8th DIMACS Implementation Challenge:

The Traveling Salesman Problem

http://www.research.att.com/∼dsj/chtsp/

David S. Johnson
AT&T Labs – Research
Florham Park, NJ 07932-0971

dsj@research.att.com

http://www.research.att.com/∼dsj/

Co-Organized with

Lyle McGeoch, Fred Glover, Cesar Rego

DIMACS Implementation Challenges

1. Network Flows and Matching, 1990-91

2. Clique, Coloring, and Satisfiability, 1992-93

3. Parallel Computing on Trees and Graphs, 1993-94

4. Fragment Assembly and Genome Rearrangment, 1994-95

5. Priority Queues and Dictionaries, 1995-96

6. Near Neighbor Searches in High Dimension, 1997-98

7. Semidefinite Programming, 1999-2000

8. The Traveling Salesman Problem, 2000...

OUTLINE OF TALK

- **Why a Challenge**
- **Who should Participate**
- **How to Participate**
- **Preliminary Results**
  - **Machine Speeds and Normalizations**
  - **Algorithmic Comparisons**
- **Future Directions**

SCIENTIFIC GOALS

- Determine the current state of the art with respect to tradeoffs between running time and quality of solution for the TSP.

- Identify promising algorithmic ideas for the TSP worthy of further investigation.

- Gain insight into combinatorial optimization in general by seeing how various generic ideas are best adapted to the TSP context.

- Explore how best to conduct a distributed algorithmic comparison project of this sort, and how best to analyze and display the resulting data.

- Produce a DIMACS technical report summarizing what we learn, with all participants as co-authors.

OTHER AGENDAS

- Obtain source material for a summary chapter on experimental analysis of TSP algorithms to be written with Lyle McGeoch for an upcoming book on the TSP edited by Gutin and Punnen.

- Establish a long-lived mechanism for future researchers to evaluate their algorithms in comparison to works of the past.

- Stop the flow of uninformed papers on the TSP.

DESIRED PARTICIPANTS

- Current TSP researchers.

- Researchers who have published experimental results about TSP algorithms in the past, so that those results can be put in perspective.

- New TSP researchers interested in investigating new ideas and unanswered questions.

- Future TSP researchers who want to compare with previous results.

ARENAS FOR COMPETITION

(Currently Restricted to Symmetric TSP)

1. Heuristics

- Tour Construction Heuristics
- Simple Local Optimization
  (2-Opt, 3-Opt, and Variants)
- Lin-Kernighan Variants
- Chained Lin-Kernighan Variants
- Other Metaheuristics
  (Simulated Annealing, Tabu Search, Neural Nets, Genetic Algorithms, etc.)

2. Fast Lower Bound Algorithms

3. Optimization Algorithms

4. Open to Suggestions...

HOW TO PARTICIPATE

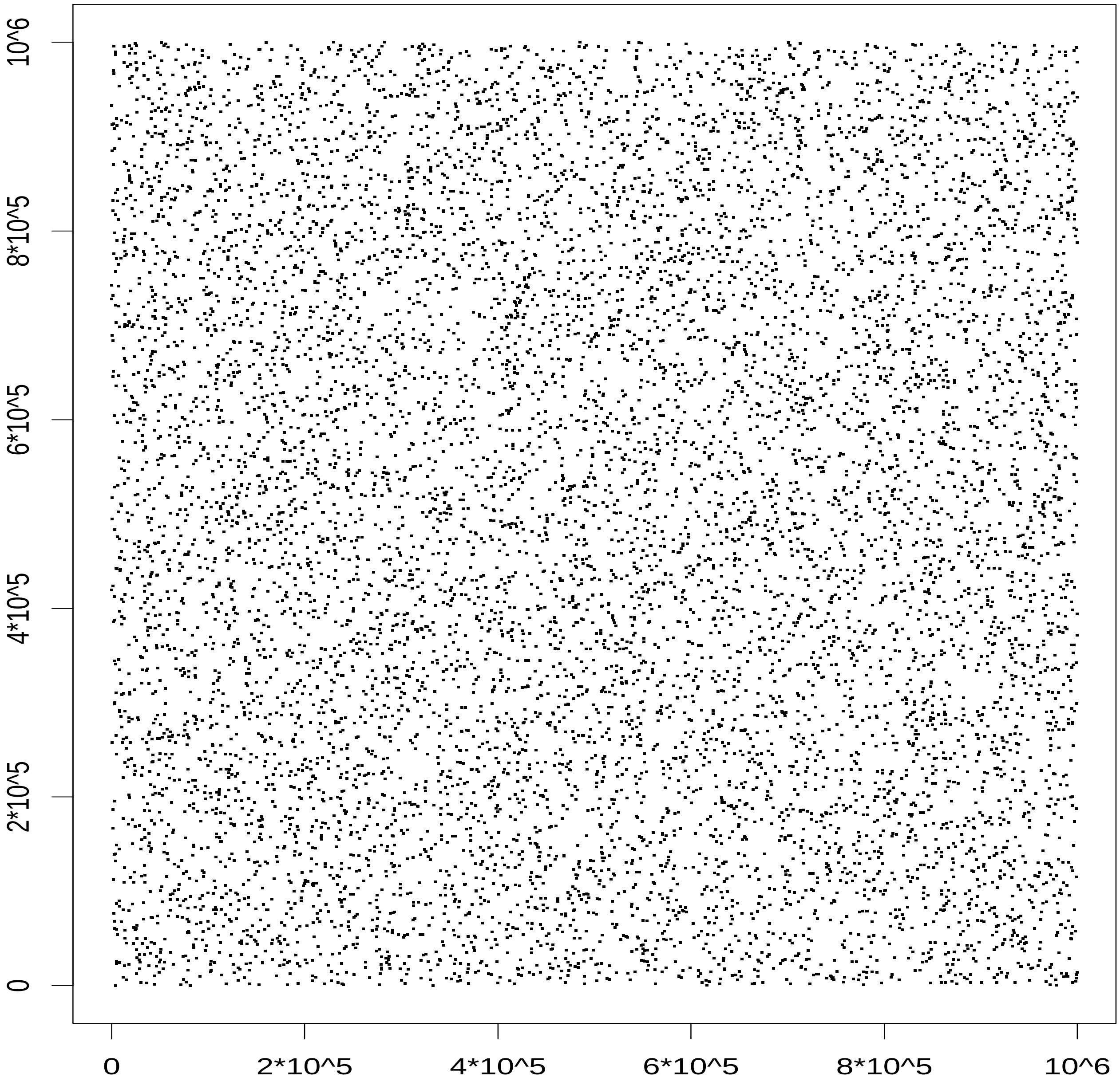
1. Download Instances, Instance Generators, and Benchmark Codes from the website.
2. Compile Generators and Benchmark Codes (`C` code) using your standard compilers on your standard machine.
3. Run the Generators to generate the random instances in the testbed, comparing to downloaded samples to verify that Generators are performing correctly.
4. Run the Benchmark Greedy code on selected random instances (as specified on the website) to (roughly) benchmark your machine's speed as a function of instance size. Do this for all the specified instance that will fit in your machine's memory.
5. Run your own codes on the all the Benchmark Instances that they can handle. Allowed excuses for failure to run: Instance too big, Running time too long, Code can't handle instances of this type (distance matrices, fractional coordinates, etc.)
6. Send results to `dsj@research.att.com` using formats specified at the website. (Tentative initial deadline: 30 September 2000.)
7. **Extra Credit:** Perform extra experiments as suggested by DSJ or other participants. Suggest extra experiments to be performed by DSJ or other participants.

