CSC 373: Algorithm Design and Analysis
Lecture 29

Allan Borodin

April 1, 2013

Announcements and Outline

Announcements

- Hand in assignments now
- Last tutorial today
- Last term test Wed in lecture

Today's outline

- Answer questions on problem set 3.
- "One size does not necessarily fit all": What kind of algorithms can be used for the unweighted and weighted MIS on various graph classes?
- The following two concluding topics will not be on the term test or final exam.
  1. A randomized algorithm for 2-SAT and its (improved exponential) extension to k-SAT (the underlying idea for "Walksat" algorithms)
  2. A randomized algorithm for primality testing.

One size (i.e. paradigm) does not necessarily fit all

We recall:

- For (unweighted) MIS on chordal graphs, a greedy algorithm (using the PEO) is optimal and provides a k-approximation for inductive k independent graphs.
- For weighted MIS on $k+1$ claw free graphs, a greedy algorithm (sorting by non decreasing weight) provides a k approximation.
- For unweighted MIS on $k+1$ claw free graphs, an oblivious local search provides a $\frac{k+1}{2}$ approximation.
- Can we use local search to optimally solve the unweighted MIS on interval and chordal graphs and then hopefully improve the approximation bound for MIS or even weighted MIS on inductive k independent graphs?

One size does not necessarily fit all continued

We recall:

- We cannot solve the weighted interval scheduling problem by a greedy algorithm (given a reasonably broad definition of what we mean by a greedy algorithm).
- Instead, we resorted to dynamic programming DP (see Lecture 6).
- One could hope that the same kind of DP could solve the weighted MIS problem for chordal graphs and then hopefully help to give an approximation for the inductive k-independence graphs.
- However, the DP we are using exploits an additional property of the PEO defined by non-decreasing finishing times:
  - For a given v_j, the conflicting vertices earlier in the ordering are a consecutive sequence of intervals $v_{j-1}, \ldots, v_i$ for some $i < j$.

"Stack algorithms"

- There is a "local ratio" or primal dual algorithm (with reverse delete) that can be modeled as a "stack algorithm" (utilizing the PEO) that optimally solves the weighted MIS for chordal graphs.
- (Aside) This stack algorithm can be reinterpreted as a DP for weighted MIS on chordal graphs but this DP is not as conceptually simple as in the weighted interval scheduling DP
- The same stack algorithm provides a k approximation for the weighted MIS for inductive k-independence graphs (using the k-PEO) to determine the order in which vertices are either rejected or placed on a stack.

Local ratio (simple primal dual) algorithm for weighted MIS on chordal graphs (Akcoglu et al)

Let $V = \{v_1, \ldots, v_n\}$ be sorted so that it is a PEO.
Stack $:= \varnothing$
For $i : 1, \ldots, n$
    $w'_i := w_i$ % w'_i will be the current residual profit
End For
For $j : 1, \ldots, n$
    Push v_j onto Stack if $w'_j > 0$
    For all vertices v_k with $k > j$ intersecting I_j
        $w'_k := w'_k - w'_j$
    End For
End For

Continuation of local ratio algorithm.

$\mathcal{S} := \varnothing$
While Stack $\neq \varnothing$
    Pop Stack and let v be vertex popped
    If v can be feasibly scheduled with already scheduled vertices in $\mathcal{S}$:
        $\mathcal{S} := \mathcal{S} \cup \{v\}$
    End If
End While

Some concluding on algorithms for graph classes

Concluding comments on this topic

- Of course, for a specific class of graphs, there can be much better approximations (e.g. for the intersection graphs of disc graphs by exploiting additional geometric properties of these graphs) that can be achieved.
- But for JISP, the 2-approximation is the best known polynomial time approximation for the weighted case. There is a better polynomial time approximation for the unweighted case but to achieve any meaningful approximation improvement the run time of the algorithm is too costly.

Papadimitriou's random walk algorithm for 2-SAT

- It is not difficult to show that 2-SAT (determining if a 2CNF formula is satisfiable) is efficiently computable (reducible to directed ST connectivity) whereas we know that 3-SAT is NP complete.
- We will provide a conceptually simple 1-sided randomized algorithm (RWALK) running in time $O(n^2)$ to show that 2-SAT is computationally easy.
- The same basic approach can be used to derive a randomized algorithm (which in turn has led to a deterministic variant of the idea) for 3SAT that runs in time $(1.324)^n$. It is a big open question if one can get time $2^{o(n)}$ algorithm for 3-SAT.
- This random walk idea is the basis for a widely used class of algorithms known as Walk-Sat algorithms for SAT problems.

Random walk algorithm for 2-SAT

RWALK algorithm for 2CNF formula F

Choose a random or arbitrary truth assignment τ
For $i = 1, 2, \ldots, (c \cdot n^2)$
    % with a sufficiently large c to obtain any desired probability of success
  If τ satisfies F **then**
    Report success and quit
  Else
    Let C be an unsatisfied clause and choose one of its literals ℓ_i at random
    Flip the truth value of the literal ℓ_i to change τ
  End If
End For

Claim

If f is satisfiable, then with say probability at least $\frac{1}{2}$ the RWALK algorithm will succeed in finding a satisfying truth assignment.

- We can either increase c or run RWALK many times to increase the probability of success.

Why RWALK works

Claim

Let τ^* be a truth assignment satisfying an n variable 2CNF F. Then we can view RWALK as a random walk on a line graph (with nodes $1, 2, \ldots, n$) that is trying to reach node n where node i indicates that τ matches τ^* in i coordinates.

- Since the clause C was not satisfied, at least one of its literals must be set different than τ^*. (It could be that both literals are different.)
- This means that the probability (in terms of the walk on the line) of getting closer to node n is at least $\frac{1}{2}$.
- It can happen that as we are randomly walking, we may come across another satisfying assignment but that will only shorten the time needed.
- What remains to be shown is that a random walk on the line with probability $\frac{1}{2}$ to move left or right will hit every point on the line in expected time $2n^2$.
- More generally, a uniform random walk (starting at any node) on a connected graph $G = (V, E)$ will hit all nodes in expected time $2|E||V|$.