Principal Investigator/Project Director: Philip Kaminsky
Institution: University of California, Berkeley.
Award Number: DMI-0092854 *Program:* DMII – OR/SEE
Project Title: CAREER: Scheduling of Large Scale Systems

On-Line Algorithms for Due Date Quotation with Lateness Penalties

Zu-Hsu Lee
Philip Kaminsky
Dept. of Industrial Engineering and Operations Research
University of California, Berkeley

October 2002

Abstract

We present a novel model for due date quotation. Jobs arrive at a single server or m-machine flow shop over time. Upon arrival, the processing times on each machine of the job are known, and due dates must be quoted for each job. The jobs are sequenced on the machines without preemption. Associated with each job, there are three measures of disadvantages to the inventory cost and the service level: quoted lead time cost, earliness penalties (holding cost), and lateness (or tardiness) penalties. The objective of this problem is to minimize the sum of weighted quoted lead times, weighted earliness, and weighted lateness of all jobs. We propose an on-line heuristic, partially characterize its performance relative to a first come, first served policy, and computationally test its performance.

1 Introduction

When firms operate in a make-to-stock environment, in order to compete effectively they must set due dates which are both relatively soon in the future, and can be met reliably. Clearly, in systems for which the future is uncertain, there is an inherent tradeoff between short due dates and reliable ones. Nevertheless, the vast majority of due date scheduling research assumes that due dates for individual jobs are exogenously determined. Typically, scheduling models which involve due dates focus on sequencing jobs at various stations in order to optimize some measure of the ability to meet the given due dates. However, in practice, firms need an effective approach for quoting lead times, and for sequencing jobs to meet these lead times.

There have been a variety of models of due date quotation which have appeared in the literature. In some of these models, it is assumed that customers place orders independent of the length of the quoted lead time, where the quoted lead time is typically defined to be the time span from the arrival of a job until

its quoted start-processing time. The objective in this case is frequently to minimize average quoted lead time subject to some constraint on the number of tardy jobs or the amount of tardiness [1]. The majority of papers based on these models have been simulation based. For instance, Eilon and Chowdhury [2], Weeks [3], Miyazaki [4], and Bertrand [5] consider various combinations of due date assignment and sequencing policies, and in general demonstrate that policies which use estimates of shop congestion and job content information lead to better shop performance than policies based solely on job content. Some polynomial algorithms and analytical results do exist for limited versions of these models ([6], [7], [8], [9], [10], and [11]), which consist of deterministic, common due date models, and static models (all jobs are available at the start). However, these algorithms and analytical results focus on very restricted versions of the problem, and on single processor instances (see Lee [12] for a detailed discussion of references).

In other models, not all potential jobs are processed, and customers' responses are taken into account. For instance, the Lead Time Quotation (LTQ) problem [13] assumes that a maximum lead time is associated with each customer and the firm decides whether or not to accept the order. If the order is accepted, the processing of the order has to be started within its quoted lead time to ensure 100% reliability. The objective typically involves a revenue function which decreases with increasing quoted lead times. Effective algorithms exist for special cases of this problem, but not for the most general cases. Some similar approaches to LTQ assume that given a quoted lead time, the customer decides whether or not to place an order. The probability that a customer places an order decreases with increasing lead time. The objective is a function of revenue associated with each job, and a penalty for late jobs. These models have generally been considered in a queueing theoretic framework (see [14], [15], [16], and [17]). As with the approaches described above, these approaches have typically been limited to single machine or processor models.

Kaminsky and Lee [18] [19] introduce a new due date quotation model (DDQP) for both single machine and flow shop cases, which captures some of the key elements of the models and approaches above. In their model, jobs or orders arrive to a single server, representing a manufacturing organization, over time. All jobs are accepted, and due dates must be quoted immediately upon job arrival. The objective is to minimize average quoted lead time (or quoted due date), and all due dates are met. Based on this 100% reliable due date quotation model, they develop an on-line due date quotation algorithm (DDQ) with several variations, characterize the asymptotic performance of this algorithm, and then analyze asymptotic probabilistic bounds on its performance.