TESTBED, Part I - 55 Random Instances

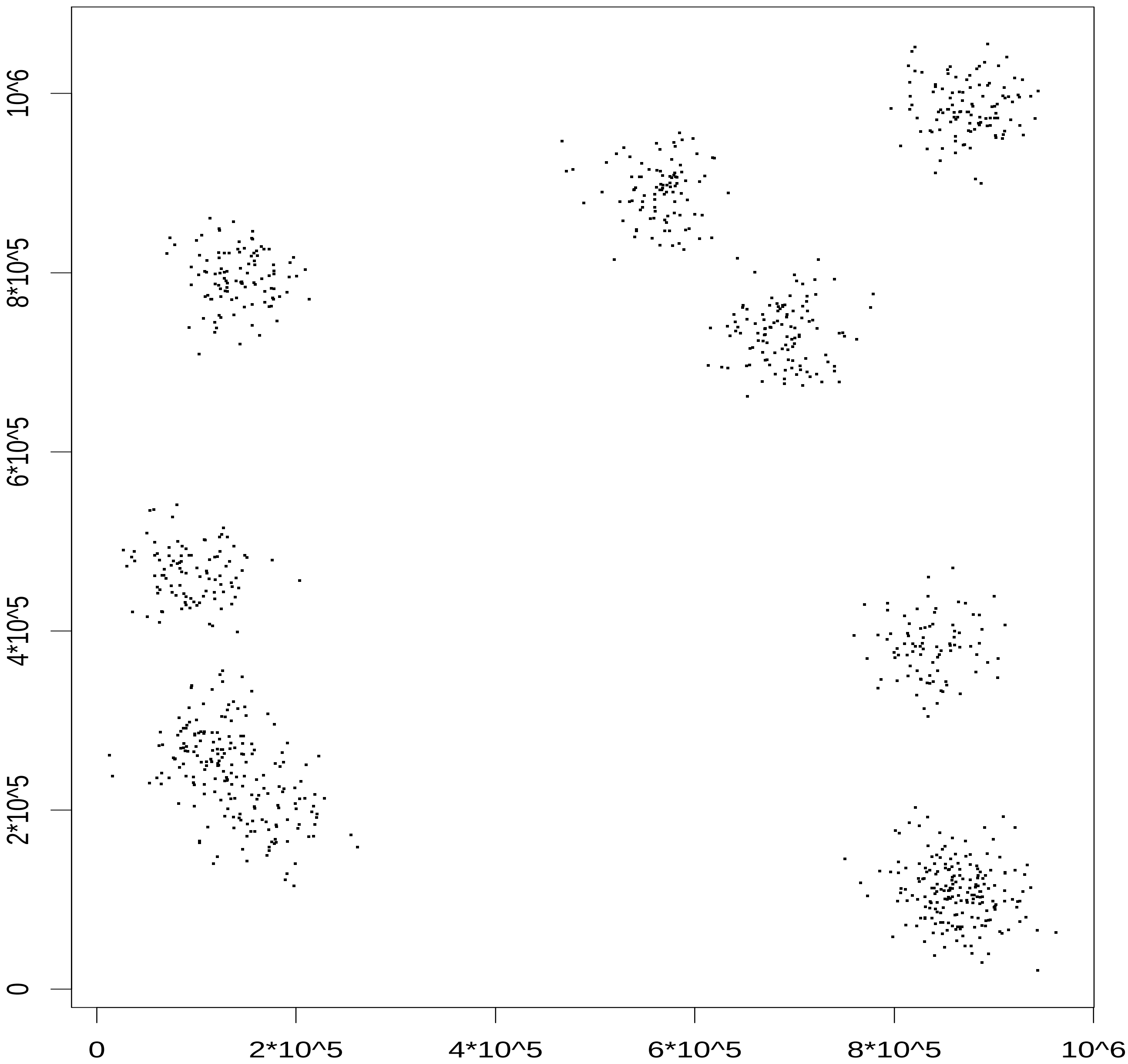
1. **Uniform Random Euclidean Instances**
   (Points uniform in the $10^6 \times 10^6$ square)
   Sizes increasing by factors of $\sqrt{10}$ from 1,000 to 10,000,000
   - Ten 1,000-city instances
   - Five 3,162-city instances
   - Three 10,000-city instances
   - Two 31,623-city instances
   - Two 100,000-city instances
   - One each of $10^{5.5}$-, 10^6-, $10^{6.5}$-, and 10^7-city instances.
2. **Uniform Random Euclidean Instances**
   (Points clustered in the $10^6 \times 10^6$ square)
   - Ten 1,000-city instances
   - Five 3,162-city instances
   - Three 10,000-city instances
   - Two 31,623-city instances
   - Two 100,000-city instances
3. **Random Distance Matrices**
   (Distances chosen uniformly from $[0, 10^6]$)
   - Four 1,000-city instances
   - Two 3,162-city instances
   - One 10,000-city instance

TESTBED, Part II - 34 TSPLIB Instances

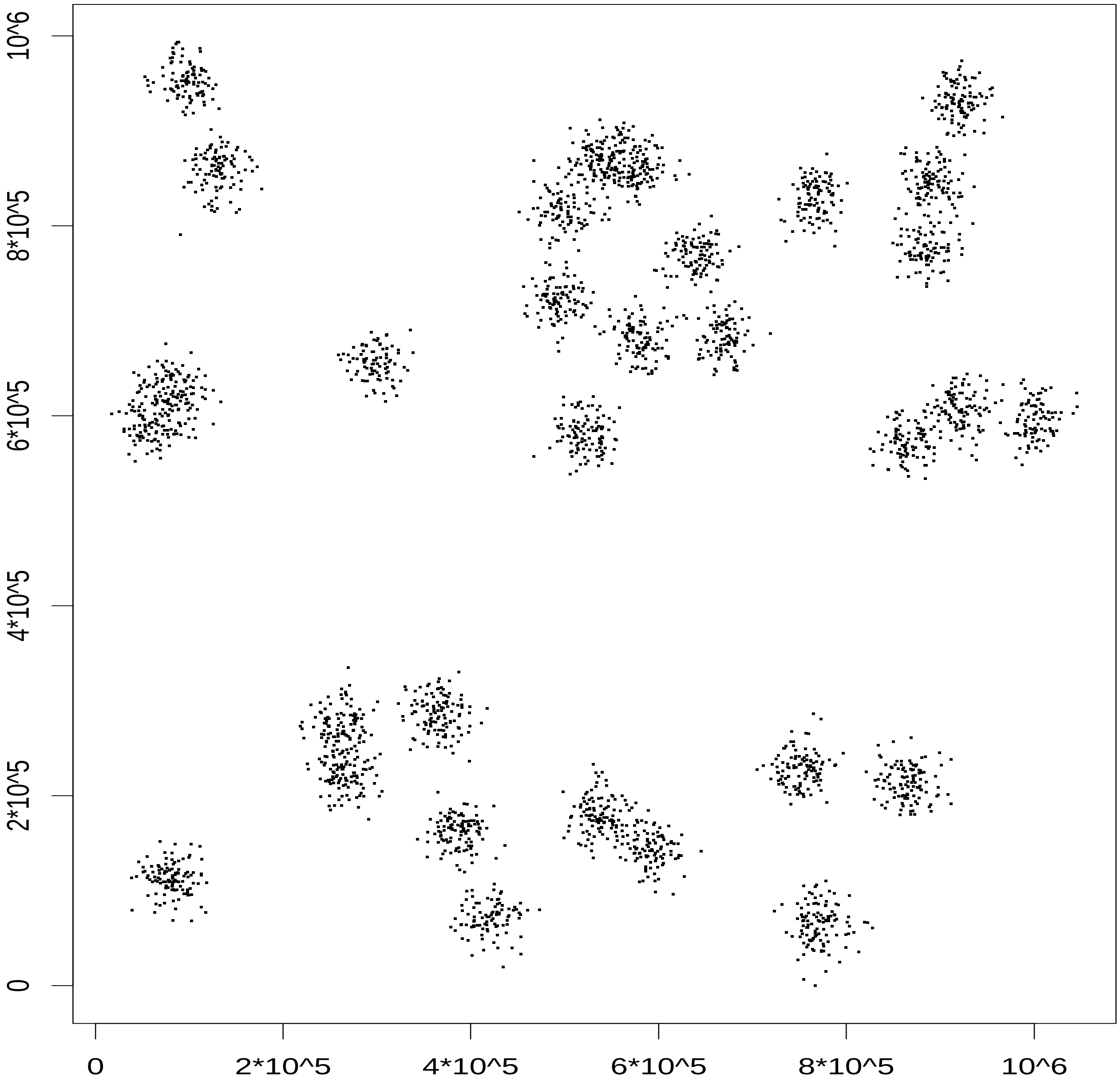
dsj1000	d2103
pr1002	u2152
si1032	u2319
u1060	pr2392
vm1084	pcb3038
pcb1173	fl3795
d1291	fnl4461
rl1304	rl5915
rl1323	rl5934
nrw1379	pla7397
fl1400	rl11849
u1432	usa13509
fl1577	brd14051
d1655	d15112
vm1748	d18512
u1817	pla33810
rl1889	pla85900



