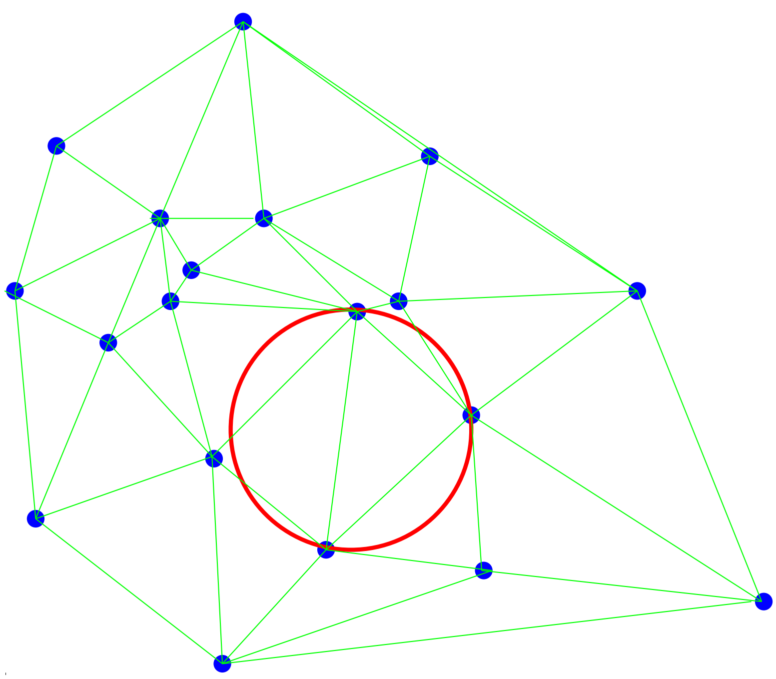
Developing a Practical Projection–Based Parallel Delaunay Algorithm

Guy Blelloch
Gary L. Miller
Dafna Talmor

Carnegie-Mellon University

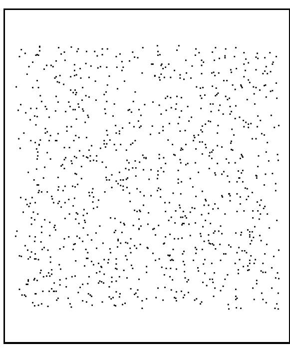
Delaunay Triangulation

- Given a set of points $P \in \mathbb{R}^2$ find their Delaunay triangulation.
- T is a Delaunay triangle if its circumcircle contains no points from P in its interior.
- Many applications; we are motivated by scientific computing applications such as mesh generation.

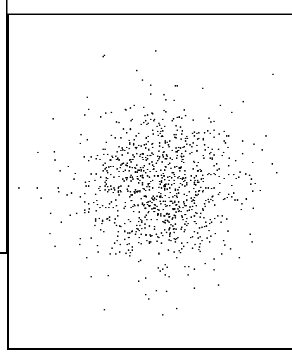


Goal of this Work

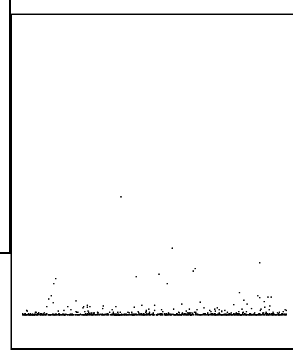
Developing a practical parallel Delaunay algorithm that works well for a variety of distributions



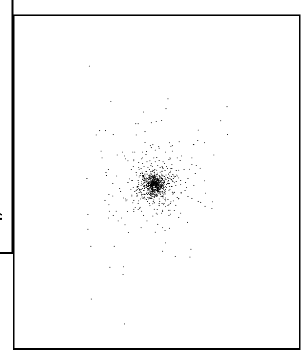
Uniform



Normal



Line



Kuzmin

Carnegie Mellon

Sequential Delaunay Algorithms

Variety of theoretical paradigms:

