

Utility-based Machine Scheduling with Applications to Object Transmission in Communication Networks

Ye Xia

University of Florida

301 CSE Building, P.O. Box 116120

Gainesville, Florida 32611-6120

Abstract

The motivation for this study is to optimize the downloading process of files or other information objects when the communication capacity is limited. Consider a scenario where a user (or users) is downloading a collection of objects, each of which has certain utility to the user. Suppose the utilities of the objects are non-increasing functions of the completion (i.e., arrival) times of the objects. The problem is to schedule the transmission of the objects so that the total utility received by the user is maximized. In this paper, we examine variations of this problem with different utility functions. We present some new results on single machine scheduling that are relevant to data communication scenarios that involve the transmission of a collection of information objects.

keyword single machine scheduling, utility function, shortest processing time schedule, object scheduling, optimization

1 Introduction

This paper addresses the question of how to optimize the satisfaction of a user or users when they download files or data objects from a computer server. A single-user example is web browsing. By web objects, we mean items on a web page, such as a piece of text, an image, an audio or video clip, etc. We do not have to limit the notion of web objects to those items with complete semantic boundaries, such as a complete JPEG image. A block of a JPEG image, or a low-resolution copy of a multi-resolution-encoded JPEG image can also be considered an object. The defining property of an object is that, as a whole, it has some utility to the

user and any part of it either has no utility or cannot be rendered by the application. A multi-user example is when a number of users make simultaneous requests to retrieve different objects or files from a server, where each user places a value to the object he is retrieving.

Both the single-user and multi-user applications will be formulated as the same abstract problem. We will focus on the single-user case. More precisely, consider a downloading session in which a user wishes to retrieve n objects. The rate of communication between the sever and the user is a constant, μ , possibly limited at a slow link in the network or at the busy server. Suppose the user assigns certain value to each object, which is a decreasing function of the object arrival (completion) time. The question is how the server should schedule the transmission of these n objects in order to maximize the total value received by the user.

The problem stated here belongs to the general area of single machine scheduling. This paper presents some results in this area that are relevant to our problem. Due to the online nature of the problem, we pay particular attention to the time complexity of the scheduling algorithms. We discuss specific requirements and characteristics of data object transmission and make recommendations toward solving this problem. Single machine scheduling is a mature area with many known results that can be applied to the problem of retrieving files other information objects. We will mention some of the known results in this paper.

The paper is organized as follows. In the remaining part of this section, we formulate the scheduling problem of object transmission as a utility optimization problem and describe a few families of objective functions. In Section 2, we discuss known results related to some of these families of objective functions. In Section 3, we give our results to the remaining families of functions. In section 4, we introduce more results for the linear utility functions under time-varying bandwidth and under random transmission times. In section 5 and 6, we discuss two related problem formulations. We conclude in section 7.

1.1 Basic Problem Formulation

Before dealing with the question of how to optimize the user's satisfaction, we need to consider how the satisfaction is expressed. We assume that each object has certain utility to the user. Since an object cannot be used until it arrives at the user completely, its utility should depend on the completion time and can be denoted by $v_i(C_i)$, where C_i is the completion time of object i , i.e., the arrival time of object i at the user. The total utility of all objects to the user becomes,

$$v(C_1, C_2, \dots, C_n) = \sum_{i=1}^n v_i(C_i) \quad (1)$$

¹ Since one would expect that the user is more satisfied if the transmission is completed sooner, the function $v_i(C_i)$ should be non-increasing. The resulting total utility function, $\sum_{i=1}^n v_i(C_i)$, belongs to the class of regular measures (objective functions). A measure is called *regular* if it is non-increasing with respect to each C_i (when the measure represents a utility function to be maximized). It is known that for regular measures, optimality can be achieved by non-preemptive schedules (See [1] and [4]). A corollary is that processor sharing does not make further improvement. Hence, the scheduling problem is to find a transmission order of the objects so that (1) is maximized.

On the other hand, if the object can be displayed progressively in very fine granularity, we can idealize the situation by assuming every infinitesimal piece of data is useful. Then, the utility of object i can be denoted by the function $v_i(t, x)$, which represents the value received by the user when x bits have arrived by time t . In this case, a preemptive schedule may be required to achieve optimality. This formulation will not be the focus of the paper.

Traditionally, single machine scheduling problems are formulated as minimizing some particular cost functions, such as the *mean or weighted completion time*, *mean or weighted tardiness*, and *number of tardy jobs*. Most of these objective functions have the same separable form as on the right hand side of (1). One of the contributions of this paper is to investigate the optimal schedules corresponding to some new utility functions, v_i , not previously considered in the traditional machine scheduling literature. For easy reference, we list the notations and definitions used in the paper.

s_i The size of object i .

p_i The transmission time of object i . $p_i = s_i/\mu$. Without loss of generality, we assume throughout the paper that $p_i > 0$ for every object i .

C_i The completion time of object i , i.e., its arrival time at the user.

d_i The deadline (or due date) of object i , i.e., the expected completion time for object i .

T_i The *tardiness* of object i . $T_i = \max\{C_i - d_i, 0\}$.

1.2 Several Regular Measures and Their Optimal Schedules

We will consider several families of the function v_i , for a generic object i . These functions are summarized in Table 1. In most cases, the measures are interpreted as utility functions that are to be maximized.

¹In the multi-user scenario where each user downloads a single object, v_i can be understood as the utility of user i 's object.

Occasionally, we relate them to their corresponding cost functions used in the machine scheduling literature.

Table 1: Objective Functions

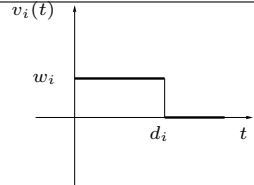
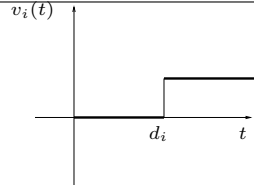
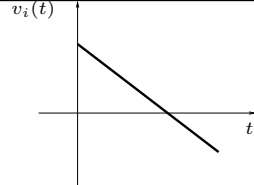
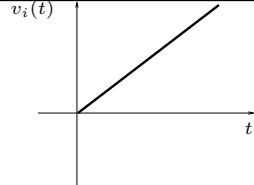
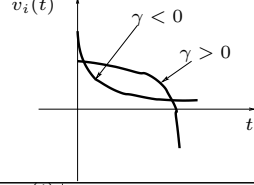
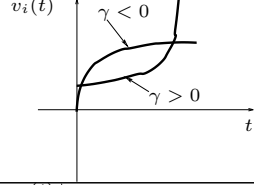
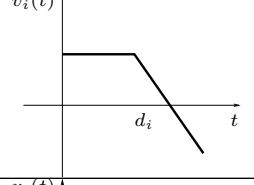
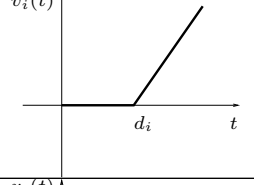
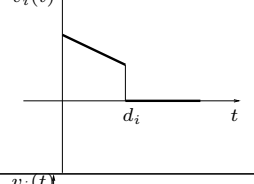
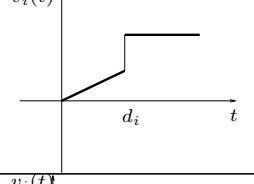
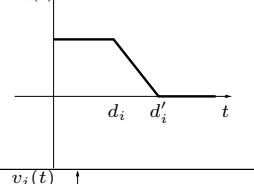
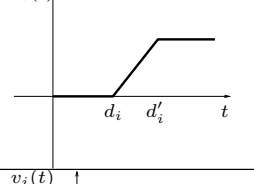
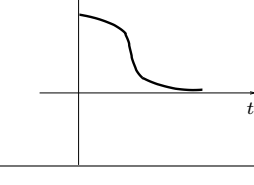
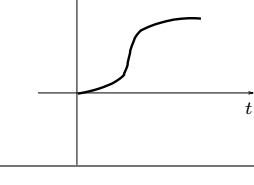
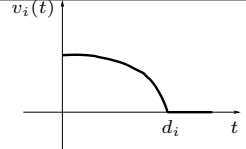
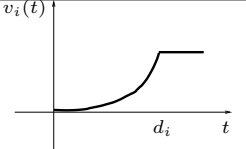
	Function Expression	Utility	Cost
I	$v_i(t) = \begin{cases} w_i & \text{if } 0 \leq t \leq d_i \\ 0 & \text{otherwise} \end{cases}$		
II	$v_i(t) = \alpha_i t + \beta_i$		
III	$v_i(t) = \alpha_i e^{\gamma t} + \beta_i$		
IV	$v_i(t) = \begin{cases} w_i & \text{if } 0 \leq t \leq d_i \\ \alpha_i t + \beta_i & \text{otherwise} \end{cases}$		
V	$v_i(t) = \begin{cases} \alpha_i t + \beta_i & \text{if } 0 \leq t \leq d_i \\ 0 & \text{otherwise} \end{cases}$		
VI	$v_i(t) = \begin{cases} w_i & \text{if } 0 \leq t \leq d_i \\ \alpha_i t + \beta_i & \text{if } d_i \leq t \leq d'_i \\ 0 & \text{otherwise} \end{cases}$		
VII	$v_i(t) = \beta_i - \frac{\beta_i}{1 + \alpha_i \exp(-\gamma t)}$		