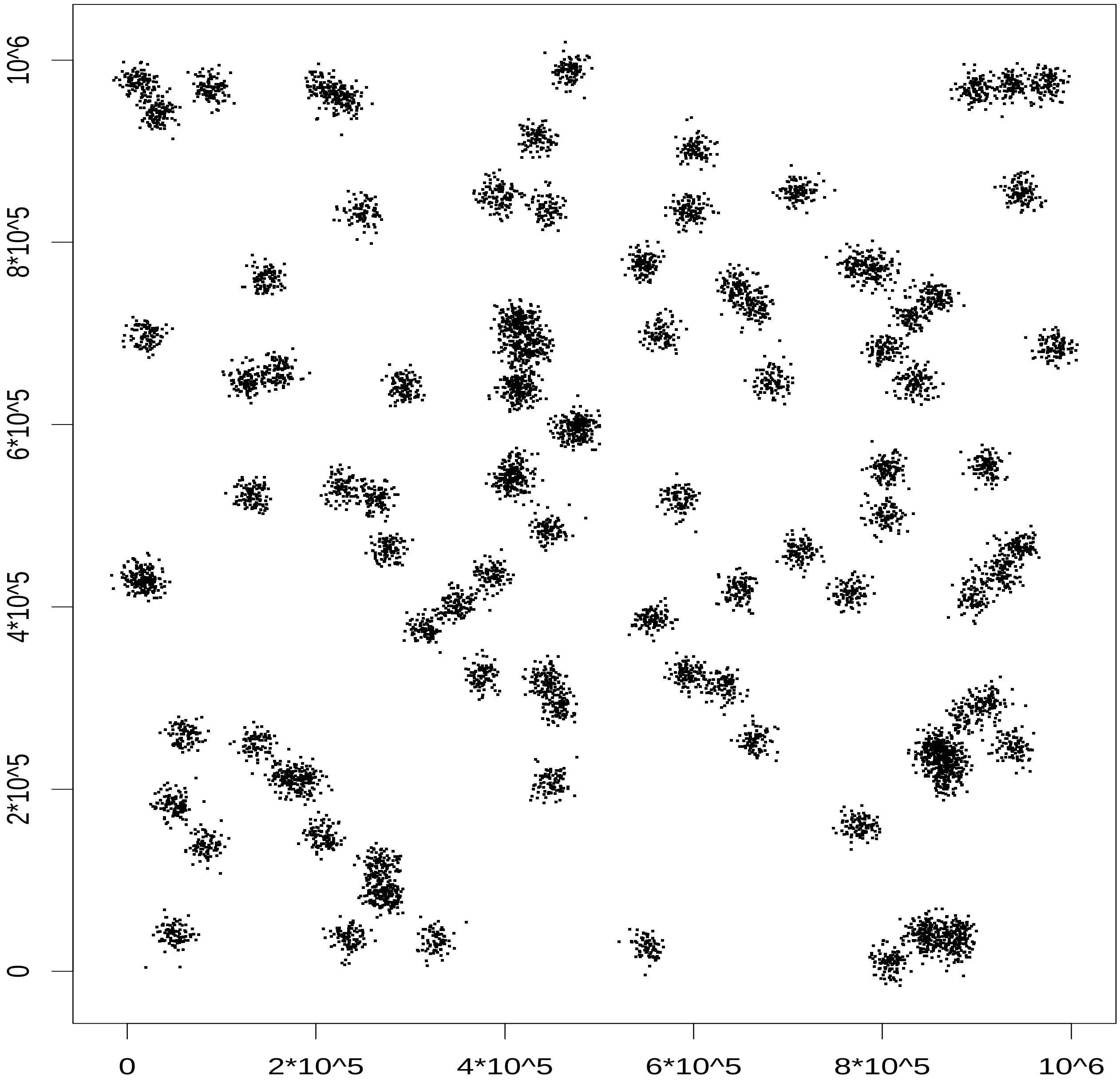
10,000-City Uniform Random Euclidean Instance



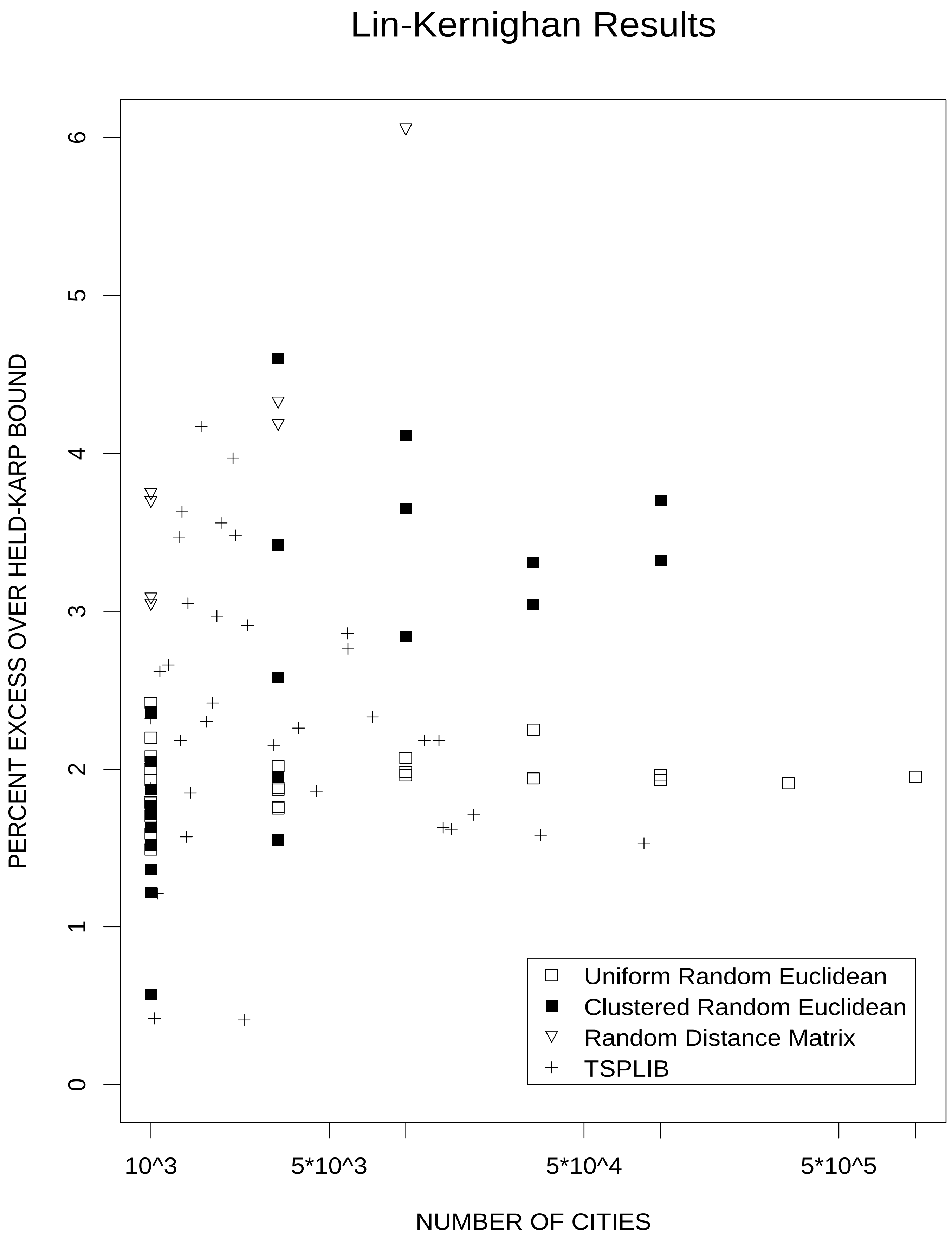
1,000-City Clustered Random Euclidean Instance

