Discrete Scheduling and Critical Utilization

Oleg S. Pianykh , Sebastian Perez , Chengzhao “Richard” Zhang

PII: S0377-2217(23)00461-7
DOI: https://doi.org/10.1016/j.ejor.2023.06.010
Reference: EOR 18518

To appear in: *European Journal of Operational Research*

Received date: 19 June 2022
Accepted date: 4 June 2023

Please cite this article as: Oleg S. Pianykh , Sebastian Perez , Chengzhao “Richard” Zhang , Discrete Scheduling and Critical Utilization, *European Journal of Operational Research* (2023), doi: https://doi.org/10.1016/j.ejor.2023.06.010

Highlights

- Discrete scheduling efficiency degrades hyperbolically as schedule utilization increases
- Concise expression for maximum schedule utilization can be found analytically
- For schedules with N slots, keeping $\sqrt{N}$ slots open helps avoid worst delays
- Utilization-based decision-making helps schedulers avoid bottlenecking

Discrete Scheduling and Critical Utilization

Oleg S. Pianykh[1*], Sebastian Perez[2], Chengzhao "Richard" Zhang[1]

[1]Harvard Medical School, Massachusetts General Hospital,

25 New Chardon Street, #470, Boston, MA 02114, USA

[2]Department of Mathematics, Massachusetts Institute of Technology

77 Massachusetts Avenue, Cambridge, MA 02139, USA

[*] *Corresponding author, email*: *opianykh@mgh.harvard.edu*

Declarations of interest: none

Abstract

Efficient scheduling is essential for optimizing resource allocation and robust system performance in a wide range of real-life applications. In most of these cases, the success of scheduling largely depends on one's ability to ensure that system resources can be utilized to their maximum capacity, yet without overloading the system. In this work, we study the problem of critical utilization and efficient scheduling by considering systems with discrete schedules, widely used in real-life workflows. Using an implementation-based approach, we introduce discrete scheduling by developing its analytic equations, which enables us to express the behavior of the scheduling metrics with respect to system utilization. Using this result, we define critical resource utilization and solve for its exact value as a function of schedule length. Finally, we compare our results with the equations from the classical queueing theory, and discuss their applicability. Our findings have immediate practical implications in developing robust schedules and controlling for optimal system performance.

Keywords

Scheduling, Utilization, Queueing

1 Introduction

Efficient and robust scheduling presents a well-known challenge in many practical areas from engineering to healthcare to e-commerce (Blake, Carter, & Richardson, 1996), (López, García, Díaz, & García, 2000), (Dhall & Liu, 1978), (Kumar, 2001), and becomes particularly demanding, when an expensive resource - such as a complex device or an operating room - needs to be utilized to its full capacity to justify the cost. However, as classical queueing theory suggests (Little, 1961), (Pollaczek, 1930), and practical experiences confirm (Brown, Gans, Mandelbaum, Sakov, & Shen, 2005) (Kc & Terwiesch.), increasing resource utilization inevitably leads to higher wait times and more fragile, bottlenecking workflows. In particular, working in a busy healthcare facility, we are constantly observing faster-than-linear escalation of delays as hospital resource utilization increases (Figure 1). Knowing the exact nature of this trend and its critical utilization threshold, responsible for escalating delays, presents one of the most central problems of practical operations management.

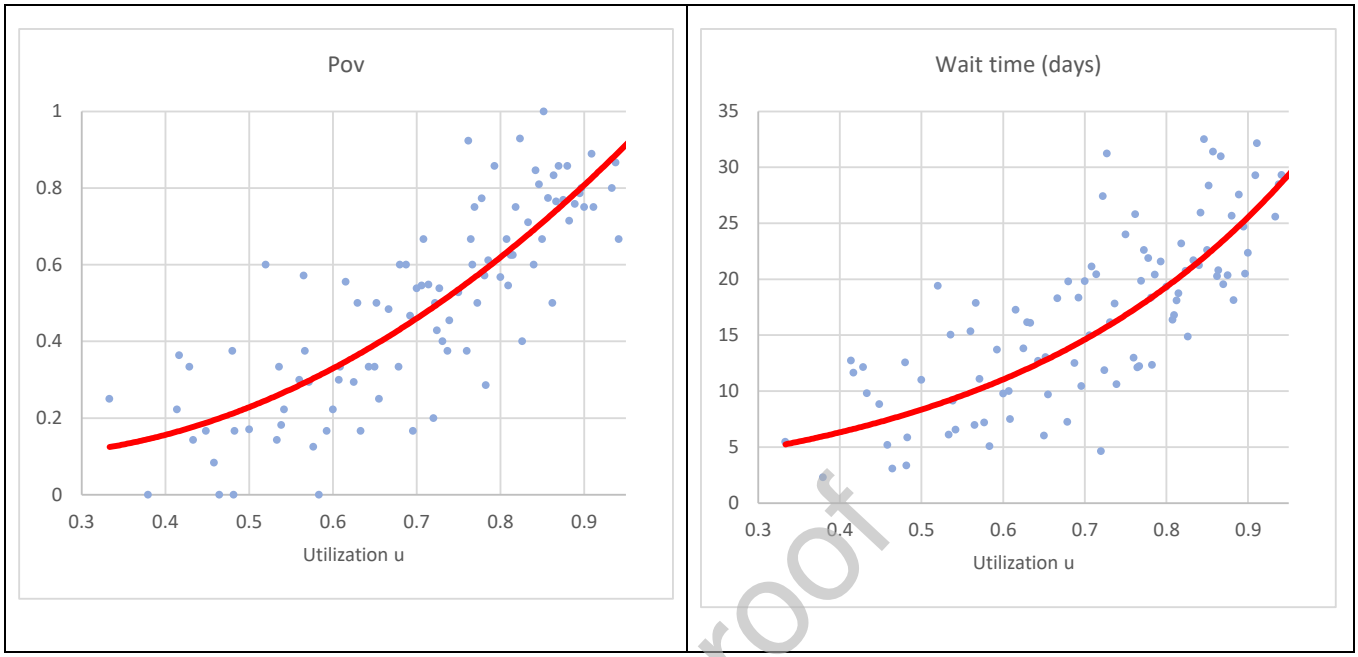

Figure 1: Patient wait time W to schedule an appointment, and probability P_{ov} of rescheduling to a later time, as functions of scheduled resource utilization, in a busy outpatient healthcare facility. Individual dots correspond to the actual observed values, smooth lines – to the resulting trends.

The problems of processing queues and delays have been extensively studied for more than a century (Erlang, 1909), and classical queueing theory developed a number of useful theoretical models to estimate waiting lines, times, and utilization (Cooper, 1981), (Kleinrock, 1975). However, these models tend to rely on a rather continuous view of queueing, where the queue is formed by a random flow of tasks, arriving to a busy server. This view does not fit well with many real-life problems, where the principal challenge comes not from the dynamics of the arrival and service timing, but from the complexity of assigning tasks to a limited number of predefined (*scheduled*) slots – such as assigning patients to hospital rooms, or assigning airplanes to airport gates. Consequently, classical queueing theory offers no advice on finding the optimal number of busy slots to avoid bottlenecking.

To investigate discrete slot-allocation problems more efficiently, more computational approaches have been developed, solving sophisticated queueing problems numerically (Neuts M. F., 1973) (Klimko & Neuts, 1973) (Neuts & Klimko, 1973) (Heimann & Neuts, 1973). This led to a wide range of scheduling algorithms based on Markov chains (Neuts M. F., 1973) (Chan & Maa, 1978), dynamic programming (LaGanga & Lawrence, 2012), convex steepest descent (Begen & Queyranne, 2011), Monte Carlo (Punitha, 2018), integer linear programming (Zacharias & Yunes, Multimodularity in the Stochastic Appointment Scheduling, 2019), discrete event simulation (DES) (Wainer, 2009), (Petrean, 1998), and

more (including healthcare scheduling, where numerical simulations were frequently used to optimize emergency room utilization (Blake, Carter, & Richardson, 1996), patient no-shows (Zacharias & Pinedo, Appointment Scheduling with No-Shows and Overbooking, 2014), and allocation of staff and resources (Marchesi, Hamacher, & Fleck, 2020), (Huggins, Claudio, & Eduardo, 2014)). This extensive analysis of numerical solvers also helped advance their theoretical grounds – including the proofs of multimodular and convex optimization for scheduling cost functions (Zacharias & Yunes, Multimodularity in the Stochastic Appointment Scheduling, 2019). However, by their very nature, numerical optimization methods were focused on simulating very specific scenarios, making their findings hard to generalize. Also, the ever-increasing complexity of computational equations and math made them prohibitevy expensive for routine, real-life applications, and did not produce any interpretable guidance for the scheduling practitioners.

