The Parking Permit Problem

Adam Meyerson [*]

November 3, 2004

Abstract

We consider online problems where purchases have *time durations* which expire regardless of whether the purchase is used or not. The *Parking Permit Problem* is the natural analog of the well-studied ski-rental problem in this model, and we provide matching upper and lower bounds on the competitive ratio for this problem.

1 Introduction

We introduce the online problem of purchasing items which have a duration in time. Online algorithms deal with situations in which the future is unknown. However, in the canonical examples of online algorithms, the goal is to assemble some structure based on purchases which remain forever. For example, in the online steiner tree problem, we must construct a tree to connect various terminals which are made known to the algorithm one at a time. But each tree edge, once purchased, remains forever [3]. One of the first examples given in an online algorithms class is the ski-rental problem [4], where the goal is to obtain skis for a sequence of ski trips, without prior knowledge of the number of trips which will be made. The skis effectively have a duration in terms of "number of trips;" in the simplest example we have a rental which lasts for a single trip and a purchase which has infinite duration, but we can add skis which "last" for some number of trips without modifying the essential structure of the algorithm. In this paper, we are interested in purchases which have a duration in *time*. We consider an analog of the ski rental problem in which we purchase parking permits, each of which expires after some fixed number of days *whether we use it or not*.

Giving time durations to our purchases has many natural applications. Many objects have implied maintenance costs. For example, a web server has an initial purchase price but also has a cost for power, internet connection, and occasional service. If we were to consider the online problem faced by a company setting up and maintaining these servers (with the goal of providing some level of service to a customer base which changes in an online fashion), it becomes important to include these maintenance costs in our pricing model. Placement of web servers has been studied in a number of theoretical results [6, 10, 8] but these have tended to assume a "facility location" type model where, once purchased, a server remains indefinitely for no extra charge. As another example, the buy-at-bulk [11, 2, 5] (and associated rent-or-buy [7, 9]) problems involve purchasing network connections so as to allow traffic between various given terminals. As currently stated, the rent-or-buy problem is similar to ski rental, in that renting a connection allows only *one* use whereas buying allows infinite uses. Buy-at-bulk allows the purchase of many different types of connections, each good for a different number of *uses*. However, the real cost of a connection will depend on the *frequency* of use, not just the number of uses over some indefinite time. From a logistical standpoint, it is much cheaper

[*]University of California, Los Angeles. Email: awm@cs.ucla.edu

for a company to transport occasional large shipments over a link than to transport frequent small shipments totaling the same aggregate amount of product.

As a first problem in dealing with time durations, we consider a natural analog of the ski rental problem, which we call the *Parking Permit Problem*. Suppose that on certain days we drive to work. The choice of driving (as opposed to walking) is made based on many factors (for example weather) which are not known in advance. On any driving day, we must obtain a valid parking permit. These permits are available for various numbers of days (for example: a daily permit, weekly permit, monthly permit, and yearly permit). Permits have different costs (with longer duration permits tending to cost less per day). If we were to drive every day, it would be most efficient to purchase the longest-duration permit, but since we drive only occasionally, determining which permit to buy requires predicting an unknown future.

If the schedule of driving days were given to us in advance, we could solve the problem by using dynamic programming; simple greedy approaches do not attain optimum solutions. In the more interesting online version, we consider various competitive algorithms. The ski rental problem has a 2-competitive result even in the case where many skis (each lasting for a specified number of trips) are available. However, adding in time durations makes the problem substantially more difficult. We show that for a deterministic algorithm (a la the ski rental algorithm), the best possible competitive ratio is linear in the number of permits. Similarly, a randomized algorithm without any knowledge of the schedule's history (one which simply chooses a permit at random according to some distribution every time a permit is needed) has a best possible competitive ratio linear in the number of permits. However, a technique based on ideas for online set cover [1] yields a randomized algorithm with competitive ratio logarithmic in the number of permits. We show that this is best-possible.

This paper is a first step towards analyzing online algorithms with time durations. Almost any online problem may be reasonably considered from this perspective, giving many opportunities for future work. The techniques and lower bounds described in this paper may eventually yield similar results for a wide spectrum of problems.