Table 1: Continued

	Function Expression	Utility	Cost
VIII	$v_i(t) = \begin{cases} \alpha_i e^{\gamma t} + \beta_i & \text{if } 0 \leq t \leq d_i \\ 0 & \text{otherwise} \end{cases}$		

2 Known Results Relevant to Object Transmission

2.1 Function I - Step Functions

In this case,

$$v_i(t) = \begin{cases} w_i & \text{if } 0 \leq t \leq d_i \\ 0 & \text{otherwise} \end{cases}$$

where w_i 's are positive numbers, representing the values of the objects. In this model, a utility w_i is gained if the object is received before the deadline d_i . Otherwise, the object becomes useless. A minimization version of the problem is as follows.

$$v_i(t) = \begin{cases} 0 & \text{if } 0 \leq t \leq d_i \\ w_i & \text{otherwise} \end{cases} \quad (2)$$

where w_i 's are also positive numbers, representing a penalty when an object is completed after its deadline. The corresponding scheduling problem is proved to be NP-hard in [7]. A pseudo-polynomial solution based on dynamic programming is found for the minimization problem by Lawler and Moore [9], which has a complexity $O(nT)$, where $T = \sum_{i=1}^n p_i$. The technique can be naturally extended to the maximization problem. Fully polynomial time approximation algorithms are found in [13] for the maximization version of the problem with time complexity $O(n^2/\epsilon)$, and in [5] for the minimization version of the problem with time complexity $O(n^2 \log n + n^2/\epsilon)$.

We will introduce the algorithm by Lawler and Moore [9] because the technique also serves as basis for some other more difficult problems in the paper. Consider the following general formulation. Suppose n jobs are to be processed in the *fixed* order, 1, 2, ..., n . Each job can be performed in either one of two different modes. In the first mode, the processing time of job j is a_j^1 and the cost function is $\rho_j^1(t)$. In the

second mode, the processing time of job j is a_j^2 and the cost function is $\rho_j^2(t)$. The objective is to assign one mode to each object so that the total cost is minimized. Let $f(j, t)$ be the minimum total cost for the first j jobs, subject to the constraint that job j is completed no later than time t . The dynamic programming solution for this problem is listed in Algorithm 1. The assignment problem is solved by computing $f(n, T)$, where T is a sufficiently large number. For instance, we can choose $T = \sum_{i=1}^n \max\{a_i^1, a_i^2\}$ when the cost functions are regular. The computation requirement is $O(nT)$.

Algorithm 1 Lawler and Moore

$$\begin{aligned}
 f(0, t) &= 0 && (t \geq 0), \\
 f(j, t) &= +\infty && (j = 0, 1, \dots, n; t < 0), \\
 f(j, t) &= \min \left\{ \begin{array}{l} f(j, t - 1) \\ \rho_j^1(t) + f(j - 1, t - a_j^1) \\ \rho_j^2(t) + f(j - 1, t - a_j^2) \end{array} \right\} && (j = 1, 2, \dots, n; t \geq 0)
 \end{aligned}$$

Going back to the object sequencing problem that minimizes the total cost $\sum_{i=1}^n v_i(C_i)$, where v_i is the step function as in (2), we only need to partition the objects into two groups: those that are completed before their deadlines and those that are tardy. In the actual schedule, all tardy objects simply follow those on-time objects in arbitrary order among themselves. Given an optimal sequence, we can always order those objects that are on time by the earliest-due-date (EDD) schedule. These objects will still be completed before their deadlines in the resulting new order. Hence, an algorithm for finding an optimal schedule is as follows. First, order the n objects by the EDD schedule. Without loss of generality, we assume this order is $1, 2, \dots, n$. For each object j , if it is in the first mode, let,

$$\begin{aligned}
 a_j^1 &= p_j \\
 \rho_j^1(t) &= \begin{cases} 0 & \text{if } 0 \leq t \leq d_j \\ \infty & \text{otherwise} \end{cases}
 \end{aligned}$$

If it is in the second mode, let

$$\begin{aligned}
 a_j^2 &= 0 \\
 \rho_j^2(t) &= w_j
 \end{aligned}$$

Then, apply Lawler and Moore's algorithm. The objects with the second mode assignment are tardy ones.

2.2 Function II - Linear Functions

In this case,

$$v_i(t) = \alpha_i t + \beta_i$$

When v_i is interpreted as the utility function, we require $\alpha_i < 0$, and $\beta_i \geq 0$ for all i . That is, the value of the object decreases linearly with the completion time. The corresponding minimization version, where $\alpha_i > 0$ for all i , is recognized as minimizing weighted completion times. For both problems, the optimal schedule is to transmit the objects in increasing order of $p_i/|\alpha_i|$, which has a complexity $O(n \log n)$. Notice that β_i 's play no role in determining the optimal sequencing. In fact, as long as all objects have to be transmitted, the β_i 's contribute the same constant term in any permutation of the n objects. The proof of optimality of the schedule is the standard *adjacent-pair-interchange* argument. One possible drawback with function II is that it decreases indefinitely with t . Jobs that are completed late are penalized when v_i becomes negative. In the context of object transmission, it may be more reasonable to assume that a late object has zero or a small positive value to the user instead of a negative value. We will examine the linear utility function again in Section 4.

2.3 Function IV

$$v_i(t) = \begin{cases} w_i & \text{if } 0 \leq t \leq d_i \\ \alpha_i t + \beta_i & \text{otherwise} \end{cases}$$

In order to interpret v_i as the utility function for object i , we require $\alpha_i \leq 0$ and $\alpha_i d_i + \beta_i = w_i$. However, w_i 's have no influence on the optimal schedule.

It is more convenient to discuss the minimization version of the problem, which minimizes total weighted tardiness, $\sum_{i=1}^n \alpha_i T_i$. In this case, the weights α_i are positive and can be interpreted as the prices to pay per unit of time for completing the object late. The problem is proved to be NP-hard in the strong sense in [11] and [8]. If all weights are equal, the problem is NP-hard in the ordinary sense [6] and can be solved in pseudo-polynomial time by Lawler and Moore's algorithm (1) [8]. If all deadlines are equal and the weights are arbitrary, the problem is NP-hard in the ordinary sense [17], and can also be solved by Lawler and Moore's algorithm (1). If precedence constraints among the objects are present, the problem is NP-hard even with equal deadlines and equal weights [10]. Researchers have developed branch-and-bound and

integer programming techniques for solving this type of scheduling problems, as well as many heuristic techniques for finding approximate solutions. Many general textbooks on scheduling discuss these problems, e.g. [15] and [12].

3 New Results Related to Object Transmission

3.1 Function III - Exponential Functions

$$v_i(t) = \alpha_i e^{\gamma t} + \beta_i$$

where $v_i(t)$ is interpreted as a utility function and decreases with t . We can have two types of decaying exponential functions corresponding to $\gamma < 0$ and $\gamma > 0$. In order for the function to represent the “value” of an object, when $\gamma < 0$, we require $\alpha_i > 0$ and $\beta_i \geq 0$ such that the function remains positive for all $t \geq 0$. If $\beta_i = 0$, α_i is the initial value of the object and $1/|\gamma|$ can be interpreted as a “soft” deadline. When $\gamma > 0$, we require $\alpha_i < 0$ and $\beta_i + \alpha_i > 0$ in order to have an appropriate utility function. One should be cautious in this case, since the value of v_i becomes negative and rapidly decreases when t becomes sufficiently large. Such v_i may be appropriate when the deadline needs to be rigorously enforced. One potential drawback of the exponential function is that a single parameter γ is used for all objects.

Theorem 3.1 *The optimal schedule is to sort the n objects in decreasing order of $(\alpha_i e^{\gamma p_i}) / (1 - e^{\gamma p_i})$.*

Proof: We use the adjacent-pair-interchange argument. Suppose the optimal sequence is $\pi = (1, 2, \dots, n)$. Consider object i and $i + 1$. In the optimal schedule, object i starts to be transmitted at time C_{i-1} , where we assume $C_0 = 0$. Starting with the optimal sequence, switching the position of object i and $i + 1$ decreases the utility of object i by

$$(\alpha_i e^{\gamma(C_{i-1} + p_i)} + \beta_i) - (\alpha_i e^{\gamma(C_{i-1} + p_{i+1} + p_i)} + \beta_i) = \alpha_i e^{\gamma C_{i-1}} e^{\gamma p_i} (1 - e^{\gamma p_{i+1}})$$

and increases the utility of object $i + 1$ by

$$(\alpha_{i+1} e^{\gamma(C_{i-1} + p_{i+1})} + \beta_{i+1}) - (\alpha_{i+1} e^{\gamma(C_{i-1} + p_i + p_{i+1})} + \beta_{i+1}) = \alpha_{i+1} e^{\gamma C_{i-1}} e^{\gamma p_{i+1}} (1 - e^{\gamma p_i})$$

Since the sequence π is optimal, the total change of utility should be non-positive. Therefore,

$$-\alpha_i e^{\gamma C_{i-1}} e^{\gamma p_i} (1 - e^{\gamma p_{i+1}}) + \alpha_{i+1} e^{\gamma C_{i-1}} e^{\gamma p_{i+1}} (1 - e^{\gamma p_i}) \leq 0$$

Equivalently,

$$\frac{\alpha_i e^{\gamma p_i}}{1 - e^{\gamma p_i}} \geq \frac{\alpha_{i+1} e^{\gamma p_{i+1}}}{1 - e^{\gamma p_{i+1}}} \quad (3)$$

■

Notice again that β_i 's play no role in determining the optimal schedule.

