CSC2420: Algorithm Design, Analysis and Theory **Fall 2017**

Lecture 4: Dynamic Programming and Local Search

Lecturer: Allan Borodin, Nisarg Shah *Scribe: Lily Li*

4.1 Dynamic Programming Again

Example 4.1 Bellman-Ford *algorithm for finding the shortest path in a graph with possible negative edges (no negative cycles though). Running time is* $O(n^3)$*. Let* $C[i, v]$ *be the minimum cost of a simple path with path length at most* i *from source* s *to* v *(if there are no paths, cost is* ∞*). Again, either take an edge or don't. Calculate the cost as usual.*

You can actually do all-pairs-shortest path (APSP) in $O(n^3)$ *using DP. The idea is to find the mid-point of the expected path instead of the immediate predecessor. Currently no* $O(n^{3-\epsilon})$ *algorithm. There is an APSP-conjecture that one cannot do better than* $O\left(n^{3-o(1)}\right)$*.*

Note: some exponential algorithm are better than others. You can use a DP to get one of the better exponential algorithms. For example if you are doing ham path, there is an $O(n^2 2^n)$ algorithm which is better than the trivial $O(n!)$ algorithm.

4.1.1 ETH Diversion

Recall the exponential time hypothesis (ETH) and strong exponential time hypothesis (SETH). The former just says that for $\delta > 0$ such that $\forall \delta \exists k$ such that $k - SAT$ cannot be done in time $O(2^{\delta n})$. While the latter says that $3 - SAT$ not doable in time $O\left(2^{o(n)}\right)$.

4.1.2 More Applications

Consider Calinescu's [2011] resource allocation problem. Here we have the bandwidth define to be less than the total bandwidth of all jobs. You still need to schedule the jobs into the given bandwidth. Idea is to divide the jobs into *big* and *small* jobs. DP on the big jobs then try to fit in small jobs in the gaps.

Second example is Baptiste and Chuzhoy et al's throughput problems. The former started this discussion, his processing times are the same and jobs have deadlines.

4.2 Local Search

Vanilla local search paradigm:

1. Initialize some solution S
2. While there is a better solution S' in the *local neighbourhood* of S update S.

Oblivious and non-oblivious local search options: in the former your moves depends only on the objective function. In the latter there is a potential function related to the objective function and that is the function determining your moves.

Example 4.2 *(Weighted) Max-Cut Problem: given a undirected graph $G = (V, E)$ where edges have non-negative weights. Our goal is to find a partition (A, B) of V so as to maximize the value of a cut. Here you can just move nodes between A and B. This is an* 0.5*-approximation to the sum of all the edge weights (this is actually quite a bit better then the* 0.5*-approximation of the opt).*

Analysis of such algorithms proceeds by a locality gap (what is the worst that can happen if you are moving around locally). See slides for week 4 in the single move case. Note that the algorithm terminates but you might have to have exponentially many moves. It is unknown if a polynomial number of moves can be enough (examples of the exponential moves exist).

It turns out that if you are taking bigger neighbourhoods (more points allowed to be moved at each step) you don't get a much better approximation.