DIMACS Technical Report 2001-49
December 2001

Approximation algorithms for Offline scheduling with set-ups

by

Srikrishnan Divakaran [1]
Department of Computer Science
Hofstra University
West Hempstead, NY
cscsyd@hofstra.edu

Michel Saks [2]
Department of Mathematics
Rutgers University
New Brunswick, NJ
saks@math.rutgers.edu

[1] Supported in Part by NSF grant CCR-9700239 and DIMACS.
[2] Supported in Part by NSF grant CCR-9700239.

ABSTRACT

In this paper, we present new approximation results for the offline problem of scheduling n independent jobs with sequence independent set-ups and common release times on a single machine system for the following optimality criteria :

- *total weighted completion time*
- *maximum lateness*

For the total weighted completion time criterion, we present a 2-approximation algorithm, the first known polynomial time constant approximation algorithm for this problem. For this criterion, when the number of job types is arbitrary, the computational complexity is open and prior to our results there were no known polynomial time constant approximation algorithms even when all jobs have unit weight and all job types have unit set-up times.

For the maximum lateness criterion, we present an algorithm that obtains an optimal solution in a relaxed framework where our algorithm is given strictly more resources (i.e. a machine with higher processing speed) than the optimal offline algorithm to which it is compared. For this criterion, when the number of job types is arbitrary, this problem is known to be NP-hard even when there are two jobs in each type or when all job types have unit set-up times, there are at most three jobs in each type and there are at most three distinct due dates. Also, for this criterion, when the number of job types and the due dates are arbitrary, there are no constant approximation algorithms unless $P = NP$.

1 Introduction

The *problem of scheduling with set-ups* [1,2,3,4,6,8,9,10,14,17,22, 23,26,27,28,33,34] occurs naturally in systems that involve processing jobs of different types on common facilities where a set-up is incurred when there is a switch from processing a job of one type to a job of another type. The set-up time in general depends on both the type of the most recently serviced job as well as the type of the currently serviced job. This general case is referred to as *sequence dependent set-ups.* The special case where the set-up time depends only on the type of the currently serviced job is referred to as *sequence independent set-ups.*

Some specific examples of *scheduling with set-ups* were described in Monma and Potts [23], Baker [2,3] and Scutten et al [27,28]. The first example involves the production of different colors of paint on the same machine. A set-up time for cleaning the machine is incurred whenever the machine switches from manufacturing a paint of one color to another. The amount of time required to clean the machine depends both on the color being removed and the color for which the machine is being prepared. In this model, a set-up time is required at the start of production and on each occasion when the machine switches from manufacturing one color paint to another. In this scenario, a batch is a maximal set of jobs that are scheduled contiguously on a machine and share a set-up. Large batches have the advantage of high machine utilization because the number of set-ups is small. On the other hand, processing in large batches may cause some jobs to incur a large delay. So, in this manufacturing system the system designer needs to trade-off between the time spent on set-ups and the maximum delay of a job. Note that if all processing costs are zero this problem reduces to an instance of the well known travelling salesperson problem. A second example is the problem of scheduling tasks in computer systems. In this problem, a collection of tasks is to be processed on a computer where each task has a requirement for a particular compiler to be resident in the computer's memory. If the appropriate compiler is resident, then the task may start processing immediately; otherwise a setup time is incurred to bring the relevant compiler into memory. In this example, the setup time depends only on the time to load the compiler for the current job, and does not depend on the previous job. A third example is a manufacturing system where the labor force is a limiting resource and workers must be switched from machine to machine, incurring a set-up time, in order to complete all tasks.

In this paper, we focus on single machine scheduling problems with sequence independent set-ups and common release times. For these problems, the *total weighted completion time* and *maximum lateness* are two natural optimality criteria for evaluating solutions. For the *total weighted completion time* criterion, each job is associated with a penalty or non-negative weight for waiting, and the objective is to minimize the total penalty for waiting. For the maximum lateness criterion, each job has a due date and the objective is to minimize the maximum amount of time by which the completion time of any job deviates from its due date.

Related Work : For scheduling problems with sequence independent set-ups and common release times, associated with the total weighted completion time and maximum lateness criteria, Monma and Potts[23] provide a dynamic programming algorithm that finds an optimal solution in $O(n^F)$ time, where n is the number of jobs and F is the number of job types. This algorithm is polynomial if F is fixed, but exponential otherwise. For the total weighted completion time criterion, when the number of job types is arbitrary, the complexity of scheduling with set-ups is open. For this criterion, even when all jobs have unit weight and all job types have unit set-up times, there are no known polynomial time algorithms. For the maximum lateness criterion, when the number of job types is arbitrary, Bruno and Downey[6] proved the problem of scheduling with sequence independent set-ups to be NP-hard even when there are two jobs in each type or when all job types have unit set-up times, there are at most three jobs in each type and there are at most three distinct due dates. Also, for this criterion, when the number of job types and the due dates are arbitrary, there are no constant approximation algorithms unless $P = NP$ (see Hall[16]). However, when the the due dates are non-positive, Zdrzalka[33,34] provided constant approximation algorithms.

Our Results : In this paper, we give some new approximation results on the problem of single machine scheduling with sequence independent set-ups and common release times for the total weighted completion time and the maximum lateness criteria. For the total weighted completion time, we present the first known constant approximation algorithm. For the maximum lateness criterion, we analyze the problem of scheduling with set-ups in a relaxed framework [19,24] where the performance of our algorithm is compared to the performance of an optimal offline algorithm with strictly less resources (lower processing speed). In this framework, we present a constant approximation algorithm.