Remark In the cases of function II and III, the position of an object in the optimal schedule depends only on some function of the object's own parameters. This type of function is called a priority-generating function in [16].

3.2 Function V

$$v_i(t) = \begin{cases} \alpha_i t + \beta_i & \text{if } 0 \leq t \leq d_i \\ 0 & \text{otherwise} \end{cases}$$

When considered as a utility function, we require $\alpha_i \leq 0$, and $\beta_i > 0$ for all i . We also require $\alpha_i d_i + \beta_i \geq 0$ so that the function is non-increasing. This function can be viewed as a compromise between the step function I and the linear function II. We will prove the following new result.

Theorem 3.2 *The scheduling problem is NP-hard in the strong sense.*

Two other problems appear to be related to the current problem, the $n|1||\sum \alpha_i T_i$ and $n|1|C_i \leq d_i|\sum \alpha_i C_i$ problems, where α_i 's are non-negative weights. The standard machine scheduling notation is used here. The first problem stands for n jobs, single machine and minimizing *total weighted tardiness*. The second problem minimizes *total weighted completion time* subject to the constraint that all jobs are completed before their due dates. As mentioned in the case of function IV, the first problem is NP-hard in the strong sense. So is the second problem, as proved in [16] and suggested in [11]. Notice that the second problem can be considered as having the following extended cost function v_i .

$$v_i(t) = \begin{cases} \alpha_i t & \text{if } 0 \leq t \leq d_i \\ \infty & \text{otherwise} \end{cases}$$

where $\alpha_i \geq 0$.

We now return to our problem. When $\alpha_i \leq 0$ for all i and all objects are required to be completed before their due dates, the problem is NP-hard in the strong sense, because it is equivalent to the second

problem above. We also know that, when $\alpha_i \leq 0$, the problem is at least NP-hard in the ordinary sense. To see this, let $\alpha_i = 0$, and $\beta_i > 0$ for all i , then v_i becomes function I. The proof of strong NP-hardness is accomplished by reducing the 3-partition problem, which is known to be NP-hard in the strong sense, to the current problem. We will adapt the proof for the $n|1||\sum \alpha_i T_i$ problem in [8]. We also work with the “decision” version of the problem rather than the “optimization” version. If the optimization problem has a polynomial-time solution, then the decision problem trivially has a polynomial-time solution. Therefore, to show the optimization problem is NP-hard, it is sufficient to show the decision problem is NP-complete. We will work with the minimization version of the problem. Let us first redefine $v_i(t)$.

$$v_i(t) = \begin{cases} \alpha_i t & \text{if } 0 \leq t \leq d_i \\ w_i & \text{otherwise} \end{cases} \quad (4)$$

where $\alpha_i \geq 0$ and $w_i \geq \alpha_i d_i$.

3-Partition Problem: Given a set of $3n$ integers a_1, a_2, \dots, a_{3n} between 1 and $B - 2$ such that $\sum a_i = nB$. Is there a partition of the a_i 's into groups of 3, each summing to B ?

Scheduling Problem:

“X”-jobs: $X_i, 1 \leq i \leq n.$

“A”-jobs: $A_i, 1 \leq i \leq 3n.$

Processing times: $p(X_i) = L = (16B^2)^{\frac{n(n+1)}{2}} + 1, 1 \leq i \leq n.$

$p(A_i) = B + a_i, 1 \leq i \leq 3n.$

Weights $\alpha(X_i) = 0, 1 \leq i \leq n.$

$\alpha(A_i) = p(A_i) = B + a_i, 1 \leq i \leq 3n.$

Due dates $d(X_i) = iL + (i - 1)4B, 1 \leq i \leq n.$

$d(A_i) = \sum_{i=1}^n p(X_i) + \sum_{i=1}^{3n} p(A_i) + 1, 1 \leq i \leq 3n.$

Constants $w(X_i) = W = (L + 4B)(4B)^{\frac{n(n+1)}{2}} + 1, 1 \leq i \leq n.$

$w(A_i)$ can be any value greater than to equal to $\alpha(A_i)d(A_i), 1 \leq i \leq 3n.$

Question: Is there a schedule π with total cost $R(\pi) \leq W - 1$?

We need to establish that the 3-partition problem has a solution if and only if the scheduling problem

above has a solution.

Proof: Suppose the desired partition exists. Without loss of generality, suppose $\langle a_{3j-2}, a_{3j-1}, a_{3j} \rangle$ is a group, $i \leq j \leq n$. Consider the schedule

$$\pi = \langle X_1, A_1, A_2, A_3, X_2, A_4, A_5, A_6, X_3, \dots, X_i, A_{3i-2}, A_{3i-1}, A_{3i}, \dots, X_n, A_{3n-2}, A_{3n-1}, A_{3n} \rangle$$

Since $\sum_{i=-2}^0 p(A_{3j+i}) = 4B$, for $1 \leq j \leq n$, X_i finishes at time $d(X_i) = iL + (i-1)4B$, for $1 \leq i \leq n$. That is, the X-jobs all finish on time. Note that the common due date for the A-jobs is chosen so that all jobs are completed before the due date in any work-conserving schedule. A_{3j-2}, A_{3j-1} and A_{3j} all finish by $j(L+4B)$ and their total weight is $4B$, $1 \leq j \leq n$. Their total cost is no greater than $j(L+4B)4B$. Therefore,

$$R(\pi) \leq \sum_{j=1}^n j(L+4B)4B = (L+4B)(4B) \frac{n(n+1)}{2} = W - 1 \quad (5)$$

Conversely, suppose there is a schedule π such that $R(\pi) \leq W - 1$. We need to show there is a 3-partition. First, notice that no X-job can be tardy, because the cost contributed by any tardy X-job is W . Next, define W_i to be the total weight of the A-jobs following X_i , with $W_{n+1} = 0$ by convention. The total cost due to the group of A-jobs between X_i and X_{i+1} is no less than $(W_i - W_{i+1})iL$. Hence,

$$R(\pi) \geq \sum_{i=1}^n (W_i - W_{i+1})iL = L \sum_{i=1}^n W_i$$

Note that W_i is also the total processing time for the A-jobs following X_i . The total processing time for the A-jobs preceding X_i is $(4B)n - W_i$. Since all X-jobs meet their due dates, we must have

$$iL + (i-1)4B \geq (4B)n - W_i + iL \quad \text{for } 1 \leq i \leq n$$

Equivalently,

$$W_i \geq (n-i+1)4B \quad \text{for } 1 \leq i \leq n$$

Suppose for some i , $W_i \geq (n-i+1)4B + 1$. Then,

$$\sum_{i=1}^n W_i \geq 1 + \sum_{i=1}^n (n-i+1)4B = \frac{n(n+1)}{2}4B + 1$$

Then,

$$\begin{aligned}
R(\pi) &\geq L\left(\frac{n(n+1)}{2}4B+1\right) \\
&= L\frac{n(n+1)}{2}4B+(16B^2)\frac{n(n+1)}{2}+1 \\
&= (L+4B)(4B)\frac{n(n+1)}{2}+1 \\
&= W
\end{aligned}$$

This contradicts (5). Hence, it must be true that $W_i = (n-i+1)4B$, for $1 \leq i \leq n$. From this we conclude that the total weight for the group of A-jobs between X_i and X_{i+1} must be $4B$ in π , $1 \leq i \leq n-1$. Similarly, the total weight for the group of A-jobs following X_n must also be $4B$. Since every A-job, say A_i for each i , satisfies $B+1 \leq \alpha(A_i) \leq 2B-2$, each such group must contain exactly 3 A-jobs. The n groups of 3 jobs correspond to the desired partition. ■

3.2.1 Case of Common Deadline

When all objects have a common deadline, d , the algorithm of Lawler and Moore (1) gives a pseudo-polynomial solution for finding an optimal sequence. Again, consider the minimization version of the problem. An optimal sequence divides the objects into two groups: those that are completed before the deadline and those that are tardy. The objects that are on time must be ordered in increasing order of p_i/α_i , with the understanding that $p_i/0 = \infty$, and the tardy objects can be arbitrarily ordered. Therefore, to find an optimal schedule, first sort the n objects in this order. Without loss of generality, we assume the order is $1, 2, \dots, n$. For each object i , if it is in the first mode, let

$$\begin{aligned}
a_i^1 &= p_i \\
\rho_i^1(t) &= \begin{cases} \alpha_i t & \text{if } 0 \leq t \leq d \\ \infty & \text{otherwise} \end{cases}
\end{aligned}$$

If it is in the second mode, let

$$\begin{aligned}
a_i^2 &= 0 \\
\rho_i^2(t) &= w_i
\end{aligned}$$

Then apply Algorithm 1.

