CS-E400204: Approximation Algorithms **Spring (Fifth Period) 2022**

Final Exam (May 29): Final Exam

Due Date: June 13 *Due Time: 23:59 pm*

Question 1. Max-SAT

a) Show that the following algorithm is a 1/2-approximation for MAX-SAT. Let τ be an arbitrary truth assignment and τ' be its complement, that is, a variable is `true` in τ if and only if it is `false` in τ'. Compute the weight of clauses satisfied by τ and by τ', then output the better assignment.

[2 points]

In the following, let a k-CNF-SAT instance denote a formula in conjunctive normal form, where each clause contains exactly k literals and every variable occurs at most once per clause.

b) Recall the algorithm that just outputs a uniformly random truth assignment. It runs in polynomial time and we proved a lower bound on the expected weight of the satisfied clauses. We then derandomized this algorithm. Consider instead the following algorithm for MAX-3-CNF-SAT: keep guessing truth assignments until you find one that satisfies 7/8 of the clauses. Show that this algorithm runs in expected polynomial time.

[2 points]

c) We introduce here the MAXCUT problem: Given a graph $G = (V, E)$, find a set $S \subseteq V$ such that the number of edges between S and its complement is as large as possible. Solve MAXCUT using MAX-2-CNF-SAT.

[2 points]

Question 2. Scheduling on Parallel Identical machines

Here we are given n jobs to be processed, and there are m identical machines (running in parallel) to which jobs may be assigned. Each job $j = 1, \ldots, n$, must be processed on one of these machines for p_j time units without interruption, and each job is available for processing at time 0. Each machine can process at most one job at a time. The aim is to complete all jobs as soon as possible; that is, if job j completes at time C_j (the schedule starting at time 0), then we wish to minimize $C_{max} = \max_{j=1,\ldots,n} C_j$, which is called the *makespan* or *length* of the schedule.

Consider the following algorithm, called *list scheduling*. Look at the list of all jobs (in some arbitrary order) and greedily assign each job to the machine that, in the partial schedule as constructed so far, finishes soonest.

a) Show that list scheduling has approximation factor 2.

[2 points]

Now we sort the list of jobs in order of non-increasing processing time: this is called the *longest processing time rule*.

b) Show that this version of list scheduling is optimal if each job has length strictly larger than one-third the optimal makespan C^*_{max}.

[3 points]

c) Show that this version of list scheduling has approximation factor $4/3$.

[3 points]

Question 3. Finding Long Paths in Traceable Graphs

First, we introduce a problem known as the Minimum-Degree Spanning Tree. In this problem, given is a connected graph $G = (V, E)$, and the goal is to compute a spanning tree T with minimum $\Delta(T)$. Here $\Delta(T)$ refers to the maximum degree of a node in the tree T. It is known that a natural local search algorithm provides a tree T (in poly time) with $\Delta(T) \leq 2 \cdot \mathrm{OPT} + \lceil \log_2 n \rceil$.

We want to use this result (as a black-box) to solve another problem. A graph G is called traceable if it contains a Hamiltonian path. Given a traceable graph $G = (V, E)$ (with an unknown Hamiltonian path) with $|V| = n$, give a polynomial time algorithm that finds a simple path of length $\Omega\left(\frac{\log n}{\log \log n}\right)$.

[6 points]

Question 4. Randomized Rounding for Set Cover

Consider the following randomized algorithm for Set Cover.

Algorithm 1 RandomizedRounding$(U, \mathcal{S}, c)$

```
Compute optimal solution x for LP-Relaxation of SetCover from the lecture
C ← ∅
for all S ∈ S do
  Add S with probability x_S to C
end for
return C
```

a) Prove that the expected cost of $\mathcal{C}$ is exactly OPT.

[2 points]

b) Let $u \in U$ be an arbitrary element of the ground set. Prove that u is *not* covered by $\mathcal{C}$ with probability at most $1/e$.

[2 points]

Let d be a sufficiently large constant. We now run the algorithm RandomizedRounding exactly $d \cdot \log n$ times. Let $\mathcal{C}'$ be the union of all sets selected this way.

c) Prove that every element $u \in U$ is *not* selected by $\mathcal{C}'$ with probability at most $1/(4n)$, as long as d was chosen large enough.

[2 points]

d) Prove that the cost of the set $\mathcal{C}'$ is greater than $4d \log n \cdot \mathrm{OPT}_{\mathrm{relax}}$ with probability at most $1/4$.

[3 points]

e) Prove that $\mathcal{C}'$ is a *feasible* solution with cost at most $4d \log n \cdot \mathrm{OPT}_{\mathrm{relax}} = O(\log n) \cdot \mathrm{OPT}_{\mathrm{relax}}$ with probability at least $1/2$.

[3 points]

Question 5. Metric Weighted k-Centre

Recall that in the metric k-CENTRE problem, we where given points of a metric and a parameter k, and the goal was to choose k "centres" so that the maximum distance of a point to its closest centre was minimized. Here, we consider a more generalized version with weights on the vertices, which affects the cost of connecting a vertex to a centre:

- **Input**:
  - A complete graph $G = (V, E)$
  - Edge costs $c : E \to \mathbb{Q}_{\geq 0}$ satisfying triangle inequality
  - Vertex weights $w : V \to \mathbb{Q}_{\geq 0}$
  - A parameter $k \in \mathbb{N}_{>0}$
- **Output**:
  - A set of *centres* S of size k such that $\max_{v \in V}\{w(v) \cdot d(v, S)\}$ is minimized. Here $d(v, S) \triangleq \min_{u \in S}\{d(u, v)\}$.

Provide a 2-approximation algorithm for the metric WEIGHTED k-CENTRE problem (with proof of the factor and correctness).

[7 points]

Question 6. Alternative Local Search for k-Median

Consider the following local search algorithm instead of what we say in the lecture:

Algorithm 2 Local Search for k-MEDIAN

$S^0 \leftarrow$ any subset of F of size k
for $j \leftarrow 1$ to T **do**
  let (i, i'), $i \in S, i' \in F - S$, be the cheapest swap
  $S^j \leftarrow S^{j-1} - i + i'$
end for
return S^T

where $T \triangleq k \cdot \lceil \ln(n \cdot \Delta) \rceil$ (Δ or the aspect ratio is defined as in the lecture), and by the cheapest swap we mean one that minimizes the cost at this iteration the most. For simplicity of the analysis, assume there are no co-located points.

Prove that this algorithm is a 5-approximation for k-MEDIAN (i.e., without the extra ε). In other words, prove $cost(S^T) \leq 5 \cdot \mathrm{OPT}$.

Suggestion: Since all the costs are initially rational, you can assume that by scaling, we have cleared all the denominators of the costs. That is, $\min_{i,j}\{c(i, i)\} \geq 1$ and therefore, $\Delta \leq \max_{i,j}\{c(i, j)\}$.

[5 points]