Algorithm by:	Paradigm	Major Subroutines
Shamos and Hoey [75] Guibas and Stolfi [83]	divide and conquer	stitching two subdiagrams
Dwyer [87]	divide and conquer with bucketing	stitching two subdiagrams
Fortune [87]	sweepline	advancing a front of Delaunay edges
...	incremental construction	planar point location
	⋮	⋮

All have been implemented and well–studied:

- Surveys by Su and Drysdale[95] and Fortune[90].
- Algorithms' run times within a factor of 2 of each other.
- **Dwyer's algorithm:**
  - Generally the best: run times, operation counts.
  - Guaranteed $O(n \log n)$.
  - On some distributions (e.g. uniform) expected $O(n)$.
  - Bucketing: merge subsolutions into rows; merge rows.

Carnegie Mellon

Parallel Delaunay Algorithms

- **Variety of theoretical paradigms:**

Algorithm	Paradigm	Major Subroutines
Aggarwal et al. [88]	divide and conquer	parallelize stitching step
Reif and Sen [89]	polling - Randomized divide and conquer	compute sub-diagram; divide with duplication
Edelsbrunner and Shi [91]	marriage before conquest: projection–based	planar point location; 2D CH; linear programming

- **Implementations not based on theory:**
  - Implementations based on bucketing algorithms and local search: Su[94], Merriam[92],Teng et al. [93]
  - Efficient only for uniform distributions: performance degrades to $O(n^2)$ work for clustered points.
  - Until now, no work addressed at general distributions.
- **The problem: inefficiency of theoretical algorithms**
  - High constant factors can not be offset by available parallelism.
  - **We have to develop more efficient variants**

Work–Efficiency

- **Work:** Total number of operations.
- **Estimating Efficiency:** Measuring the constant factors in work complexity.

> program A is α*–work efficient* with respect to program B if $w(A) \leq \frac{1}{\alpha}w(B)$.

Work–efficiency in our case:

- The base–line we picked is Dwyer's program.
- Work : floating point operation count.
- Experimental measurements over our test-suite.

Restating our goal: developing a parallel Delaunay algorithm which is

- work-efficient with respect to Dwyer's algorithm over our test-suite.
- parallel.

Which Paradigm to pick?