However, in the real world, a manufacturer might want to postpone some current waiting jobs past there due date, thus incurring penalties, in order to start processing "better" or higher priority jobs. The requirement of 100% reliability may not accurately model the problem facing the manufacturer. In order to capture this ability to allow jobs to complete late in the due date quotation problem, we generalize the DDQP model for both single machine and flow shop cases. In subsequent sections of this paper, we develop a on-line due date quotation algorithm for this generalized model, analyze bounds on its performance, and

then present computational results which demonstrate the effectiveness of the algorithm. In the next section, we formally present the model.

2 The Model

We define the general due date quotation problem, **GDDQP**, in the m-machine flow shop and in the single machine case (i.e. $m = 1$ flow shop), as follows. Jobs arrive over time and due dates are quoted upon job arrival. Jobs must be sequentially processed on m machines, without preemption on any of the machines. Each machine can handle at most one job at a time, and a job can be processed on at most one machine at a time. Associated with each job i, there is a release time r_i (the arrival time of the job at the system) and a set of processing times on all machines (p_{ij}, $j = 1, 2, \ldots, m$). As in other literature, we define quoted lead time to equal the upper bound on the length of the time within which the job is expected to start processing after it arrives, $d_i - \sum_{j=1}^{m} p_{ij} - r_i$ for job i. We know that the quoted lead time affects customers' satisfaction as they prefer shorter lead times. Furthermore, the earliness and lateness of a job may lead to increased holding costs, as well as late costs and decreased customer satisfaction. In order to model these elements of the problem, we let w_i^c, w_i^h, and w_i^l to be weights associated with the quoted lead time, earliness, and lateness of job i respectively. In our model, the parameters, w_i^c, w_i^h, and w_i^l are known upon the arrival of job i. The objective is to determine a schedule of jobs on all machines, and due dates for each job so that the sum of the weighted quoted lead times, the weighted earliness, and the weighted lateness of all the jobs is minimized.

We formally present the model below:

Parameters:

p_{ij} = processing time of job i on machine j
r_i = release time of job i

Decision Variables:

d_i = quoted due date of job i
C_{ij} = completion time of job i on machine j

$$
\begin{aligned}
(\textbf{GDDQP}) \quad & \min \textstyle\sum_{i=1}^{n} w_i^c (d_i - \sum_{j=1}^{m} p_{ij} - r_i) + \sum_{i=1}^{n} w_i^h (d_i - C_{im})^+ + \sum_{i=1}^{n} w_i^l (C_{im} - d_i)^+ \\
& s.t. \\
& C_{i1} \geq r_i + p_{i1} && i = 1, 2, \ldots, n \\
& C_{i,k+1} \geq C_{ik} + p_{i,k+1} && k = 1, \ldots, m-1 \\
& C_{ik} - C_{jk} \geq p_{ik} \quad or \quad C_{jk} - C_{ik} \geq p_{jk} && i, j = 1, 2, \ldots, n, i \neq j, k = 1, \ldots, m
\end{aligned}
$$

Note that $(d_i - C_{im})^+$ and $(C_{im} - d_i)^+$ stand for the earliness and lateness respectively, where $x^+ = (x + |x|)/2$.

Clearly, if all job information is known in advance, the entire processing sequences on all machines can be determined in advance, and the due date for each job can be set in order to minimize the objective for this schedule. We call the version of this problem for which all information is known in advance the *off-line*

case. In the off-line case, given a feasible schedule Π, d_i will be determined according to the following policy so that the objective value given Π is minimized:

- If $i \in \{j | w_j^c \geq w_j^l\}$, d_i is set to $r_i + \sum_{j=1}^{m} p_{ij}$.
- If $i \in \{j | w_j^c < w_j^l\}$, d_i is set to C_{im}.

Therefore the objective value for schedule Π will be

$$\sum_{i \in \{j | w_j^c \geq w_j^l\}} w_i^l (C_{im} - \sum_{j=1}^{m} p_{ij} - r_i) + \sum_{i \in \{j | w_j^c < w_j^l\}} w_i^c (C_{im} - \sum_{j=1}^{m} p_{ij} - r_i).$$

Note that any given instance, r_i and $\sum_{j=1}^{m} p_{ij}$ for all job i are constants, so the off-line version of our general due date quotation model is equivalent to minimizing the total weighted completion time, $\sum_{i=1}^{n} w_i C_{im}$, where $w_i = \min\{w_i^c, w_i^l\}$.

