J Sched (2008) 11: 187–204
DOI 10.1007/s10951-008-0053-0

Effective on-line algorithms for reliable due date quotation and large-scale scheduling

Philip Kaminsky · Zu-Hsu Lee

Published online: 26 January 2008

Abstract We consider the sequencing of a series of jobs that arrive at a single processor over time. At each job's arrival time, a due date must be quoted for the job, and the job must complete processing before its quoted due date. The objective is to minimize the sum (or average) of quoted due dates, or equivalently, the average quoted lead time. In this paper, we propose on-line heuristics for this problem and characterize the conditions under which these heuristics are asymptotically optimal. Computational testing further demonstrates the relative effectiveness of these heuristics under various conditions.

Keywords Due date quotation · Scheduling · On-line algorithms · Asymptotic analysis · Probabilistic analysis

1 Introduction

When firms operate in a make-to-order environment, in order to compete effectively they must set due dates (or lead times) 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 that involve due dates focus on sequencing jobs 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 due dates (or lead times), and for sequencing jobs in order to meet these due dates. In this paper, we consider a scheduling model that contains elements of this practical problem, develop heuristics for this model, and analyze the performance of these heuristics.

Researchers have introduced a variety of models in an attempt to understand effective due date quotation. In some models, the assumption is made that orders will be placed 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-of-processing time (or sometimes, quoted due date or end-of-processing time, although in this paper we use quoted start-of-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. The majority of papers based on these models have been simulation based. For instance, Eilon and Chowdhury (1976), Weeks (1979), Miyazaki (1981), Baker and Bertrand (1981), Bertrand (1983), Hsu and Sha (2004), and Chang et al. (2005) consider various 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 analytical results do exist for limited versions of these models. Primarily, these consist of deterministic, common due date models, where a single due date must be assigned for all jobs, and static models, where all jobs are available at time 0. For these simplified models, a variety of polynomial algorithms have been developed (see Brucker

Both authors made equal contributions to this paper and are listed in alphabetical order.

P. Kaminsky (✉)
Department of Industrial Engineering and Operations Research, University of California, Berkeley, USA
e-mail: kaminsky@ieor.berkeley.edu

Z.-H. Lee
Department of Management and Information Systems, Montclair State University, Montclair, NJ 07043, USA
e-mail: leez@mail.montclair.edu

1998; Kahlbacher 1992; Panwalkar et al. 1982; Hall and Posner 1991; Seidmann et al. 1981; Chand and Chhajed 1992; Portougal and Trietsch 2006); however, these results don't extend in an obvious way to more complex models.

In other models, not all potential jobs are processed. In some of these models, a maximum lead time is associated with each customer. The firm decides whether or not to accept a particular offer, and if the offer is accepted, a perfectly reliable lead time (which must be less than the customer's maximum lead time) is quoted. In other words, an accepted job has to start processing within its quoted lead time. The objective typically involves a revenue function which decreases with increasing quoted lead times. This is known as the Lead Time Quotation (LTQ) problem (see Keskinocak et al. 2001 and the references therein). Again, effective algorithms exist for special cases of this problem, but not for the most general cases.

Some researchers consider LTQ models within a queuing theoretic framework. For example, Wein (1991) considers a multiclass M/G/1 queuing system under the objective of minimizing the weighted average lead time subject to the constraints of the maximum fraction of tardy jobs and the maximum average tardiness. Spearman and Zhang (1999) further characterize heuristic performance for these types of systems. In order to capture the impact of quoted lead times on demand, some models 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. Duenyas and Hopp (1995) consider such a system using a queuing model, and provide effective heuristics under various problem characteristics. In their model, there is a single class of customers; the net revenue per customer is constant, customers have the same preferences for lead times, and the arrival and processing times follow the same distribution. Duenyas (1995) extends their results to multiple customer classes, with different net revenues and lead time preferences, and Slotnick and Sobel (2005) consider the value of information in such systems. Kaminsky and Hochbaum (2004) and Cheng and Gupta (1989) survey due date quotation models in detail.

In this paper, we introduce a simple due date quotation model that aims to capture some of the key elements of a variety of the models and approaches described above. A set of jobs that arrive over time must be processed nonpreemptively on a single machine. Each job has an associated processing time and arrival time, and no job can be processed before its arrival time. At its arrival time, each job is assigned a due date. This models a plant to which customer orders arrive over time. When a customer arrives, the processing time of the customer's job is known, and plant management must quote to the customer a date by which the job will be completed. Clearly, in order to continue to attract business, the firm must achieve two goals: on average, quoted lead times must be less than those of competitors, and quoted lead times must be met. Hence, our model requires that all due dates be met, with the objective of minimizing average quoted due date. We call this model the 100% Reliable Due Date Quotation (RDDQ) model. We propose three on-line heuristics, which we call First Come First Serve Quotation (FCFSQ), Sequence/Slack I (SSI), and Sequence/Slack II (SSII), and analyze these heuristics employing the tools of probabilistic analysis, randomly generating instances of the problem, so that each instance resembles a realization of a finite G/G/1 queuing system. Our heuristic design is intimately bound up with our analysis, which focuses on the limiting behavior of a heuristic as the size of instances gets very large—in other words, we design our heuristics to have good asymptotic probabilistic performance. This limiting behavior can be used to infer the average-case performance of the heuristic for large-scale finite problems, and we confirm this behavior with computational testing. Under certain conditions, these heuristics provide asymptotically optimal solutions for the RDDQ model.

To put our work into perspective, we highlight other research relating to probabilistic analysis of scheduling problems. To the best of our knowledge, none of this work has focused on due date quotation models, and the vast majority has focused on relatively simple objectives, and the analysis of relatively simple algorithms. Much of this work has focused on parallel machine problems, including the work of Coffman et al. (1982), Loulou (1984) and Frenk and Rinnooy Kan (1987), who analyze the parallel machine scheduling problem when the objective is to minimize the makespan, and Spaccamela et al. (1992) and Webster (1993), who analyze the parallel machine *weighted completion time* model. Ramudhin et al. (1996) who analyze the 2-machine flow shop makespan model. Kaminsky and Simchi-Levi (1998, 2001), Xia et al. (2000), and Liu et al. (2005) analyze the flow shop average completion time problem with release dates. Finally, Kaminsky (2003) considers the flow shop delivery time problem, which is closely related to the flow shop maximum lateness problem, a model which involves due dates, although they are given rather than determined by the model.

Our model uses the objective of average due date for ease of exposition, by relating it to the total completion time problem. We note at the outset that in some cases it may be more realistic to focus on minimizing average lead time (due date minus arrival time) rather than minimizing average due date for probabilistic analysis. Although given an instance, these two objectives are equivalent, in some cases the rate of growth of average lead time will be different than that of average due date, and thus their asymptotic behavior

can be different. This implies that our analysis and heuristic design may need modification when average lead time is considered. We leave this problem for future research.

In the next section, we introduce our model.

2 The model and asymptotic lower bound

We formally define the RDDQ model as follows. Consider a set of n jobs that have to be processed one at a time on a single machine without preemption. Each job $i, i = 1, 2, \ldots, n$, is characterized by a processing time p_i and arrival time (or release date) r_i, where $r_1 < r_2 < \cdots < r_n$, and no job can begin processing before its arrival time. At its arrival time, each job is assigned a due date d_i by which time the job must be completed. Let C_i represent the actual completion time of job i. The objective is to determine a sequence of jobs on the machine, and to determine a set of due dates d_i, such that each job is completed by its due date so that $C_i \leq d_i \ \forall i$ and $\sum_{i=1}^{n} d_i$ is minimized. Let Z_n^* represent the optimal objective value of an n-job instance for this model.

Note that p_i and r_i are given parameters, and thus minimizing this objective is equivalent to minimizing $\sum_{i=1}^{n}(d_i - p_i - r_i)$. This quantity $d_i - p_i - r_i$ represents, as is typical for literature in this area, the quoted lead time, equal to the upper bound on the length of the time within which the job i has to start processing after it arrives. Although these models are equivalent, we focus on the $\sum_{i=1}^{n} d_i$ objective, both for ease of exposition, and because some of our analysis would need be modified for the $\sum_{i=1}^{n}(d_i - p_i - r_i)$ objective.

We focus on *on-line* scheduling for the RDDQ model. In this context, an on-line algorithm at any time uses only information pertaining to jobs that have been released by that time. In particular, the processing time of a job is only known when it arrives, and there is no knowledge of future arrivals. This models many real world problems, where job information is not known until a job arrives, and information about future arrivals is not known until these jobs arrive. In contrast, off-line algorithms may use information about jobs that will be released in the future. It is clear that in order to find the optimal solution to this model, it may be necessary to account for future arrivals when assigning due dates, and it may be necessary to insert idle time into the schedule while waiting for certain jobs to arrive. This implies that off-line algorithms will perform better than on-line algorithms. Indeed, if all job information is known in advance, the entire processing sequence can be determined in advance and thus the due dates can be set equal to the completion times, $d_i = C_i \ \forall i$. That is, the off-line version of RDDQ is equivalent to minimizing $\sum_{i=1}^{n} C_i$, the total completion time of all jobs.

Thus, RDDQ is an NP-hard problem (Lenstra et al. 1977). However, the heuristic Shortest Processing Time Among Available Jobs (SPTA) is well known to be effective for the total completion time problem. Under the SPTA heuristic, each time a job completes processing on the machine, the shortest available job which has not yet been processed is selected for processing. This sequencing rule is effective computationally, and Kaminsky and Simchi-Levi (2001) prove the following result, where C_i^{SPTA} represents the completion time of job i when sequenced by the SPTA heuristic, and C_i^* is the completion time of job i in an optimal *offline* schedule for the completion time problem.

Theorem 2.1 *Consider a sequence of randomly generated n-job instances, where processing times are drawn from identical independent bounded distributions. Similarly, let the inter-arrival times, $r_i - r_{i-1}$, $i = 1, 2, \ldots, n$, $r_0 \equiv 0$, be drawn from identical independent bounded distributions. With probability one,*

$$\lim_{n\to\infty} \frac{\sum_{i=1}^{n} C_i^*}{n^2} = \lim_{n\to\infty} \frac{\sum_{i=1}^{n} C_i^{\mathrm{SPTA}}}{n^2}.$$

We consider a sequence of randomly generated realizations of the online RDDQ problem. That is, we have a $GI/GI/1$ system consisting of a finite number of n jobs with i.i.d. service times that are bounded by some constant $p_{\max}$, and with expected value $E(P)$, where P is the random processing time, and with bounded i.i.d. interarrival times with expected value $E(T)$, where T is the random interarrival time, and processing and interarrival times are independent. In what follows, we refer to these as the $GI/GI/1$ assumptions of the model. The objective values resulting from applying our heuristics to these instances are characterized as the size of the instances n grows to infinity.

Clearly, the optimal off-line RDDQ schedule has a sequence such that the completion time of each job is equal to $C_i^* \ \forall i$, and $d_i = C_i^* \ \forall i$. Thus, the SPTA schedule with $d_i = C_i^{\mathrm{SPTA}} \ \forall i$ is an asymptotically optimal solution for the off-line version of the RDDQ model (when instances of the model are generated as in Theorem 2.1). Let Z_n^{SPTA} be the RDDQ objective value of this schedule and Z_n^H be that of the resulting schedule by any RDDQ heuristic H. The optimal solution to the off-line version of the model is a lower bound on the solution to the on-line version, since additional information is available in the off-line case. Thus, we can conclude for a GI/GI/1 model:

Lemma 2.2 *With probability one,*

$$\lim_{n\to\infty} \frac{\sum_{i=1}^{n} Z_n^{\mathrm{SPTA}}}{n^2} = \lim_{n\to\infty} \frac{\sum_{i=1}^{n} C_i^*}{n^2} \leq \lim_{n\to\infty} \frac{Z_n^*}{n^2} \leq \lim_{n\to\infty} \frac{Z_n^H}{n^2}.$$

Intuitively, to optimally solve the RDDQ problem, sequencing and due date quotation need to be considered simultaneously. However, in this paper, we adopt an approach in which we at least conceptually solve these problems sequentially, by first designing an approach to sequencing, and then designing an approach to due date quotation which is based on our sequencing approach. More specifically, as each job arrives at the system, it is either processed immediately if no jobs are waiting, or assigned a position in the queue of jobs waiting to be processed if such a queue exists. Since each of the jobs in the queue has already arrived, their processing times are known, and can be used to sequence jobs and determine the position of the newly arrived jobs. A due date is then assigned to this new job, taking into account its position in the queue, and how that position might change later when other jobs arrive. Although we are solving this problem in effect sequentially, we demonstrate that our approach is theoretically and computationally effective. In Sect. 3, we describe these sequencing rules, in Sects. 4 and 5 we analyze these rules, and in Sect. 6 we describe how we use the results of our analysis to design a set of due date quotation approaches. We conclude with a computational study.

