CS602

Assignment-3

Saksham Rathi
22B1003

CS 602 - Applied Algorithms: Assignment 3

Total Marks - 60

Instructions. Please try to be brief, clear, and technically precise. Use pseudo-codes to describe the algorithms. To solve the problems, one may assume that the instances are in general position, unless stated otherwise. Novelty in the answer carries marks.

- **Question 1:** [20 Marks]*Ski rental problem*

  At a ski resort, renting a ski costs 1 rupee per day, while buying skis costs B rupees. A skier arrives at the ski resort for a ski vacation and has to decide whether to rent or buy skis. However, an unknown factor is the number of remaining skiing days left before the snow melts. A randomized $\frac{e}{e-1}$ *competitive algorithm exists.*

  1. Construct a simple deterministic 2- competitive algorithm for the problem.
  2. Formulate the problem as a linear program.
     Hint: define an indicator variable which is set to 1 if the skier buys the skis, and for each day $i, i \in [1, k]$ *(which is unknown in advance), define another indicator variable which is set to 1 if the skier decides to rent skis on day* j*. The constraints guarantee that on each day, we either rent skis or buy them.*
  3. Formulate the dual program for this.
  4. Construct a 2-competitive algorithm using primal-dual method.
     Hint: Whenever we have a new ski day, the primal program is updated by adding a new constraint. The dual pržogram is updated by adding a new dual variable. The online requirement is that previous decisions cannot be undone. In other words, the primal variables are monotonically non-decreasing over time.

- **Question 2:** [20 Marks] Suppose we have one machine and jobs released over time: job i is released at time r_i, has size w_i, benefit b_i, and deadline

d_i. Jobs are allowed to be preempted (i.e., interrupted and later resumed) and/or partially executed (as long as it is before the deadline). Denote by $p_i \leq w_i$ the total time job i was processed before its deadline. Our goal is to maximize $\sum_i \frac{p_i}{w_i} b_i$. Assume that r_i, w_i, d_i (but not b_i) are integers and the algorithm as well as OPT are allowed to preempt jobs only at integer times. Design a 2-competitive algorithm for the problem.

- **Question 3:** [20 Marks] *MARKING Problem* : Consider a randomized paging algorithm that processes a request sequence in phases. At the beginning of each phase, all pages in the memory system are unmarked. Whenever a page is requested, it is marked. On a fault, a page is chosen uniformly at random from among the unmarked pages in fast memory, and this pages is evicted. A phase ends when all pages in fast memory are marked and a page fault occurs. Then, all marks are erased and a new phase is started.

  The competitive ratio of a randomized online algorithm A is defined with respect to an adversary. The adversary generates a request sequence σ and it also has to serve σ. When constructing σ, the adversary always knows the description of A. We hereby define the notion of oblivious adversary.

  Oblivious Adversary: The oblivious adversary has to generate a complete request sequence in advance, before any requests are served by the online algorithm. The adversary is charged the cost of the optimum online algorithm for that sequence.

  Prove that the MARKING algorithm is $2H_k$-competitive against any oblivious adversary, where $H_k = \sum_{i=1}^{k} \frac{1}{i}$ is the k-th Harmonic number.

Question 1 (1) A 2-Competitive algorithm ensures that the cost incurred by the algorithm is at most twice the cost of an optimal offline solution.

We can design a deterministic algorithm as follows:

- The skier rents skis for B days (where B is the cost of buying the skis).
- On the B^{th} day, if the skier still needs skis, they buy them.
- After $(B)^{th}$ day, the cost is 0 for skier, since it has already bought the ski.

Proof that it is 2-competitive:

- If the skier skies for fewer than B days, the cost is at most B which is already optimal
- If the skier skies for $(B+x)$ days $x \geq 0$

$\Rightarrow$ optimal $= B$

$\Rightarrow$ from deterministic algo $= (B-1) + B = 2B-1$

ratio $= \frac{2B-1}{B} \leq 2$

Hence, this is a deterministic 2-competitive algorithm

(2) x is a binary variable that is 1 if the skier buys the skis

y_i is a binary variable that is 1 if the skier rents skis on day i, where $i \in [1, k)$ (k = unknown)

Cost $C = Bx + \sum_{i=1}^{k} y_i$

Objective: minimize C

Constraints: On each day j:

$$y_i + x \geq 1 \quad \text{for } j = 1 \dots k$$

Either the skier rents skis or has already bought them.

x and y_i are binary

(3)

