CS1200: Intro. to Algorithms and their Limitations Anshu & Vadhan

Problem Set 7

Harvard SEAS - Fall 2024 *Due: Wed Nov. 13, 2024 (11:59pm)*

Your name:
Collaborators:
No. of late days used on previous psets:
No. of late days used after including this pset:

The purpose of this problem set is to develop skills in implementing graph algorithms, appreciate the impact of different kinds of worst-case exponential algorithms in practice, and practice reducing problems to SAT.

1. (Another coloring algorithm) In the Github repository for PS7, we have given you basic data structures for graphs (in adjacency list representation) and colorings, an implementation of the coloring algorithm from ps5, and a variety of test cases (graphs) for coloring algorithms. For Windows users, we will work on getting a Google Colab up and running, so check Ed for updates – in the meantime, we recommend trying to run your code in WSL.

   (a) Implement the reduction from 3-coloring to SAT given in class in the function `sat_3_coloring`, producing an input that can be fed into the SAT Solver Glucose, and verify its correctness by running `python3 -m ps7_tests 3`.

   (b) Compare the efficiency of Exhaustive-Search 3-coloring, the $O(1.45^n)$-time MaximalIS+2COL algorithm for 3-coloring from problem set 5 (feel free to use the staff solution or your own implementations from problem set 5), and your implementation from Part 1a using `ps7_experiments`. In the experiments file, we've provided code to generate two types of graphs (lines of rings and clusters of independent sets) and some new hard graph instances. For each of those types of graphs, how many of the given instances, if any, can each algorithm solve within 10 seconds (same time limit as problem set 5)? You should fill out the table and briefly discuss your findings, as well as why these algorithms perform the way they do given what we have discussed in lecture.

Algorithm	Exhaustive	ISET BFS	SAT Color
# Solvable Ring Instances			
# Solvable Cluster Instances			
# Solvable Hard Graphs			

2. (Resolution) Use the algorithm `ResolutionInOrder` that we saw in Lecture 16 to decide the satisfiability of the following formulas, and use the algorithm `ExtractAssignment` to obtain a satisfying assignment for any that are satisfiable. (Please make sure to follow both algorithms *exactly*, including the order in which the clauses are processed. A correct final solution that does not show all of the intermediate steps of both algorithms will not receive full score.)

   (a) $\varphi(x_0, x_1, x_2, x_3) = (x_0 \vee x_1) \wedge (\neg x_2 \vee x_1 \vee x_3) \wedge (x_3 \vee \neg x_1) \wedge (\neg x_3) \wedge (x_1 \vee x_2)$.

   (b) $\varphi(x_0, x_1, x_2, x_3) = (x_0 \vee x_1 \vee \neg x_3) \wedge (x_2 \vee x_3) \wedge (x_0 \vee \neg x_2) \wedge (x_2)$.

(c) $\varphi(x_0, x_1, x_2) = (x_0) \wedge (\neg x_0 \vee x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)$.

3. (Reductions to SAT) In Lecture 13, we saw an efficient algorithm to solve the Maximum Matching problem in Bipartite Graphs. A variant of Maximum Matching is Maximum 3-D Matching, where we are given not a graph $G = (V, E)$ but a "3-uniform hypergraph" $H = (V, E)$, where the *hyperedges* $e \in E$ now are *triples* of vertices. Analogously to restricting to bipartite graphs, we restrict to *tripartite* hypergraphs where V is the union of 3 disjoint sets $V_0 \cup V_1 \cup V_2$ and every hyperedge $e \in E$ has exactly one vertex from each of V_0, V_1, V_2. For example, hyperedges may consist of compatible patient-donor-timeslot triples (if there are only certain timeslots in which a patient and donor are available for a surgery), or compatible surfer-surfboard-fins triples (if each surfer only likes to ride certain surfboards with certain fins installed). Now the goal is to find a maximum-sized set $M \subseteq E$ such that every vertex $v \in V$ is contained in at most one edge of M. Unfortunately, unlike matching in graphs, there is no polynomial-time algorithm known for Maximum 3-D Matching, even if we restrict to tripartite hypergraphs and even if we only want to find *complete matchings* — ones that match every vertex in V_0 (i.e. we are looking for a matching of size $|M| = |V_0|$):

Input	**:** Three disjoint finite sets of vertices V_0, V_1, V_2 and a set E of hyperedges such that each $e \in E$ contains exactly one vertex from each of V_0, V_1, and V_2.
Output	**:** A set M of hyperedges such that every vertex is in at most one hyperedge in M, and every vertex in V_0 is in a hyperedge in M (if one exists).

Computational Problem 3dCompleteMatching

(a) Show that there is a polynomial-time reduction of 3dCompleteMatching to SAT that, on hypergraphs with $n = |V_0| + |V_1| + |V_2|$ vertices and $m = |E|$ hyperedges, makes one oracle query on a formula with m variables and $O(n + m^2)$ clauses, and runs in time $O(n + m^2)$. Be sure to prove the correctness of your algorithm and analyze its runtime. Optionally, justify why the reduction can be implemented in time $O(m^2)$ (this can be part of an attempt at an R+ on this problem, but is not required for full credit).

Thus, even though the fastest known algorithms for 3dCompleteMatching run in exponential time, we can use SAT Solvers to solve it much more efficiently on many instances that arise in practice.

(b) (optional) Come up with a polynomial-time reduction for the version of 3dMatching where we are also given a number $k \in \mathbb{N}$ as part of the input and want to find a matching of size k (rather than one of size $|V_0|$). (Hint: you will probably want to use more than m boolean variables, at least $k \cdot m$, possibly more, depending on how you approach the problem.)

4. (Reflection): Describe two concrete ways in which you have supported, or will try to support, your classmates' learning in the course since the last time we asked this question (ps2). Be specific, connecting your answer to the structure of cs1200.

Note: As with the previous psets, you may include your answer in your PDF submission, but the answer should ultimately go into a separate Gradescope submission form.

Quick note on grading: Good responses are usually about a paragraph, with something like 7 or 8 sentences. Most importantly, please make sure your answer is specific to this class and your experiences in it. If your answer could have been edited lightly to apply to another class at Harvard, points will be taken off.

5. Once you're done with this problem set, please fill out this survey so that we can gather students' thoughts on the problem set, and the class in general. It's not required, but we really appreciate all responses!