3 On-line sequencing rules

The First-Come-First-Serve (FCFS) sequencing rule is the most naive on-line method to achieve 100% reliable due date quotation for the RDDQ model, since jobs are processed in the order of their arrival times, and it is therefore trivial to quote a due date precisely equal to a job's completion time at its arrival time. In particular, for the ith arrival, the due date is set as follows:

$$d_i = \max_{1\le k\le i}\left\{r_k + \sum_{l=k}^{i} p_l\right\}.$$

We denote this sequencing and quotation rule FCFSQ.

As mentioned above, the sequencing rule and due date setting decision are clearly interdependent for the on-line case. Although by utilizing FCFSQ it is easy to quote accurate due dates, the FCFSQ may not lead to a desirable job sequence (recall that our goal is to minimize the average due date). Indeed, we have already observed that the SPTA rule is asymptotically optimal for minimizing completion times, and that if we could sequence jobs in this order and set due dates equal to completion times, we could effectively reduce our objective. Unfortunately, RDDQ requires that the due date be quoted upon job's arrival. The on-line nature of the problem (quoting the due date upon job's arrival without knowledge of the future) makes this impossible. Nevertheless, inspired by the SPTA rule, we propose the following sequencing procedure, which we call SP. However, unlike the SPTA rule, this procedure needs to take into account the constraint that all due dates have to be met.

SP proceeds by maintaining an ordered list, which we call the *waiting list*, of jobs that have been released and are waiting to be processed. Each time a job completes processing, the first job on the ordered list is selected for processing, and each job moves up one position in the list. If the machine is idle when a job arrives (no job is waiting), this job is processed immediately and the due date is quoted to be the completion time. If the machine is busy when a job arrives, this newly released job is first added to the end of the waiting list. Then the procedure attempts to move it up (by exchanging with a job before it) on the list ahead of the jobs that are longer than and immediately positioned prior to it, without violating due dates. It first attempts to move the new job up by one position, then by two if infeasible by one, three if infeasible by two, and so on. SP is formally presented in Algorithm 1 for inserting a new arrival j into the appropriate place in the waiting list, after we define some notation:

- $list_j$ = the set of the jobs in the waiting list when job j arrives at time r_j but has not joined the queue.
- $|list_j|$ = the number of jobs in $list_j$.
- $list[k]$ = the job at the kth position in the waiting list during the sequencing of the new arrival. Note that $list[1]$ will be the job that is processed next when the machine is available.
- $pos[j]$ = the position of job j in the waiting list during the sequencing.

Algorithm 1 SP

for $j = 1, 2, \ldots n$ **do**
  Put j in the last position of $list_j$, $pos[j] = |list_j| + 1$.
  $u \leftarrow pos[j]$
  $v \leftarrow u - 1$
  while $(p_{list[u]} < p_{list[v]})$ and $(u > 0)$ and $(v > 0)$ **do**
    Exchange the positions of the two jobs $list[v]$ and $list[u]$.
    Determine the completion time of the job currently positioned u, $CT_{list[u]}$.
    (If C_i is the completion time of job i currently in process, then $CT_{list[u]} = C_i + \sum_{k=1}^{u} p_{list[k]}$.)
    if $CT_{list[u]} \le d_{list[u]}$ **then**
      $u \leftarrow v$
      $v \leftarrow u - 1$
    else
      Exchange the jobs $list[v]$ and $list[u]$ back.
      $v \leftarrow v - 1$
    end if
  end while
  $pos[j] \leftarrow u$
end for

SP starts operating whenever a new job j arrives, and initially it puts j at the end of the list. Let job k be the job with the greatest position in the list such that $p_k < p_j$ if there is one; otherwise, let $k = 0$. SP attempts to exchange the positions of job j and the job before it until either (1) job j is being sequenced right after job k, or (2) none of the jobs between j and k can be exchanged with j without violating its due date.

The algorithm outputs an updated waiting list while job j is inserted into the final position. During the sequencing, the waiting list keeps being updated until the final $pos[j]$ is determined.

Although SP attempts to sequence shorter jobs before longer ones to achieve the SPTA sequence, it may not always be able to do so, because previously assigned due dates may prevent this from happening, unless due dates are set very far (especially for longer jobs) in the future. However, due dates set far in the future are clearly bad for the RDDQ objective, minimizing the sum of due dates. Thus, an effective due date quotation rule *for this sequencing approach* needs to in effect estimate the impact of the use of SP on future arrivals, and set due dates appropriately.

Indeed, this suggests that when the final position of a new job in the waiting list is determined under SP, we might want to assign this job a due date later than the time it will complete if the jobs currently ahead of it in the list are processed and then it is processed. The difference between the due date and this currently expected completion time is called "slack" which is used to account for jobs whose arrival times are later than the currently arriving job, but would ideally be processed before the currently arriving job is processed. An effective slack estimation procedure, along with SP, should be able to provide a resulting schedule that has a sequence close (or identical) to the SPTA sequence and assign due dates so that most or all of the due dates are near (or equal to) completion times. In the subsequent sections of this paper, we characterize on-line sequencing rules FCFS, SP, and SPTA, for both $E(P) < E(T)$ and $E(P) > E(T)$ instances, and then build on these insights to develop simple yet effective slack estimation rules. We note that the analysis and effectiveness of our rules for cases where $E(P) = E(T)$ remains an open question.

Observe that FCFS, SP, and SPTA are non-idling (or work-conserving) schedules. In a work-conserving schedule, the processor is never kept idle when there is a job waiting to be processed. Also, observe that SP can provide the FCFS or SPTA sequence, depending on the slack assigned to each job. When each job is assigned a zero slack, the FCFS sequence is generated, and when all jobs are assigned a large slack (e.g., $np_{\max}$), the SPTA sequence is generated.

In addition, clearly any schedule is made up of *chains*, or consecutively scheduled jobs without idle time. That is, each chain represents a busy period, so the number of jobs in a chain is the number of jobs in a busy period. The following observation is self evident for work-conserving schedules:

Observation 3.1 *Consider any instance of the RDDQ model and consider any two work-conserving schedules on the processor. Number the chains consecutively in both of these schedules. There will be the same number of chains in both sequences, and the kth chain in either sequence will contain exactly the same jobs, and will begin and end processing at exactly the same time on the processor.*

Throughout the operation of SP, until the final position of the new job j is determined, jobs are continually being resequenced, and so the anticipated completion time of job j is continually changing. At any point in the operation of the algorithm, let $R\bar{S}L_i$ be the current remaining slack for job i ($i \neq j$) in the waiting list, which is defined to be the difference between the due date assigned to job i and its completion time based on its current position in the list. Clearly, when the new job j arrives and is being sequenced, $R\bar{S}L_i$ can potentially change with each iteration of the algorithm. The following observation, which follows from the specific steps of algorithm SP, will be useful for the analysis in subsequent sections:

Observation 3.2 *When job i is currently positioned to be completed at CT_i, $R\bar{S}L_i$ is equal to $d_i - CT_i$. Suppose, at some point during the operation of SP, that the new arrival j is at some position x after job k ($p_k > p_j$), and $R\bar{S}L_k$ is not large enough to exchange j with k to account for j and the jobs positioned after k and before j to be processed prior to k. Then, SP finds the next job l at some position y (i.e., $l = list[y]$) ahead of job k ($p_l > p_j$), with sufficient $R\bar{S}L_l$ to be exchanged with j. When made, this exchange increases the current remaining slack of the jobs between positions x and y (including $R\bar{S}L_k$) by $p_l - p_j$ (because these jobs have a shorter job j before it instead of l, which makes them able to be completed earlier). This exchange also decreases $R\bar{S}L_l$ by the sum of processing times of job j and the jobs between positions x and y.*

4 Characterizing on-line sequencing for $E(P) < E(T)$ cases

In this section, we characterize the performance of on-line sequencing rules for $E(P) < E(T)$ case, and present an asymptotically optimal on-line sequencing rule for this case.

First, the following was proved by Gazmuri (1985):

Lemma 4.1 *Under the $GI/GI/1$ assumptions of the RDDQ model with $E(P) < E(T)$, let M be a random variable representing the number of jobs in a chain of a work-conserving schedule. Then $E(M)$ and $E(M^2)$ are bounded by constants that are independent of n.*

We combine this result with Observation 3.1, to conclude:

Lemma 4.2 *Under our $GI/GI/1$ assumptions, almost surely*:

$$\lim_{n\to\infty} \frac{\sum_{i=1}^{n} C_i^{\mathrm{SPTA}}}{n^2} = \lim_{n\to\infty} \frac{\sum_{i=1}^{n} C_i^{H}}{n^2},$$

where C_i^H stands for the completion time of job i under any heuristic that generates a work-conserving schedule, such as SP or FCFSQ.

Proof Recall Observation 3.1 and let $N(n)$, M_k, and l_k be the number of chains if there are n jobs, the number of jobs in chain k, and the job index in chain k, respectively. Jobs are indexed $1, 2, \ldots, M_k$ in chain k. Clearly,

$$\sum_{l_k=1}^{M_k} C_{l_k}^{\mathrm{SPTA}} - \sum_{l_k=1}^{M_k} C_{l_k}^{H} \le M_k(M_k p_{\max}).$$

Summing over all chains from $k = 1$ to $k = N(n)$ gives

$$0 \le \left| \sum_{i=1}^{n} C_i^{\mathrm{SPTA}} - \sum_{i=1}^{n} C_n^{H} \right| \le p_{\max} \sum_{k=1}^{N(n)} \sum_{l_k=1}^{M_k} M_k^2.$$

Dividing by n^2 and taking the limit, we get that

$$0 \le \lim_{n\to\infty} \left| \frac{\sum_{i=1}^{n} C_i^{\mathrm{SPTA}}}{n^2} - \lim_{n\to\infty} \frac{\sum_{i=1}^{n} C_i^{H}}{n^2} \right|$$

$$\le p_{\max} \lim_{n\to\infty} \frac{N(n)}{n^2} \frac{\sum_{k=1}^{N(n)} M_k^2}{N(n)} = 0.$$

Combining Lemma 4.1 with the following:

$$\lim_{n\to\infty} \frac{n}{N(n)} = E(M) \quad \text{a.s.},$$

implies that $N(n)$ grows $O(n)$, so that

$$\lim_{n\to\infty} \frac{\sum_{k=1}^{N(n)} M_k^2}{N(n)} = E\left(M^2\right) \quad \text{a.s.}$$

Therefore, the following holds almost surely:

$$\lim_{n\to\infty} \frac{\sum_{i=1}^{n} C_i^{\mathrm{SPTA}}}{n^2} = \lim_{n\to\infty} \frac{\sum_{i=1}^{n} C_i^{H}}{n^2}.$$

This completes the proof. □

Although Lemma 4.2 implies that the total completion times of all work-conserving schedules are asymptotically equal for $E(P) < E(T)$ instances (asymptotically equal to Z_n^{SPTA} or Z_n^*), recall that the RDDQ objective is $Z_n^H = \sum_{i=1}^{n} C_i^H + \sum_{i=1}^{n} (d_i - C_i^H)$, where the term of $\sum_{i=1}^{n} (d_i - C_i^H)$ may not be always negligible for any on-line H. An intuitive on-line rule to have this term vanish is FCFSQ, or SP when all jobs have a zero slack. *Thus, we conclude that FCFSQ, or equivalently SP with zero slack for all jobs, generates an asymptotically optimal RDDQ schedule when $E(P) < E(T)$.*

5 Characterizing on-line sequencing for $E(P) > E(T)$ cases

In this section, we characterize the asymptotic performance of on-line sequencing rules for $E(P) > E(T)$ case, which is significantly more complex than the case presented above. We use this characterization in the next section to develop effective slack quotation rules for this model.

