Parallel and exact algorithms for path problems in directed graphs

How do theoreticians usually think of parallel computing? One can imagine a computation as a circuit consisting of gates performing elementary logical or arithmetic operations on either input values or outputs of other gates, without cyclic dependencies. While designing an efficient sequential algorithm means coming up with a circuit with possibly smallest number of gates (a quantity called *time*), in parallel algorithm design the objective is twofold. A circuit with possibly smallest number of gates, now called *work*, and possibly shortest longest chain of sequential dependencies between the gates, called *depth*, is sought.

The depth measures, accurately enough, to what extent the computation can be parallelized. Whereas it is unimaginable that a W-work computation is performed on a parallel machine (with arbitrarily many processors and shared memory) faster that proportionally to its depth D, the famous Brent's law states that it can indeed be scheduled on P processors to run in near-optimal time proportional to $W/P + D$.

Of course, an ideal parallel algorithm would match the best known sequential algorithm for the same problem in terms of work, and have depth bounded by a constant. There are few problems for which this has been proved possible, though. Achieving this seems very challenging, e.g., for most of the fundamental graph problems in directed graphs such as reachability or shortest paths. Early theoretical research on parallel algorithms focused on optimizing depth, under the constraint that work is polynomial. However, it is equally interesting to study what kind of *work-depth tradeoffs* are possible. For instance, in practical scenarios it is desirable to construct *work-efficient* algorithms, i.e., parallel algorithms whose work is close to state-of-the-art sequential time, and whose depth is optimized.

The general goal of this project is to study parallel aspects of some of the most fundamental computational problems on directed graphs – reachability (*"is there a path between two nodes s and t of a graph?"*), and its optimization variants: shortest paths (*"which path from s to t has smallest total weight?"*), and maximum flow (*"how many paths from s to t one can pick at once, so that no edge e is picked more than $c(e)$ times?"*).

More specifically, we will try to address the following questions:

- What kind of parallel work-depth tradeoffs are possible for these problems?
- What tradeoffs can be achieved deterministically, that is, without using randomization?
- What does it mean to compute exact solutions to graph optimization problems?
- What is the connection between parallel and exact optimization algorithms?