Scheduling Jobs Before Shut-Down

Vincenzo Liberatore[1]

[1]UMIACS, A. V. Williams Building, University of Maryland, College Park, MD 20742 (vliberatore@acm.org). URL: http://www.umiacs.umd.edu/users/liberato/.

Abstract

Distributed systems execute background or alternative jobs while waiting for data or requests to arrive from another processor. In those cases, the following *shut-down scheduling problem* arises: given a set of jobs of known processing time, schedule them on m machines so as to maximize the total weight of jobs completed before an initially unknown deadline. We will present optimally competitive deterministic and randomized algorithms for shut-down scheduling. Our deterministic algorithm is parameterized by the number of machines m. Its competitive ratio increases as the number of machines decreases, but it is optimal for any given choice of m. Such family of deterministic algorithm can be translated into a family of randomized algorithms that use progressively less randomization and that are optimal for the given amount of randomization. Hence, we establish a precise trade-off between amount of randomization and competitive ratios.

1 Introduction

Internet users experience substantial delays while retrieving Web documents. Web performance is particularly degraded by the self-similar nature of Internet traffic [CB98, WP98], in which lulls alternate with spikes of extreme activity. As a result, Web performance is especially improved when operations are moved from peak periods to intervening lulls. For example, idle periods can be exploited by servers to speculatively disseminate data to dial-up clients, thus substantially reducing the latency experienced to retrieve subsequent documents [FCJ99]. The "polite push" framework is currently the major feature of a commercial system[1]. Delays can stall the execution of a distributed query in a Web-based database system so as to trigger alternate queries or query plans [UFA98].

Shut-Down Scheduling. Such systems share the following core optimization problem: a set of alternative or background jobs can be scheduled during a lull. A lull has unknown duration because it ends asynchronously when a message is received from the network. We will refer to such problem as *shut-down scheduling* because jobs execution is unpredictably interrupted. The off-line version of shut-down scheduling is a *maximum 0/1 multiple knapsack problem* where all knapsacks have the same capacity. A book summarizes results in the theoretical and practical solution of the multiple knapsack problem [MT90]: it is strongly NP-hard [MT90], and a polynomial-time approximation scheme has been recently discovered [Kel99]. Several authors have considered an on-line single knapsack problem where the deadline is known in advance, and jobs arrive on-line [Lue95, MSV95]. The on-line knapsack problem can be regarded as the dual of shut-down scheduling and it is substantially an *admission control* problem [BEY98, GGK+97]. In general, shut-down scheduling is related to on-line *call control* [BEY98, GGK+97], *load balancing* [BEY98, Gra66], and *bin packing* [BEY98, CGJ97, GGU73]. Scheduling with *machine breakdowns* has been considered as well [KP94, KP97, AS98]: jobs must be scheduled on m processors so as to complete in the presence of permanent or transient faults. Breakdowns are different from shut-down in several respects, as, for example, arrival times, objective function, job restart, redundant scheduling, and for the techniques and results of the analysis.

Our results. We will present optimally competitive deterministic and randomized algorithms for shut-down scheduling. We have obtained preliminary experimental results on actual Web trace simulations [LD99]. Such results indicate that a competitive algorithm indeed outperforms natural, but non-competitive strategies. We will also express the optimal competitive ratio as a function of the number of randomization bits. Consequently, we will establish a precise trade-off between competitive ratios and the amount of randomization. Randomized algorithms can be fully derandomized provided that there are sufficiently many machines. If there is only a small number m of machines, we will give an optimal deterministic algorithm CSM that is parameterized by m. The competitive ratio of CSM increases as m decreases, but, for any given choice of m, our CSM algorithm is optimal. We will also interpret CSM as a family of randomized algorithms that use progressively less randomization at the price of a worse competitive ratio. Our algorithm is optimal for any given choice of the amount of randomization and coincides with the optimal deterministic and randomized algorithms in the two extreme cases. Thus, such algorithm establishes the claimed trade-off between randomization and competitive ratio. Deterministic algorithms and lower bounds are transformed into randomized algorithms and lower bounds by a technique that is simple and that might be more generally applicable to other scheduling problems.

[1] See `http://www.backweb.com/` or the i icon on Compaq and HP personal computers.

Probabilistic Analysis. We will also conduct a probabilistic analysis of algorithms for shut-down scheduling on $m = 1$ machine. Probabilistic analyses of knapsack problems have been performed by several authors [DP82, Lue82, GMS84, MRKSV90, SL87]. A probabilistic analysis was also performed for the on-line case [Lue95, MSV95]. We will focus on shut-down scheduling and on the case when the deadline D is exponentially distributed. Such probabilistic assumptions models the case of a Web traffic lull that is interrupted by the arrival of a client request [FCJ99], and client arrivals follow a Poisson process. We will show a policy that maximizes the expected profit for the exponential distribution. We also present a shut-down schedule that breaks ties among jobs so as to minimize variance without worsening expected profit. Therefore, the resulting strategy is, in the parlance of portfolio theory, *E, V efficient* [Mar70]. Our tie-breaking procedure provides some justification to the proxy server scheduling algorithm in [FCJ99] if the deadline D is uniformly distributed.

Contents. The paper is organized as follows. In §2, we introduce our notation for shut-down scheduling. In §3, we present competitive analyses and give optimal deterministic and randomized algorithm for shut-down scheduling. In §4, we conduct a probabilistic analysis.

2 Preliminaries

In this section, we give definitions and notations for the shut-down scheduling problem. First, we introduce our notation for the $m = 1$ machine problem. The *maximum 0/1 knapsack* problem is: given *lengths* $l(i) \in \mathbb{N}$ ($i \in [n] \stackrel{\text{def}}{=} \{1, 2, \ldots, n\}$), *profits* $p(i) \in \mathbb{N}$ ($i \in [n]$), and a *deadline* D, find a subset $J \subseteq [n]$ such that $\sum_{i \in J} l(i) \leq D$ that maximizes $\sum_{i \in J} p(i)$. We will now introduce some notation. The profit $p(J)$ and length $l(J)$ of a set $J \subseteq [n]$ are defined in the obvious way: $p(J) = \sum_{i \in J} p(i)$ and $l(J) = \sum_{i \in J} l(i)$. Let $p^*(D) = \max_{J : l(J) \leq D} p(J)$ be the optimum 0/1 knapsack objective value. Let $\pi = (\pi_1, \pi_2, \ldots, \pi_k)$ be a k-permutation of $[n]$, define $J_i = \{\pi_1, \pi_2, \ldots, \pi_i\}$ ($1 \leq i \leq k$) and $p(\pi, D) = p(J_s)$ where s is the largest integer such that $l(J_s) \leq D$. Note that when $k < n$ and $D > l(J_k)$, the machine will remain idle after the completion of the k scheduled jobs. Although it does not seem intuitive, some algorithms will in fact exploit the fact that only some of the jobs are scheduled. Finally, we will omit the reference to D in $p^*(D)$ and $p(\pi, D)$ when the deadline D is clear from the context.