3.2.2 Case of Continuous Function

For the minimization problem with the cost functions in (4), where $\alpha_i \geq 0$, let us set $\alpha_i d_i = w_i$ for all i . It turns out this continuity requirement makes the problem easier. There is a pseudo-polynomial algorithm for finding an optimal sequence. Let π be an optimal sequence and A_π be the subset of all objects that are on time in π . Then,

Lemma 3.3 *In schedule π , the objects in A_π must be ordered in increasing order of p_i/α_i .*

Proof: First, notice that, in any optimal sequence, all objects in A_π must be contiguous and occupy the first $|A_\pi|$ positions, followed by the tardy objects. Suppose the lemma is not true. There must exist two neighboring objects i and j , $i, j \in A_\pi$, with i preceding j , such that $p_i/\alpha_i > p_j/\alpha_j$. By exchanging i and j , the cost of j decreases by $\alpha_j p_i$, and the cost of i increases by *no more than* $\alpha_i p_j$. But, because $\alpha_j p_i > \alpha_i p_j$, exchanging the positions of i and j reduces the total cost, which contradicts the optimality of π . ■

The optimal sequencing problem can be solved by applying Lawler and Moore's algorithm. Without loss of generality, let us suppose the objects are numbered so that $p_1/\alpha_1 \leq p_2/\alpha_2 \leq \dots \leq p_n/\alpha_n$. Given that the objects are ordered from 1 to n , we need to decide, for each object j , whether it should be completed before d_j . Apply Lawler and Moore's algorithm with the following parameters, for $1 \leq j \leq n$.

$$\begin{aligned} a_j^1 &= p_j \\ \rho_j^1(t) &= \begin{cases} \alpha_j t & \text{if } 0 \leq t \leq d_j \\ \infty & \text{otherwise} \end{cases} \\ a_j^2 &= 0 \\ \rho_j^2(t) &= w_j \end{aligned}$$

3.3 Function VI

$$v_i(t) = \begin{cases} w_i & \text{if } 0 \leq t \leq d_i \\ \alpha_i t + \beta_i & \text{if } d_i \leq t \leq d'_i \\ 0 & \text{otherwise} \end{cases}$$

where $w_i > 0$, $0 \leq d_i \leq d'_i$ and $\alpha_i < 0$. In order to have a non-increasing utility function, we also require $w_i \geq \alpha_i d_i + \beta_i$ and $\alpha_i d'_i + \beta_i \geq 0$. The value of the object starts at a constant until time d_i , decreases linearly on $[d_i, d'_i]$, and becomes zero after d'_i . This function is more versatile in approximating “real” utility functions than function V. Since this function becomes function V when $d_i = 0$ for all i , the scheduling problem is NP-hard in the strong sense. When $d_i > 0$, the problem is still strongly NP-hard because we can reduce any scheduling problem associated with function IV into a problem under function VI by letting the d'_i 's large enough. It remains open to find good approximation algorithms. Two other functions (VII and VIII) can approximate this function when their parameters are properly chosen, as seen from figure 1.

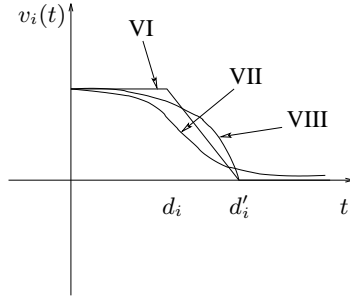


Figure 1: Function VI, VII and VIII

3.4 Function VII

$$v_i(t) = \beta_i - \frac{\beta_i}{1 + \alpha_i \exp(-\gamma t)}$$

where we assume $\gamma > 0$, $\alpha_i \gg 1$ and $\beta_i > 0$. With these constraints on the parameters, v_i is a logistic function reflected against $t = 0$ and vertically shifted. The function has three operating regions. It starts at $v_i(0) = \frac{\beta_i \alpha_i}{1 + \alpha_i} \approx \beta_i$ and decreases gradually for small t . At around $t = \frac{\ln \alpha_i}{\gamma}$ is a transition region where the function decreases to nearly zero. After that, it continues to decrease, tending to zero. This v_i can be interpreted as a smooth version of function I or VI. The time complexity of the associated scheduling problem is unknown. It will be interesting to find an efficient solution or approximation algorithms for this problem.

When $\alpha_i = \alpha$ for all i , we can find a “locally” optimal solution easily. Suppose object i and j are

adjacent in an optimal schedule, π , in that order. Let t be the starting time of object i . Denote

$$\phi(i, t) = \frac{\beta_i e^{-\gamma p_i}}{(1 + \alpha e^{-\gamma(t+p_i)})(1 - e^{-\gamma p_i})}$$

Using the adjacent-pair-interchange argument, one can show that in π ,

$$\phi(i, t) \geq \phi(j, t) \tag{6}$$

Our algorithm starts at time $t_o = 0$. We choose the object with the largest value of $\phi(i, t_o)$ among all n objects to be the first one. Suppose the k^{th} object in our schedule finishes at time t_k . The $(k + 1)^{th}$ object in the schedule should have the largest value of $\phi(i, t_k)$ among the remaining objects. The resulting schedule cannot be improved by exchanging any two neighboring objects.

3.5 Function VIII

$$v_i(t) = \begin{cases} \alpha_i e^{\gamma t} + \beta_i & \text{if } 0 \leq t \leq d_i \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

where we assume $\gamma > 0$, $\alpha_i < 0$, $\alpha_i + \beta_i > 0$, and $\alpha_i e^{\gamma d_i} + \beta_i \geq 0$. With these choices, the utility function first decreases as a concave exponential function until time d_i . After that, its value becomes zero. The complexity of the corresponding scheduling problem is not known. Note that if we allow $\alpha_i \leq 0$ for all i , the problem is NP-hard since we can set $\alpha_i = 0$ for all i and get the step function I. We construct a minimization version of the problem, which is useful later, by defining $\hat{v}_i(t) = \beta_i + \alpha_i - v_i(t)$ as the cost function for object i . That is,

$$\hat{v}_i(t) = \begin{cases} \alpha_i - \alpha_i e^{\gamma t} & \text{if } 0 \leq t \leq d_i \\ \alpha_i + \beta_i & \text{otherwise} \end{cases} \tag{8}$$

The cost function thus defined is non-negative.

3.5.1 Case of Common Deadline

Suppose all objects have the same deadline, i.e., $d_i = d$ for all i . We again use Lawler and Moore's algorithm (1) to find a pseudo-polynomial solution for the scheduling problem associated with the cost functions in (8). An optimal sequence divides the objects into two groups: those that are completed before the deadline and those that are tardy. The objects that are on time must be ordered in decreasing order of $(\alpha_i e^{\gamma p_i}) / (1 - e^{\gamma p_i})$.

Therefore, to find an optimal schedule, first sort the n objects in this order. Without loss of generality, we assume the order is $1, 2, \dots, n$. For each object i , if it is in the first mode, let

$$a_i^1 = p_i$$

$$\rho_i^1(t) = \begin{cases} \alpha_i - \alpha_i e^{\gamma t} & \text{if } 0 \leq t \leq d \\ \infty & \text{otherwise} \end{cases}$$

If it is in the second mode, let

$$a_i^2 = 0$$

$$\rho_i^2(t) = \alpha_i + \beta_i$$

Then apply Algorithm 1. Finally, move the objects assigned to the second mode after those assigned to the first mode.