2 Problem Description

Model Description : We consider a system consisting of a single machine M that is capable of sequentially serving one job at a time. Each job belongs to one of F pre-determined job types $1, 2, ..., F$ and is available for servicing at time 0. The machine services a job in two stages : the set-up stage and the processing stage. In the set-up stage, the machine checks whether the most recently serviced job is of the same type as the job that is currently serviced. If so, the machine proceeds immediately to the processing stage. Otherwise, the machine is configured (or set-up) to process jobs of that type. The set-up for a job is needed only if at the start of servicing, the most recently serviced job is not of the same type as the current job. In the processing stage, the machine processes the job without interruption. The *set-up time* for a given job depends only on the type of the currently serviced job and is referred to as *sequence independent set-ups*.

The incoming job sequence is typically denoted by $J = (J_1, J_2, ..., J_N)$. Each job $J_i \in J$ is of type $\tau_i \in [1..F]$, requires a processing time of p_i. Each job J_i is available for processing at time 0 and prior to processing J_i the machine requires a minimum setup time of 0 if the most recently processed job is of type τ_i and s_{τ_i} otherwise. In addition, each job may be associated with a non-negative weight w_i and a due date d_i.

Preliminaries : A *schedule* σ for a job sequence J specifies for each job $J_i \in J$, the time A_i^σ at which the set-up of J_i starts, the time B_i^σ at which processing of J_i starts and the time C_i^σ at which J_i completes. A schedule σ for job sequence J is said to be *feasible* if

- for any pair of jobs $J_i, J_j \in J$, the time intervals $[A_i^\sigma, C_i^\sigma]$ and $[A_j^\sigma, C_j^\sigma]$ are disjoint (i.e. the machine M can serve only one job at a time).
- for $i \in [1..n]$, $B_i^\sigma - A_i^\sigma$ is greater than or equal to the set-up time for J_i. Note that if the type of the most recently serviced job is of the same type as this job, then its set-up time is considered to be zero.
- for $i \in [1..n]$, $C_i^\sigma - B_i^\sigma \geq p_i$ (i.e. the time interval between the completion and the start of processing of a job is greater than or equal to its processing time).

We denote the set of feasible schedules for J by $\Phi(J)$. The following are some of the objective functions (or optimality criteria) that we use in evaluating a feasible schedule in $\Phi(J)$

- Maximum Lateness : $L_{max}^\sigma = \max_{i\in[1..n]} L_i^\sigma = \max_{i\in[1..n]} C_i^\sigma - d_i$
- Total Weighted Completion Time : $\sum WC^\sigma = \sum_{i\in[1..n]} w_i C_i^\sigma$.

Each of these objective functions are functions of set of job completion times, so their general form is $Z = f(C_1, C_2, ..., C_n)$. Furthermore, these objective functions are *r egular*, that is Z can increase only if the completion time of at least on e of the jobs increase.

For a given objective function f, a schedule $\sigma \in \Phi(J)$ is said to be optimal if $f(C_1^\sigma, ..., C_n^\sigma) \leq f(C_1^\phi, ..., C_n^\phi)$ for any other schedule $\phi \in \Phi(J)$. In the future, for ease of notation, we will often denote $f(C_1^\sigma, ..., C_n^\sigma)$ by f^σ.

A scheduling problem $\mathcal{P}$ consists of a set $\mathcal{J}$ of possible input sequences and a regular objective function f. Associated with every input sequence J is a set of feasible schedules $\Phi(J)$, and associated with each feasible schedule $\sigma \in \Phi(J)$ is a real number f^σ representing the value of objective function for the schedule σ.

Given any input sequence J, a scheduling algorithm G for a scheduling problem $\mathcal{P}$ assigns to each job sequence J a schedule $G(J) \in \Phi(J)$. The value of the objective function associated with the schedule $G(J)$ is denoted by $f^{G(J)}$. An *optimal algorithm* OPT is such that for all sequences J, $f^{OPT(J)} = \min_{\sigma\in\Phi(J)} f^\sigma$. It is easy to show that for a regular objective function every sequence has an optimal schedule.

A scheduling algorithm G is a *c-approximation algorithm* [5] for a minimization problem $\mathcal{P}$ if there is a constant $\alpha \geq 0$ such that for all input sequences J, $f^{G(J)} - cf^{OPT(J)} \leq \alpha$.

More precisely, such an algorithm is called a *c-approximation algorithm*. We reserve the term *strict c-approximation algorithm* to mean $\alpha = 0$. There are certain scheduling problems for which there exist no constant approximation algorithms unless $P = NP$. For these problems, we use a relaxed framework (please see Kalyanasundaram and Pruhs[19]) where an algorithm is analyzed by comparing its performance to an optimal algorithm with strictly less resources (i.e. memory or processing speed). In this relaxed framework, for a minimization problem $\mathcal{P}$, a scheduling algorithm G is a *s-speed c-approximation algorithm*, if for all input sequences J, using a machine that is at most s times faster than the machine used by an optimal offline algorithm , G finds a schedule $G(J)$ such that $f^{G(J)} \leq cf^{OPT(J)}$.

Problem Definition : We now define the problems for which we present some new approximation results.

- **Total Weighted Completion Time** $(1|s_f, w_i|\sum WC)$ **:** Given a single machine system with sequence independent set-ups and a job sequence $J = (J_1 J_2, ..., J_n)$, where each job J_i belongs to one of F job types $\{1, ..., F\}$, is available for processing at time 0, requires a processing time p_i and has a non-negative weight w_i, determine a schedule $\sigma \in \Phi(J)$ that minimizes the total weighted completion time of jobs in J.

- **Maximum Lateness** $(1|s_f, d_i|\mathbf{L}_{max})$: Given a single machine system with sequence independent set-ups and a job sequence $J = (J_1 J_2, ..., J_n)$, where each job J_i belongs to one of F job types $\{1, ..., F\}$, is available for processing at time 0, requires a processing time p_i and has a due date d_i, determine a schedule $\sigma \in \Phi(J)$ that minimizes the maximum lateness of any job in J.