As a result, in their comprehensive review of discrete scheduling methods in healthcare, (Cayirli & Veral, 2003) conclude that "discussions on implementation issues reveal how misleading it can be to view the problem as a "pure optimization" problem". Working in this field, we completely agree with this verdict: we seriously lack approaches facilitating real-life scheduling decisions. Complex optimizers cannot be run nor maintained in routine settings; complex logic cannot be followed by human schedulers and facility administrators; complex probabilistic data may not be known, or may greatly vary over time, invalidating its models. As a result, and especially in the areas with significant process variability (such as healthcare), a true real-life scheduling solution must be *optimizing day-to-day scheduling decisions* – rather than optimizing whatever is computationally possible.

To bridge this gap, we propose a new approach to discrete scheduling problems, and study it to develop a practically-applicable solution.

2 Discrete scheduling problem and "one more task" approach

We define the problem of *discrete scheduling* as the problem of assigning tasks to a finite number N of discrete schedule slots (scheduling cycle, such as a workday), as shown in Figure 2. The time window of each slot is defined ahead of time, and remains constant – thus forming a preset scheduling grid (in contrast with conventional "walk-in" task queueing). Tasks can be processed only if and when they are assigned to the remaining open slots, one task per slot.

This definition significantly changes the problem-solving paradigm: while the main goal in "walk-in" queueing is to *process the tasks on time*, the main goal in discrete scheduling is to *find the time to process*. To solve this problem in the most correct, pragmatic way, we have to put ourselves in the position of a scheduling manager, who was asked, at a random time, to accomodate *one more task*. At this point, the manager is not already concerned with the previously-scheduled tasks, which cannot be changed, even if scheduled for the future slots. Likewise, the manager is not concerned with any new tasks which might (or might not) arrive later. The only real problem the manager has to solve is whether accepting one more task is possible, and if so, whether this task will wreck operational havoc. The scheduler needs a simple and reliable rule to make this decision.

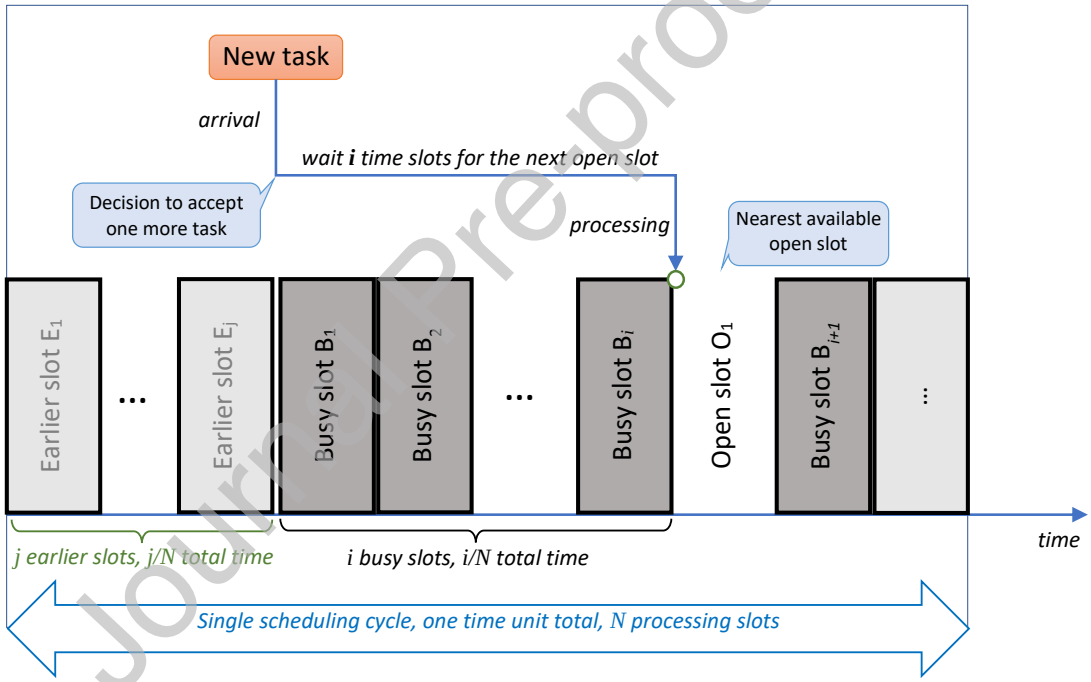

Figure 2: Discrete scheduling problem, where arriving tasks have to be assigned to a limited number of processing slots N. As the number of busy slots K increases, the new tasks will need to wait longer for the next open slot.

This new game-theory-like approach "freezes" the problem at a specific decision time, placing us in a very interesting position, significantly different from the previous scheduling optimization research. We do not need to know the temporal properties of the system, such as task arrival distribution, queueing discipline, or distribution of the service times: at a single "frozen" time point, we are not solving for the

entire scheduling template. Moreover (and most pragmatically), the temporal properties of the system may not be known to the scheduler, may be too complex, and may significantly vary in time (for instance, we do not know at the time of scheduling how much a patient will deviate from the prescribed care path). Our scheduler, just like an average chess player, is concerned only with the current move, and cannot think ten moves into the future.

Consequently, we need to assume that at least at the time of decision-making, (1) all tasks have equal priority, (2) all open slots have equal availability, and (3) all tasks can be performed during their scheduled times. The first two assumptions follow from the fact that we ignore the temporal flow of the tasks. The third, although looking the most restrictive, reflects the underlying assumption of feasible scheduling – otherwise, the schedule cannot be followed, and the process collapses to walk-in queueing, so elaborated in the classical queueing theory.

Finally, the assumption of equal slot availability leads to assuming that all N slots have the same fixed duration of $1/N$ time units (considering the duration of the entire schedule as a single time unit). This is another assumption often rejected in complex scheduling models, trying to fine-tune slot durations to specific task processing patterns. However, having same-size slots is critical to make them interchangeable – which is much more important practically, than overoptimized slot durations. Thus, unlike the "pure optimization" methods, we deliberately sacrifice optimality to achieve more realistic scheduling – which is an absolute must, if we want to implement our solution. For example, note that same-size slots are dominating healthcare scheduling.

Then what does our scheduler know? In complex, varying workflows we can count on only one bit of ground truth: our scheduler does know, how many slots K have been already taken. This means that the "one more task" acceptance will be decided by this number (or equivalently, by the number of open slots N-K). That is, the task-accepting decision becomes a function of the schedule utilization $u=K/N$, defined as the fraction of currently taken slots.

Thus, considering discrete task scheduling problem from the point of real-life decision-making, we state two principal questions that must be answered by any pragmatic scheduling approach:

- Investigating how discrete scheduling processing efficiency depends on the schedule utilization $u=K/N$, and based on this

- Determining the maximum utilization value, which should not be exceeded to avoid significant system overload

In the following sections, we solve these two problems by developing mathematical equations for discrete scheduling systems, deriving their analytical solutions, and using these solutions to propose an upper bound on optimal system utilization.

3 Discrete Scheduling Equations

3.1 Scheduling metrics

To decide on adding "one more task" to a busy schedule, we consider a discrete schedule with N time slots, where $K \leq N$ slots have been already allocated to previously assigned tasks, resulting in the current system utilization of $u=K/N$. At this point, a new $(K+1)^{st}$ task presents at a random time and needs to be scheduled in the remaining open slots.

Lacking any other knowledge on system properties or scheduler's strategies, we choose to study the best, most optimistic task assignment scenario – assigning the new task to the nearest open slot. Note that we assume this only for the current, $(K+1)^{st}$ task. That is, we want to know how well the system will perform if, running at utilization u, it schedules one more task in the most time-efficient way (Figure 2).

Applying the optimal strategy only to one single task presents another significant difference from the previous research, but derives from the practical scheduling experience as well. In real life, what makes scheduling decisions most challenging is the need to accommodate new, unexpected tasks. For example, in a healthcare schedule most appointments might be allocated well in advance, with least disruption. It is the addition of one more unexpected patient that will require decision-making, and can escalate processing delays (the infamous "last straw" challenge).

