6-29-1999

Existence Theorems for Scheduling to Meet Two Objectives

April M. Rasala
Dartmouth College

Dartmouth College Computer Science Technical Report PCS-TR99-347

Revised version of June 29, 1999

Existence Theorems for Scheduling to Meet Two Objectives

April Rasala
Advisor: Cliff Stein

Abstract

We will look at the existence of schedules which are simultaneously near-optimal for two criteria. First, we will present some techniques for proving existence theorems, in a very general setting, for bicriterion scheduling problems. We will then use these techniques to prove existence theorems for a large class of problems. We will consider the relationship between objective functions based on completion time, flow time, lateness and the number of on-time jobs. We will also present negative results first for the problem of simultaneously minimizing the maximum flow time and average weighted flow time and second for minimizing the maximum flow time and simultaneously maximizing the number of on-time jobs. In some cases we will also present lower bounds and algorithms that approach our bicriterion existence theorems. Finally we will improve upon our general existence results in one more specific environment.

1 Introduction

This thesis will focus on the issue of scheduling a set of jobs on a set of resources to simultaneously minimize two criteria. This work builds both on ideas from the literature of single criterion scheduling and also on some recent work in bicriterion scheduling. Scheduling has been studied formally for nearly half a century and has been motivated by the common problem of having a limited number of resources on which to efficiently process jobs or requests. This general problem has been broken into many specific scheduling models that allow for various machine environments, optimality criteria and additional constraints to be considered. Some surveys on the existing work in scheduling theory have been done by Lawler, Lenstra, Rinnooy Kan and Shmoys [17], Graham, Lawler, Lenstra and Rinnooy Kan[8], Karger, Stein and Wein[14] and Pinedo[22].

Many natural scheduling problems that arise are concerned with only optimizing a single objective. Consider for instance the example of scheduling a single machine to complete a set of tasks as quickly as possible. In this case the only objective is to finish all jobs in the shortest possible amount of time. Another criterion that has been extensively studied is the average weighted completion time of a schedule. This is often used in situations where the resources that are being scheduled are shared among a group of users. In this setting one particular user may not care about the total length of the schedule but may be very concerned with how quickly his own tasks or requests are completed. If the scheduler is only worried about insuring that all users feel their requests are being completed reasonably fairly then an approach that minimizes the average completion time of all jobs will probably be best.

At times problems occur that require that more than one optimality criterion be considered. Suppose in the last example that the resources being scheduled were actually a group of people with a variety of skills and multiple bosses. This group of people wants to keep all the bosses happy by completing requests as promptly as possible and also wants to insure that the total time needed to complete all requests will not exceed the end of the work day. In essence this group of workers is concerned with both the average completion time of all the jobs received over the course of the day and also with the total length of time it will take to complete this set of jobs. If the workers relied on either criterion by itself, the resulting schedule could turn out to be far from optimal in terms of the other criterion.

While the workers in the previous example are most concerned with knowing how to schedule themselves to come close to meeting both objectives, the first question that needs to be answered is whether such a schedule exists. This thesis will primarily address the question of existence and show that in most bicriterion settings, schedules exists which are simultaneously close to optimal for both criteria.

The organization of this thesis is as follows. In the next section we will briefly mention some of the related work. In the third section we will look at some new results and two pieces of related work that present general methods for proving bicriterion existence results, finding lower bounds that match these results and finally in some cases finding approximation algorithms that come close to achieving the best possible bicriterion schedules. The fourth section will extend these ideas to new bicriterion scheduling problems. In examining new problems we will also consider two instances in which no schedule exists that is simultaneously a constant factor approximations for both criteria. While the existence results from the third and fourth sections apply to a very general class of scheduling problems, the fifth section will briefly look at one specific environment and prove a better existence theorem for this restricted case. Finally we will conclude by presenting some open problems.

2 Background

2.1 Notation

Very generally, this thesis deals with scheduling n jobs on a set of m resources. Each job j requires some non-negative processing time, p_j, on one of the machines. We define a schedule S to be an assignment of jobs to machines over time. For a schedule S we will denote the completion time of j in S as C_j^S. We may omit the superscript S when the schedule is obvious from the context. In some cases we will consider instances in which each job j may have an associated non-negative weight w_j, release date r_j and/or a due date d_j. By weighting jobs differently we can designate the relative importance between jobs. Release dates allow us to model situations where jobs arrive over time and are therefore not all available for processing at the start of the schedule. Due dates represent a time at which we would preferably like the job completed. We will consider models in which the lateness of a job is of importance or others in which we are only concerned with whether or not a job finishes after its due date.

2.1.1 Optimality Criteria

Optimality Criteria Based on Completion Times: The average weighted completion time (ACT) of S is defined to be $\frac{1}{n}\sum w_j C_j^S$. Notice that minimizing $\frac{1}{n}\sum w_j C_j^S$ is equivalent to minimizing the total weighted completion time which is given by $\sum w_j C_j^S$. We will use the terms

average weighted completion time and total weighted completion time interchangeably since they are equivalent objectives. We will let C_j^* denote the completion time of job j in the optimal ACT schedule and therefore the optimal value is written $\sum_j w_j C_j^*$.

The makespan, or schedule length, is defined to be the maximum completion time of any job in the schedule. We define the makespan of a schedule S to be $C_{\max}^S = \max_j C_j^S$. For a given problem instance we will denote the minimum makespan, over all possible schedules S, as

$$C_{\max}^{\text{opt}} = \min_S C_{\max}^S.$$

Optimality Criteria Based on Lateness: When each job j has a deadline d_j, then the lateness of j is defined to be $L_j = C_j - d_j$. Just as the average completion time and maximum completion time of a schedule are of interest, similarly the average weighted lateness denoted $\sum_j w_j L_j$ and the maximum lateness given by $L_{\max} = \max_j L_j$ are also natural objective functions. Unfortunately, L_j as defined above can be difficult to work with. Since L_j can be zero or negative, the optimal value of either criteria could likewise be zero or negative. Therefore L_j, in this form, does not easily lend itself to meaningful approximation techniques.

One way to avoid this is to insure that all jobs have a positive lateness in the optimal schedule. Consider the problem instance created by subtracting from each d_j a uniform positive constant δ. Let $d'_j = d_j - \delta$. If δ is large enough, say $\delta > \max_j d_j$, then each job j will have $d'_j < 0$. As a result, when we compute the lateness of each job based on the new deadline d'_j every job will have an optimal lateness of strictly positive value. Also notice that an optimal solution for the new deadlines is also an optimal solution for the old deadlines. However, having negative deadlines is a somewhat unnatural construction. Note also that by taking δ to be arbitrarily large, we can prove arbitrarily good approximation ratios for any algorithm.

Now suppose that instead of associating deadlines with jobs we associate delivery times. We let q_j be the non-negative delivery time required by job j. Problems involving delivery times occur naturally in many production environments. Consider a pizza shop that guarantees that all pizzas will arrive at your home within 30 minutes of an order being placed. We can let C_j be the time that the j^{th} pizza finishes cooking, and q_j be the amount of time it will take for that pizza to be delivered to the costumer. In this case, the pizza shop is not looking at minimizing the maximum completion time but instead is minimizing the maximum completion time plus delivery time. Therefore, the pizza shop wants to know that they can guarantee that

$$\max_j (C_j + q_j) \leq 30 \texttt{ minutes}$$

in order to meet the customer's expectations. One may wonder why we don't just adjust the processing time of each job to include the delivery time required by that job. The reason for keeping p_j and q_j separate is that one could consider p_j to be the amount of time that the job requires a scarce resource. In our pizza shop example imagine that ovens are very expensive but we can hire many inexpensive teenagers to deliver the pizzas. Therefore we want to be able to schedule the oven based on knowing how long each pizza is going to need to actually be in the oven.

We will now see the connection between delivery time problems and problems involving lateness. Recall that we made our deadlines negative to insure that all jobs had optimal positive lateness. Suppose that we transform an instance involving due dates to one involving delivery times as follows. For each job j we let $q_j = -d'_j$. As stated above, an optimal solution in terms of adjusted deadlines will also be an optimal solution for our original lateness problem. Therefore we know that if we find an optimal solution in the delivery time model of lateness it is also an optimal solution for

the corresponding lateness problem. While the delivery time model does not solve the problem of determining the quality of an approximate schedule in terms of approximating an optimal lateness solution, we will use it since it does have another natural interpretation. We will overload notation and let $L_j = C_j + q_j$. We will let $L_{\max}^{\text{opt}}$ be the optimal maximum lateness and L_j^* be the lateness of j in the optimal average lateness schedule with value $\sum w_j L_j^*$.

For the rest of this paper when we talk about lateness we will mean lateness as used in the delivery time model of lateness.

Optimality Criteria Based on the Number of On-Time Jobs: While measuring the lateness of a job can be of importance, in some cases one is only interested in whether or not a job is late. For this problem we will define the variable U_j for each job j. For a particular schedule S, we will use U_j^S to indicate whether or not job j meets its deadline in schedule S. In other words U_j^S will have the following value

$$U_j^S = \begin{cases} 0 & \text{if } C_j^S \leq d_j \\ 1 & \text{otherwise.} \end{cases}$$

Using U_j we will look at maximizing the number of jobs that meet their deadline. For a particular schedule S we compute the total number of jobs that complete on-tome as $\sum(1-U_j^S)$. If we have an optimal schedule S^* we will let U_j^* be 0 if job j meets its deadline in S^* and 0 otherwise. We will then define the optimal number of jobs finishing on time to be $\sum(1-U_j^*)$.

Optimality Criteria Based on Flow Time: Lastly, the flow time of a job j is defined to be $F_j = C_j - r_j$. The flow time can be thought of as representing the amount of time a job actually spends in the system before it is completed. Flow time is considered an accurate measure of the responsiveness of a system. Again, two interesting criteria are the maximum flow time of any job in schedule S, denoted $F_{\max}^S = \max_j F_j^S$, and the average weighted flow time of S which is given by $\sum w_j F_j$. We will let

$$F_{\max}^{\text{opt}} = \min_S F_{\max}^S$$

be the minimum over all possible schedules of the maximum flow time of a set of jobs. We will also define F_j^* to be the flow time of job j in the optimal average flow time schedule and denote the value of this optimal schedule as $\sum_j w_j F_j^*$.

2.1.2 Environments

When we need to specify an environment, we will follow the notation of Graham, Lawler, Lenstra and Rinnooy Kan[8] and use a triplet $\alpha|\beta|\gamma$ to define the problem. In this notation α contains 1 entry and represents the machine environment and β contains anywhere from zero to multiple entries and specifies any additional constraints such as release dates or precedence constraints. We will extend the notation to allow the γ field, which is used for the optimization criterion, to contain 1 or more entries.

2.1.3 Definitions

For simplicity when stating many of the existence theorems we will not specify a particular environment but say instead that the result applies to *any* scheduling problem. We borrow the definition of *any* from Stein and Wein[25] and mean any problem that meets the following two conditions:

1. Truncation at time t: If we take a valid schedule S and remove from it all jobs that complete after time t, the schedule remains a valid schedule for those jobs that remain.

2. Composition: Given two valid schedules S_1 and S_2 for two sets J_1 and J_2 of jobs (where $J_1 \cap J_2$ is potentially nonempty), the composition of S_1 and S_2, obtained by appending S_2 to the end of S_1, and removing from S_2 all jobs that are in $J_1 \cap J_2$, is a valid schedule for $J_1 \cup J_2$.

These two conditions encompass most environments. For instance, although we have specified a jobs' release time as r_j in order to simplify notation, these two conditions also allow for jobs to have machine dependent release dates. However, the Composition condition is violated by problems in which the processing time of a job is not constant over time or a job is required to run at a specific time. Strict precedence constraints that require anything of the form job j must *immediately* precede job i could violate either of the two conditions.

2.1.4 Combine(S_1, S_2, t)

Suppose we have two valid schedules S_1 and S_2 for a set of jobs J. Then we will use the following construction to create a valid schedule S' for J.

Combine(S_1, S_2, t)

1. Let K be the set of jobs which complete after time t in schedule S_2.

2. Create schedule S_2' from schedule S_2 by removing from S_2 all jobs in K.

3. Create schedule S_1' by removing from S_1 all jobs in $J - K$.

4. Create schedule S' by appending S_1' to the end of S_2'.

Clearly S' runs all jobs in J. Since jobs are only delayed with respect to their start time in either S_1 or S_2, we know that S' does not violate any release dates. Furthermore S' also respects precedence constraints. If we consider any pair of jobs j_i and j_k with the constraint that job j_i must complete before j_k can start then we know by our construction that either both jobs will be run according to either schedule S_1 or S_2 or the j_i will run according to S_2 and finish before time t in S' and j_k will run according to S_1 and therefore will not start before time t in S'. In general our two conditions on any scheduling problem are enough to show that S' is a valid schedule.

2.1.5 Bicriterion Approximations

When working with single criterion minimization problems, we say that an algorithm is a ρ-approximation algorithm if, in the worst case, the solution returned is within a multiplicative factor of ρ from optimal. In other words if we have an objective function $f(x)$ with optimal value OPT, then solution s is a ρ-approximation if it has the property that

$$f(S) \leq \rho OPT.$$

Note that this definition only makes sense for $\rho \geq 1$. Likewise we call an algorithm A a ρ-approximation for the f if even in the worst case the solution S returned by A is a ρ-approximation for OPT. For bicriterion optimization, Stein and Wein [25] introduced the following notation. Suppose we have criteria (A, B), then we say that S' is an (α, β)-schedule if S' is simultaneously at

most an α-approximation for A and a β-approximation for B. Similarly an (α, β)-approximation algorithm returns a schedule that, in the worst case, is within α of optimal for criterion A and β of optimal for criterion B. While most of the problems that we consider are minimization problems, suppose that A is a criterion that is optimized by being maximized. Then we say that a schedule or an algorithm is an (α, β)-approximation if it is simultaneously at least a $\frac{1}{\alpha}$-approximation for A and a β-approximation for B. Similarly if B is a maximization problem.

2.1.6 Negative Results

In this thesis we will be presenting some negative results and therefore it is necessary for us to define what we mean by a negative result or lower bound for a bicriterion scheduling problem. Suppose we are concerned with optimizing two criteria A and B. Then a negative result for the bicriterion problem $\alpha|\beta|(A, B)$ will be any result that shows that for some a and b instances exist for which no (x, y)-schedule exists with $x < a$ and simultaneously $y < b$.

2.2 Related Work

While some papers have explicitly set out to address bicriterion scheduling problems, other results have been the byproduct of work on single criterion scheduling problems. For instance, Graham showed in 1966 that using any list scheduling algorithm for the problem of scheduling jobs on parallel identical machines will produce a schedule of length at most twice optimal[7]. One list-scheduling algorithm schedules jobs according to non-increasing ratio of weight to processing time. This turns out to produce a schedule with average weighted completion time at most $(\sqrt{2}+1)/2$ times the optimal average weighted completion time[15]. In the special case where the weights are all equal this actually achieves the optimal value[5].