3 Paper Outline

The technical results in this paper are organized as follows : In Section 4, we present the general structure of solutions for problems in scheduling with sequence independent set-ups when all jobs have a common release time. In Section 5, we make use of the results presented in Section 4 to design a 2-approximation algorithm for the *Total Weighted Completion Time* Problem $(1|s_f, w_i|\sum WC)$. In Section 6, we make use of the results presented in Section 4 to design a 2-speed-1 approximation algorithm for the *Maximum Lateness* problem $(1|s_f, d_i|L_{max})$.

4 Structure of Problems in Scheduling with Set-ups with Common Release Times

For problems in scheduling with sequence independent set-ups, we first briefly discuss the dynamic programming algorithm due to Monma and Potts[23]. Then, we present our framework for designing polynomial time constant approximation algorithms. First, we introduce certain definitions.

4.1 Preliminaries

In the general model described in Section 2, jobs in a job sequence are indexed in non-decreasing value of release times. In the case when all jobs have a common release time, we are free to choose any convenient ordering. In this situation of common release times, our choice of ordering will depend on the optimality criterion.

Previously, in Section 2 we describe the input job sequence as a single sequence of jobs. In the present context, it will be convenient to divide the job sequence into a collection of job sequences, one for each job type. Now we will introduce some definitions that will help in presenting our general framework.

Definitions **4.1:** Let J be the given job sequence. For $f \in [1..F]$, let $J^{<f>} = (J_1^{<f>}, J_2^{<f>}, ..., J_{n_f}^{<f>})$ denote the sequence of jobs of type f in J, $J_i^{<f>}$ the ith job in $J^{<f>}$, $p_i^{<f>}$ the processing time of $J_i^{<f>}$, $r_i^{<f>}$ the release time of $J_i^{<f>}$, $w_i^{<f>}$ the weight associated with $J_i^{<f>}$ and $d_i^{<f>}$ the due date of $J_i^{<f>}$.

Definitions **4.2:** A *job priority function* maps each job $J_i \in J$ to a real number $priority(J_i)$ referred to as the priority of J_i. For a given problem of scheduling with set-ups, a job priority function is said to be *good*, if for any job sequence J, there exists an optimal schedule in $\Phi(J)$, such that the jobs within each type are processed in non-decreasing value of job priority.

Definitions **4.3:** A *batch* is a sequence of jobs that are of the same type. For a job sequence J and a job priority function u, a *u-decomposition of* $J^{<f>}$ is a partitioning of $J^{<f>}$ into a set of batches $\{B_1^{<f>}, B_2^{<f>}, ..., B_{\eta_f}^{<f>}\}$ such that

- the jobs within each batch are ordered in non-decreasing value of job priority
- for $i \in [1..\eta_f - 1]$, the priority of each job in $B_i^{<f>}$ is less than or equal to the priority of every job in $B_{i+1}^{<f>}$.

Notice that the sequence of batches $(B_1^{<f>}, B_2^{<f>}, ..., B_{\eta_f}^{<f>})$ can be viewed as a natural ordering of jobs within each job type in an optimal schedule in $\Phi(J)$.

For a job sequence J and a job priority function u, a *u-decomposition* is obtained by taking the union of the batches in a u-decomposition of $J^{<f>}$, one for each job type $f \in [1..F]$.

Example 1 : Let $J = (J_1, J_2, J_3)$ be a sequence of jobs with J_1 and J_3 of type 1 and J_2 of type 2, the set-up times for all job types are 1, all jobs have unit weights, all jobs are released at time 0, J_1 has a processing time of 3, J_2 has a processing time of 2 and J_3 has a processing time of 1. Let u be the job priority function that assigns to each job $J_i \in J$ a real number priority $u(J_i) = p_i/w_i$. For the above job sequence J and job priority function u, $\{(J_3), (J_1)\}$ and $\{(J_3, J_1)\}$ are the only two distinct u-decompositions of $J^{<1>}$, $\{(J_2)\}$ is the only distinct u-decomposition of $J^{<2>}$, and $\{(J_3, J_1), (J_2)\}$, and $\{(J_3), (J_1), (J_2)\}$ are the only two distinct

u-decompositions of J. Notice that $\{(J_1, J_3)\}$ is not a valid u-decomposition of $J^{<1>}$ because the jobs in batch (J_1, J_3) are not ordered in non-decreasing value of job priority.
Notice that from our definition of a feasible schedule (in Section 2), we allow schedules where processing of each job is preceded by a set-up whose duration is greater than the minimum set-up time for that job.
We say a schedule is *canonical* if the set-up time for each job of type f is either 0 or s_f. Note that this definition of a canonical schedule allows for unnecessary set-ups. We define a schedule to be *wasteless* if it is canonical and the set-ups are only performed when necessary. For a given job priority function u, we say a schedule σ is *u-canonical*, if σ is canonical and in σ jobs within each type are processed in non-decreasing value of job priority.
Example 2 : For the sequence in example 1, let $\sigma \in \Phi(J)$ be a schedule where the set-up and processing of J_3 is done during the time intervals $[0, 1]$ and $[1, 2]$ respectively, the set-up and processing of J_1 is done during the time intervals $[2, 2]$ and $[2, 5]$ respectively, and the set-up and processing of J_2 is done during the time intervals $[5, 6]$ and $[6, 8]$ respectively. Let $\phi \in \Phi(J)$ be a schedule where the set-up and processing of J_3 is done during the time intervals $[0, 1]$ and $[1, 2]$ respectively, the set-up and processing of J_1 is done during the time intervals $[2, 3]$ and $[3, 6]$ respectively, and the set-up and processing of J_2 is done during the time intervals $[6, 7]$ and $[7, 9]$ respectively. Let $\rho \in \Phi(J)$ be a schedule where the set-up and processing of J_1 is done during the time intervals $[0, 1]$ and $[1, 4]$ respectively, the set-up and processing of J_3 is done during the time intervals $[4, 4]$ and $[4, 5]$ respectively, and the set-up and processing of J_2 is done during the time intervals $[5, 6]$ and $[6, 8]$ respectively. In this example, σ and ρ are examples of canonical schedules that are wasteless and ϕ is an example of a canonical schedule that is not wasteless. Also, σ and ϕ, are schedules that are u-canonical, whereas ρ is not u-canonical because in ρ jobs of type 1 are not processed in non-decreasing value of job priority.
It is clear that given any canonical schedule σ that is not wasteless, we can easily convert it into a wasteless schedule that processes jobs in the same order and has an objective function value that is less than or equal to that of σ.
Given any u-canonical schedule σ, we can observe that the non-zero set-ups in σ divide the set of jobs into a sequence $(B_1^\sigma, B_2^\sigma, ..., B_\eta^\sigma)$ of batches. This sequence of batches uniquely defines a u-canonical schedule. Furthermore, if we restrict this sequence of batches to jobs of type f, for $f \in [1..F]$, then we get a u-decomposition of $J^{<f>}$. Therefore, a u-canonical schedule can be considered an interleaving of u-decompositions of $J^{<f>}$, for $f \in [1..F]$ (i.e. interleaving of the batches in a u-decomposition of J). Now, we will introduce some definitions that will help in analyzing different u-canonical schedules that have the same u-decomposition of J. Let Ψ be some arbitrary u-decomposition of J. Let $\rho(\Psi)$ denote the set of all u-canonical schedules that correspond to different interleavings of the batches in Ψ.