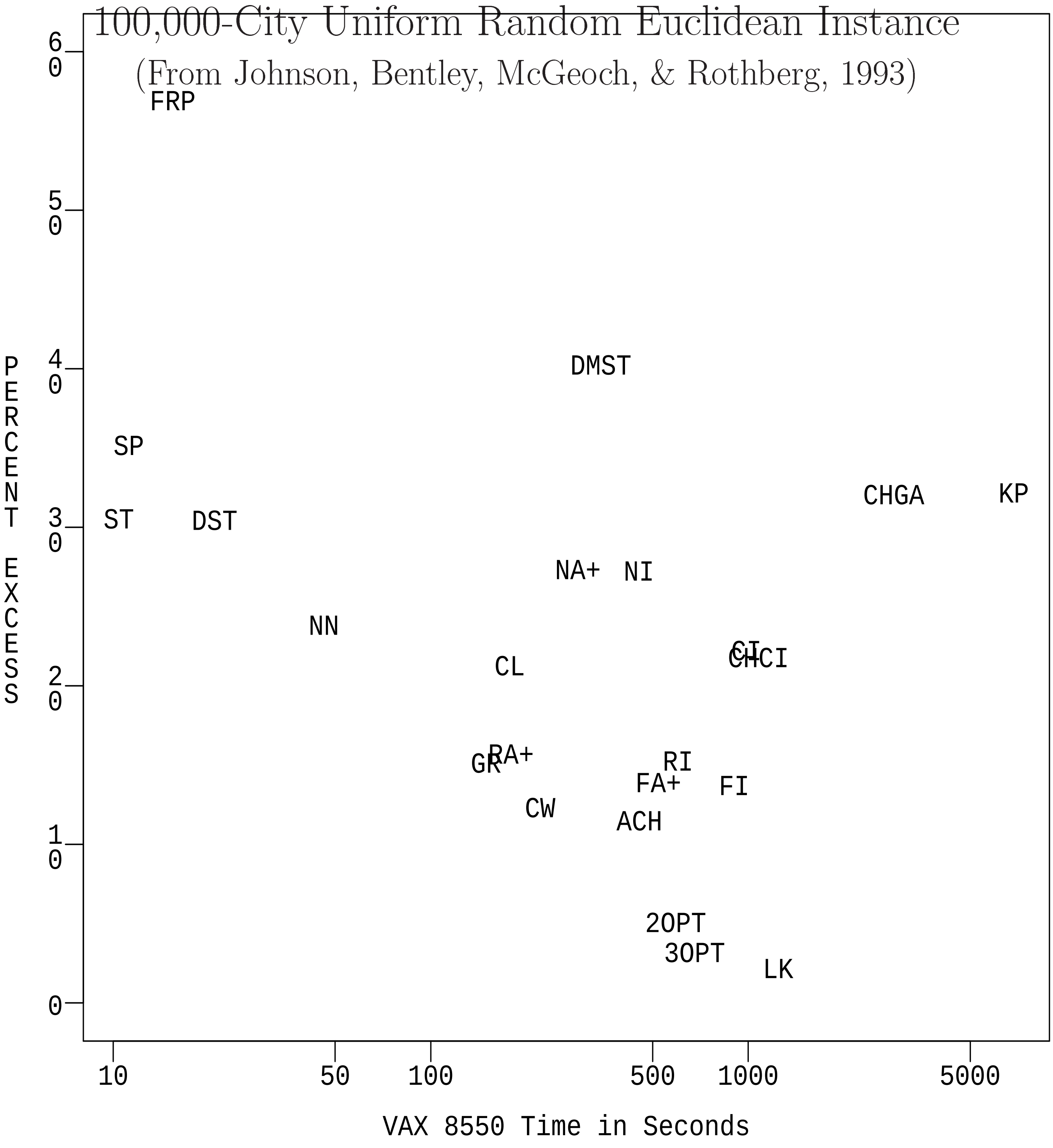


3,162-City Clustered Random Euclidean Instance



10,000-City Clustered Random Euclidean Instance


Lin-Kernighan Results
PERCENT EXCESS OVER HELD-KARP BOUND
0
1
2
3
4
5
6
10^3
5*10^3
5*10^4
5*10^5
NUMBER OF CITIES
Uniform Random Euclidean
Clustered Random Euclidean
Random Distance Matrix
TSPLIB



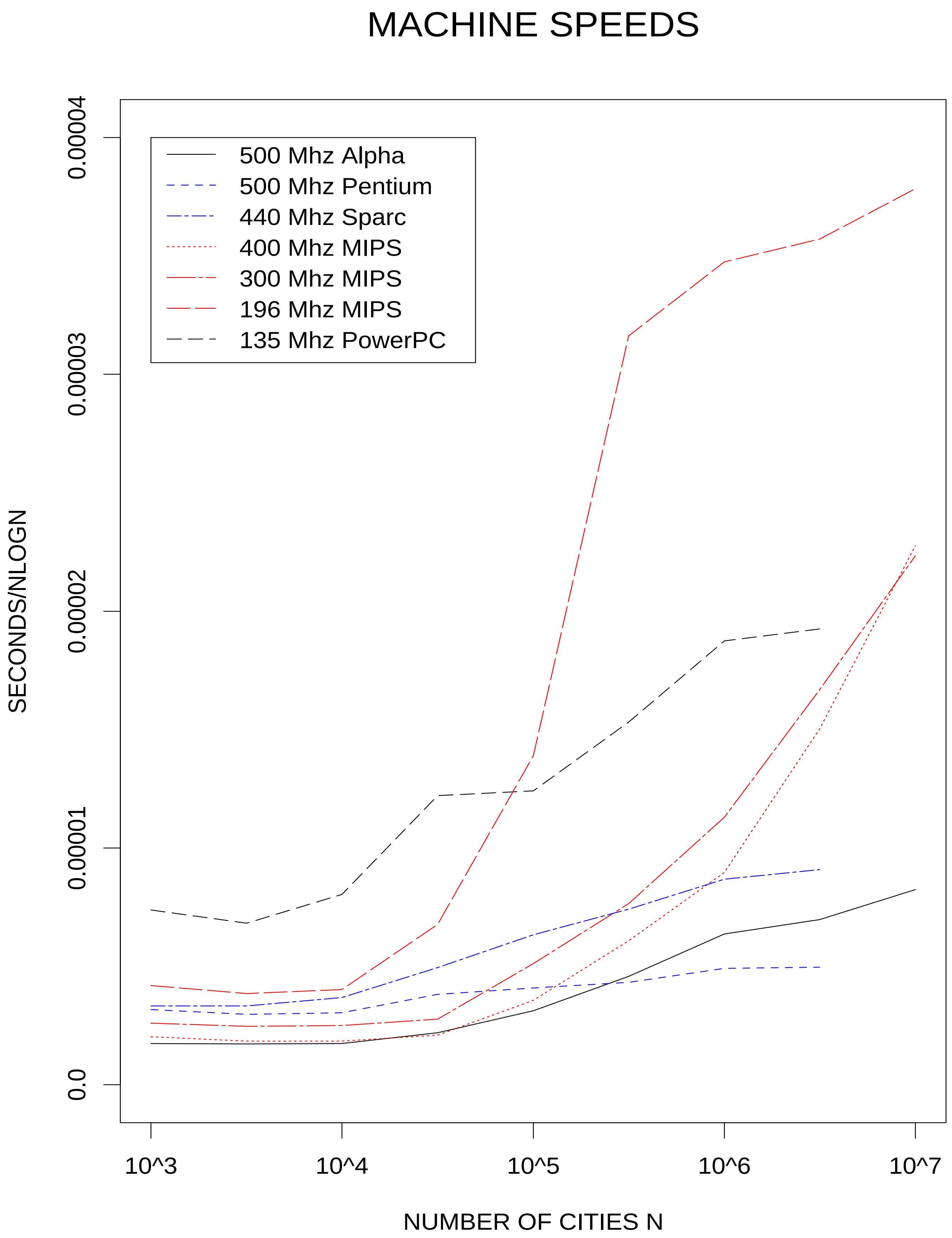
100,000-City Uniform Random Euclidean Instance
(From Johnson, Bentley, McGeoch, & Rothberg, 1993)
PERCENT EXCESS
60
50
40
30
20
10
0
FRP
DMST
SP
CHGA
KP
ST
DST
NA+
NI
NN
CI
CHCI
CL
RA+
GR
RI
FA+
FI
CW
ACH
2OPT
3OPT
LK
10
50
100
500
1000
5000
VAX 8550 Time in Seconds

