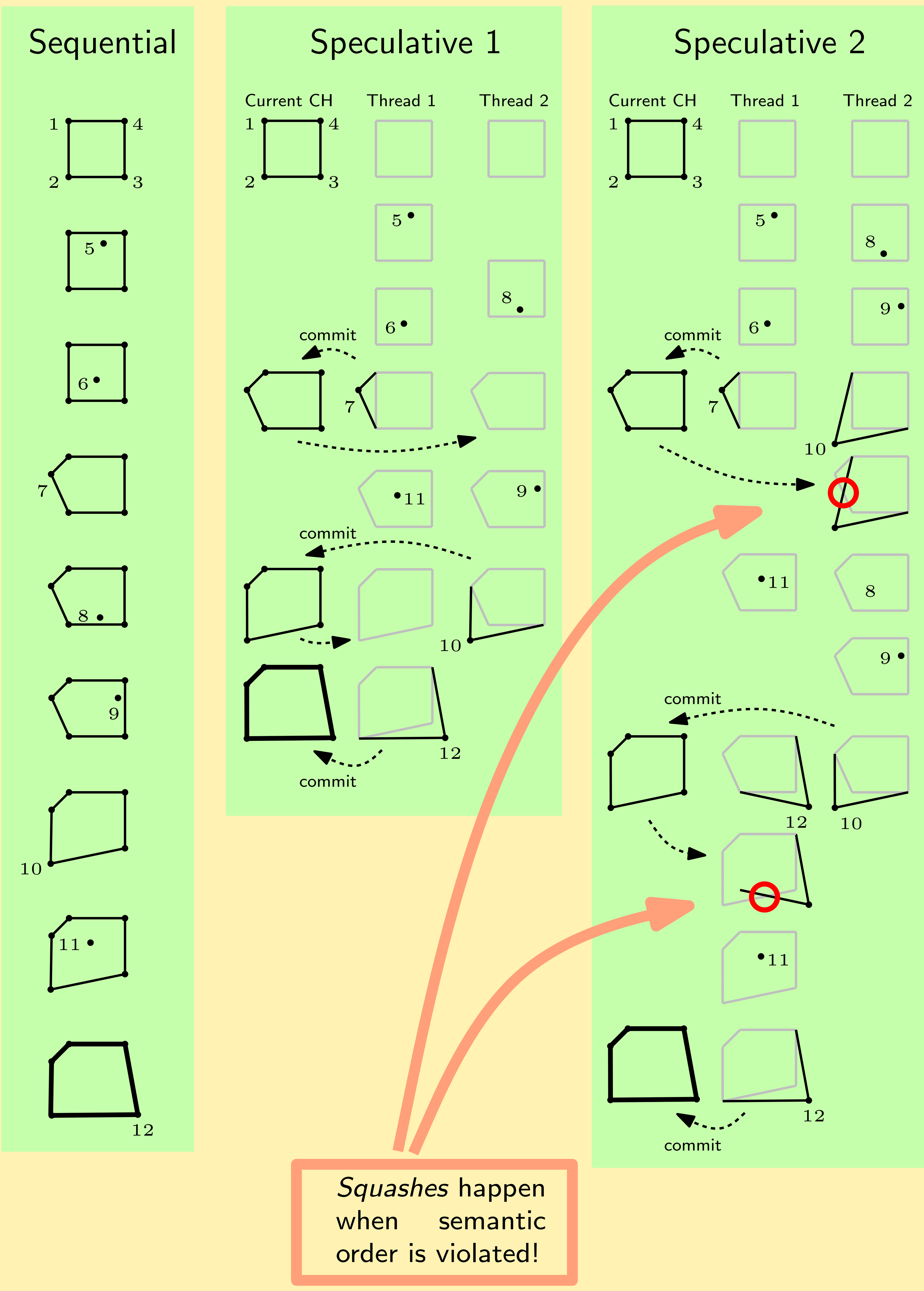
Parallelizing non-parallelizable geometric algorithms

D. R. Llanos (UVa) D.Orden (UAH) B. Palop (UVa)

Finding a way to use several kernels at the same time without developing parallel code is a very difficult task. In this work we focus on how to use speculative parallelization in order to run sequential algorithms in parallel (hopefully, decreasing the running time!) We focus on randomized incremental constructions, known to have competitive running times, and also known to be difficult (if not impossible) to parallelize by hand.

We show that, using specialized mechanisms that distribute the work-load among processors carefully, we can speedup the running time up to 6×.



