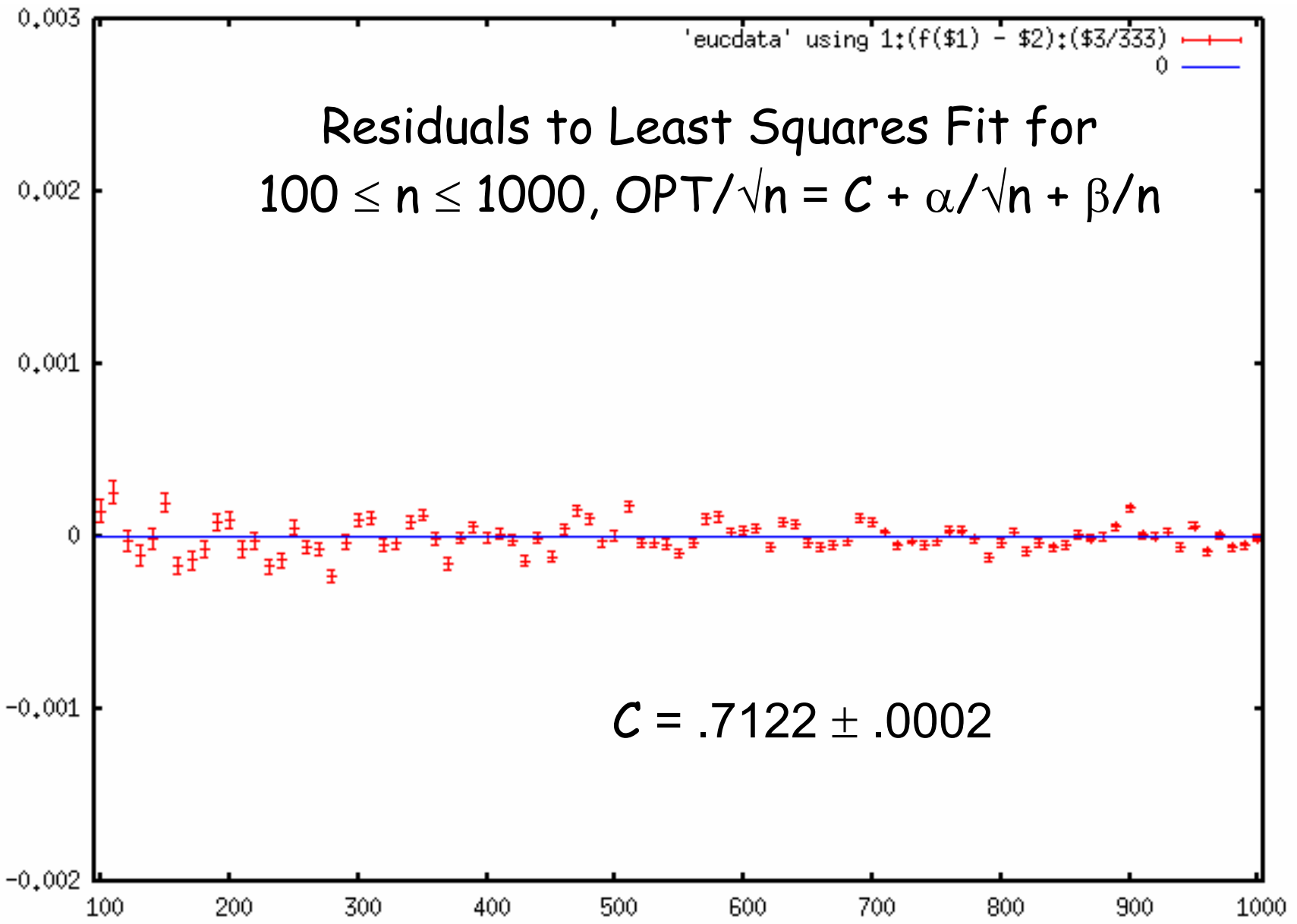
Residuals to Least Squares Fit:
 $OPT/\sqrt{n} = C + \alpha/\sqrt{n} + \beta/n + \gamma/n^2 + \delta/n^3$

$C = .7125 \pm .0002$



'eucldata' using 1:(f(\$1) - \$2):(\$3/333)  

Residuals to Least Squares Fit for
 $100 \leq n \leq 1000, OPT/\sqrt{n} = C + \alpha/\sqrt{n} + \beta/n$