We first present our results for a specific set of discrete known processing time distributions, and then indicate how the results can be generalized. Suppose that we have three types of jobs A, B, and C, with processing times p_A, p_B, and p_C, respectively, where $p_A < p_B < p_C$. Also for a given n-job instance I_0, there are n_A A-jobs, n_B B-jobs, and n_C C-jobs, where the fraction of each type is $\alpha_i = n_i/n$, $i \in \{A, B, C\}$. Let $\mathbf{I}_A(P)$ be an indicator variable which is 1 if a job is of Type A (i.e., $P = p_A$) and 0 otherwise. Similarly, $\mathbf{I}_B(P)$ and $\mathbf{I}_C(P)$ are indicators for Type B and Type C.

We define the offered load for job type A, $\rho_A = E(\mathbf{I}_A(P)P)/E(T)$, and the offered load for job types A and B, $\rho_{A+B} = E(\mathbf{I}_A(P)P + \mathbf{I}_B(P)P)/E(T)$, and further restrict the distributions of processing time and inter-arrival time as follows:

$$\begin{aligned} \rho_A &= \frac{E(\mathbf{I}_A(P)P)}{E(T)} = 1 - \sigma < 1, \\ \rho_{A+B} &= \frac{E(\mathbf{I}_A(P)P + \mathbf{I}_B(P)P)}{E(T)} = 1 + \sigma > 1, \end{aligned} \tag{1}$$

where σ is some positive number less than 1. Equation (1) implies $E(P)/E(T) > 1$ since $E(P) = E(\mathbf{I}_A(P)P + \mathbf{I}_B(P)P + \mathbf{I}_C(P)P)$. We then create two new instances I_1 and I_2 based on I_0. I_1 is identical to I_0 except B- and C-jobs are discarded upon arrival (equivalently, replaced by zero processing times). I_2 is identical to I_0 except C-jobs are discarded when arriving. So, there are n_A A-jobs in I_1 and $n_A + n_B$ A- and B-jobs in I_2. Then SP, FCFS, and SPTA are applied to these three instances, I_0, I_1, and I_2. When SP is used, we assign a zero slack to A-jobs and B-jobs, but a "very large" slack (e.g., $np_{\max}$) to C-jobs.

We denote by ${}_jC^H_{[i_j]}(I_k)$ the completion time of the i_jth completed j-type ($j \in \{A, B, C\}$) job in the resulting schedule of I_k ($k = 0, 1, 2$) with sequencing rule H. Note that the possible values of j depend on the instance; for example, j only can be job type 'A' when $k = 1$. $C^H_{[i]}(I_k)$ has the same meaning except that the job type is not specified, and $r_i(I_k)$ is the arrival time of the ith job arrival in I_k. Using results from queuing theory, the chain structure of the resulting I_0, I_1, and I_2 sequences can be characterized as follows as the instance size n approaches infinity.

5.1 The sequence generated under SP

When offered load is larger than 1, it is well known that the length of the final chain of the queue grows to infinity, so that the jobs in chains before the final chain do not (asymptotically) contribute to the total completion time. Equation (1) implies that the offered load is less than 1 for I_1, but larger than 1 for I_2 and I_0. Thus, Lemma 4.2 applies to the resulting I_1 sequence (Fig. 1).

However, in the I_2 sequence, the length of the final chain almost surely grows to infinity, so the asymptotic total completion time will be entirely determined by the final chain. That is, the last chain consists of $\Theta(n_A + n_B)$ jobs (see Fig. 2). This implies that the last chain will dominate the total completion time of the I_2 sequence.

Now we apply SP (with zero slacks for A- and B-jobs) to instance I_2. Since they have zero slacks, A- and B-jobs follow the FCFS sequence. Let $TC^{\mathrm{SP}}_{\mathrm{l.c.}}(I_2)$ be the sum of completion times of the jobs in the last chain of the resulting I_2 sequence (consisting of A- and B-jobs). Then, almost surely,

$$\lim_{n\to\infty} \frac{\sum_{i=1}^{n_A+n_B} C^{\mathrm{SP}}_{[i]}(I_2)}{n^2} = \lim_{n\to\infty} \frac{TC^{\mathrm{SP}}_{\mathrm{l.c.}}(I_2)}{n^2}. \tag{2}$$

Compare the resulting I_1 and I_2 sequences generated under SP. We construct two extreme cases for the I_2 sequence from the I_1 sequence. In the first case, each A-job of the I_1 sequence is postponed by n_B B-jobs. In the second case, all B-jobs arrive at or after the end of the last chain of the I_1 sequence. These two cases provide the upper bounds for the completion time of each A-job and each B-job in the I_2 sequence, respectively. That is,

$${}_AC^{\mathrm{SP}}_{[i_A]}(I_2) \le n_B p_B + {}_A C^{\mathrm{SP}}_{[i_A]}(I_1),$$

$${}_BC^{\mathrm{SP}}_{[i_B]}(I_2) \le n_B p_B + \max\bigl\{r_{n_A+n_B}(I_2),\ {}_AC^{\mathrm{SP}}_{n_A}(I_1)\bigr\}.$$

Therefore,

$$\begin{aligned}
&\left|\sum_{i=1}^{n_A+n_B} C^{\mathrm{SP}}_{[i]}(I_2) - \sum_{i=1}^{n_A} C^{\mathrm{SP}}_{[i]}(I_1)\right| \\
&= \left|\sum_{i=1}^{n_A} {}_AC^{\mathrm{SP}}_{[i_A]}(I_2) - \sum_{i=1}^{n_A} {}_AC^{\mathrm{SP}}_{[i_A]}(I_1) + \sum_{i=1}^{n_B} {}_BC^{\mathrm{SP}}_{[i_B]}(I_2)\right| \\
&\le n_A n_B p_B + n_B^2 p_B \\
&\quad + n_B \max\bigl\{r_{n_A+n_B}(I_2),\ {}_AC^{\mathrm{SP}}_{n_A}(I_1)\bigr\},
\end{aligned} \tag{3}$$

where $\max\{r_{n_A+n_B}(I_2),\ {}_AC^{\mathrm{SP}}_{n_A}(I_1)\}$ is $\Theta(n)$ under the $GI/GI/1$ assumptions of the model.

Now, compare the resulting I_2 and I_0 sequences generated under SP. The I_0 sequence can be constructed by sequentially adding C-job arrivals at their arrival times, without altering the processing order of A- and B-jobs, to the I_2 sequence. The procedure above may increase the completion times of A- and B-jobs in the I_2 sequence accordingly (see Fig. 3), and at the arrival time of a C-job, if the machine is busy, this C-job will be added to the end of the queue. Note that for I_0, the A- or B-jobs that arrive after a C-job can be positioned prior to it in the queue under SP due to the large slack of the C-job. Within one chain of the I_2 sequence, when one A- or B-job is completed, there is always another new A- or B-job available. So, any C-job that has arrival time within the processing of this chain will not be processed before the chain is finished (e.g., the second C-job in Fig. 3).

An AB-group is defined to be a set of A- or B-jobs, or both, consecutively processed without being interrupted by idle times or other types of jobs. So, an AB-group is actually a chain for the I_2 sequence. The construction of the I_0 sequence from the I_2 sequence above can result in some chains of the I_2 sequence being combined to be one AB-group of the I_0 sequence. For example, as shown in Fig. 3, the added C-jobs may cause some A- and B-jobs to be completed later and thus take up the idle time before next chain in the I_2 sequence. Thus, the last AB-group in the I_0 sequence includes all A- and B-jobs in the last chain of the I_2 sequence, and possibly other A- and B-jobs before this last chain.

Recall that the last I_2 chain dominates the sum of completion times of the I_2 schedule. The number of C-jobs that arrive before the last I_2 chain starts must be added to the I_2 sequence to construct the I_0 sequence, but will not impact the asymptotic total completion time of the I_0 sequence.

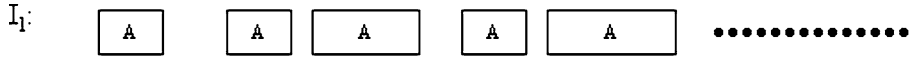

Fig. 1 The chain structure of the resulting I_1 sequence

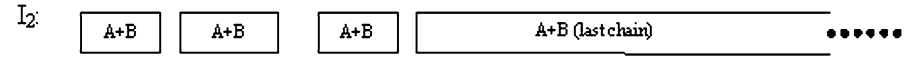

Fig. 2 Chain structure vs. job type for the resulting I_2 sequence

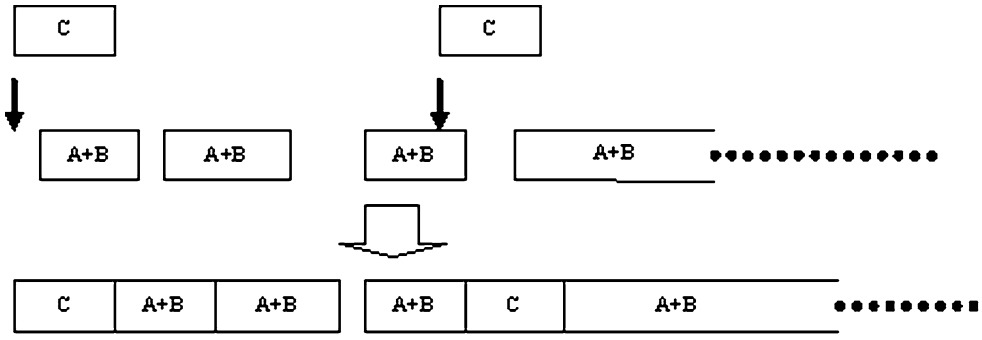

Fig. 3 Constructing the I_0 sequence by adding C-jobs to the I_2 sequence

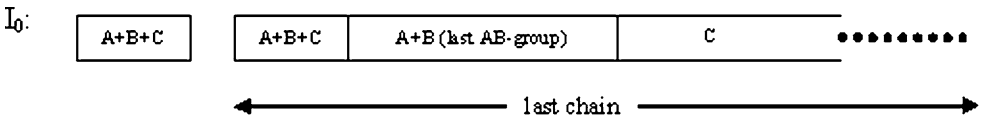

Fig. 4 The distribution of A-, B-, and C- jobs over the chains of the resulting I_0 sequence

Hence, C-jobs of order $\Theta(n_C)$ will be processed consecutively after the last AB-group in the I_0 sequence, and will not be interrupted by idle times. These C-jobs and the last AB-group are included in the last I_0 chain. Note that at the time the machine is busy in the I_2 schedule, the machine must be busy in the I_0 schedule. This implies that the last chain of the I_0 sequence will not start later than the last chain of the I_2 sequence.

Since the last I_0 chain includes the last AB-group (equivalently, the last chain) of the I_2 sequence, the number of jobs in the last AB-group of the I_2 sequence dominates the number of A- and B-jobs in the last I_0 chain. Figure 4 presents a graphical example of the I_0 schedule. Note that in any AB-group of the I_0 sequence, A- and B-jobs are sequenced according to their arrival order (FCFS).