In contrast to this off-line case, in the on-line case, each scheduling and due date quotation decision must be made utilizing only information pertaining to jobs which have been released by the time that the decision is made. Thus, the entire processing schedule cannot be determined in advance, and quoted due dates must take into account an estimate of future job arrivals. Thus, it is much more difficult to quote an effective on-line schedule.

Also, although the off-line case is equivalent to minimizing total weighted completion time, this is itself an NP-hard problem. However, heuristic WSPTA (Weighted Shortest Processing Time among Available jobs) can solve this problem efficiently. WSPTA suggests that, every time we need to schedule a job when the machine is idle, we consider all the jobs that have been released but not yet processed and select the job with the smallest ratio of $\sum_{j=1}^{m} p_{ij}/w_i$ to be processed next (w_i stands for the weight of job i). Chou and Simchi-Levi [20] show that, under the assumption that job processing times and weights are bounded, WSPTA provides an asymptotically solution for the single machine ($m = 1$) total weighted completion time minimization problem. However, for the flow shop ($m > 1$) total weighted completion time minimization problem, Liu and Simchi-Levi [21] show that the WSPTA permutation is asymptotically optimal under certain the probabilistic assumptions on the problem instance: p_{ij} are bounded i.i.d. for any i, j, all weights are i.i.d. bounded on $(0, 1]$, and inter-arrival times $t_i = r_i - r_{i-1}$ for all i are non-negative and bounded. Also, we note that under these conditions, the time a job remains in the flow shop after completing processing on the first machine will be $o(n)$, as demonstrated by Xia, Shanthikumar, and Glynn [22].

In the following sections, we restrict our attention to the set of permutation schedules for flow shop case. A H permutation schedule is defined as follows: once the sequence of all jobs on the first machine is determined by some heuristic H, jobs are processed on all other machines on a first come first serve (FCFS) basis. In other words, jobs are processed in the same sequence on all machines.

3 On-Line Algorithm Development

In this section we develop an on-line algorithm for quoting due dates and sequencing instances of the general due date quotation problem. The algorithm quotes due dates and sequences for any deterministic instance of this problem.

Recall that Kaminsky and Lee [18] [19] propose several variations of on-line algorithms for the single machine and flow shop cases with 100% reliable due date quotation. However, here we propose generalized due date quotation algorithms, **GDDQ**, for single machine and flow shop respectively where some jobs can complete after their assigned due dates.

The on-line algorithm, GDDQ, works as follows. The queue in front of the first machine is sequenced. Whenever a job completes processing on the first machine, the next job in the queue in front of the first machine is processed (note that the first machine means the only machine in the single machine case). In the flow shop case, all jobs are processed on all machines in the same order as on the first machine, as quickly as possible (without delays). When a new order arrives, we apply a sequencing and due date setting rule **SDDS**, which is explained below in Subsection 3.2, to insert it into the processing sequence in the queue in front of the first machine (that is, its relative sequencing priority is determined) and to quote a the due date for the job. Note that when we refer to a job's priority, we mean its position in the sequence, so that the job with priority 1 is sequenced first in the queue. GDDQ can be regarded as an extension of the DDQ algorithm proposed by Kaminsky and Lee [18] [19].

3.1 The Structure of On-Line Algorithm GDDQ

Before we formally state the algorithm, we introduce the following notation:

- $Q'(t)$ = the set of the jobs in queue on the first machine in priority sequence immediately before time t
- $Q(t)$ = the set of the jobs in queue on the first machine in priority sequence at time t
- $D'(t)$ ($D(t)$) = the set of quoted due dates for the jobs in $Q'(t)$ ($Q(t)$)
- $J'_{(k)}$ ($J_{(k)}$) = the job with k^{th} priority in $Q'(t)$ ($Q(t)$)
- d_i = the quoted due date of job i
- $seq(l)'$ ($seq(l)$) = the priority of job l in $Q'(t)$ ($Q(t)$). Note that the priority 1 job is first to be processed, the priority 2 job is next in line to be processed, etc.
- EC_{lj} = the earliest possible completion time on machine j for job l in $Q(t)$ (accounting for all jobs which need to be processed ahead of job l at time t)

- $tp_i = \sum_{j=1}^{m} p_{ij}$ = the total processing time over all machines of job i

Note that $Q(t)$, $Q'(t)$, $D(t)$, $D'(t)$, $J_{(k)}$, $J'_{(k)}$, $seq(l)$, $seq(l)'$, and EC_{ij} are dynamic – they change with time. Moreover, $Q(t) = Q'(t)$, $D(t) = D'(t)$, $J_{(k)} = J'_{(k)}$, and $seq(l) = seq(l)'$ only if there is no job arrival at time t.