To measure the feasibility and efficiency of this "one more task" assignment, we define two principal metrics:

- *Overload probability* $P_{ov}(N,K)$ – the probability of not finding any open slot, when all slots after the new task arrival are already taken, and

- *Wait time* $W(N,K)$ – the time that the new task will have to wait for the next open slot

Both metrics can be viewed as scheduling cost functions, expressing scheduling efficiency as a function of utilization. While using the wait time metric is very typical for queuing and scheduling problems, including overload probability emphasizes the time-independent, discrete nature of slot assignment, outlined earlier. In either case, we want to investigate the dependency of these two principal metrics on the current system utilization $u=K/N$, and use this to define the critical system utilization value u_c, corresponding to the most rapid escalation of system bottlenecking.

3.2 Discrete scheduling equations

In discrete scheduling with equally-sized time slots, it is natural to measure time in slot durations. Assuming that $K \leq N$ slots in the schedule are currently taken, let ω_i denote the event when a new arriving task had to wait for i busy slots ($i \leq K$), and let $P(\omega_i)$ denote the probability of this event.

For a task to wait for i slots, the task must either (1) arrive into a system where there are i occupied time slots immediately after arrival and the $(i+1)^{st}$ time slot is unoccupied, or (2) arrive into a system with exactly i time slots remaining, all of which are occupied (corresponding to the system overload event). In the second case the new task cannot be taken and will have to wait at least i slot units until the schedule ends, and new scheduled slots could be potentially allocated. We do not know, whether this can be done, and working overtime may not be possible at all. But regardless of this, adding the new task right after the scheduled time simply reflects the same optimistic assumption of processing the $(K+1)^{st}$ task as early as possible, to estimate the lower bound on the costs. Note that this lower bound is also realistic – for example, in many healthcare workflows arriving patients are never turned down, and will be taken after all current patients are processed (Zacharias & Yunes, Multimodularity in the Stochastic Appointment Scheduling, 2019).

Using this model, we can derive the equations for expected task wait $W(N,K)$ and schedule overload probability $P_{ov}(N,K)$. In the first case, when the new task is scheduled to the $(i+1)^{st}$ time slot, the task should arrive with at least $(i+1)$ time slots remaining in the schedule. This is possible if and only if the task arrives anywhere during the first j slots ("earlier slots" in Figure 2), such that $j+(i+1) \leq N$, or $j \leq N-(i+1)$. With total slot count of N, the probability of arrival within the first $N-(i+1)$ slots is $\frac{N-(i+1)}{N}$.

Conditioning on this, one needs to find the probabilities of each of the next i time slots being occupied. Since we know that the total number of occupied slots is K, the probability that the time slot immediately after arrival is occupied is $\frac{K}{N}$, the one after it $\frac{K-1}{N-1}$, and so on until we reach $\frac{K-(i-1)}{N-(i-1)}$. The probability that the $(i+1)^{st}$ time slot is open is then given by $\frac{N-K}{N-i}$, since there are $(N\text{-}K)$ as of yet unidentified free slots in the schedule, and $(N\text{-}i)$ remaining possible locations for these slots in the schedule. By independence, we multiply all these terms together, giving us the first term in (Eq. 1). Similarly, in the second case, the task arrives with exactly i time slots remaining, which occurs with probability $\frac{1}{N}$. Conditioning on this, we have the same term denoting the probability that all remaining i timeslots are all occupied, and there are no more slots left. This gives us the second "overload" term in (Eq. 1).

$$P(\omega_i) = \left(\frac{N-(i+1)}{N}\right)\left(\frac{K}{N}\cdot\frac{K-1}{N-1}\cdot\ldots\cdot\frac{K-(i-1)}{N-(i-1)}\right)\left(\frac{N-K}{N-i}\right) + \frac{1}{N}\left(\frac{K}{N}\cdot\frac{K-1}{N-1}\cdot\ldots\cdot\frac{K-(i-1)}{N-(i-1)}\right) \quad \textit{(Eq. 1)}$$

Factoring and simplifying, we have

$$P(\omega_i) = \left(\frac{K\cdot K-1\cdot\ldots\cdot K-(i-1)}{N\cdot N-1\cdot\ldots\cdot N-(i-1)}\right)\left(\frac{N-1-i}{N}\frac{N-K}{N-i}+\frac{1}{N}\right) =$$

$$= \frac{1}{N}\left(\frac{K!}{(K-i)!}\frac{(N-i)!}{N!}\right)\left(\frac{(N-1-i)(N-K)}{N-i}+1\right) \quad \textit{(Eq. 2)}$$

Consequently, the total probability of schedule overload is computed as the sum of all overload probabilities over possible slot utilization values of K (second terms in (Eq. 1)):

$$P_{ov}(N,K) = \sum_{i=1}^{K}\frac{1}{N}\left(\frac{K!}{(K-i)!}\frac{(N-i)!}{N!}\right) = \frac{1}{N}\frac{K!}{N!}\sum_{i=1}^{K}\frac{(N-i)!}{(K-i)!} \quad , \quad \textit{(Eq. 3)}$$

and the expected wait time for the new task is found as

$$W(N,K) = \sum_{i=0}^{K} P(\omega_i) \cdot \frac{i}{N} = \frac{1}{N^2}\frac{K!}{N!}\sum_{i=0}^{K}\left(\frac{(N-1-i)(N-K)}{N-i}+1\right)\frac{(N-i)!}{(K-i)!}\, i, \qquad \textit{(Eq. 4)}$$

In the Appendix (Theorem 1), we prove that the summations in (Eq. 3) and (Eq. 4) can be reduced to a much shorter form:

$$P_{ov}(N,K) = \frac{1}{N}\frac{K}{N-K+1} = \frac{1}{N}\frac{u}{1+\frac{1}{N}-u} = P_{ov}(u), \quad u = \frac{K}{N}, \quad 0 \le u \le 1 \qquad \textit{(Eq. 5)}$$

$$W(N,K) = \frac{K(N^2+N-KN-1)}{N^2(N-K+1)(N-K+2)} = \frac{1}{N}\frac{u(1+\frac{1}{N}+\frac{1}{N^2}-u)}{(1+\frac{1}{N}-u)(1+\frac{2}{N}-u)} = W(u) \qquad \textit{(Eq. 6)}$$

$$W(u) \xrightarrow[N\to\infty]{} W_a(u) = \frac{1}{N}\frac{u}{(1+\frac{2}{N}-u)}$$

This provides us with a very concise set of equations, describing two principal metrics of discrete scheduling. Moreover, this important result enables us to express $W(u)$ and $P_{ov}(u)$ as functions of system utilization u. Although in discrete scheduling u can take only selected rational values $u=K/N$, our equations enable us to study $W(u)$ and $P_{ov}(u)$ as continuous functions of utilization u, to better understand their trends and behavior.

The equations, illustrated by plots for $W(u)$ and $P_{ov}(u)$ in Figure 3, lead to a few important observations.

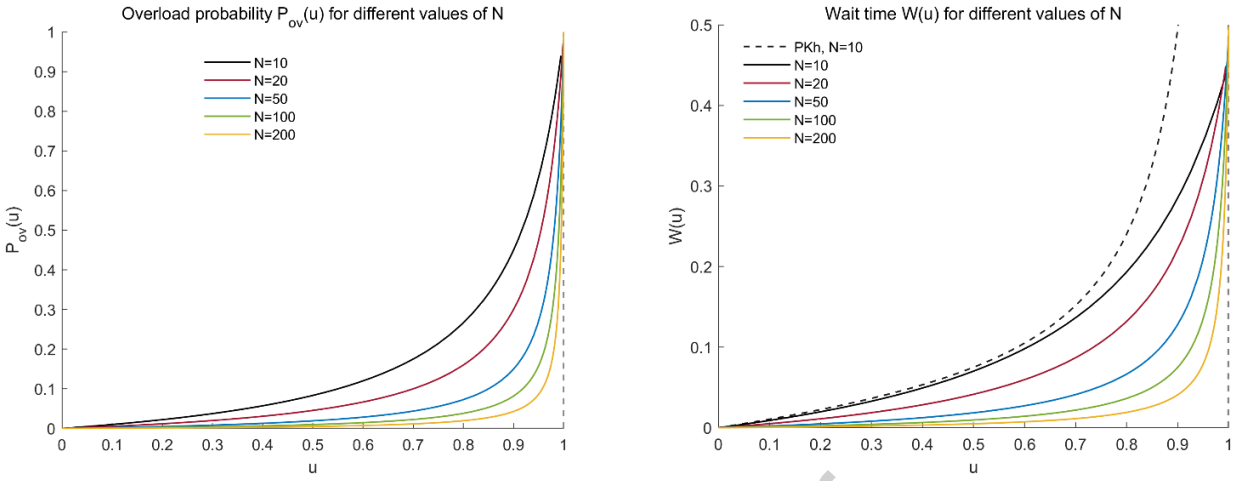

Figure 3: Plots of schedule overload probability $P_{ov}(u)$ and expected task wait $W(u)$ as functions of the system utilization u, for several schedule sizes N. Note that both cost functions are finite at u=1: $P_{ov}(1)=1$, and $W(1)=1/2+1/(2N)$. The dashed curve in $W(u)$ plot represents Pollaczek–Khinchine formula $W_{PKh}(u)$ for $N=10$, diverging to infinity as u approaches to 1.