Motivated by the results of a report by Panwalkar, Dudek and Smith[20], which concluded that many managers use multiple objectives when creating schedules, Van Wassenhove and Gelders [27] looked at the problem of simultaneously minimizing the average completion time and lateness of a schedule on one machine. A set of schedules is said to be "Pareto optimal" if no schedule exists that is simultaneously better, in terms of both criteria, than any of the schedules in that set. Van Wassenhove and Gelders outlined a polynomial time algorithm that found the set of bicriterion schedules which are Pareto optimal for the problem of minimizing the average completion time and lateness of a schedule. Hoogeveen and Velde [12] extended these results to show that the number of distinct schedules in the set of Pareto optimal schedules is polynomial. Other papers dealing with Pareto optimal sets of schedules are due to Nelson et al.[19], Garey et al.[6], McCormick and Pinedo [18], and Hoogeveen [10, 11].

Other papers have approached bicriterion scheduling by fixing the value of one of the criteria and then optimizing the other criterion. Shmoys and Tardos [23] considered the problem of fixing the makespan of a schedule at twice optimal and then minimizing the total cost incurred by that schedule. Smith[24] studied the problem of minimizing the average completion time of a set of jobs on one machine while maintaining that the maximum lateness of the resulting schedule must be optimal. While these approaches were both successful, it has been shown by Hurkens and Coster that when scheduling jobs on unrelated parallel machines there exist instances for which all optimal average completion time schedules have a makespan of $\Omega(\log n)$ times optimal[13]. By considering schedules that were only close to optimal for both criteria, Chakrabarti, Phillips, Schulz, Shmoys, Stein and Wein[3] were able to outline general techniques for creating algorithms to optimize the makespan and average weighted completion time simultaneously of a set of jobs.

More recently, Stein and Wein[25] were able to improve these results by showing for a very general class of scheduling problems that bicriterion schedules exist that are simultaneous constant factor approximations for the problem of minimizing the average weighted completion time and makespan. Specifically Stein and Wein were able to show that for many scheduling problems there exists schedules which are simultaneously at most 1.88-approximations for both criteria. The proof of this existence result also suggested a method for creating bicriterion scheduling algorithms. Using known approximations algorithms for the single criterion schedule problem, they were able to create simple bicriterion scheduling algorithms for some more specific problems.

This thesis will take the approach of Stein and Wein[25] and consider the problem of showing the existence of schedules that are simultaneous small constant factor approximations for two criteria. The ideas presented by Stein and Wein will be examined in more detail in the next section along with some news methods which are part of joint work with J. Aslam, C. Stein and N. Young[1]. We will also consider lower bounds and algorithms that achieve our existence results. Torng and Uthaisombut[26] found a matching lower bound for the problem of simultaneously minimizing the average completion time and makespan of a set of jobs on one machine with release dates. They also used some properties of α-scheduling to create a deterministic polynomial time algorithm that meets these bounds. We will look at both their lower bound and algorithm more carefully in order to extend their results to other bicriterion scheduling problems.

3 General Techniques for Bicriterion Scheduling

In this section we will first look at a technique for proving the existence of good bicriterion schedules in a very general setting. The method we will use was introduced by Stein and Wein[25] in the context of simultaneously minimizing the makespan and average weighted completion time of a schedule. After introducing their technique we will look at some methods which provide a proof of a slightly stronger existence theorem for the general bicriterion scheduling problem $(C_{\max}, \sum w_j C_j)$. Next we will present a lower bound found by Torng and Uthaisombut for $1|r_j|(C_{\max}, \sum C_j)$. This result is of particular interest because it matches our improved existence result. Finally we will look at an algorithm, for $1|r_j|(C_{\max}, \sum w_j C_j)$. For the one machine case with release dates, this algorithm by Torng and Uthaisombut takes advantage of some of the properties of α-scheduling[4] to achieve the existence results. While the results presented in this section are concerned with the problem of simultaneously minimizing the makespan and average weighted completion time of a set of jobs, we will extend these results in Section 4 to other bicriterion scheduling problems.

3.1 Previous Existence Results

Stein and Wein[25] showed that for any scheduling problem there exists schedules which are simultaneously good approximations for the makespan and total weighted completion time. While the optimal average weighted completion time could be very long, the optimal makespan schedule will, by definition, be the shortest valid schedule. Their method relied on running the optimal average completion time schedule until some fraction of the weight had completed. Then the remaining jobs, which already had relatively late completion times in the optimal average completion time schedule, could be run according to their order in the optimal makespan schedule. Completing the remaining jobs according to the optimal makespan ordering allows us to limit the length of the resulting schedule. This clearly provides an upper bound on the total length of the schedule and therefore a handle for proving approximation results in terms of $C_{\max}^{\text{opt}}$, the optimal makespan. We will state this formally in the following lemma which is implicit in the work of Stein and Wein[25].

Lemma 1 *[25] For any scheduling problem, if we have two valid schedules S_1 and S_2, where the length of $S_1 = M$, then for any $\lambda \geq 0$ the length of S'= **Combine**$(S_1, S_2, \lambda M)$ will be at most $(1 + \lambda)M$.*

Proof: By the two conditions of "any" scheduling problem, we know that S' is a valid schedule. We also know that the length of S' will be at most $(1 + \lambda)M$ since we run S_2 for time λM and then finish the remaining jobs in at most M time. □

By providing an upper bound on the completion time of all jobs, we also bound the increase in a jobs completion time in terms of the optimal average completion time schedule. Therefore, assuming one had an optimal average weighted completion time schedule S^* and an optimal makespan schedule S^M, then one can use **Combine**(S^M, S^*, t) to create schedule S'. For a good choice of t, S' will run most of the critical jobs according to S^* and then quickly run the rest of the jobs according to S^M.

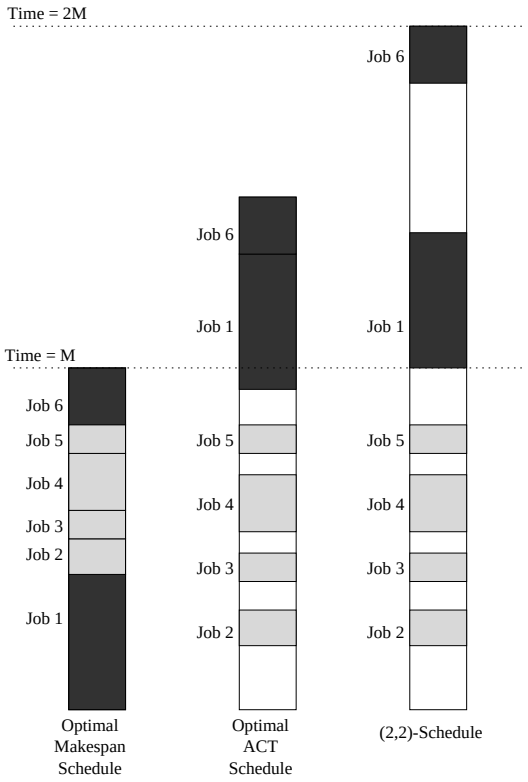

Figure 1: Construction of a (2,2) schedule. We create our (2,2) schedule by truncating the optimal average weighted completion time schedule at time M and appending the optimal makespan schedule to the truncated average completion time schedule.

In order to demonstrate this, Stein and Wein presented the following simple construction that shows the existence of a $(2, 2)$-schedule for any scheduling problem. Let $S' =$ **Combine**(S^M, S^*, M) where $M = C^{\text{opt}}_{\max}$ is the length of the optimal makespan schedule S^M. By construction, S' will be the schedule that runs schedule S^* until time M and then finishes the remaining jobs according to S^M. By Lemma 1, S' will have length at most $2M = 2C^{\text{opt}}_{\max}$. Now all that remains to show is that

the total weighted completion time of S' is at most twice optimal. Let C_j^* and C_j' be the completion times of job j in S^* and S', respectively. Since S' is the same as S^* until time M, any job j that completes before M in S^* will have $C_j' = C_j^*$. Job j with $C_j^* > M$ will run in the second half of S' according to S^M. Since we've already shown that the length of S' is at most $2M$ we know any job j which runs in the second section of S' have $C_j' \leq 2M \leq 2C_j^*$. Finally, because $C_j' \leq 2C_j^*$ for all jobs, $\sum w_j C_j' \leq 2 \sum w_j C_j^*$. Figure 1 shows how a $(2,2)$-schedule might be created for the single machine case with release dates. In Figure 1, notice that the length of the $(2,2)$-schedule is at most $2M$ even if we replace all jobs that are run in the first part of the schedule with idle time. In most cases we would expect that removing this idle time would actually make the resulting schedule better than a 2-approximation for the makespan.

In the $(2,2)$ construction, M is the breakpoint of S', or the time at which S' stops running according to S^* and starts scheduling jobs by their order in S^M. By considering more carefully the distribution of weight in the optimal average completion time schedule and choosing the best breakpoint out of 3 different possibilities, Stein and Wein were able to improve their results to show the existence of $(2, 1.735)$, $(1.785, 2)$ and $(1.88, 188)$-schedules for makespan and average weighted completion time. Figure 2 shows the same example as used in Figure 1 but now we consider 3 possible breakpoints aM, bM, and cM where $0 < a < b < c \leq 1$.

3.2 Improved Existence Results

A natural extension of these results is to consider infinitely many breakpoints and choose the best one according to the distribution of weight in the optimal average completion time schedule. In order to do this, we first observe that by normalizing the weights as necessary, we can characterize an average completion time schedule as a continuous probability density function(pdf). The advantage of probability density functions is that their continuous nature allows us to consider infinitely many breakpoints and then calculate an upper bound on the average completion time of the resulting schedule based on the best possible breakpoint for a given pdf. Since it is possible to map any average completion time schedule to a unique probability density function, the set of all probability density functions must contain the set of all possible average completion time schedules. Therefore finding an upper bound on the worst case pdf for our analysis also provides an upper bound in terms of schedules.

In this section we will present the details of this new approach as well as consider the applicability of these methods to other bicriterion scheduling problems.

3.2.1 The details of multiple breakpoints

In this section we will look at some of the details of analyzing multiple breakpoints. The work in this section was done jointly with J. Aslam, C Stein and N. Young and is also presented in [1]. As stated above, the first step in considering all possible breakpoints is to map average completion time schedules, which are discrete functions, to continuous probability density functions. First we notice that we can normalize the weights so that $\sum w_j C_j^* = 1$[1]. Although the total weighted completion time of a schedule is usually expressed as a sum over all jobs of the weighted completion time of each job, we notice that our worst case analysis is concerned only with the distribution of weight completing over time. This means that we can also compute the average weighted completion time of a schedule by taking the sum over all time of the amount of weight completing at that time times the completion time.

[1]To simplify notation we overload w_j to also denote the new normalized weight.

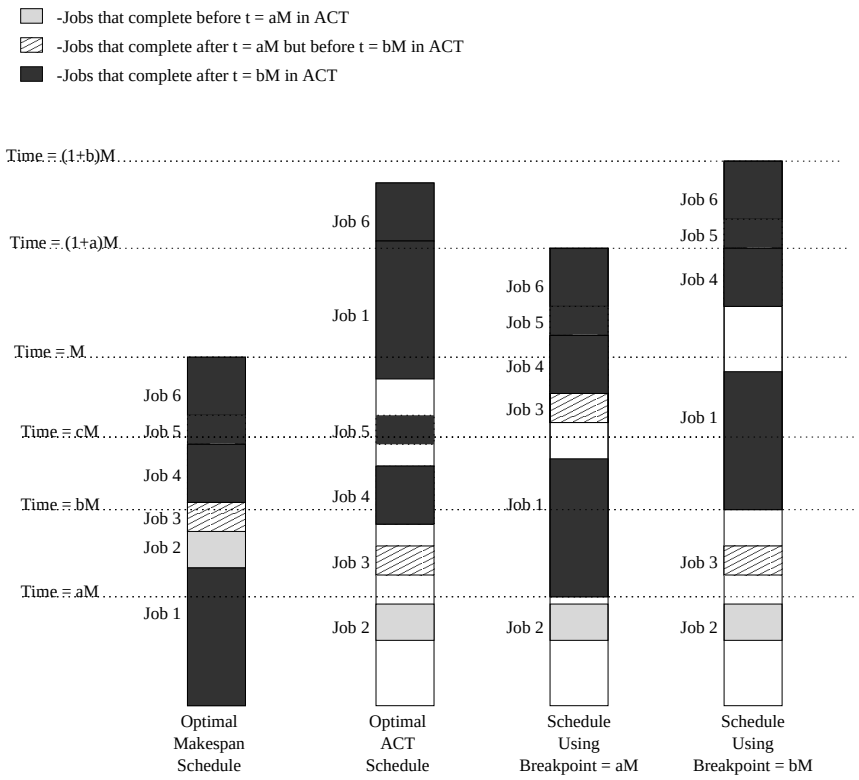

Figure 2: Using multiple breakpoints we can consider more than one time at which we might truncate the optimal average weighted completion time schedule (ACT). In this example we show the schedules created by choosing Time $= aM$ and Time $= bM$ as the breakpoint. We create our new schedules by truncating our optimal ACT schedule at aM and bM respectively and appending the optimal makespan schedule to the end of each truncated ACT schedule.

Mapping Schedules to Probability Density Functions: Let $g(z) = \sum_j w_j C_j^* \delta(C_j^* - z)$ where $\delta(\cdot)$ is Dirac's delta function. Dirac's delta function is defined to be the function that satisfies the conditions $\delta(x) = 0$ for all $x \neq 0$ and $\int_{-\infty}^{\infty} \delta(x)dx = 1$. This places an impulse at $x = 0$. Notice that this means that for a particular z the sum $\sum_j w_j C_j^* \delta(C_j^* - z)$ will actually only include those jobs with $C_j^* = z$ since for all other completion times not equal to z, $\delta(C_j^* - z) = 0$. Having transformed a schedule into a continuous function, we can then find the total completion time of that schedule by integrating the corresponding probability density function $g(z)$ over all time. When we integrate $g(z)$ over all time we get that the area under $g(z)$ at each time is equivalent to the amount of weight completing in the ACT schedule at time z times the completion time. Remember that we normalized our weights so that $\sum w_j C_j^* = 1$. With these normalized weights we know that $\int_0^{\infty} g(z)\, dz = \sum w_j C_j^* = 1$. Since $g(z) \geq 0$ and has integral 1, g is a pdf.

We can conceptualize this transformation from a discrete function to a continuous function using Dirac's delta function by thinking of creating, for every time z, a block of width $\frac{1}{10}$ and area $\sum_{j|C_j^*=z} w_j C_j^*$ centered at z. Consider the following 5 job example:

job	C_j^*	w_j
1	1	$\frac{1}{2}$
2	2	$\frac{1}{16}$
3	$\frac{1}{10}$	$\frac{10}{8}$
4	$\frac{1}{2}$	$\frac{1}{6}$
5	1	$\frac{1}{6}$

We now rewrite this table to show the schedule indexed by time as opposed to by job.

time t	Jobs w/ $C_j^* = t$	weight completing at t	$\sum_{j\|C_j^*=t}(w_j C_j^*)$
$\frac{1}{10}$	3	$\frac{10}{8}$	$\frac{1}{8}$
$\frac{1}{2}$	4	$\frac{1}{6}$	$\frac{1}{12}$
1	1 & 5	$\frac{1}{2} + \frac{1}{6} = \frac{2}{3}$	$\frac{2}{3}$
2	2	$\frac{1}{16}$	$\frac{1}{8}$

For this example, Figure 3 shows what this function looks like. As an impulse function, Dirac's delta function is the limit of this "block" as the width approaches 0 but the total area remains the same. Figure 4 shows how our example can be represented uses Dirac's delta function. Therefore, defining $g(z) = \sum_j w_j C_j^* \delta(C_j^* - z)$ gives us a continuous representation of an average completion time schedule.