We will consider a two-person zero-sum game which is based on the maximum 0/1 knapsack problem and which we call the *knapsack game*. In the knapsack game, all the values $l(i)$ and $p(i)$ ($i \in [n]$) are known at the beginning, but the deadline D is not. The player G selects a permutation π of $[n]$ and the player H chooses D. If $p^*(D) > 0$, the quantity $v(\pi, D) \stackrel{\text{def}}{=} p(\pi, D)/p^*(D)$ will be G's *payoff* corresponding to the strategies π and D. If $p^*(D) = 0$, we define G's payoff to be one. The objective of G is to maximize its payoff in the game. Since G's payoff is always at most one, we can assume without loss of generality that $D \geq \min_{i \in [n]} l(i)$. Notice that G has at most $n!$ strategies and H has at most 2^n strategies, so that, for a given n, the knapsack game is a finite matrix game. We interpret the knapsack game as an on-line problem as follows. We have a set of n jobs numbered from 1 to n. Each job has a profit $p(i)$ and it takes $l(i)$ units of time to be completed. The on-line algorithm G starts to schedule jobs on one machine according to some ordering π. At time D, the adversary shuts the machine down, and G gains the values of all the jobs completed before D. A strict *competitive ratio* is an upper bound to the inverse of the game value. We do not allow additive terms in the competitive ratio because the game is finite. We remark the difference among the following quantities relative to an (on-line) algorithm:

Profit Total profit of jobs completed before the deadline

Payoff Ratio of the algorithm's profit over the adversary's. The payoff is relative to the chosen strategies for the on-line algorithm and for the adversary.

Game Value Best payoff an on-line player can achieve.

Competitive Ratio An upper bound on the inverse of the game value.

It can be noticed that the competitive ratio is defined in terms of inverse of game values, which is the correct choice in maximization on-line games [BEY98]. We notice that the knapsack game is trivial if all $p(i)$'s are equal (choose the jobs in non-decreasing length order) or if all $l(i)$'s are equal (choose the jobs in non-increasing profit order). In the more general scenario, we will assume that a job can be scheduled on any one of m machines that run at the same speed. The adversary will shut all machines down at the deadline D.

Henceforth, we will use natural logarithms because they simplify notation and derivatives. Of course, the logarithm base does not alter the order of asymptotic bounds. Finally, we introduce some quantities that will be fundamental to the analysis below. Define $V = \max_{i\in[n]}\{p(i)\}/\min_{i\in[n]}\{p(i)\}$ as the ratio of the largest to the smallest profit. Another important quantity is L, the number of distinct length values in the job set, that is, $L = |\{l(j) : j \in [n]\}| \leq n$. Finally, we define the *critical number of machines* $\mu = \min\{L-1, \ln V\}$.

3 Competitive Analysis

In this section, we conduct competitiveness analysis for the shut-down scheduling problem.

3.1 Randomized Algorithms

We present strongly competitive randomized algorithms for shut-down scheduling. Here, we will focus on the case when $L, V \neq O(1)$, and we will obtain different competitive ratios depending on the relative growth rate of L and V. We begin with a lower bound on the case of $m = 1$ machine

Lemma 3.1 *No randomized algorithm for the knapsack game can be better than $\Omega(L)$-competitive when $V = \Omega(2^L)$ and better than $\Omega(\log V)$-competitive when $V = o(2^L)$.*

The proof will exploit the minimax principle [BEY98].

Proof Sketch. Let $\rho = 1/\sqrt[L-1]{V}$, $l(i) = n + i - 1$ and $p(i) = \lfloor p(1)\rho^{1-i}\rfloor$ for all $i \in [n]$. Notice that $p(1) \leq p(2) \leq \ldots \leq p(n) \leq V$, so that the ratio of the largest to the smallest value is indeed bounded by V. The two fundamental points of the proof are the following. First, if $D \leq 2n-1$, at most one job can be scheduled before the deadline. If the on-line player guesses the right job, its payoff is one. If it guesses a job that is longer than the deadline, its payoff is naught. Finally, if it guesses a job that is shorter than the deadline, its payoff is limited by the exponential growth of profits. The second point is to use the minimax principle as follows. Let $c = n(1-\rho)+\rho$. We can show a probability distribution over D that forces any deterministic on-line strategy to have payoff $O(1/c)$. By the minimax principle, the value of the game is $O(1/c)$, and so the competitive ratio of a randomized on-line algorithm is $\Omega(c) = \Omega(n(1-\rho))$. An asymptotic analysis of c completes the proof. $\square$

We now show that the same lower bound holds for an arbitrary number m of machines.

Corollary 3.2 *No randomized algorithm for the shut-down scheduling on m machines can be better than $\Omega(L)$-competitive when $V = \Omega(2^L)$ and better than $\Omega(\log V)$-competitive when $V = o(2^L)$.*

Proof. The proof is a reduction to the case of $m = 1$ machine. Consider the same counterexample as in Lemma 3.1 on n' jobs and replicate each job for m times, so that the total number of jobs is now $n = n'm$. The number L of length classes remains unchanged. Again, at most one job can complete on any machine when $D \leq 2n - 1$. We will show how to convert any randomized algorithm for the m machine instance into a randomized algorithm for the original one-machine problem so that the two schedules achieve the same payoff. If $D \leq 2n-1$, any randomized strategy for m machines is completely characterized by the expected number f_i of machines starting a job of length $n+i-1$. Let h be the index with $D = l(h)$. By linearity of expectation, the on-line expected profit is $\sum_{i=1}^{h} f_i p(i)$. Meanwhile, the adversary's profit is $mp(h)$, so that the on-line expected payoff is $\sum_{i=1}^{h} f_i p(i)/(mp(h))$. Consider an on-line algorithm for the one machine instance that schedules job i with probability f_i/m. Its expected payoff is exactly the same as the m machine algorithm for any choice of deadline $D \leq 2n - 1$. Hence, the same lower bound as in Lemma 3.1 applies, and the proof is complete. □

If $m > 1$ and all $l(i)$'s are equal, then shut-down scheduling is trivial (schedule jobs in non-increasing profit order). We turn to the case when the $p(i)$'s are equal, and show an $O(1)$-competitive algorithm. Such algorithm is an intermediate step to solve the general case of different profits. First, notice that when all profits are equal, our objective is to maximize the number of completed jobs. Define the *canonical job scheduling* algorithm for a set $C \subseteq [n]$ as a list scheduling algorithm [Gra66] that orders the jobs from the shortest to the longest.