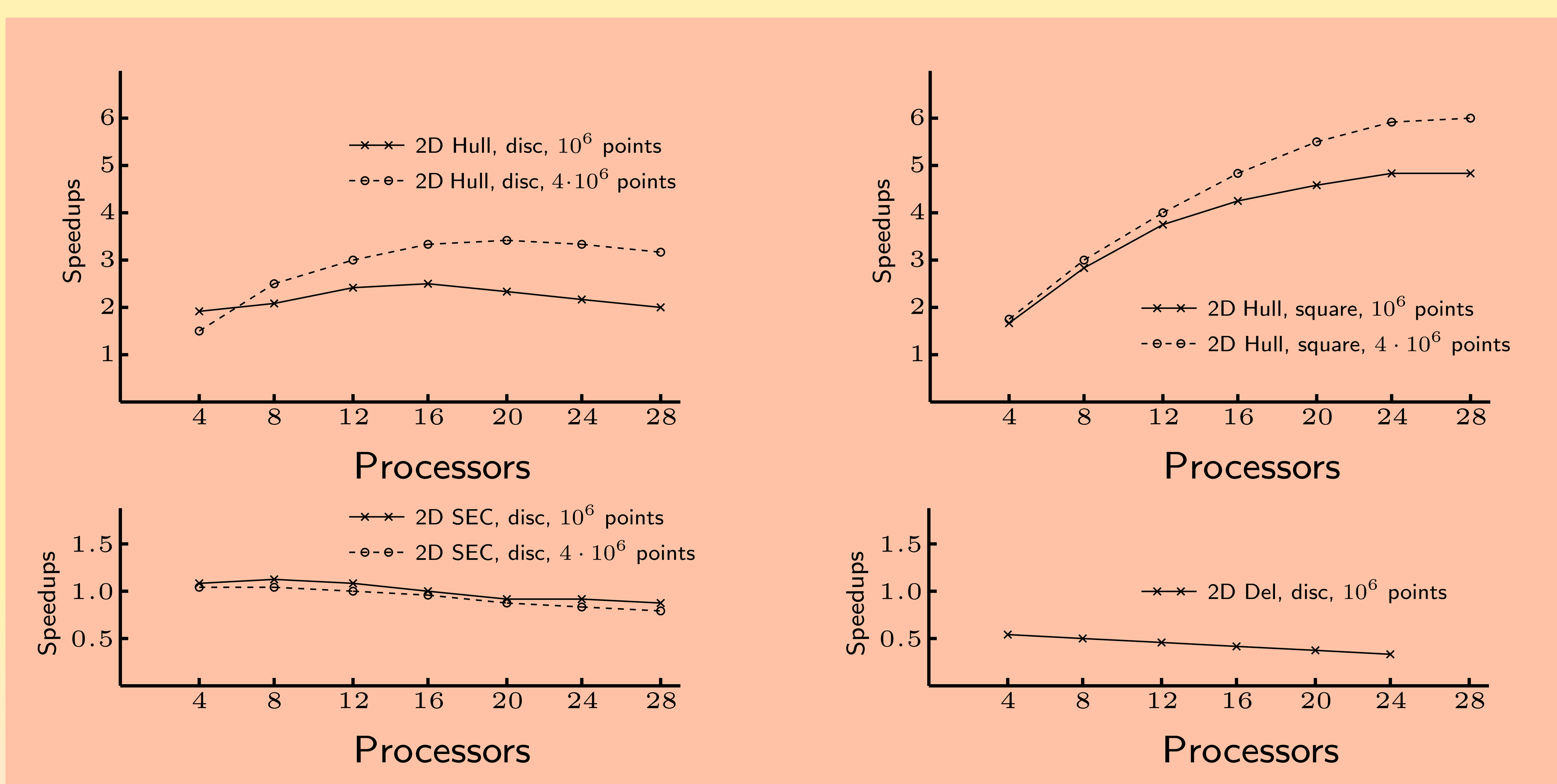
Compiler-based automatic parallelization fails to parallelize iterations in a loop if there is any suspicion that there might be some dependences among iterations.

Speculative parallelization [1] is, on the other hand, a very optimistic mechanism: Loops are executed in parallel until a dependence violation is found. In this case, all threads working on wrong data will be squashed and reexecuted.

Randomized Incremental Algorithms share some properties that make them specially interesting for speculative parallelization: Hard to parallelize, the incremental paradigm implies that every iteration has to rely on the results of the previous one. Fortunately, since the dependence pattern of these algorithms is known, we are able to estimate the number of dependences and choose an appropiate block size.

We have executed in parallel three randomized incremental algorithms: Clarkson et al.'s for the 2D Convex Hull problem [2], Welzl's for the 2D Smallest Enclosing Circle problem [3] and the one in [4] for the Delaunay triangulation.

We have used four different standard inputs: two sets of 10 and 40 million points randomly distributed inside a square (resp. a disc). These sets have been generated and shuffled with CGAL [5]. The experiments were performed on a Sun Fire 15K symmetric multiprocessor (SMP), equipped with 900 MHz UltraSparc III processors, with 1 GByte of shared memory per processor.



[1] M. Cintra, D. R. Llanos. Design space exploration of a software speculative parallelization scheme. *IEEE Transactions on Parallel and Distributed Systems*, 16(6):562–576, 2005.

[2] K. L. Clarkson, K. Mehlhorn, R. Seidel. Four results on randomized incremental constructions. *Computational Geometry: Theory and Applications*, 3(4):185–212, 1993.

[3] E. Welzl. Smallest enclosing disks (balls and ellipsoids). *New Results and New Trends in Computer Science*, H. Maurer, ed., 359–370, 1991.

[4] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf. Computational Geometry. Springer, 2000.

[5] CGAL, Computational Geometry Algorithms Library. `http://www.cgal.org`