In other words, the final AB-group and the C-jobs scheduled after it in the last I_0 chain almost surely dominate the sum of completion times of the I_0 sequence. Also, the total completion time of the jobs in the last I_2 chain and that of the last AB-group in the last I_0 chain approach each other as n approaches infinity. Let $LC^{\mathrm{SP}}_{\mathrm{l.AB.}}(I_0)$ be the time when the last job of the last AB-group is completed in the I_0 sequence (see Fig. 4), and $TC^{\mathrm{SP}}_{\mathrm{l.AB.}}(I_0)$ be the sum of completion times of the A- and B- jobs in this AB-group. We quantify our observations as follows:

$$\lim_{n\to\infty}\frac{\sum_{i=1}^{n}C^{\mathrm{SP}}_{[i]}(I_0)}{n^2}=\lim_{n\to\infty}\frac{TC^{\mathrm{SP}}_{\mathrm{l.AB.}}(I_0)+n_CLC^{\mathrm{SP}}_{\mathrm{l.AB.}}(I_0)+p_Cn_C(n_C+1)/2}{n^2}, \tag{4}$$

and

$$\lim_{n\to\infty}\frac{TC^{\mathrm{SP}}_{\mathrm{l.AB.}}(I_0)}{n^2}=\lim_{n\to\infty}\frac{TC^{\mathrm{SP}}_{\mathrm{l.c.}}(I_2)}{n^2}. \tag{5}$$

Combining (2), (4), and (5), and using the fact that all terms are positive, we get that almost surely:

$$\lim_{n\to\infty}\frac{\sum_{i=1}^{n}C^{\mathrm{SP}}_{[i]}(I_0)}{n^2}=\lim_{n\to\infty}\frac{\sum_{i=1}^{n_A+n_B}C^{\mathrm{SP}}_{[i]}(I_2)+n_CLC^{\mathrm{SP}}_{\mathrm{l.AB.}}(I_0)+p_Cn_C(n_C+1)/2}{n^2}. \tag{6}$$

5.2 The sequences generated under FCFS and SPTA

FCFS, SPTA, and SP all generate a work-conserving schedule, and thus Observation 3.1 applies to their resulting schedules for I_1, I_2, and I_0. When FCFS is applied, the resulting I_1 and I_2 sequences (Figs. 1 and 2) will be the same as when SP is applied. However, recall that for I_0 the SP sequence has some jobs (the C-jobs) with a large slack value, and thus the FCFS sequence (equivalently, letting all slacks be zero and using SP), although it has the same chain structure as the SP sequence, may differ from the SP sequence.

When SPTA is applied, only the resulting I_1 sequence will be the same as when SP (or FCFS) is applied. Moreover, the last chain will once again dominate the total completion time of the I_2 sequence. Let $TC^{\mathrm{SPTA}}_{\mathrm{l.c.}}(I_2)$ be the sum of com-

pletion times of the jobs in the last chain of the resulting I_2 sequence (consisting of A- and B-jobs). Then,

$$\lim_{n\to\infty}\frac{\sum_{i=1}^{n_A+n_B} C_{[i]}^{\mathrm{SPTA}}(I_2)}{n^2}=\lim_{n\to\infty}\frac{TC_{\mathrm{l.c.}}^{\mathrm{SPTA}}(I_2)}{n^2}. \tag{7}$$

Also for each chain, in the resulting I_2 schedule A-jobs are more likely sequenced before B-jobs under SPTA. However, to compare the I_1 and I_2 sequences under SPTA, we can employ the approach used to show (3) to conclude:

$$\begin{aligned}&\left|\sum_{i=1}^{n_A+n_B} C_{[i]}^{\mathrm{SPTA}}(I_2)-\sum_{i=1}^{n_A} C_{[i]}^{\mathrm{SPTA}}(I_1)\right|\\&\quad\le n_An_Bp_B+n_B^2p_B\\&\qquad+n_B\max\left\{r_{n_A+n_B}(I_2),{}_AC_{n_A}^{\mathrm{SPTA}}(I_1)\right\}.\end{aligned} \tag{8}$$

We use the same method of constructing the I_0 sequence under SP from the I_2 sequence under SP (Sect. 5.1) to construct the I_0 sequence under SPTA from the I_2 sequence under SPTA. We conclude that for the I_0 sequence under SPTA, the number of the C-jobs that will be processed consecutively immediately after the last AB-group is $\Theta(n_3)$ and these C-jobs are processed without being interrupted by idle times. As we observed before, this last AB-group has $\Theta(n_A+n_B)$ A- and B- jobs. Let $LC_{\mathrm{l.AB.}}^{\mathrm{SPTA}}(I_0)$ be the time when the last job of the last AB-group is completed in the I_0 sequence under SPTA, and $TC_{\mathrm{l.AB.}}^{\mathrm{SPTA}}(I_0)$ be the sum of completion times of the jobs in this AB-group. Apply the same reasoning used to obtain (4) and (5) and we have

$$\lim_{n\to\infty}\frac{\sum_{i=1}^{n} C_{[i]}^{\mathrm{SPTA}}(I_0)}{n^2}=\lim_{n\to\infty}\frac{TC_{\mathrm{l.AB.}}^{\mathrm{SPTA}}(I_0)+n_CLC_{\mathrm{l.AB.}}^{\mathrm{SPTA}}(I_0)+p_Cn_C(n_C+1)/2}{n^2}, \tag{9}$$

$$\lim_{n\to\infty}\frac{TC_{\mathrm{l.AB.}}^{\mathrm{SPTA}}(I_0)}{n^2}=\lim_{n\to\infty}\frac{TC_{\mathrm{l.c.}}^{\mathrm{SPTA}}(I_2)}{n^2}. \tag{10}$$

We combine (7), (9), and (10) to get that almost surely,

$$\lim_{n\to\infty}\frac{\sum_{i=1}^{n} C_{[i]}^{\mathrm{SPTA}}(I_0)}{n^2}=\lim_{n\to\infty}\frac{\sum_{i=1}^{n_A+n_B} C_{[i]}^{\mathrm{SPTA}}(I_2)+n_CLC_{\mathrm{l.AB.}}^{\mathrm{SPTA}}(I_0)+p_Cn_C(n_C+1)/2}{n^2}. \tag{11}$$

5.3 Analytical results and implications

Combining (6) and (11) gives the following:

$$\begin{aligned}&\lim_{n\to\infty}\frac{|\sum_{i=1}^{n} C_{[i]}^{\mathrm{SPTA}}(I_0)-\sum_{i=1}^{n} C_{[i]}^{\mathrm{SP}}(I_0)|}{n^2}\\&\le\lim_{n\to\infty}\frac{|\sum_{i=1}^{n_A+n_B} C_{[i]}^{\mathrm{SPTA}}(I_2)-\sum_{i=1}^{n_A+n_B} C_{[i]}^{\mathrm{SP}}(I_2)|}{n^2}\\&\qquad+\alpha_3\lim_{n\to\infty}\frac{|LC_{\mathrm{l.AB.}}^{\mathrm{SPTA}}(I_0)-LC_{\mathrm{l.AB.}}^{\mathrm{SP}}(I_0)|}{n}.\end{aligned} \tag{12}$$

Firstly, the I_0 sequence under SPTA has the same chain structure as the I_0 sequence under SP (Observation 3.1). In these two sequences, the last chains start at the same time. Note that both SP and SPTA have A- and B-jobs able to be selected prior to C-jobs in the queue. Thus, in the last chains of the two sequences above, the last AB-groups end at the same time. That is,

$$LC_{\mathrm{l.AB.}}^{\mathrm{SPTA}}(I_0)=LC_{\mathrm{l.AB.}}^{\mathrm{SP}}(I_0). \tag{13}$$

Secondly, it is obvious that

$${}_AC_{[n_A]}^{\mathrm{SP}}(I_1)={}_A\,C_{[n_A]}^{\mathrm{SPTA}}(I_1), \tag{14}$$

$$\sum_{i=1}^{n_A} C_{[i]}^{\mathrm{SP}}(I_1)=\sum_{i=1}^{n_A} C_{[i]}^{\mathrm{SPTA}}(I_1). \tag{15}$$

In addition, using (3), (8), and (15), it follows that:

$$\begin{aligned}&\lim_{n\to\infty}\frac{|\sum_{i=1}^{n_A+n_B} C_{[i]}^{\mathrm{SPTA}}(I_2)-\sum_{i=1}^{n_A+n_B} C_{[i]}^{\mathrm{SP}}(I_2)|}{n^2}\\&\le\lim_{n\to\infty}\frac{|\sum_{i=1}^{n_A+n_B} C_{[i]}^{\mathrm{SPTA}}(I_2)-\sum_{i=1}^{n_A} C_{[i]}^{\mathrm{SPTA}}(I_1)|}{n^2}\\&\qquad+\lim_{n\to\infty}\frac{|\sum_{i=1}^{n_A} C_{[i]}^{\mathrm{SPTA}}(I_1)-\sum_{i=1}^{n_A} C_{[i]}^{\mathrm{SP}}(I_1)|}{n^2}\\&\qquad+\lim_{n\to\infty}\frac{|\sum_{i=1}^{n_1} C_{[i]}^{\mathrm{SP}}(I_1)-\sum_{i=1}^{n_A+n_B} C_{[i]}^{\mathrm{SP}}(I_2)|}{n^2}\\&=2\alpha_A\alpha_Bp_B+2\alpha_B^2p_B\\&\qquad+2\alpha_B\lim_{n\to\infty}\frac{\max\{r_{n_A}(I_1),{}_AC_{[n_A]}^{\mathrm{SP}}(I_1)\}}{n}.\end{aligned}$$

Combining the above, (12), (13), (14), and (15), we conclude

$$\lim_{n\to\infty} \frac{|\sum_{i=1}^{n} C_{[i]}^{\mathrm{SPTA}}(I_0) - \sum_{i=1}^{n} C_{[i]}^{\mathrm{SP}}(I_0)|}{n^2}$$
$$\leq 2\alpha_A\alpha_B p_B + 2\alpha_B^2 p_B + 2\alpha_B \lim_{n\to\infty} \frac{\max\{r_{n_A}(I_1),\ {}_AC_{[n_A]}^{\mathrm{SP}}(I_1)\}}{n}. \quad (16)$$

Recall the $GI/GI/1$ assumptions of the model. These imply that $\max\{r_{n_A}(I_1),\ {}_AC_{[n_A]}^{\mathrm{SP}}(I_1)\}$ is $O(n_A)$ (or $O(n)$). Thus, we can observe in (16) that for an instance of RDDQ, controlling the number of B-jobs (or α_B) is the key to bounding the difference between the total completion time of the SP sequence and that of the SPTA sequence. Note that, when the total completion time is obtained by using SP, we have assumed that A- and B-jobs have zero slack and C-jobs have a very large slack.