Lemma 3.3 *Let $C \subseteq [n]$ be a set of jobs. The canonical schedule of C completes at least $1/5$ of the jobs completed by any other algorithm before the deadline D.*

Define the *load* of a machine as the total length of jobs completed on that machine before the deadline and the *makespan* as the maximum load of any one machine. The proof will exploit a result for load balancing of permanent jobs [Gra66].

Proof. The proof is organized as follows. We partition the jobs executed by the optimum into five classes, depending on their starting and completion time with respect to the deadline D and the makespan of the canonical schedule. Then, we show that no class contains more jobs than those completed by the canonical schedule before the deadline D. Hence, the canonical schedule completes at least $1/5$ of the jobs completed by the optimum, which will complete the proof.

Assume without loss of generality that jobs are numbered in non-decreasing order of length, that is, $l(j+1) \geq l(j)$ for $j = 1, 2, \ldots, n-1$. Let G be the set of jobs completed by the canonical schedule before D and let M_G the makespan of G, that is the time the last job in G is completed. By definition, $M_G \leq D$. Observe that initially G starts by assigning one job in $[m]$ to each machine. Let H be a the largest set of jobs completed before D. Suppose first that $|H| \leq m$. Then, D is at least the length of the longest job in H, which is at least $l(|H|)$. Hence, G completes at least one job on at least $|H|$ machines and the claim is proven. Assume now $|H| > m$, so that $D \geq l(|H|) \geq l(m)$. Hence, $|G| \geq m$. We now make the following definition: if $X \subseteq [n]$ is a set of jobs, then M_X^* is the minimum makespan to complete X. Observe that if $X \subseteq Y$, then $M_X^* \leq M_Y^*$. Analogously, if $X, Y \subseteq [n]$ and if there is a one-to-one mapping $f : Y \to X$ with $l(j) \geq l(f(j))$ for all $j \in Y$, then $M_X^* \leq M_Y^*$. Let M_G^* be the earliest time when G can be completed. A load balancing results claims that $2M_G^* > M_G$. Schedule H on m machine so that the schedule completes before time D

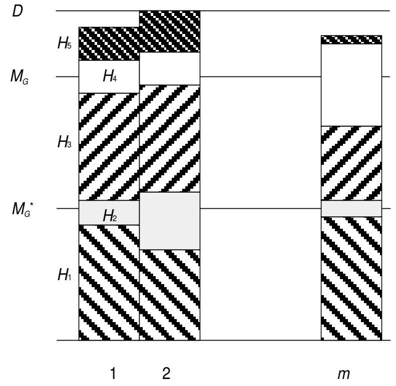

Figure 1: A partition of the optimal set H of jobs according to the optimal makespan M_G^* and the actual makespan M_G of the on-line algorithm.

and partition H into five subsets according to such schedule as follows (Figure 1 gives an example of such partition).

- The subset $H_1 \subseteq H$ is the set of jobs that complete before time M_G^*. We claim that $|H_1| \leq |G|$. Suppose by contradiction $|H_1| > |G|$. Take any proper subset $H' \subset H_1$ with $|H'| = |G|$ elements. Since G consists of the $|G|$ shortest jobs, there is a one-to-one mapping $f : H' \rightarrow G$ with $l(j) \geq l(f(j))$ for all $j \in H'$. Hence, the optimal makespan $M_{H'}^*$ of H' is not smaller than the optimal makespan M_G^* of G. Therefore, $M_G^* \leq M_{H'}^* \leq M_{H'}^*$ and a contradiction is reached. Therefore, we conclude $|H_1| \leq |G|$.

- The subset $H_2 \subseteq H$ is the set of jobs that start before time M_G^* and complete after time M_G^*. The set H_2 contains at most one job per machine, so that $|H_2| \leq m \leq |G|$.

- The subset $H_3 \subseteq H$ is the set of jobs that starts after time M_G^* and complete after time M_G. Since $M_G - M_G^* < M_G^*$, we conclude that $|H_3| \leq |G|$ with an argument similar to H_1.

- The subset $H_4 \subseteq H$ is the set of jobs that starts before time M_G and complete after time M_G. H_4 contains at most one job per machine and so $|H_4| \leq m \leq |G|$.

- The subset $H_5 \subseteq H$ is the set of jobs that start after M_G. Notice that $D - M_G < l(|G| + 1)$ or else G would have scheduled one more job. Hence, H_5 can contain at most $|G|$ documents because job $|G| + 1$ does not fit in the alloted time $D - M_G < l(|G| + 1)$.

Notice that $H_1, H_2, \ldots, H_5$ give indeed a partition of H. We conclude that $|H| \leq 5|G|$, which proves the lemma. □

Corollary 3.4 *The canonical schedule is 5-competitive for shut-down scheduling on any number m of machines when $p(i) = 1$ for all jobs i.*

The canonical schedule algorithm easily generalizes to the case of arbitrary profits by using the CRS techniques. Partition the job set into $O(\log V)$ *profit classes* such that no job is more than $O(1)$ times as profitable as any other job in the same class. Then, extract a profit class at random and execute jobs only from that class. However, if $V = \Omega(2^L)$, then jobs are partitioned according to their length in such a way that a job class contains only jobs of the same length. We conclude that

	$V = o(2^L)$	$V = \Omega(2^L)$
$m \leq \mu$	$\Theta(m\rho)$	$\Theta(m\rho)$
$m > \mu$	$\Theta(\log V)$	$\Theta(L)$

Table 1: Competitive ratios of the *best* deterministic algorithm for the m machine knapsack game, where L is the number of length classes, V is the ratio of largest and smallest profit, $\mu = \min\{L - 1, \ln V\}$ is the critical number of machines, and $\rho = \sqrt[m]{V}$.

Theorem 3.5 *The best algorithm for shut-down scheduling is $\Theta(L)$-competitive when $V = \Omega(2^L)$ and $\Theta(\log V)$-competitive when $V = o(2^L)$.*

A consequence of the matching upper and lower bounds is that if we change the number m of machines, we do not help nor hamper the competitive ratio of randomized algorithms.