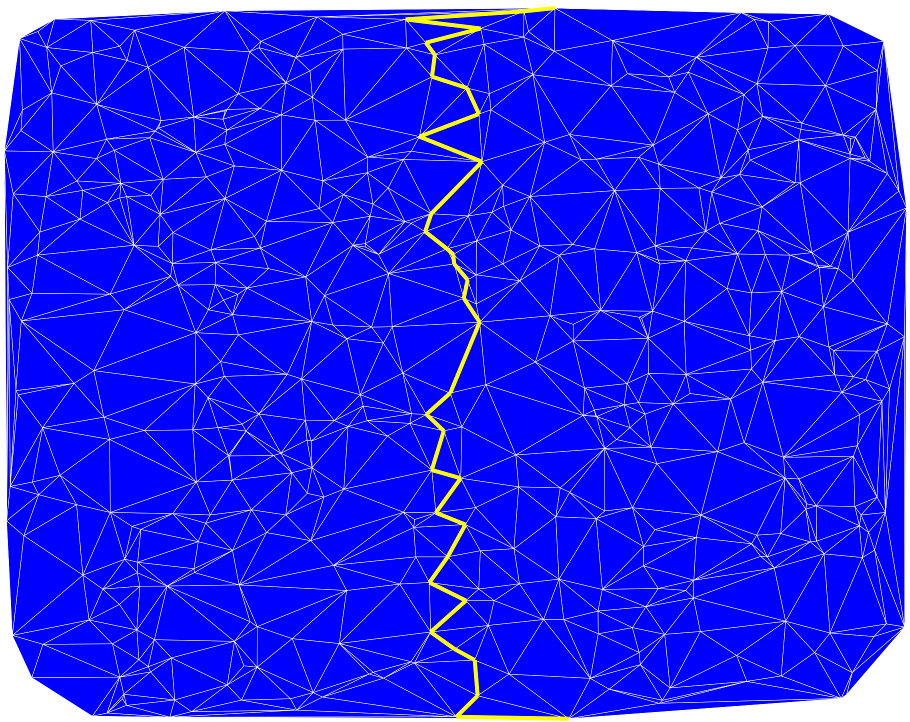
- **Obstacles to efficiency:**

Algorithm	obstacles
Aggarwal et al. "divide and conquer"	complicated data structures and subroutines
Reif and Sen "polling"	study by Su: duplication causes expansion factor of 6
Edelsbrunner and Shi "marriage before conquest"	complexity $O(n \log^2 n)$ subroutines: linear programming; planar point location; 2D convex hull

- **Our Algorithm:**
  - "Marriage before conquest".
  - Projection–based.
  - A simpler algorithm:
    - solves a simpler problem: Edelsbrunner and Shi find 3D CH, we find 2D Delaunay triangulation.
    - only subroutine used: 2D CH.

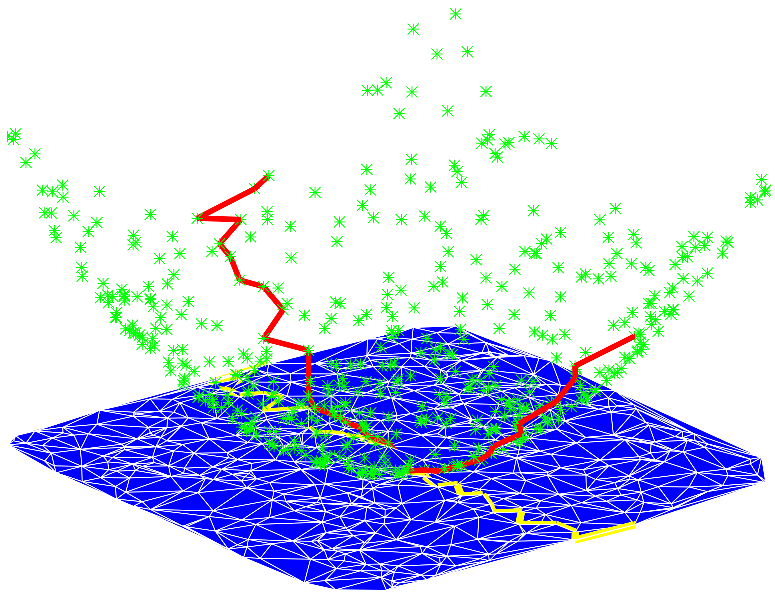
Carnegie
Mellon

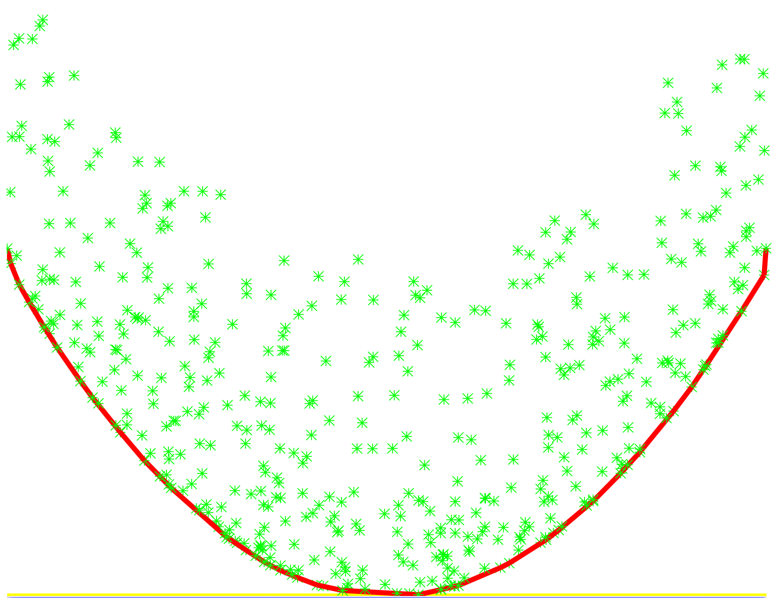
Algorithm: “Marriage before Conquest”