Algorithm 1 GDDQ for Single Machine

while job arrival or completion events are still occurring **do**
  if the event is the arrival of job j with processing time, $tp_j = p_{j1}$, at time r_j **then**
    if the machine is idle **then**
      · process job j on the machine immediately
      · quote due date d_j equal to the completion time, $d_j = r_j + tp_j$
    else
      · a job i is in process on the machine to be completed at C_i
      · apply **SDDS** to determine the position of job j in the queue of the machine, q_j, and its due date d_j
      ($Q(r_j)$ updated from $Q'(r_j)$, $D(r_j)$ updated from $D'(r_j)$)
    end if
  end if
  if the event is the completion of processing of job i **then**
    · process the job waiting with priority 1
    · advance the priority of all remaining waiting jobs by 1
  end if
end while

Formally, Algorithm 1: GDDQ and Algorithm 2: GDDQ describe the procedure which is completed each time a *job completion* or *job arrival* event occurs for the single machine and flow shop cases, respectively. Note that in the GDDQ algorithm, if a job is assigned a due date equal to its earliest completion time (that is, there is no slack time associated with the due date assignment), the job may be complete either just in time or late. In that case, holding cost will not be incurred.

3.2 The Sequencing and Due Date Setting Rule

We describe sequencing and due date setting rule **SDDS** below. Consider a queue of waiting (sequenced) jobs and a newly arriving job. The waiting jobs have each been assigned process priorities and due dates upon their arrival. The job in queue with the 1^{st} priority will be processed when the currently processing job is completed. In the spirit of the WSPTA permutation schedule in the flow shop and WSPTA for the single machine, on the first machine, we would prefer to process the jobs with a smaller ratio of total processing time to weight before the jobs with larger ratios. This is because a WSPTA sequence is effective (indeed, asymptotically optimal) for the off-line version of this problem. In other words, when a job with a smaller ratio of total processing time to weight is switched ahead of a job with a larger ratio in the queue, this new sequence will in many cases provide a better objective value than the sequence before the switch. In SDSS, when a new job arrives, we try to move it ahead of jobs with larger processing time to weight ratios without

Algorithm 2 GDDQ for Flow Shop

while job arrival or completion events are still occurring **do**
  if the event is the arrival of job j with processing times, $p_{j1}, p_{j2}, \ldots, p_{jm}$, at time r_j **then**
    if machine 1 is idle **then**
      · process job j on machine 1 immediately
      · job j will be processed sequentially on machines $2, \ldots, m$ immediately after previous jobs are completed
      · quote due date d_j equal to the completion time of job j on machine m
    else
      · a job i is in process on machine 1
      · apply **SDSS** to determine the position of job j in the queue of machine 1, q_{j1}, and its due date d_j ($Q(r_j)$ updated from $Q'(r_j)$, $D(r_j)$ updated from $D'(r_j)$)
    end if
  end if
  if the event is the completion of processing of job i on machine l **then**
    if machine $l+1$ is idle **then**
      · process job i on machine $l+1$
    else
      · put i at the end of the queue of machine $l+1$
    end if
    · process the job waiting with priority 1 on machine l
    · advance the priority of all remaining waiting jobs on machine l by 1
  end if
end while

making the objective value increase. In essence, we try to increase the new job's priority by one position at a time to the best possible priority. We try to move it up by two if we can't move it up by one, by three if we can't move it up by two, and so on. Once we have determined the job's position in queue, we assign it a due date based on its position in queue, perhaps with some additional slack to account for later arrivals.