3.2 Deterministic Algorithms

We now turn to deterministic algorithms. It is helpful during the discussion to refer to table 1 which summarizes our results. First, we argue that if there is a sufficiently large number $m > \mu$ of machines, then, we can find deterministic algorithms that match the randomized lower bound. We derandomize the CRSlog algorithm as follows. If we have $m \geq \mu + 1 \geq \ln V + 1$ machines, we can assign $m' = \lceil m/(\ln V + 1)\rceil$ machines to process jobs in each profit class according to the canonical schedule. Roughly speaking, the derandomized version translates the probability of executing jobs in class C_i into the fraction of machines assigned to class C_i. It is critical that profit classes be disjoint sets, as otherwise a job would have to be scheduled on more than one machine.

Lemma 3.6 *The derandomized version of the CRSlog algorithm is $O(\log V)$-competitive for shut-down scheduling on $m \geq \ln V + 1$ machines.*

Proof. At any time, the algorithm has completed at least 1 / 5 of the jobs in a certain class that are completed by any other algorithm that uses m' machines for that class. Hence, the adversary completes in each class at most $5m/m' = O(\log V)$ jobs more than the derandomized CRSlog algorithm, and, on each job, it earns less than e times as much as the derandomized CRSlog. Thus, such algorithm is $O(\log V)$-competitive. □

We can analogously derandomize the $O(L)$-competitive algorithm as long as we have $m \geq \mu + 1 \geq L$ machines. It remains to establish deterministic competitive ratios for $m \leq \mu$ machines. In this case, Corollary 3.2 is tight for randomized algorithms, but gives a weak lower bound for deterministic algorithms. Intuitively, the weakness of Corollary 3.2 stems from the fact that it is not always possible to execute simultaneously all deterministic strategies that compose a randomized algorithm if only few machines are available. Define $\rho \overset{\text{def}}{=} \sqrt[m]{V}$ (such notation is independent of that in Lemma 3.1) and notice that $\rho > 1$. We will frequently use the equalities $\ln \rho = (\ln V)/m$ and $m = \mu$.

Lemma 3.7 *If $m \leq \min\{L - 1, \ln V\}$, then no deterministic algorithm can be better than $\Omega(m\rho)$-competitive.*

Notice that $m\rho = \omega(m \log \rho) = \omega(\log V)$. On the other hand, if $V = \Omega(2^L)$, then $\rho = \Omega(2^{L/m}) = \omega(L/m)$, and so $m\rho = \omega(L)$. Hence, Lemma 3.7 dominates Corollary 3.2 when $m \leq \mu$.

Proof. The proof is based on an instance with the property that the adversary will be able to choose a bad deadline for any on-line algorithm. The instance consists of $m+1$ classes of m identical jobs such that jobs in class i are ρ times more valuable than jobs in class $i-1$. Job lengths are chosen in such a way that only one job can complete on any one machine, which is similar to the length distribution of Lemma 3.1. Since there are more classes than machines, the on-line algorithm does not schedule any job from a certain class. The adversary chooses the deadline so that the optimum strategy schedules jobs only from that class, while the on-line algorithm achieves a small profit. We will now give the details of the arguments.

Set-up. The proof is based on an instance where there are m identical jobs of profit $\lfloor p_0 \rho^i \rfloor$ $(i = 0, 1, 2, \ldots, m)$ for some minimal profit p_0. Hence, the total number of jobs is $n = m(m+1)$. Notice that $L = m + 1 > m$. Observe that the minimum profit is p_0 and the maximum profit is no more than $p_0 V$. A profit class is a set of jobs with the same profit. We will think of profit classes as ordered by the profit of the jobs they contain. A job of profit $p_0 \rho^i$ has length $2m + i$. If $D \leq 3m$, then at most one job can complete on any one machine.

Holes. Since the number of profit classes is $m+1$, there is at least one profit class from which no job is completed before the deadline $D \leq 3m$. We will say that a *hole* is a maximal non-empty sequence of profit classes with the property that no job has been scheduled from any class in the hole. Clearly, there is at least one hole. If the first class C_0 is in a hole, then the adversary will set $D = 2m$, and the on-line algorithm achieves no profit. Therefore, we can assume from now that the first class is not in a hole without loss of generality. Suppose that the holes are $\mathcal{H}_1 \prec \mathcal{H}_2 \prec \ldots \prec \mathcal{H}_l$. Let k_i be the number of jobs scheduled from the class immediately preceding hole $\mathcal{H}_i$. We claim that there is at least one hole $\mathcal{H}_i$ for which $k_i \leq |\mathcal{H}_i| + 1$. Suppose that this is not true. Then, denote by ν the number of classes that are not immediately followed by a hole and observe that the total number of jobs is at least $\nu + 2l + \sum_{i=1}^{l} |\mathcal{H}_i| = m + l > m$, which is a contradiction. The adversary considers an hole $\mathcal{H}$ such that $|\mathcal{H}| + 1$ is a bound on the number of jobs scheduled in the class immediately before the hole.

Payoff. Suppose that hole $\mathcal{H}$ extends from class $h+1$ to class $h+\alpha$ for some $\alpha \geq 1$. The on-line algorithm schedules at most $\alpha + 1$ jobs in of value ρ^h. The adversary chooses $D = 2m + h + \alpha$ and achieves a profit $p^* \geq m(\rho^{h+\alpha} p_0 - 1)$. Meanwhile, the on-line algorithm achieves a profit $p(G) \leq (\alpha+1)\rho^h p_0 + (m - (\alpha+1))\rho^{h-1} p_0$. Hence, for any $\epsilon > 0$,

$$\frac{p^*}{p(G)} \geq \frac{m\rho^{\alpha+1}}{(\alpha+1)\rho + m - (\alpha+1)} - \epsilon \,.$$

The function $f(x) = m\rho^x/(x\rho + m - x)$ is increasing whenever $x\rho + m - x > (\rho - 1)/\ln \rho$, that is, for all x's with $x > 1/\ln \rho - m/(\rho - 1)$. However, $1/\ln \rho - m/(\rho-1) \leq m/\ln V \leq 1$, which implies that $f(x)$ is increasing when $x \geq 2$. Hence, the profit ratio is minimized whenever $\alpha = 1$, so that

$$\frac{p^*}{p(G)} \geq \frac{m\rho^2}{2\rho + m - 2} - \epsilon \,.$$

If $\rho < m$, then $p^*/p(G) > m\rho^2/(3\rho - 2) - \epsilon \geq m\rho^2/\rho - \epsilon = m\rho - \epsilon$. If $\rho \geq m$, then $p^*/p(G) \geq m\rho^2/(3m-2) - \epsilon \geq m\rho^2/m - \epsilon \geq m\rho - \epsilon$, and the proof is complete. $\square$

Such lower bound is matched by the following CSM (Canonical Schedule for m machines) algorithm. First, normalize job profits so that the minimum profit is one and the maximum profit is V. CSM divides the job set into m classes according to job profit, where class i consists of jobs of profit $\rho^{i-1} \le p(j) < \rho^i$. CSM schedules jobs of class i on machine i according to the canonical schedule.