Definitions **4.4:** A *batch priority* function is a function which assigns a non-negative real number to a sequence of jobs of the same type (batch). For a given problem of scheduling with set-ups and a good job priority function u, a batch priority function is said to be *good*, if for any u-decomposition Ψ of J, there exists a schedule that is optimal among the schedules in $\rho(\Psi)$ and the batches in that schedule are arranged in non-decreasing value of batch priority.

4.2 Pseudo-polynomial Time Algorithms

For problems in scheduling with set-ups for which there are good job priority functions, Monma and Potts[23] provide a dynamic programming algorithm that finds an optimal solution in $O(n^F)$ time, where n is the number of jobs and F is the number of job types. This algorithm is polynomial if F is fixed but exponential otherwise. In this paper, due to space restrictions, we do not present the dynamic programming algorithm of Monma and Potts. For details on this result, we refer the reader to the paper of Monma and Potts[23] and Potts and Kovalyov[26].

4.3 General Schema for Approximation Algorithms

For problems in scheduling with set-ups for which we have good job priority and good batch priority functions, the problem of constructing a good schedule is greatly simplified. Let $\mathcal{P}$ be such a problem and of scheduling with set-ups, let u be a good job priority function and v be a good batch priority functions that can be evaluated in time that is polynomial in the size of input. Then, an algorithm for obtaining a solution for $\mathcal{P}$ can be described as follows :

Algorithm Schema S
[Step 1] Order the jobs in J in non-decreasing value according to u;
[Step 2] Determine a u-decomposition of J;
[Step 3] Order the batches in the u-decomposition of J obtained in step 2 according to v.

Given u and v, implementing steps 1 and 3 is trivial. So, the problem of scheduling with set-ups reduces to performing step 2 (i.e. finding a u-decomposition of J).
We will now focus our attention on obtaining an u-decomposition of J that helps in obtaining good approximation algorithms for problems in scheduling with set-ups. In terms of a real number parameter α, we now describe a u-decomposition of J that helps in designing good approximation algorithms. First, we will introduce definitions that help in defining this u-decomposition of J.

Definitions **4.5:** For a given sequence J and a real number $\alpha > 0$, the α- *segmentation* of J is defined to be the partitioning of jobs in $J^{<f>}$ for each $f \in [1..F]$ into batches by scanning $J^{<f>}$ from left to right such that

- the sum of the processing times of all jobs except the first job in a batch does not exceed $\alpha * s_f$;

- each batch is maximal;

The batches in the α-*segmentation* of J are referred to as α-*segments* of J.

Notice that, if we fix the ordering of jobs in J using a good job priority function u, then an α-segmentation of J defines a unique u-decomposition of J. Also, it is easy to observe that an α- segmentation of J can be obtained by partitioning each sequence $J^{<f>}$, for $f \in [1..F]$ into segments by scanning it from left to right.

4.4 Algorithm $A(\alpha)$

For each real number $\alpha > 0$, we define algorithm $A(\alpha)$ to be the algorithm obtained from the general schema S above by choosing the u-decomposition in step 2 to be the α-segmentation of the job sequence J.

In Sections 5 and 6, for the total weighted completion time and the maximum lateness problem, we first fix a good job priority function and batch priority function and then use the above algorithm to obtain a schedule with objective function value within a constant times the objective function value of an optimal schedule.

5 An approximation algorithm for total weighted completion time problem

In this section, we focus on the total weighted completion time problem $(1|s_f, w_i|\sum WC)$. For this problem, we first define a good job priority and a good batch priority function. Then, we show that for $\alpha = 1$, the algorithm $A(\alpha)$ (defined in Section 4) is a 2-approximation algorithm for this problem. For general discussion of results on related problems, we refer the reader to the survey papers of Hall[16], Baker and Webster[2] and Potts and Kovalyov[26].

5.1 Good job priority and batch priority functions

For the total weighted completion time problem, we now define good job priority and batch priority functions that can be evaluated in time polynomial in the size of input.

Definitions **5.1:** Let u be a function that maps each job $J_i \in J$ to a real number priority $u(J_i) = p_i/w_i$.

Definitions **5.2:** Let v be a function that maps each batch B in any u-decomposition of J to a real number priority $v(B) = P(B)/W(B)$, where $P(B)$ is the sum of the set-up time and the processing time of the jobs in B, and $W(B)$ is the sum of the weights of the jobs in B.