First, both $W(u)$ and $P_{ov}(u)$ behave as hyperbolas (see Corollary 1 and 2 in the Appendix), starting nearly flat at u=0, but becoming increasingly vertical as u approaches to 1. This can be seen by considering the first derivatives of these functions w.r.t utilization u:

$$P'_{ov}(0) = \frac{1}{N+1}, \quad P'_{ov}(1) = N+1,$$

$$W'(0) = \frac{N^2+N-1}{N(N+1)(N+2)} \xrightarrow[N\to\infty]{} \frac{1}{N}, \qquad W'(1) = \frac{N^2-N-2}{4N} \xrightarrow[N\to\infty]{} \frac{N}{4}, \qquad \textit{(Eq. 7)}$$

Note that this theoretical hyperbolic behavior matches the trends observed in many real-life data – such as those presented in Figure 1. Furthermore, as (Eq. 7) suggests, hyperbolic behavior becomes particularly pronounced as the number of slots N increases. In practical applications this means that discrete scheduling systems, steady at low values of u, become progressively unstable as u approaches to 1 (100%). This agrees with real-life experiences, when overcrowded processes break down not only because they lead to escalating wait time, but also because finding an empty slot for a new task

becomes highly improbable (overload probability $P_{ov}(u{=}1){=}1$) – which often leads to increased errors, stress and overburn, so visible in the human-driven environments.

Second, it is instructive to compare our equations to the classical queueing theory Pollaczek–Khinchine formula $W_{PKh}(u) = \frac{1}{\lambda}\left(u + \frac{u^2+\lambda^2 V}{2(1-u)}\right)$, where λ denotes the task arrival rate, and V - service time variance. In our case of tasks fitting into the scheduled slots, we can assume $V{=}0$, $\lambda{=}N$, leading to

$$W_{PKh}(u) = \frac{1}{\lambda}\left(u + \frac{u^2+\lambda^2 V}{2(1-u)}\right) = \frac{1}{N}\frac{u\left(1-\frac{1}{2}u\right)}{1-u}\ , \qquad \textit{(Eq. 8)}$$

bearing visible similarities with our $W(u)$ and $W_a(u)$ in (Eq. 6). However, although all these functions have a hyperbolic trend w.r.t utilization u, discrete scheduling wait $W(u)$ remains finite even when utilization u approaches to 100% (see $W_{PKh}(u)$ plot in Figure 3):

$$W(N, K=N) = W(u=1) = \frac{1}{2} + \frac{1}{2N},$$

thus asymptotically converging to ½ for large N (task arriving to a fully-scheduled system will have to wait ½ day on average to get processed at the end). Undoubtedly, non-diverging, finite wait offers a more realistic reflection of practical scheduling experience (Figure 1).

4 Critical utilization

4.1 Definition and equations

Critical utilization can be defined as the “last straw” utilization limit, when the cost of accepting one more task becomes prohibitively high. Although hard to formalize theoretically, finding the most appropriate critical utilization value presents one of the most central challenges in real-life operations management. For example, one would always want to utilize the most expensive resource as fully as possible, but avoid overcrowding and stress associated with overloads. Therefore, while it is clear that approaching 100% utilization leads to fragile workflows with no cushions to absorb new work, no solution has been offered to determine what exact threshold under 100% should not be exceeded.

As a result, utilization threshold is often estimated by various external indicators of system overload (overcrowded waiting rooms, long waiting times, dropped calls, stress, and so on). However, these indicators call for their own thresholds, equally elusive, and failing to generalize to a universal principle. This challenge is only augmented by the hyperbolic nature of the principal utilization metrics that we have discovered with (Eq. 5) and (Eq. 6). Sharing the same hyperbolic behavior, these functions and all their derivatives are monotonically increasing with utilization u, perfectly reflecting escalating system instability, but suggesting no specific point of "breakage".

However, there is one particular property of hyperbolas which serves to our advantage – as can be seen in Figure 3, especially for larger values of N. Starting nearly horizontally at $u=0$, hyperbolas become more and more vertical as u approaches 1 (see (Eq. 7)). This L-shaped behavior is captured with a very specific "turning point" of the sharpest change from the horizontal to the vertical trend. Mathematically, this sharpest turn occurs when the curvature of the hyperbolic function reaches its maximum.

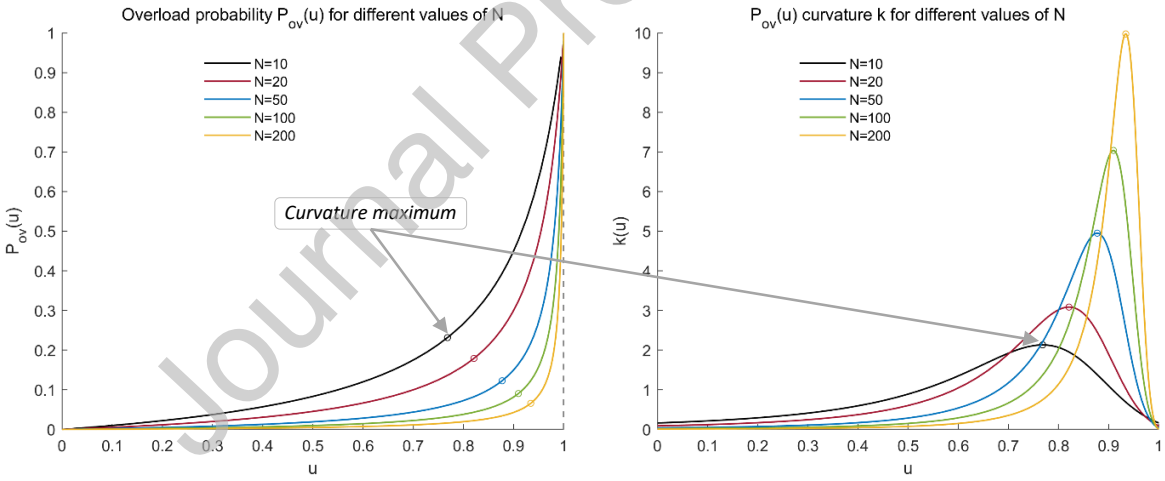

Figure 4: Left: Overload probability $P_{ov}(u)$ with maximum curvature points. Right: Corresponding curvature functions. Note that as schedule size N increases, curvature peak becomes sharper and closer to 1.

In practical scheduling this means that accepting one more task at this point will produce the sharpest turn into the escalating, "vertical" trend, which is exactly what any scheduler needs to avoid. Therefore, we define *critical utilization u_c as the point of the maximum curvature in the utilization metric function* –

when the metric and its underlying process experience the most abrupt shift into the overloaded pattern (Figure 4).