Suppose there is a set of m_j waiting jobs on the first machine, $Q'(r_j)$, when job j arrives. If job j is inserted into position k and quoted date date d_j, that is, $Q'(r_j)$ ($D'(r_j)$) is updated to $Q(r_j)$ ($D(r_j)$) with job j assigned priority k in the queue and due date d_j, we denote by f_j^k the current lowest possible contribution to the objective by job j and the waiting jobs sequenced after it. Hence, we know d_j should be the earliest possible completion time at the last machine if $w_j^c < w_j^l$, and d_j should be $r_j + tp_j$ otherwise. Let $w_j = \min\{w_j^c, w_j^l\}$ and we have

$$\begin{aligned} f_j^k &= w_j(EC_{jm} - tp_j - r_j) + \sum_{l=k+1}^{m_j+1} w_{J(l)}^c (d_{J(l)} - r_{J(l)} - tp_{J(l)}) + \sum_{l=k+1}^{m_j+1} w_{J(l)}^h (d_{J(l)} - EC_{J(l)m})^+ \\ &\quad + \sum_{l=k+1}^{m_j+1} w_{J(l)}^l (EC_{J(l)m} - d_{J(l)})^+. \end{aligned}$$

EC_{jm} can be calcualted based on the current information in the shop. For instance, in the single machine case, if job j is inserted in the queue at position k, the earliest possible completion time EC_{j1} above will be $C_i + \sum_{l=1}^{k-1} tp_{J(l)} + tp_j$, where C_i is the completion time of the current job i in process.

Algorithm SDDS takes as input:

- a new job j, the j^{th} arrival, arriving at time r_j with processing times, $p_{j1}, p_{j2}, \ldots, p_{jm}$
- $Q'(r_j)$ (the new job j not included), $|Q'(r_j)| = m_j$
- $D'(r_j)$
- The identity (and other associated information) of the jobs currently in process and in queue on all of the machines

and produces as output

- a priority sequence, $Q(r_j)$, updated from $Q'(r_j)$ with the newly prioritized job j
- a set of the due dates of the jobs in $Q(r_j)$, $D(r_j)$, updated from $D'(r_j)$

Before stating SDDS, we first define a subroutine. This subroutine moves the job with priority v to priority u ($v < u$) and the job with priority u to priority v.

Subroutine SWITCH(v,u)

$seq(J_{(u)}) \leftarrow m_j + 2$
$seq(J_{(v)}) \leftarrow u$
$seq(J_{(m_j+2)}) \leftarrow v$

A formal description of the sequencing and due date setting rule is listed as Algorithm 3: SDDS. We define the weight w_i as $\min\{w_i^c, w_i^l\}$ for job i. The algorithm outputs u, the position of the new job in the machine one queue, equal to q_{j1} for flow shop, q_j for single machine. Note that in SDDS, when arrival j has been sequenced at position u and the algorithm is considering a switch with the job at position v, the quantity $g_j^u - f_j^v$ represents a bound on the decrease in objective value if this switch is made, and UB_j is a bound on the decrease in objective value due to accumulated switches. Also, once the position of job j is determined, its due date is set equal to its earliest possible completion time plus some slack equal to:

$$SL_j = \beta_j \frac{UB_j}{w_j^c + w_j^h}$$

where β_j is an algorithm parameter, and $\frac{UB_j}{w_j^c + w_j^h}$ is selected in order to make it possible to characterize the performance of this algorithm in Section 4.

We have three simple β selection rules for choosing β_j for job j:

Algorithm 3 SDDS

for $k = 1, 2, \ldots, m_j$ **do**
  $seq(J'_{(k)}) \leftarrow k$
end for
$seq(j) \leftarrow m_j + 1$
$u \leftarrow m_j + 1$
$v \leftarrow u - 1$
while $(tp_j/w_j < tp_{J_{(v)}}/w_{J_{(v)}})$ and $(u > 0)$ and $(v > 0)$ **do**
  $g_j^u \leftarrow f_j^u + \sum_{k=v}^{u-1} w_{J(k)}^c (d_{J(k)} - r_{J(k)} - tp_{J(k)}) + \sum_{k=v}^{u-1} w_{J(k)}^h (d_{J(k)} - EC_{J(k)m})^+ + \sum_{k=v}^{u-1} w_{J(k)}^l (EC_{J(k)m} - d_{J(k)})^+$
  SWITCH(v,u)
  determine $EC_{J_{(k)}m}$ for $k = v+1, \ldots, m_j + 1$
  $UB_j \leftarrow 0$
  if $f_j^v < g_j^u$ **then**
    $UB_j \leftarrow UB_j + (g_j^u - f_j^v)$
    $u \leftarrow v$
    $v \leftarrow u - 1$
  else
    SWITCH(v,u)
    $v \leftarrow v - 1$
  end if
end while
if $w_j^c < w_j^l$ **then**
  · choose β_j s.t. $0 \le \beta_j \le 1$ and $SL_j = \beta_j \frac{UB_j}{w_j^c + w_j^h}$
  · quote due date $d_j \leftarrow EC_{jm} + SL_j$
else
  · quote due date $d_j \leftarrow r_j + tp_j$
end if