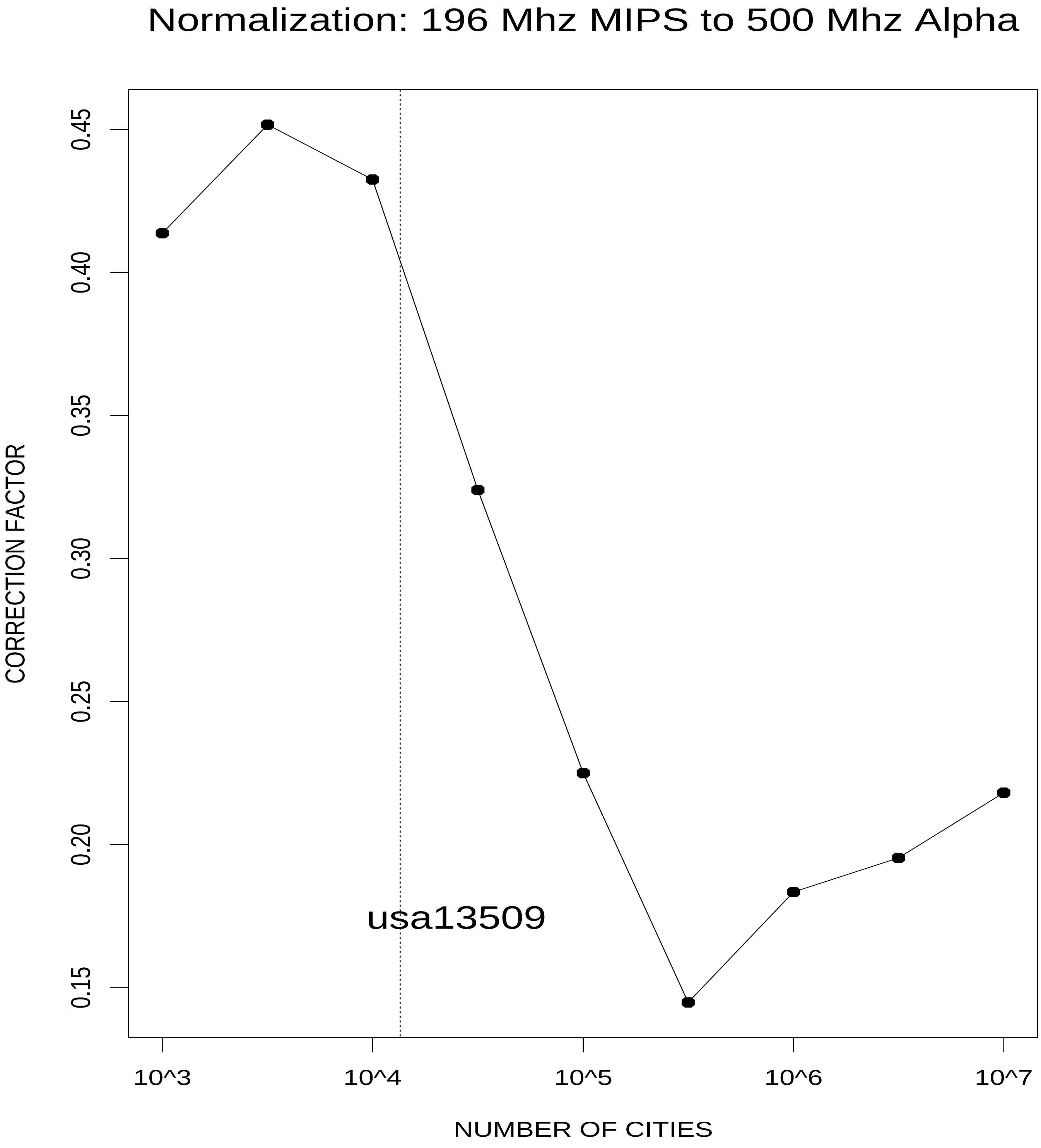
The Test Battery

```
time greedy E1k.0 1000
time greedy E3k.0 316
time greedy E10k.0 100
time greedy E31k.0 32
time greedy E100k.0 10
time greedy E316k.0 3
time greedy E1M.0 1
time greedy E3M.0 1
time greedy E10M.0 1
```

