15-859(D) Randomized Algorithms 11/5/98

Local Optimization Algorithms: Go with the winners

Presentation: Adam Berger
Notes: Bjarni Halldórsson

When optimizing and no guarenteed polynomial time algorithms are known one will usually try some sort of local search heuristic. The common basic idea behind them is that if one has found a good feasible point it is plausible that the optima is close to it so we should try to walk around it and see if we can get something better.

Two of the most commonly used heuristics are simulated annealing and genetic algorithms. Both try to mimic some sort of optimization done by nature.

- Simulated annealing tries to mimic the cooling process of metals. In essence a particle is in the beginning let wander freely around our search space (graph), this being the general behavior of particles in a warm metal, but then successively we let the particle wander less and walk more deterministically towards a better valued point, this corresponding to the bevavior of particals in a cooler metal.
- Genetic algorithms try to mimic the behavior of genetics. Between generations there is some mutation (random step) and if the algorithm is sexual two solutions are combined to give an offspring. Then the fittest of the new particals survive.

Both of these algorithm types, in particular simulated annealing, perform well in practice but have been hard to analyse. Questions we want answered include:

- Both algorithms require a choice of parameters (how fast should the cooling be?, how much mutation should we allow?, when to give up?). We don't know how to choose these parameters optimally.
- We have no type of convergence guarentee, don't know if we are close to optima or when we will get there.
- On what problem structure will these algorithms work well?

The "Go With the Winners" algorithm for layered graphs is an attempt to come up with a local improvement scheme that is easier to analyse.

We need a few very general assumptions about our problem:

- The number of feasible solutions is exponential in problem size (Else exhaustive search is easy).
- There is a neighbourhood structure that allows us to construct a graph G of all solutions with edges between two solutions if they are neighbours.
- The number of neighbours of a given solution is poly(problemsize).

- If a solution has value i then its neighbours have value $i-1$ or $i+1$ so the graph can be considered a layered graph with layers $L_0, L_1, ..., L_d$.
- d (number of layers) is poly(problemsize).

Most hard combinatorial optimization can be set up to satisfy these assumptions, as an example we look at Max-Clique: The set of feasible cliques is in general exponential in number of nodes (n). We define two cliques to be neighbours if one can be obtained from the other by adding or removing one node. Each clique then has at most n neighbours. Each layer contains all cliques of size i and the number of layers is at most $n+1$.

Basic idea of "Go With the Winners": Suppose we had a way of generating all solutions in some given layer $d-k$ then from that we could generate all solutions in layer $d-k+1$ etc. Even better if there are many paths from layer $d-k$ to an optima we only need to know a random sample of the solutions in layer $d-k$ to have a reasonable chance of getting somehow to an optima. Our basic idea is then to generate a random sample of a constant size, B of the solutions in layer k and use that random sample to generate a random sample of size B in layer $k+1$.

"Go with the winners" algorithm for layered graphs

- Generate B random solutions in layer i_0. Place a particle in each of these solutions. Let $i = i_0$.
- Stage i. Proceed in three phases
  - Predistribution: Calculate the down degrees of all the particles, if all are 0 then stop. Else remove old particles, create B new particles and place each one, independently, at the position of a particle x from the previous stage with probability proportional to its down degree.
  - Do a random walk in odd number of steps restricting your walks between layers i and $i+1$ so all new particals will end up in layer $i+1$.
  - Post distribute all particals inversely proportional their up degrees. I.e. for each particle x calculate u_x and let each of the B particles be at the position of x with probability $\frac{1/u_x}{\sum 1/u_x}$. Go to next stage.

What properties should the graph G have for the algorithm to work well?

- There are many paths that lead to an optima (this will allow a smaller sample size).
- The probability that you can get to the next level (and further levels) is not highly dependant on where in the current level your partical is located. I.e.
$$\frac{E_W(Pr^2(can\ get\ to\ level\ j \mid am\ at\ node\ W\ in\ level\ i))}{E_W(Pr(can\ get\ to\ level\ j \mid am\ at\ node\ W\ in\ level\ i)^2)}$$
is bounded.

- Number of nodes between layers decrease or increase at most by poly(problemsize). Else almost all children are a dead ends and probability we can continue is small.

- Each node level $i+1$ has a decent number of parents in level i (this is needed for our random walk to work).

Max-Clique(again): There are always very many paths leading to the optima, we note for instance that the number of solutions at level d is at least $\binom{d}{\frac{d}{2}}$ (exponential in d) even if there is only one local optima. So for Max-Clique we have few nodes in the low and high levels but exponentially many in the medium levels. Most algorithms would thus spend a lot of effort in the medium levels and many like "Random Restarts" or "Complete Search" usually never get out of that region. In the "Go With the Winners" framework we however spend equal time at each level and it thus should be efficient given that our solution graph G meets the structural properties needed.

References

[1] David Aldous, Umesh Vazirani, *"Go With the Winners" Algorithms*, FOCS '94, 492-501

[2] Tassos Dimitriou, Russell Impagliazzo, *Towards an analysis of local optimization algorithms*, STOC '96, 304-313