Curvature analysis provides a universal mathematical framework, independent of the specific problem and overload definitions. Moreover, this approach enabled us to derive the exact critical utilization solutions for the two major metrics used in this study: $P_{ov}(u)$ and $W_a(u)$ (asymptotic form of $W(u)$), as we prove in the Appendix (Corollary 3):

$$u_c = 1 + \frac{2}{N} - \sqrt{\frac{1}{N} + \frac{2}{N^2}} = 1 - \frac{1}{\sqrt{N}} + \frac{2}{N} - \frac{1}{N\sqrt{N}} + O\left(\frac{1}{N^2\sqrt{N}}\right) = u_c^{sqrt} + O\left(\frac{1}{N}\right) \quad \textit{(Eq. 9)}$$

$$u_c^{ov} = 1 + \frac{1}{N} - \sqrt{\frac{1}{N} + \frac{1}{N^2}} = 1 - \frac{1}{\sqrt{N}} + \frac{1}{N} - \frac{1}{2N\sqrt{N}} + O\left(\frac{1}{N^2\sqrt{N}}\right) = u_c^{sqrt} + O\left(\frac{1}{N}\right) \quad \textit{(Eq. 10)}$$

$$u_c^{sqrt} = 1 - \frac{1}{\sqrt{N}}$$

Both critical utilization functions u_c^{ov} and u_c are shown in Figure 5, which also shows their asymptotic approximation u_c^{sqrt}. Figure 5 also includes the u_c^{true} curve – the curvature of the original $W(u)$ function. Direct application of curvature math to the $W(u)$ in (Eq. 6) leads to a 12$^{\text{th}}$ degree equation, with no analytically-tractable solution – the problem we overcame by introducing the $W_a(u)$ function as an asymptotic approximation to $W(u)$ (see Corollary 2 in the Appendix). Yet interestingly enough, the analytical solution for the critical curvature u_c, that we have found from $W_a(u)$, is virtually indistinguishable from the true solution u_c^{true}, found numerically (see Figure 5). Consequently, the u_c formula from (Eq. 9) can be used as a very accurate substitute for u_c^{true}, at least for $N \geq 10$ shown in Figure 5.

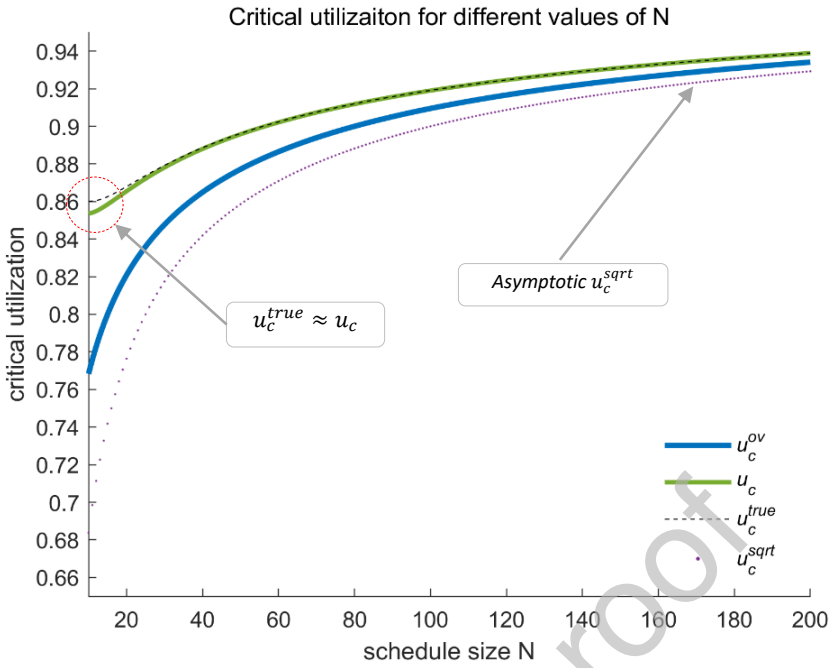

Figure 5: Discrete schedule critical utilization as function of the schedule size N. The u_c^{true} curve shows maximum curvature of the original $W(u)$ function, which has no analytical solution, and was found numerically. $u_c^{sqrt} = 1 - \frac{1}{\sqrt{N}}$ corresponds to the simple approximation, to which u_c^{true}, u_c , and u_c^{ov} converge asymptotically.

One can also notice that both critical utilization solutions – u_c^{ov} for $P_{ov}(u)$ and u_c for $W_a(u)$ – have very similar expressions, coming only $1/N$ (one slot size) apart from each other ((Eq. 9), (Eq. 10)). This means that despite two conceptually-different choices of scheduling metrics – time-based $W(u)$ and availability-based $P_{ov}(u)$ – they identify the same point of the scheduling process breakdown, a very unexpected and interesting result.

Second, at the same "one slot" margin of error, both critical utilization values in (Eq. 9) and (Eq. 10) converge to $u_c^{\mathrm{sqrt}} = 1 - \frac{1}{\sqrt{\mathrm{N}}}$. Since we defined utilization as the fraction of busy slots (u=K/N), this leads to a very simple estimate of the critical busy slot count K_c:

$$K_c = Nu_c = N\left(1 - \frac{1}{\sqrt{\mathrm{N}}}\right) = \ N - \sqrt{N}$$

As Figure 5 demonstrates, $u_c^{\mathrm{sqrt}} = 1 - \frac{1}{\sqrt{\mathrm{N}}}$ slightly underestimates u_c^{ov} for $P_{ov}(u)$ and u_c for $W_a(u)$, thus setting a "safe" threshold on utilization, not to be exceeded. This short and elegant result – "keep at least $\sqrt{N}$ slots open to avoid schedule breakdown" - provides a concise, universal, and practical guidance

for managing discrete schedules of any size N. Moreover, using the time-based interpretation provided by the $W(u)$ function, we can rephrase this result as "keep at least $\sqrt{N}$ slot time open to avoid schedule breakdown" – which becomes more applicable for the processes where utilization is measured in the resource busy time, rather than slots. This "time-availability equivalence" of the critical utilization value can be seen as one of the most fundamental properties of discrete scheduling, independent of the scheduling metric.

Finally, as schedule size N increases, critical utilization u_c approaches to 1, which means that schedules with shorter slots are generally more immune to overloads. This can be intuitively true – "one more task" is easier to accept when the task duration is getting shorter – but has its own disadvantage, very visible in Figure 4 and (Eq. 7). Large values of N result in sharper curvature change, and more vertical escalation of delays after exceeding the critical u_c point. Thus, going beyond critical utilization for large schedule sizes N will have more devastating effects on the process stability and operations.

4.2 Extension to queueing theory

The idea of using maximum curvature to identify the point of critical change is general enough to be applied to any process with L-shaped cost function (convex and monotonically increasing in our case). Moreover, recognizing inherent similarities between discrete scheduling and queueing theory equations, it becomes natural to apply the same concept to find the critical utilization of the latter.

To do so, we consider the original Pollaczek–Khinchine formula (first equation in (Eq. 8)). Although developed for a very different case (M/G/1 "walk in" queue with Poisson arrivals), Pollaczek–Khinchine formula produces a similar hyperbolic wait time trend, and leads to the same practical question of determining the critical utilization value. We were still able to derive the exact expression for the Pollaczek–Khinchine critical utilization as well, as proven in the Appendix (see Corollary 4, (Eq. 28)).

$$u_c^{PKh} = 1 - \sqrt[4]{\frac{(1+\lambda^2 V)^2}{1+4\lambda^2}} \qquad \textit{(Eq. 11)}$$

First, it is interesting to observe, that queueing critical utilization u_c^{PKh} decreases as processing time variability V grows. Not only this supports a well-known observation that high processing variability

"kills" any organized processing, but now provides a numerical way to control for the variability value, to ensure safe utilization limits.

Second, using Pollaczek–Khinchine result to approximate our discrete scheduling by setting λ=N, V=0, yields

$$u_c^{PKh} = 1 - \sqrt[4]{\frac{1}{1+4N^2}} = 1 - \frac{1}{\sqrt[4]{4}\sqrt{N}} + \frac{1}{16\sqrt[4]{4}N^2\sqrt{N}} + O\left(\frac{1}{N^4\sqrt{N}}\right) \qquad \textit{(Eq. 12)}$$

That is, as N increases, critical utilization u_c^{PKh} for the queued task wait tends to 1 as $1 - \frac{1}{\sqrt[4]{4}\sqrt{N}}$, which is very similar, but faster compared to $u_c = 1 - \frac{1}{\sqrt{N}}$ in the case of discrete scheduling ((Eq. 9), (Eq. 10), Figure 5).