3.5.2 Case of Continuous Function

For the minimization problem with the cost functions in (8), where $\alpha_i > 0$, let us set $\alpha_i e^{\gamma d_i} + \beta_i = 0$ for all i , and hence, $\hat{v}_i(t)$ is continuous at d_i for each i . There is a pseudo-polynomial algorithm for finding an optimal sequence. Let π be an optimal sequence and A_π be the subset of all objects that are on time in π . Then,

Lemma 3.4 *In schedule π , the objects in A_π must be ordered in decreasing order of $\alpha_i e^{\gamma p_i} / (1 - e^{\gamma p_i})$.*

Proof: First, notice that, in any optimal sequence, all objects in A_π must be contiguous and occupy the first $|A_\pi|$ positions, followed by the tardy objects. Suppose the lemma is not true. There must exist two neighboring objects i and j , $i, j \in A_\pi$, with i preceding j , such that

$$\frac{\alpha_i e^{\gamma p_i}}{1 - e^{\gamma p_i}} < \frac{\alpha_j e^{\gamma p_j}}{1 - e^{\gamma p_j}} \quad (9)$$

By exchanging i and j , the cost of j decreases by $|\alpha_j| e^{\gamma(\tau+p_i+p_j)} - |\alpha_j| e^{\gamma(\tau+p_j)}$, where τ denotes the start time of service for object i in the schedule. The cost of i increases by *no more than* $|\alpha_i| e^{\gamma(\tau+p_i+p_j)} - |\alpha_i| e^{\gamma(\tau+p_i)}$. Because of (9), exchanging the positions of i and j would reduce the total cost, which contradicts the optimality of π . ■

The optimal sequencing problem can be solved by applying Lawler and Moore's algorithm. Without loss of generality, let us suppose the objects $1, \dots, n$ are ordered in decreasing order of $\alpha_i e^{\gamma p_i} / (1 - e^{\gamma p_i})$. We need to decide, for each object i , whether it should be completed before d_i . Apply Lawler and Moore's algorithm with the following parameters, for $1 \leq i \leq n$.

$$\begin{aligned}
 a_i^1 &= p_i \\
 \rho_i^1(t) &= \begin{cases} \alpha_i - \alpha_i e^{\gamma t} & \text{if } 0 \leq t \leq d_i \\ \infty & \text{otherwise} \end{cases} \\
 a_i^2 &= 0 \\
 \rho_i^2(t) &= \alpha_i + \beta_i
 \end{aligned}$$

4 Linear Utility Function

In this section, we examine the linear utility function in more complex but practical situations and show the simplicity of the optimal schedules. We will also evaluate the performance of the shortest-processing-time (SPT) schedule, which corresponds to the case where the utility functions for all objects have the same slope.

The processing capacity (or transmission bandwidth) is determined jointly by the server's capacity and the transmission bandwidth in the network path from the server to the user. It can vary due to many factors, such as variation of cross-traffic load in the path, variation of server load, the congestion control algorithm, and retransmission of lost packets. The capacity fluctuates on different time scales, depending on the causes. We may either model it as a time-varying but deterministic quantity or a random quantity.

4.1 Linear Utility Function and Time-Varying Bandwidth

Suppose the bandwidth is deterministic but time-varying and piece-wise continuous, denoted by $\mu(t)$, and suppose the sizes of the n objects are fixed. It is still true that, for a regular utility function, there exists an optimal schedule that is (i) work-conserving and (ii) non-preemptive. (iii) The optimal sequence in general depends on the bandwidth trajectory. For many utility functions, the scheduling problem becomes very hard. (iv) For the linear utility functions with the identical slope, an optimal schedule is to sequence the objects by their sizes in increasing order, which is independent of the bandwidth trajectory. We will give justifications to these claims.

Suppose that in an optimal schedule, the server has a period of inactivity from time t_1 to t_2 . Then, eliminating the inactive gap by moving ahead all objects scheduled after time t_2 does not decrease the utility, since the utility function is non-increasing in the reception times. This proves (i).

Now, suppose π is a work-conserving optimal schedule, in which object i is preempted by other objects before its completion. Let t_f be the completion time of object i in π and let s_i be the size of object i . Define

$$t^* = \sup\{t : \int_t^{t_f} \mu(\tau) d\tau = s_i\}$$

Let π^* be the new schedule in which object i starts service at time t^* and ends at t_f . Furthermore, to obtain π^* from π , all service times assigned to i in π prior to t^* are removed, and the resulting gaps are filled by moving ahead (time-shifting) the complete or partial objects, other than i , that follow the gaps but before t_f , without changing their relative order. The new schedule π^* is at least as good as π , and hence, is also optimal. This procedure can be repeated for each preempted object to get an optimal non-preemptive schedule, and hence, (ii).

To show (iii), consider a case with two objects. Let the size $s_1 = 5$ and $s_2 = 10$. Let the utility functions be as follows.

$$v_1(t) = \begin{cases} 10 & \text{if } 0 \leq t \leq 3 \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

$$v_2(t) = 20 - 2t$$

Let the schedule $\pi^1 = \{1, 2\}$ and $\pi^2 = \{2, 1\}$. Consider a bandwidth trajectory $\mu^1(t) = 5$. The total utility gained from each schedule is: $v(\pi^1) = 24$ and $v(\pi^2) = 26$. Hence, π^2 is the optimal sequence. Consider another bandwidth trajectory $\mu^2(t) = 2.5$. In this case, $v(\pi^1) = 18$ and $v(\pi^2) = 12$. Hence, π^1 is the optimal sequence.

Now suppose each object has a linear utility function of the form $v_i(t) = \beta_i - \alpha_i t$, where $\alpha_i > 0$ and $\beta_i \geq 0$. Given a fixed bandwidth trajectory $\mu(t)$ and an optimal sequence π , let us suppose object i and j are adjacent in π and i proceeds j . Suppose i starts service at time t_1 , ends service at t_2 , and j ends service at t_3 . Let π' be the sequence with object j and i interchanged. In π' , object j starts service at t_1 , ends service at t'_2 , and object i ends service at t_3 . Since π is optimal, we must have,

$$\alpha_i t_2 + \alpha_j t_3 \leq \alpha_j t'_2 + \alpha_i t_3$$

which yields,

$$\frac{t_3 - t'_2}{\alpha_i} \leq \frac{t_3 - t_2}{\alpha_j} \quad (11)$$

Let us denote $p_i(t)$ as the service time for object i when it ends service at time t . We then have $p_i(t_3) = t_3 - t'_2$ and $p_j(t_3) = t_3 - t_2$. Suppose all α_i 's are identical. Then, (11) becomes

$$p_i(t_3) \leq p_j(t_3) \quad (12)$$

Since at any time t , $s_i \leq s_j$ if and only if $p_i(t) \leq p_j(t)$, the only sequence that satisfies (12) at every completion time is the one that orders the objects in increasing order of their sizes. This demonstrates (iv).

When α_i 's are not identical, finding an optimal schedule seems to be a very difficult problem. In this case, the condition in (11) is only necessary but not sufficient to determine an optimal sequence. The following algorithm can find a “locally” optimal sequence in the sense that the resulting sequence cannot be improved by exchanging the positions of two neighboring objects.

Let t_n be the completion time for the entire transfer session. The last object to be transmitted, denoted by $\pi(n)$, should have the largest value of $p_i(t_n)/\alpha_i$ of all objects. Suppose we have determined the last k objects to be transmitted, $\pi(n-k+1), \pi(n-k+2), \dots, \pi(n)$. Let t_{n-k+1} be the completion time of object $\pi(n-k+1)$. Then, $t_{n-k} = t_{n-k+1} - p_{\pi(n-k+1)}(t_{n-k+1})$ is the starting time of object $\pi(n-k+1)$. Then, the $(n-k)^{th}$ object, $\pi(n-k)$, should have the largest value $p_i(t_{n-k})/\alpha_i$ among the remaining objects yet to be scheduled. The total running time is $O(n^2)$ for this algorithm, where n is the number objects.

In reality, the complete bandwidth trajectory may not be known ahead of time. In that case, one can modify the above algorithm as follows. After finishing transmission of an object at time t , the object with the smallest value of $\frac{s_i/\mu(t)}{\alpha_i}$ among all remaining objects is chosen for the next transmission, where $\mu(t)$ is the bandwidth at time t .

4.2 Evaluation of the SPT Schedule under Random Transmission Times

In the previous sections, we have assumed that the transmission time of each object is deterministic. In this section, we consider the case where the transmission times are random variables. The main objective is to model the case where a user retrieves a random page with n objects and to evaluate the performance of the SPT schedule.

Let us denote the random transmission (or processing) time of object i by X_i , $i = 1, 2, \dots, n$, which are not necessarily independent of each other at this point. Since the object completion times are random, our

goal is to schedule the n objects in order to optimize the expected utility. It is shown in [12] that, in the case of the linear utility functions, sequencing the objects in increasing order of $\mathbf{E}X_i/\alpha_i$ maximizes the expected utility in the class of non-preemptive static policies and also in the class of non-preemptive dynamic policies.

It is not a trivial task for the server to know the processing-time distributions, which are essential for forming the optimal schedule. Suppose the distributions are such that, for each object, the expected processing time is proportional to the object size, and hence,

$$\frac{\mathbf{E}X_i}{\mathbf{E}X_j} = \frac{s_i}{s_j} \quad (13)$$

for any pair of objects $i, j \in \{1, 2, \dots, n\}$. Then, in the case of the linear utility function, the sequence that follows the increasing order of $s_i/|\alpha_i|$ is optimal in the class of *non-preemptive static or dynamic policies*. It is reasonable to believe that the condition in (13) can be satisfied in many realistic situations. For instance, it is satisfied when each object is fragmented into packets of identical sizes, and the transmission times for the packets are independently and identically distributed.