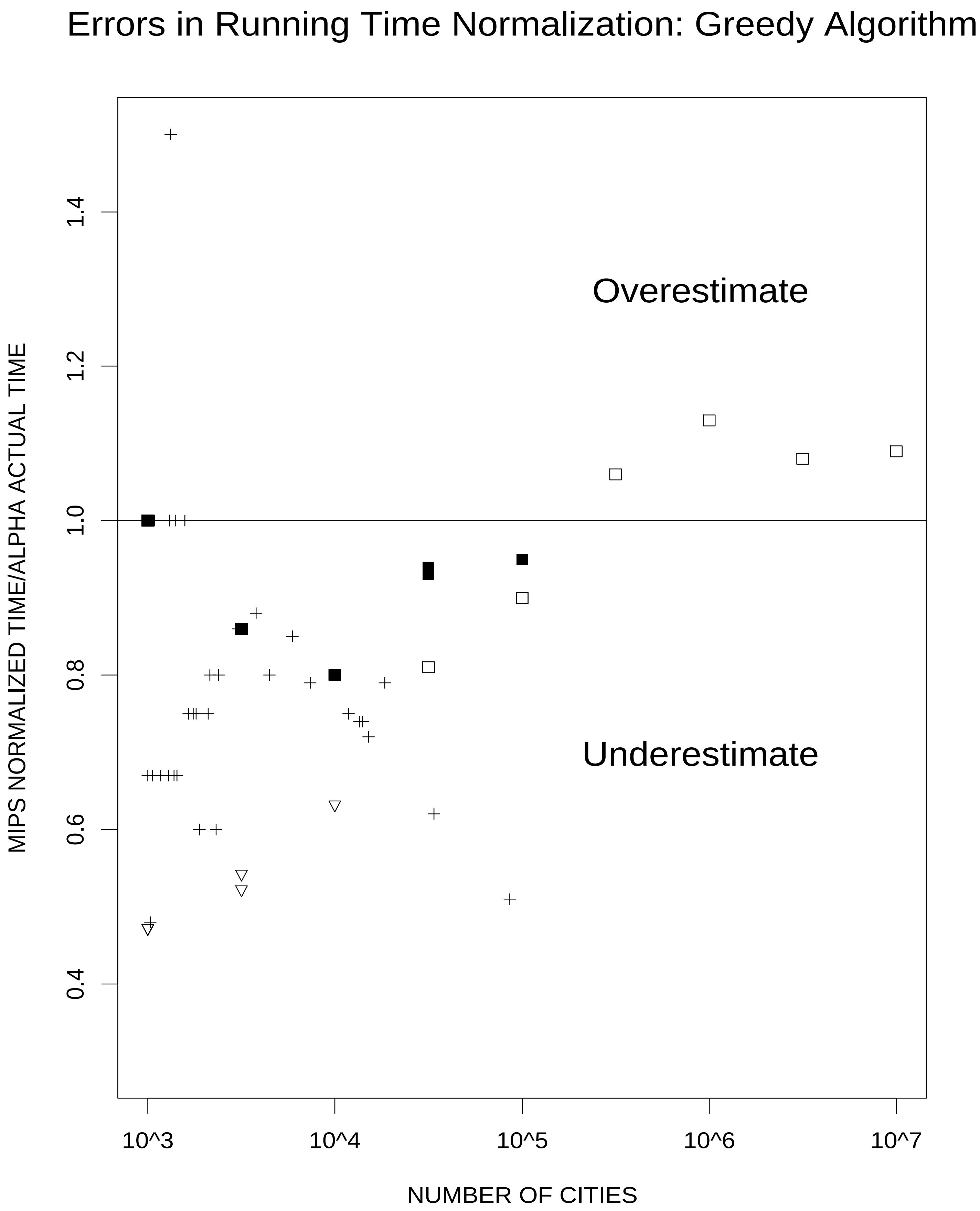
User Seconds

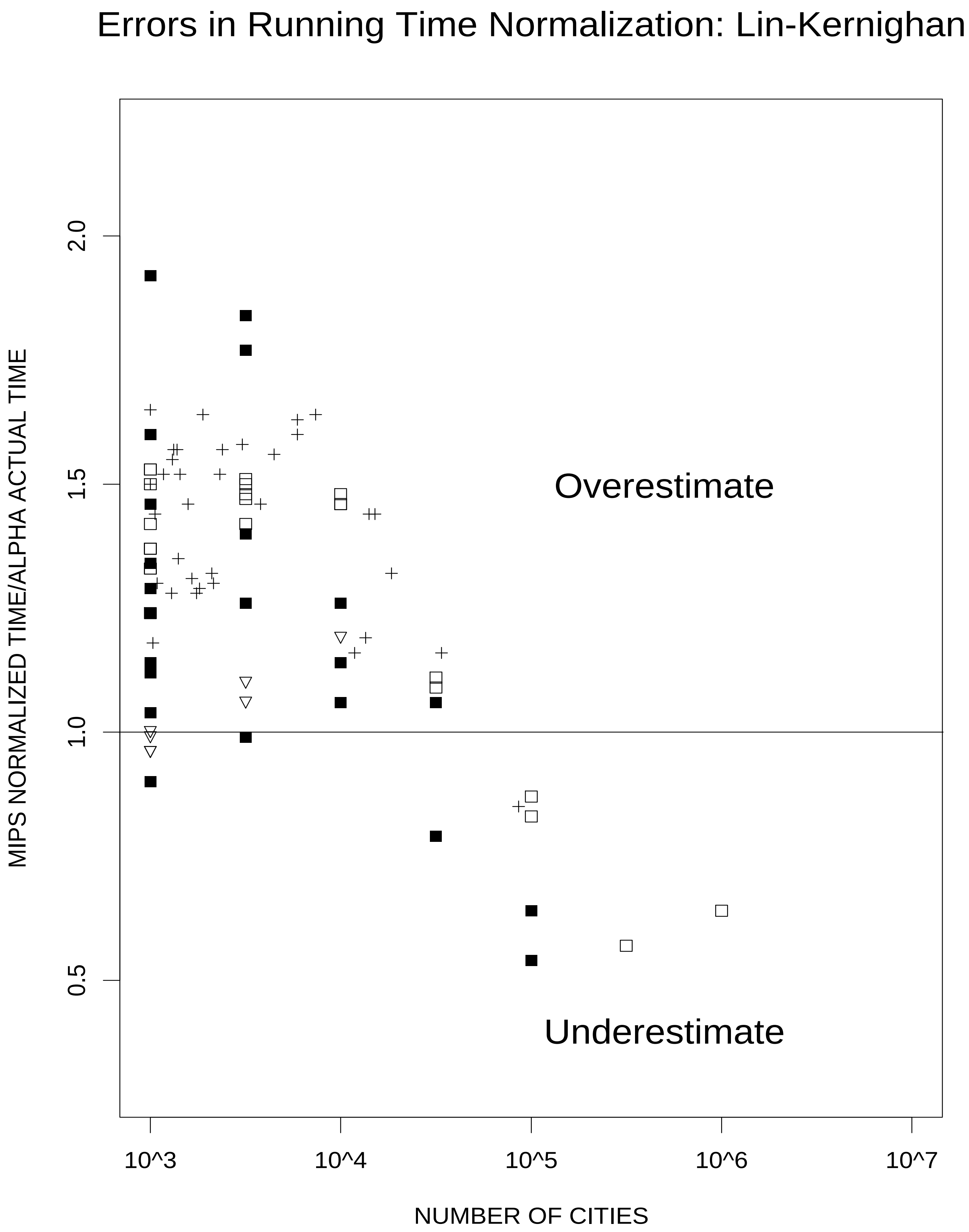
Instance	500 Mhz Alpha	400 Mhz MIPS R12000	300 Mhz MIPS R12000	500 Mhz Pentium III	440 Mhz Sparc Ultra 10	196 Mhz MIPS R10000	135 Mhz IBM Power2
1000 x 1000	12	14	18	22	23	29	51
316 x 3162	14	15	20	24	27	31	55
100 x 10,000	16	17	23	28	34	37	74
32 x 31,623	23	22	29	40	52	71	128
10 x 100,000	36	41	59	47	73	160	143
3 x 316,228	55	73	92	52	89	380	184
1 x 1,000,000	88	124	156	68	120	480	259
1 x 3,162,278	330	711	790	235	430	1690	911
1 x 10,000,000	1330	3670	3600	--	--	6100	--


MACHINE SPEEDS
500 Mhz Alpha
500 Mhz Pentium
440 Mhz Sparc
400 Mhz MIPS
300 Mhz MIPS
196 Mhz MIPS
135 Mhz PowerPC
SECONDS/NLOGN
0.0
0.00001
0.00002
0.00003
0.00004
10^3
10^4
10^5
10^6
10^7
NUMBER OF CITIES N

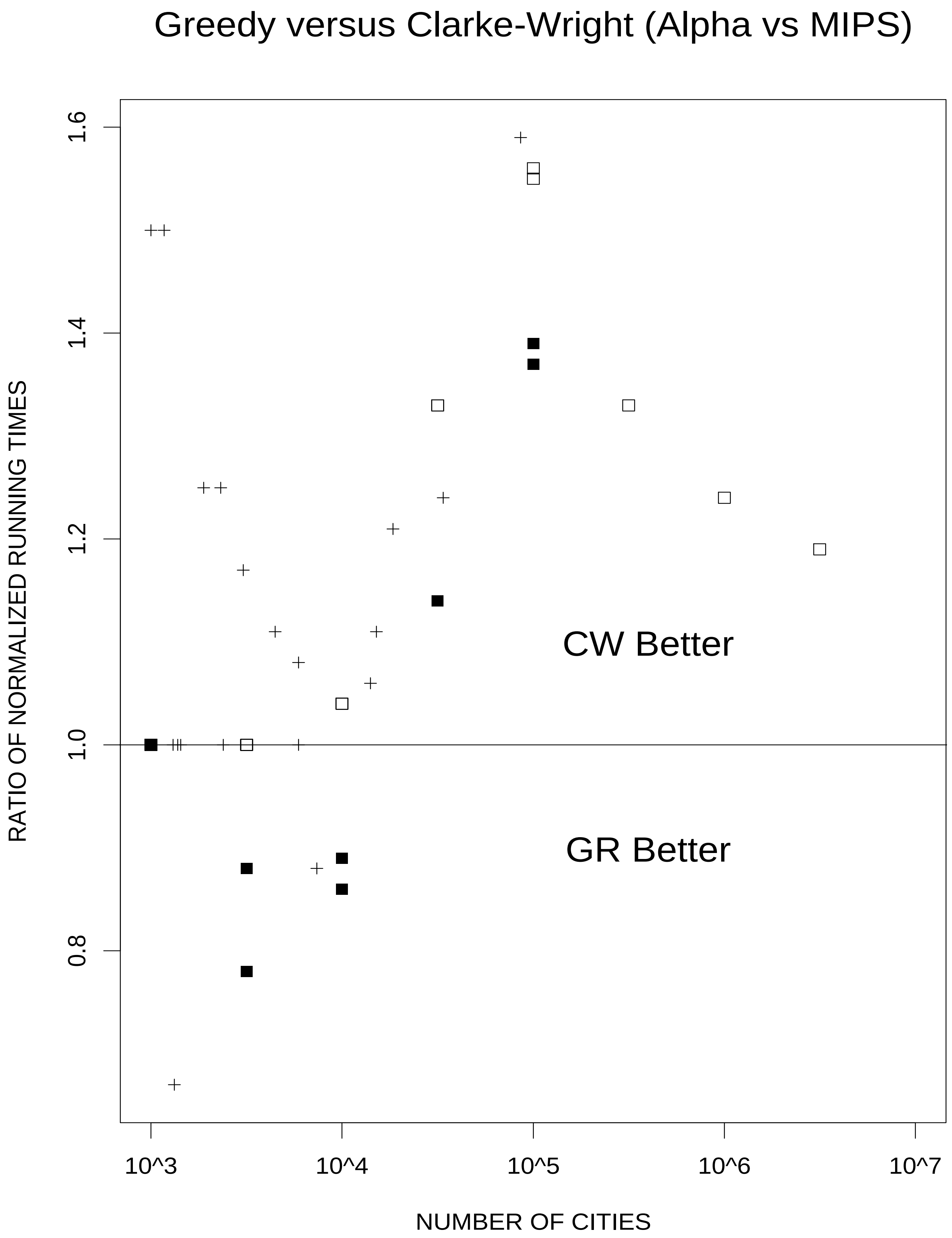


$$\text{Correction Factor} = \frac{\text{Benchmark Greedy time for Alpha}}{\text{Benchmark Greedy time for MIPS}}$$