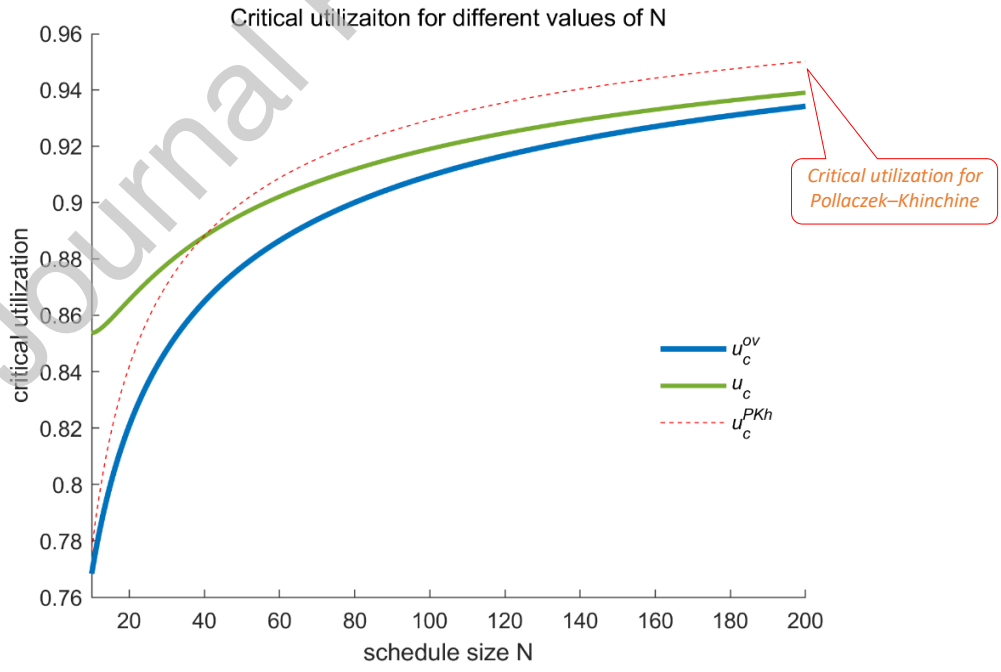

Figure 6: Comparing critical utilization u_c^{PKh} found for the queueing theory wait function (Pollaczek–Khinchine equation, dashed line) to our critical utilization for discrete scheduling wait u_c .

Higher critical utilization values found in queueing, $u_c^{PKh} > u_c$ nicely correspond to our intuition: in general, queueing is a more robust process compared to discrete scheduling. In the case of discrete scheduling, tasks have to be fitted into slots, or some of the server future slots may be already assigned, thus reducing overall processing capacity, and making the discrete systems easier to overwhelm. However, discrete scheduling is a more predictable and "satisfying" form of processing from the task (customer) perspective, when it is much more appreciated to get service at a guaranteed time, rather than wait for it in a line. This certainly justifies widespread use of scheduling in real-life operations.

5 Limitations

Our "one more task" approach was developed by deliberately ignoring any temporal pattern or strategy in scheduling the previous K tasks – otherwise, if these tasks were allocated in some optimal way, this could have made the scheduling of the next $(K+1)^{st}$ task more possible and more optimal. However, more optimal scheduling can be achieved only when (1) more knowledge about the process properties is readily available, (2) this knowledge does not change, (3) one has enough mathematical expertise and computing power to process this information with a model, and (4) the resulting optimal schedule can be executed with minimal disruptions. Unfortunately, none of the above can be guaranteed in most real-life scheduling processes, healthcare included.

Consequently, we had to consider the lower (optimistic) bound on the scheduling costs W and P_{ov}, meaning that our critical utilization formula corresponds to the upper (optimistic) bound. This implies that in reality system bottlenecking can develop even for the lower utilization values, and our upper bound should be treated as the value that should not be exceeded.

We also had to assume that our schedule is feasible (tasks fit into slots), and the slots are equally-sized – borrowing the latter from healthcare scheduling. Although this does not invalidate our approach and critical utilization definition, our principal equations will need to be adjusted to reflect a specific slot time distribution. Still, we have to note that schedules with different slot times are less common, especially in processes with scheduling alterations: different slot sizes make swapping task assignments

impossible. The ability to have modifiable schedules is essential for many practical areas (such as healthcare, where patients often need to cancel or reschedule their appointments), which calls for same-slot-size schedules, making them very ubiquitous (Rosenthal & Pianykh, 2021).

Finally, we used $\lambda=N$, $V=0$ as a simple means to reveal the intrinsic similarity between the classical queueing theory (Pollaczek–Khinchine result) and our discrete scheduling equations. This was done only to compare the two results using reasonably close settings. One should still bear in mind, that Pollaczek–Khinchine was derived under a different set of assumptions (such as Poisson task arrivals), and for a different processing model (queueing).

6 Conclusion

The problem of discrete schedules and their utilization limits represents the most important challenge in real-life applications, but has been given very little treatment in the previous operational research. Therefore, in our work we studied discrete scheduling through the lens of real-life scheduling decision-making, using mathematical analysis to discover practically-applicable scheduling decision rules.

The first major contribution of our analysis was in deriving closed-form analytical expressions for $P_{ov}(u)$ and $W_a(u)$, which in turn made possible the study of their hyperbolic behavior and limits. This work led to our second principal contribution: proposing a new definition for the critical system utilization threshold u_c, as the point at which system performance experiences the most abrupt change from the stable to the escalating pattern. As a result, we were able to solve for the exact analytical expression for u_c, and demonstrate its asymptotical $1-\frac{1}{\sqrt{N}}$ behavior. This yielded a reliable, easy-to-use practical estimate to control for scheduling system overload. Keeping $\sqrt{N}$ slots open provides a remarkably simple rule for many scheduling processes, be it a small physician office, or a large hospital with many inpatient beds, or a shared supercomputer CPU time, or seats in a large stadium.

Finally, the principal novelty of our work is largely based on our “one more task” approach, where we consider the reality of the scheduling decision-making process, which leads to a concise, closed-form analytical solution on maximum utilization threshold. This result has a very clear practical meaning and can be used to develop efficient scheduling guidelines. Knowing robust scheduling bounds and heuristics is critical in many fields, and can help with developing better operational strategies.

7 Acknowledgements

We would like to thank Drs. James Brink, Daniel Rosenthal, and Michael Gee from Massachusetts General Hospital, Department of Imaging, for their constant support in developing operational improvements, and finding the optimal workflow solutions.

8 Appendix: Proofs

In this appendix, we provide the formal proofs for all key results presented in this work.

Some proofs will be using the following geometric progression summation formulas:

$$\sum_{i=0}^{K} q^i i = \frac{1 - q^{K+1}}{1 - q} \qquad q \neq 1 \qquad \textit{(Eq. 13)}$$

$$\sum_{i=0}^{K} q^i i = \frac{Kq^{K+2} - (K+1)q^{K+1} + q}{(1-q)^2} \qquad \textit{(Eq. 14)}$$

$$\sum_{i=0}^{K} q^i i^2 = \frac{-K^2 q^{K+3} + (2K^2 + 2K - 1)q^{K+2} - (K+1)^2 q^{K+1} + q^2 + q}{(1-q)^3} \qquad \textit{(Eq. 15)}$$

The first sum represents a geometric progression, and the other two can be derived from it by taking first and second derivatives w.r.t progression factor q. Therefore, treating these results as well-known (see 0.112, 0.113, and 0.114 in (Gradshteĭn & Ryzhik, 2007)), we omit their proofs here.

We will also use a standard definition of the "polynomial coefficient" operator $[x^i]: g(x)$, returning the coefficient g_i for the power term x^i in polynomial (power series) $g(x)$:

$$[x^i]: g(x) = [x^i]: \left\{ \sum_{j=0} g_j x^j \right\} = \ g_i \qquad \textit{(Eq. 16)}$$

In particular, by definition of binomial coefficient,

$$[x^i]: (1+x)^m = \binom{m}{i} \qquad \textit{(Eq. 17)}$$

With these baseline results in mind, our proofs start with the following

Lemma 1 ("binomial summation"):

For any non-negative integer numbers N and K, $K \leq N$

$$S_0 = \sum_{i=1}^{K} \binom{N-i}{K-i} = \binom{N}{N-K+1} \qquad \textit{(Eq. 18)}$$

$$S_1 = \sum_{i=0}^{K} \binom{N-i}{K-i} i = \binom{N+1}{N-K+2} \qquad \textit{(Eq. 19)}$$

$$S_2 = \sum_{i=0}^{K} \binom{N-i}{K-i} i^2 = \binom{N+1}{N-K+3} + \binom{N+2}{N-K+3} \qquad \textit{(Eq. 20)}$$

Proof:

Considering the sum for any non-negative integer power p, and using the definition of binomial coefficient, we rewrite

$$S_p = \sum_{i=0}^{K} \binom{N-i}{K-i} i^p = \sum_{i=0}^{K} \binom{N-i}{N-K} i^p$$

Using (Eq. 17)

$$S_p = \sum_{i=0}^{K} \binom{N-i}{N-K} i^p = \sum_{i=0}^{K} [x^{N-K}]: (1+x)^{N-i}\, i^p = [x^{N-K}]: \left\{ (1+x)^N \sum_{i=0}^{K} (1+x)^{-i} i^p \right\}$$

