Parallel Sorting of Roughly-Sorted Sequences

CSCI 5172 Fall '16 Project

Anthony Pfaff, Jason Treadwell

December 6, 2016

1 The Array Sorting Problem

Sorting a collection according to some ordering among its items is among the most classic problems of computer science. A well-established result is the linearithmic (i.e. $O(n \lg n)$) optimal upper bound for sorting sequences of length n by comparison.

2 Sorting Roughly-Sorted Sequences

We can exploit the ordering of *roughly-sorted* sequences to sort them in $O(n \lg k)$ time, where k is the *radius* of a sequence S or the smallest k such that S is *k-sorted*.[2] A k-sorted sequence $\{a_0, a_1, \cdots, a_n\}$ satisfies $a_i \leq a_j \ \forall\ 1 \leq i \leq j \leq n, i \leq j - k$. Since an unsorted sequence can be at most n-sorted, the worst-case runtime of this algorithm has complexity $O(n \lg n)$.

3 Sequential Implementation

4 Sorting Arrays in Parallel

5 Parallel Radius Determination

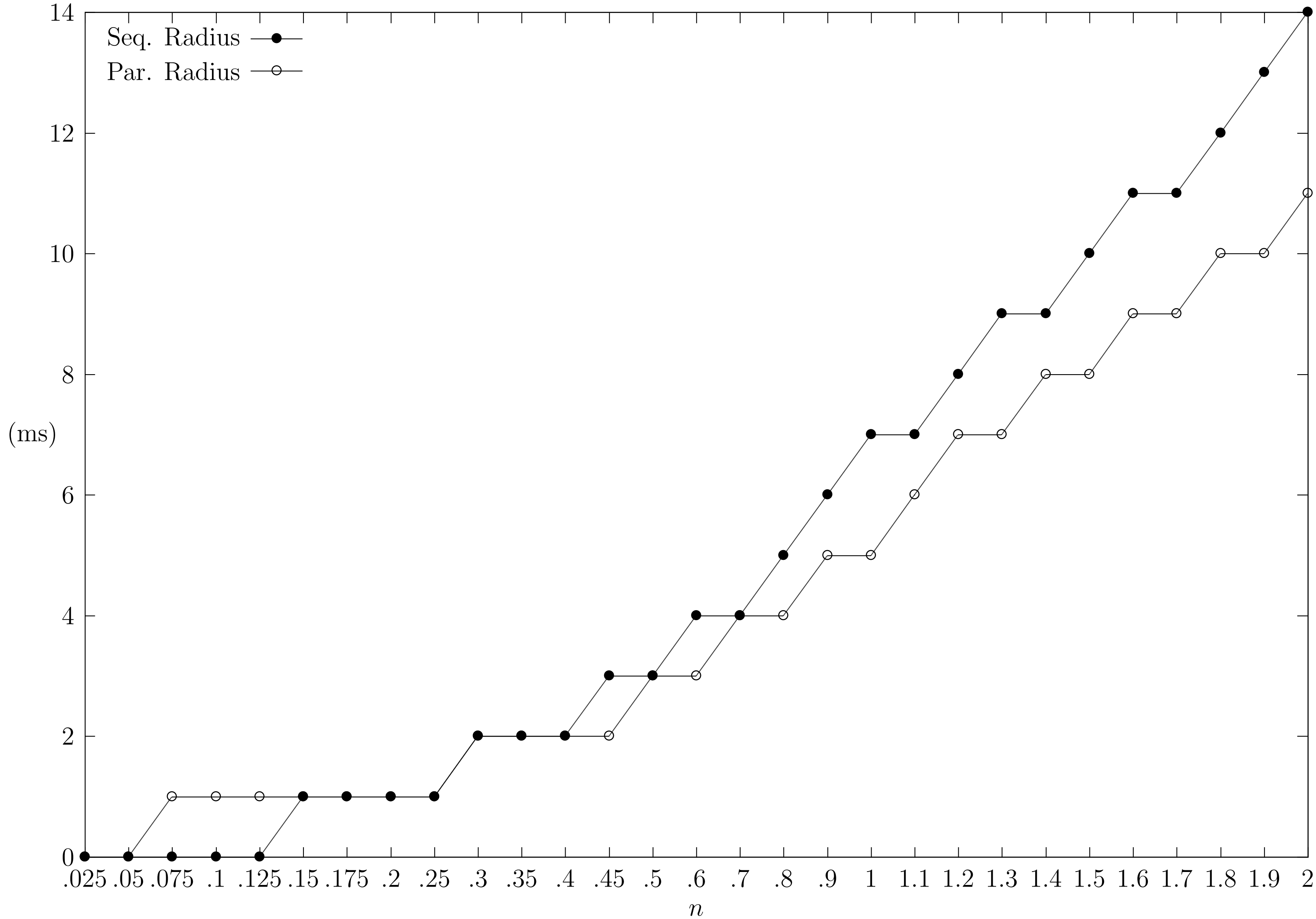

Figure 1: *Radius Determination Runtimes over Arrays of Length* $n \cdot 10^6$, $k = 2$