R1. Assign β_j to 1 for all j.

R2. Assign β_j to 0 for all j.

R3. Let $\bar{W}^c_{(j)}$ be the estimate of the expected weight of w^c_k for the jobs k which have arrived before j. Assign β_j to 1 for job j if $w^c_j < \bar{W}^c_{(j)}$, 0 if $w^c_j \geq \bar{W}^c_{(j)}$.

R3 is motivated by the observation that slack is most beneficial when lead time weight is less significant. We present the results of computational tests of theses three rules in Section 5.

4 Characterizing the Performance of the Algorithms

Although the heuristic presented in the previous section will not necessarily lead to optimal due dates and sequences, it is useful to analytically characterize its performance. In this section, we bound the performance of this algorithm by relating the performance of the GDDQ permutation schedule to that of the First Come First Served (FCFS) permutation schedule for flow shop, the performance of the GDDQ schedule to that of the First Come First Served (FCFS) schedule on single machine.

Note that we can easily quote a due date equal to the completion time for a FCFS sequence upon job arrival. Thus, the objective value of FCFS permutation schedule for flow shop or FCFS for single machine ($m = 1$) is

$$Z^{FCFS}_m = \sum_{i=1}^{n} w^c_i (C^{FCFS}_{im} - r_i - tp_i),$$

where C^H_{im} represents the completion time of job i on the last machine in the H permutation schedule ($m > 1$) or the the completion time of job i on the single machine ($m = 1$).

Now, we introduce a heuristics, the Modified First Come First Served (MFCFS) heuristic. We process the jobs in FCFS sequence. However, when job i arrives, the quoted due date equals $r_i + tp_i$ if $w^c_i \geq w^l_i$, and C^{MFCFS}_{im} otherwise. Clearly, the objective value of the MFCFS permutation schedule for flow shop or MFCFS for the single machine is

$$Z^{MFCFS}_m = \sum_{i=1}^{n} w_i (C^{FCFS}_{im} - r_i - tp_i),$$

where $w_i = \min\{w^c_i, w^l_i\}$. Then we have

$$Z^{MFCFS}_m \leq Z^{FCFS}_m.$$

Recall that the GDDQ algorithm actually incorporates sequencing policy with the idea behind MFCFS. Whenever the new arrival j at priority u is switched with a waiting job at priority v in the queue ($v < u$), the due date of job j is reset to the earliest possible completion time at position v from the earliest possible

completion time at position u if $w_j^c < w_j^l$; otherwise, the due date remains at $r_j + tp_j$. Moreover, SDDS always ensures that this action will bring the objective value lower by $g_j^u - f_j^v$ (recall the condition in SDDS, $f_j^v < g_j^u$, for this action) and the total amount of decrease due to switched is captured by UB_j. Therefore, sequencing j using SDDS improves the objective value by UB_j over putting j at the end of the queue. However, we also assign additional slack SL_j to the due date of job j. We note that when job j has $w_j^c < w_j^l$, the objective value will increase by $w_j^c SL_j + w_j^h SL_j$ if the due date is assigned to $EC_{jm} + SL_j$ instead of EC_{jm}. This increase is bounded by UB_j because $SL_j \leq UB_j/(w_j^c + w_j^h)$. This observation and the fact that we only move a job from the last position in the queue if the objective (ignoring this job's slack) decreases implies that:

$$Z_m^{GDDQ} \leq Z_m^{MFCFS} \leq Z_m^{FCFS}.$$

That is, our algorithm will perform no worse than the commonly used FCFS algorithm.

