We can even autogenerate those recursive algorithms (known as R-DPs)!

But R-DPs may have lower parallelism than iterative wavefront algorithms

Consider the DP for the Longest Common Subsequence (LCS) problem as an example

LCS has a simple R-DP using linear space

We can systematically transform an R-DP to WR-DP by incorporating the timing functions

Each DP implementation or specification

Recursive DP (R-DP)

AutoGen (PPoPP 2016)

Bellmania (OOPSLA 2016)

But LCS R-DP loses parallelism due to false dependencies

span: running time when #processors is unbounded.

We propose recursive wavefront algorithms (WR-DPs) – R-DPs with improved parallelism

WR-DP: approximates iterative wavefront order from within the recursive structure of an R-DP but retains all good properties of that R-DP.

We can systematically transform an R-DP to WR-DP by incorporating the timing functions

Completion time, $C(x)$ = number of times in wavefront order where a DP table cell $x$ is updated.

Start time, $S(x)$ = smallest number of times in wavefront order at which the update is performed.

End time, $E(x)$ = time that tracks the update with the largest timestep

Our WR-DPs use timing functions to launch recursive functions in wavefront order

Other work partially supported by CNS-1553510 (PI: Rezaul Chowdhury)

- An efficient cache-oblivious parallel Viterbi algorithm (Chowdhury, Ganapathi, Pradhan, Tithi, Xiao), Euro-Par 2016.
- External-memory RAM-oblivious GPU algorithms for dynamic programs (Chowdhury, Das, Ganapathi, Javanmard, Tschudi), under review.