Lemma 3.8 *The CSM algorithm is $O(m\rho)$-competitive for the knapsack game on m of machines.*

Proof. Define the hth profit class as $C_h = \{j : \rho^{h-1} \le p(j) < \rho^h\}$. Let x_h be the number of jobs in class h scheduled by the optimum before the deadline D. Hence, the optimum profit from class h is less than $x_h \rho^h$. The CSM algorithm schedules at least a $x_h/(5m)$ jobs in class h, so that its profit is at least $x_h \rho^{h-1}/(5m)$. Hence, CSM has a payoff of at least

$$\frac{\sum_{h=1}^{m} x_h \rho^{h-1}}{m \sum_{h=1}^{m} x_h \rho^h} \ge \frac{\sum_{h=1, x_h \neq 0}^{m} x_h \rho^{h-1}}{m \sum_{h=1, x_h \neq 0}^{m} x_h \rho^h} \ge \frac{1}{5m} \frac{1}{\rho} \ge \frac{1}{5m\rho} .$$

The competitive ratio is the inverse of the payoff, and thus the proposition is proven. □

The lower bound and the CSM algorithm are summarized by:

Theorem 3.9 *The best algorithm for the knapsack game on $m \le \min\{L-1, \ln V + 1\}$ machines is $\Theta(m \sqrt[m]{V})$-competitive.*

3.3 Reduced Randomization

The CSM algorithm can be translated into a randomized algorithm, where a random class of jobs is scheduled according to the canonical schedule. We will name the resulting randomized algorithm CSMr.

Lemma 3.10 *The CSMr algorithm is $O(m\rho)$-competitive for the shut-down scheduling problem.*

Proof. Since there are m profit classes, jobs in the same classes have profits that are within a factor of ρ, and the canonical schedules has a performance guarantee of 5, we obtain that the payoff of CSM is at least $1/(5m\rho)$. □

Hence, CSMr gives a precise trade-off between randomization and competitive ratio. Indeed, if no randomization is allowed, then the best algorithm is $\Theta(V)$-competitive. As the amount of randomization increases to m strategies ($m \le \ln V$), performance improves as $\Theta(m \sqrt[m]{V})$. Finally, when $m = \ln V + 1$, the best algorithm is $\Theta(\log V)$-competitive and, if $V = o(2^L)$, no further improvement stems from adding more machines. Meanwhile, we claim that CSMr achieves optimal performance.

Theorem 3.11 *The best randomized algorithm that is a distribution over only m deterministic strategies is $\Theta(m\rho)$-competitive.*

Proof Sketch. Consider the proof of Lemma 3.7 and replace the number of machine starting a certain job class with the expected number of machines. □

4 Probabilistic Analysis

In this section, we will conduct probabilistic analyses of shut-down scheduling on $m = 1$ machine. Let $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ be a permutation of $[n]$ and $J_i = \{\pi_1, \pi_2, \ldots, \pi_i\}$. Then,

$$E[p(\pi)] = \sum_{i=1}^{n} p(J_i)Pr[l(J_i) \leq D < l(J_{i+1})] = \sum_{i=1}^{n} p(\pi_i)Pr[D \geq l(J_i)] \ . \tag{1}$$

Our objective is to find a permutation π that maximizes (1). A corresponding decision problem is to find a permutation π such that $E[p(\pi)] \leq \bar{p}$ for some given $\bar{p}$. Such decision problem is easily seen to be NP-complete as it reduces to a knapsack problem when there is a t with $Pr[D = t] = 1$. First, we give a general optimality criterion.

Lemma 4.1 *If a permutation π maximizes (1), then, for all $i \in [n-1]$,*

$$p(\pi_i)Pr[l(J_i) \leq D < l(J_{i+1})] \geq p(\pi_{i+1})Pr[l(J_{i-1}) + l(\pi_{i+1}) \leq D < l(J_{i+1})] \ .$$

Moreover, if π maximizes (1) and

$$p(\pi_i)Pr[l(J_i) \leq D < l(J_{i+1})] = p(\pi_{i+1})Pr[l(J_{i-1}) + l(\pi_{i+1}) \leq D < l(J_{i+1})] \ ,$$

then the permutation π' obtained by exchanging π_i and π_{i+1} is also optimal.

Proof Sketch. If this were not so, exchange jobs π_i and π_{i+1} to increase the profit. Analogously, if equality holds, the profit remains unchanged, and thus optimal. □

A simple corollary is that if all $p(i)$'s are equal, then the optimal solution is to arrange jobs in increasing order of length, independently of the distribution of D. Another simple consequence is that if D is uniformly distributed in an interval $[0, A]$ of the real line with $A \geq l([n])$, then he optimal permutation is to arrange jobs in non-increasing *profit density* $p(i)/l(i)$ (*PD-order*). Such result for the uniform distribution also follows by noticing that, under such distribution, the objective (1) defines a weighted completion time problem, and we can apply Smith's rule [Smi56].

We now turn to the case when the deadline D is extracted according to an exponential distribution with rate λ. The exponential distribution models the case when client requests arrive according to a Poisson process, and each request terminates a lull. First, recall that, if D is exponentially distributed, we have $Pr[D \geq t] = e^{-\lambda t}$. Then, expression (1) becomes

$$E[p(\pi)] = \sum_{i=1}^{n} p(\pi_i)e^{-\lambda l(J_i)} \ . \tag{2}$$

Define the *exponential density* of job i as the ratio $d_e(i) = p(i)/(e^{\lambda l(i)} - 1)$. The *Exponential Profit Density* (EPD) algorithm arranges jobs in non-increasing order of exponential profit density. We will say that a permutation is in *EPD-order* if its jobs are in non-increasing order of exponential profit density.

Theorem 4.2 *If $Pr[D \geq t] = e^{-\lambda t}$, then a permutation minimizes (2) if and only if it is in EPD-order.*

Proof. If a permutation is optimal, then the optimality condition of Lemma 4.1 implies that it is in EPD-order. Conversely, assume that the identity is an optimal EPD permutation, and suppose we exchange two terms h and $h+1$ with the same exponential value density. Lemma 4.1 implies that the new permutation is optimal as well. Any permutation in EPD-order can be obtained by a finite exchange of jobs with the same exponential profit density, and the proposition is proven. $\Box$

The previous theorem suggests that in some sense lengths are exponentially more important than values for an exponential distribution. On the other hand, an exponential distribution can be approximated by a uniform distribution when λ is large, in which case we can show that PD is within 1.1312 of the optimum.