2 Problem Definition and Equivalence

In the parking permit problem, we are given K different types of permits which we can purchase. Permit k has cost C_k dollars and duration D_k days. We are given a schedule on which certain days are marked as driving days, and asked to select a set of permits such that the cost is minimized and every driving day is covered. This problem can be solved in time $\Theta(Kn)$ where n is the number of days in the schedule (via dynamic programming); however, we are interested in the *online* version of the problem where the schedule is revealed one day at a time. Our goal is to minimize the competitive ratio $\alpha(K)$ of the cost paid by our algorithm to cover all driving days versus the cost paid by the optimum (offline) algorithm which sees the schedule in advance. We will obtain a ratio which depends only upon the number of permit types (K); in principle the competitive ratio might also depend on the specific values of C_k and D_k given, but such dependence is undesirable.

We will first define several restrictions on the problem, and show that within a $\Theta(1)$ factor, the competitive ratio for each of these restricted versions is the same as the competitive ratio for the original problem.

Theorem 2.1. *Scaling: For each permit type* $1 < k \leq K$*, we can assume that* $C_k \geq 2C_{k-1}$ *and* $D_k \geq D_{k-1}$*. This assumption causes us to lose at most a factor of two in our competitive ratio.*

Proof. We first observe that the optimum algorithm (or any reasonable algorithm) will never use a permit of type k if there exists a permit of type j with $C_j \leq C_k$ and $D_j \geq D_k$. After eliminating such permits, we order the permit types by cost (so $C_k \geq C_{k-1}$ and $D_k \geq D_{k-1}$).

Suppose we have an $\alpha(K)$ competitive algorithm when permits increaes in cost by factors of two. We start with type K and work downwards, deleting every permit type such that $C_k > \frac{1}{2}C_{k+1}$. After these deletions, we have $K' \leq K$ permit types. Consider the optimum solution to the original problem. We replace any deleted permit type with the cheapest permit type of higher cost. Since deletion implies that some remaining permit has at most twice the cost, this gives a new solution which has cost at most twice the original optimum ($OPT' \leq 2OPT$). Our algorithm produces a solution with cost $ALG \leq \alpha(K')OPT' \leq 2\alpha(K')OPT$. Assuming that the competitive ratio $\alpha(K)$ is non-decreasing with the number of permits (this is true of any reasonable algorithm since we can always add additional "useless" permits), we will have $ALG \leq 2\alpha(K)OPT$. □

Theorem 2.2. *Interval Model: We may assume a version of the problem in which each permit is only available over specific time spans. For example, consider month-long permits which span the days of a specific month. Each permit of type k will have $R_k = \frac{D_k}{D_{k-1}}$ permits of type $k-1$ embedded within it. At any given time, there is exactly one possible permit of type k (for every k) which could cover this time. Within a $\Theta(1)$ factor, any online algorithm for this version of the problem is competitive for the original version, and vice-versa.*

Proof. Consider the optimum solution for the original problem. Suppose we divide the time line into intervals of length D_k, starting from the beginning. Any type k permit intersects at most two of these intervals of length D_k. So we can produce a new solution to the interval version of the problem which purchases those two permits of type k instead. Thus the optimum solution for the interval problem is at most a factor of two larger than the optimum solution for the original problem.

Given an algorithm for the interval version, this guarantees $ALG \leq \alpha OPT_{interval} \leq 2\alpha OPT_{original}$. On the other hand, if we have an algorithm for the original version of the problem we can modify it to purchase, instead of a permit of type k, the two permits of type k which cover that permit under the interval model. This at most doubles the cost of the algorithm, so we will have $ALG_{doubled} \leq 2ALG_{original} \leq 2\alpha OPT_{original} \leq 2\alpha OPT_{interval}$, noticing that the interval optimum cannot be less than the original optimum. □

3 A Deterministic Approach

At first glance, the parking permit problem seems very similar to the ski rental problem [4]. For ski rental, the online algorithm rents skis until the total payment for rentals equals the cost to purchase, at which point it purchases skis. Even if we allow multiple types of ski (each with different cost-to-buy and duration in terms of number-of-trips), this algorithm still gives a $\Theta(1)$ competitive ratio. The deterministic approach to the parking permit problem involves adapting this algorithm.

