Exercises AI: Principles and Techniques

Exercises Complexity, P and NP

Main aspect to test yourself here is: can you explain membership in P and in NP, and do you understand time complexity (best case, worst case, average case). You do not need to reproduce reductions at the exam, but you do need to be able to explain P, NP, NP-hard, NP-complete, and have some intuition on why ‘verifying’ appears to be easier than ‘deciding’. See the lecture slides and the lecture notes.

1. P, NP, reductions (old exam question)

a. Explain the terms NP, NP-hard, and NP-complete

b. We have a problem *X* for which we have a trivial brute-force algorithm that takes O(2^n) time, but we have no clue how to solve *X* in polynomial time. We suspect *X* to be NP-hard. Explain in your own words the procedure of proving that *X* is NP-hard.

2. Worst and best case complexity

a. What are the *worst* case and *best* case time complexity of the algorithm *FindMax* below? Use the O(.) notation and explain your answer!

```
FindMax(array A[n])          // A[n] is (unsorted) array of positive integers
max = -1
for i = 1 to n do
        if A[i] > max
                max = A[i]
        end if
end for
return max
```

b. Could you, for arbitrary inputs, solve this problem faster (in O(.) terms) than with this algorithm? If so, show how; if not, explain your answer.

Have a look at the following algorithm *Find*:

```
Find(array A[n], integer Z)   // A[n] is array of positive integers
for i = 1 to n do
        if A[i] == Z
                return i
        endif
endfor
return -1
```

c. What is the worst- and best-case complexity if A[n] is unsorted?

d. What is the worst- and best-case complexity if A[n] is sorted from low to high? Can you come up with a smarter algorithm?

3. Membership of P and NP – for handing in

For which of the following problems can you prove membership in P?

a. Given a graph G. Does G contain a circuit of length 4?

b. Given a graph G. Is G bipartite?

c. Given n integers. Is there a subset of them whose sum is an even number?

d. Given n integers A[1..n] and another integer K. Is there a subset of the given integers A[1..n] whose sum is K?

e. Given a graph G. Does G contain an Euler circuit?

For which of the following problems can you prove membership in NP?

f. Same as **d.**: Given n integers A[1..n] and another integer K. Is there a subset of the given integers A[1..n] whose sum is K?

g. Given a graph G and an integer K. Does G contain a *path* of length K?

h. Given a graph G and an integer K. Does G contain a *clique* of size K?

i. Given a Boolean formula F. Is there a truth instantiation to its variables that satisfies F?

j. Given a Boolean formula F. Is F a tautology?

4. Old exam question on Complexity and Local Search (6+7+7 = 20 points in total)

In this question we consider the Weighted MAX CNF SAT decision problem, defined as follows: (recall: CNF means 'conjunctive normal form', i.e., the formula consists of a conjunction of clauses; each clause consists of a disjunction of variables or their negations)

Input: A Boolean formula in CNF-form, with m clauses $C_1..C_m$ and n variables $x_1..x_n$; each clause has an associated weight $w_1..w_m$, where weights are natural numbers. In addition, we have a natural number W. If a truth assignment satisfies a clause C_j its value is w_j, otherwise it is 0.
Question: Is there a truth assignment to the variables $x_1..x_n$ such that the total weight of the clauses is larger than W?

a. Show that Weighted MAX CNF SAT is NP-hard by reducing it from (un-weighted) MAX CNF SAT. That is, show that every instance of MAX CNF SAT can be translated in polynomial time into an instance of Weighted MAX CNF SAT. This problem (which is NP-hard) is defined as a decision problem as follows:

Input: A Boolean formula in CNF-form, with m clauses $C_1..C_m$ and n variables $x_1..x_n$. In addition, a number $C < m$.
Question: Is there a truth assignment to the variables $x_1..x_n$ such that at least C clauses are satisfied?

Note: sub-questions b and c require knowledge of simulated annealing as taught in the block on search; you may want to revisit these questions after this block. Particularly c. gives insight in that local search is neither guaranteed to be optimal nor guaranteed to run in polynomial time.

Being faced with this intractability result, we try to solve (possibly sub-optimally) the optimization variant of Weighted MAX CNF SAT (i.e., find a truth assignment that maximizes the total weight of the clauses) by local search, in particular simulated annealing. We do so by starting with a random truth assignment, and in each iteration step changing one variable into a different value.

b. Explain how simulated annealing works on this problem. Explain when candidate solutions are accepted or rejected, what the role of 'temperature' is in simulated annealing, and why we may end up in a local maximum using this search strategy. You do not need to give exact formulas, but you do need to explain what is happening in simulated annealing.

c. Show that, even with a temperature of 0 degrees, we may need an exponential number of improvement steps before stopping in a (possibly even local) maximum in this strategy. Hint: take an example instance where the weights are $\{1, 2, 4, 8, \ldots 2^{m-1}\}$ and assume that the initial truth assignment does not satisfy any clause. Explain what may happen at each iteration