In the following, we will evaluate the improvement of the SPT schedule over a random schedule. To model the situation, let us suppose the sizes of the objects are independently and identically distributed (IID) random variables and the transmission bandwidth is a constant, say, equal to 1. Then, the transmission times of the objects are IID random variables drawn from some distribution F . We will consider two distributions, the exponential distribution and the Pareto distribution. For each realization of the transmission times, we apply separately the SPT schedule and the First-Come-First-Serve (FCFS) schedule, which simply transmits the n objects in the order they are given. The FCFS schedule is equivalent to the random schedule, which transmits the objects in a uniformly random order. The performance criterion is the expected mean completion time $\mathbf{E}\bar{C}_n = \frac{1}{n} \sum_{i=1}^n \mathbf{E}C_i$, where C_i is the completion time of object i in the schedule. We denote the mean completion time by \bar{C}_n to emphasize its dependence on n , the number of objects. We already know that the SPT schedule minimizes \bar{C}_n , and therefore, $\mathbf{E}\bar{C}_n$. Given the object transmission times, X_1, X_2, \dots, X_n , let $X_{(k)}$ be the k^{th} order statistics of $\{X_1, X_2, \dots, X_n\}$, i.e., $X_{(1)} \leq X_{(2)} \leq \dots \leq X_{(n)}$. In the case of the SPT schedule, $\bar{C}_n^{\text{SPT}} = \sum_{i=1}^n (n-i+1)X_{(i)}/n$.

4.2.1 Exponential Transmission Times

Suppose each X_i is distributed exponentially with mean $1/\nu$, for $i = 1, 2, \dots, n$. The first moment of any order statistics has a closed-form expression (see page 49 in [3]).

$$\mathbf{E}X_{(k)} = \frac{1}{\nu} \sum_{i=1}^k \frac{1}{n-i+1} \quad (14)$$

for $k = 1, 2, \dots, n$. It can be shown that, for the SPT schedule,

$$\mathbf{E}\bar{C}_n^{SPT} = \frac{n+3}{4} \frac{1}{\nu} \quad (15)$$

For the random schedule

$$\mathbf{E}\bar{C}_n^{RAND} = \frac{n+1}{2} \frac{1}{\nu} \quad (16)$$

The expected mean completion time in the SPT schedule is about half of that in the random schedule. Since the completion times for the last object are the same in both schedules, it must be true that some other objects are completed earlier in the SPT schedule. In fact, for each realization of transmission times, the SPT schedule finishes more objects than any other schedule at any time. The user may feel the objects arrive faster in the SPT schedule. It is not surprising that, in both schedules, $\mathbf{E}\bar{C}_n$ is linear in the number of objects. In the case of fine-grained multiplexing, where each object is further divided into m smaller objects of identical sizes, the mean size of each smaller object becomes $1/(m\nu)$. Hence, for large n , $\mathbf{E}\bar{C}_n$ is not affected by the fine-grained multiplexing.

4.2.2 Pareto Transmission Times

Previous statistical studies on the file/object sizes give strong indication that the object sizes follow a heavy-tail distribution [2]. In this subsection, we will assume the object sizes have Pareto distribution with distribution function $F(x) = 1 - K^\alpha x^{-\alpha}$, where $x \geq K$ and $\alpha > 0$. For our purpose, we further require $\alpha > 1$, in which case the mean of the distribution exists and is equal to $\frac{\alpha K}{\alpha-1}$. The first moments of the order statistics for the transmission times are, (See page 50 in [3].)

$$\mathbf{E}X_{(k)} = \frac{n!}{(n-k)!} \frac{\Gamma(n-k+1-1/\alpha)K}{\Gamma(n+1-1/\alpha)} \quad (17)$$

for $k = 1, 2, \dots, n$. In the above, for $s > 0$, the gamma function is defined by,

$$\Gamma(s) = \int_0^\infty e^{-x} x^{s-1} dx$$

We will evaluate $\mathbf{E}\bar{C}_n$ numerically for a few realistic cases. In case 1, we pick $\alpha = 1.1$ and $K = 100$ (bytes) for the Pareto distribution, and hence, the mean object size is 1100 bytes. In case 2, we pick $\alpha = 1.5$ and $K = 100$, and the resulting mean object size is 300 bytes. The expected mean completion times $\mathbf{E}\bar{C}_n$ for the SPT schedule and the random schedule are shown in figure 2 and figure 3 for the two cases. The values for $\mathbf{E}\bar{C}_n$ are approximately linear in n . In the first case, the SPT schedule has a much smaller value for $\mathbf{E}\bar{C}_n$ than the random schedule, with $\mathbf{E}\bar{C}_n^{SPT} \approx \frac{1}{6}\mathbf{E}\bar{C}_n^{RAND}$. In the second case, $\mathbf{E}\bar{C}_n^{SPT} \approx \frac{1}{2}\mathbf{E}\bar{C}_n^{RAND}$. The improvement of the SPT schedule depends on the actual statistics of the object sizes.

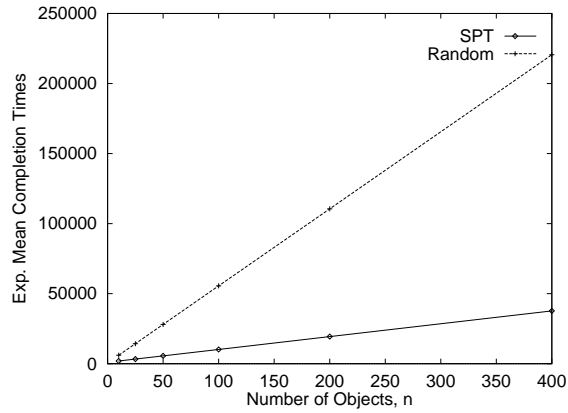


Figure 2: $K = 100, \alpha = 1.1$

To understand how the performance improvement of the SPT schedule depends on the parameters for the Pareto distribution, we resort to an asymptotic theorem regarding linear combinations of order statistics, found in [14]. Let us define $S_n = \bar{C}_n/(n + 1)$, and a function $J(u) = 1 - u$, for $0 \leq u \leq 1$. Then,

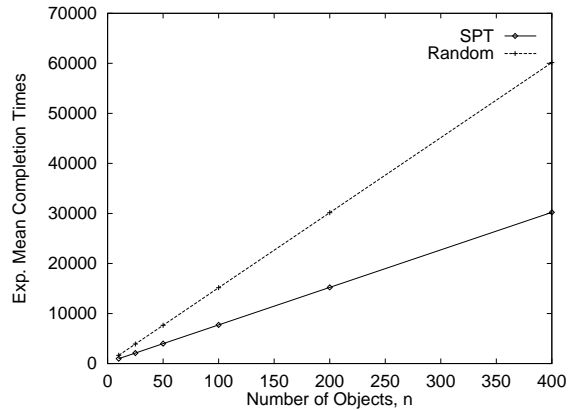


Figure 3: $K = 100, \alpha = 1.5$

$S_n = \frac{1}{n} \sum_{i=1}^n J(\frac{i}{n+1})X_{(i)}$ in the SPT schedule. Theorem 3 in [14] says, if $\mathbf{E} | X_i | < \infty$ and $J(u)$ is bounded and continuous, then as $n \rightarrow \infty$, $\mathbf{E}S_n \rightarrow \theta(J, F)$, where

$$\theta(J, F) = \int_0^1 J(u)F^{-1}(u)du = \int_{-\infty}^{\infty} xJ(F(x))dF(x) \quad (18)$$

When $F(x)$ is Pareto, it is easy to show

$$\theta(J, F) = \frac{\alpha K}{2\alpha - 1}$$

Hence, for large n , $\mathbf{E}\bar{C}_n^{SPT}$ grows approximately linearly with $n+1$ at a slope $\theta(J, F)$. As $\alpha \downarrow 1$, i.e., as the distribution becomes more heavy-tailed, the slope approaches K ; as $\alpha \rightarrow \infty$, the slope approaches $K/2$. In the random schedule, $\mathbf{E}\bar{C}_n^{RAND} = \frac{\alpha K}{2(\alpha-1)}(n+1)$. As $\alpha \downarrow 1$, its slope approaches ∞ ; as $\alpha \rightarrow \infty$, its slope approaches $K/2$.

We define a useful comparison measure, $\gamma = \mathbf{E}\bar{C}_n^{SPT} / \mathbf{E}\bar{C}_n^{RAND}$, which is approximately $\frac{2(\alpha-1)}{2\alpha-1}$ for large n . As the file size becomes more and more heavy-tailed, the expected mean completion time for the SPT schedule becomes an increasingly smaller fraction of that for the random schedule. This is in sharp contrast with the case of exponential file sizes where the same ratio, approximately $1/2$, does not depend on the parameters of the distribution. As examples for the Pareto case, for $\alpha = 1.5$ or 1.1 , $\gamma = 1/2$ or $1/6$, respectively. These numbers are in agreement with figure 2 and figure 3. The figures also indicate that the approximation by the asymptotic result becomes fairly accurate for even small n .

Since $0 \leq J(\cdot) \leq 1$, from (18), we know

$$\theta(J, F) \leq \int_{-\infty}^{\infty} x dF(x) = \mathbf{E}X_i$$

Hence, if we keep the mean file size the same, the worst case distribution in terms of the expected mean completion time is the deterministic distribution.