3.1 The Online Algorithm

We consider the interval version of the problem described in theorem 2.2. Each time we need to drive, we will purchase permits of type 1, until there exists some type 2 interval where the optimum solution (assuming the sequence of requests so far is the entire schedule) would purchase a permit of type 2. For any interval of type k, as soon as the optimum offline solution (using only the schedule seen so far) would purchase this permit, we purchase it.

Theorem 3.1. *The deterministic algorithm described gives an $O(K)$-competitive result for the parking permit problem.*

Proof. We will prove by induction that over any interval of type k, the algorithm pays at most k times what the optimum would pay if this interval was the entire input. For $k = 1$, if we do not drive, then we pay zero, as does optimum. If we do drive, the optimum must pay at least C_1 and we pay C_1.

For a larger value of k, if the optimum pays less than C_k, then the optimum doesn't buy the permit which covers this interval. It follows that the optimum pays for each sub-interval separately. Our algorithm pays at most $k - 1$ times what the optimum must pay on each sub-interval of type $k - 1$, so we pay at most $k - 1$ times what optimum would pay if this interval of type k was the whole input. Otherwise, optimum pays C_k. Consider restricting our set of driving days (by deleting days starting from the end of the interval) such that the optimum would not pay C_k. Over this set of days, our algorithm pays at most $(k - 1)C_k$ by the argument above. Now we consider what happens on the first "deleted" day. Our algorithm notices that the optimum would now buy a permit of type k, so the algorithm does as well. Thus the algorithm pays at most kC_k which is at most k times what the optimum pays. □

3.2 Deterministic Lower Bound

Is it possible to do better than this? After all, the ski rental problem did not have a competitive ratio which depended upon the number of ski types. Two questions remain to be answered: was the analysis for the deterministic algorithm tight, and could a better deterministic algorithm exist? We prove that no deterministic algorithm can obtain competitive ratio better than $\Omega(K)$ (linear in the number of permits).

Theorem 3.2. *No deterministic algorithm whose competitive ratio is dependent solely on the number of permits (K) can obtain better than $\Omega(K)$ competitive ratio.*

Proof. Suppose we have K permits with costs $C_k = 2^k$ and durations $D_0 = 1$ and $D_k = 2KD_{k-1} = (2K)^k$. The adversary drives if and only if the algorithm has no valid permit. We assume the interval model, where permits apply to specific time periods.

Consider all the intervals of type k where the adversary gives a nonzero number of driving days. There are three possibilities for for the algorithm during this interval:

- The algorithm eventually purchases a permit of type k for this interval. Suppose this happens n_k times.
- The algorithm eventually purchases a permit of type $j > k$ which covers this interval. This happens at most $\Sigma_{j>k} n_j$ times.
- The algorithm never purchases a permit of type k or higher which covers this interval. Suppose this happens r_k times.

We let ALG represent the cost paid by our algorithm and OPT represent the cost paid by the optimum. We immediately have $ALG = \Sigma_{k=1}^{K} n_k C_k$. On the other hand, we observe that we could maintain a valid permit on all drive days by purchasing a permit of type k for every interval of type k where we drive a nonzero number of times (we could do this for any choice of k and it would work). So we conclude that for any k:

$$OPT \leq C_k(r_k + \Sigma_{j \geq k} n_j)$$

Consider any interval of length D_k starting with a driving day. We can prove inductively that the algorithm must pay at least C_k on this interval. This is clear for $k = 1$. In general, if the algorithm buys a permit of type k it spends C_k, and if not we can divide the interval into $2K$ intervals of length D_{k-1} each starting with a driving day. Each of these intervals recursively costs C_{k-1} so the total paid is at least KC_k.