Carnegie Mellon

Algorithm: Projection–Based



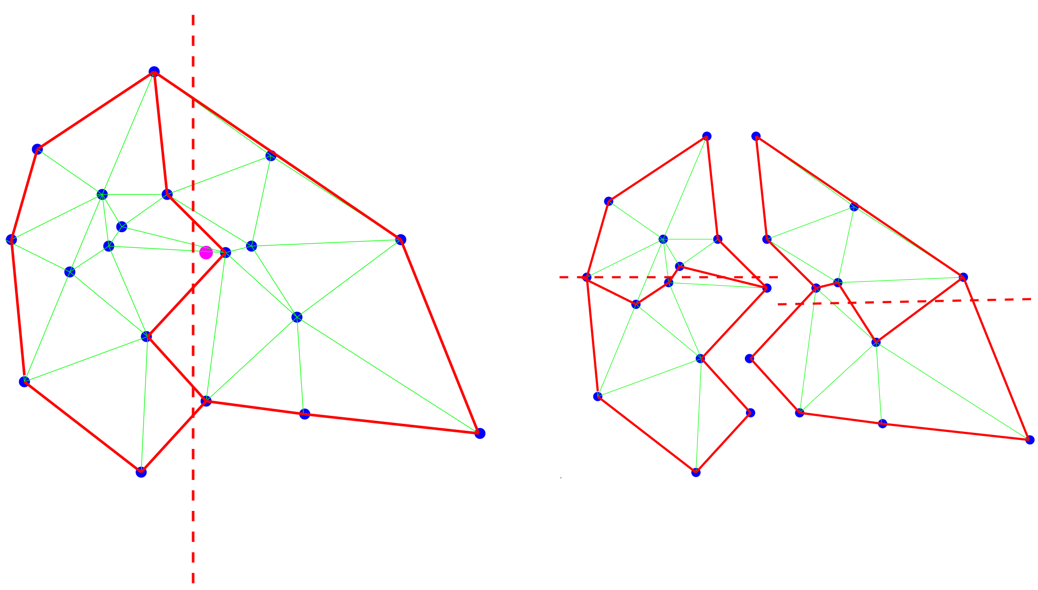


Algorithm: Quality of Divide

- **Lemma:** If the path is derived from a parabola centered on a line L, then the left sub–problem is composed of points:
  - Left of L or
  - On the path.

Two important implications:

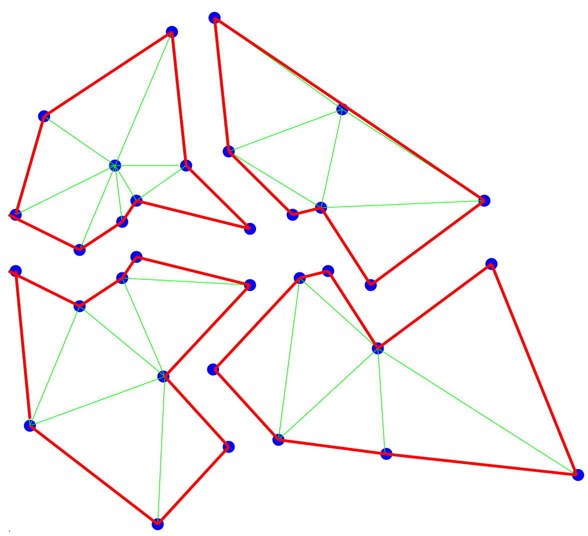
1. To decide if a point is in the left sub-problem, need only its orientation with respect to L (no planar point location).
2. If L is a median line, number of internal points is halved.