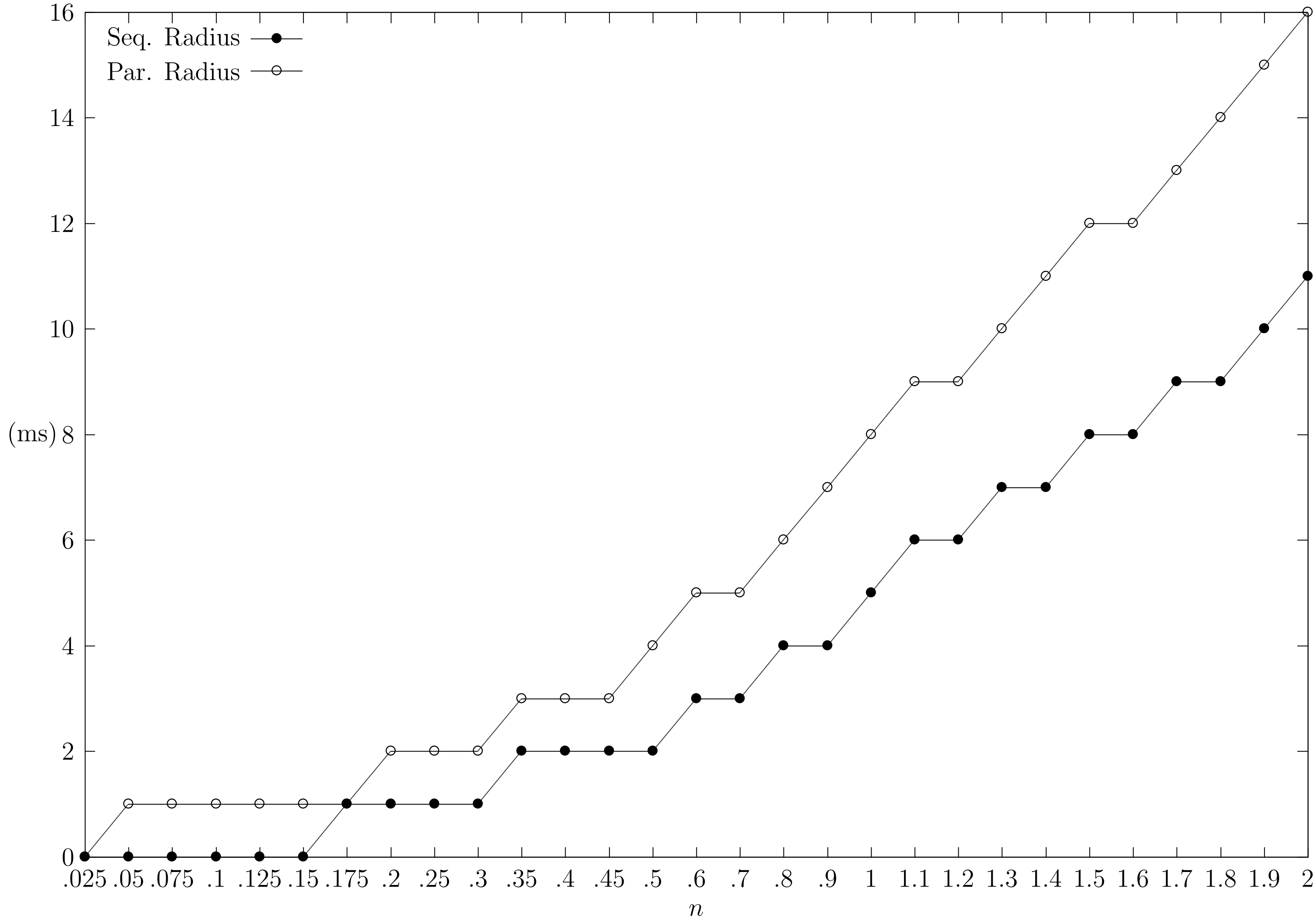

Figure 2: *Radius Determination Runtimes over Arrays of Length* $n \cdot 10^6$, $k = 100$