Now that we have shown that we can map schedules to pdfs, we need to see how to analyze a breakpoint of a schedule using the corresponding probability density function. Although we are considering the problem of simultaneously minimizing the makespan and average completion time of our schedule, we will consider multiple breakpoints in a more general setting. Assume that we have an optimal average completion time S^* and we have some other schedule S^T of length T. Suppose we are considering a breakpoint of αT, where $0 < \alpha \leq 1$. Then we define $S' =$ **Combine**$(S^T, S^*, \alpha T)$. By construction, S' will run S^* until time αT followed by the remaining jobs according to S^T. By Lemma 1, the makespan of S' is at most $(1 + \alpha)T$. The completion time of any job that completes before αT in S^* will be unchanged in S'. For jobs where $C_j^* > \alpha T$, which will be run in the second section of S', $C_j' \leq (1 + \alpha)T$. This means that for any job with $C_j^* = z$, the completion time of j in S' is at most $\frac{(1+\alpha)T}{z}$ times optimal. Therefore we can express the average completion time of S' as

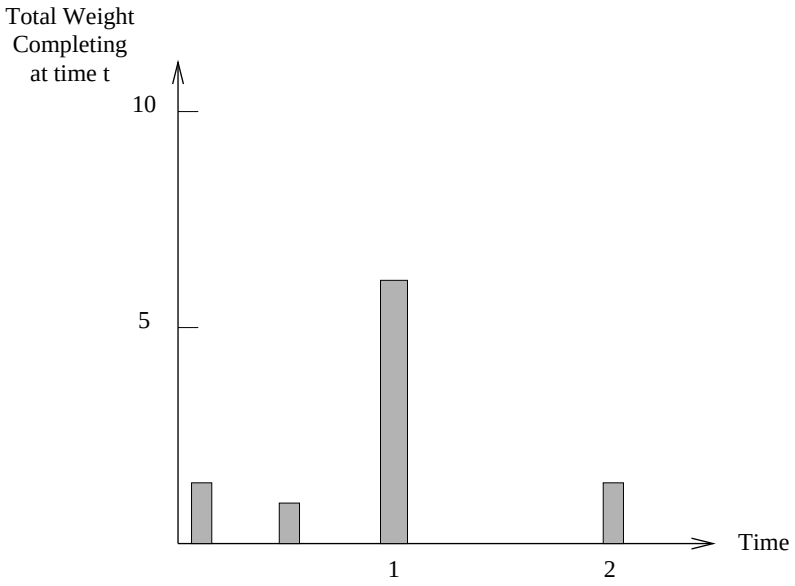

Figure 3: Representation of the average weighted completion time using "blocks" of width $\frac{1}{10}$.

$$\begin{aligned}\sum w_j C_j^{S'} &\leq \int_0^{\alpha T} g(z)\,dz + \int_{\alpha T}^{\infty} \frac{(1+\alpha)T}{z} g(z)\,dz \\ &= \int_0^{\infty} g(z)\,dz + \int_{\alpha T}^{\infty} \frac{(1+\alpha)T - z}{z} g(z)\,dz. \\ &= 1 + \int_{\alpha T}^{\infty} \frac{(1+\alpha)T - z}{z} g(z)\,dz.\end{aligned}$$

We now want to choose the the best breakpoint to minimize the value of the above expression. Notice that this corresponds to choosing the breakpoint that minimizes the integral $\int_{\alpha T}^{\infty} \frac{(1+\alpha)T-z}{z} g(z)\,dz$. In order to insure that the makespan of S' is still small we will restrict α to be in $[0, \rho]$ where $0 < \rho \leq 1$. Then we are guaranteed that any breakpoint we choose to minimize the average completion time will result in a schedule of length no more that $(1+\rho)T$. For a particular schedule $g(z)$, this step corresponds to the following calculation

$$\min_{0 \leq \alpha \leq \rho} \int_{\alpha T}^{\infty} \frac{(1+\alpha)T - z}{z} g(z)\,dz.$$

Our goal is to find an upper bound on this value for all possible schedules. Since the set of all pdfs includes all possible schedules, finding an upper bound on

$$\max_{g} \min_{0 \leq \alpha \leq \rho} \int_{\alpha T}^{\infty} \frac{(1+\alpha)T - z}{z} g(z)\,dz,$$

where $g(z)$ is a pdf over $[0, \infty)$ yields an upper bound on the average weighted completion time of the worst case schedule as well. Using the change of variables $z = Tx$ we can rewrite this as

$$\int_{\alpha}^{\infty} \frac{(1+\alpha)T - Tx}{Tx} g(Tx)T\,dx = \int_{\alpha}^{\infty} \frac{1+\alpha-x}{x} g(Tx)T\,dx.$$

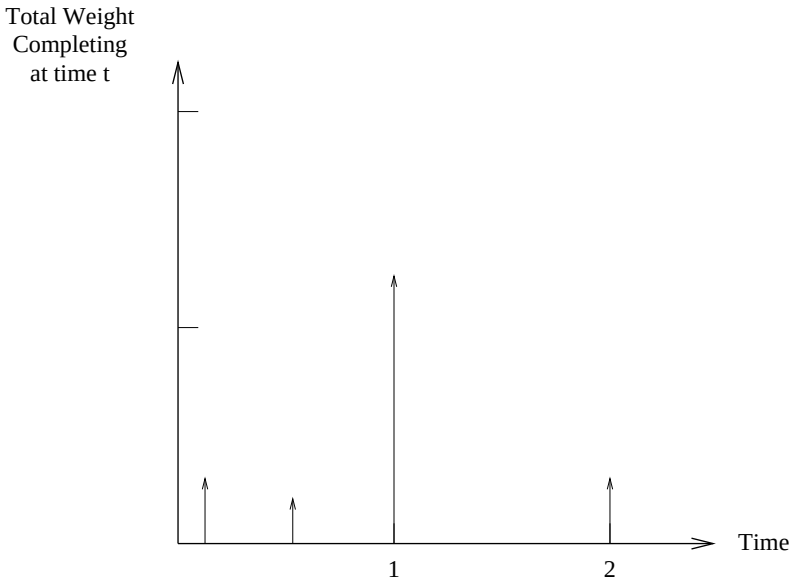

Figure 4: Representation of the average weighted completion time using Dirac's delta function.

Now we show that $f(x) = T \cdot g(Tx)$ is a distribution if g is a distribution by showing that $\int_0^\infty f(x)\,dx = 1$.

$$\begin{aligned}\int_0^\infty f(x)\,dx &= \int_0^\infty T \cdot g(Tx)\,dx \\ &= \int_0^\infty g(Tx)\,dTx \\ &= \int_0^\infty g(y)\,dy \\ &= 1\end{aligned}$$

So by considering all possible distributions f we notice that our previous problem is equivalent to

$$\max_f \min_{0\le\alpha\le\rho} \int_\alpha^\infty \frac{1+\alpha-x}{x} f(x)\,dx. \tag{1}$$

Now we will show how to solve this problem. Suppose that $f_{\max}$ is the pdf that achieves the maximum value, $c_{\max}$, of this integral. Notice that finding an upper bound on this integral for all pdfs is equivalent to finding an upper bound on $c_{\max}$. We observe that $f_{\max}$ must have the following characteristics.

Lemma 2 *If $f_{\max}$ is the distribution that maximizes expression 1, then $\int_\rho^\infty f_{\max}(x)\,dx > 0$.*

Proof: By contradiction, if we set $\alpha = \rho$ then, $\int_\alpha^\infty \frac{1+\alpha-x}{x} f(x)\,dx = 0$. Since simple distributions, such as the uniform distribution over $[0,2]$, exist for which $\int_\alpha^\infty \frac{1+\alpha-x}{x} f(x)\,dx > 0$ this contradicts the claim that $f_{\max}$ is the distribution achieving the maximum value of expression 1. □

Lemma 2 tells us that $\int_\rho^\infty f_{\max}(x)\,dx > 0$. By looking at the weighting function in the integral, we notice that as x increases $\frac{1+\alpha-x}{x}$ strictly decreases for $x \ge 0$.

Lemma 3 *If $f_{\max}$ is the distribution maximizing expression 1,then $\int_{\rho^+}^{\infty} f_{\max}(x)\,dx = 0$.*

Proof: By contradiction, assume that $\int_{\rho^+}^{\infty} f_{\max}(x)\,dx = c$ for some $c > 0$. Create the distribution f' by moving all weight c to ρ^+. By the preceding discussion, $\int_{\alpha}^{\infty} \frac{1+\alpha-x}{x} f'(x)\,dx > \int_{\alpha}^{\infty} \frac{1+\alpha-x}{x} f_{\max}(x)\,dx$ which is a contradiction. □

By Lemma 2 we know that $\int_{\rho}^{\infty} f_{\max}(x)\,dx > 0$ and by Lemma 3 we can say that $\int_{\rho^+}^{\infty} f_{\max}(x)\,dx = 0$. This give us the following corollary.

Corollary 4 *The distribution $f_{\max}$ contains a strictly positive point mass at $x = \rho$ and is 0 for all $x > \rho$.*

At this point we know that $f_{\max}$ will have the following form

$$f_{\max}(x) = \begin{cases} f(x) & 0 \le x < \rho \\ k \cdot \delta(0) & x = \rho \\ 0 & x > \rho. \end{cases} \tag{2}$$

Now we will use this information about $f_{\max}(x)$ to find an upper bound on $c_{\max}$. First we need the following definition.

Definition 5 *If $f_{\max}$ is the distribution maximizing expression 1, then*

$$A(\alpha) = \int_{\alpha}^{\rho} \frac{1-x+\alpha}{x} f_{\max}(x)dx.$$

By definition, $c_{\max}$ is the minimum value of $A(\alpha)$ for $\alpha \in [0, \rho]$. Therefore

$$\int_0^{\rho} c_{\max} d\alpha \le \int_0^{\rho} A(\alpha)d\alpha.$$

Now consider

$$\int_0^{\rho} e^{\alpha} A(\alpha)d\alpha.$$

Since $e^{\alpha} \ge 0$ for all $\alpha \in [0, \rho]$,

$$\int_0^{\rho} e^{\alpha} c_{\max} d\alpha \le \int_0^{\rho} e^{\alpha} A(\alpha)d\alpha.$$

If we evaluate the left hand side we get

$$c_{\max}(e^{\rho} - 1) \le \int_0^{\rho} e^{\alpha} A(\alpha)d\alpha.$$

Now we substitute the definition of $A(\alpha)$ into the right hand side.

$$c_{\max}(e^{\rho} - 1) \le \int_0^{\rho} e^{\alpha} \left[\int_{\alpha}^{\rho} \frac{1-x+\alpha}{x} f_{\max}(x)dx \right] d\alpha.$$

Next we switch the order of integration

$$c_{\max}(e^{\rho} - 1) \le \int_0^{\rho} \left[\int_0^{x} e^{\alpha} \left(\frac{1-x+\alpha}{x} \right) f_{\max}(x)d\alpha \right] dx.$$

Since $f_{\max}$ does not depend on α,

$$c_{\max}(e^{\rho}-1) \leq \int_0^{\rho} \left[\int_0^{x} e^{\alpha} \left(\frac{1-x+\alpha}{x} \right) d\alpha \right] f_{\max}(x)dx$$

Now we separate the inner integral into two terms

$$c_{\max}(e^{\rho}-1) \leq \int_0^{\rho} \left[\frac{1-x}{x} \int_0^{x} e^{\alpha} d\alpha + \frac{1}{x} \int_0^{x} \alpha e^{\alpha} d\alpha \right] f_{\max}(x)dx.$$

Evaluating the inside integrals and rearranging terms

$$\begin{aligned} c_{\max}(e^{\rho}-1) &\leq \int_0^{\rho} \left[\frac{1-x}{x}(e^{x}-1) + \frac{1}{x}(xe^{x}-e^{x}) \right] f_{\max}(x)dx \\ &= \int_0^{\rho} \left[\frac{e^{x}}{x} - e^{x} - \frac{1}{x} + 1 + e^{x} - \frac{e^{x}}{x} + \frac{1}{x} \right] f_{\max}(x)dx \\ &= \int_0^{\rho} f_{\max}(x)dx \\ &= 1. \end{aligned}$$

Therefore

$$c_{\max} \leq \frac{1}{e^{\rho}-1}.$$

Lemma 6 *For any pdf f,*

$$\max_{f} \min_{0 \leq \alpha \leq \rho} \int_{\alpha}^{\infty} \frac{1+\alpha-x}{x} f(x)dx \leq \frac{1}{e^{\rho}-1}.$$

Proof: The upper bound follows from the discussion above. □

Since the average completion time of our new schedule is at most

$$1 + \max_{f} \min_{0 \leq \alpha \leq \rho} \int_{\alpha}^{\infty} \frac{1+\alpha-x}{x} f(x)dx \leq 1 + \frac{1}{e^{\rho}-1} = \frac{e^{\rho}}{(e^{\rho}-1)},$$

we know that for any scheduling problem if we choose the best breakpoint, αT, between $[0, \rho T]$ then the average completion time of our new schedule will be at most $\frac{e^{\rho}}{(e^{\rho}-1)}$ times the optimal average completion time.

Lemma 7 *For any $\rho \in [0,1]$, for any scheduling problem, if we have an optimal average completion time schedule S^{*} and another schedule S^{T} where the length of S^{T} is $T = \max C_j^{T}$, then there exists a schedule which is an $\left(\frac{e^{\rho}}{e^{\rho}-1}\right)$-approximation for $\sum w_j C_j$ and has length at most $(1+\rho)T$.*

Proof: This is a direct result of Lemmas 1 and 6. □

If we now consider the special case when S^{T} is actually S^{M}, the optimal makespan schedule of length M, we arrive at the following bicriterion existence result for the makespan and average completion time of a set of jobs.

Theorem 8 *For any $\rho \in [0,1]$, for any scheduling problem, there exists a $\left(1+\rho, \frac{e^{\rho}}{e^{\rho}-1}\right)$-approximation for $(C_{\max}, \sum w_j C_j)$.*

Proof: This is a direct result of Lemma 7.

□

Using Theorem 8 and picking particular values of ρ we arrive at the following corollary which gives improved bounds compared to those presented by Stein and Wein[25].

Corollary 9 *For any scheduling problem, there exists a* $(2, 1.582)$*-schedule, a* $(1.695, 2)$*-schedule and a* $(1.806, 1.806)$*-schedule for* $(C_{\max}, \sum w_j C_j)$.