5 Computational Testing

In the previous section, we concluded that the objective value of the FCFS schedule for single machine or the FCFS permutation schedule for flow shop is an upper bound on that of the GDDQ schedule for single machine or the GDDQ permutation schedule for flow shop. In this section, we complete limited computational tests in order to explore the effectiveness of our GDDQ algorithm.

Thus far, we have completed experimental analysis of the single machine case. The instances are generated with weights, inter-arrival times, and processing times from exponential distribution and uniform distribution. Without loss of generality, we set the arrival time of the first job to time 0. In Tables 1 and 2, we present the ratios of GDDQ to MFCFS objectives, and GDDQ to FCFS objectives, for a variety of problem parameters. Each trial provides this pair of ratios for each β selection rule (R1, R2, and R3), based on the same set of problem parameters. Each ratio in the Tables is the average of 3 trials. In all cases, we see that our GDDQ algorithm outperforms both FCFS and MFCFS approaches. Of course, performance level will depend on instance parameters, and we have only tested a limited set of instances. Also, although we have not yet completed analysis of the flow shop model, our experience in [19] suggests that the performance of GDDQ in for these more complex models will be similar.

6 Conculsion

We have developed a conceptual model for the general due date quotation problem both on a single machine and in a flow shop, proposed an on-line heuristic for this model, analyzed the performance of this heuristic, and completed limited computational testing of the heuristic. It is appropriate to point out that the analysis of the performance of GDDQ holds for any instances without any probabilistic assumptions. However, the

p_{min}	p_{max}	ET	EW^c	EW^h	EW^l	$\frac{EP}{ET}$	β	$n = 500$		$n = 2000$	
								$\frac{Z_1^{GDDQ}}{Z_1^{MFCFS}}$	$\frac{Z_1^{GDDQ}}{Z_1^{FCFS}}$	$\frac{Z_1^{GDDQ}}{Z_1^{MFCFS}}$	$\frac{Z_1^{GDDQ}}{Z_1^{FCFS}}$
0.5	3.5	0.5	0.5	0.5	0.5	4	R1	0.7136	0.4842	0.7175	0.4799
							R2	0.5484	0.3722	0.5542	0.3707
							R3	0.5904	0.4007	0.5965	0.3990
0.5	1.5	0.5	0.5	0.5	0.5	2	R1	0.6966	0.4727	0.6916	0.4612
							R2	0.4795	0.3255	0.4838	0.3225
							R3	0.5408	0.3670	0.5447	0.3632
0.2	0.6	0.5	0.5	0.5	0.5	0.8	R1	0.8475	0.5767	0.8388	0.5652
							R2	0.7383	0.5019	0.6930	0.4669
							R3	0.7819	0.5314	0.7469	0.5032

Table 1: Uniform $[0, 2ET]$ inter-arrival times, Uniform $[0, 2EW^c]$ weights of quoted lead time, Uniform $[0, 2EW^h]$ weights of holding cost, Uniform $[0, 2EW^l]$ weights of lateness, and Uniform $[p_{min}, p_{max}]$ processing times

p_{min}	p_{max}	ET	EW^c	EW^h	EW^l	$\frac{EP}{ET}$	β	$n = 500$		$n = 2000$	
								$\frac{Z_1^{GDDQ}}{Z_1^{MFCFS}}$	$\frac{Z_1^{GDDQ}}{Z_1^{FCFS}}$	$\frac{Z_1^{GDDQ}}{Z_1^{MFCFS}}$	$\frac{Z_1^{GDDQ}}{Z_1^{FCFS}}$
0.5	3.5	0.5	0.5	0.5	0.5	4	R1	0.7156	0.4856	0.7084	0.4710
							R2	0.5487	0.3724	0.5480	0.3644
							R3	0.5886	0.3994	0.5889	0.3915
0.5	1.5	0.5	0.5	0.5	0.5	2	R1	0.6900	0.4524	0.6883	0.4603
							R2	0.5037	0.3304	0.4726	0.3160
							R3	0.5546	0.3638	0.5356	0.3582
0.2	0.6	0.5	0.5	0.5	0.5	0.8	R1	0.7657	0.5227	0.7814	0.5181
							R2	0.5929	0.4058	0.5862	0.3890
							R3	0.6391	0.4371	0.6469	0.4291

Table 2: Exponential inter-arrival times with rate $1/ET$, Uniform $[0, 2EW^c]$ weights of quoted lead time, Uniform $[0, 2EW^h]$ weights of holding cost, Uniform $[0, 2EW^l]$ weights of lateness, and Uniform $[p_{min}, p_{max}]$ processing times

computational results clearly depend on the problem instances.