6 Parallel Roughsort Implementation and Results

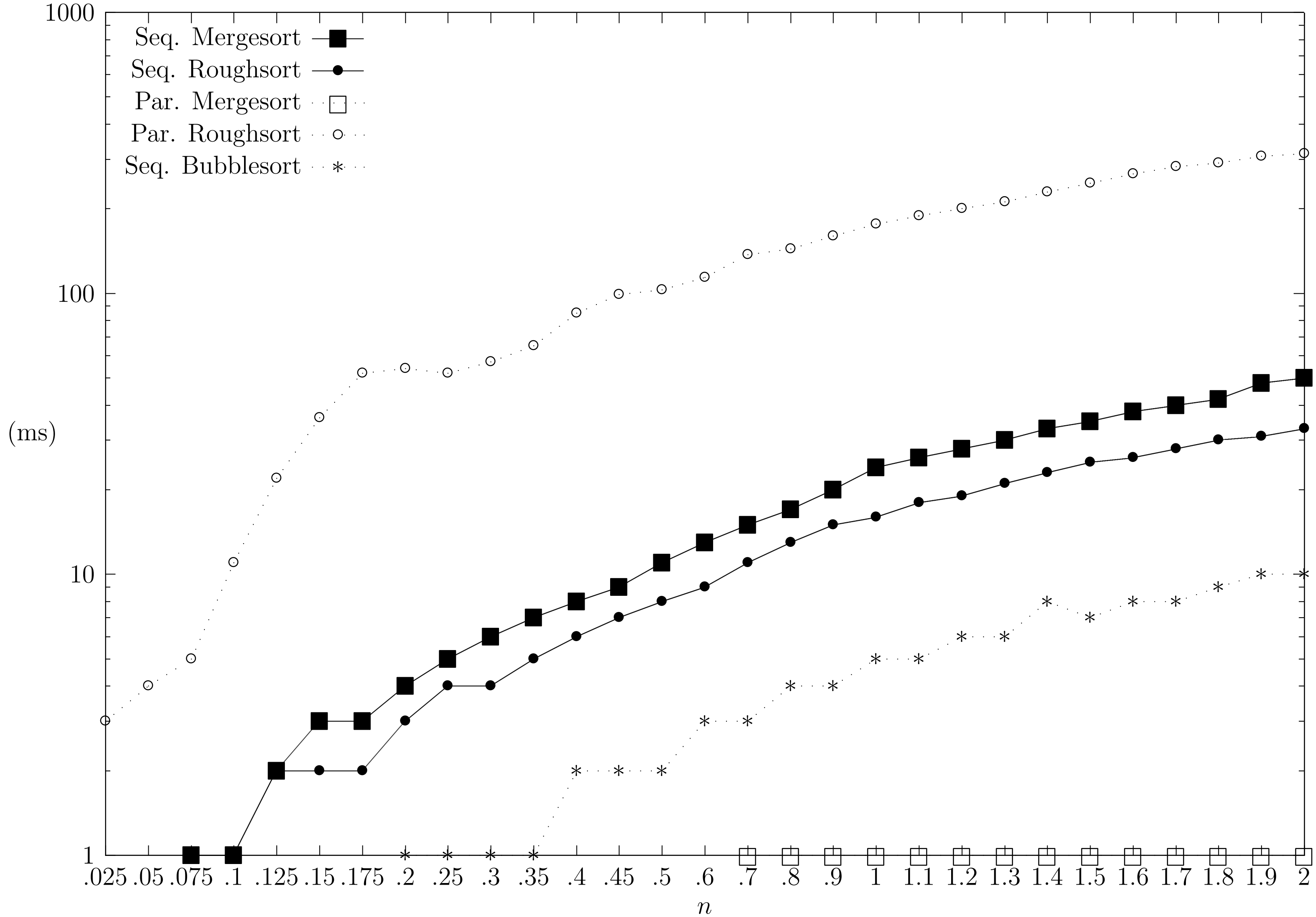

Figure 3: *Sort Runtimes over Arrays of Length* $n \cdot 10^6$, $k = 2$

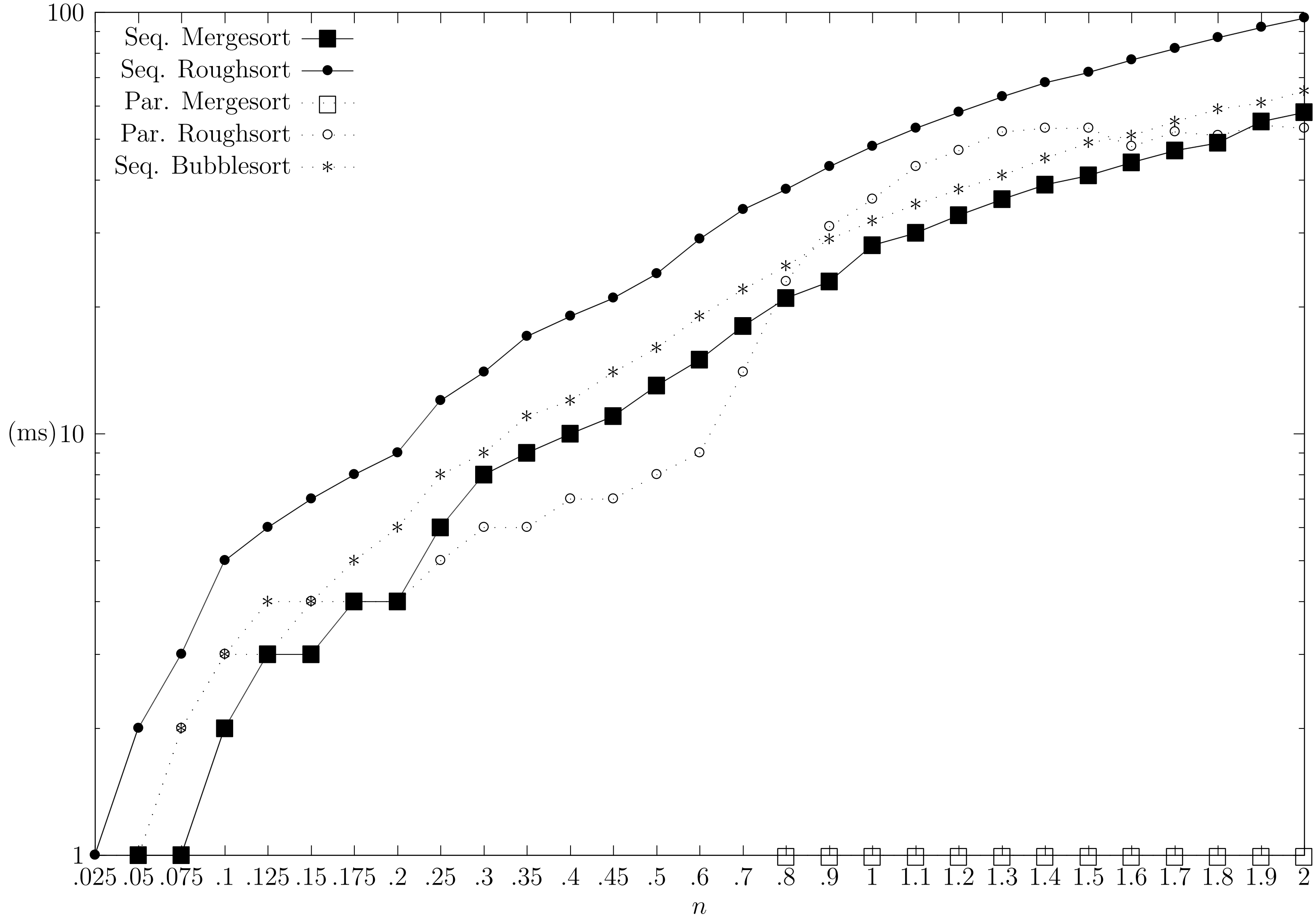

Figure 4: *Sort Runtimes over Arrays of Length* $n \cdot 10^6$, $k = 15$

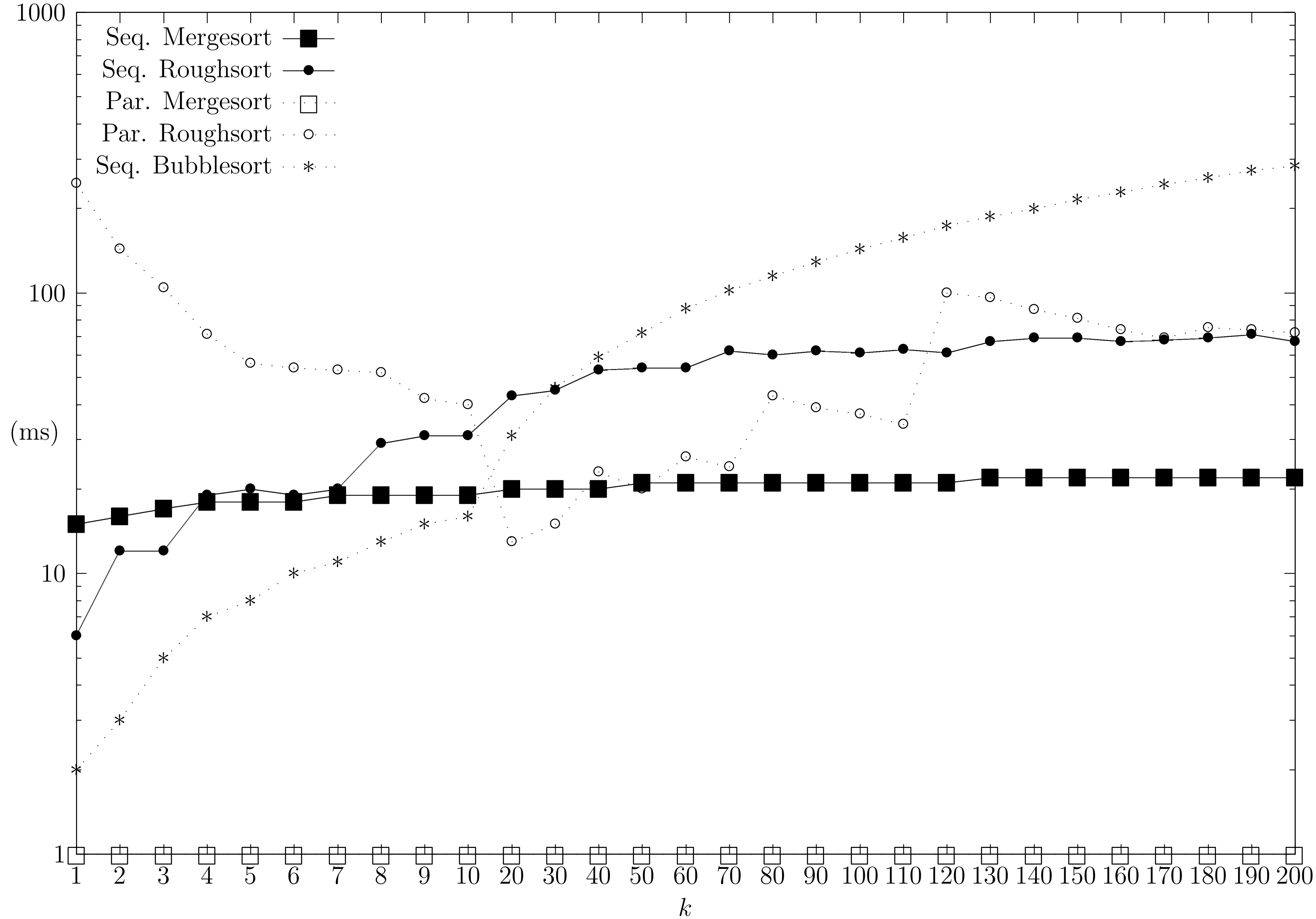

Figure 5: *Sort Runtimes over Arrays of Radius k*, $n = 0.75 \cdot 10^6$

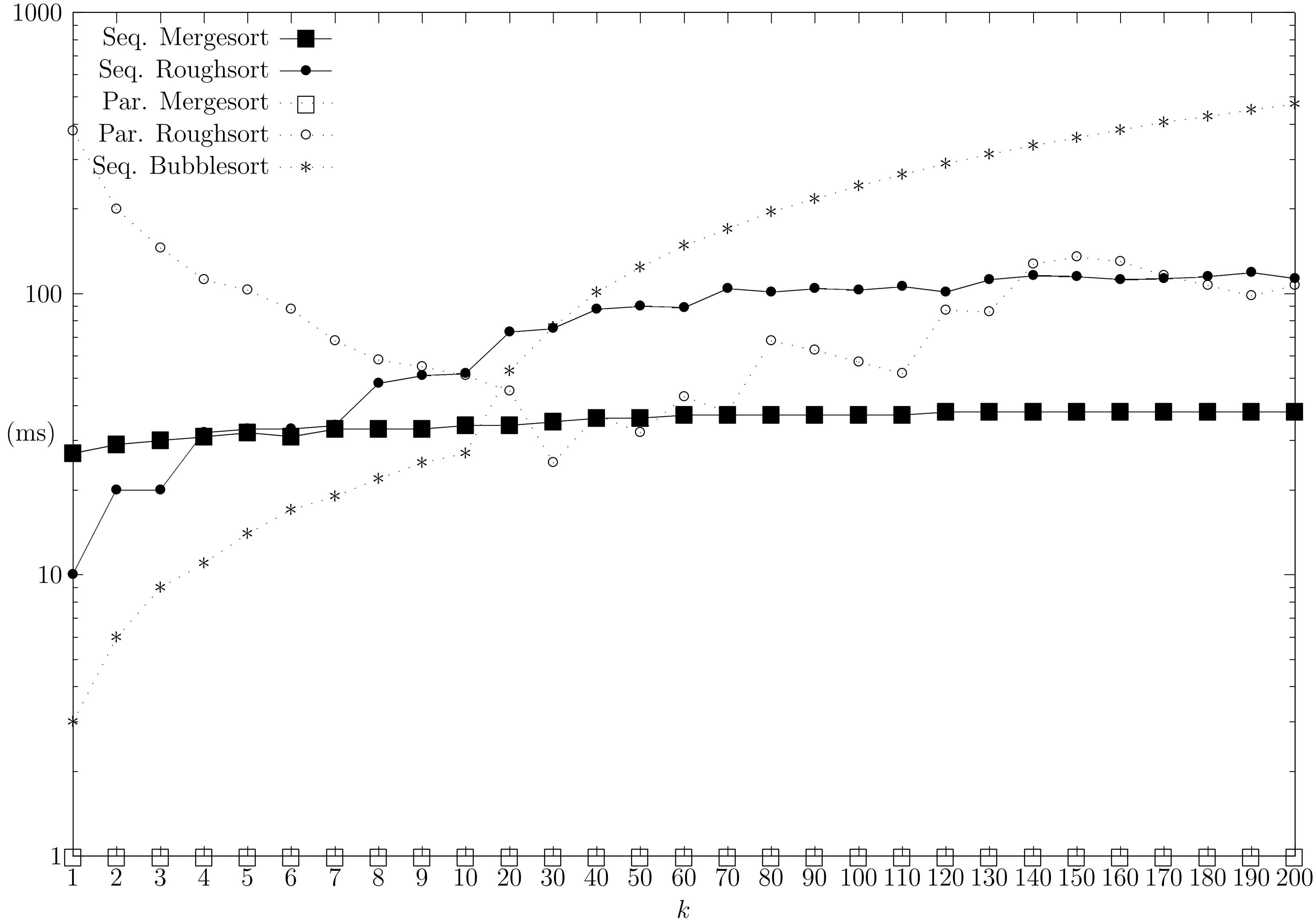

Figure 6: *Sort Runtimes over Arrays of Radius k, $n = 1.25 \cdot 10^6$*