We now consider permits of type k. Consider an interval of length D_k for which no permit of type k or higher was ever bought. We must pay at least $2KC_{k-1} = KC_k$. We have $ALG \geq Kr_kC_k$. By summing the bound on OPT over permit types k, we have:

$$kOPT \leq \Sigma_k(C_k(r_k+\Sigma_{j\geq k}n_k)) = \Sigma_k(n_k\Sigma_{j=1}^{k}C_k+C_kr_k) < \Sigma_k(2n_kC_k+r_kC_k) = 2ALG+\Sigma_kr_kC_k \leq 3ALG$$

This implies that the algorithm is at least $\frac{K}{3}$ times more expensive than the optimum, for the desired $\Omega(K)$ lower bound. □

Combining the algorithm with the negative result, we conclude that the best deterministic online algorithm for the problem has competitive ratio $\Theta(K)$; within a constant factor of the ratio attained by the algorithm given.

4 A Randomized, Memory-Less Approach

We naturally turn to randomized algorithms next. For the ski rental problem, we can give an algorithm which purchases skis with probability equal to the ratio of rental cost to purchase cost. These determinations are made each time we ski, independently until skis are purchased. This memory-less randomized algorithm gives an expected $\Theta(1)$ competitive ratio for ski rental. Even if we consider multiple types of skis, a similar approach will give $\Theta(1)$ competitive ratio. We attempt to apply the same techniques to the parking permit problem.

4.1 The Online Algorithm

Every time we need to drive and don't have an active permit, we will purchase a permit of type k with probability $\frac{C_1}{C_k}$, independently. Notice that we always purchase a permit of type 1 so this will never result in driving without a permit.

Theorem 4.1. *The expected cost paid by the algorithm is at most $O(K)$ times the optimum cost.*

Proof. Each time we drive, we pay an expected $\Sigma_kC_k\frac{C_1}{C_k} = KC_1$. Now consider an interval for which the optimum purchased a permit of type k, paying C_k. Let x be the number of times we pay for permits during this interval. The expected total payment for the algorithm is:

$$ALG = \Sigma_{i=1}^{\infty}Pr[x \geq i]KC_1 = KC_1E[x]$$

Since the expected value of x is at most the waiting time for us to purchase a permit of type k, we have $E[x] \leq \frac{C_k}{C_1}$ which gives the required bound. □

4.2 Memoryless Lower Bound

Theorem 4.2. *No randomized algorithm which simply selects a permit according to some distribution each time a permit must be purchased (i.e. without depending upon past requests) can attain better than $\Omega(K)$ competitive ratio.*

Proof. Suppose that any time we need a permit, we buy a permit of type k with probability P_k. We assume that our permits are such that $C_k << D_k/D_{k-1}$ and $C_k = 2^k$. Suppose that we were to drive once every D_{k-1} days. Each time we drive, there is some probability that we purchase a a permit of type k or higher. The expected waiting time for this to occur is $\frac{1}{\Sigma_{j \geq k} P_j}$. Let μ be the expected cost paid each time we need to purchase a permit. We can write:

$$\mu = \Sigma_{k=1}^{K} P_k C_k$$

If we have an interval of length D_k where we drive once every D_{k-1} days, then our expected total cost paid is $\mu \frac{1}{\Sigma_{j \geq k} P_j}$. If our competitive ratio is α, then this must be at most αC_k. So we have:

$$\mu \leq \alpha C_k \Sigma_{j \geq k} P_j$$

If we sum both sides over all k, on the right hand side, we notice that P_j appears once with coefficient C_k for each $k \leq j$; since each cost is twice the last, these will sum to at most $2C_k$. Thus we have:

$$K\mu \leq \alpha \Sigma_k P_k (2C_k) = 2\alpha\mu$$

Canceling, we have $\alpha \geq \frac{K}{2}$ which proves the $\Omega(K)$ bound.

□

5 A Randomized Approach

By observing the lower bounds of the previous sections it becomes natural to ask whether any online algorithm can beat the $\Omega(K)$ competitive bound. We show that a randomized algorithm which makes use of the past can, in fact, obtain $\Theta(\log K)$ competitive ratio. Before proving this, we first show an equivalence between an online deterministic fractional solution and a randomized algorithm.