Proposition 1 (Bruno and Sethi (see [3]): For the total weighted completion time problem, u is a good job priority function.
Due to space restrictions, we do not present the proof of this proposition.
Proposition 2 (Mason and Anderson[22]) : For the total weighted completion time problem,

v is a good batch priority function.
Due to space restrictions, we do not present the proof of this proposition.
Notice, that for any job sequence (batch sequence) of size n, the functions u and v can be evaluated in $O(n \log n)$ time by sorting the jobs (batches) in non-decreasing value of priority. Also, notice that if the ordering of jobs in J are fixed using u, then the dynamic programming algorithm of Monma and Potts[23] obtains an optimal solution in $O(n^F)$ time, where n is the number of jobs and F is the number of job types. This algorithm is polynomial time if F is fixed, but otherwise it is exponential.

5.2 Analysis of Algorithm $A(\alpha)$

We recall from the discussion in Section 4 that, for a problem of scheduling with set-ups, if we have a good job priority function and a good batch priority function, then this problem reduces to the problem of finding an optimal u-decomposition of J.
For the total weighted completion time problem, from Propositions 1 and 2, we know that u and v are good job priority and batch priority functions respectively. For the above choices of u and v, we now prove that for $\alpha = 1$, $A(\alpha)$ (as defined in Section 4) is a 2-approximation algorithm.

Theorem 1: For $\alpha = 1$, $A(\alpha)$ is a 2 approximation algorithm for the total weighted completion time problem.

Proof Let $\alpha = 1$. Let Ψ denote the u-decomposition of J obtained by $A(\alpha)$ and let $\rho(\Psi)$ denote the set of all u-canonical schedules that correspond to different interleavings of the batches in Ψ.
From the definition of a good batch priority function, it is clear that algorithm $A(\alpha)$ determines a u-canonical schedule that is optimal among the schedules in $\rho(\Psi)$. From the definition of a good priority function, and Proposition 1, we know that there exists a u-canonical schedule in $\Phi(J)$ that is optimal. So, without loss of generality, let σ^* be a u-canonical schedule that is optimal among all the schedules in $\Phi(J)$. Now, we show that $A(\alpha)$ is a 2-approximation algorithm by obtaining a schedule $\bar{\sigma^*} \in \rho(\Psi)$ from σ^* such that $\sum WC^{\bar{\sigma^*}} \leq 2 \sum WC^{\sigma^*}$.
The following lemma shows that given any u-canonical schedule σ, we can obtain a schedule $\bar{\sigma} \in \rho(\Psi)$ such that $\sum WC^{\bar{\sigma}} \leq 2 \sum WC^{\sigma}$.

Lemma 2: Given any u-canonical schedule $\sigma \in \Phi(J)$, we can obtain a schedule $\bar{\sigma} \in \rho(\Psi)$ such that $\sum WC^{\bar{\sigma}} \leq 2 \sum WC^{\sigma}$.

Now, assuming that Lemma 2 is true, the proof follows immediately. □
Now, we present the proof of Lemma 2.
Proof of lemma 2 : Let $\sigma \in \Phi(J)$ be any arbitrary u-canonical schedule. Now, we will use the following procedure *transform* to transform σ to a schedule $\bar{\sigma} \in \rho(\Psi)$ such that $\sum WC^{\bar{\sigma}} \leq 2 \sum WC^{\sigma}$. Now, we present the procedure *transform*.

procedure transform : For $f \in F$, starting with the first batch of type f, we scan the batches of type f in σ in stages. During a stage, if the current batch does not contain an α-segment of J, then we transform the current batch into a α-segment by repeatedly moving the next job of type f in σ to the end of the current batch until the current batch becomes an α-segment (say s) of J. Otherwise, if the current batch properly contains an α-segment (say s)of J , we transform the current batch into an α segment by introducing a set-up of type f immediately after the last job in segment s. Now, we proceed to the next stage by scanning from the first job of type f that appears after s.

From the construction of $\bar{\sigma}$, it is clear that each batch in $\bar{\sigma}$ is an α-segment of J. Therefore, $\bar{\sigma}$ is a schedule in $\rho(\Psi)$. Now, we will show that $\sum WC^{\bar{\sigma}} \leq 2\sum WC^{\sigma}$ by showing that $w_i C_i^{\bar{\sigma}} \leq 2w_i C_i^{\sigma}$ for an arbitrary job $J_i \in J$.
Let J_i be any arbitrary job in J. For σ, let $K \subseteq J$ be the set of jobs that are processed before J_i, G be the set of job types of jobs in K, $\bar{K} \subset J$ be the set of jobs that are processed after J_i and $\bar{G}$ be the set of job types of jobs in $\bar{K}$. For $f \in [1..F]$, let T_f^{σ} and $T_f^{\bar{\sigma}}$ denote the set of jobs of type f that were processed before the completion of J_i in σ and $\bar{\sigma}$ respectively, and $P(T_f^{\sigma})$ and $P(T_f^{\bar{\sigma}})$ denote the sum of processing times of jobs in T_f^{σ} and $T_f^{\bar{\sigma}}$ respectively. Now, we establish this theorem by obtaining a lower bound on the weighted completion time of J_i in σ and obtaining an upper bound on the completion time of J_i in $\bar{\sigma}$. For σ, the completion time of J_i is

$$C_i^{\sigma} \geq \sum_{f \in G} s_f + \sum_{f \in G} P(T_f^{\sigma}) \tag{1}$$

Now, for $\bar{\sigma}$, we will get an upper bound on the completion time of J_i. From the definition of $\bar{G}$ and the construction of $\bar{\sigma}$, we know that in $\bar{\sigma}$, no job of type $f \in \bar{G}$ is processed prior to J_i. So, we only need to consider jobs of type $f \in G$ in order to obtain an upper bound on the completion time of J_i in $\bar{\sigma}$. Notice that the completion time of J_i in $\bar{\sigma}$ may be greater than the completion time of J_i in σ due to :

- introduction of additional set-ups in $\bar{\sigma}$;
- processing some jobs in $\bar{K}$ prior to J_i in $\bar{\sigma}$;