Thus far the results for $E(P) > E(T)$ cases are based on the case that processing time is discrete and its distribution is known. Now, we return to the case where processing time is continuous and drawn from some known and bounded distribution. We can choose a set of p_A, p_B, and p_C such that $p_{\min} < p_A < p_B < p_C = p_{\max}$, where each job with processing time P is set to be of Type A if $p_{\min} \leq P < p_A$ ($\mathbf{I}_A(P) = 1$), Type B if $p_A \leq P < p_B$ ($\mathbf{I}_B(P) = 1$), or Type C if $p_B \leq P \leq p_C$ ($\mathbf{I}_C(P) = 1$). For some job i with processing time p_i, let $\mathbf{I_i}(P)$ be the indicator variable which is 1 if $P < p_i$ or 0 otherwise, and let ρ_i be the offered load of a system made up of the jobs from our model with processing time less than that of p_i, so that $\rho_i = E(\mathbf{I_i}(P)P)/E(T)$. Under the condition specified by (1), when job i is of Type A, we have $p_i < p_A$ and the following:

$$\mathbf{I_i}(P) < \mathbf{I_A}(P) \quad \text{if } p_i \leq P < p_A,$$
$$\mathbf{I_i}(P) = \mathbf{I_A}(P) \quad \text{otherwise,}$$

which implies $\rho_i < \rho_A < 1$. Similarly, $\rho_i \geq \rho_{A+B} > 1$ if job i is of Type C. However, when job i is of Type B, $\rho_i \geq \rho_A$ and $\rho_i < \rho_{A+B}$ imply that ρ_i can be larger than, smaller than, or equal to 1. Thus, by choosing p_A, p_B, and p_C, we can have $\rho_i < 1 - \sigma$ if job i is of Type A, $\rho_i \geq 1 + \sigma$ if job i is of Type C, and $1 - \sigma \leq \rho_i < 1 + \sigma$ if job i is of Type B, where we can let p_A and p_B be arbitrarily close such that σ will be arbitrarily small.

Also, almost surely, α_A, α_B, and α_C approach $Pr(\mathbf{I}_A(P) = 1)$, $Pr(\mathbf{I}_B(P) = 1)$, and $Pr(\mathbf{I}_C(P) = 1)$, respectively, as n approaches infinity. Again, when p_A and p_B are arbitrarily close, α_B (or $Pr(\mathbf{I}_B(P) = 1)$) can be made arbitrarily small.

Given an n-job instance, we can employ the same procedure as in the discrete case, assigning zero slack to A- and B-jobs and a very large slack to C-jobs. For this continuous processing time case, the significant majority of A- and B-jobs will still be sequenced prior to C-jobs in the queue as in the discrete processing time case. Then, (12) holds following the reasoning behind (2) through (11), although the term $p_C n_C(n_C + 1)/2$ in (4), (6), (9), and (11) must be replaced by another expression for the total completion time of $\Theta(n_C)$ C-jobs sequenced after the last AB-group (because C-jobs now have a range of different processing times instead of constant p_C). Equations (13) and (14) also hold. Although (15) will not be true, we know almost surely

$$\lim_{n\to\infty} \frac{|\sum_{i=1}^{n_A} C_{[i]}^{\mathrm{SPTA}}(I_1) - \sum_{i=1}^{n_A} C_{[i]}^{\mathrm{SP}}(I_1)|}{n^2} = 0, \quad (17)$$

according to Lemma 4.2. We can let p_A and p_B be arbitrarily close such that α_B is arbitrarily small. Then, using (12), (13), (14), (17), and (16), for randomly generated $E(P) > E(T)$ instances from some continuous, bounded distribution, almost surely

$$\lim_{n\to\infty} \frac{|\sum_{i=1}^{n} C_{[i]}^{\mathrm{SPTA}}(I_0) - \sum_{i=1}^{n} C_{[i]}^{\mathrm{SP}}(I_0)|}{n^2} = \epsilon, \quad (18)$$

where ϵ is an arbitrarily small positive number.

Thus, we have shown that our sequencing procedure SP can lead to a sequence arbitrarily close to the SPTA sequence which is asymptotically optimal for total completion time. Completion time is not our objective, however, and in the next section we consider how to appropriately assign slack for our objective.

6 Slack assignment rules

Recall that (18) is based on the assumption that A- and B-jobs are assigned zero slacks and C-jobs are assigned a very large slack. The question is, what is the effective "very large" slack that will not adversely affect the objective and will still lead to the same result? In the previous discussion (the $E(P) > E(T)$ case) for discrete processing times, for a C-job that is sequenced after the last AB-group in the I_0 sequence under SP (see Fig. 4), the A- and B-jobs that arrive after it turn out to be processed before it. Thus, this C-job can be assigned a slack which is the waiting time due to the sum of processing times of the A- and B-jobs that will arrive after it. Let P_i be the random variable P for job i. This implies that if job i is of Type C, $\sum_{k=i+1}^{n}(\mathbf{I}_A(P_k)P_k + \mathbf{I}_B(P_k)P_k)$ may be an effective slack amount for most C-jobs.

Now, consider the continuous processing time case. Recall that if the distribution is known, in order to apply (18), we can choose a set of p_A, p_B, and p_C to define a job to be of Type A, B, or C, and that A- and B-jobs are assigned zero slacks and C-jobs are assigned a very large slack. However, the proper (instead of "very large") slack for a C-job (more precisely, for C-jobs completed after the last AB-group of

the last chain) when using SP, can be the sum of processing times of the jobs (which can be of Type A, B, or C) that are shorter than this C-job and will arrive after it. In other words, we can modify the previous approach as follows. If job i is of Type C, $\sum_{k=i+1}^{n}(\mathbf{I_i}(P_k)P_k)$ would be a reasonable slack for it if it were known, and $\rho_i > 1$ must hold. If job i is of Type A, it is assigned a zero slack and $\rho_i < 1$ must hold. However, if job i is of Type B, either zero or $\sum_{k=i+1}^{n}(\mathbf{I_i}(P_k)P_k)$ can be a slack for it, because the slack for B-jobs may not significantly impact the objective when the number of B-jobs is made small by choosing p_A to be very close to p_B. Note that a B-job i can have $\rho_i < 1$ or $\rho_i > 1$.

The above suggests that job i be assigned a slack $\sum_{k=i+1}^{n}(\mathbf{I_i}(P_k)P_k)$ if $\rho_i > 1$, or 0 otherwise. Denote by SL_i the slack assigned to job i, and recall that the offered load ρ_i may be expressed as:

$$\rho_i = \frac{E(\mathbf{I_i}(P)P)}{E(T)} = \frac{E[\mathbf{I_i}(P)P|\mathbf{I_i}(P)=1]Pr(P<p_i)}{E(T)} = \frac{E[P|P<p_i]Pr(P<p_i)}{E(T)}. \tag{19}$$

Clearly, the value of SL_i depends on ρ_i. In an on-line environment, SL_i and ρ_i may need to be estimated based on the information of the events that have occurred and the limited information of the future (if available).

We introduce two scenarios with different amounts of available information in Sects. 6.1 and 6.2. Recall that, when a new job j is sequenced under SP, the time it is expected to be completed based on its current position on the waiting list, CT_j, is first determined as follows. Let $\tilde{t}_j$ be the remaining processing time for the job currently in process at time r_j. Then,

$$CT_j = r_j + \tilde{t}_j + \sum_{k=1}^{pos[j]} p_{list[k]}.$$

After SL_j is determined, we have $d_j = CT_j + SL_j$.

6.1 Instance size is unknown

Suppose that we do not have any information pertaining to the future at any time. We can estimate ρ_i for job i, but it is unlikely that we will be able to estimate $\sum_{k=i+1}^{n}(\mathbf{I_i}(P_k)P_k)$. Thus, a different approach is needed. Note that when using the SPTA rule, longer jobs are likely to wait in the queue longer than shorter jobs, because longer jobs will be more likely to have more later arriving jobs sequenced before them. This suggests that the total processing time of the jobs that arrive after job j and will be sequenced before it could be positively correlated with the processing time p_j. Letting $SL_j = Kp_j$ is a simple approach that does this. However, in order to avoid the use of external parameter(s) (e.g., the K above), we attempt to utilize information that is available in the shop, and that represents this expected positive relationship between slack and job processing time. The slack assignment rule that follows is based on this observation.

Let $n_j(p_j)$ be the total number of jobs that have arrival times less than r_j with processing time less than p_j, and let $T_j(p_j)$ be the total processing time of these jobs. Let $N_{pos[j]}$ be the number of jobs in the waiting list, immediately after job j is inserted into that list, with processing times less than p_j and positions in the list between position 1 and position $pos[j]$. Define $\hat{P}(p_j)$ to be the estimate of $E(P|P<p_j)$. That is,

$$\hat{P}(p_j) = \frac{T_j(p_j)}{\max\{n_j(p_j), 1\}}.$$

Then, we calculate the slack to be added to job j's due date, SL_j, as follows:

$$SL_j = N_{pos[j]}\hat{P}(p_j).$$

The goal of this rule is to dynamically adapt to the condition of the job, and estimate the total processing time of the jobs that are expected to be sequenced before j in the queue if the SPTA sequence can be achieved.

The slack assignment rule for this first scenario is denoted by "Slack I"(SI). Here we call the heuristic that uses SP along with SI, "Sequence/Slack I" (SSI). We conclude

Theorem 6.1 *Under our $GI/GI/1$ assumptions, with $E(P) < E(T)$, almost surely,*

$$\lim_{n\to\infty}\frac{Z_n^*}{n^2} = \lim_{n\to\infty}\frac{Z_n^{\mathrm{FCFSQ}}}{n^2} = \lim_{n\to\infty}\frac{Z_n^{\mathrm{SSI}}}{n^2}.$$

Proof Recall the definitions of $N(n)$, M_k, and l_k in the proof of Lemma 4.2. The remaining slack of job i ($R\bar{S}L_i$) in the resulting SSI schedule is $d_i - C_i^{\mathrm{SSI}}$. Observation 3.2 implies that, in chain k,

$$d_{l_k} - C_{l_k}^{\mathrm{SSI}} \le SL_{l_k} + M_k(p_{\max} - p_{\min}),$$

where $p_{\min} = \min_{i=1,\ldots,n} p_i$.

For SSI, SL_{l_k} can be bounded by $M_k p_{\max}$, since, in chain k, $N_{\mathrm{pos}[j]} \le M_k$ and $T_j(p_j)/\max\{n_j(p_j), 1\} \le p_{\max}$. Thus,

$$\sum_{i=1}^{n} SL_i = \sum_{k=1}^{N(n)}\sum_{l_k=1}^{M_k} SL_{l_k} \le p_{\max}\sum_{k=1}^{N(n)} M_k^2.$$

Combining these terms:

$$0 \le \sum_{i=1}^{n} d_i - C_i^{\mathrm{SSI}} \le (2p_{\max} - p_{\min})\sum_{k=1}^{N(n)} M_k^2.$$

$\sum_{k=1}^{N(n)} M_k^2/n^2$ approaches 0, as n approaches infinity (see the proof of Lemma 4.2). Therefore,

$$\lim_{n\to\infty} \frac{\sum_{i=1}^{n} Z_n^{\mathrm{SSI}}}{n^2} = \lim_{n\to\infty} \frac{\sum_{i=1}^{n} C_i^{\mathrm{SSI}}}{n^2} \quad \text{a.s.}$$

Combining this result with Lemmas 2.2, 4.2, and Theorem 2.1 completes the proof. □

In Sect. 7, we demonstrate the effectiveness of SSI for smaller instances with $E(P) < E(T)$ cases through computational testing. We also demonstrate the effectiveness of SSI for $E(P) > E(T)$ cases.

6.2 Instance size is known

Now, consider a scenario in which we know the total number of jobs n in advance. This applies to certain real-world scenarios, such as when management establishes production goals through contracts, the strategic forecasting of demand, or the control of input volume in relation to production capacity.

Recall that we would like to assign job i a slack $\sum_{k=i+1}^{n}(\mathbf{I_i}(P_k)P_k)$ if $\rho_i > 1$, or 0 otherwise. First, we consider the case in which the processing time distribution and mean $E(T)$ of the inter-arrival time distribution are known. When $E(P) < E(T)$, we have $\rho_i < 1$ (see (19)) $\forall i$ and thus assign each job a zero slack. This generates an asymptotically optimal RDDQ schedule (see the discussion for Lemma 4.2).