A fractional algorithm for the parking permit problem is allowed to purchase fractional permits. For example, we might buy $\frac{1}{2}$ a permit of type i. On each driving day, the sum of total fractional permits must be at least one. We can think of such an algorithm as solving the linear relaxation of the integer program version of the parking permit problem. In an offline setting, the linear program will have an integral optimum! However, an online algorithm for the problem may perform better if allowed to maintain fractional solutions during its run.

Theorem 5.1. *There exists a randomized algorithm for the parking permit problem with competitive ratio $\Theta(\alpha(K))$ if and only if there exists a deterministic fractional algorithm with competitive ratio $\Theta(\alpha(K))$.*

Proof. We will let $f_a^b(t)$ represent the fractional value of the permit starting at time a and ending at time b, which the online algorithm owns at time t. The fractional solution guarantees that these values are nonnegative for every permit, and are nondecreasing with time. Since there's no reason to purchase a permit before its start time, we may assume without loss of generality that $f_a^b(t) = 0$ for $t < a$ and that $f_a^b(t) = f_a^b(b)$ for $t > b$. In order to maintain feasibility, the fractional solution requires that for any t which corresponds to a driving day, we have $\Sigma_{a \leq t}\Sigma_{b \geq t} f_a^b(t) \geq 1$. If the algorithm terminates at time T, the cost of the algorithm is:

$$C = \Sigma_{i=1}^{K}\Sigma_a f_a^{a+D_i}(T)C_i$$

First, let's suppose that we have a randomized online algorithm for the problem. Let $P_a^b(t)$ represent the probability that our algorithm has purchased the permit which starts at time a and ends at time b, by time t. Of course, these probabilities will be nonzero and nondecreasing with time. For any time t which corresponds to a driving day, the randomized algorithm must guarantee a valid permit. It follows that $\Sigma_{a \leq t}\Sigma_{b \geq t} P_a^b(t) \geq 1$. The expected cost of the online algorithm may be decomposed into the sum of expected cost for each permit, which is just:

$$C = \Sigma_{i=1}^{K}\Sigma_a P_a^{a+D_i}(T)C_i$$

From the above observations, we may design a fractional algorithm by setting $f_a^b(t) = P_a^b(t)$. This algorithm has cost equal to the expected cost of the randomized algorithm, and maintains all the required feasibility properties deterministically.

Now suppose we have a fractional online algorithm for the problem. We will assume the "interval" version of the problem where permits may be purchased only at specific times and no two permits of the same duration overlap. This assumption cannot lose more than a factor of two in the competitive ratio by theorem 2.2 and is therefore essentially without loss of generality. We design a randomized algorithm as follows. At the beginning of the randomized algorithm we select τ uniformly between zero and one. We now run the fractional procedure. At any given time t, we have K potential permits which could be valid at this time. Let $F_i(t)$ represent the fractional value of the permit of type i which covers t, at time t; this is just $F_i(t) = f_a^{a+D_i}(t)$ with appropriate choice of a. Our randomized algorithm purchases this permit of type i at time t if:

$$\Sigma_{j=i+1}^{j=K} F_j(t) < \tau \leq \Sigma_{j=i}^{j=K} F_j(t)$$

Since τ is between zero and one, and the sum of the $F_j(t)$ must be at least one for any driving day, there will always be some valid permit any time we drive. Now we must bound the expected cost. Consider the permit starting at time a and ending at time $a + D_i$. We could only purchase this permit at times $a \leq t \leq a + D_i$. Since the fractional values of permits are nondecreasing, in order to purchase the permit at some time t we must have:

$$\Sigma_{j=i+1}^{j=K} F_j(a) \leq \Sigma_{j=i+1}^{j=K} F_j(t) < \tau$$

$$\tau \leq \Sigma_{j=i}^{j=K} F_j(t) \leq \Sigma_{j=i}^{j=K} F_j(a + D_i)$$

So the probability of purchasing the permit is at most $\Sigma_{j=i}^{j=K} F_j(a+D_i) - \Sigma_{j=i+1}^{j=K} F_j(a) = F_i(a+D_i) + \Sigma_{j=i+1}^{j=K}(F_j(a + D_i) - F_j(a))$. The cost of the permit is C_i. We sum this to determine the total expected cost of permits of type i. The terms in the summations for $j > i$ will telescope, observing that each permit of type i begins when the last one terminated, yielding an expected cost of:

$$\Sigma_a f_a^{a+D_i}(T)C_i + \Sigma_{j=i+1}^{K}\Sigma_a f_a^{a+D_j}(T)C_i$$

If we sum this over all permit types i, we can regroup the terms to get an expected cost of:

$$\Sigma_{i=1}^{K} f_a^{a+D_i}(T)\Sigma_{j=1}^{i} C_i$$

Since we can assume by theorem 2.1 that $C_{i+1} \geq 2C_i$, the sum of costs simplifies to give an overall expected cost of at most twice the fractional cost. This implies that any fractional algorithm gives rise to a randomized algorithm with expected cost within $\Theta(1)$ of the fractional cost, completing the proof. □

5.1 The Online Algorithm

We will now present an online fractional algorithm. This can be transformed into a randomized (integral) algorithm using the techniques described in the proof of theorem 5.1. We will consider the interval version of the problem (losing only a constant factor in competitivity, because of theorem 2.2).

We initially set all permits to fraction zero. If at any time we need to drive, and our total fractional permits for this time sum to less than one, we will perform a sequence of operations until the fraction exceeds one. An operation proceeds as follows:

- For each $1 \leq i \leq K$, multiply the fraction by which the currently valid type i permit is purchased by a factor of $1 + \frac{1}{C_i}$.
- For each $1 \leq i \leq K$, increase the fraction by which the currently valid type i permit is purchased by adding $\frac{1}{KC_i}$.

Notice that some finite number of operations will always increase the fractional permit to at least one. Thus this is a valid fractional algorithm. We just need to bound the cost of this algorithm relative to the optimum. We will first prove a lemma about the cost of an operation.

Lemma 5.2. *Each operation increases the fractional cost by at most* 2*.*

Proof. Suppose that at time t we perform an operation. We must have $\Sigma_{i=1}^{i=K} F_i(t) < 1$ before the operation is performed. In the first step, we multiply $F_i(t)$ by $1 + \frac{1}{C_i}$. This increases the fractional value by an additive $\frac{F_i(t)}{C_i}$, thus increasing the cost by $F_i(t)$. Summing this over all i, the total cost increases by at most 1. In the second step, we increase the value of $F_i(t)$ additively by $\frac{1}{KC_i}$, which increases the cost by $\frac{1}{K}$. Summing this over all i yields another increase of 1. So the total increase in cost is at most 2. □

Now if we can bound the total number of operations, we can bound the total cost. This allows us to compare the cost to optimum.

Theorem 5.3. *The online fractional algorithm is* $O(\log K)$*-competitive.*

Proof. Consider some interval during which the optimum solution purchases a permit of type i. The optimum pays C_i. We analyze the amount paid by the fractional algorithm over this interval. The first C_i operations each increase the fractional value for this permit by at least $\frac{1}{KC_i}$, so after C_i operations we have at least $\frac{1}{K}$ of the permit. From this point on, each operation multiplies the value by a factor of $1 + \frac{1}{C_i}$. So after $O(C_i \log K)$ operations, we have purchased the permit in its entirety. Once this happens, no more operations can occur during the permit's duration. Applying the lemma, we pay at most $O(\log K)$ times more than the optimum during this interval; the same reasoning applies to any interval so our overall payment is bounded as described. □

5.2 Randomized Lower Bound

Suppose we have K different permit types, with permit type i having cost $C_i = 2^i$. The duration of a type $i+1$ permit is *much* longer than the duration of a type i permit; in effect we assume that an infinite number of permits of the next lower type would be required to span the duration of any given permit. Of course, this is not a real problem instance, but we can achieve effectively the same thing by allowing the ratio of durations to be much larger than K.

Instead of considering randomized algorithms directly, we will consider deterministic algorithms operating on a randomized sequence of driving days. We then produce a randomized lower bound.