For $\bar{\sigma}$, let k_f be the number of set-ups of type f prior to processing J_i. So, the completion time of J_i in $\bar{\sigma}$ is

$$C_i^{\bar{\sigma}} \leq \sum_{f \in G} k_f s_f + \sum_{f \in G} P(T_f^{\bar{\sigma}}) \tag{2}$$

Now, we make use of the following observations to relate Equations (1) and (2).
Observation 1 : For $f \in G$, $P(T_f^{\sigma}) \geq (k_f - 1)s_f$.
Proof Let $f \in G$ be some arbitrary job type. For $j \in [1..k_f]$, let L_j denote the set consisting of the jobs in the first $j-1$ batches of type f in $\bar{\sigma}$ and the first job in the jth batch of type f in $\bar{\sigma}$. From the definition of a u-canonical schedule and the construction of $\bar{\sigma}$, we can observe that in σ, all jobs in $L_{k_f} - J_i$ are processed before J_i. From the construction of $\bar{\sigma}$, we can

observe that for $j \in [1..k_f]$, the sum of processing times of jobs in L_j is at least $(j-1)s_f$. Therefore, $P(T_f^\sigma) \geq (k_f - 1)s_f$ $\square$

Observation 2 : For $f \in G$, $P(T_f^{\bar{\sigma}}) \leq P(T_f^\sigma) + s_f$

Proof From the construction of $\bar{\sigma}$, the total processing time of jobs of type f that were scheduled after J_i in σ and before J_i in $\bar{\sigma}$ is at most s_f. $\square$

From Equation (1) and Observation 1, we get

$$C_i^\sigma \geq \sum_{f \in G} (s_f + P(T_f^\sigma)) \geq \sum_{f \in G} k_f s_f \tag{3}$$

From Equation (2) and Observation 2, summing over $f \in G$, we get

$$C_i^{\bar{\sigma}} \leq \sum_{f \in G} (k_f s_f + P(T_f^{\bar{\sigma}})) \leq \sum_{f \in G} (k_f s_f + s_f + P(T_f^\sigma)) \tag{4}$$

Adding Equations (1) and (3) and comparing with Equation (4), we get $C_i^{\bar{\sigma}} \leq \sum_{f \in G}(k_f s_f + s_f + P(T_f^{\bar{\sigma}})) \leq 2C_i^\sigma$. Therefore, $w_i C_i^\sigma \leq 2 w_i C_i^{\bar{\sigma}}$. $\square$

6 An approximation algorithm for maximum lateness problem

In this section, we focus on the maximum lateness problem $(1|s_f, d_i|L_{max})$. For this problem, we first define a good job priority and a good batch priority function. Then, we show that for $\alpha = 1$, the algorithm $A(\alpha)$ (defined in Section 4) is a 2-speed-1- approximation algorithm for this problem. For general discussion of results on related problems, we refer the reader to the survey papers of Hall[16], Baker and Webster[2] and Potts and Kovalyov[26].

6.1 Good job priority and batch priority functions

In this section, we define good job priority and batch priority functions that can be evaluated in time polynomial in the size of input.

Definitions **6.1:** Let u be a function that maps each job $J_i \in J$ to a real number priority $u(J_i) = d_i$.

Definitions **6.2:** Let v be a function that maps each batch B in any u-decomposition of J to a real number priority $v(B) = d_B$, where d_B is the due date of a job in B with the earliest due date.

Proposition 1: [Monma and Potts[23]] For the maximum lateness problem, u is a good job priority function.

Due to space restrictions, we do not present the proof of this proposition.

Proposition 2: For the maximum lateness problem, v is a good batch priority function.

Due to space restrictions, we do not present the proof of this proposition.
Notice, that for any job sequence (batch sequence) of size n, the functions u and v can be evaluated in $O(n \log n)$ time by sorting the jobs(batches) in non-decreasing value of priority. Also, notice that if the ordering of jobs in J are fixed using u, then the dynamic programming algorithm of Monma and Potts[23] obtains an optimal solution in $O(n^F)$ time, where n is the number of jobs and F is the number of job types. This algorithm is polynomial time if F is fixed, but otherwise it is exponential. For arbitray number of job types, Bruno and Downey [6] established the NP-hardness of the maximum lateness problem even for very restricted special cases. In addition the results in the survey paper of Hall[16] show that when the due dates are arbitrary, unless $P = NP$ there exists no constant approximation algorithms even for very restricted special cases of the maximum lateness problem. So, we will focus our attention on obtaining a constant approximation algorithm in a relaxed framework where the approximation algorithm is given a faster processor than the optimal algorithm to which it is compared.

6.2 Analysis of Algorithm $A(\alpha)$

We recall from the discussion in chapter 4 that for a problem of scheduling with set-ups, if we have a good job priority function u and a good batch priority function v, then the problem of obtaining an optimal solution reduces to the problem of finding an optimal u-decomposition of J. For the maximum lateness problem, from Propositions 1 and 2, we know that u and v are good job priority and batch priority functions respectively. For the above choices of u and v, we now prove that for $\alpha = 1$, $A(\alpha)$ (as defined in Section 4) is a 2-speed-1-approximation algorithm for the maximum lateness problem.

Theorem 3: For $\alpha = 1$, $A(\alpha)$ is a 2-speed-1-approximation algorithm for the *maximum lateness* problem.

Proof Let $\alpha = 1$. Let J be any arbitrary job sequence. For the maximum lateness problem, let σ^* be the schedule determined by an optimal algorithm OPT and σ be the schedule determined by algorithm $A(\alpha)$ when it uses a machine with processing speed twice that of OPT. That is, for $A(\alpha)$, the set-up time for job type f, $f \in [1..F]$ is $s'_f = s_f/2$, and the time to process a job $J_i \in J$ is $p'_i = p_i/2$.