Let us try to understand some peculiar features of the Pareto transmission times. Among n independently and identically distributed Pareto random variables, the maximum is a special one. This can be seen by looking at the quantity $\mathbf{E}X_{(k+1)} / \mathbf{E}X_{(k)}$, for $1 \leq k < n$. By using (17) and the familiar

$$\Gamma(s+1) = s\Gamma(s) \quad \text{for } s > 0$$

we get,

$$\frac{\mathbf{E}X_{(k+1)}}{\mathbf{E}X_{(k)}} = \frac{n-k}{n-k-1/\alpha}$$

As $\alpha \downarrow 1$, $\mathbf{E}X_{(n)}/\mathbf{E}X_{(n-1)}$ increases to infinity. On the other hand, $\mathbf{E}X_{(k+1)}/\mathbf{E}X_{(k)} < 2$ for $1 \leq k < n-1$, and for most values of k , the ratio is in fact close to 1. In other words, the value of $\mathbf{E}X_{(k)}$ increases very slowly except for the last few k . It has a sudden upward jump at $k = n$.

Depending on α , the expectation of the maximum value can be significantly greater than that of any other order statistics. For the values of α and K that are relevant to our problem and when n is not so large, the maximum is often larger than or comparable to the sum of the rest random variables. For instance, when $K = 100$, $\alpha = 1.1$ and $n = 50$, $\mathbf{E}X_{(n)} = 36839$ and $\mathbf{E}\sum_{i=1}^{n-1} X_{(i)} = 18160$. We are led to consider an even simpler schedule: move the largest object to the last position and leave the rest of objects where they were. We call this schedule Largest-To-Last (LTL). It is expected to perform well when the object sizes are very heavy-tailed and when the number of objects is not so large. To quantify its performance, note that

$$\mathbf{E}[X_i | X_i < X_{(n)}] = \frac{n\mathbf{E}X_i - \mathbf{E}X_{(n)}}{n-1}$$

$$\begin{aligned} \mathbf{E}\bar{C}_n^{LTL} &= \frac{1}{n} \left(\sum_{k=2}^n k \mathbf{E}[X_k | X_k < X_{(n)}] + \mathbf{E}X_{(n)} \right) \\ &= \frac{n+1}{2} \mathbf{E}X_i - \frac{1}{2} \mathbf{E}X_{(n)} + \frac{1}{2n} \mathbf{E}X_{(n)} \end{aligned} \quad (19)$$

(19) is valid for any distributions for which the expectations are defined. It indicates that the LTL schedule should greatly improve over the random schedule when the maximum is comparable to the sum, which is often the case for the data files. Table 2 shows the performance comparison for the three schedules. When the number of objects is less than 50, the LTL scheduler is nearly optimal.

Table 2: Performance comparison for the SPT, LTL and random schedules

α	n	$\mathbf{E}\bar{C}_n^{SPT}$	$\mathbf{E}\bar{C}_n^{LTL}$	$\mathbf{E}\bar{C}_n^{RAND}$
1.1	10	1925	2199	6050
	50	5592	9999	28050
	100	10175	21321	55550
1.5	10	975	1084	1650
	50	3975	5864	7650

5 Class-Based Utility Function Assignment

As we have seen, many optimal sequencing problems are very difficult. When the problem is strongly NP-hard, finding approximations with performance guarantee can also be difficult. In this section, we consider the situation where the n objects can be grouped into a small number of classes. This formulation may either faithfully reflect reality, or may be regarded as a systematic approximation to reality. In either case, we hope for simpler solutions coming out of this formulation. We will use function V as an example, since it is both very versatile and very hard. We will show that, indeed, there exist algorithms with polynomial complexity in n , the number of objects.

A group of objects with identical d_i , α_i , β_i and p_i are considered as one object class. Suppose there are total K classes, from 1 to K . From now on, let the subscript indices denote classes. Let n_i be the number of objects in class i , $1 \leq i \leq K$. Without loss of generality, let $d_1 \leq d_2 \leq \dots \leq d_K$. These K deadlines divide $[0, \infty)$ into $K + 1$ intervals or semi-intervals. Let us call $(d_{j-1}, d_j]$ the j^{th} interval, $1 \leq j \leq K$, where $d_0 = 0$, and call (d_K, ∞) the $(K + 1)^{\text{th}}$ interval. Let the non-negative integer, n_i^j , be the number of class- i objects that are completed on the j^{th} interval. It must be true that $\sum_{j=1}^{K+1} n_i^j = n_i$, $1 \leq i \leq K$. The problem is, therefore, to determine a feasible set of $\{n_i^j; 1 \leq i \leq K, 1 \leq j \leq K + 1\}$ so that the resulting sequence is optimal. Feasibility means that the sequence indeed has the property that n_i^j class- i objects are completed on the j^{th} interval, for all i and j .

After time d_{k-1} , the value of any class- i object becomes zero, where $i < k$. In an optimal schedule, if any class- i object, $i < k$, is serviced on the k^{th} interval, it can be moved after all class l objects on the interval, for every $l \geq k$, without affecting the optimality. We also know that, on the k^{th} interval, the objects from classes $l \geq k$ should be processed in increasing order of $p_l/|\alpha_l|$. Therefore, on the k^{th} interval, if we know the set of numbers n_i^k , $1 \leq i \leq K$, the order of these $\sum_{i=1}^K n_i^k$ objects can be made unambiguous. Hence, given the set $\{n_i^j; 1 \leq i \leq K, 1 \leq j \leq K + 1\}$, the order of the n objects can be made unambiguous. The feasibility of this set of numbers can be checked after the objects are ordered. Let \underline{n} denote the matrix $(n_i^j), 1 \leq i \leq K, 1 \leq j \leq K + 1$, and let the resulting value from the schedule corresponding to a feasible \underline{n} be $v(\underline{n})$. The problem is to find a feasible \underline{n} such that $v(\underline{n})$ is maximized. This can be accomplished by enumerating all possible \underline{n} 's.

We can compute the number of possible matrices \underline{n} . Let us first focus on class i . When $\sum_{j=1}^{K+1} n_i^j = n_i$, and each n_i^j is a non-negative integer, the total number of distinct vectors $(n_i^1, n_i^2, \dots, n_i^{K+1})$ is $M_K(i) = (n_i + 1)^K / 2 + (n_i + 1) / 2$. This can be shown inductively. First, when $K = 1$, it is clear that the vector

has the form $(k, n_i - k)$, $0 \leq k \leq n_i$. Hence, $M_1(i) = n_i + 1$. Suppose for $K = m$, $M_m(i) = (n_i + 1)^m/2 + (n_i + 1)/2$. Then, when $K = m + 1$, n_i^{m+1} can take values from 0 to n_i . Therefore,

$$\begin{aligned} M_{m+1}(i) &= \left(\frac{(n_i + 1)^m}{2} + \frac{n_i + 1}{2}\right) + \left(\frac{(n_i + 1)^m}{2} + \frac{n_i + 1}{2} - 1\right) \\ &\quad + \dots + \left(\frac{(n_i + 1)^m}{2} + \frac{n_i + 1}{2} - n_i\right) \\ &= \frac{(n_i + 1)^{m+1}}{2} + \frac{n_i + 1}{2} \end{aligned}$$

Hence, the total number of possible matrices \underline{n} is

$$\begin{aligned} \prod_{i=1}^K M_K(i) &= \prod_{i=1}^K \left(\frac{(n_i + 1)^K}{2} + \frac{n_i + 1}{2}\right) \\ &\leq \prod_{i=1}^K (n_i + 1)^K \\ &\leq \left(\frac{n + K}{K}\right)^{K^2} \\ &\sim \frac{n^{K^2}}{K^{K^2}} \quad \text{as } n \rightarrow \infty \end{aligned}$$

Therefore, the algorithm for finding an optimal sequence has a complexity $O(n^{K^2}/K^{K^2})$, which is polynomial in n . More careful counting shows that the power to n can be reduced further. Notice that, after time d_k , we do not need to consider the class- k objects anymore because they all have zero value. Hence, on the $(k + 1)^{th}$ interval, i.e., $(d_k, d_{k+1}]$, we need only to determine n_l^{k+1} for $l > k$. Very crudely, the number of matrices, \underline{n} , to be considered is $O(n^{(K-1)K/2})$. With this complexity, the corresponding algorithm is only practical for $K \leq 3$ when n is around 100.