The sequence of driving days will be as follows. We operate over a time span equal to the duration of the longest permit. For $i > 1$, each interval of type i is divided into a very large number of intervals of type $i-1$. Every interval is classified as either active or inactive. The top-level interval of type K is always active. If an interval is active, its first sub-interval of type $i-1$ is always active. Subsequent sub-intervals of type $i-1$ are active only if the preceding sub-interval was active, and then only with probability $\frac{1}{2}$. So the jth sub-interval is active with probability $\frac{1}{2^{j-1}}$. If an interval is inactive, all its sub-intervals are inactive as well. At the bottom level, any active interval of type 1 contains exactly one driving day.

Theorem 5.4. *The expected cost of any deterministic algorithm is at least 2^K.*

Proof. Consider the choice of whether to buy a permit of type k. Purchasing the permit will cost 2^k. On the other hand, we could instead purchase a permit of type $k-1$ for the current interval, then purchase another permit of type $k-1$ for each remaining active interval until time D_k expires. The expected cost of this will be $\Sigma_{i=0}^{R_k} 2^{k-1}\frac{1}{2^i} < 2^k$. So the deterministic algorithm always does better by not purchasing a permit of type k. This holds for every $k > 1$, so the best expected performance is obtained by the algorithm which purchases only type one permits on each driving day. The expected cost paid by this algorithm is the expected number of driving days. Let d_k be the expected number of driving days during an active interval of type k. We have $d_1 = 1$ and:

$$d_k = \Sigma_{i=0}^{R_k} d_{k-1}\frac{1}{2^i} \approx 2d_{k-1}$$

The approximate equality would be equality if in fact we could set $R_k = D_k/D_{k-1} = \infty$. However, for sufficiently large R_k (say $R_k > k$) we will be close enough. □

Theorem 5.5. *The expected cost of an optimum offline solution is at most $\frac{2^{K+1}}{\log K}$.*

Proof. We consider the following offline algorithm. We purchase a permit of type $k+1$ if and only if it contains at least $1 + \log k$ active intervals of type k.

Let γ_k represent the expected cost which this offline algorithm pays for an active interval of length D_k. We will inductively prove that $\gamma_k \leq \frac{2^{k+1}}{\log k}$. This is obvious for $k \leq 4$ since of course $\gamma_k \leq C_k = 2^k$. We can write the following recurrence:

$$\gamma_{k+1} = (\Sigma_{i=1}^{i=\log k} Pr[\rho = i] i \gamma_k) + Pr[\rho > \log k] 2^{k+1}$$

Here ρ is the number of active intervals of the next shorter type of permit. By definition, the probability that $\rho = i$ is just 2^{-i} so we can rewrite the expression as:

$$\gamma_{k+1} = (\gamma_k \Sigma_{i=1}^{i=\log k} 2^{-i} i) + \frac{1}{k} 2^{k+1}$$

We can simplify this expression by noting that $\Sigma_{i=1}^{i=\log k} i2^{-i} = 2 - \frac{2+\log k}{k}$. We now make use of our inductive hypothesis, assuming that γ_k meets the desired bound. This substitution along with the evaluation of the sum yields:

$$\gamma_{k+1} \leq \frac{2^{k+1}}{\log k}(2 - \frac{2+\log k}{k}) + \frac{2^{k+1}}{k}$$

$$\gamma_{k+1} \leq \frac{2^{k+2}}{\log k} - \frac{2^{k+2}}{k \log k}$$

$$\gamma_{k+1} \leq 2^{k+2}(\frac{1}{\log k} - \frac{1}{k \log k})$$

In order to complete the proof, we need to show that $\frac{k-1}{k \log k} \leq \frac{1}{\log(k+1)}$. This is always true for $k \geq 3$. $\square$

Theorem 5.6. *Any randomized algorithm for the Parking Permit problem has expected competitive ratio at least* $\Omega(\log K)$.