When $E(P) > E(T)$, job i with $\rho_i < 1$ is assigned a zero slack. However in an on-line environment, the value of $\sum_{k=i+1}^{n}(\mathbf{I_i}(P_k)P_k)$ generally cannot be calculated at the time when job i arrives, if $\rho_i > 1$. However, $(n-i)E(\mathbf{I_i}(P)P)$ can be used as a reasonable estimate of this quantity. Note that, when a job i arrives, SP first attempts to move this job from the last position ahead in the waiting list by exchanging it with one job at a time until it is ahead of all jobs with processing times greater than p_i. Of course, sometimes this is not possible without violating the due dates of some jobs. That is, SP may be "blocked" by an earlier job (or jobs) that currently does not have enough remaining slack to exchange with i to account for i and other jobs between it and i. In this case, if possible, SP will skip these "blocking" jobs and exchange i with a job that appears in the list before these "blocking" jobs. The following lemma addresses the characteristics of these "blocking" jobs (see Fig. 5).

Lemma 6.2 *Under the $GI/GI/1$ assumptions of the RDDQ model, suppose the distribution of P and $E(T)$ are known, and that $E(P) > E(T)$. Let $SL_i = (n-i)E(\mathbf{I_i}(P)P)$ if $\rho_i > 1$, or $SL_i = 0$ otherwise. Suppose that j_n^* and j_n^0 are the first pair of two jobs on the waiting list*

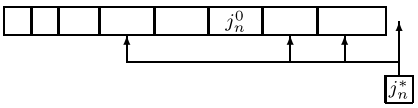

Fig. 5 Diagram of SP sequencing j_n^* by exchanging it with jobs, and being blocked by j_n^0

where $\rho_{j_n^0} > 1$, $p_{j_n^0} > p_{j_n^*}$, and $r_{j_n^*} > r_{j_n^0}$, and SP cannot exchange their positions because of violation of $d_{j_n^0}$, although we would like to make this switch since $p_{j_n^0} > p_{j_n^*}$. Then, the following holds almost surely:

$$\lim_{n\to\infty} \frac{j_n^*}{n} = 1. \tag{20}$$

Proof It is clear that the current remaining slack of job j_n^0, $R\bar{S}L_{j_n^0}$, must be less than $p_{j_n^*}$, because it is infeasible to exchange j_n^* with j_n^0. However, SP might be able to move j_n^* before j_n^0 by exchanging with jobs scheduled prior to j_n^0 in the waiting list, which are longer than j_n^* and have an enough remaining slack to switch (see Fig. 5). Recall that this is precisely what SP looks for. The above indicates that

$$\begin{aligned} 0 \le R\bar{S}L_{j_n^0} = SL_{j_n^0} - \sum_{k=j_n^0+1}^{j_n^*-1} (\mathbf{I_{j_n^0}}(P_k)P_k) < p_{j_n^*}, \\ \frac{-p_{j_n^*}}{n-j_n^0} \le E(\mathbf{I_{j_n^0}}(P)P) - \frac{j_n^*-j_n^0}{n-j_n^0} \\ \times \frac{\sum_{k=j_n^0+1}^{j_n^*} (\mathbf{I_{j_n^0}}(P_k)P_k)}{j_n^*-j_n^0} < 0. \end{aligned} \tag{21}$$

We may choose a small constant λ, $0 < \lambda < 1$. For randomly generated instances of n jobs, either of the following two cases holds for an instance:

Case 1: $(n-j_n^0)/n \ge \lambda$.
When n gets large, $n - j_n^0$ increases as Cn, where $\lambda \le C \le 1$, since $n - j_n^0 \le n$, and thus $p_{j_n^*}/(n-j_n^0)$ decreases. $\sum_{k=j_n^0+1}^{j_n^*} (\mathbf{I_{j_n^0}}(P_k)P_k)$ grows as $O(j_n^* - j_n^0)$. Also, $j_n^* - j_n^0$ cannot be $o(n-j_n^0)$, otherwise (21) cannot hold, as n approaches infinity. Moreover, by the strong law of large numbers, $\sum_{k=j_n^0+1}^{j_n^*} (\mathbf{I_{j_n^0}}(P_k)P_k)/(j_n^* - j_n^0)$ approaches $E(\mathbf{I_{j_n^0}}(P)P)$ almost surely, as $j_n^* - j_n^0$ approaches infinity. This implies that $j_n^* - j_n^0$ has to increase, as $n - j_n^0$ increases, and

$$\lim_{n\to\infty} \frac{j_n^* - j_n^0}{n - j_n^0} = 1.$$

Since $(j_n^* - j_n^0)/(n-j_n^0) = 1 - (n-j_n^*)/(n-j_n^0)$ and $n - j_n^0$ increases as Cn, the equation above implies (20).

Case 2: $(n - j_n^0)/n < \lambda$.

Since $j_n^* > j_n^0$, we know that $n - j_n^* < n - j_n^0 < \lambda n$. This leads to (20).

When n approaches infinity, these two cases converge to the same result, completing the proof. □

Recall that, for the $E(P) > E(T)$ cases, we can define A-, B-, and C-jobs by choosing p_A, p_B, and p_C if processing time is drawn from a continuous known distribution, so that (18) is obtained by assigning a zero slack to A- and B-jobs and a very large slack to C-jobs. Now, for job i, if $\rho_i > 1$, then $SL_i = (n-i)E(\mathbf{I_i}(P)P)$ and if not, $SL_i = 0$. That is, for job i, if it is of Type A, we must have $\rho_i < 1$ and $SL_i = 0$. If it is of Type B, both $\rho_i < 1$ and $\rho_i > 1$ are possible, so some B-jobs have a zero slack and other B-jobs do not. If job i is of Type C, we must have $\rho_i > 1$ and $SL_i = (n-i)E(\mathbf{I_i}(P)P)$. Although now not every B-job has a zero slack in contrast to the assumptions behind (18) (all B-jobs have a zero slack), and this difference leads to changes in sequencing between some A- and B-jobs, this difference is negligible when taking p_A very close to p_B (i.e., α_B or the number of B-jobs is very small). In addition, before j_n^* arrives, any job i ($i < j_n^*$) with $\rho_i > 1$ has a sufficient slack (which works like a very large slack) to allow for exchanges with future jobs shorter than p_i that arrive before j_n^*.

Therefore, by time $r_{j_n^*}$, the sequence of jobs 1 through $j_n^* - 1$ is as effective as if SPTA were applied to these jobs, in terms of the sum of completion times. Furthermore, Lemma 6.2 states that the number of jobs after j_n^* is $o(n)$. So, we conclude that the schedule of all n jobs by the slack determination rule ($SL_i = (n-i)E(\mathbf{I_i}(P)P)$ if $\rho_i > 1$, or $SL_i = 0$ otherwise, for job i) with SP should provide a total completion time which approaches that provided under SPTA, as n gets large. That is, (18) still applies. Next, we address the difference between due dates and completion times:

Lemma 6.3 *Under the $GI/GI/1$ assumptions of the RDDQ model, suppose the distribution of P and $E(T)$ are known, and that $E(P) > E(T)$. Let $SL_i = (n-i)E(\mathbf{I_i}(P)P)$ if $\rho_i > 1$, or $SL_i = 0$ otherwise. Then, the following holds almost surely when SP is applied*:

$$\lim_{n\to\infty} \frac{\sum_{i=1}^{n}(d_i - C_i)}{n^2} = 0.$$

Proof Recalling the definition of j_n^*, consider the schedule (denoted by $S_{j_n^*-1}$) of jobs 1 through $j_n^* - 1$ by the time $r_{j_n^*}$. For any job in $S_{j_n^*-1}$, its slack is sufficient enough to allow any jobs shorter than it to be sequenced before it, if these shorter jobs arrive when it is in the waiting list.

Recall the determination of the type of a job in Sect. 5.3 using p_A, p_B, and p_C when processing time is continuous, and apply it to jobs 1 through $j_n^* - 1$. Figure 4 presents $S_{j_n^*-1}$ in terms of the types of jobs, in which the last group of C-jobs after the last AB-group is denoted by S^0. Note that "a group of jobs" represents a series of jobs consecutively processed without interruption by idle times. Consider job i, where $i \in S^0$ and $p_i > p_B$ (so $\rho_i > 1 + \sigma$). If we re-define Type C to be at least p_i (equivalently, letting p_B be p_i) and A- and B-jobs to be less than p_i as before, Fig. 4 will still show $S_{j_n^*-1}$ in terms of the types of jobs. That is, job i (Type C) will be sequenced after the last AB-group with associated processing time less than p_i. So, for job i, all future jobs shorter than it arrive when it is in the queue, and its $SL_i = (n-i)E(\mathbf{I_i}(P)P)$ allow for their being sequenced before it. These shorter jobs ($< p_i$, i.e., A- and B-jobs) join the last group of jobs shorter than p_i which dominates the number of jobs shorter than p_i.

Denote by $R\bar{S}L_i^*$ the remaining slack of job i, $i \in S^0$, by time $r_{j_n^*}$. The argument above indicates:

$$\begin{aligned} R\bar{S}L_i^* &= (n-i)E\big(\mathbf{I_i}(P)P\big) - \sum_{k=i+1}^{j_n^*-1}\big(\mathbf{I_i}(P_k)P_k\big) \\ &= \big(n - j_n^* + 1\big)E\big(\mathbf{I_i}(P)P\big) \\ &\quad + \big(j_n^* - 1 - i\big)E(\mathbf{I_i}(P)P) - \sum_{k=i+1}^{j_n^*-1}\big(\mathbf{I_i}(P_k)P_k\big), \end{aligned}$$

$$\begin{aligned} \sum_{i\in S^0} R\bar{S}L_i^* &\le \sum_{i\in S^0}\big(n - j_n^* + 1\big)E\big(\mathbf{I_i}(P)P\big) \\ &\quad + \big(j_n^* - 1\big)\sum_{i=1}^{j_n^*-1}\left|E\big(\mathbf{I_i}(P)P\big)\right. \\ &\quad \left. - \frac{\sum_{k=i+1}^{j_n^*-1}(\mathbf{I_i}(P_k)P_k)}{(j_n^* - 1) - i}\right|. \end{aligned} \tag{22}$$

Define an array $\{a_q, q = 1, 2, \ldots, j_n^* - 1\}$, where $a_1 = E(\mathbf{I_{(j_n^*-1)}}(P)P)$ and for $q > 1$,

$$a_q = E(\mathbf{I_{(j_n^*-q)}}(P)P) - \frac{\sum_{j=2}^{q}(\mathbf{I_{(j_n^*-q)}}(P_j)P_j)}{q-1}.$$

Thus, from (22):

$$\begin{aligned} \frac{1}{n^2}\sum_{i\in S^0} R\bar{S}L_i^* &\le \frac{1}{n^2}\sum_{i\in S^0}\big(n - j_n^* + 1\big)E\big(\mathbf{I_i}(P)P\big) \\ &\quad + \frac{(j_n^* - 1)^2}{n^2}\frac{\sum_{q=1}^{j_n^*-1}|a_q|}{j_n^* - 1}. \end{aligned}$$

$\sum_{i\in S^0}(n - j_n^* + 1)E(\mathbf{I_i}(P)P) \le |S^0|(n - j_n^* + 1)p_{\max}$ which can be seen to grow as $o(n^2)$ using (20). Since $a_{j_n^*-1}$ ap-

proaches 0 almost surely (the strong law), as j_n^* approaches infinity, we have

$$\lim_{j_n^* \to \infty} \frac{\sum_{q=1}^{j_n^*-1} |a_q|}{j_n^* - 1} = 0 \quad \text{a.s.}$$

Combined with (20), we conclude that $\sum_{i \in S^0} R\bar{S}L_i^*$ is $o(n^2)$ almost surely. However, $S_{j_n^*-1}$ includes jobs in S^0, A- (shorter than p_A) and B- (between p_A and p_B) jobs, and other C-jobs which do not have all future shorter jobs arrive when in the queue. $|S^0|$ dominates the number of C-jobs i in $S_{j_n^*-1}$. $RSL_i^* = 0$ if job i is of Type A ($\rho_i \le 1 - \sigma$). B-jobs can be made insignificant by choosing p_A and p_B close to each other (σ gets small). This implies that $\sum_{i=1}^{j_n^*-1} R\bar{S}L_i^*$ grows $o(n^2)$ almost surely.