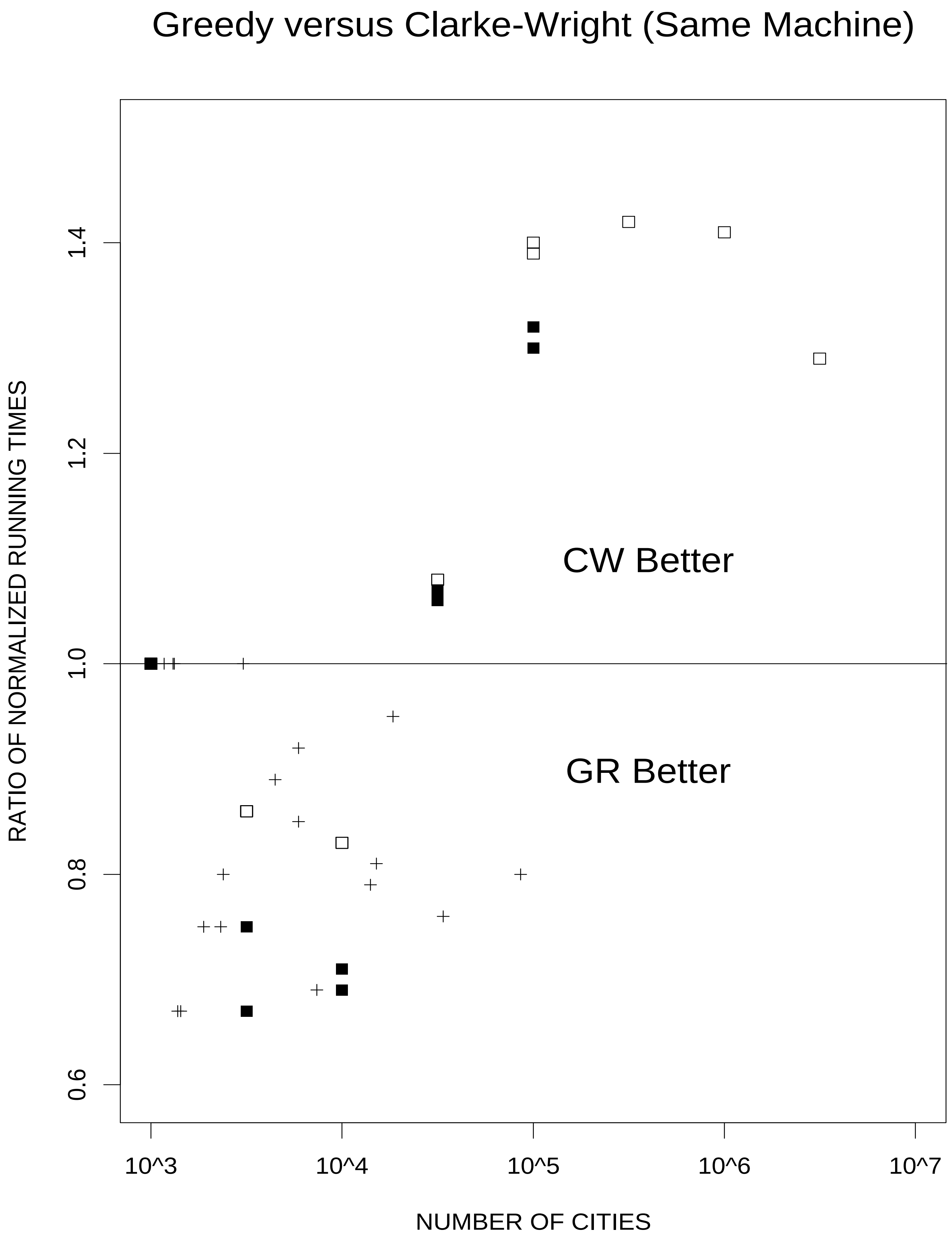

Errors in Running Time Normalization: Greedy Algorithm
Overestimate
Underestimate
MIPS NORMALIZED TIME/ALPHA ACTUAL TIME
1.4
1.2
1.0
0.8
0.6
0.4
10^3
10^4
10^5
10^6
10^7
NUMBER OF CITIES

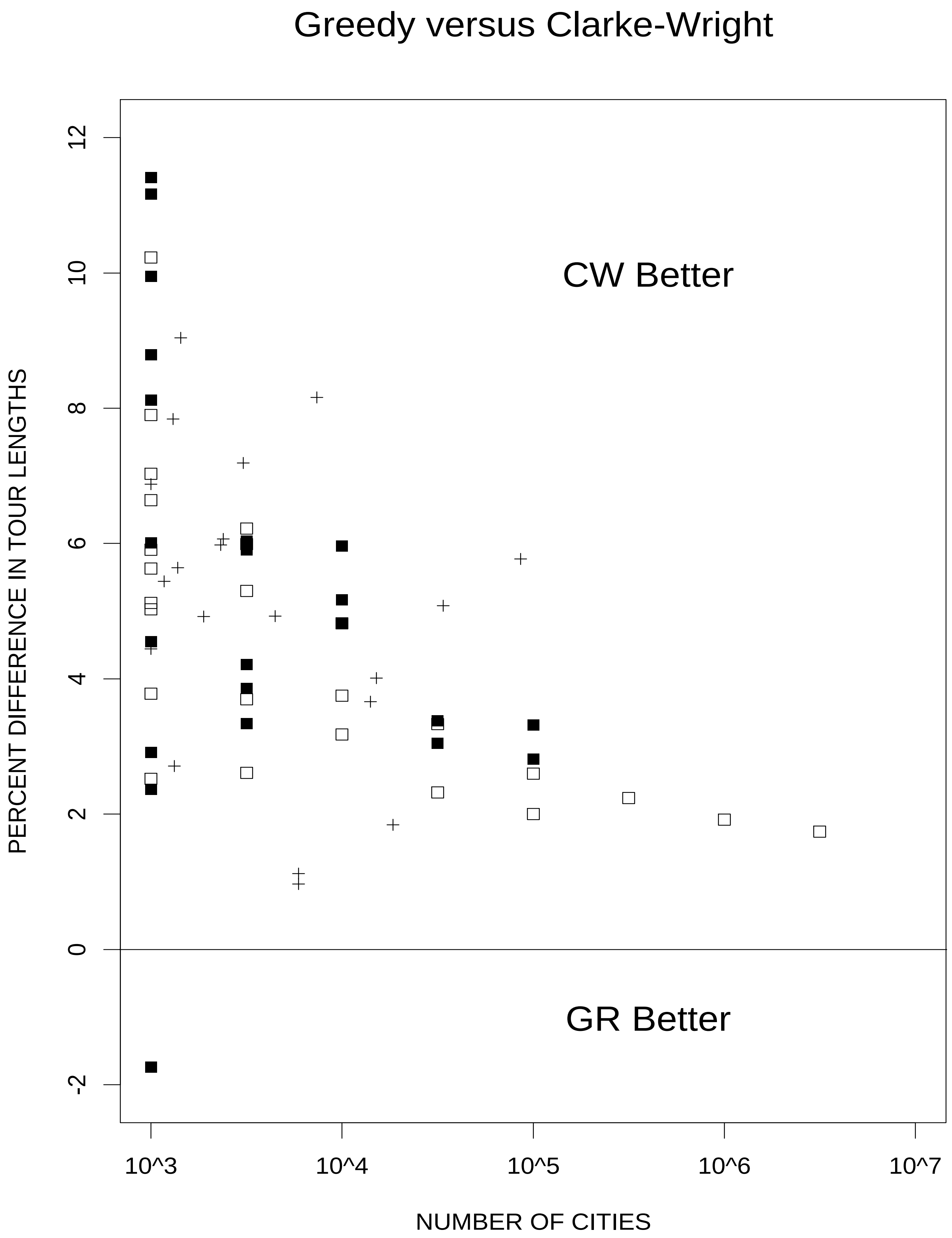


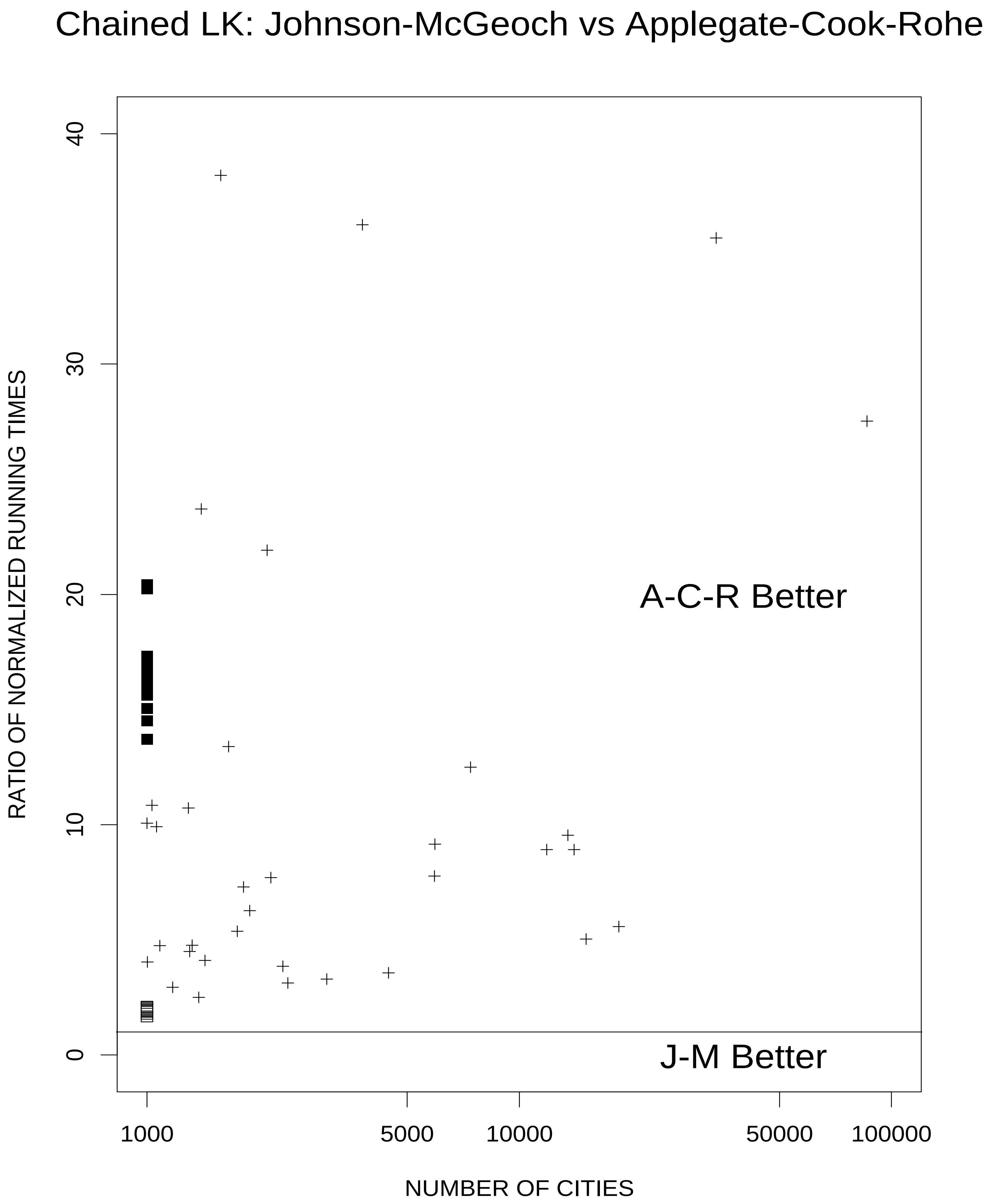
Errors in Running Time Normalization: Lin-Kernighan
Overestimate
Underestimate
MIPS NORMALIZED TIME/ALPHA ACTUAL TIME
2.0
1.5
1.0
0.5
10^3
10^4
10^5
10^6
10^7
NUMBER OF CITIES