Note that for the three choices of p=0, 1, 2 that we have to prove, the expression under summation directly corresponds to the sums in (Eq. 13), (Eq. 14), and (Eq. 15) for $q=(1+x)^{-1}$. For example, the most complex case of p=2 leads to

$$S_2 = \sum_{i=0}^{K} \binom{N-i}{N-K} i^2 = [x^{N-K}]: \left\{ (1+x)^N \sum_{i=0}^{K} (1+x)^{-i} i^2 \right\} = [x^{N-K}]:$$

$$\left\{ (1+x)^N \frac{\frac{-K^2}{(1+x)^{K+3}} + \frac{(2K^2+2K-1)}{(1+x)^{K+2}} - \frac{(K+1)^2}{(1+x)^{K+1}} + \frac{1}{(1+x)^2} + \frac{1}{(1+x)}}{\left(1-\frac{1}{(1+x)}\right)^3} \right\} =$$

$[x^{N-K}]$:

$$\left\{ \frac{(1+x)^N}{x^3} \left(\frac{-K^2}{(1+x)^K} + \frac{(2K^2+2K-1)}{(1+x)^{K-1}} - \frac{(K+1)^2}{(1+x)^{K-2}} + (1+x) + (1+x)^2 \right) \right\}$$

Moving x^3 under the coefficient operator, and observing that only the last two terms under the curly brackets have powers higher than N-K+3, we conclude:

$$\begin{aligned} S_2 = [x^{N-K+3}]: \{ & -K^2(1+x)^{N-K} + (2K^2+2K-1)(1+x)^{N-K+1} - (K+1)^2(1+x)^{N-K+2} \\ & + (1+x)^{N+1} + (1+x)^{N+2} \} = [x^{N-K+3}]: \{ (1+x)^{N+1} + (1+x)^{N+2} \} \\ & = \binom{N+1}{N-K+3} + \binom{N+2}{N-K+3} \end{aligned}$$

The equations for S_0 and S_1 are proven in the same exact way using (Eq. 13) and (Eq. 14) respectively, therefore we omit their derivations here for brevity.

q.e.d.

Theorem 1 ("discrete scheduling"):

The expected overload probability $P_{ov}(N,K)$ and wait time $W(N,K)$ for discrete schedule with N slots (K of which are busy) can be found as

$$P_{ov}(N,K) = \frac{1}{N}\frac{K!}{N!}\sum_{i=1}^{K}\frac{(N-i)!}{(K-i)!} = \frac{1}{N}\frac{K}{N-K+1}, \qquad \textit{(Eq. 21)}$$

$$W(N,K) = \frac{1}{N^2}\frac{K!}{N!}\sum_{i=0}^{K}\left(\frac{N-(i+1)}{N-i}(N-K)+1\right)\frac{(N-i)!}{(K-i)!}\,i = \frac{K(N^2+N-KN-1)}{N^2(N-K+1)(N-K+2)}, \qquad \textit{(Eq. 22)}$$

Proof:

Converting the factorials under the sum into a binomial coefficient for $P_{ov}(N,K)$ yields

$$P_{ov}(N,K) = \frac{1}{N}\frac{K!\,(N-K)!}{N!}\sum_{i=1}^{K}\binom{N-i}{K-i}$$

Using the first equation (Eq. 18) from Lemma 1

$$P_{ov}(N,K) = \frac{1}{N}\frac{K!\,(N-K)!}{N!}\sum_{i=1}^{K}\binom{N-i}{K-i} = \frac{1}{N}\frac{K!\,(N-K)!}{N!}\binom{N}{N-K+1}$$
$$= \frac{1}{N}\frac{K!\,(N-K)!}{N!}\frac{N!}{(N-K+1)!\,(K-1)!} = \frac{1}{N}\frac{K}{N-K+1}$$

thus proving the equation for $P_{ov}(N,K)$.

Following the same approach for $W(N,K)$:

$$W(N,K) = \frac{N-K}{N^2}\frac{K!}{N!}\sum_{i=0}^{K}\left(\frac{N-(i+1)}{N-i}(N-K)+1\right)\frac{(N-i)!}{(K-i)!}\ i$$

$$= \frac{N-K}{N^2}\frac{K!}{N!}\sum_{i=0}^{K}\frac{(N-1-i)(N-K)+N-i}{N-i}\frac{(N-i)!}{(K-i)!}\ i =$$

$$= \frac{N-K}{N^2}\frac{K!}{N!}\sum_{i=0}^{K}\big((N-1-i)(N-K)+N-i\big)\frac{(N-1-i)!}{(K-i)!}\ i =$$

$$= \frac{N-K}{N^2}\frac{K!\,(N-K-1)!}{N!}\sum_{i=0}^{K}\big((N-1-i)(N-K)+N-i\big)i\ \binom{N-1-i}{K-i}$$

Simplifying the first factor under the summation as

$$(N-1-i)(N-K)+N-i = N^2-NK+K-i(N-K+1)$$

we rewrite $W(N,K)$ as

$$W(N,K) = \frac{N-K}{N^2}\frac{K!\,(N-K-1)!}{N!}$$
$$\times\left\{(N^2-NK+K)\sum_{i=0}^{K}\binom{N-1-i}{K-i}i\ -(N-K+1)\sum_{i=0}^{K}\binom{N-1-i}{K-i}i^2\right\}$$

The two summations in this formula can be computed using our equations from Lemma 1 (replacing N by N-1), leading to

$$W(N,K) = \frac{N-K}{N^2}\frac{K!\,(N-K-1)!}{N!}$$
$$\times\left\{(N^2-NK+K)\binom{N}{N-K+1}-(N-K+1)\left(\binom{N+1}{N-K+3}+\binom{N+2}{N-K+3}\right)\right\}$$

Now, similarly to $P_{ov}(N,K)$, we expand binomial coefficients into factorials, and reduce the factorials arriving to the final result:

$$W(N,K) = \frac{1}{N^2}\frac{K!}{N!}\sum_{i=0}^{K}\left(\frac{N-(i+1)}{N-i}(N-K)+1\right)\frac{(N-i)!}{(K-i)!}\; i = \frac{K(N^2+N-KN-1)}{N^2(N-K+1)(N-K+2)}$$

q.e.d.

Corollary 1 ("utilization equations"):

Rewriting (Eq. 21) and (Eq. 22) as functions of the system utilization u=K/N results in

$$P_{ov}(u) = \frac{1}{N}\frac{u}{1+\frac{1}{N}-u} = \frac{\varepsilon u}{1+\varepsilon-u}, \qquad \varepsilon = \frac{1}{N} \in (0,1], \quad u \in [0,1], \qquad \textit{(Eq. 23)}$$

$$W(u) = \frac{1}{N}\frac{u}{1+\frac{1}{N}-u} = \frac{1}{N}\frac{u(1+\frac{1}{N}+\frac{1}{N^2}-u)}{(1+\frac{1}{N}-u)(1+\frac{2}{N}-u)} = \frac{\varepsilon u(1+\varepsilon+\varepsilon^2-u)}{(1+\varepsilon-u)(1+2\varepsilon-u)} \qquad \textit{(Eq. 24)}$$

In particular, note that $P_{ov}(0)=0$ (no overload for completely open schedule), and $P_{ov}(1)=1$ (full overload for schedule with no open slots), which certainly corresponds to our intuition.

Now we demonstrate that the formula for $W(N,K)$ can be efficiently approximated with a more simple expression, to which it also converges for the large values of N:

Corollary 2 ("asymptotic W"):

Rewriting (Eq. 21) and (Eq. 22) as functions of the system utilization $u=K/N$ results in

$$W(u) = \frac{\varepsilon u(1+\varepsilon+\varepsilon^2-u)}{(1+\varepsilon-u)(1+2\varepsilon-u)} \quad \xrightarrow[N\to\infty]{} \quad W_a(u) = \frac{\varepsilon u}{1+2\varepsilon-u}, \qquad \varepsilon = \frac{1}{N} \qquad \textit{(Eq. 25)}$$

$$|W(u) - W_a(u)| \leq \frac{1}{2N} \quad \text{for} \quad u \in [0,1]$$

Proof:

As N increases, ε tends to 0, therefore one can neglect the highest order ε^2 term in the numerator, which reduces $W(u)$ to the expression for $W_a(u)$. Note that $W(u) = W_a(u)\left(1 + \frac{\varepsilon^2}{(1+\varepsilon-u)}\right)$, where $\frac{\varepsilon^2}{(1+\varepsilon-u)} \leq \varepsilon = \frac{1}{N}$ for any $u \in [0,1]$. Since $W_a(u) \leq W_a(1) = \frac{1}{2}$, this leads to $|W(u) - W_a(u)| \leq \frac{1}{2N}$.

q.e.d.