5.1 Case of Continuous Function

Suppose $\alpha_i < 0$ and $\alpha_i d_i + \beta_i = 0$ for all i . In this case, an optimal schedule can be found using the algorithm from section 3.2.2. This algorithm does not take advantage of knowledge about the object classes and has a running time $O(n \sum_{i=1}^K n_i p_i)$. We will present another algorithm whose complexity is independent of p_i 's. Without loss of generality, suppose $p_1/|\alpha_1| \leq p_2/|\alpha_2| \leq \dots \leq p_K/|\alpha_K|$. By lemma 3.3, there exists an optimal schedule that starts with m_1 class-1 objects, followed by m_2 class-2 objects, ..., followed by m_K class- K objects, all of which are on time, then followed by the tardy objects in arbitrary order. We only need to determine the integers m_i 's through enumeration, where $0 \leq m_i \leq n_i$, $1 \leq i \leq K$. Since $\sum_{i=1}^K n_i = n$,

for $K > 1$, the total number of choices is

$$\begin{aligned} \prod_{i=1}^{K-1} (n_i + 1) &\leq \left(\frac{n}{K-1} + 1\right)^{K-1} \\ &\sim \frac{n^{K-1}}{(K-1)^{K-1}} \quad \text{as } n \rightarrow \infty \end{aligned}$$

where the upper bound above is obtained by the standard maximization technique. The resulting algorithm is polynomial in n . For n around 100, the algorithm is practical for $K \leq 5$.

6 Version Selection

In this section, we consider a different model for object transmission. Suppose each object is encoded at a set of different resolutions, resulting in different versions of the same object. When a user requests the page, we would like to choose a resolution for each object and a transmission schedule so that the total received utility is maximized. To model this situation, let each object i have m versions, with processing time p_i^j and utility function $v_i^j(t)$, $1 \leq j \leq m$. The problem seems to be very difficult even for the linear utility functions (However, we were unable to show it is NP-hard in this case). In the following, we'll focus on the step functions. For each object $1 \leq i \leq n$ and version $1 \leq j \leq m$, let the utility function be

$$v_i^j(t) = \begin{cases} w_i^j & \text{if } 0 \leq t \leq d_i \\ 0 & \text{otherwise} \end{cases}$$

where $w_i^j > 0$ is the value of the object if it arrives before the deadline d_i . Note that, for each object, the deadline is common for all versions. The objective is to choose a version assignment for each object and a transmission order so that $\sum_{i=1}^n v_i^j(C_i)$ is maximized. It is clear that there exists an optimal schedule in which the objects are sequenced in increasing order of their deadlines. We therefore arrange the objects in that order. Without loss of generality, we assume this order is $1, 2, \dots, n$. Version assignment can be made by straightforward extension to Lawler and Moore's algorithm (1). Let us define the $(m+1)^{th}$ "version" for each object i , which corresponds to object i being unable to finish before its deadline.

$$\begin{aligned} v_i^{m+1}(t) &= 0 \\ p_i^{m+1} &= 0 \end{aligned}$$

We will convert the problem into the minimization version in order to reuse some notations from Algorithm (1). For each $1 \leq i \leq n$, without loss of generality, let us suppose $w_i^m \geq w_i^j$ for all $1 \leq j \leq m$. For

each $1 \leq i \leq n$ and each $1 \leq j \leq m$, define

$$\rho_i^j(t) = \begin{cases} w_i^m - w_i^j & \text{if } 0 \leq t \leq d_i \\ +\infty & \text{otherwise} \end{cases}$$

For each $1 \leq i \leq n$, define

$$\rho_i^{m+1}(t) = w_i^m$$

Then, the dynamic programming relation is captured by, for $j = 1, 2, \dots, n$ and $t \geq 0$,

$$f(j, t) = \min \begin{cases} f(j, t - 1) \\ \rho_j^k(t) + f(j - 1, t - p_j^k) \quad \text{for } k = 1, 2, \dots, m + 1 \end{cases}$$

A different version of the problem is to maximize $\sum_{i=1}^n v_i^j(C_i)$, subject to the constraint that all objects should be completed before their deadlines. In this case, we simply remove the artificial $(m + 1)^{th}$ version of the objects from the above algorithm.

7 Design Example and Concluding Remarks

7.1 Linear Utility Function for Object Transmission

If there is freedom in choosing the utility function, the discussions in the paper suggest that the linear utility function is among the good choices. Without precedence constraints, an optimal schedule is to sequence the n objects in increasing order of $p_i/|\alpha_i|$. Since p_i is not necessarily known due to the uncertainty in the transmission bandwidth, we propose to sequence the objects in increasing order of $s_i/|\alpha_i|$, where s_i is the size of object i . The algorithm leads to an optimal or near-optimal schedule that maximizes the expected sum of utilities for a wide class of random transmission times. It is very likely that, in reality, the distribution of the transmission times belongs to this class. Since the parameter β_i in the utility function is irrelevant for finding an optimal schedule, the utility function requires only one parameter, α_i , to be specified for each object.

The optimal schedule follows *Weighted Shortest Processing Time* (WSPT) first rule, which is an extension of the *Shortest Processing Time* (SPT) scheduling rule. To some degree, the WSPT schedule inherits some of the advantages of the SPT schedule. Given a fix time, the SPT schedule completes more objects than any other schedule. This is especially beneficial when the collection of objects, say on a web page,

contains many small objects and one or a few large ones, because the large objects are pushed to the end of the transmission sequence and most objects will arrive during the early period of the transfer session. The SPT schedule is also friendly to fine-grained encoding of information objects. For instance, in the case of multi-resolution encoding, the sizes of the lower-resolution objects are small, and can arrive early and be displayed quickly. It is possible that, in many situations, most of the value is delivered to the user at this point. There can be a valuable trade-off in sending some smaller objects first at the expense of slight increase in the delay of larger objects. On the other hand, in the case when some larger objects are much more valuable to the user or when the delay increase for the larger objects becomes significant, the WSPT schedule can move the larger objects ahead of the smaller objects.

7.2 Concluding Remarks

In this paper, we study a collection of scheduling problems motivated by the desire to arrange the downloading order of information objects. We formulated these problems as to optimize the total utilities received by the user. This formulation is somewhat different from traditional single machine scheduling, and as a result, new scheduling problems arise. We hope the solutions we provide can have a wider range of applications of similar nature. We stress that it is very natural to think about the objects as having some utility to the user. In this paper, we do not address how the utility functions are obtained. Presumably, they can either be measured, or directly assigned by the users. It is also possible that the system designer chooses a particular family of utility functions and the users choose the function parameters. All our utility functions are non-increasing with time, and possibly have one or two discontinuities (or discontinuities in the first derivative), which can be interpreted as deadlines. A salient feature of this paper is that the solutions are often based on the simple building blocks of the linear or exponential functions with Lawler and Moore's algorithm (1) to handle the deadlines.

References

- [1] Kenneth R. Baker. *Introduction to Sequencing and Scheduling*. John Wiley & Sons, 1974.
- [2] Mark E. Crovella and Azer Bestavros. Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes. *IEEE/ACM Transactions on Networking*, 5(6):835–846, December 1997.
- [3] H. R. David. *Order Statistics*. John Wiley & Sons, second edition, 1981.

- [4] Simon French. *Sequencing and Scheduling - An Introduction to the Mathematics of the Job-Shop*. John Wiley & Sons, 1982.
- [5] G.V. Gens and E.V. Levner. Fast Approximation Algorithm for Job Sequencing with Deadlines. *Discrete Applied Mathematics*, 3, pages 313–318, 1981.
- [6] J.Du and J.Y.-T. Leung. Minimizing Total Tardiness on One Machine is NP-Hard. *Math. Oper. Res.*, 1989.
- [7] Richard M. Karp. Reducibility among Combinatorial Problems. In *Complexity of Computer Computation*, pages 85–103. Plenum Press, 1972.
- [8] E.L. Lawler. A Pseudo-Polynomial Algorithm for Sequencing Jobs to Minimize Total Tardiness. *Annals of Discrete Mathematics*, 1, pages 331–342, 1977.
- [9] E.L. Lawler and J. M. Moore. A Functional Equation and Its Application to Resource allocation and Sequencing Problems. *Management Science*, pages 77–85, 1969.
- [10] J.K. Lenstra and A.H.G. Rinnooy Kan. Complexity of Scheduling under Precedence Constraints. *Annals of Discrete Mathematics*, pages 22–35, 1978.
- [11] J.K. Lenstra, A.H.G. Rinnooy Kan, and P. Brucker. Complexity of Machine Scheduling Problems. *Annals of Discrete Mathematics*, 1, pages 343–362, 1977.
- [12] Michael Pinedo. *Scheduling - Theory, Algorithms and Systems*. Prentice-Hall, Inc., 1995.
- [13] Sartaj K. Sahni. Algorithms for Scheduling Independent Tasks. *Journal of the Association for Computing Machinery*, Vol. 23, No. 1, pages 116–127, 1976.
- [14] Stephen M. Stigler. Linear Functions of Order Statistics with Smooth Weight Functions. *The Annals of Statistics*, 2(4):676–693, 1974.
- [15] Dileep R. Sule. *Industrial Scheduling*. PWS Pub. Co., 1997.
- [16] V.S. Tanaev, Y.N. Scotskov, and V.A. Strusevich. *Scheduling Theory - Single-Stage Systems*. Kluwer Academic Publishers, 1989.
- [17] J. Yuan. NP Hardness of the Single Machine Common Due Date Weighted Tardiness Problem. *System Sci. Math. Studies*, 5:328–333, 1992.