We will now show that A is a 2-speed-1-approximation algorithm by showing that for any arbitrary job $J_i \in J$, $L_i^\sigma \leq L_{max}^{\sigma^*}$. Let J_i be any arbitrary job in J. Let $K \subseteq J$ be the set of jobs in J with due date less than or equal to d_i and G be the set of job types of jobs in K. For the schedule σ, let k_f, for $f \in [1..F]$, denote the number of set-ups of type f that were performed before the completion of J_i, T_f^σ denote the set of jobs of type f in J that were processed before the completion of J_i. For the optimal schedule σ^*, let $P(T_f^\sigma)$ denote the sum of the processing time of jobs in T_f^σ. For the schedule σ, let $P'(T_f^\sigma)$ denote the sum of processing times of jobs in T_f^σ. We want to remind the reader that $A(\alpha)$ uses a machine that has twice the processing power of the machine used by OPT. So, $P(T_f^\sigma) = 2P'(T_f^\sigma)$.

Now, for σ, we will get an upper bound on the lateness of J_i. From the definition of $A(\alpha)$, we know that in σ, only jobs of type $f \in G$ are processed prior to J_i. This follows from the observation that there are no jobs of type for $f \notin G$ with due date less than or equal to d_i. So, we only need to consider jobs of type $f \in G$ in order to obtain an upper bound on the completion time of J_i in σ. For σ, the completion time of J_i is $C_i^\sigma \leq \sum_{f \in G}(k_f s'_f + P'(T_f^\sigma))$ Therefore, in σ, the lateness of J_i is

$$L_i^\sigma = \sum_{f \in G}(k_f s'_f + P'(T_f^\sigma)) - d_i \tag{5}$$

Now, we make use of the following lemma to get a lower bound on the maximum lateness of σ^* :

Lemma 4: Let $J_l \in K$ be the last job in K to be processed in σ^*. For σ^*, the sum of set-up and processing times of jobs of type f in T_f^σ that are processed before the completion of J_l is at least $2max\{P'(T_f^\sigma), k_f s'_f\}$

Assuming Lemma 4 is true, summing over $f \in [1..F]$, we get the sum of set-up and processing times of jobs that are processed before the completion of J_l is at least $\sum_{f \in G} 2 * max\{P'(T_f^\sigma), k_f s'_f\}$. So, in σ^*, the lateness of J_l is

$$\begin{aligned} L_l^{\sigma^*} \geq \sum_{f \in G} 2max\{P'(T_f^\sigma), k_f s'_f\} - d_l &\geq \sum_{f \in G} 2max\{P'(T_f^\sigma), k_f s'_f\} - d_i \\ &\geq \sum_{f \in G} 2 * (P'(T_f^\sigma) + k_f s'_f)/2 - d_i = L_i^\sigma \end{aligned} \tag{6}$$

From Equations (5) and (6), we get $L_{max}^{\sigma^*} \geq L_l^{\sigma^*} \geq L_i^\sigma$. □
We now need to prove Lemma 4.

6.3 Proof of Lemma 4

Let f be some arbitrary job type in G. For σ^*, let S denote the the sum of set-up time and processing time of jobs of type f that are processed before the completion of J_l. Let $T_f^\sigma = \cup_{j \in [1..k_f]} B_j^\sigma$, where for $j \in [1..k_f]$, B_j^σ denotes the jth batch of type f in σ. For the schedule σ, let L_j , for $j \in [1..k_f]$, denote the set consisting of jobs in the $j-1$ batches B_1^σ, B_2^σ, ..., B_{j-1}^σ and the first job in the batch B_j^σ. For the optimal schedule σ^*, let $P(L_j)$ denote the sum of processing times of jobs in L_j. For the schedule σ, let $P'(L_j)$ denote the sum of processing times of jobs in L_j. We want to remind the reader that $A(\alpha)$ uses a machine that has twice the processing power of the machine used by OPT. So, $P(L_j) = 2P'(L_j)$.
Now, for σ^*, we will get a lower bound on S (i.e. the sum of set-up time and processing time of jobs of type f that are processed before the completion of J_l). From the construction of σ, we can observe that every job in L_{k_f} has a due date less than or equal to the due date of job J_i. From this observation and the definition of J_l, we notice that in σ^*, all jobs in L_{k_f} are processed on or before the completion of J_l. So,

$$S \geq s_f + P(L_{k_f}) \geq 2(s'_f + P'(L_{k_f})) \tag{7}$$

Now, in order to prove this lemma, we need to express $P'(L_{k_f})$ in terms of $P'(T_f^\sigma)$. From the construction of σ and the fact that $A(\alpha)$ uses a machine with twice the processing power of OPT, we observe that for the schedule σ the sum of processing times of all jobs in $B_{k_f}^\sigma$ except the first job is $\leq s'_f$. Therefore,

$$P'(L_{k_f}) \geq P'(T_f^\sigma) - s'_f \tag{8}$$

From Equation (8) and Equation (7), we get

$$S \geq 2P'(T_f^\sigma) \tag{9}$$

Now, we will make use of the following observation to express S in terms of k_f and s'_f.
Observation 1 : $P'(L_{k_f}) \geq (k_f - 1)s'_f$.

Proof From the construction of σ and the fact that $A(\alpha)$ uses a machine with twice the processing power of OPT, we observe that for $j \in [1..k_f]$, the sum of processing times of jobs in L_j is at least $(j-1)s'_f$. Therefore, $P'(L_{k_f}) \geq (k_f - 1)s'_f$. $\square$
From Observation 1 and Equation (7), we get

$$S \geq 2k_f s'_f \tag{10}$$

From, Equations (9) and (10), we get $S \geq = 2max\{k_f s'_f, P'(T_f^\sigma)\}$.