Carnegie Mellon

Algorithm: End Game (Theory)

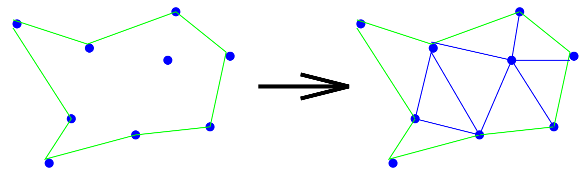
- No internal points - our strategy no longer $O(n \log n)$ work.
  - Edelsbrunner and Shi's strategy works till the end.
  - The strategy uses linear programming, ham sandwich cuts and planar point location.
- Finding triangulation of a polygon (theory):
  - $O(n)$ sequential algorithm by Wang and Chin [95].
  - Switch to other $O(n \log n)$ parallel algorithms.



Algorithm: Theoretical view

Our algorithm: using certain subroutines we get the first $O(n \log n)$ work projection–based algorithm.

Delaunay (P, B)

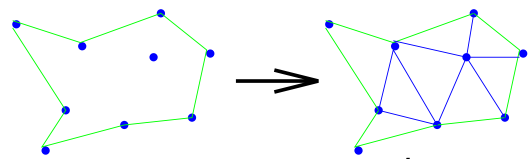


	depth	work
If (no internal points) then return OTHER_DELAUNAY(P)	$O(\log^2 n)$	$O(n \log n)$
find median line L=(x,0) or L=(0,y) Q = projection(P)	$O(\log^2 n)$ $O(1)$	$O(n)$ $O(n)$
find Delaunay path H using Q: H= OVERMARS(Q)	$O(\log^2 n)$	$O(n)$
split (P,B) into (P’,B’) and (P’’,B’’) return Delaunay(P’,B’) U Delaunay(P’’,B’’)	$O(1)$	$O(n)$
	$O(\log^3 n)$	$O(n \log n)$

Algorithm: Experimental view

Our implementation: worst case $O(n^2)$, efficient in practice.

Delaunay (P, B)



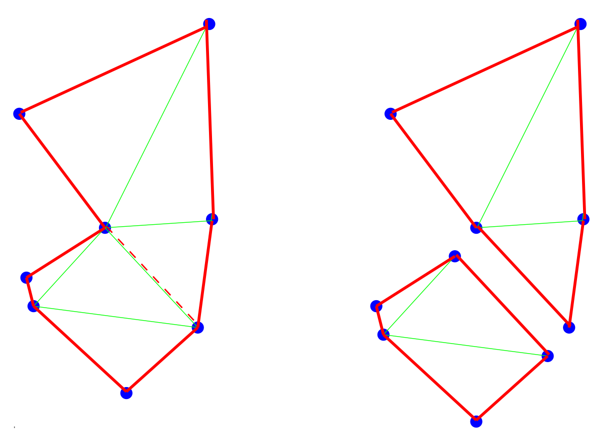
	depth	worst case work	experimental work
If (no internal points) then return OUR_END_GAME(B)	O(n)	$O(n^2)$	O(nlogn)
find median line L=(x,0) or L=(0,y) Q = projection(P)	$O(\log^2 n)$ O(1)	O(n) O(n)	O(n) O(n)
find Delaunay path H using Q: H= OUR_CH(Q)	$O(\log^2 n)$	O(nlogn)	O(n)
split (P,B) into (P',B') and (P'',B'') return Delaunay(P',B') U Delaunay(P'',B'')	O(1)	O(n)	O(n)
	O(n)	$O(n^2)$	O(nlogn)

Algorithm: End Game (Practice)

- End–game subproblems: 10-20 points.
- Switch strategy once problem size is small.

Our strategy for finding a triangulation of a simple Delaunay polygon:

- Pick some node u, find one edge out of it.
- Cost: small constant factor $O(n)$ work.
- Use edge to split into two Delaunay polygons.
- Worst case $O(n^2)$.



