Project Title: Average-Case Hardness of 3-SAT

Project Mentor(s): Pravesh Kothari

Project Web Page: Include the URL of a web page which you will be maintaining for your group. This web page will initially contain a link to your proposal, and you will later add links to your various milestone reports, etc.

Project Description: Include all of the information requested above for your oral presentation

Who:

Mentor: Professor Kothari

What:

In 3-SAT, we are given n boolean variables and m 3-varaible clauses (clauses are of the form a OR b OR c). We are then asked whether it is possible to satisfy all m clauses with some Boolean assignment to the n variables. 3-SAT is known to be NP-hard, meaning that unless P = NP, we don't believe that we can, in the worst case, solve this problem in polynomial time. In practice, however, we're not as worried about worst-case runtime as we are about average-case runtime, which suggests interest in understanding how quickly we can solve 3-SAT on random instances, for some notion of "solve" and some notion of "random".

In particular, a well-posed and studied version of this question is to ask what the smallest guaranteed asymptotic bound we can achieve on an algorithm that always returns either "not satisfiable" or "don't know" such that, when run on a random instance, our algorithm 1) is always correct when it returns "not satisfiable" and 2) returns not satisfiable with high probability. Here, a random instance is generated by taking m uniformly random clauses of the variables and independently negating each variable in each clause with probability ½. There are many existing results that provide lower and upper bounds on the runtime of such algorithms for input instances in which m, as a function of n, lies in a particular range.

There are a couple different avenues for further investigation into this problem and tangentially related problems if time permits. Firstly, we can attempt to produce faster algorithms for where there's currently a gap between easiness and hardness results (e.g. where m/n lies between O(1) and O($\sqrt{n}$)). We could also consider generalizing easiness results to a variant of this problem that assumes even less randomization than current results. For example, what if instead of considering random 3-SAT instances, we force ourselves to provide a guarantee over worst-case instances that have only been perturbed slightly (by, say, negating each literal independently with some probability). We can attempt to generalize easiness results to CSPs with non-boolean predicates (i.e. our variables are no longer just true or false, but can now take on a value in [k]). We can also attempt to prove average-case hardness results for 3-SAT or other problems by using the relatively novel concept of average-case reductions.

So What?

From a practical perspective, as was mentioned previously, there is more high-level interest in knowing how well algorithms can perform on random instances than on worst-case instances, which is what this research addresses. (The guarantees provided by our algorithms, however, may be too convoluted to be of much practical use.)

From a theoretical perspective, most believe that we cannot hope to solve NP-complete problems in polynomial time, so it's naturally an interesting question to ask whether we can find polynomial time solutions to slightly easier problems and then to ask what's the least amount of "easiness" we can add to our problem and still come up with an efficient solution. In this case, we make our problem easier by considering random instances and having some error tolerance by allowing an "I don't know" response. The proposed research questions, if addressed successfully, could provide more insight into this question of how quickly we can tackle random CSP instances by either a) improving upon the most efficient known algorithm for solving randomized 3-SAT for a fixed notion of "randomized", b) decreasing the degree of randomness needed to achieve some level of efficiency for randomized 3-SAT, or c) by generalizing existing results from 3-SAT to CSPs with non-boolean predicates. Since 3-SAT is one of the simplest NP-complete problem to state, it's natural to focus research on this problem, however, it's also possible that these results would have implications for other problems of practical interest.

Project Goals (75%, 100%, 125%):

75%: understand relevant theory and figure out how to generalize some 3-SAT easiness results to non-boolean predicates

100%: 75% + use average-case reduction framework to prove some non-trivial average-case hardness result

125%: 100% + generalize some 3-SAT easiness result to a model involving a lesser degree of randomization, or provide a result that narrows the gap between existing easiness and hardness randomized 3-SAT results

Milestones:

1st Technical Milestone for 07-300:

Read relevant talks and papers (such as the ones listed in "Literature Search")

Bi-weekly Milestones for 07-400:

It's quite tricky to know in what direction this research will end up going, but here's one possible timeline:

February 1st: do background research on 3-SAT average-case results

February 15th: generalize 3-SAT results to CSPs with non-boolean predicates

March 1st: generalize 3-SAT results to CSPs with non-boolean predicates

March 15th: do background research on average-case reductions

March 29th: prove some non-trivial average-case reduction result

April 12th: prove some non-trivial average-case reduction result

April 26th: work on generalizing some 3-SAT easiness result to a model involving a lesser degree of randomization or providing a result that lessens the gap between existing easiness and hardness randomized 3-SAT results

Literature Search:

Relevant talks:

Peter Manohar: Algorithms and Certificates for Boolean CSP Refutation

Guy Bressler: A Few Simple Average-Case Reduction Techniques And Their Surprising Effectiveness

Relevant papers:

https://arxiv.org/abs/1605.00058

https://arxiv.org/abs/1701.04521

https://arxiv.org/abs/1305.0948

https://arxiv.org/abs/1304.0828

Resources Needed:

No software required