2nd Lt David Crow

ENG/20M

CSCE 686 Advanced Algorithms, Homework 2

**Problem 1 – Talbi 1.1**

Given an undirected graph , a clique of the graph is a subset of where any two vertices in are adjacent. In other words, .

A maximum clique is a clique with the largest cardinality. The problem of finding the maximum clique in a graph is NP-hard.

Consider the following problems:

1. The subset of maximum cardinality such that the set of edges of the subgraph induced by is empty
2. Graph coloring

Find the relationships between the above problems and the maximum clique (MC) problems.

Problem 1 concerns the maximum independent set (MIS) problem. MIS and MC are complementary problems; if is the maximal clique in some graph , then is the maximal independent set in , where is the inverse of . We find the inverse of a graph by simply creating the complete graph over and then deleting all edges (that is, contains exactly those edges that are not present in ). Because MIS and MC are complementary, and . Thus, both problems are NP-hard.

Problem 2, of course, concerns the graph-coloring (GC) problem (let’s assume “graph coloring” here means “vertex coloring”). For some graph , we know that the subgraph induced by the maximal clique is complete; this means that every node must be colored a unique color. We can rephrase this: if the maximal clique in a graph contains nodes, then we require at least unique colors to properly color the graph. In other words, . Furthermore, computing the chromatic number of a graph is an NP-hard problem, so we know that and .

How are these problems identified in the literature?

In the literature, problem 1 is called the maximum independent set problem. It is NP-hard. Problem 2 is simply called graph coloring or, specifically (in its most common form), vertex coloring. It is also NP-hard.

**Problem 2**

Is the set partition problem an “easy NP-hard problem?”

The set partition problem (SP) is often described as the “easiest NP-hard problem.” This is because dynamic programming approaches can optimally solve SP in pseudo-polynomial time. Furthermore, several heuristics can solve SP for many problem instances (either approximately or optimally).

Although we can often solve SP optimally, it’s still NP-hard based on reducibility from known NP-hard problems to SP.

**Problem 3**

Define a ZPP complexity class problem for a military example (beyond P class). Give references as appropriate.

As defined in the Complexity Zoo, ZPP is the class of problems such that a probabilistic Turing Machine :

* runs in expected polynomial time;
* always returns , , or ;
* returns or on at least half of all inputs; and
* is always correct when answering or .

We can now give an example of a ZPP problem the Air Force currently faces.

Air Force pilots in an operational flying squadron must meet certain currency requirements to continue flying. For a given pilot, these requirements include flying hours in said pilot’s designated airframe, overall flying hours, simulator hours, and additional requirements pertaining to ground-based knowledge. Furthermore, pilots perform additional duties – those that are often wholly separate from flying – to ensure overall mission success.

Scheduling this training is a difficult task. Each wing has a limited number of aircraft, of simulator hours, of maintenance personnel, and of servicemembers capable of completing additional duties. Effectively and efficiently scheduling *all* necessary tasks (i.e. flying, aircraft maintenance, and additional duties) to ensure currency for *all* pilots on a given base is an example of a ZPP problem.

A Las Vegas algorithm can generate a random schedule to optimize a given month’s working hours. Such an algorithm could generate a schedule that either does (answers ) or does not (answers ) satisfy all requirements; it’s also possible that no valid schedule exists for the month (although this is rare in practice), so the Las Vegas algorithm will search endlessly (answers ). Such an algorithm could potentially behave in the following way:

SquadronScheduler(requirements)

while True:

let schedule = generateRandomSchedule()

if requirements.metBy(schedule)

return schedule

It’s clear, then, that this scheduling problem is in ZPP.