Now that we have found an upper bound on $c_{\max}$ we will also show that a pdf exists which actually achieves this upper bound. Let $h_{\max}$ be a pdf for which the optimal value of

$$\max_{f} \min_{0 \le \alpha \le \rho} \int_{\alpha}^{\infty} \frac{1+\alpha-x}{x} f(x)\, dx \tag{3}$$

is $c_{\max}$. We know that $h_{\max}$ has the form

$$h_{\max}(x) = \begin{cases} h(x) & 0 \le x < \rho \\ k \cdot \delta(0) & x = \rho \\ 0 & x > \rho. \end{cases} \tag{4}$$

Next we need to determine $h_{\max}(x)$ for the region $[0, \rho)$ and k. We do this by looking at the function $B(\alpha) = \int_{\alpha}^{\infty} \frac{1+\alpha-x}{x} h_{\max}(x)\, dx$. We use our knowledge from expression 4 about $h_{\max}$ to first write $B(\alpha)$ as

$$\begin{aligned} B(\alpha) &= \int_{\alpha}^{\infty} \frac{1+\alpha-x}{x} h(x)\, dx \\ &= \int_{\alpha}^{\rho^-} \frac{1+\alpha-x}{x} h(x)\, dx + \frac{1+\alpha-\rho}{\rho} k \\ &= \int_{\alpha}^{\rho^-} \frac{1-x}{x} h(x)\, dx + \alpha \int_{\alpha}^{\rho^-} \frac{h(x)}{x} dx + \frac{1+\alpha-\rho}{\rho} k. \end{aligned}$$

Then we take the derivative[2] of this with respect to α

$$\begin{aligned} B'(\alpha) &= -\frac{1-\alpha}{\alpha} h(\alpha) - \alpha \frac{h(\alpha)}{\alpha} + \int_{\alpha}^{\rho^-} \frac{h(x)}{x} dx + \frac{k}{\rho} \\ B'(\alpha) &= -\frac{h(\alpha)}{\alpha} + \int_{\alpha}^{\rho^-} \frac{h(x)}{x} dx + \frac{k}{\rho}. \end{aligned}$$

Now we assume that $B(\alpha)$ is constant over all α and therefore the derivative with respect to α of $B(\alpha)$ is 0. While this assumption simplifies the calculation considerably, it does not weaken the result. Therefore

$$0 = -\frac{h(\alpha)}{\alpha} + \int_{\alpha}^{\rho^-} \frac{h(x)}{x} dx + \frac{k}{\rho}.$$

Next we substitute $g(t) = h(t)/t$ and $G = \int g$ and say that the above expression holds for $h(t)$ if and only if

[2] We assume that $B'(\alpha)$ is defined

$$
\begin{aligned}
g(\alpha) - \int_{\alpha}^{\rho^-} g(x)\,dx &= \frac{k}{\rho} && (5)\\
\Leftrightarrow \quad G'(\alpha) - G(\rho^-) + G(\alpha) &= \frac{k}{\rho} && (6)\\
\Leftrightarrow \quad G'(\alpha) + G(\alpha) &= \frac{k}{\rho} + G(\rho^-). && (7)
\end{aligned}
$$

We notice that equation 7 is a differential equation of the form

$$y' + y = c_1$$

and therefore the solution must look like $y(x) = c_1 + c_2 e^{-x}$. We use this to rewrite equation 7 as

$$G(x) = \frac{k}{\rho} + G(\rho^-) + c_2 e^{-x}.$$

The above expression holds for all values of x including ρ^-. We can solve for c_2 by setting $x = \rho^-$. This yields $c_2 = -\frac{e^\rho k}{\rho}$ and therefore

$$G(x) = \frac{k}{\rho} + G(\rho^-) - \frac{e^\rho k}{\rho} e^{-x}.$$

With this expression for G we can use the fact that $G = \int g$,

$$
\begin{aligned}
g(x) &= G'(x)\\
&= \frac{e^\rho k}{\rho} e^{-x}.
\end{aligned}
$$

Next, to arrive at an equation for $h(x)$ we substitute $h(t) = t \cdot g(t)$ and obtain

$$h(x) = \frac{e^\rho k}{\rho} x e^{-x}. \tag{8}$$

Now that we have an expression for $h_{\max}(x)$ in the range $[0, \rho)$, we find k by noticing that $\int_0^\infty h_{\max}(x)\,dx = \int_0^{\rho^-} h(x)\,dx + k$ and because $h_{\max}$ is a pdf, $\int_0^{\rho^-} h(x)\,dx + k = 1$. Solving for k we get

$$
\begin{aligned}
k &= 1 - \int_0^{\rho^-} h(x)\,dx\\
&= 1 - \int_0^{\rho^-} \frac{e^\rho k}{\rho} x e^{-x}\,dx\\
&= 1 - \left[-\frac{e^\rho k}{\rho} e^{-x}(1+x) \Big|_0^{\rho^-} \right]\\
&= 1 + \frac{e^\rho k}{\rho} \left((1+\rho)e^{-\rho} - 1 \right)\\
&= 1 + \frac{k}{\rho}(1 + \rho - e^\rho).
\end{aligned}
$$

The last equation yields $k = \rho/(e^\rho - 1)$. Therefore, by Equations 4 and 8 the form of $h_{\max}$ is

$$h_{\max}(x) = \begin{cases} \frac{e^\rho}{e^\rho - 1} x e^{-x} & 0 \le x < \rho \\ \frac{\rho}{e^\rho - 1}\delta(0) & x = \rho \\ 0 & x > \rho. \end{cases}$$

Since we assumed that $B(\alpha) = c_{\max}$ for all α we can compute $c_{\max}$ using $B(\rho)$ and

$$c_{\max} = B(\rho) = \frac{1 + \rho - \rho}{\rho} k = \frac{1}{e^\rho - 1}.$$

Therefore $h_{\max}$ is a pdf that achieves the maximum value of

$$\max_f \min_{0 \le \alpha \le \rho} \int_\alpha^\infty \frac{1 + \alpha - x}{x} f(x)\, dx. \tag{9}$$

3.2.2 Applicability of Existence Results to Other Criteria

The ideas used to arrive at Theorem 8 turn out to apply to many other bicriterion scheduling problems. Since delivery times are positive, L_j will always be at least as large as C_j and therefore in section 4.1 we will be able to prove that **Combine**(S_1, S_2, t) also produces a schedule that is a constant factor approximation for the maximum lateness of schedule S_1. We will also relate the average completion time of any schedule to the average lateness and then apply Theorem 8 to the problem of minimizing the makespan and average lateness of a set of jobs.

Lemma 6 will be used when exploring the relationship between the makespan of a schedule and the average flow time. While this problem can be solved using a similar derivation, we will obtain the results in section 4.2 by noticing that after the release of the last job, minimizing the average weighted flow time of the remaining jobs is equivalent to minimizing the average weighted completion time of those jobs with respect to the last release date. Therefore by truncating at some point after $C_{\max}^{\text{opt}}$, instead of before it, we can directly apply these results to the average flow time problem.

Finally, while these methods are very general, we will also present instances in which they fail. The best example of this is when we consider the case of the maximum flow time of a schedule verses the average flow time of the same schedule. In this case we can show that instances exist for which no one schedule is simultaneously a constant factor approximation for both criteria.

3.3 Lower Bounds and Algorithms for Special Cases

In this section we will look at two results from Torng and Uthaisombut[26]. Both results deal with the special case of scheduling unweighted jobs with release dates on one machine to minimize the makespan and average completion time simultaneously. This problem is denoted $1|r_j|(C_{\max}, \sum C_j)$. They present a deterministic algorithm that achieves the existence results derived in the last section and prove a lower bound that matches the upper bound from Theorem 8. We will look at the lower bound and then explore their algorithm.

3.3.1 Lower Bound for $1|r_j|(C_{\max}, \sum C_j)$

The work presented in this section follows closely from [26]. In this section we will refer to the average completion time of the schedule and prove a lower bound for $1|r_j|(C_{\max}, \frac{1}{n}\sum C_j)$. As stated earlier, since $\frac{1}{n}\sum C_j$ and $\sum C_j$ are equivalent object functions except for the scaling factor $\frac{1}{n}$, this also proves a lower bound for $1|r_j|(C_{\max}, \sum C_j)$.

Theorem 10 *[26] For $0 < \beta < 1$, there exists an (infinite-size) instance such that no schedule is an (x, y)-schedule with $x < 1 + \beta$ and $y < \frac{e^\beta}{e^\beta - 1}$ for $(C_{\max}, \sum C_j)$.*

Proof: The intuition behind the instance that produces this lower bound is to use the idea of probability density functions to create a scheduling problem where small jobs are released so that if they are run immediately the average completion time will be $\frac{e^\beta}{e^\beta - 1}$. Describing the release of the small jobs using probability density functions allows us to create an instance where the number of zero size jobs, jobs with $p_j = 0$, approaches infinity. Since we will be evaluating the average completion time, we can include 1 unit-size job and claim that as the number of jobs approaches infinity the average completion time of any schedule need only be calculated based on the completion times of the zero sized jobs. If this relatively large job is released at time $t = 0$, then either it can be run right away, displacing the small jobs, or it can wait until time β and begin processing then. By proving that these two schedules are the schedules with the best bicriterion bounds for this problem instance, we arrive at the result.

Now we will present the details of the proof. We have $n + 1$ total jobs, n of which are jobs of size 0 and the other one job is of size 1. The job of size 1 is released at time 0. The following pdf f specifies, for some fixed β where $0 < \beta < 1$, the release dates and therefore also optimal average completion times of the n zero sized jobs:

$$f(x) = \begin{cases} \frac{1}{e^x} & 0 \le x < \beta \\ \delta(0)\frac{1}{e^\beta} & x = \beta \\ 0 & x > \beta. \end{cases}$$

Given this specification for $f(x)$, Torng and Uthaisombut then noticed that any schedule can be described by the parameter s representing the starting time of the job of unit size. All zero size jobs that are released before s will run at their release date and therefore attain their optimal completion time in terms of the ACT schedule. Any job released after the job of size 1 is started will need to wait until it completes before being scheduled. Following their notation, we will let $C^s_{\max}$ and C^s_{avg} denote the makespan and average completion time of a schedule created with the unit-sized job starting at s. Notice that the only schedules of interest occur with $0 \le s \le \beta$. We know that the makespan will be $C^s_{\max} = (1 + s)$. Since the optimal makespan is 1, we have that $C^s_{\max} = (1+s)C^{\text{opt}}_{\max}$. Next we analyze the effect of s on the average completion time of the schedule. We will ignore the completion time of the job of size 1 since as n approaches infinity, it becomes negligible.

$$\begin{aligned} C^s_{\text{avg}} &= \int_0^s x f(x)dx + (s+1)(1 - \int_0^s f(x)dx) \\ &= \int_0^s \frac{x}{e^x}dx + (s+1)(1 - \int_0^s \frac{1}{e^x}dx) \\ &= \begin{cases} 1 & \text{if } 0 \le s < \beta \\ \frac{e^\beta - 1}{e^\beta} & \text{if } s = \beta \end{cases} \end{aligned}$$

From the above calculation, it is clear that the optimal average completion time is $\frac{e^\beta - 1}{e^\beta}$ which is achieved by a schedule with makespan of $1 + \beta$. We also notice that any schedule starting the unit sized job before time $1+\beta$ will have an average completion time of 1. Therefore all such schedules are $\frac{e^\beta}{e^\beta - 1}$-approximations for the average completion time. On the other hand, the optimal makespan schedule starts the job of size 1 at time 0 and has a makespan of 1. Since all schedules that start

the unit size job before time β are $\frac{e^\beta}{e^\beta-1}$-approximations for the optimal average completion time and all other schedules must start the unit size job after time β and are therefore at least $1+\beta$ times the optimal makespan we arrive at the theorem. □

3.3.2 A Deterministic Algorithm for $1|r_j|(C_{\max}, \sum C_j)$

The previous two sections have shown that for the case of scheduling unweighted jobs on 1 machine with release dates, the best possible bicriterion schedules for some instances will be a $(1+\beta, \frac{e^\beta}{e^\beta-1})$-approximation of $(C_{\max}, \sum C_j)$. Torng and Uthaisombut, using ideas from α-scheduling, showed first a randomized algorithm that achieves these bounds and then described a deterministic polynomial time algorithm with the same bounds. To understand their algorithm, it will first be useful to describe the general technique of α-scheduling.

The idea behind α-scheduling is to use an optimal preemptive schedule to obtain an ordering of jobs for the non-preemptive case. Consider the schedule P created by scheduling jobs preemptively by the shortest remaining precessing time($\mathcal{SRPT}$). Baker[2] showed in 1974 that $\mathcal{SRPT}$ returns an exact solution for $1|r_j, prmt|\sum C_j$. For some $0 < \alpha \leq 1$, let $C_j^P(\alpha)$ be the time at which an α-fraction of j completes. A non-preemptive schedule can then be created by list scheduling jobs according to non-decreasing $C_j^P(\alpha)$. The idea of scheduling jobs according to the ordering of fraction completion times in a preemptive schedule appears in Phillips, Stein and Wein[21] and Hall, Schulz, Shmoys and Wein [9]. Chekuri et al.[4] extended these results by introducing the idea of choosing α randomly. The intuition behind a randomized choice of α is that while a particular choice of α could be bad for one job, it won't be bad for every job. Chekuri et al. [4] provide the following two lemmas for this approach to α-scheduling.

Lemma 11 *[4] The makespan of any α-schedule is at most $1+\alpha$ times the optimal makespan.*

The following lemma refers to the randomized algorithm $\mathcal{RANDOM}$ which chooses α randomly from a distribution $f(x)$ and then uses that α to create a non-preemptive α-schedule.

Lemma 12 *[4] The expected average completion time of $\mathcal{RANDOM}$ is at most $1+\delta$ times the optimal preemptive average completion time where*

$$\delta = \max_{0<t\leq 1} \int_0^t \frac{1+\alpha-t}{t} f(\alpha) d\alpha.$$

Choosing α randomly from the probability distribution

$$f(\alpha) = \begin{cases} \frac{e^\alpha}{e^\beta-1} & 0 \leq \alpha < \beta \\ 0 & \alpha > \beta, \end{cases}$$

Torng and Uthaisombut create the randomized algorithm $\mathcal{RANDOM} - \beta$.

Theorem 13 *[26] For $0 < \beta \leq 1$, $\mathcal{RANDOM} - \beta$ is a randomized $(1+\beta, \frac{e^\beta}{e^\beta-1})$-approximation algorithm for $1|r_j|(C_{\max}, \sum C_j)$.*

Finally using the observation by Chekuri et al.[4] that $\mathcal{SRPT}$ creates a preemptive schedule with at most $n-1$ preemptions, we know that there are at most $n-1$ interesting choices of α. This means that there are at most n distinct non-preemptive schedules that can be derived by using α-scheduling to convert the preemptive schedule to a non-preemptive schedule. Chekuri et

al.[4] use this to show that by searching all n possible schedules and choosing the best one, we can in polynomial time, find a non-preemptive schedule that matches the expected bounds for the randomized algorithm. Torng and Uthaisombut use these ideas to create $\mathcal{BEST}$-β, the deterministic algorithm that achieves the expected bounds of $\mathcal{RAND} - \beta$.