Now consider the final schedule which adds jobs j_n^* through n to $S_{j_n^*-1}$. Since these jobs might be blocked by some jobs on the waiting list when being sequenced under SP, Observation 3.2 applies, and thus each of these additional jobs might add at most $(p_{\max} - p_{\min})$ to $R\bar{S}L_i^*$ for $i \in S_{j_n^*-1}$. More precisely, for $i \in S_{j_n^*-1}$,

$$0 \le d_i - C_i \le R\bar{S}L_i^* + \big(n - j_n^* + 1\big)(p_{\max} - p_{\min}).$$

However, for $i = j_n^*, j_n^* + 1, \ldots, n$, the observation above also applies. Thus,

$$\begin{aligned} 0 &\le d_i - C_i \le SL_i + \big(n - j_n^* + 1\big)(p_{\max} - p_{\min}) \\ &\le \big(n - j_n^* + 1\big)p_{\max} + \big(n - j_n^* + 1\big)(p_{\max} - p_{\min}). \end{aligned}$$

Note that $(n - j_n^* + 1)$ grows as $o(n)$ (see (20)). Thus, the above implies that $\sum_{i=1}^{j_n^*-1}(d_i - C_i)$ and $\sum_{i=j_n^*}^{n}(d_i - C_i)$ grow like $o(n^2)$ almost surely, completing the proof. □

Lemma 6.3 suggests that the difference between quoted due dates and completion times is negligible relative to the order of the total completion time. Combined with (18) and Lemma 4.1, we can conclude that this approach is asymptotically optimal under the conditions listed below:

Theorem 6.4 *Under the $GI/GI/1$ assumptions of the RDDQ model, suppose the distribution of P and $E(T)$ are known, continuous, and bounded, and that $E(P) > E(T)$. Let $SL_i = (n - i)E(\mathbf{I_i}(P)P)$ if $\rho_i > 1$, or $SL_i = 0$ otherwise. Almost surely,*

$$\lim_{n \to \infty} \frac{Z_n^*}{n^2} = \lim_{n \to \infty} \frac{\sum_{i=1}^{n} d_i}{n^2},$$

where d_i is the due date, when SP is applied.

Up to this point in this subsection, we have assumed that the processing time distribution and $E(T)$ are known in Theorem 6.4. This may not always be the case, however. Below, we sketch an approach when these quantities need to be estimated.

Recall that for SL_i of job i with $\rho_i > 1$, $E(\mathbf{I_i}(P)P) = E(P|P < p_i)Pr(P < p_i)$. In order to utilize the approach for assigning slack described above, at the time r_i, we need to estimate the parameters that we previously assumed to be known (e.g., $E(T)$, $E(P|P < p_i)$, $Pr(P < p_i)$) based on the jobs that have previously arrived.

In SP (Algorithm 1), job j represents a new arrival. Let $\hat{T}_j$ represent the current estimate of the expected inter-arrival time. Recall that $n_j(p_j)$ represents the number of jobs that have arrived before job j with processing times less than p_j and $T_j(p_j)$ is the total processing time of these jobs, and recall that we have defined $\hat{P}(p_j) = \frac{T_j(p_j)}{\max\{n_j(p_j),1\}}$ to be the estimate of $E(P|P < p_j)$. In addition, let $\hat{p}(p_j)$ be the fraction of arrivals with processing times less than p_j by time r_j (i.e., an estimate of $Pr(P < p_i)$). Thus,

$$\hat{p}(p_j) = \frac{n_j(p_j)}{j}.$$

Then, an estimate of ρ_j is

$$\rho_j' \leftarrow \frac{\hat{P}(p_j)}{\hat{T}_j}\hat{p}(p_j),$$

which we use in this case to determine SL_j.

Note that even when the distribution of P and $E(T)$ are known, some jobs i with $\rho_i > 1$ may be sequenced before jobs shorter than p_i (e.g., this can happen to a job i with $\rho_j > 1$ sequenced before the last chain). That is, for these jobs, $SL_i = (n - i)E(\mathbf{I_i}(P)P)$ is an excessive slack that accounts for all jobs shorter than p_i arriving after r_i. This suggests that it is probably better not to assign the slack $SL_i = (n - i)E(\mathbf{I_i}(P)P)$ to job i with $\rho_i > 1$ until a certain number of jobs have arrived and the last chain has started. Moreover, inaccurate estimates may also result in unnecessarily long due dates (e.g., $\rho_j' > 1$ for job j when $\rho_j < 1$) or insufficient slack (e.g., $\rho_j' < 1$ for job j when $\rho_j > 1$). In order to practically use this heuristic, we therefore should only use these estimates to determine slack after a certain number of jobs have arrived, so the estimates can be more accurate. Here we use $\sqrt{n}$ as the starting job for on-line slack assignment. Since it is $o(n)$, the jobs before it do not affect the overall objective significantly. On the other hand, $\sqrt{n}$ is big enough that the problems described above may be avoided.

The updated slack assignment rule is detailed in Algorithm 2, which we call "Slack II" (SII). The heuristic that uses SP and SII is denoted "Sequence/Slack II" (SSII). Note that r_1 is some strictly positive constant representing the arrival time of the first job.

Clearly, $\hat{P}(p_j)$ and $\hat{p}(p_j)$ will be good estimates for $E(P|P < p_i)$ and $Pr(P < p_i)$, as j gets larger. In the following lemma, we show that ρ_j' is an effective estimate of

Algorithm 2

$\hat{T}_j \leftarrow \frac{r_j}{j}$ and calculate ρ'_j.
if $\rho'_j \leq 1$ or $j < \sqrt{n}$ **then**
  $SL_j \leftarrow 0$
else
  $SL_j \leftarrow (n-j) \cdot \hat{P}(p_j) \cdot \hat{p}(p_j)$
end if

ρ_j. This further implies the effectiveness (even the asymptotic optimality) of SSII.

Lemma 6.5 *When the instance size n approaches infinity, almost surely, the number of jobs that have* $\rho_j < 1$ *but are estimated* $\rho'_j \geq 1$, *and the number of jobs that have* $\rho_j > 1$ *but are estimated* $\rho'_j \leq 1$, *do not affect the asymptotic objective.*

Proof Let

$$S_j = \sum_{k=1}^{j-1} \left(\mathbf{I_j}(P_k)P_k - T_k\right) - r_1,$$

so S_j can be viewed as the location after $j-1$ steps of a one-dimensional random walk, starting at location $-r_1$. The kth step moves $\mathbf{I_j}(P_k)P_k - T_k$, where $E(\mathbf{I_j}(P_k)P_k - T_k) = E(T)(\rho_j - 1)$. It is well known (see, for example, Ross 1996) that S_j approaches negative infinity, as j approach infinity, if $E(\mathbf{I_j}(P)P - T) < 0$, that is, if $\rho_j < 1$. In this case, almost surely, there are an insignificant number of visits to locations $\geq M$, where M is a positive constant. This implies that, as n approaches infinity, almost surely S_j is greater than $M = 0$ an insignificant number of times from $j = 1$ to $j = n$ if $\rho_j < 1$. Note that

$$\rho'_j = \frac{T_j(p_j)/j}{r_j/j} = \frac{\sum_{k=1}^{j-1} \mathbf{I_j}(P_k)P_k}{r_1 + \sum_{k=1}^{j-1} T_k},$$

so $S_j \geq 0 \Leftrightarrow \rho'_j \geq 1$. Thus, S_j is ≥ 0 (and $\rho'_j \geq 1$) an insignificant number of times from $j = 1$ to $j = n$ if $\rho_j < 1$, as n goes to infinity. Similarly, S_j is ≤ 0 (and $\rho'_j \leq 1$) an insignificant number of times from $j = 1$ to $j = n$ if $\rho_j > 1$. □

7 Computational testing

We note that FCFSQ, SSI, and SSII are all polynomial algorithms, although FCFSQ is $\Theta(n)$, while SSI and SSII are $\Theta(n^2)$. In this section, we present the results of a variety of computational tests we performed to determine the effectiveness of these heuristics. We generated different size random problem instances with inter-arrival times and processing times drawn from exponential distributions and uniform distributions, in order to observe how the algorithm performance changes depending on the size of the instance, and the degree of saturation (that is, the offered load or the relationship between inter-arrival time and processing time).

In the simulation results that follow, we show the ratios of each heuristic to the SPTA objective value which is an asymptotic lower bound as discussed in Sect. 2. Each ratio is based on the average of 10 trials for each data point. The results confirm the theoretical results developed elsewhere in the paper and give some sense of the rate of convergence of asymptotic results, at least for parameters tested.

For our computational testing of these heuristics, we generated instances of size $n = 500$, 1000, 2500, and 5000 with inter-arrival times and processing times drawn from exponential distributions and uniform distributions. Without loss of generality, the arrival time of the first job is set to time 0. Define the following:

- R_0 = the ratio of Z_n^{FCFSQ} to Z_n^{SPTA};
- R_1 = the ratio of Z_n^{SSI} to Z_n^{SPTA};
- R_2 = the ratio of Z_n^{SSII} to Z_n^{SPTA};
- σ_P = standard deviation of processing time;
- ε_i = sample standard deviation of R_i, $i = 0, 1, 2$.

Note that if $E(P)/E(T) < 1$, Z_n^{FCFSQ}, Z_n^{SSI}, and Z_n^{SSII} approach Z_n^{SPTA}, as the total number of jobs is large, as suggested by Theorem 6.1. These results can be found in Tables 1, 2, and 3. The performance is comparable in each of the three tables, although slightly worse with exponential processing times. Also, SSI performs better when $E(P)/E(T)$ or σ_P is bigger. When $E(P)/E(T) > 1$, the performance of SSII is substantially better than SSI, and indeed, SSII is observed to approach Z_n^{SPTA}, as n gets larger (Theorem 6.4).

8 Concluding remarks

We have developed a conceptual model for the due date quotation problem, proposed a class of on-line heuristics for it, provided a probabilistic analysis of the performance of these algorithms, and computationally tested them. We have characterized conditions under which these heuristics are asymptotically optimal, and our computational testing has shown them to be effective for smaller instances and in situations in which they are not asymptotically optimal. In particular, we have found that a broad class of these heuristics are effective when the system is undersaturated, and a smaller and more complex class of heuristics are effective when the system is oversaturated. Analysis of the system when $E(P) = E(T)$ remains an open question. To the best of our knowledge, this and related problem have generally not been addressed using analytical tools, and in particular, probabilistic analysis, elsewhere in the literature.