Carnegie Mellon

Algorithm: Convex Hull (Practice)

- Simple quickhull: $O(n^2)$.
- Guaranteed $O(n \log n)$ 2D CH:
  - Chan et al. [SODA 95]
  - An efficient version of Kirkpatrick and Seidel's ultimate convex hull.
- A hybrid algorithm:
  - Few levels of quickhull followed by the optimal algorithm:
  - Try to reduce problem size quickly using quickhull.
  - Switch to guaranteed method.

Carnegie Mellon

Experimental Techniques: Language

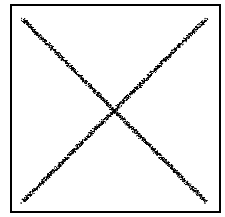
The NESL language:

- Nested data parallelism: well suited for irregular algorithms
- Good prototyping language:
  - Bridges between the PRAM model and the processor based model.
  - Measuring work and depth: complexity guarantees for primitives.
  - Portable to various parallel architectures.
  - Easy debugging on workstation.
  - Work in progress: compiled into C with MPI primitives.

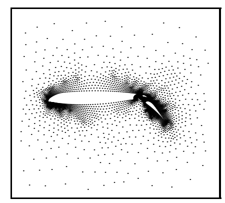
Goals of the NESL implementation

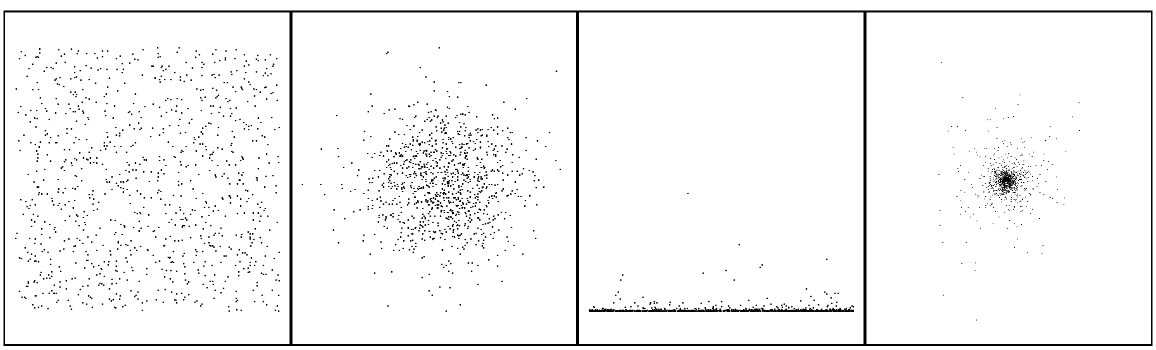
- Measure work efficiency
- Measure parallelism (depth)

Experimental Techniques: Test Suite



- Scientific Computing Motivated
  - No artificial distributions
- Related to the uniform distribution via a Lipschitz function
- Easy to generate
  - No "one–sized" examples.