In the same way that many of the existence results will prove useful when examining other criteria, the ideas behind $\mathcal{BEST}$-β will also turn out to provide similar results for the one machine case with release dates and unweighted jobs. We will look at applying $\mathcal{BEST}$-β to problems involving lateness in Section 4.1

4 Bounds for other criteria

We will now look at extending the ideas from the previous section to other bicriterion scheduling problems. We will use the idea of creating a composite schedule from two optimal schedules to prove existence theorems for models involving lateness, flow time and the number of on-time jobs.

For many of the bicriterion scheduling problems involving lateness we will be able to show a connection to objectives involving only the completion time. When we consider the flow time of jobs we will look at five bicriterion objective problems. First we will consider the problem of simultaneously minimizing the schedule length and average flow time. For this case we will show that if we choose our breakpoint later in the average flow time schedule than we had in the average completion time schedule, then this problem reduces to the problem of simultaneously minimizing the schedule length and average completion time. Next we consider the problem of minimizing the maximum flow time and average completion time. We will still use the idea of composite schedules but will have to modify our analysis to provide an upper bound now on the maximum flow time as opposed to the total schedule length. Next we will present an example for the bicriterion schedule problem concerned with minimizing the maximum flow time and average flow time to show that no schedule exists which is a simultaneous constant factor approximation for both criteria. We will then show that for the problem of minimizing the maximum lateness and average weighted flow and also for the problem of minimizing the maximum flow time and average weighted lateness of a set of jobs, we can prove directly from our earlier results that schedules exist which are simultaneously a constant factor approximation for both criteria. The last objective function we will consider is the number of on-time jobs. We will prove that for every instance there exists $(2, 2)$-schedules for the makespan and the number of on-time jobs. We will then use this result to arrive at a proof that $(2, 2)$-schedules exist for maximum lateness and the number of on-time jobs. Finally we will prove our second negative result which shows that instances exist for which no schedule is simultaneously a constant factor approximation for the maximum flow time and the number of on-time jobs.

4.1 Lateness

Previously we have looked at models in which our objective was to simultaneously minimize the maximum completion time and the total completion time of all jobs in a particular schedule. Likewise, two objectives that arise when we consider problems where each job j has a deadline d_j associated with it, are the maximum lateness over all jobs and the total(or average) lateness of a particular schedule. As described in Section 2.1.1, we will be working with the delivery time model of lateness. In this model, each job has a positive delivery time q_j which represents an amount of time which j must wait after it has completed. We define the lateness of j to be $L_j = C_j + q_j$. After presenting some existence results for $(L_{\max}, \sum w_j C_j)$, $(C_{\max}, \sum w_j L_j)$ and $(L_{\max}, \sum w_j L_j)$,

we will see how the lower bounds and algorithmic results from Torng and Uthaisombut[26] apply to lateness.

4.1.1 Existence Results

We will begin by looking at the problem of simultaneously minimizing the maximum lateness and average weighted completion time of a schedule.

Maximum Lateness and Average Weighted Completion Time As with our previous results, we start by assuming that we have an optimal maximum lateness schedule S^L and an optimal average weighted completion time schedule S^*. We will let C_j^L be the completion time of job j in S^L and $C_{\max}^L$ be the length of S^L. Then we consider the schedule $S^\lambda = \textbf{Combine}(S^L, S^*, \lambda C_{\max}^L)$. By construction, S^λ runs S^* until time $\lambda C_{\max}^L$, for $0 < \lambda \leq 1$, then finishes the remaining jobs according to S^L. As with our previous results, we will first upper bound the maximum lateness of any schedule created using a breakpoint of $\lambda C_{\max}^{\text{opt}}$ where $0 \leq \lambda \leq \rho$. Given this upper bound, we will then choose the best breakpoint that minimizes the average weighted completion time of the resulting schedule. Since we are using a similar construction, once we have an upper bound on the maximum lateness of S^λ we can apply Lemma 1 to arrive at a bound for $\sum w_j C_j^\lambda$.

Lemma 14 *For any scheduling problem, if we have valid schedules S_1 and S_2, where the length of S_1 is $C_{\max}^1$ and the maximum lateness of any job in S_1 is L, then any schedule $S^\lambda = \textbf{Combine}(S_1, S_2, \lambda C_{max}^1)$ will have a makespan of at most $(1+\lambda)C_{\max}^1$ and maximum lateness of at most $(1+\lambda)L$.*

Proof: We know by Lemma 1 that the length of S^λ will be $C_{\max}^\lambda \leq (1+\lambda)C_{\max}^1$. To find an upper bound on the maximum lateness of any job in S^λ we need to consider both the maximum lateness of the jobs run in the second half of S^λ and also the maximum lateness of the jobs run according S_2. We begin by looking at the jobs that complete after $\lambda C_{\max}^L$ in S_2. This set of jobs is run in S^λ according to their order in schedule S_1. Therefore we know that in the worst case each job j will complete C_j^1 time units after we start running S_1. We begin running S_1 at time $\lambda C_{\max}^1$ and as a result

$$C_j^\lambda \leq \lambda C_{\max}^1 + C_j^1. \tag{10}$$

By the definition of L_j we know, in general, that the lateness of job j will be at least as large as the completion time of j. In particular we know that

$$L \geq C_{\max}^1,$$

and by definition

$$L = \max_j (C_j^1 + q_j).$$

Combing these with inequality 10, we get

$$\begin{aligned} L_j^\lambda &= C_j^\lambda + q_j \\ &\leq \lambda C_{\max}^1 + C_j^1 + q_j \\ &\leq (1+\lambda)L. \end{aligned}$$

Therefore we can say that any job j with $C_j^2 > \lambda C_{\max}^1$ will have $L_j^\lambda \leq (1+\lambda)L$.

Now we consider the jobs which complete before the breakpoint. These jobs are scheduled in S^λ according to schedule S_2. If job j is run according to S_2 then we have that

$$\begin{aligned} L_j^\lambda &= C_j^2 + q_j \\ &\leq \lambda C_{\max}^1 + q_j \\ &\leq \lambda C_{\max}^1 + C_j^1 + q_j \\ &\leq (1+\lambda)L. \end{aligned}$$

Therefore any job j with $C_j^2 \leq \lambda C_{\max}^1$ will have $L_j^\lambda \leq (1+\lambda)L$. □

At this point we know that we can truncate our average completion time schedule at $\lambda C_{\max}^*$ and by Lemma 14 the resulting schedule will have a maximum lateness of at most $(1+\lambda)$ times optimal. Therefore for a particular upper bound $\rho \in [0,1]$ on λ we can then use Lemma 7 to show that for any schedule we can find a schedule that is simultaneously a most a $(1+\rho)$-approximation for the maximum lateness and an $\frac{e^\rho}{e^\rho - 1}$-approximation for the average completion time.

Theorem 15 *For any $\rho \in [0,1]$, for any scheduling problem, there exists a $\left(1+\rho, \frac{e^\rho}{(e^\rho-1)}\right)$-schedule for the delivery time model of maximum lateness and average weighted completion time.*

Proof: The result follows directly from Lemma 7 and Lemma 14. □

In deriving Theorem 15 we see how to apply Lemma 7 to the problem of simultaneously minimizing the maximum lateness and average completion time. Next we will show that any schedule that is an α-approximation in terms of the optimal average completion time is also an α-approximation for the average weighted lateness of a schedule. This allows us to prove existence theorems for the bicriterion scheduling problems $(C_{\max}, \sum w_j L_j)$ and $(L_{\max}, \sum w_j L_j)$.

Lemma 16 *Let $\sum_j w_j C_j^*$ and $\sum_j w_j L_j^*$ be the optimal total weighted completion time and lateness respectively of a set of jobs. If for some $\alpha \geq 1$*

$$\sum_j w_j C_j^S \leq \alpha \sum_j w_j C_j^*$$

for schedule S, then

$$\sum_j w_j L_j^S \leq \alpha \sum_j w_j L_j^*.$$

Proof: We know by the definition of lateness

$$\sum_j w_j L_j = \sum_j (C_j + q_j) = \sum_j w_j C_j + \sum_j w_j q_j.$$

Suppose we have some schedule S which is an α-approximation for the optimal average completion time. Then we can say that

$$\begin{aligned} \sum_j w_j L_j^S &= \sum_j w_j C_j^S + \sum_j w_j q_j \\ &\leq \alpha \sum_j w_j C_j^* + \sum_j w_j q_j \end{aligned}$$

and since $\alpha \geq 1$

$$\begin{aligned}\sum_j w_j L_j^S &\leq \alpha \sum_j w_j C_j^* + \alpha \sum_j w_j q_j \\ &\leq \alpha \sum_j w_j L_j^*.\end{aligned}$$

□

With Lemma 16, Theorem 15, and Theorem 8 we arrive at the following two theorems.

Theorem 17 *For any $\rho \in [0,1]$, for any scheduling problem, there exists a $(1+\rho, \frac{e^\rho}{(e^\rho-1)})$-schedule for $(C_{\max}, \sum w_j L_j)$.*

Proof: Immediate from Theorem 8 and Lemma 16. □

Theorem 18 *For any $\rho \in [0,1]$, for any scheduling problem, there exists a $(1+\rho, \frac{e^\rho}{(e^\rho-1)})$-schedule for $(L_{\max}, \sum w_j L_j)$.*

Proof: Immediate from Theorem 15 and Lemma 16. □

It is important to point out that Lemma 16 also tells us that minimizing the average weighted lateness of a schedule in the delivery time model is equivalent to minimizing the average weighted completion time. This will be useful in Section 4.1.2 when we look at applying the results of Torng and Uthaisombut [26] to models involving deadlines.

4.1.2 Lower Bounds and Algorithms

As a result of the general existence theorems from the previous section, it is natural to ask if we can find matching lower bounds. The question of matching lower bounds is easily answered. In the case where all delivery times are 0 or some small ϵ, the lateness of any job becomes equivalent to the job's completion time. Therefore the lower bound of Torng and Uthaisombut [26] for $1|r_j|(C_{\max}, \sum w_j C_j)$ also provides a lower bound for $1|r_j|(L_{\max}, \sum w_j C_j)$, $1|r_j|(C_{\max}, \sum w_j L_j)$, and $1|r_j|(L_{\max}, \sum w_j L_j)$.

Along with asking about lower bounds, we can also look for algorithms that achieve, or come close to achieving, our existence results. Torng and Uthaisombut[26] give a deterministic algorithm, $\mathcal{BEST} - \beta$ that produces a $(1+\beta, \frac{e^\beta}{e^\beta-1})$-approximation for $1|r_j|(C_{\max}, \sum C_j)$. In section 3.3.2 we examined the algorithm $\mathcal{BEST} - \beta$. Now we will see how it can be used as a simultaneous approximation algorithm for these problems now involving deadlines. By Lemma 16 we get the following corollary to their result.

Corollary 19 *For $0 < \beta \leq 1$, $\mathcal{BEST} - \beta$ is a deterministic $(1+\beta, \frac{e^\beta}{e^\beta-1})$-approximation algorithms for $1|r_j|(C_{\max}, \sum L_j)$ where L_j is the delivery time formulation of the maximum lateness problem.*

Proof: This is a direct result of the work by Torng and Uthaisombut[26] and Lemma 16. □

By the following lemma their algorithm also produces a schedule that is a $(2+\beta, \frac{e^\beta}{e^\beta-1})$-approximation for $1|r_j|(L_{\max}, \sum C_j)$.

Lemma 20 *Any $(1+\beta)$-schedule for $1|r_j|C_{\max}$ is also a $(2+\beta)$-approximation for $1|r_j|L_{\max}$.*

Proof: Recall that

$$\begin{aligned} L^{\text{opt}}_{\max} &\geq C^{\text{opt}}_{\max} \\ L^{\text{opt}}_{\max} &\geq \max_j q_j. \end{aligned}$$

We use these to upper bound the maximum lateness $L^S_{\max}$ of a schedule with $C^S_{\max} \leq (1+\beta)C^{\text{opt}}_{\max}$ by observing that

$$\begin{aligned} L^S_{\max} &= \max_j (C_j + q_j) \\ &\leq \max_j (C^S_{\max} + q_j) \\ &\leq (1+\beta)C^{\text{opt}}_{\max} + \max_j (q_j) \\ &\leq (1+\beta)L^{\text{opt}}_{\max} + L^{\text{opt}}_{\max} \\ &\leq (2+\beta)L^{\text{opt}}_{\max}. \end{aligned}$$

□

Corollary 21 *For $0 < \beta \leq 1$, $\mathcal{BEST} - \beta$ is a deterministic $(2+\beta, \frac{e^\beta}{e^\beta - 1})$-approximation algorithm for $1|r_j|(L_{\max}, \sum C_j)$.*

Proof: This is a direct result of the work by Torng and Uthaisombut[26] and Lemma 20. □

Finally, by Lemmas 16 and 20 we also know that $\mathcal{BEST} - \beta$ is a $(2+\beta, \frac{e^\beta}{e^\beta - 1})$-approximation for $1|r_j|(L_{\max}, \sum L_j)$.

Corollary 22 *For $0 < \beta \leq 1$, $\mathcal{BEST} - \beta$ is a deterministic $(2+\beta, \frac{e^\beta}{e^\beta - 1})$-approximation algorithm for $1|r_j|(L_{\max}, \sum L_j)$.*

Proof: This is a direct result of the work by Torng and Uthaisombut[26] and Lemmas 16 and 20. □

Corollaries 21 and 22 show that while $\mathcal{BEST} - \beta$ is an approximation algorithm for $1|r_j|(L_{\max}, \sum C_j)$ and $1|r_j|(L_{\max}, \sum L_j)$ it does not meet the lower bounds given in the beginning of this section. It is still open whether another deterministic algorithm could do better for $1|r_j|(L_{\max}, \sum C_j)$ and $1|r_j|(L_{\max}, \sum L_j)$.

4.2 Flow Time

The flow time of a job, given by $F_j = C_j - r_j$, can be thought of as the amount of time it stays in the system before being completed. Two optimality criteria associated with flow time are the average weighted flow time and the maximum flow time of any job.

The average (or total) weighted flow time is $\sum w_j F_j$. The average flow time is often considered both a very accurate measure of the responsiveness of a system and also one of the most difficult optimality criteria to work with. In our existence proofs, we will ignore the problem of actually finding an optimal average weighted flow time schedule for a set of jobs. Instead, we will assume that we have an optimal average weighted flow time schedule. The flow time of job j in this

optimal schedule will be F_j^*. We will denote the value of the optimal schedule as $\sum w_j F_j^*$. The completion time of j in the schedule achieving the optimal value will be written C_j^F. The second criteria that we will consider will be the optimal maximum flow time of a schedule. We will let $F_{\max}^S = \max_j(F_j^S) = \max_j(C_j^S - r_j)$ be the maximum flow time of schedule S and $F_{\max}^{\text{opt}}$ be the optimal maximum flow time of a set of jobs.