Proof. Consider the randomized sequence described. By theorem 5.4, any deterministic algorithm has expected cost at least 2^K. Since any randomized algorithm is just a probability distribution over deterministic algorithms, it follows that any randomized algorithm has expected cost at least 2^K on this randomized sequence. On the other hand, the expected cost of the optimum is at most $\frac{2^{K+1}}{\log K}$. If for every sequence of driving days, the expected cost of a particular algorithm was less than $\frac{\log K}{2}$ times the optimum cost, then a contradiction would be reached. So for any randomized algorithm, there must be some particular sequence for which the expected cost of the algorithm is at least $\frac{\log K}{2}$ times the offline optimum. $\square$

6 Results Depending upon Durations or Costs

We can model the competitive ratio as depending upon the durations or costs, instead of the number of permits. The natural parameters are $\frac{C_K}{C_1}$ and $\frac{D_K}{D_1}$. We observe that by applying theorem 2.1, we can make the assumption that $K \leq \log \frac{C_K}{C_1}$. Via a similar scaling technique, we can assume that $K \leq \log \frac{D_K}{D_1}$. The following theorems follow:

Theorem 6.1. *There is a deterministic algorithm for the parking permit problem which obtains competitive ratio* $O(\log \frac{C_K}{C_1})$*, and a randomized algorithm which obtains competitive ratio* $O(\log \log \frac{C_K}{C_1})$.

Theorem 6.2. *There is a deterministic algorithm for the parking permit problem which obtains competitive ratio* $O(\log \frac{D_K}{D_1})$*, and a randomized algorithm which obtains competitive ratio* $O(\log \log \frac{D_K}{D_1})$.

In proving our lower bounds, we notice that in each case we used $C_k = 2^k$, which allows the same lower bounds to provide lower bounds in terms of the ratio of C_K to C_1. However, in the randomized lower bound the ratio of durations was assumed to be very large. We can choose $D_k = 2^{k^2}$ and still obtain good results, and since we are examining the logarithm of the logarithm, our lower bound will still match the upper bound.

Theorem 6.3. *No deterministic algorithm can obtain competitive ratio better than* $\Omega(\log \frac{C_K}{C_1})$ *for the parking permit problem. No randomized algorithm can obtain competitive ratio better than* $\Omega(\log \log \frac{C_K}{C_1})$.

Theorem 6.4. *No deterministic algorithm can obtain competitive ratio better than* $\Omega(\log \frac{D_K}{D_1} / \log \log \frac{D_K}{D_1})$ *for the parking permit problem. No randomized algorithm can obtain competitive ratio better than* $\Omega(\log \log \frac{D_K}{D_1})$.

References

[1] N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, and J. Naor. The online set cover problem. *ACM Symposium on Theory of Computing*, 2003.

[2] B. Awerbuch and Y. Azar. Buy-at-bulk network design. *Proceedings of the 38th IEEE Symposium on Foundations of Computer Science*, pages 542–47, 1997.

[3] P. Berman and C. Coulston. On-line algorithms for steiner tree problems. *Proceedings of the 29th ACM STOC*, 1997.

[4] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1996.

[5] S. Guha, A. Meyerson, and K. Munagala. Improved combinatorial algorithms for single sink edge installation problems. *ACM Symposium on Theory of Computing*, 2001.

[6] S. Guha and K. Munagala. Improved algorithms for the data placement problem. *ACM Symposium on Discrete Algorithms*, 2002.

[7] A. Gupta, A. Kumar, M. Pal, and T. Roughgarden. Approximation via cost-sharing: A simple approximation algorithm for the multicommodity rent-or-buy problem. *IEEE Symposium on Foundations of Computer Science*, 2003.

[8] M. Korupolu, G. Plaxton, and R. Rajaraman. Placement algorithms for hierarchical cooperative caching. *ACM Symposium on Discrete Algorithms*, 1999.

[9] A. Kumar, A. Gupta, and T. Roughgarden. A constant-factor approximation algorithm for the multicommodity rent-or-buy problem. *IEEE Symposium on Foundations of Computer Science*, 2002.

[10] A. Meyerson, K. Munagala, and S. Plotkin. Web caching using access statistics. *ACM Symposium on Discrete Algorithms*, 2001.

[11] F. S. Salman, J. Cheriyan, R. Ravi, and S. Subramanian. Buy-at-bulk network design: Approximating the single-sink edge installation problem. *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 619–628, 1997.