We observe that there are in general several scheduling strategies in PD-order (EPD-order). Although any such strategy maximizes the expected profit, we will show that some optimal strategies have smaller variance than others. Variance analysis is based on the optimality conditions and on the following

Lemma 4.3 *If π is an optimal permutation that has minimum $Var[p(\pi)]$ among all optimal permutations and $p(\pi_i)Pr[l(J_i) \leq D < l(J_{i+1})] = p(\pi_{i+1})Pr[l(J_{i-1}) + l(\pi_{i+1}) \leq D < l(J_{i+1})] \neq 0$, then $p(\pi_i) \leq p(\pi_{i+1})$.*

Proof Sketch. Since π is optimal, Lemma 4.1 holds. Hence, we can only exchange jobs for which the equality condition holds. Furthermore, $p(\pi)$ is a constant among all optimal permutation, so that minimizing the variance is tantamount to maximizing the second moment $E[p^2(\pi)]$. Therefore, we seek optimality conditions for the problem where $p(j)$ is replaced by $-p^2(j)$, subject to the constraints given by Lemma 4.1. Such optimality conditions are found by an exchange argument and the lemma is proven. $\Box$

It can be seen that a tie breaking procedure for the uniform and exponential distribution is to favor shorter jobs. Indeed, suppose that job i and $i+1$ have the same (exponential) profit density and $p(i) \leq p(i+1)$. Then, $l(i) \leq l(i+1)$. Hence, the optimal strategy that minimizes risk is to arrange jobs in PD-order (EPD-order) and break ties by scheduling shortest jobs first.

Acknowledgments

We are grateful to Brian Davison, Kevin Christian, Samir Khuller, and Xiao-Tong Zhuang for helpful discussions.

References

[AS98] Susanne Albers and Guenter Schmidt. Scheduling with unexpected machine breakdowns. Technical Report MPI-I-98-1-021, Max-Planck-Institut Fuer Informatik, 1998.

[BEY98] Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis.* Cambridge University Press, 1998.

[CB98] Mark Crovella and Paul Barford. The network effects of prefetching. In *Proceedings of IEEE INFOCOM*, 1998.

[CGJ97] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson. Approximation algorithms for bin packing: A survey. In Dorit S. Hochbaum, editor, *Approximation Algorithms for NP-hard Problems*, chapter 2. PWS, Boston, 1997.

[DP82] Gianfranco D'Atri and Claude Puech. Probabilistic analysis of the subset-sum problem. *Discrete Appl. Math.*, 4(4):329–334, 1982.

[FCJ99] Li Fan, Pei Cao, and Quinn Jacobson. Web prefetching between low-bandwidth clients and proxies: Potential and performance. In *Proceedings ACM SIGMETRICS '99*, pages 178–187, 1999.

[GGK+97] Juan A. Garay, Inder S. Gopal, Shay Kutten, Yishay Mansour, and Moti Yung. Efficient on-line call control algorithms. *J. Algorithms*, 23(1):180–194, 1997.

[GGU73] M. R. Garey, R. L. Graham, and J. D. Ullman. An analysis of some packing algorithms. In *Combinatorial algorithms (Courant Comput. Sci. Sympos., No. 9, 1972)*, pages 39–47. Algorithmics Press, New York, 1973.

[GMS84] Andrew V. Goldberg and Alberto Marchetti-Spaccamela. On finding the exact solution to a zero-one knapsack problems. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing*, pages 359–368, 1984.

[Gra66] R. L. Graham. Bound for certain multiprocessor anomalies. *Bell System Technical Journal*, 45:1563–1581, 1966.

[Kel99] Hans Kellerer. A polynomial time approximation scheme for the multiple knapsack problem. In *Proceedings of The 2nd International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, 1999.

[KP94] Bala Kalyanasundaram and Kirk Pruhs. Fault-tolerant scheduling. In *Proceedings of the Twentysixth Annual ACM Symposium on the Theory of Computing*, pages 115–124, 1994.

[KP97] Bala Kalyanasundaram and Kirk Pruhs. Fault-tolerant real-time scheduling. In *Algorithms—ESA '97 (Graz)*, pages 296–307. Springer, Berlin, 1997.

[LD99] Vincenzo Liberatore and Brian D. Davison. Data dissemination on the Web: Speculative and unobtrusive. Technical Report 99-23, UMIACS, College Park, MD, April 1999.

[LT94] Richard J. Lipton and Andrew Tomkins. Online interval scheduling. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (Arlington, VA, 1994)*, pages 302–311, New York, 1994. ACM.

[Lue82] George S. Lueker. On the average difference between the solutions to linear and integer knapsack problems. In *Applied probability—computer science: the interface, Vol. I (Boca Raton, Fla., 1981)*, pages 489–504. Birkhäuser Boston, Boston, Mass., 1982.

[Lue95] George S. Lueker. Average-case analysis of off-line and on-line knapsack problems. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 179–188, 1995.

[Mar70] Harry M. Markowitz. *Portfolio Selection. Efficient Diversification of Investments.* Blackwell, New Haven, 1970.

[MRKSV90] M. Meanti, A. H. G. Rinnooy Kan, L. Stougie, and C. Vercellis. A probabilistic analysis of the multiknapsack value function. *Math. Programming*, 46(2 (Ser. A)):237–247, 1990.

[MSV95] A. Marchetti-Spaccamela and C. Vercellis. Stochastic on-line knapsack problems. *Math. Programming*, 68(1, Ser. A):73–104, 1995.

[MT90] Silvano Martello and Paolo Toth. *Knapsack problems.* John Wiley & Sons Ltd., Chichester, 1990.

[SL87] Krzysztof Szkatuła and Marek Libura. On probabilistic properties of greedy-like algorithms for the binary knapsack problem. In *Stochastics in combinatorial optimization (Udine, 1986)*, pages 233–254. World Sci. Publishing, Singapore, 1987.

[Smi56] Wayne E. Smith. Various optimizers for single-stage production. *Naval Res. Logist. Quart.*, 3:59–66, 1956.

[UFA98] Tolga Urhan, Michael Franklin, and Laurent Amsaleg. Cost-based query scrambling for initial delays. In *ACM SIGMOD Intl. Conference on Management of Data (SIGMOD)*, 1998.

[WP98] Walter Willinger and Vern Paxson. Where Mathematics meets the Internet. *Notices of the AMS*, 45(8):961–970, September 1998.

A Randomized Algorithms

We now give the proofs relative to the competitiveness analysis of randomized algorithms.