Uniform Normal Line Kuzmin

Experimental Techniques: Measurements

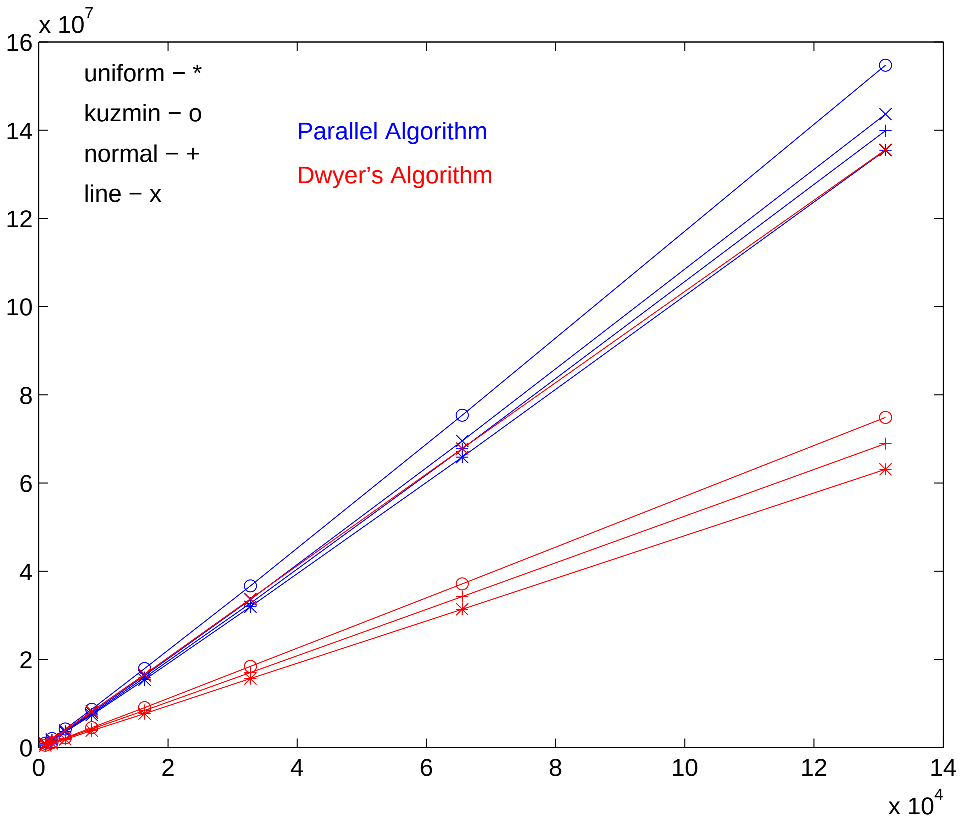
We compare the number of floating point operations between our parallel program and Dwyer's implementation:

- Correlated with run–time for this type of programs.
- Can be used to compare programs with different primitives.
- Primitive counts do not account for the following:
  - Orientation test(CCW): costs 5.
  - N orientation tests with the same line: cost $3N + 5$.
- Particular implementation of Dwyer's known to be efficient.

Our experimentation shows our program is close to 0.5-work-efficient.

Experimental Results: Efficiency

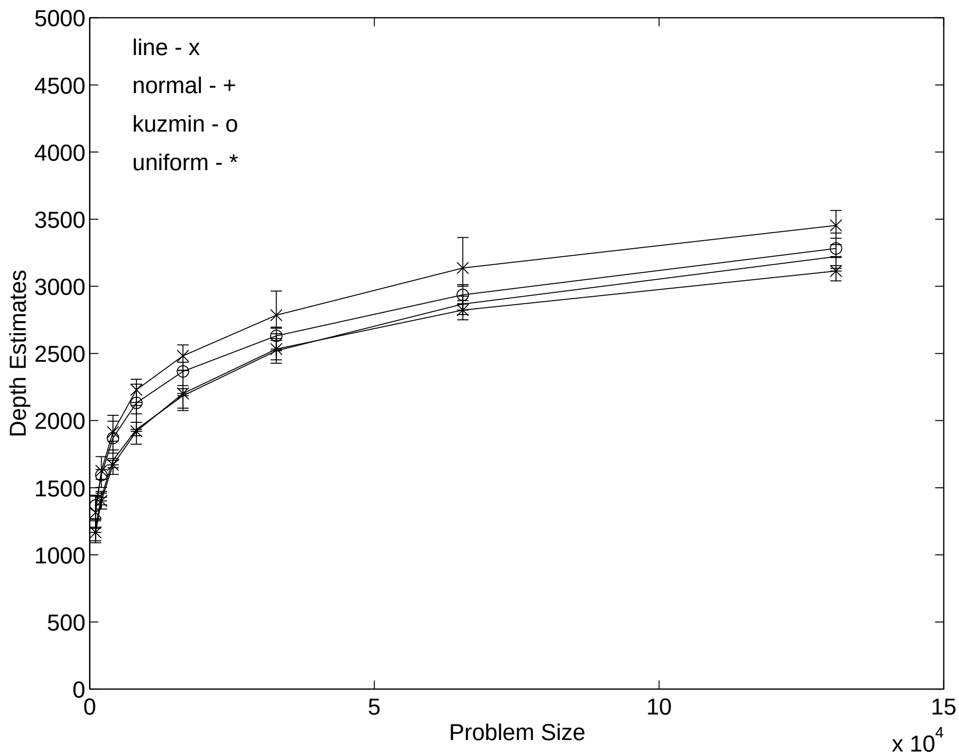
- Our algorithm performs almost uniformly on the various distributions.
- Dwyer's smarter cuts and merge order bring less savings on the Line distribution.