Table 1 Uniform $[0, 2E(T)]$ inter-arrival times and uniform $[p_{\min}, p_{\max}]$ processing times

$p_{\min}$	$p_{\max}$	$E(T)$	$\frac{E(P)}{E(T)}$	σ_P	$n = 500$			$n = 1000$			$n = 2500$			$n = 5000$		
					R_0	R_1	R_2	R_0	R_1	R_2	R_0	R_1	R_2	R_0	R_1	R_2
					ε_0	ε_1	ε_2	ε_0	ε_1	ε_2	ε_0	ε_1	ε_2	ε_0	ε_1	ε_2
0.75	7.25	1	4	1.87	1.32	1.28	1.04	1.32	1.27	1.04	1.33	1.28	1.02	1.33	1.27	1.02
					0.03	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.00	0.00
0.5	2.5	0.5	3	0.58	1.23	1.22	1.05	1.24	1.22	1.03	1.24	1.22	1.02	1.24	1.21	1.01
					0.01	0.02	0.01	0.01	0.01	0.01	0.00	0.01	0.00	0.01	0.00	0.00
0.5	1	0.5	1.5	0.14	1.07	1.07	1.04	1.07	1.07	1.02	1.07	1.07	1.01	1.07	1.07	1.01
					0.01	0.01	0.01	0.00	0.00	0.01	0.00	0.00	0.01	0.00	0.00	0.00
0.2	0.6	0.5	0.8	0.11	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
					0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.2	0.4	0.5	0.6	0.06	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
					0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 2 Exponential inter-arrival times with rate $1/E(T)$ and exponential processing times with rate $1/E(P)$

$E(P)$	$E(T)$	$\frac{E(P)}{E(T)}$	σ_P	$n = 500$			$n = 1000$			$n = 2500$			$n = 5000$		
				R_0	R_1	R_2	R_0	R_1	R_2	R_0	R_1	R_2	R_0	R_1	R_2
				ε_0	ε_1	ε_2	ε_0	ε_1	ε_2	ε_0	ε_1	ε_2	ε_0	ε_1	ε_2
4	1	4	4	1.79	1.53	1.11	1.78	1.50	1.05	1.78	1.48	1.05	1.79	1.49	1.04
				0.08	0.08	0.03	0.04	0.03	0.02	0.03	0.03	0.02	0.02	0.02	0.02
1.5	0.5	3	1.5	1.68	1.45	1.09	1.68	1.42	1.08	1.69	1.42	1.04	1.69	1.43	1.03
				0.07	0.06	0.03	0.03	0.03	0.03	0.03	0.02	0.02	0.02	0.02	0.01
0.75	0.5	1.5	0.75	1.30	1.19	1.07	1.27	1.18	1.05	1.31	1.20	1.04	1.31	1.20	1.02
				0.06	0.04	0.03	0.04	0.03	0.02	0.02	0.02	0.01	0.02	0.01	0.01
0.4	0.5	0.8	0.4	1.01	1.01	1.01	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
				0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.3	0.5	0.6	0.3	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
				0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

This model can be extended to more complex scheduling environments. For example, in the flow shop RDDQ model, it can be assumed that each job is quoted a due date upon its arrival and must be completed on the last machine at or by the due date. The objective in this case is to determine the sequences of all jobs on each machine as well as a set of due dates for all jobs such that the sum of due dates (or lead times) is minimized. In a flow shop, as in our original model, after a new job arrives and is sequenced in the waiting list on the first machine, some future arrivals might be sequenced before it. This can cause the delays in processing this new job on machine one and on the following machines. Xia et al. (2000) have shown that for a flow shop, in a permutation schedule, the time that each job spends waiting after it has completed processing on the first machine can be $o(n)$ almost surely if certain conditions hold. This implies that an effective sequencing and slack assignment rule for a single machine might be used to generate an effective permutation schedule for a flow shop. Lee (2003) builds on this insight and proposes a flow shop RDDQ model, and on-line algorithms for this model. He shows that under certain conditions (analogous to the conditions in this paper), some of these heuristics are asymptotically optimal for this problem.

For the objective of minimizing $\sum_{i=1}^{n} d_i$, $\lim_{n\to\infty} Z_n/n^2$ (Z_n generally represents the objective value of the model) is a reasonable criterion under our model because the total completion time is $\Theta(n^2)$ and $\sum_{i=1}^{n} d_i \geq \sum_{i=1}^{n} C_i$. However, if the objective is to minimize $\sum_{i=1}^{n}(d_i - p_i - r_i)$

Table 3 Exponential inter-arrival times with rate $1/E(T)$ and uniform $[p_{\min}, p_{\max}]$ processing times

$p_{\min}$	$p_{\max}$	$E(T)$	$\frac{E(P)}{E(T)}$	σ_P	$n = 500$			$n = 1000$			$n = 2500$			$n = 5000$		
					R_0	R_1	R_2	R_0	R_1	R_2	R_0	R_1	R_2	R_0	R_1	R_2
					ε_0	ε_1	ε_2	ε_0	ε_1	ε_2	ε_0	ε_1	ε_2	ε_0	ε_1	ε_2
0.75	7.25	1	4	1.87	1.32	1.28	1.05	1.33	1.28	1.04	1.33	1.27	1.03	1.33	1.27	1.02
					0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.00
0.5	2.5	0.5	3	0.58	1.24	1.22	1.05	1.24	1.22	1.03	1.24	1.22	1.02	1.24	1.21	1.02
					0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.00	0.01	0.00	0.01	0.00
0.5	1	0.5	1.5	0.14	1.07	1.07	1.05	1.07	1.07	1.03	1.07	1.07	1.02	1.07	1.07	1.01
					0.01	0.01	0.01	0.00	0.00	0.01	0.00	0.00	0.01	0.00	0.00	0.00
0.2	0.6	0.5	0.8	0.11	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
					0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.2	0.4	0.5	0.6	0.06	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
					0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

instead, this criterion is not necessarily appropriate for $E(P) < E(T)$ cases. In these cases, $\sum_{i=1}^{n}(C_i - p_i - r_i)$ grows as n, and so $\lim_{n\to\infty} Z_n/n$ will be a better testing criterion. Our algorithms don't achieve asymptotic optimality in this sense – that is, it is not true that $\lim_{n\to\infty} \sum_{i=1}^{n}(d_i - C_i)/n = 0$ almost surely for these cases. Designing an algorithm for these cases will be a focus of future research.

Clearly there are other objectives that are important to management. Also, many of the conditions we require in our probabilistic analysis, such as stationary processing times and independent processing and inter-arrival times, and infinitely many jobs, are frequently inappropriate for real life problems. Nevertheless, it is encouraging to see that a relatively simple algorithm performs well on these problems, and that this performance can at least partially be explained analytically. We hope to build on this insight in future research as we consider more complex models and objectives.

Acknowledgements The authors would like to thank Professors Sheldon Ross and Hyun-soo Ahn for their helpful insights, and the anonymous associate editor and referees for their many helpful comments. This research was partially supported by NSF Grants DMI-0092854 and DMI-0200439.

References

Baker, K. R., & Bertrand, J. W. M. (1981). A comparison of due-date selection rules. *AIIE Transactions*, *13*, 123–131.

Bertrand, J. W. M. (1983). The effect of workload dependent due-dates on job shop performance. *Management Science*, *29*, 799–816.

Brucker, P. (1998). *Scheduling algorithms*. New York: Springer.

Chand, S., & Chhajed, D. (1992). A single machine model for determination of optimal due date and sequence. *Operations Research*, *40*, 596–602.

Chang, P., Hieh, J., & Liao, T. W. (2005). Evolving fuzzy rules for due date assignment problem in semiconductor manufacturing factory. *Journal of Intelligent Manufacturing*, *16*(4–5), 549.

Cheng, T. C. E., & Gupta, M. C. (1989). Survey of scheduling research involving due date determination decisions. *European Journal of Operational Research*, *38*, 156–166.

Coffman, E. G., Frederickson, G. N., & Lueker, G. S. (1982). Probabilistic analysis of the LPT processor scheduling heuristic. In M. A. H. Dempster et al. (Eds.), *Deterministic and stochastic scheduling* (pp. 319–331). Dordrecht: Reidel.

Duenyas, I. (1995). Single facility due date setting with multiple customer classes. *Management Science*, *41*, 608–619.

Duenyas, I., & Hopp, W. J. (1995). Quoting customer lead times. *Management Science*, *41*, 43–57.

Eilon, S., & Chowdhury, I. G. (1976). Due dates in job shop scheduling. *International Journal of Production Research*, *14*, 223–237.

Frenk, J. B. G., & Rinnooy Kan, A. H. G. (1987). The asymptotic optimality of the LPT rule. *Mathematics of Operations Research*, *12*, 241–254.

Gazmuri (1985). Probabilistic analysis of a machine scheduling problem. *Mathematics of Operations Research*, *10*, 328–339.

Hall, N. G., & Posner, M. E. (1991). Earliness-tardiness scheduling problems, I: weighted deviation of completion times about a common due date. *Operations Research*, *39*, 836–846.

Hsu, S. Y., & Sha, D. Y. (2004). Due date assignment using artificial neural networks under different shop floor control strategies. *International Journal of Production Research*, *42*(9), 1727–1745.

Kahlbacher, H. (1992). *Termin-und Ablaufplanung—ein analytischer Zugang*. PhD thesis, University of Kaiserslautern, cited in [3].

Kaminsky, P. (2003). The effectiveness of the longest delivery time rule for the flow shop delivery time problem. *Naval Research Logistics*, *50*(3), 257–272.

Kaminsky, P., & Hochbaum, D. (2004). Due date quotation models and algorithms. In J. Y. Leung (Ed.), *Handbook on scheduling algorithms, methods and models*. London: Chapman Hall/CRC.

Kaminsky, P., & Simchi-Levi, D. (1998). Probabilistic analysis and practical algorithms for the flow shop weighted completion time problem. *Operations Research*, *46*, 872–882.

Kaminsky, P., & Simchi-Levi, D. (2001). Probabilistic analysis of an on-line algorithm for the single machine completion time problem with release dates. *Operations Research Letters*, *29*, 141–148.

Keskinocak, P., Ravi, R., & Tayur, S. (2001). Scheduling and reliable lead-time quotation for orders with availability intervals and lead-time sensitive revenues. *Management Science*, *47*, 264–279.

Lee, Z. (2003). *Design and analysis of algorithms for due date quotation*. PhD dissertation, University of California, Berkeley.

Lenstra, J. K., Rinnooy Kan, A. H. G., & Brucker, P. (1977). Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, *1*, 343–362.

Liu, H., Queyranne, M., & Simchi-Levi, D. (2005). On the asymptotic optimality of algorithms for the flow shop problem with release dates. *Naval Research Logistics*, *52*, 232–242.

Loulou, R. (1984). Tight bounds and probabilistic analysis of two heuristics for parallel processor scheduling. *Mathematics of Operations Research*, *9*, 142–150.

Miyazaki, S. (1981). Combined scheduling system for reducing job tardiness in a job shop. *International Journal of Production Research*, *19*, 201–211.

Panwalkar, S. S., Smith, M. L., & Seidmann, A. (1982). Common due date assignment to minimize total penalty for the one machine scheduling problem. *Operations Research*, *30*, 391–399.

Portougal, V., & Trietsch, D. (2006). Setting due dates in a stochastic single machine environment. *Computers & Operations Research*, *33*(6), 1681–1694.

Ramudhin, A., Bartholdi, J. J., Calvin, J., Vande Vate, J. H., & Weiss, G. (1996). A probabilistic analysis of 2-machine flowshops. *Operations Research*, *44*, 899–908.

Ross, S. M. (1996). *Stochastic processes* (2nd ed.). New York: Wiley.

Seidmann, A., Panwalker, S. S., & Smith, M. L. (1981). Optimal assignment of due dates for a single processor scheduling problem. *International Journal of Production Research*, *19*, 393–399.

Slotnick, S. A., & Sobel, M. J. (2005). Manufacturing lead-time rule: customer retention versus tardiness costs. *European Journal of Operational Research*, *163*(3), 825–856.

Spaccamela, A. M., Rhee, W. S., Stougie, L., & van de Geer, S. (1992). Probabilistic analysis of the minimum weighted flowtime scheduling problem. *Operations Research Letters*, *11*, 67–71.

Spearman, M., & Zhang, R. Q. (1999). Optimal lead time policies. *Management Science*, *45*, 290–295.

Webster, S. (1993). Bounds and asymptotic results for the uniform parallel processor weighted flow time problem. *Operations Research Letters*, *41*, 186–193.

Weeks, J. K. (1979). A simulation study of predictable due-dates. *Management Science*, *25*, 363–373.

Wein, L. M. (1991). Due-date setting and priority sequencing in a multiclass M/G/1 queue. *Management Science*, *37*, 834–850.

Xia, C., Shanthikumar, G., & Glynn, P. (2000). On the asymptotic optimality of the SPT rule for the flow shop average completion time problem. *Operations Research*, *48*, 615–622.