Given the difficulty of approximating the average weighted flow time[16], we will present only existence results in this section. First we will examine the relationship between the makespan and average weighted flow time of a schedule. We will prove the existence of schedules which are simultaneously good approximations for both criteria. Next we will consider the maximum flow time and average weighted completion time of a schedule. Again we will be able to prove existence results for this set of criteria. We will use these results and the ideas from section 4.1 to prove existence results for two bicriterion scheduling problems involving flow time and lateness. Finally, we will show that instances exist for which no (α, β)-schedule exists, with α and β both constant, for the maximum flow time and average weighted flow time.

Makespan and Average Weighted Flow Time We will consider the problem of simultaneously approximating the makespan and average weighted flow time of a set of jobs. We will use a similar construction to the one presented when we proved our bicriterion existence result for the problem of minimizing the makespan and average completion time. In other words, if S^F is the optimal average flow time schedule and S^M is the optimal makespan schedule of length M, we will create the schedule $S^\lambda = \textbf{Combine}(S^M, S^F, \lambda M)$. Immediately we can use Lemma 1 to say that S^λ will have length at most $(1+\lambda)M$. If we then restrict λ to be less than ρ then we know that the makespan of the schedule with the best breakpoint will still have a makespan of at most $(1+\rho)$ times the optimal schedule length.

Given this construction we now need to determine an upper bound on the average flow time of our new schedule.

Lemma 23 *Given two schedules S^F, the optimal average flow time schedule, and some other schedule S_1 of length $K = C_{\max}^1$, for any $\rho \in [1,2]$ there exists a schedule that is an $(\frac{e^\rho}{e^\rho - 1})$-approximation for the optimal average flow time and of length at most $(2+\rho)K$.*

Proof: One thing to notice about flow time is that the only general lower bound we have for the flow time of job j is

$$F_j \geq p_j.$$

If a job has zero processing time, its flow time in the optimal average weighted flow time schedule could also be zero. As with lateness, this makes approximating the average flow time of a schedule difficult. We will work around this difficulty as follows. Let $r_{\max} = \max_j r_j$. Therefore we get the following lower bound on the flow time of job j:

$$\begin{aligned} F_j &= C_j^F - r_j \\ F_j &\geq C_j^F - r_{\max}. \end{aligned}$$

We know that all jobs must be released before the end of schedule S_1 and therefore $r_{\max} \leq K$. We can use this to say

$$F_j \geq C_j^F - K. \tag{11}$$

Notice that this lower bound will be negative, and therefore somewhat useless, for jobs that complete before time K in the optimal average weighted flow time schedule. In the case when all jobs complete before K, then we know that the length of the optimal average flow time is at most K. However, if some jobs complete after K than this provides a lower bound for their flow time in the optimal average flow time schedule.

We will create the schedule $S^\lambda = \textbf{Combine}(S_1, S^F, \lambda K)$. In order to take advantage of lower bound 11, we now consider truncation points between K and $(1+\rho)K$ for $\rho \in [0,1]$. This corresponds to choosing λ from the range $[1,(1+\rho)]$. By Lemma 1 we know that a particular choice of λ will result in a schedule S^λ of length at most $(2+\lambda)K$. Furthermore by Lemma 1, for any choice of λ in the range $[1,1+\rho]$ we know that the length of the resulting schedule will be at most $(2+\rho)K$. Therefore, once we have ρ we need to choose λ to minimize the average flow time of our new schedule.

We start by considering the average flow time of our new schedule with a particular λ. A job j with $C_j^F < \lambda K$ will have its completion time and therefore flow time unchanged. For a job with $C_j^F \geq \lambda K$ we let $t = C_j^F$. Since the length of the new schedule will be at most $(1+\lambda)K$ we know that the new flow time of job j in our schedule created by using λ as our break point can be characterized as

$$\begin{aligned} F_j^\lambda &= C_j^\lambda - r_j \\ &= (C_j^\lambda - C_j^F) + C_j^F - r_j \\ &= (C_j^\lambda - C_j^F) + F_j^* \\ &\leq (1+\lambda)K - t + F_j^* \\ &= \left(\frac{(1+\lambda)K - t}{F_j^*} + 1\right) F_j^*. \end{aligned}$$

Then by applying 11 we get

$$\frac{(1+\lambda)K - t}{F_j^*} \leq \frac{(1+\lambda)K - t}{t - K}.$$

Therefore the flow time of job j in the new schedule is at most

$$\begin{aligned} F_j^\lambda &\leq \left(\frac{(1+\lambda)K - t}{t-K} + 1\right) F_j^* \\ &= \frac{(1+\lambda)K - t + t - K}{t-K} F_j^* \\ &= \frac{\lambda K}{t-K} F_j^*. \end{aligned}$$

This upper bound on F_j^λ looks very similar to the upper bound we arrived at for the completion time of each job in our new schedule when we were considering simultaneously minimizing the makespan and average completion time of a schedule. Just as in that analysis, the next step is to move to a continuous analysis. In the average completion time case, we were able to exactly model the average weighted completion time of a schedule by taking the sum over all time t of the amount of weight completing at t multiplied by t. This is not true when we are dealing with flow times,

since jobs with the same flow time could complete at different times. However, if we are concerned only with upper bounding the increase in flow time of our new schedule, we notice that our lower bound

$$F_j^* \quad \geq \quad t - K$$

is the same for all jobs j with $C_j^F = t$.

Therefore for a particular time y, we know that

$$\sum_{j|(C_j^F=y)} w_j F_j* \geq \sum_{j|(C_j^F=y)} w_j(y-K).$$

We will now express this lower bound as a continuous function $g(y) = \sum_j w_j(C_j^F - K)\delta(C_j^F - y)$. Recall that $\delta(\cdot)$, Dirac's delta function, is 0 for all $x \neq 0$. This means that $g(y)$ will be an impulse of "area" equal to $\sum_{j|C_j^F=y} w_j(C_j^F - K)$ centered at y. Given this representation of the weight completing at a certain time, we can now integrate over all time to arrive at a lower bound for our schedule.

Lemma 24 *The worst case schedule $g(y)$ will have*

$$\int_0^K g(y)dy = 0.$$

Proof: By contradiction. Assume $\int_0^K g(y)dy = c$ with $c > 0$. By our above argument, all jobs that complete before time λK will be run according to the optimal average flow time schedule. Therefore any such jobs will maintain their optimal flow time value in our new schedule S^λ. Since $\lambda \geq 1$, we know that for all choices of λ all jobs that complete before K will maintain their optimal flow time in S^λ. Therefore we can create a worse schedule by moving all jobs that complete before K to some time after K. □

We can assume wlog that the weights have been normalized so that

$$\sum_j w_j(C_j^F - K) = 1.$$

By Lemma 24 with our normalized weights we also get

$$\sum_{j|C_j^F \geq K} w_j(C_j^F - K) = 1.$$

Then for the worst case schedule we have that $\int_0^\infty g(y)\,dy = 1$ and $g(y) \geq 0$, and therefore $g(y)$ is a pdf. At this point we can write an upper bound on the average weighted flow time of S^λ as

$$\begin{aligned} & \int_0^{\lambda K} g(y)\,dy + \int_{\lambda K}^\infty \frac{\lambda K}{y-K} g(y)\,dy \\ = & \int_0^\infty g(y)\,dy + \int_{\lambda K}^\infty \frac{(1+\lambda)K - y}{y-K} g(y)\,dy \\ = & 1 + \int_{\lambda K}^\infty \frac{(1+\lambda)K - y}{y-K} g(y)\,dy. \end{aligned}$$

Notice that we only need to consider the second half of the above expression. For a particular $g(y)$ we can find the best λ in the range $[1, 1+\rho]$ to minimize the above integral. To arrive at an upper bound on the average flow time of a schedule with makespan of at most $(2+\rho)K$, we find and upper bound on the pdf that maximizes this calculation. This corresponds to solving the problem

$$\max_{g} \min_{1 \leq \lambda \leq \rho} \int_{\lambda K}^{\infty} \frac{(1+\lambda)K - y}{y - K} g(y)\, dy, \tag{12}$$

where g is a probability distribution over $[0, \infty)$. As in our proof of Lemma 6, this can be shown to be equivalent to the expression

$$\max_{f} \min_{1 \leq \lambda \leq \rho} \int_{\lambda}^{\infty} \frac{1 + \lambda - x}{x - 1} f(x)\, dx, \tag{13}$$

where now f ranges over all distributions.

Now we let $\alpha = \lambda - 1$ be chosen in the range $[0, 1]$. Finally, to shift our summation, we also need to allow $n = x - 1$. The result of this set of transformations is

$$\max_{f} \min_{0 \leq \alpha \leq \rho} \int_{\alpha}^{\infty} \frac{1 + \alpha - n}{n} f(n)\, dn \tag{14}$$

which is exactly expression 1 which we solved in Lemma 6. □

We can gain some intuition as to why we were able to reduce this problem to the one we solved in the average completion time case by looking at our construction again. Suppose we consider starting our schedule at time $-K$. We must adjust all r_j to be $r_j^{adj} = r_j - K$ and all C_j^F, the completion time of j in the optimal average weighted flow time schedule, to be $C_j^{adj} = C_j^F - K$. Since $K \geq \max r_j$, all jobs will now be released before time 0. By Lemma 24 we know that in the worst case pdf, all jobs will have positive completion times in our shifted problem. Similarly, because $g(t)$ was only a lower bound on the flow time of each job, it was equivalent to the case were all jobs were released at time K which is now time 0. In other words, previously our lower bound on the flow time of jobs completing after time K was $F_j^* \geq C_j^F - K$. With our shifted problem, our lower bound now becomes just $F_j^* \geq C_j^{adj}$. In the case when the lower bound actually holds with equality we know that all jobs would now have to be released at time 0 for our shifted problem. In other words, we allow the optimal average weighted flow time schedule to run until a time K when we know all jobs have been released. Then we treat the remaining portion of the optimal average weighted flow time schedule as an optimal average weighted completion time schedule. Furthermore, we argue that our worst case optimal average flow time schedule will have no job complete before K.

If we let S_1 from Lemma 23 be the optimal makespan schedule S^M, then we arrive at the following theorem.

Theorem 25 *For any $\rho \in [0, 1]$, for any scheduling problem, there exists a $(2+\rho, e^\rho/(e^\rho - 1))$ approximation for makespan and average weighted flow time.*

Proof: This follows from Lemma 23. □

While Theorem 25 proves that schedules exist which are simultaneously constant factor approximations for the makespan and average weighted flow time of a set of jobs, it is not clear that it provides the best constants possible. Since this theorem proves a weaker bound than that proven

in Theorem 8 for the problem of simultaneously minimizing the makespan and average weighted completion time it seems probable that a stronger statement can be made. At this time we do not have a lower bound for this problem. We can say that a lower bound matching this existence result will not be found by looking at the one machine case with release dates. In that case we know that after all jobs have been released there will no longer be any unforced idle time in the optimal average flow time schedule and therefore the length of this schedule can be at most $2C^{\text{opt}}_{\max}$.

Maximum Flow Time and Average Weighted Completion Time Now we will look at the problem of simultaneously minimizing the maximum flow time of a schedule and the average weighted completion time. We will use techniques similar to those presented in the previous sections to prove the following theorem.

Theorem 26 *For any $\rho \in [0,1]$, for any scheduling problem, there exists a $(1+\rho, 2+\frac{1}{\rho})$-schedule for the maximum flow time and average weighted completion time.*

Proof: We will assume that we have an optimal maximum flow time schedule S^F and an optimal average weighted completion time schedule S^*. We let $F^{\text{opt}}_{\max}$ be the optimal maximum flow time and $\sum w_j C^*_j$ be the optimal average completion time. We will consider the schedule $S^\lambda = \textbf{Combine}(S^F, S^*, \lambda F^{\text{opt}}_{\max})$.

First we say that the maximum flow time of schedule S^λ will be at most $(1+\lambda)F^{\text{opt}}_{\max}$. We arrive at this by the following argument. Any job that is run according to the optimal average completion time schedule must complete before time $\lambda F^{\text{opt}}_{\max}$ and therefore must have a flow time of at most $\lambda F^{\text{opt}}_{\max}$. All other jobs are run according to their order in the optimal maximum flow time schedule. When this schedule was started at time 0, this ordering guaranteed a maximum flow time of $F^{\text{opt}}_{\max}$. Since any job j that is run in this portion of S^λ is delayed by at most $\lambda F^{\text{opt}}_{\max}$ with respect to its start time in S^F, its flow time can increase by at most $\lambda F^{\text{opt}}_{\max}$. Therefore, the flow time of job j will be at most $(1+\lambda)F^{\text{opt}}_{\max}$. Figure 5 shows this construction and its effect on F_j.

Now we need to analyze the average completion time of S^λ. To simplify notation we will let $F = F^{\text{opt}}_{\max}$. We know that all jobs that complete before time λF in S^* will have their completion times unchanged in S^λ. Therefore we need only consider the jobs with $C^*_j \geq \lambda F$. By the above argument, we know that the flow time of each job in S^λ is at most $(1+\lambda)F$. This leads to the following upper bound on C^λ_j.

$$\begin{aligned} C^\lambda_j &\leq r_j + (1+\lambda)F \\ &\leq C^*_j + (1+\lambda)F \\ &\leq C^*_j + \frac{(1+\lambda)}{\lambda} C^*_j \\ &= \frac{(1+2\lambda)}{\lambda} C^*_j \\ &= (2+\frac{1}{\lambda}) C^*_j. \end{aligned}$$

Since the completion time of each job is at most $(2+\frac{1}{\lambda})$ times its completion time in the optimal average weighted completion time schedule, then we know that

$$\sum_j w_j C^\lambda_j \leq (2+\frac{1}{\lambda}) \sum_j w_j C^*_j.$$

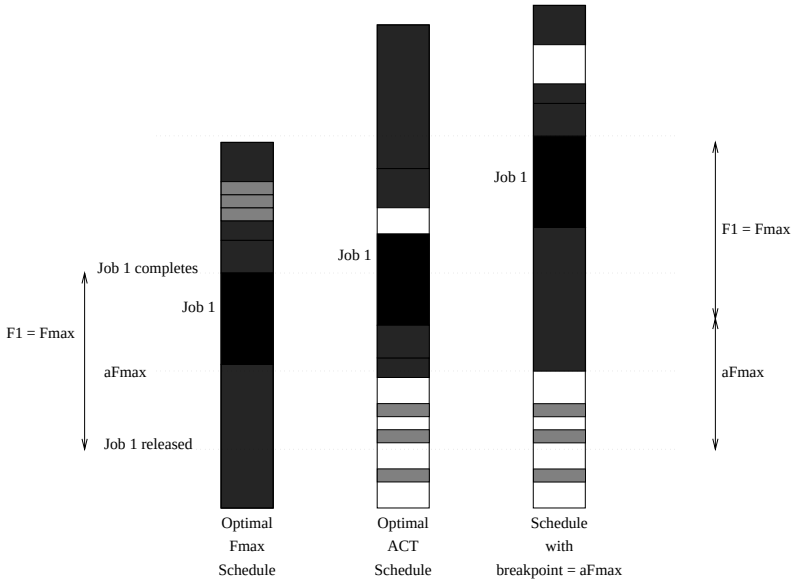

Figure 5: In this example Job 1 is the job with the maximum flow time in the optimal $F_{\max}$ schedule. The dark jobs are the jobs which complete after time $\alpha F_{\max}$ in the optimal average completion time schedule. The grey jobs are the jobs that complete before time $\alpha F_{\max}$ in the optimal average completion time schedule. When Job 1 is run in the schedule created by truncating the optimal ACT schedule at time $aF_{\max}$, it has a flow time of at most $aF_{\max} + F_{\max} = (1+a)F_{\max}$.