7 Acknowledgments

The first author would like to thank Professors Eric Allender, Michael Grigoriadis, Howard Karloff, Liviu Iftode and Bill Steiger for their time and help in reviewing this paper, Professors Ken Baker, J.M.J.Schutten and Dr. Dunstall for their time and generous help in providing me copies of their papers and relevant research articles in scheduling with set-ups. We would also like to thank the National Science Foundation and DIMACS for supporting part of this research through their grants CCR-9700239, CCR-9988526 and CCR91-19999.

References

[1] Ahn, B.H., Hyun, J.H., 1990. Single facility multi class job scheduling, Computers and Operations Research, 265-272(17)

[2] Baker., K.R. and Webster, S., 1995. Scheduling groups of jobs on a single machine, Operations Research, 692-703 (43-4).

[3] Baker, K.R. and Magazine, M.J., 2000. Minimizing Maximum Lateness with Job Families, European Journal of Operational Research, 126-139(127).

[4] Baker, K.R. and Magazine, M.J., 2000. Minimizing Maximum Lateness with Job Families, European Journal of Operational Research, 126-139(127).

[5] Borodin, A. and El-Yaniv, R., 1998. Online Computation and Competitive Analysis, Cambridge University Press.

[6] Bruno, J. and P.Downey. 1978. Complexity of task sequencing with deadlines, setup times and changeover costs. SIAM Journal of Computing 7, 393-404.

[7] Charikar, M , Motwani, R, Raghavan, P, Silverstein, C. Constrained TSP and Low-Power Computing, 5th International Workshop, WADS' 97, LNCS 1272, 104-115

[8] Crauwels, H.A.J., Potts, C.N., Van Wassenhove, L.N. 1997. Local search heuristics for single machine scheduling with batch set-up times to minimize the total weighted completion time, Annals of Operations Research, 261-279(70)

[9] Divakaran, S., Saks, M., 2000, An online algorithm for scheduling with job set-ups, Dimacs Technical report, 2000-34.

[10] Dunstall, S., Wirth, A., Baker,K.R., 1999, Lower bounds and algorithms for flow minimization on a single machine with setup times, personal communication with Professor Baker and Dr.Dunstall.

[11] Fiat, A., Karp, R.M., Luby, M., McGeoch, L.A., Sleator, D.D., Young, N.E., 1988. Competitive paging algorithms, Journal of Algorithms, 12 : 685-699.

[12] Garey, M.R. and Johnson, D.S. 1979. Computers and Intractability A Guide to the Theory of NP-Completeness. W.H.Freeman and Company, San Francisco.

[13] Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., 1979. Optimization and approximation in deterministic sequencing and scheduling, Annals of Discrete Math, 5:287-326.

[14] Gupta, J.N.D., 1988. Single facility scheduling with multiple job classes, European Journal of Operational Research, 42-45 (33)

[15] Hall, L.A., Shmoys, D.B., Wein, J. 1996. Scheduling to minimize average completion time : off-line and on-line algorithms. Proceedings of the sixth ACM-SIAM Symposium on Discrete Algorithms.

[16] Hall, L.A. 1999. Approximation algorithms for scheduling, survey in Approximation algorithms for NP-hard problems, edited by Dorit S. Hochbaum.

[17] Hariri, A.M.A. and Potts, C.N. 1997. Single machine scheduling with batch set-up times to minimize maximum lateness, Annals of Operations Research 70, 75-92.

[18] Jackson, J.R., 1955., Scheduling a production line to minimize maximum tardiness, Research report, UCLA.

[19] Kalyanasundaram, B., Pruhs., K, 1995., Speed is as powerful as clairvoyance, Proceedings of 36th Annual Symposium on Foundations of Computer Science, 214-225.

[20] Lenstra, J.K., 1977. Sequencing by enumerative methods. Mathematical centre Tracts Vol 69, North Holland Publications, ISBN 90 6196 125 4.

[21] Lenstra, J.K., A.H.G.Rinnooy Kan and P.Brucker. 1977. Complexity of machine scheduling problems. Annals of Discrete Math 1, 343-362

[22] Mason, A.J., Anderson, E.J., 1991. Minimizing flow time on a single machine with job classes and set-up times. Naval research logistics 335-350(38).

[23] Monma, C.L. and C.N.Potts. 1989. On the complexity of scheduling with batch set-up times. Operations Research. 37, 798-804.

[24] Phillips, C.A., Stein, C., Torng, E., Wein, J., 1998. Optimal Time-Critical Scheduling via resource augmentation, personal communications.

[25] Phillips, C., Stein, C., Wein., 1995. Scheduling jobs that arrive over time, Proceedings of 4th WADS, Lecture notes in computer science, 955, Springer, Berlin, 86-97..

[26] Potts, C.N., Kovalyov, M.Y., 2000. Scheduling with batching : a review, European journal of operational research, 228-249 (120).

[27] Schutten, J.M.J, 1996 Shop floor scheduling with set-up times. PhD Thesis, personal communication with Professor Schutten.

[28] Schutten, J.M.J, Van de Velde, S.L., Zijm, W.H.M. 1996. Single machine scheduling with release dates, due dates and family setup times. Management Science. 42(8), 1165-1174

[29] Sleator, D.D. and Tarjan, R.E., 1985. Amortized efficiency of list update and paging rules, Communications of the ACM, 28(2):202-208.

[30] Smith, W., 1956. Various optimizers for single stage production, Naval Research Logistics Quarterly, 59-66 (3)

[31] Torng, E., 1998. A unified analysis of paging and caching, Algorithmica, 20: 175-200

[32] Young, N.E., 1991. Competitive paging and dual guided online weighted caching and matching algorithms, Ph.D. dissertation. Department of Computer Science, Princeton University.

[33] Zdrzalka,S. 1991. Approximation algorithms for single machine sequencing with delivery times and unit batch set-up times, European Journal of Operational Research 51, 199-209.

[34] Zdrzalka,S. 1995. Analysis of approximation algorithms for single machine sequencing with delivery times and sequence independent batch set-up times, European Journal of Operational Research 51, 371-380.