Lemma A.1 *No randomized algorithm for the knapsack game can have a competitive ratio better than*

$$\Omega\left(L\left(1-\frac{1}{V^{\frac{1}{L-1}}}\right)\right) .$$

Proof. The number of jobs is $n = L$. Moreover, for any $\epsilon > 0$, there is a $p(1)$ such that $p(i)/p(1) \leq \rho^{1-i} \leq (1+\epsilon)p(i)/p(1)$. The adversary chooses D according to the distribution:

$$Pr[D=t] = \begin{cases} \frac{1-\rho}{c} & \text{if } t = l(i) \text{ for } i \in [n-1] \\ \frac{1}{c} & \text{if } t = l(n) \\ 0 & \text{otherwise} \end{cases}$$

Notice that the probabilities sum up to $(1+(n-1)(1-\rho))/c = 1$. Let G_h be any deterministic strategy that schedules job h first, and let $p(G_h)$ be its gain. If $D < l(h)$, then $p(G_h) = 0$. If $D = l(i) > l(h)$, then $p(G_h) = p(h) \leq p(1)\rho^{1-h}$, and its payoff is $p(G_h)/p^* \leq (1+\epsilon)\rho^{i-h}$. The expected payoff for G_n is

$$E\left[\frac{p(G_n)}{p^*}\right] \leq (1+\epsilon)Pr[D = l(n)] = \frac{1+\epsilon}{c} .$$

Let $q_i \stackrel{\text{def}}{=} Pr[D \geq l(i)]$. The expected payoff for G_h, when $1 \leq h \leq n-1$, is

$$\begin{aligned} E\left[\frac{p(G_h)}{p^*}\right] &\leq (1+\epsilon)\left(\sum_{i=h}^{n}\rho^{i-h}q_i\right) = \\ &= (1+\epsilon)\left(q_h + \rho\left(\sum_{i=1}^{n-h}\rho^i q_{h+i}\right)\right) = \\ &= (1+\epsilon)\left(q_n + \rho E\left[\frac{v(G_{h+1})}{v^*}\right]\right) = \\ &= (1+\epsilon)\left(\frac{1-\rho}{c} + \frac{\rho}{c}\right) = \\ &= \frac{1+\epsilon}{c} . \end{aligned}$$

The inverse of $E[v(G_h)/v^*]$ is $\Omega(c)$, and the lemma is proven. □

The most significant difference between our worst-case instance and the *on-line marriage problem* [LT94] is that in shut-down scheduling, the number of jobs n is known in advance, whereas in the on-line marriage problem no such information is available. We will say that the worst-case example in the previous lemma is the *finite on-line marriage problem.*

Proof of Lemma 3.1. We use the notation and result in the previous lemma. First, assume that $V = \Omega(2^L)$. Then,

$$\rho = V^{-\frac{1}{L-1}} \leq 2^{-\frac{L}{L-1}} \leq \frac{1}{2} .$$

and so $c \geq L/2$, and the first part of the theorem is proved.

Next, assume that $L = \omega(\log V)$. Let $x = V^{\frac{1}{L-1}} - 1$ and observe that $\ln(1+x) = (\ln V)/(L-1)$, so that $\lim_{V\to\infty} \ln(1+x) = 0$, which implies $\lim_{V\to\infty} x = 0$. Hence,

$$\begin{aligned}
\lim_{V\to\infty} \frac{c}{\ln V} &= \lim_{V\to\infty} \frac{L(1-\rho)}{\ln V} = \\
&= \lim_{V\to\infty} \frac{L}{L-1} \frac{L-1}{\ln V} \left(1 - \frac{1}{V^{\frac{1}{L-1}}}\right) = \\
&= \lim_{x\to 0} \frac{x}{(1+x)\ln(1+x)} = \\
&= {}^* \lim_{x\to 0} \frac{1}{1+\ln(1+x)} = 1 ,
\end{aligned}$$

and so the competitive ratio is $\Omega(\log V)$, which completes the proof. □

We illustrate the general CRS paradigm in the case of $m = 1$ shut-down scheduling. The job set into k disjoint classes $C_1, C_2, \ldots, C_k$ according to some criteria. then, we will provide deterministic strategies $G_1, G_2, \ldots, G_k$ that schedules only jobs from class C_i and that within a constant factor from the optimum. We will obtain a randomized algorithm chooses one of the G_i's at random and is k-competitive. We now detail this insight. Let $C_1, C_2, \ldots, C_k$ be a partition of $[n]$. If π is a job permutation, we define $p_i(\pi, D)$ as the profit of π from jobs in class C_i that are completed before the deadline, that is, $p_i(\pi, D) = p(J_s \cap C_i)$ where s is the largest integer such that $l(J_s) \le D$. Analogously, $p_i^*(D)$ is the profit of the optimum schedule on jobs in class C_i. Suppose that for all i's there is a deterministic algorithm G_i that schedules only jobs from class C_i and such that $p_i(G_i, D) = p_i^*/\alpha$ for some α. Let G the algorithm that extracts one of the G_i's uniformly at random. Then, G's expected profit is

$$E[p(G)] = \frac{1}{k}\sum_{i=1}^{k} p(G_i) \ge \frac{1}{k}\sum_{i=1}^{k} p_i(G_i) \ge \frac{1}{\alpha k}\sum_{i=1}^{k} p_i^* = \frac{p^*}{\alpha k} . \quad (3)$$

We conclude that G is $O(\alpha k)$-competitive.

We now turn to describe an on-line algorithm for the general case that matches the lower bound up to a constant factor. First, we present an algorithm CRSlog that is $O(\log V)$-competitive. Partition the job set into $\lceil \ln V + 1 \rceil$ *profit classes* $C_i = \{j : e^i \le p(j)/p(1) < e^{i+1}\}$ $(0 \le i \le \lceil \ln V \rceil)$ such that no job is more than e times as profitable as any other job in the same class. Choose one of the classes at random and schedule jobs only from that class on m machines according to the canonical scheduling algorithm.

Lemma A.2 *The CRSlog algorithm is $O(\log V)$-competitive for the shut-down scheduling problem.*

Proof. Let p_i be the profit earned by CRSlog if class i is chosen and p_i^* be the profit earned by the optimum on jobs in class i. Then, the optimum profit is $p^* = \sum_{i=0}^{\ln V} p_i^*$. Suppose that CRSlog chooses class C_h. Since CRSlog schedules jobs in canonical order, Lemma 3.3 implies that no other algorithm completes more than five times as many jobs in C_h as CRSlog, and on each one of these jobs, it earns at most e times the profit gained by CRSlog. Hence, the expected profit of CRSlog is

$$\frac{1}{1+\ln V}\sum_{i=0}^{\ln V} p_i \ge \frac{1}{1+\ln V}\sum_{i=0}^{\ln V} \frac{p_i^*}{5e} = \frac{p^*}{5e(1+\ln V)} ,$$

and we conclude that CRSlog is $O(\log V)$-competitive. $\square$

Define the CRSL algorithm as the algorithm that partitions the job set into length classes where all jobs in the same class have the same length, chooses one class at random, and schedules the jobs in that class in non-increasing order of profit.