In future research, we hope to build on the insights in this paper as we consider more complex models and objectives in more complex shop environments.

References

[1] Cheng, T.C.E. and M.C. Gupta (1989), Survey of Scheduling Research Involving Due Date Determination Decisions. *European Journal of Operational Research* **38**, pp. 156-166.

[2] Eilon, S. and I.G. Chowdhury (1976), Due Dates in Job Shop Scheduling. *International Journal of Production Research* **14**, pp. 223-237.

[3] Weeks, J.K. (1979), A Simulation Study of Predictable Due-Dates. *Management Science* **25**, pp. 363-373.

[4] Miyazaki, S. (1981), Combined Scheduling System for Reducing Job Tardiness in a Job Shop. *International Journal of Production Research* **19**, pp. 201-211.

[5] Bertrand, J.W.M. (1983), The Effect of Workload Dependent Due-Dates on Job Shop Performance. *Management Science* **29**, pp. 799-816.

[6] Brucker, P. (1998), *Scheduling Algorithms*. Springer, New York.

[7] Kahlbacher, H. (1992), Termin-und Ablaufplanung - ein analytischer Zugang, *Ph.D. thesis*, University of Kaiserslautern, cited in [6]

[8] Panwalkar, S.S., M.L. Smith, and A. Seidmann (1982), Common Due Date Assignment to Minimize Total Penalty for the One Machine Scheduling Problem. *Operations Research* **30**, pp. 391-399.

[9] Hall, N.G. and M.E. Posner (1991), Earliness-Tardiness Scheduling Problems, I: Weighted Deviation of Completion Times about a Common Due Date. *Operations Research* **39**, pp. 836-846.

[10] Seidmann, A., S.S. Panwalker, and M.L. Smith (1981), Optimal Assignment of Due Dates For a Single Processor Schedluing Problem. *International Journal of Production Research* **19**, pp. 393-399.

[11] Chand, S. and D. Chhajed (1992), A Single Machine Model for Determination of Optimal Due Date and Sequence. *Operations Research* **40**, pp. 596-602.

[12] Lee, Z. (2002) Design and Analysis of Algorithms for Due Date Quotation. *PhD Thesis*, University of California, Berkeley. In process.

[13] Keskinocak, P., R. Ravi, and S. Tayur (1997), Algorithms for Reliable Lead Time Quotation. *GSIA Working Paper*, Carnegie Mellon University, Pittsburgh, PA.

[14] Wein, L. M. (1991) Due-date Setting and Priority Sequencing in a Multiclass M/G/1 Queue. *Management Science* **37**, pp. 834-850.

[15] Duenyas, I. (1995), Single Facility Due Date Setting with Multiple Customer Classes. *Management Science* **41**, pp. 608-619.

[16] Duenyas, I and W.J. Hopp (1995), Quoting Customer Lead Times. *Management Science* **41**, pp. 43-57.

[17] Spearman, M., and R.Q. Zhang (1999), Optimal Lead Time Policies. *Management Science* **45**, pp. 290-295.

[18] Kaminsky, P. and Z. Lee (2001), Anaysis of On-Line Algorithms for Due Date Quotation. *Submitted Paper*, University of California, Berkeley, CA.

[19] Kaminsky, P. and Z. Lee (2002), On-Line Algorithms for Flow Shop Due Date Quotation. *Submitted Paper*, University of California, Berkeley, CA.

[20] Chou, C.F. and D. Simchi-Levi (1999), The Asymptotic Performance of an On-Line Algorithms for the Single Machine Weighted Completion Time Problem with Release Dates. *Working Paper*, Northwestern University.

[21] Liu, H. and D. Simchi-Levi (1999), On The Asymptotic Optimality of On-Line Algorithms for The Flow Shop Problem with Release Dates. To appear in *Operations Research*.

[22] Xia, C., G. Shanthikumar, and P. Glynn (2000), On The Asymptotic Optimality of The SPT Rule for The Flow Shop Average Completion Time Problem. *Operations Research* **48**, pp. 615-622.