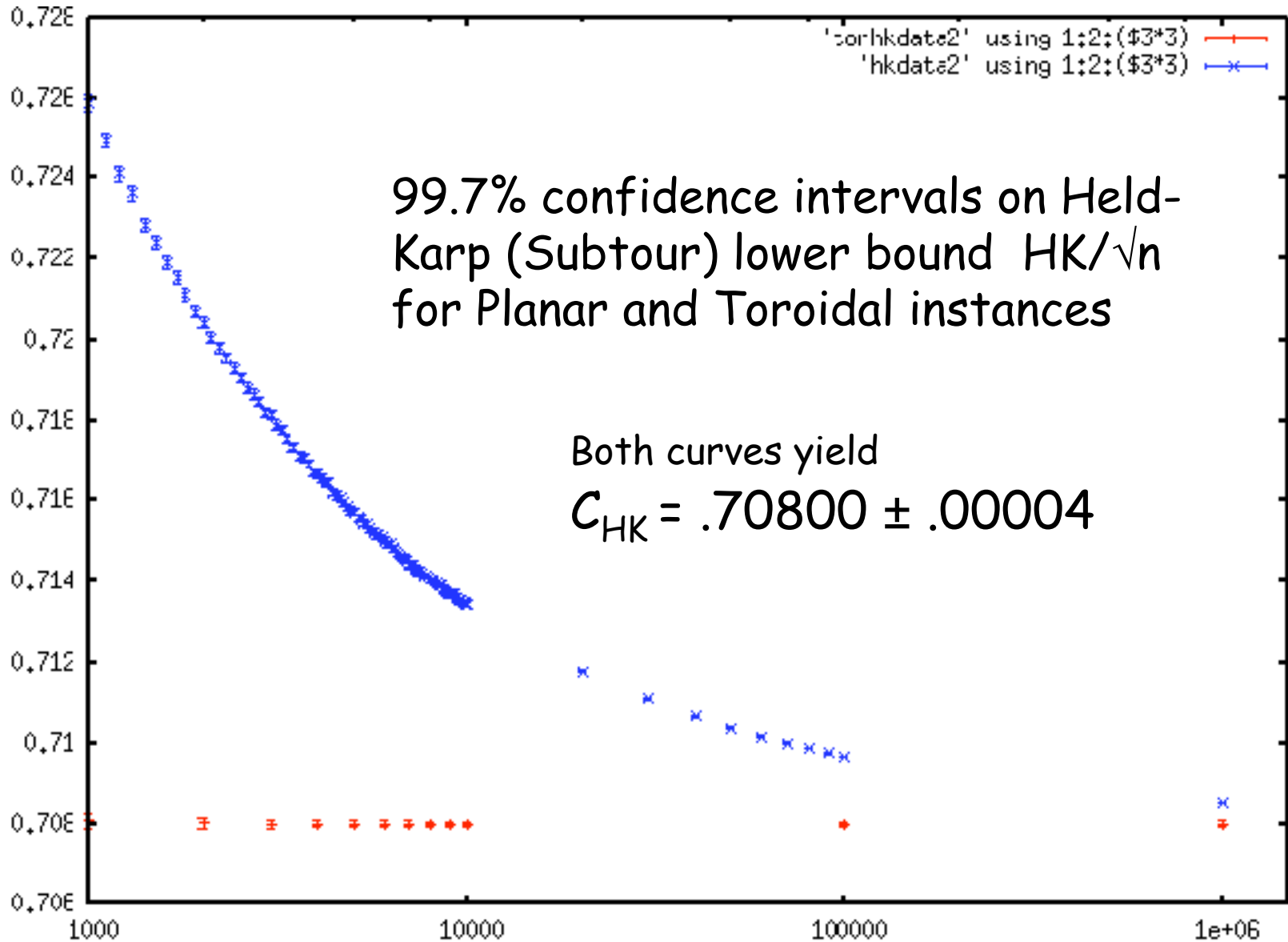


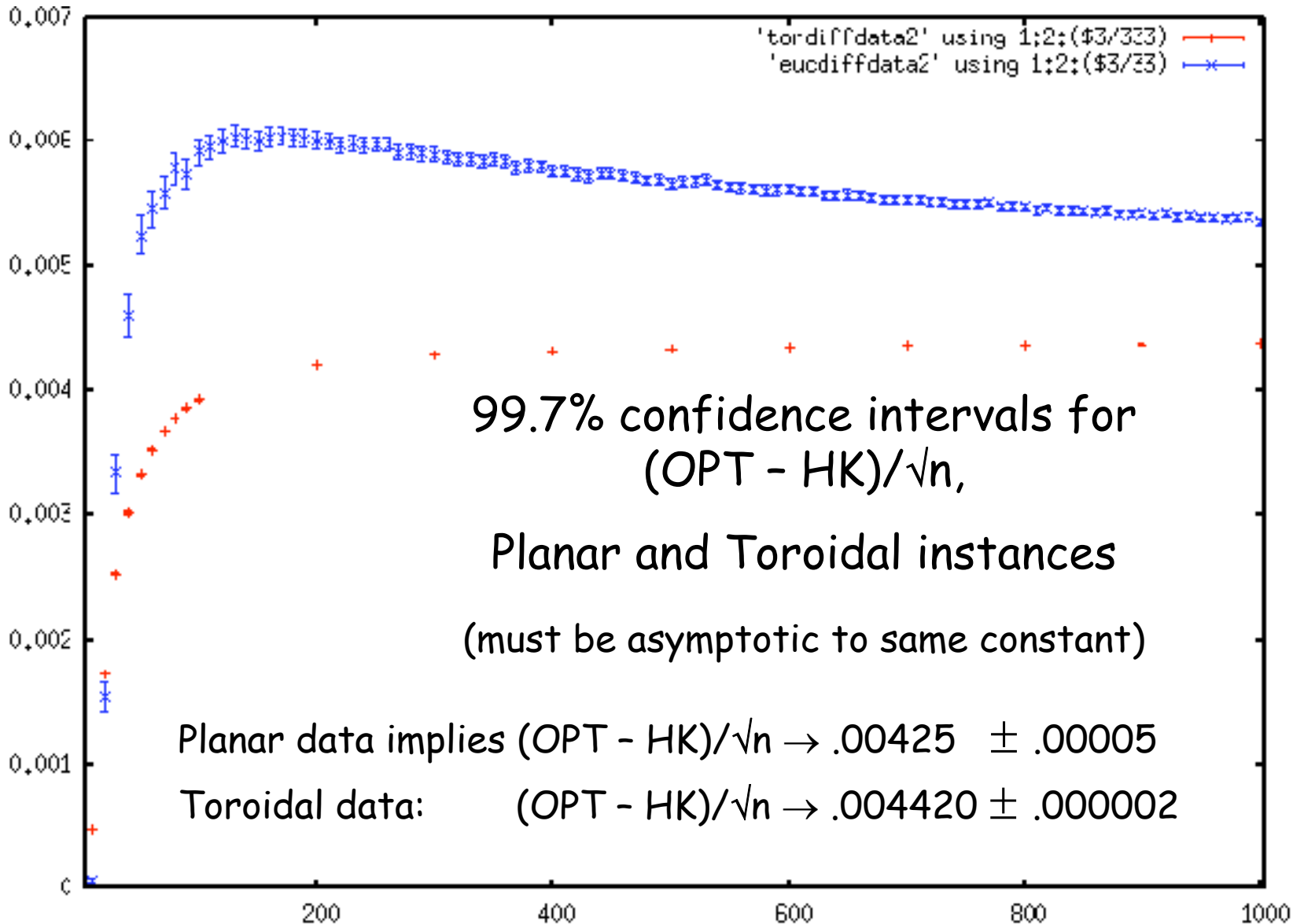
$C = .7122 \pm .0002$

Yet Another Approach

(Suggested by [Johnson, McGeoch, Rothberg, 1996])

- The Held-Karp (or subtour) lower bound on OPT is much easier to compute than OPT - feasible even for $n = 1,000,000$
- HK/\sqrt{n} is also asymptotic to a constant, C_{HK} , which, as with OPT, is the same for both Planar and Toroidal instances.
- C_{HK} can be determined quite accurately, given data for n as large as 1,000,000.
- OPT and HK are highly correlated - for a given n , the standard deviation on $(OPT - HK)$ is roughly 1/10 of that for OPT.
- Thus one can get very good estimates on $C(n) - C_{HK}(n)$, hence on $C - C_{HK}$, hence on $C = C_{HK} + (C - C_{HK})$.





Various Estimates on C

Percus-Martin [1997] - physics argument combined with experimental data on heuristic optima for 10- to 100-city toroidal instances	.7120	\pm .0002
Johnson-McGeoch-Rothberg [1996] HK + HK gap estimates, Toroidal	.7124	\pm .0002
Cook data, Toroidal, 20 - 1000, 3 terms	.712401	\pm .000006
Cook data, Toroidal, 100 - 1000, 2 terms	.712404	\pm .000006
Cook data, Planar, 10 - 1000, 4 terms	.7120	\pm .0002
Cook data, Planar, 10 - 1000, 5 terms	.7125	\pm .0002
Cook data, Planar, 100 - 1000, 3 terms	.7122	\pm .0002
Cook data, Planar, HK + HK gap, 100 - 1 million	.7123	\pm .0001
Cook data, toroidal, HK + HK gap, 100 - 1 million	.71242	\pm .00004