7 Explanation of Results

8 Conclusion

References

[1] T. Altman and B. Chlebus. Sorting roughly sorted sequences in parallel. *Information Processing Letters*, 33(6), February 1990.

[2] T. Altman and Y. Igarashi. Roughly sorting: Sequential and parallel approach. *Journal of Information Processing*, 12(2), 1989.

[3] J. Cheng, M. Grossman, and T. McKercher. *Professional CUDA C Programming*. Wrox, 1st edition, September 2014.

[4] R. Cole and C. Yap. A parallel median algorithm. *Information Processing Letters*, 20(3), April 1985.

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 3rd edition, 2009.

[6] Forsell and Martti. On the performance and cost of some pram models on cmp hardware. *International Journal of Foundations of Computer Science*, 21(03):387–404, 2010.

[7] Intel. *Intel Digital Random Generator (DRNG) Software Implementation Guide*, 1.1 edition, August 2012.

[8] D. E. Knuth. *The Art of Computer Programming*, volume 3. Addison-Wesley, 2nd edition, 1998.

[9] N. Leischner, V. Osipov, and P. Sanders. GPU sample sort. September 2009.

[10] NVIDIA Corporation. CUDA C Programming Guide. `https://docs.nvidia.com/cuda/cuda-c-programming-guide`, September 2016. Accessed: 2016-12-06.

[11] OrangeOwl. Sorting with CUDA libraries. `http://www.orangeowlsolutions.com/archives/900`, May 2014. Accessed: 2016-12-06.

[12] S. Rajasekaran and J. H. Reif. Optimal and sublogarithmic time randomized parallel sorting algorithms. *SIAM Journal on Computing*, 18(3):594–607, 1989.

[13] R. Senser. *GPU Declarative Framework*. PhD thesis, University of Colorado Denver, November 2014.

[14] Y. Shiloach and U. Vishkin. *Finding the maximum, merging and sorting in a parallel computation model*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1981.

[15] Y. Shiloach and U. Vishkin. An O(log n) parallel connectivity algorithm. *Journal of Algorithms*, 3(1):57–67, 1982.

[16] N. Wilt. *The CUDA Handbook: A Comprehensive Guide to GPU Programming*. Addison-Wesley Professional, 1st edition, June 2013.