CS-E3190 Principles of Algorithmic Techniques

09. Tradeoff – Tutorial Exercise

1. **Karger's algorithm.** Let us complete the proof of the lecture slides. Prove that the contraction algorithm outputs a minimum cut with probability $\geq 2/n(n-1)$.

   Solution. Let $C \subseteq E$ denote the edges of a specific minimum cut of size k. The contraction algorithm from the lecture slides returns C if none of the randomly selected edges belong to the subset C. Observe that the first edge contraction avoids "hitting" set C with probability $1 - k/|E|$. Observe also that the minimum degree of G is k (otherwise the minimum degree vertex w would induce a smaller cut where one partitions would contain w and the other partition $G \setminus w$), and hence

   $$|E| = 1/2 \cdot \sum_{v \in V} \deg(v) \geq 1/2 \cdot \sum_{v \in V} k = nk/2.$$

   Thus, the probability that the contraction algorithm picks an edge from C is

   $$\frac{k}{|E|} \leq \frac{k}{nk/2} = \frac{2}{n}.$$

   The probability P_n that the contraction algorithm on an n-vertex graph avoids C satisfies the recurrence $P_n \geq (1 - 2/n)P_{n-1}$, with $P_2 = 1$ (since we terminate when having 2 nodes left). The previous recurrence can be expanded to

   $$\begin{aligned} P_n &\geq \prod_{i=0}^{n-3} \left(1 - \frac{2}{n-i}\right) \\ &= \prod_{i=0}^{n-3} \frac{n-i-2}{n-i} = \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdots \frac{3}{5} \cdot \frac{2}{4} \cdot \frac{1}{3} = \frac{2 \cdot (n-2)!}{n!} = \frac{2}{n(n-1)}, \end{aligned}$$

   completing the proof.

2. **FPTAS.** An approximation scheme is said to be a polynomial time approximation scheme, or PTAS, if for each fixed $\epsilon > 0$ (error parameter), its running time is bounded by a polynomial in the size of instance n. This, however, means that it could be exponential with respect to $1/\epsilon$, in which case getting closer to the optimal solution is incredibly difficult.

   Thus the fully polynomial time approximation scheme, or FPTAS, is an approximation scheme for which the algorithm is bounded polynomially in both the size of the instance n and by $1/\epsilon$.

   Consider the $0/1$ Knapsack problem of the Dynamic Programming week. Let $P \in \mathbb{N}$ be the value of the most valuable item. Assume that we know how to solve the problem in $O(n^2 P)$ time using a dynamic programming algorithm (the specifics of the algorithm are not important here). At first, our runtime seems polynomial in n with no dependency on the error term $1/\epsilon$. However, since P is unbounded, our solution as such is not a FPTAS.

   Give a FPTAS for the $0/1$ Knapsack problem.

   Hint: Scale the values of all items by some quantity (that depends on n, ϵ, and P), and then apply the dynamic programming algorithm.

   Solution. Consider the $0/1$ Knapsack problem: given a knapsack with capacity $C \in \mathbb{N}$, and a set of items $I = \{a_1, a_2, \ldots, a_n\}$ such that item i has value $v_i \in \mathbb{N}$ and weight

$w_i \in \mathbb{N}$, select a subset $S \subseteq I$ of the items to pack that maximizes total value without exceeding the weight capacity C.

From the dynamic programming algorithm, we see that if the values of the items were all small numbers, i.e. polynomial in n, then we would have a poly(n) time algorithm. We will use this knowledge to obtain an FPTAS for the $1/0$ Knapsack problem. Observe the we can scale the values down enough such that the values of all the items are polynomial in n, use dynamic programming on the new instance, and return the resulting subset. By scaling with respect to the desired ϵ, we will be able to get a solution that is at least $(1-\epsilon) \cdot$ OPT in polynomial time with respect to both n and $1/\epsilon$, giving a FPTAS. Consider the following algorithm.

(a) Given $\epsilon > 0$, let $K = \epsilon P/n$

(b) For each item a_i, define $v'(a_i) = \lfloor v(a_i)/K \rfloor$

(c) With these as values of the items, using the dynamic programming algorithm, find the most valuable set S'

(d) Output S'

Correctness: Let O be the optimal set returning the maximum value possible and let OPT be said value. Because we scale down the value of each item by K and then round down, any item a will have value $K \cdot v'(a) \leq v(a)$ with the difference at most K. Thus the most that the value of the optimal set O can decrease is at most nK for K per possible member of the set, or

$$P(O) - K \cdot P'(O) \leq nK$$

After the dynamic programming step, we get a set that is optimal for the scaled instance and therefore must be at least as good as choosing the set O with the scaled values. From that, we can see that

$$\begin{aligned} P(S') &\geq K \cdot P'(O) \\ &\geq P(O) - nK = \text{OPT} - \epsilon P \\ &\geq (1-\epsilon) \cdot \text{OPT} \end{aligned}$$

since OPT $\geq P$.

Runtime: By the analysis of the dynamic programming algorithm, the running time of the algorithm is $O(n^2 \lfloor P/K \rfloor) = O(n^2 \lfloor n/\epsilon \rfloor)$, which is polynomial in both n and $1/\epsilon$.