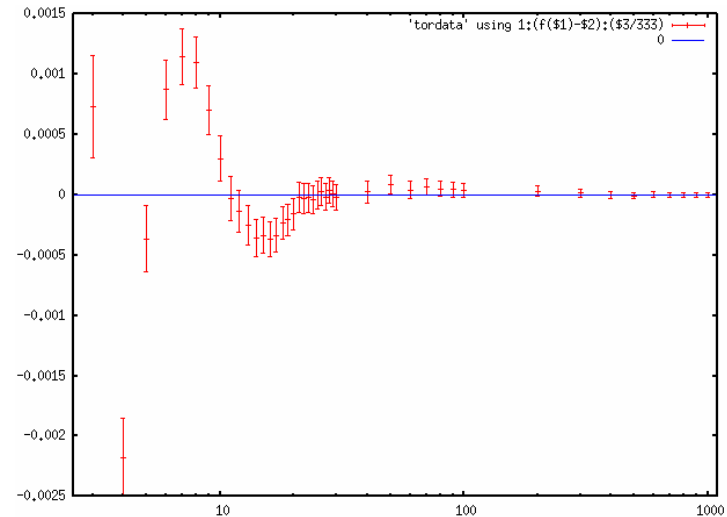
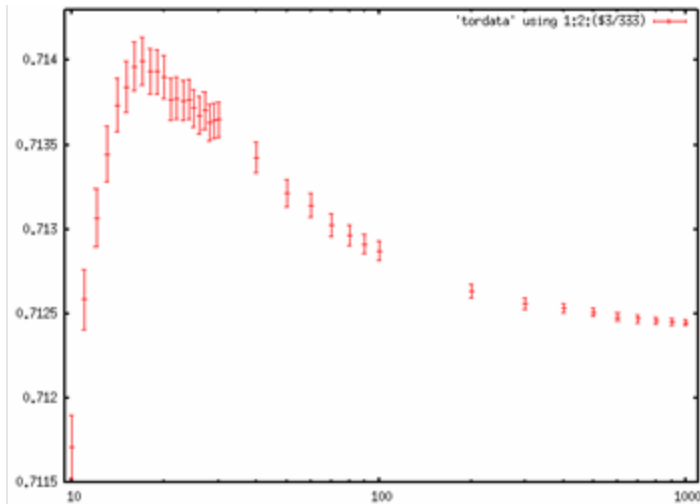
Various Estimates on C

Percus-Martin [1997] - physics argument combined with experimental data on heuristic optima for 10- to 100-city toroidal instances	.7120	$\pm .0002$
Johnson-McGeoch-Rothberg [1996] HK + HK gap estimates, Toroidal	.7124	$\pm .0002$
Cook data, Toroidal, 20 - 1000, 3 terms	.712401	$\pm .000006$
Cook data, Toroidal, 100 - 1000, 2 terms	.712404	$\pm .000006$
Cook data, Planar, 10 - 1000, 4 terms	.7120	$\pm .0002$
Cook data, Planar, 10 - 1000, 5 terms	.7125	$\pm .0002$
Cook data, Planar, 100 - 1000, 3 terms	.7122	$\pm .0002$
Cook data, Planar, HK + HK gap, 100 - 1 million	.7123	$\pm .0001$
Cook data, toroidal, HK + HK gap, 100 - 1 million	.71242	$\pm .00004$

OPEN PROBLEM

Find a function $C(n) = E[OPT/\sqrt{n}]$

matching ALL the data for $3 \leq n \leq 1000$,
and explain what is going on when $n < 20$.



Further Reading

- **A Theoretician's Guide to the Experimental Analysis of Algorithms**, D. S. Johnson. *Data Structures, Near Neighbor Searches, and Methodology: Proceedings of the Fifth and Sixth DIMACS Implementation Challenges*, M. Goldwasser, D. S. Johnson, and C. C. McGeoch, Editors, American Mathematical Society, Providence, 2002, 215-250.
- **The Traveling Salesman Problem: A Case Study in Local Optimization**, D. S. Johnson and L. A. McGeoch, *Local Search in Combinatorial Optimization*, E. H. L. Aarts and J. K. Lenstra (editors), John-Wiley and Sons, Ltd., 1997, pp. 215-310.
- **Asymptotic Experimental Analysis for the Held-Karp Traveling Salesman Bound**, D. S. Johnson, L. A. McGeoch, and E. E. Rothberg, *Proc. 7th Ann. ACM-SIAM Symp. on Discrete Algorithms*, 1996, pp. 341-350.
- **Experimental Analysis of Heuristics for the STSP**, D. S. Johnson and L. A. McGeoch, in *The Traveling Salesman Problem and its Variations*, G. Gutin and A. Punnen, Editors, Kluwer Academic Publishers, Dordrecht, 2002, 369-443.
- **Experimental Analysis of Heuristics for the ATSP**, D. S. Johnson, G. Gutin, L. A. McGeoch, A. Yeo, W. Zhang, and A. Zverovich, in *The Traveling Salesman Problem and its Variations*, G. Gutin and A. Punnen, Editors, Kluwer Academic Publishers, Dordrecht, 2002, 445-487.
- All available from <http://www.research.att.com/~dsj/papers>

Stop!