Primal

$$Ax \geq b$$

$$x \geq 0$$

$$\min(w^T x)$$

$$A = \begin{pmatrix} 1 & 1 & 0 & \dots & 0 \\ 1 & 0 & 1 & \dots & 0 \\ \vdots & & & & \\ 1 & 0 & 0 & \dots & 1 \end{pmatrix}_{k \times (k+1)}$$

$$x = \begin{pmatrix} x \\ y_1 \\ \vdots \\ y_k \end{pmatrix} \quad (k+1) \times 1$$

$$b = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \quad k \times 1$$

$$W = \begin{pmatrix} B \\ 1 \\ 1 \\ \vdots \end{pmatrix}_{(k+1) \times 1}$$

Dual

$$\max(y^T b)$$

$$A^T y \leq W$$

$$y \geq 0$$

$$y = \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_k \end{pmatrix}$$

$$\max(\lambda_1 + \dots + \lambda_k)$$

$$\max\left(\sum_{i=1}^{k} \lambda_i\right)$$

Constraints: $\lambda_i \geq 0 \quad \forall i = 1 \dots k$

$$\lambda_1 + \lambda_2 + \dots + \lambda_k \leq B$$

$$\lambda_j \leq 1 \quad \forall j = 1 \dots k$$

Dual problem

(4) We will start with an empty set of constraints in the primal. For each new day j, we will update the primal program by adding the constraint $y_i + x \geq 1$ and the corresponding

dual variable x_j.

We will increase x_j in the dual as long as the constraints $\sum_{j=1}^{k} x_j \leq B$ and $x_j \leq 1$ are satisfied

Steps of the Algorithm :

On each new day, set $y_j = 1$ to rent skis and increase x_j in the dual.

Continue until $\sum_{j=1}^{k} x_j = B$ at which point we will set $x = 1$ (buy the skis)

We are also ensuring that the previous decisions are unchanged, satisfying the online requirement.

* This algorithm aligns with the deterministic algorithm and is therefore 2-competitive.

Question-2

Job i released at time r_i, has size w_i, benefit b_i and deadline d_i

Here is a simple greedy 2-competitive algorithm ⇒

For each available job i at time t (i.e. $r_i \le t < d_i$)

→ Compute the benefit density $\frac{b_i}{w_i}$

Sort jobs by highest benefit density to lowest.

If there is a tie, prioritize the job with the nearest deadline d_i

At each integer time t, choose the highest priority job i according to the priority defined above and process it for unit amount of time.

Preempt the job if a new job j with higher priority arrives at time t+1 (i.e. $r_j = t+1$)

If p_i reaches w_i (the job is fully processed) or the time reaches d_i (i.e. the deadline), remove the job from the queue

At any time t ⇒ our algorithm will pick the job with highest $\frac{b_i}{w_i}$

At some time t, OPT is running something and our algo isn't. Let's say this has value A. We are running best possible in that time. Let's say this has value B. In the worst case scenario, OPT will run the best case later on completely. But in that frame, what OPT is running cannot be better than what we are running.

Thus, we get B and OPT gets A+B, $A \le B$

This is true for every time frame, summing over all time instants, we get that our algo is 2- competitive.

Question 3

We need to prove that the randomized marking algorithm (RMA) has competitive ratio $2H_k$ against any oblivious adversary. ($H_k = k^{th}$ harmonic number)

We divide the sequence of requests (σ) into phases. The i^{th} phase ends immediately before the $(k+1)^{st}$ distinct page is requested in the phase. If we denote the set of pages requested in phase i by P_i, then at the end of a phase i, the contents of RMA's cache is exactly P_i.

m_i = number of new requests in phase i. (not present in phase $i-1$)

Let us suppose that just before the j^{th} old page is requested, there have been l new pages requested so far in the phase. By induction, we can show that at this time there are exactly l pages in P_{i-1} that are not in the cache. These are distributed randomly (uniform) among the $k-(j+l+1)$ unmarked pages in P_{i-1}. Probability that it is not in the cache is exactly $\frac{l}{k-(j+l-1)}$ $\quad l \leq m_i \rightarrow$ by definition

$$P(\text{fault on the request to the } j^{th} \text{ old page}) \leq \frac{m_i}{k-(j+m_i-1)}$$

$$\mathbb{E}[cost_{RMA}(\sigma)] \leq m_i + \sum_{1 \leq j \leq k-m_i} \frac{m_i}{k-j-m_i+1} \leq m_i H_k$$

Summing up over all the phases, $\mathbb{E}[cost_{RMA}(\sigma)] \leq H_k \sum_i m_i$

We need to prove a lower bound for the optimal cost.

Claim: $\quad cost_{OPT}(\sigma) \geq \sum_i \frac{m_i}{2}$

Consider the $(i-1)^{st}$ and i^{th} phases. The number of distinct pages requested in both phases is $k+m_i$. OPT has only k pages in the cache at the beginning of the $(i-1)^{st}$ phase, it must incur at least m_i faults during the two phases. We can apply the same argument to every pair of adjacent phases, we have that $cost_{OPT}(\sigma) \geq \sum_i m_{2i}$ and $cost_{OPT}(\sigma) \geq \sum_i m_{2i+1}$. Therefore, OPT has cost at least the average of these i.e. $cost_{OPT}(\sigma) \geq \sum_i \frac{m_i}{2}$. Thus $\mathbb{E}[cost_{RMA}(\sigma)] \leq 2H_k \, cost_{OPT}(\sigma) \Rightarrow$ Marking algorithm = $2H_k$ competitive