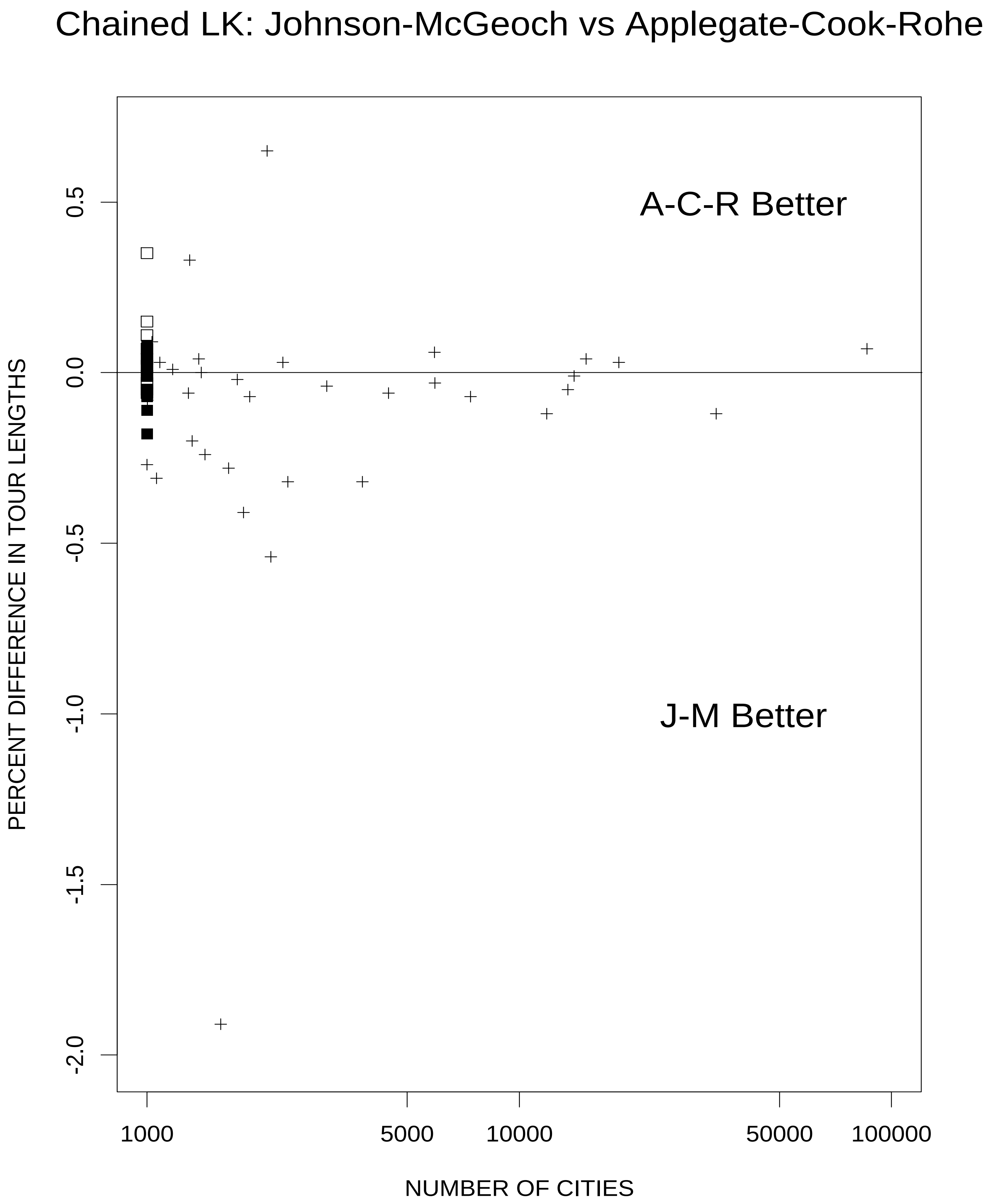


Greedy versus Clarke-Wright (Alpha vs MIPS)
RATIO OF NORMALIZED RUNNING TIMES
1.6
1.4
1.2
1.0
0.8
CW Better
GR Better
10^3
10^4
10^5
10^6
10^7
NUMBER OF CITIES


Greedy versus Clarke-Wright (Same Machine)
RATIO OF NORMALIZED RUNNING TIMES
0.6
0.8
1.0
1.2
1.4
CW Better
GR Better
10^3
10^4
10^5
10^6
10^7
NUMBER OF CITIES


Greedy versus Clarke-Wright
CW Better
GR Better
PERCENT DIFFERENCE IN TOUR LENGTHS
NUMBER OF CITIES
-2
0
2
4
6
8
10
12
10^3
10^4
10^5
10^6
10^7


Chained LK: Johnson-McGeoch vs Applegate-Cook-Rohe
RATIO OF NORMALIZED RUNNING TIMES
40
30
20
10
0
A-C-R Better
J-M Better
1000
5000
10000
50000
100000
NUMBER OF CITIES



Chained LK: Johnson-McGeoch vs Applegate-Cook-Rohe
A-C-R Better
J-M Better
PERCENT DIFFERENCE IN TOUR LENGTHS
NUMBER OF CITIES
0.5
0.0
-0.5
-1.0
-1.5
-2.0
1000
5000
10000
50000
100000

usa13509

Excess over HK Bound	Excess over Optimal	Normalized Running Time	Algorithm
17.25	16.48	0.28	greedy
4.96	4.27	87.41	2opt
3.32	2.64	1.23	acrlk
3.25	2.57	1.31	acrclk10
3.03	2.35	87.53	3opt
2.42	1.74	1.84	acrclk100
2.18	1.51	101.76	lk
1.37	0.71	6.69	acrclk1000
1.30	0.63	155.98	ilk.1N
1.04	0.38	276.02	ilk.3N
0.91	0.25	63.83	acrclkN
0.86	0.19	609.67	ilkN
0.00	-0.66	118.86	heldkarp
-0.12	-0.78	106.93	rhk1
-0.30	-0.95	26.15	rhk2
-0.35	-1.01	13.32	rhk3

3,162-City Random Distance Matrix

Excess over HK Bound	Excess over Optimal	Normalized Running Time	Algorithm
219.05	219.04	16.70	greedy
89.17	89.17	19.48	2opt
46.73	46.72	18.90	3opt
7.43	7.43	3.58	acrlk
6.08	6.08	4.77	acrclk10
4.66	4.66	12.56	acrclk100
4.18	4.18	20.19	lk
3.75	3.75	62.64	acrclk1000
2.72	2.72	31.51	ilk.1N
2.50	2.50	57.55	ilk.3N
2.31	2.31	121.50	ilkN
0.00	0.00	612.49	concorde
0.00	0.00	40.36	heldkarp

CONCLUSIONS

Yet to be derived...

Your Help Needed!