□

We can use Theorem 26 and set $\rho = 1$ to arrive at the next corollary.

Corollary 27 *For any scheduling problem, there exists a* $(2,3)$*-schedule for maximum flow time and average weighted completion time.*

While these results do confirm the existence of schedules that are simultaneous constant approximations for the maximum flow time and average weighted completion time of a set of jobs, it also seems possible that a stronger upper bound may exist. Finding a better upper bound for this problem or finding a lower bound is currently an open problem.

Flow Time and Lateness The existence results for $(C_{\max}, \sum w_j F_j)$ and $(F_{\max}, \sum w_j C_j)$ can easily be extended to show the following two corollaries.

Corollary 28 *For any* $\rho \in [0,1]$*, for any scheduling problem, there exists a* $(1+\rho, 2+\frac{1}{\rho})$*-schedule for the maximum flow time and average weighted lateness.*

Proof: This is a direct result of Theorem 26 and Lemma 16. □

Corollary 29 *For any* $\rho \in [0,1]$*, for any scheduling problem, there exists a* $(2+\rho, \frac{e^\rho}{(e^\rho - 1)})$*-schedule for* $(L_{\max}, \sum w_j F_j)$*.*

Proof: Suppose we are given two schedules S^L, the optimal maximum lateness schedule of length $C^L_{\max}$, and S^F, the optimal average flow time schedule. Then we can create a schedule $S^\lambda = \textbf{Combine}(S^L, S^L, \lambda C^L_{\max})$. Lemma 14 tells us that maximum lateness of S^λ will be at most $(1+\lambda)$ times the optimal maximum lateness. If we let ρ be in the range $[1,2]$ and choose the best λ such that $1 \leq \lambda \leq \rho$, then we know that the maximum lateness of our new schedule will be at most $(1+\rho)$ times the optimal maximum lateness. Lemma 23 tells us that choosing the best breakpoint from the range $[1,\rho]$ will result in a schedule with average weighted flow time at most $\frac{e^\rho}{(e^\rho-1)}$ times the optimal average weighted flow time. □

Maximum Flow Time and Average Flow Time Now that we have presented four existence theorems for bicriterion scheduling problems involving an objective based on flow time, we will look at $(F_{\max}, \sum w_j F_j)$. While it seems that we should be able to use some of the ideas from the previous sections to prove the existence of schedules that are simultaneously good approximations for the maximum flow time and average weighted flow time of a set of jobs, unfortunately this is not the case. The following counter-example shows that instances exist for which no schedule will be a constant factor approximation for both criteria simultaneously.

Theorem 30 *If $F^{\text{opt}}_{\max}$ is the optimal maximum flow time and $\sum F^*_j$ is the optimal total flow time of a set of N jobs, then instances exist for which there is no (α,β)-schedule with $1 \leq \alpha < \frac{\sqrt{N}}{4}$ and $1 \leq \beta < \frac{\sqrt{N}}{4}$ for $(F_{\max}, \sum F_j)$.*

Proof: Consider the following example on one machine with release dates. Let j_0 be released at $t=0$ with processing time $p_0 = \sqrt{N}$. Let jobs $j_1, j_2, \ldots, j_n$ be jobs of length 1. Let $r_i = i$ for all $j_1, j_2, \ldots, j_n$.

To prove our theorem we will start by considering the optimal maximum flow time schedule and optimal average flow time schedule. We will show that both optimal schedules are $\frac{\sqrt{N}}{3}$-approximations for the other criteria. Then we will show that no other schedule is simultaneously within a factor of $\frac{\sqrt{N}}{4}$ of optimal for both criteria.

Since j_0 has the earliest release date and the longest processing time, the schedule S^F that achieves $F^{\text{opt}}_{\max}$ runs j_0 at time $t=0$ when it is released. All N small jobs are delayed by $\sqrt{N}$ time units. $F_{\max}$ of this schedule is $F^{\text{opt}}_{\max} = \sqrt{N}$. The total flow time of this schedule, which we will denote $\sum F^{\max}_j$ is

$$\sum F^{\max}_j = \sum_{j=0}^{N} \sqrt{N} = N^{\frac{3}{2}} + \sqrt{N}.$$

On the other hand, the optimal average flow time schedule will run all small jobs $j_1, j_2, \ldots, j_n$ as they are released and run j_0 starting at time N. Since all jobs of size 1 complete 1 time unit after they are started, $F_i = 1$, $\forall i > 0$. The larger job will now have to wait until all the small jobs complete and therefore will have $F_0 = N + \sqrt{N}$. The total flow time of this schedule will be $\sum F^*_j = 2N + \sqrt{N}$. The maximum flow time of this schedule will be denoted $F^*_{\max} = N + \sqrt{N}$.

For the optimal average flow time schedule we have the following relationship for its maximum flow time compared to the optimal maximum flow time.

$$\frac{F^*_{\max}}{F^{\text{opt}}_{\max}} = \frac{N+\sqrt{N}}{\sqrt{N}}$$

$$\begin{aligned} &= \sqrt{N}+1 \\ &\geq \frac{\sqrt{N}}{3} \end{aligned}$$

Now if we consider the total flow time of the optimal maximum flow time schedule we arrive at the following lower bound.

$$\begin{aligned} \frac{\sum F_j^{\max}}{\sum F_j^*} &= \frac{N^{\frac{3}{2}}+\sqrt{N}}{2N+\sqrt{N}} \\ &= \frac{N+1}{2\sqrt{N}+1} \\ &\geq \frac{N}{3\sqrt{N}} \\ &= \frac{\sqrt{N}}{3} \end{aligned}$$

Having shown that both optimal schedules are at least a $\frac{\sqrt{N}}{3}$-approximation for the other criteria, we now need to consider all schedules that are not optimal for either criteria. First consider that any schedule in which j_0 starts before $t = N/4$ will have at least $3N/4$ small jobs with $F_j = \sqrt{N}$ and therefore a total flow time of

$$\begin{aligned} \sum_j F_j &\geq \frac{3N}{4}\sqrt{N} \\ &= \frac{3N^{\frac{3}{2}}}{4}. \end{aligned}$$

Which means

$$\begin{aligned} \frac{\sum_j F_j}{\sum_j F_j^*} &\geq \frac{\frac{3N^{\frac{3}{2}}}{4}}{2N+\sqrt{N}} \\ &= \frac{\frac{3N}{4}}{2(\sqrt{N})+1} \\ &\geq \frac{\frac{3N}{4}}{3\sqrt{N}} \\ &= \frac{\sqrt{N}}{4}. \end{aligned}$$

However, if j_0 starts after $t = N/4$ in S then $F_{\max}^S \geq N/4$. Which gives us

$$\begin{aligned} \frac{F_{\max}^S}{F_{\max}^{\text{opt}}} &\geq \frac{\frac{N}{4}}{\sqrt{N}} \\ &= \frac{\sqrt{N}}{4}. \end{aligned}$$

This means that the average flow time of a schedule starting j_0 before $\frac{N}{4}$ is greater than $\frac{\sqrt{N}}{4}$ times the optimal average flow time and the maximum flow time of any schedule starting j_0 after time $\frac{N}{4}$

is $\frac{\sqrt{N}}{4}$ times the optimal maximum flow time. Therefore no (α, β)-schedule exists with $1 \leq \alpha < \frac{\sqrt{N}}{4}$ and $1 \leq \beta < \frac{\sqrt{N}}{4}$. □

4.3 The Number of On-Time Jobs

Finally, we consider one last objective function. For this objective we again associate with each job j a deadline d_j. Instead of examining the lateness of jobs we will look at only whether or not a job is late. We do this by defining U_j to be 0 if job j meets its deadline, i.e. $C_j \leq d_j$, and 1 otherwise. To maximize the number of jobs finishing on time in schedule S we maximize $\sum_j (1 - U_j^S)$.

Makespan and the Number of On-Time Jobs

Theorem 31 *For* any *scheduling problem there exists schedules that are simultaneously* 2*-approximations for* $\sum_j (1 - U_j)$ *and* $C_{\max}$.

Proof: Assume that S^M and S^U are optimal schedules for $C_{\max}$ and $\sum_j (1 - U_j)$ respectively. Let C_j^* be the completion time of job j in the schedule with the optimal number of jobs completing on time. Then U_j^* is 0 if $C_j^* \leq d_j$ and 1 otherwise. Finally, to simplify notation, let $M = C_{\max}^{\text{opt}}$.

If $\sum_j (1 - U_j^*) = 0$ then we know that in the optimal schedule no job meets its deadline and therefore all schedules achieve the optimal value. As a result we know that the optimal makespan schedule is also optimal for $\sum_j (1 - U_j)$. If $\sum (1 - U_j^*) \geq 1$, then let $K = \{j | U_j^* = 0\}$ be the set of jobs that complete by their deadlines in schedule S^U.

If in schedule S^U more than half the jobs in K have met their deadline by M, then we create schedule $S' = \textbf{Combine}(S^M, S^U, M)$. S' will be of length at most $2M$ and at least half the jobs in K will meet their deadlines. Since the total number of optimal jobs meeting their deadlines is K this schedule is a $(2, 2)$-approximation for $C_{\max}$ and $\sum_j (1 - U_j)$. If, however, less than half of the jobs in K complete by M, then we know that at least half of the jobs that meet their deadlines in the optimal schedule do so after time M. Therefore, at least half of the jobs in K have $d_j \geq M$. In this case just run the optimal makespan schedule. This schedule will be at most a 2-approximation for $\sum_j (1 - U_j)$ because all jobs will complete before time M and therefore at least half of the jobs in K will still meet their deadlines in the optimal makespan schedule. □

We will now show that there exists at least one instance for which we cannot simultaneously do better than a 2-approximation for $C_{\max}$ and $\sum (1 - U_j)$.

Theorem 32 *There exist two-job instances for which no schedule is simultaneously within* α *of optimal for* $\alpha < 2$ *for* $\sum_j (1 - U_j)$ *and* $C_{\max}$.

Proof: Consider the following example on one machine with release dates. Given two jobs, j_1 and j_2 with $p_1 = M - 1$, $p_2 = 1$, $r_1 = 0$, $r_2 = M - 2$, $d_1 = 3M$ and $d_2 = M - 1$. The optimal makespan schedule runs j_1 followed by j_2 and has length M and $\sum_j (1 - U_j) = (1 - U_1) + (1 - U_2) = (1 - 0) + (1 - 1) = 1$. The optimal $\sum_j (1 - U_j)$ schedule runs j_2 followed by j_1. In this schedule both jobs meet their deadline and so $\sum_j (1 - U_j) = 2$ and the makespan is $M + M - 1 = 2(M) - 1$. For all M the optimal makespan schedule is a 2-approximation for $\sum_j (1 - U_j)$. As M gets large, the length of the optimal $\sum_j (1 - U_j)$ schedule approaches $2M$. Our theorem follows from the fact that these are the only two schedules. □

Theorem 33 *There exist job instances of size* $2n$ *for which no schedule on* n *identical parallel machines is simultaneously within* α *of optimal for* $\alpha < 2$ *for* $\sum_j (1 - U_j)$ *and* $C_{\max}$.

Proof: We simply extend the 1 machine example to n machines. We do this by having n large jobs of size $M-1$ released at time 0. Each of these jobs has a deadline $3M$. We also have n jobs of size 1 released at time $M-2$. Each of these small jobs has a deadline of $M-1$. The optimal $\sum_j(1-U_j)$ schedule will have all $2n$ jobs meet their deadline by delaying all large jobs until all the small jobs complete. The optimal makespan schedule will run all large jobs immediately and delay all small jobs 1 time unit. In the optimal makespan schedule all small jobs will miss their deadline and therefore only half of the optimal number of jobs will meet their deadline. The makespan of any schedule that delays even 1 large job to allow a small job to complete on time will be $2M-1$. Therefore any schedule that delays even 1 large job will be a 2-approximation for the optimal makespan.

Suppose that we consider a schedule which uses K machines to run large jobs when they are released and $q = n - K$ machines to run small jobs when they are released. Notice that any such schedule will still have a makespan of $2(M-1)$ because at least q large jobs must wait until time $M-1$ to begin processing. As stated earlier, this means that any such schedule is already a 2-approximation for the makespan. Notice also that only q small jobs will meet their deadlines because once we run q small jobs on the machines reserved to run small jobs, we are already at time $M-1$ and therefore all remaining small jobs will miss their deadlines. Therefore if we consider all schedules with a makespan of size twice optimal we find that the one with the maximum number of jobs completing on time is the optimal $\sum_j(1-U_j)$. The only schedule with a makespan of smaller than twice optimal only has half of the optimal number of jobs complete on time. Therefore no schedule exists which is an (α,β)-approximation for $(C_{\max}, \sum(1-U_j))$ with $\alpha < 2$ and $\beta < 2$. □

Maximum Lateness and the Number of On-Time Jobs We can extend our result for $(C_{\max}, \sum(1-U_j))$ to $(L_{\max}, \sum(1-U_j))$.

Corollary 34 *For* any *scheduling problem there exists schedules that are simultaneously* 2*-approximations for* $\sum_j(1-U_j)$ *and* $L_{\max}$.

Proof: Again let K be the set of jobs that complete on time in the optimal $\sum(1-U_j)$ schedule. If half of the jobs in K complete before time $C^L_{\max}$, in the optimal $\sum(1-U_j)$ schedule, then we run the optimal $\sum(1-U_j)$ schedule until time $C^L_{\max}$ and finish the remaining jobs according to the optimal maximum lateness schedule. We find that the maximum lateness of all jobs in our new schedule S' is

$$\begin{aligned} L'_j &= C'_j + q_j \\ &\leq C^L_{\max} + C^L_j + q_j \\ &\leq 2L^{\text{opt}}_{\max}. \end{aligned}$$

Since at least half of the jobs complete on time before we begin running jobs according to S^L we know that S' completes at least half of the optimal number of on-time jobs before their deadlines. If fewer than half of the jobs in K complete before $C^L_{\max}$ then running schedule S^L in which all jobs complete before $C^L_{\max}$ will have at least half of the jobs in K complete on-time. □

Maximum Flow Time and the Number of On-Time Jobs In this section we will present our second negative result.