Lemma A.3 *The CRSL algorithm is $O(L)$-competitive for the shut-down scheduling problem.*

Proof. Let p_i be the profit earned by CRSL if class i is chosen and p_i^* be the profit earned by the optimum on jobs in class i. Then, the optimum profit is $p^* = \sum_{i=0}^{\ln V} p_i^*$. Suppose that CRSL chooses class C_h. Since CRSL schedules jobs from the most to the least profitable, no other algorithm can achieve a better profit on jobs in the same class. Hence,

$$\frac{1}{L}\sum_{i=1}^{L} p_i \geq \frac{1}{L}\sum_{i=1}^{L} p_i^* = \frac{p^*}{L} ,$$

so that CRSL is L-competitive. $\square$

B Probabilistic Analysis

Proof. We will prove the lemma for the identity permutation for simplicity of notation. Job renaming will yield the result for an arbitrary permutation π. Then, we need to show that, for all $i \in [n-1]$,

$$p(i)Pr[l([i]) \leq D < l([i+1])] \geq p(i+1)Pr[l([i-1]) + l(i+1) \leq D < l([i+1])] .$$

Suppose by contradiction that the equation above does not hold for an index $i \in [n-1]$. Consider a permutation π that is obtained from the identity by exchanging job i and $i+1$. All terms in (1) remain unchanged except those relative to indices i and $i+1$. Moreover,

$$\begin{aligned}
&p(i)Pr[D \geq l([i])] + p(i+1)Pr[D \geq l([i+1])] = \\
&= p(i)Pr[l([i]) \leq D < l([i+1])] + p(i)Pr[D \geq l([i+1])] + p(i+1)Pr[D \geq l([i+1])] < \\
&< p(i+1)Pr[l([i-1]) + l(i+1) \leq D < l([i+1])] + p(i)Pr[D \geq l([i+1])] + \\
&+ p(i+1)Pr[D \geq l([i+1])] = \\
&= p(i+1)Pr[D \geq l([i-1]) + l(i+1)] + p(i)Pr[D \geq l([i+1])] .
\end{aligned} \tag{4}$$

so that the identity's expected profit is less than π's It follows that the identity is not optimal. We have reached a contradiction and the first part of the lemma is proven. Suppose now that equality holds and repeat the same calculations as in equation (4) with an inequality substituted by an equality. The expected profit remains unchanged, and so the second part of the lemma is proven as well. $\square$

Proof. If all value densities are equal, then we can assume without loss of generality that $p(i) = l(i)$ for all jobs i. Moreover, the function $x/(e^x - 1)$ is decreasing for all $x > 0$, so that longer jobs have lower exponential value density. The corollary is then proved by invoking the previous proposition. $\square$

We now compare SF and EPD. An advantage of SF over EPD is that SF does not require the knowledge of the rate λ. However, SF requires that all densities be equal. We next prove that PD is within a constant factor of the optimum independently of value densities as long as the rate λ is large enough.

Corollary B.1 *If D follows an exponential distribution with rate λ and $1/\lambda \geq l([n])$, then PD is within* 1.1312 *of the optimum value of (2).*

Proof. We approximate the function e^x with a linear function $e^x \simeq e^a(x+1-a)$, where $a = -1/(e-1)$. Elementary calculus yields that for all $x \in [-1, 0]$ and for $\epsilon = 1/e^a(1-a) - 1 \leq 0.1312$,

$$e^x \leq (1+\epsilon)e^a(x+1-a) \leq (1+\epsilon)e^x \ .$$

Notice that $(1+\epsilon)e^a(1-a) = 1$ and that $(1+\epsilon)e^a(2-a) > 0$, so that the linear approximation $(1+\epsilon)e^a(x+1-a)$ can be expressed as $1 - At$ for some $A > 0$ and all $0 \leq t \leq l([n])$. The result on uniform distributions implies that PD is optimal under such linear approximation. Assume without loss of generality that the identity permutation is in PD-order, and let π be the optimal permutation for the original exponential distribution. Then,

$$\sum_{i=1}^{n} p(i)(1 - Al([i])) \geq \sum_{i=1}^{n} p(\pi_i)(1 - Al(J_i)) \geq \frac{\sum_{i=1}^{n} p(\pi_i) e^{-\lambda l(J_i)}}{1+\epsilon} \ ,$$

so that the value of PD is no more than $1+\epsilon$ times the optimum, and the proposition is proved. □

Proof of Lemma 4.3. Notice that $Var[p(\pi)] = E[p^2(\pi)] - (E[p(\pi)])^2$ and $E[p(\pi)]$ is fixed to its optimal value, so that minimizing the variance corresponds to minimizing the second moment $E[p^2(\pi)]$ subject to the additional constraint that $E[p(\pi)]$ be optimum. In other words, we seek to minimize $E[p^2(\pi)]$ among all permutations We will prove the lemma for the identity permutation for simplicity of notation. Job renaming will yield the result for all permutations. Exchange job i and $i+1$. The resulting permutation is optimal by Lemma 4.1. The second moment $E[p^2(\pi)]$ remains unchanged except for the terms relative to index i and $i+1$. The difference is

$$\begin{aligned} 0 \ &\leq \ p^2(i)Pr[D \geq l(J_i)] + p^2(i+1)Pr[D \geq l(J_{i+1})] - \\ &\quad -p^2(i)Pr[D \geq l(J_{i+1})] - p^2(i+1)Pr[D \geq l(J_{i-1}) + l(\pi_{i+1})] = \\ &= \ p^2(i)Pr[l(J_i) \leq D < l(J_{i+1})] + p^2(i+1)Pr[l(J_{i-1}) + l(i+1) \leq D < l(J_{i+1})] \ , \end{aligned}$$

so that $p^2(i)Pr[l(J_i) \leq D < l(J_{i+1})] \geq p^2(i+1)Pr[l(J_{i-1}) + l(i+1) \leq D < l(J_{i+1})]$. Divide by $p(\pi_i)Pr[l(J_i) \leq D < l(J_{i+1})] = p(\pi_{i+1})Pr[l(J_{i-1}) + l(\pi_{i+1}) \leq D < l(J_{i+1})]$ to prove the proposition. □