Carnegie Mellon

Experimental Results:Depth

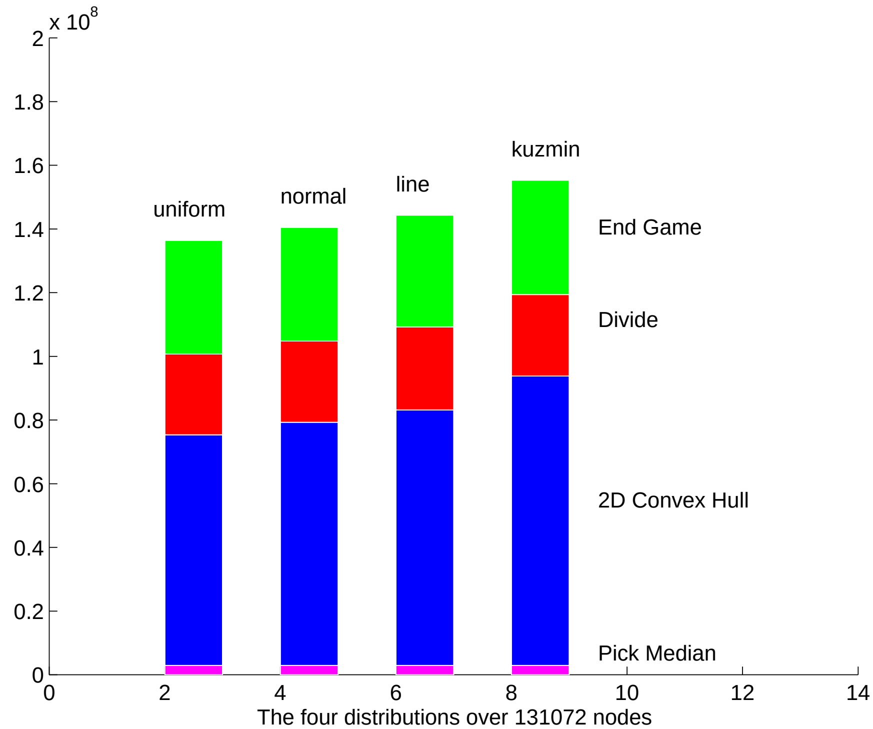
- Estimated the total depth of the call tree.
- Depth not strongly influenced by distribution.
- Parallelism $= \frac{\text{Work}}{\text{Depth}}$.
- E.g. for $N = 131072$ available parallelism is 45000.



Carnegie Mellon

Experimental Results: Work Division

- Convex Hull accounts for the largest portion of operations.
- Similar convex hull costs across the distributions.
- Similar over all work division across the distributions.



Carnegie Mellon

Conclusions and Continuations

Our contributions:

- We developed a parallel projection–based algorithm which is:
  - competitively work-efficient for a variety of distributions, even compared to the best sequential algorithms.
  - $O(n \log n)$ work (theoretically).
- An application–driven representative test–suite.

Future work:

- **Communication costs and run times:**
  - On–going work: translating to C with MPI primitives (Jonathan Hardwick).
- **Open Questions:**
  - Experimentally observed 2D CH behaviour: $O(n)$ expected run–time (for our test-suite).
  - Parallel Delaunay triangulation of simple polygons.