Theorem 35 *If $F_{\max}^{\text{opt}}$ is the optimal maximum flow time and $\sum(1-U_j^*)$ is the optimal number of on-time jobs of a set of N jobs, then instances exist for which there is no (α,β)-schedule with $1 \leq \alpha < N^{\frac{1}{4}}$ and $1 \leq \beta < N^{\frac{1}{4}}$ for $(F_{\max}, \sum(1-U_j))$.*

Proof: Consider the following example on one machine with release dates. Suppose that every $\sqrt{N}$ time units from time 0 until time $N-\sqrt{N}$, a job of size $\sqrt{N}$ is released. Let $p=\sqrt{N}$ and label these jobs $j_0, j_1, \ldots, j_p$ where $r_i = i\sqrt{N}$. Let $d_i = N^N$ for all $j_0, j_1, \ldots, j_p$. The other $N-\sqrt{N}$ jobs are released as follows. Let $\epsilon << \sqrt{N}$. Every $\sqrt{N}$ time units from time $\sqrt{N}-\epsilon$ to time $N-\sqrt{N}-\epsilon$, $\sqrt{N}$ jobs of size $\frac{\epsilon}{\sqrt{N}}$ are released. We can organize these jobs into groups $g_1, g_2, \ldots g_{p-1}$ each with $\sqrt{N}$ jobs where all jobs in group g_i are released at time $r_i = i\sqrt{N}-\epsilon$ and have a deadline of $i\sqrt{N}$.

First we will show that the optimal $F_{\max}$ schedule has a maximum flow time of at most $\sqrt{N}(1+\epsilon)$. Consider schedule, S^F, which has no idle time and runs all jobs in the order that they arrive. All groups g_i of small jobs start at time $i\sqrt{N}+(i-1)\epsilon$ and finish almost immediately and therefore have a flow time of ϵ. The maximum flow time of this schedule is achieved by the last job of size $\sqrt{N}$. Since S^F runs all the small jobs between the large jobs, the i^{th} large job is delayed an additional $i\epsilon$ time units. Therefore the last job is delayed $\sqrt{N}\epsilon$ time units and has a maximum flow time of $\sqrt{N}(1+\epsilon)$. Since the small jobs don't start until after their deadline, none of them complete on-time. All the jobs of size $\sqrt{N}$ do complete on-time and therefore $\sum(1-U_j^F) = \sqrt{N}$. Since we know at least one schedule exists with a maximum flow time of $\sqrt{N}(1+\epsilon)$, we know that the optimal schedule will have a maximum flow time of no more than $\sqrt{N}(1+\epsilon)$.

The optimal $\sum(1-U_j)$ schedule, S^U, runs all small jobs when they arrive and runs all large jobs after all small jobs have been released. Since all jobs complete on time $\sum(1-U_j^U) = N$. Since the first job of size $\sqrt{N}$ must wait until time $N-\sqrt{N}$ to start, the maximum flow time of this schedule is at least N. Both optimal schedules are clearly $\sqrt{N}$-approximations for the other criteria.

To show that all other schedules are at least a $N^{\frac{1}{4}}$-approximation for one of the criteria we will show that all schedules that delay at least $N^{\frac{1}{4}}$ large jobs are at least a $N^{\frac{1}{4}}$-approximation for the maximum flow time and any schedule that delays fewer than $N^{\frac{1}{4}}$ large jobs is at least a $N^{\frac{1}{4}}$-approximation for the number of on-time jobs.

Consider any schedule S^T which allows fewer than $N^{\frac{1}{4}}$ groups g_i to complete on-time. In this case we know that fewer than $N^{\frac{1}{4}}-1$ groups of small jobs complete on-time and all the large jobs complete on-time. Since there are $\sqrt{N}$ jobs in each group of small jobs and $\sqrt{N}$ large jobs, we know that $\sum(1-U_j^T) \leq N^{\frac{1}{4}}\sqrt{N}$. Since the optimal number of on-time jobs is N this schedule is a $\frac{1}{N^{\frac{1}{4}}}$-approximation for the optimal number of on-time jobs.

Now we will show that any schedule, S^D that completes more than $N^{\frac{3}{4}}$ small jobs on time must have a maximum flow time of at least $N^{\frac{3}{4}}$ and therefore will be a $N^{\frac{1}{4}}$-approximation for the optimal $F_{\max}$. Notice that for every $\sqrt{N}$ small jobs that complete on-time at least 1 job of size $\sqrt{N}$ must be delayed $\sqrt{N}$ time. Consider the first time t a job of size $\sqrt{N}$ is delayed $\sqrt{N}$ time. When the next job of size $\sqrt{N}$ is released S^D will still not have processed the job that was delayed. Therefore S^D will delay at least one of these two jobs another $\sqrt{N}$ time units. In this way there will always be at least 1 job of size $\sqrt{N}$ ready to be processed when a new job of size $\sqrt{N}$ is released. Since we must delay $N^{\frac{1}{4}}$ jobs of size $\sqrt{N}$ in order to complete $N^{\frac{3}{4}}$ small jobs on time, we know that S^D will delay at least one large job $N^{\frac{3}{4}}$ time units. Therefore any such schedule is a $N^{\frac{1}{4}}$-approximation for the maximum flow time.

□

5 Specific environments

Previously we looked at bicriterion scheduling in a very general setting. In the following section we will look at one specific machine environment and improve upon our general results.

5.1 $P|r_j|(C_{\max}, \sum F_j)$

We will now consider the problem of minimizing the makespan and average flow time of a set of jobs J on M parallel machines. We let r_j be the release time of job j. We denote this problem $P|r_j|(C_{\max}, \sum F_j)$.

Theorem 36 *Any optimal schedule for $P|r_j|\sum F_j$ will be at most a 3-approximation for $P|r_j|C_{\max}$.*

Proof: We know from Graham [7] that for identical parallel machines without release dates, any list-scheduling algorithm produces a schedule with makespan at most twice optimal. Scheduling jobs according to Shortest Processing Time ($\mathcal{SPT}$) creates an optimal average completion time schedule if there are no release dates and $\mathcal{SPT}$ is a list scheduling algorithm[5]. We now show that after time $\max_j(r_j)$ the problem $P|r_j|\sum F_j$ is equivalent to $P|r_j|\sum C_j$ for the remaining unscheduled jobs.

Let $r_{\max} = \max_j(r_j)$. Any job scheduled after time $r_{\max}$ will have the following flow time

$$F_j = C_j - r_j = C_j - r_{\max} + r_{\max} - r_j.$$

Let s_j be the time that job j starts processing and define $K = \{j | s_j \geq r_{\max}\}$ to be the set of jobs scheduled to start processing after the last release date. Therefore the total flow time for this set of jobs will be

$$\begin{aligned}
\sum_{j\in K} F_j &= \sum_{j\in K}(C_j - r_j) \\
&= \sum_{j\in K}(C_j - r_{\max} + r_{\max} - r_j) \\
&= \sum_{j\in K}(C_j - r_{\max}) + \sum_{j\in K}(r_{\max} - r_j).
\end{aligned}$$

Notice that after time $r_{\max}$, all jobs j in K add a cost of $(r_{\max} - r_j)$ to the total flow time regardless of the order they are scheduled. Therefore, for the remaining jobs, the problem of minimizing the average flow time after time $r_{\max}$ is equivalent to the following statement:

$$\min \sum_{j\in K}(C_j - r_{\max}).$$

Since $r_{\max}$ is a constant for all jobs, this is equivalent to minimizing the average completion time relative to some fixed point in time. As a result, minimizing the total flow time after $t = r_{\max}$ for these jobs corresponds to minimizing $\sum_{j\in K}(C_j)$. Therefore, since $\mathcal{SPT}$ is an exact list-scheduling algorithm for $P||\sum C_j$ we know that the length of this portion of the schedule must be at most $2C^{\text{opt}}_{\max}$. Using the fact that all jobs must be released before the end of the optimal makespan schedule, we know that the length of the optimal average flow time schedule must be at most $C^{\text{opt}}_{\max} + 2C^{\text{opt}}_{\max} = 3C^{\text{opt}}_{\max}$.

□

6 Conclusion and Open Problems

This thesis has presented existence results for a variety of bicriterion scheduling problems. The following table gives a summary of those results.

		β			
	(α, β)	ACT $\sum w_j C_j$	AFT $\sum w_j F_j$	AL $\sum w_j L_j$	$\sum(1 - U_j)$
α	Makespan ($C_{\max}$)	$(1+\rho, \frac{e^\rho}{e^\rho - 1})$	$(2+\rho, \frac{e^\rho}{e^\rho - 1})$	$(1+\rho, \frac{e^\rho}{e^\rho - 1})$	$(2, 2)$
	Maximum Flow Time($F_{\max}$)	$(1+\rho, 2+\frac{1}{\rho})$	NO	$(1+\rho, 2+\frac{1}{\rho})$	NO
	Maximum Lateness ($L_{\max}$)	$(1+\rho, \frac{e^\rho}{e^\rho - 1})$	$(2+\rho, \frac{e^\rho}{e^\rho - 1})$	$(1+\rho, \frac{e^\rho}{e^\rho - 1})$	$(2, 2)$

In some cases we have also looked at lower bounds for these bicriterion scheduling problems. Those results are summarized below. We also include some results that follow directly from the lower bounds presented.

		β			
	(α, β)	ACT $\sum w_j C_j$	AFT $\sum w_j F_j$	AL $\sum w_j L_j$	$\sum(1 - U_j)$
α	Makespan ($C_{\max}$)	Torng & Uthaisombut[26] $(1+\beta, \frac{e^\beta}{e^\beta - 1})$	$(1+\beta, \frac{e^\beta}{e^\beta - 1})$	$(1+\beta, \frac{e^\beta}{e^\beta - 1})$	$(2, 2)$
	Max Flow Time ($F_{\max}$)	?	$(\frac{\sqrt{N}}{4}, \frac{\sqrt{N}}{4})$	?	?
	Max Lateness ($L_{\max}$)	$(1+\beta, \frac{e^\beta}{e^\beta - 1})$	$(1+\beta, \frac{e^\beta}{e^\beta - 1})$	$(1+\beta, \frac{e^\beta}{e^\beta - 1})$	$(2, 2)$

As this table shows, while some of the general existence results have been matched by lower bounds, for many of them either no lower bound is known or there remains a gap between the known lower bound and the existence result.

For most of the problems presented, no algorithm has been shown to produce good bicriterion schedules. Finding better existence results, tighter lower bounds or good bicriterion approximation algorithms for many of these problems are all still open questions.

References

[1] J. A. Aslam, A. Rasala, C. Stein, and N. Young. Improved bicriteria existence theorems for scheduling. In *Proceedings of the 10th ACM-SIAM Symposium on Discrete Algorithms*, 1999.

To appear.

[2] K. R. Baker. *Introduction to Sequencing and Scheduling.* Wiley, 1974.

[3] S. Chakrabarti, C. A. Phillips, A. S. Schulz, D. B. Shmoys, C. Stein, and J. Wein. Improved scheduling algorithms for minsum criteria. In F. Meyer auf der Heide and B. Monien, editors, *Automata, Languages and Programming*, number 1099 in Lecture Notes in Computer Science. Springer, Berlin, 1996. Proceedings of the 23rd International Colloquium (ICALP'96).

[4] C. Chekuri, R. Motwani, B. Natarajan, and C. Stein. Approximation techniques for average completion time scheduling. In *Proceedings of the 8th ACM-SIAM Symposium on Discrete Algorithms*, pages 609–618, January 1997.

[5] R.W. Conway, W.L. Maxwell, and L.W. Miller. *Theory of Scheduling.* Addison-Wesley, 1967.

[6] M. R. Garey, R.E. Tarjan, and G. T. Wilfong. One-processor scheduling with symmetric earliness and tardiness penalties. *Mathematics of Operations Research*, 13:330–348, 1988.

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

[8] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5:287–326, 1979.

[9] L. A. Hall, A. S. Schulz, D. B. Shmoys, and J. Wein. Scheduling to minimize average completion time: Off-line and on-line approximation algorithms. *Mathematics of Operations Research*, 22:513–544, August 1997.

[10] J. A. Hoogeveen. Minimizing maximum promptness and maximum lateness on a single machine. *Mathematics of Operations Research*, 21:100–114, 1996.

[11] J. A. Hoogeveen. Single-machine scheduling to minimize a function of two or three maximum cost criteria. *Journal of Algorithms*, 21(2):415–433, 1996.

[12] J. A. Hoogeveen and S.L. van de Velde. Minimizing total completion time and maximum cost simulatneously is solvable in polynomial time. *Operations Research Letters*, 17:205–208, 1995.

[13] C.A.J. Hurkens and M.J. Coster. On the makespan of a schedule minimizing total completion time for unrelated parallel machines. Unpublished manuscript, 1996.

[14] David Karger, Cliff Stein, and Joel Wein. *CRC Handbook on Algorithms*, chapter Scheduling Algorithms. CRC Press, 1998.

[15] T. Kawaguchi and S. Kyan. Worst case bound of an LRF schedule for the mean weighted flow-time problem. *SIAM Journal on Computing*, 15:1119–1129, 1986.

[16] H. Kellerer, T. Tautenhahn, and G. J. Woeginger. Approximability and nonapproximability results for minimizing total flow time on a single machine. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pages 418–426, May 1996.

[17] E.L. Lawler, J.K. Lenstra, A.H.G. Rinooy Kan, and D.B. Shmoys. Sequencing and scheduling: Algorithms and complexity. In S.C. Graves, A.H.G. Rinnooy Kan, and P.H. Zipkin, editors, *Handbooks in Operations Research and Management Science, Vol 4., Logistics of Production and Inventory*, pages 445–522. North-Holland, 1993.

[18] S.T. McCormick and M.L. Pinedo. Scheduling n independent jobs on m uniform machines with both flow time and makespan objectives: A parametric approach. *ORSA Journal of Computing*, 7:63–77, 1992.

[19] R.T. Nelson, R.K. Sarin, and R.L. Daniels. Scheduling with multiple performance measures: the one-machine case. *Management Science*, 32:464–479, 1986.

[20] S.S. Panwalkar, R.A. Dudek, and M.L. Smith. Sequencing research and the industrial scheduling problem. In S.E. Elmaghraby, editor, *Symoposium on the Theory of Scheduling and its Applications*. Springer, New York, 1973.

[21] C. Phillips, C. Stein, and J. Wein. Minimizing average completion time in the presence of release dates. *Mathematical Programming*, 82:199–223, 1998.

[22] M. Pinedo. *Scheduling: Theory, Algorithms and Systems*. Prentice Hall, 1995.

[23] D. B. Shmoys and E. Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical Programming A*, 62:461–474, 1993.

[24] W.E. Smith. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3:59–66, 1956.

[25] C. Stein and J. Wein. On the existence of schedules that are near-optimal for both makespan and total weighted completion time. *Operations Research Letters*, 21, 1997.

[26] E. Torng and P. Uthaisombut. Lower bounds for srpt-subsequence algorithms for nonpreemptive scheduling. In *Proceedings of the 10th ACM-SIAM Symposium on Discrete Algorithms*, pages 973–974, 1999.

[27] L.N. Van Wassenhove and F. Gelders. Solving a bicriterion scheduling problem. *European Journal of Operations Research*, 4:42–48, 1980.