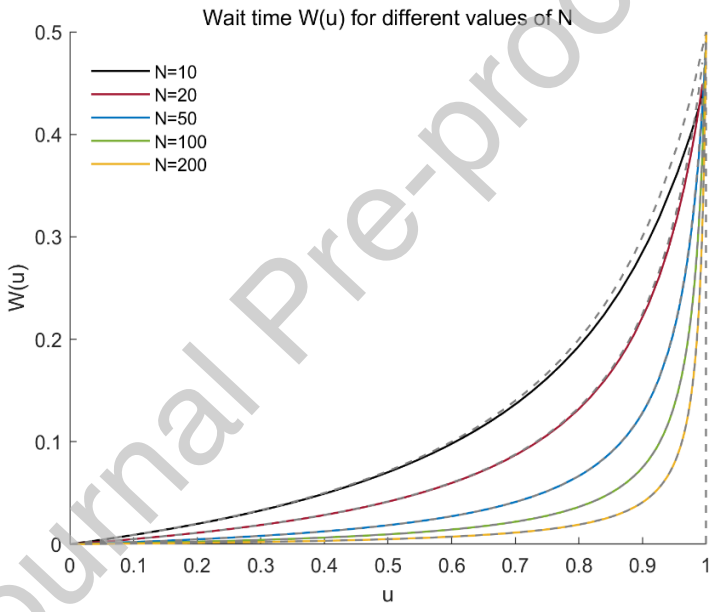

Figure 7: Comparing $W(u)$ (solid lines) and its asymptotic approximation $W_a(u)$ (dashed lines) for different schedule sizes N. As one can see, for N beyond 10 the two curves become practically identical.

As a result, one can use $W_a(u)$ as a very tight approximation to the original $W(u)$ function (Figure 7); and both overload probability $P_{ov}(u)$ and wait time $W(u)$ can be seen as having hyperbolic behavior w.r.t. system utilization u.

To obtain analytical solutions for the critical curvature value, we need to solve for the curvature extrema points, which we accomplish with the following lemma:

Lemma 2 ("hyperbolic curvature"):

Function $f(u) = \frac{a}{b-u} + d(1-u),\ \ a, b > 0,\ u < b$ has curvature

$$k(u) = \frac{2a(b-u)^3}{[(-d(b-u)^2 + a)^2 + (b-u)^4]^{3/2}}$$

which attains its maximum value at $u_c = b - \sqrt[4]{\frac{a^2}{d^2+1}}$

Proof:

Using the definition of curvature $k(u) = \frac{f''}{(1+(f')^2)^{3/2}}$, we compute the derivatives:

$f'(u) = \frac{a}{(b-u)^2} - d, \quad f''(u) = \frac{2a}{(b-u)^3}$

leading to

$$k(u) = \frac{f''}{(1+(f')^2)^{\frac{3}{2}}} = \frac{\frac{2a}{(b-u)^3}}{\left(1 + \left(\frac{-d(b-u)^2 + a}{(b-u)^2}\right)^2\right)^{3/2}} = \frac{2a(b-u)^3}{((-d(b-u)^2 + a)^2 + (b-u)^4)^{3/2}}$$

$$= \frac{2at^3}{((-dt^2 + a)^2 + t^4)^{3/2}} = k(t), \qquad t = b - u > 0$$

To find the maximum point, we set the derivative $k'(t) = 0$, yielding the following equation:

$$6at^2[(-dt^2 + a)^2 + t^4]^{-3/2} = \frac{5}{2} 2at^3[2(-dt^2 + a)(-2dt) + 4t^3][(-dt^2 + a)^2 + t^4]^{-5/2}$$

which for t>0 simplifies to

$t^4(d^2 + 1) = a^2,\ \ t = b - u > 0,$

yielding the only possible solution for $u < b$

$t_c = \sqrt[4]{\frac{a^2}{d^2+1}} = b - u_c,$

Under the assumption $a, b > 0, u < b$ this corresponds to the maximum point of $k(u)$.

q.e.d

Corollary 3 ("critical utilization in discrete scheduling"):

Critical utilization values for overload probability and asymptotic wait in discrete scheduling are

$$u_c = 1 + \frac{2}{N} - \sqrt{\frac{1}{N} + \frac{2}{N^2}} = 1 - \frac{1}{\sqrt{N}} + \frac{2}{N} - \frac{1}{N\sqrt{N}} + O\left(\frac{1}{N^2\sqrt{N}}\right) = 1 - \frac{1}{\sqrt{N}} + O\left(\frac{1}{N}\right) \quad \textit{(Eq. 26)}$$

$$u_c^{ov} = 1 + \frac{1}{N} - \sqrt{\frac{1}{N} + \frac{1}{N^2}} = 1 - \frac{1}{\sqrt{N}} + \frac{1}{N} - \frac{1}{2N\sqrt{N}} + O\left(\frac{1}{N^2\sqrt{N}}\right)$$
$$= 1 - \frac{1}{\sqrt{N}} + O\left(\frac{1}{N}\right) \quad \textit{(Eq. 27)}$$

Proof:

The proof follows directly from our definition of critical utilization and Lemma 2, since in the case of asymptotic $W_a(u)$ we have

$$W_a(u) = \frac{1}{N}\frac{u}{1+\frac{2}{N}-u} = -\frac{1}{N} + \frac{\frac{1}{N}\left(1+\frac{2}{N}\right)}{1+\frac{2}{N}-u}$$

The first constant term has no effect on curvature, and the second corresponds to

$$a = \frac{1}{N}\left(1+\frac{2}{N}\right) = \frac{1}{N} + \frac{2}{N^2}, \qquad b = 1+\frac{2}{N}, \quad d = 0$$

in Lemma 2, yielding

$$u_c = 1 + \frac{2}{N} - \sqrt{\frac{1}{N}+\frac{2}{N^2}} = 1 + \frac{2}{N} - \frac{1}{\sqrt{N}}\left(1+\frac{2}{N}\right)^{\frac{1}{2}} = 1 + \frac{2}{N} - \frac{1}{\sqrt{N}}\left\{1 + \frac{1}{N} + O\left(\frac{1}{N^2}\right)\right\}$$
$$= 1 - \frac{1}{\sqrt{N}} + \frac{2}{N} - \frac{1}{N\sqrt{N}} + O\left(\frac{1}{N^2\sqrt{N}}\right)$$

The result for $P_{ov}(u)$ is proven in the same way.

q.e.d.

It is interesting to observe that both critical utilizations, u_c^{ov} and u_c , although derived from two different cost functions, result in very similar values, converging asymptotically to $1 - \frac{1}{\sqrt{N}}$.

Corollary 4 ("critical utilization of queueing"):

Critical utilization of Pollaczek–Khinchine formula

$$W_{PKh}(u) = \frac{1}{\lambda}\left(u + \frac{u^2+\lambda^2 V}{2(1-u)}\right)$$

can be found as $u_c^{PKh} = 1 - \sqrt[4]{\frac{(1+\lambda^2 V)^2}{1+4\lambda^2}}$

Proof:

$$W_{PKh}(u) = \frac{1}{\lambda}\left(u + \frac{u^2 + \lambda^2 V}{2(1-u)}\right) = -\frac{1}{2\lambda}\,(1-u) + \frac{1 + \lambda^2 V}{2\lambda(1-u)}$$

Therefore, we can apply Lemma 2, using

$$a = \frac{1 + \lambda^2 V}{2\lambda},\;\; b = 1,\;\; d = -\frac{1}{2\lambda}$$

resulting in

$$u_c^{PKh} = 1 - \sqrt[4]{\frac{a^2}{d^2+1}} = 1 - \sqrt[4]{\frac{(1+\lambda^2 V)^2}{1+4\lambda^2}} = 1 - \frac{\sqrt{1+\lambda^2 V}}{\sqrt[4]{1+4\lambda^2}} \qquad \textit{(Eq. 28)}$$

q.e.d

In particular, setting parameters V=0, λ=N as an approximation to our discrete scheduling scenario leads to

$$u_c^{PKh} = 1 - \sqrt[4]{\frac{1}{1+4N^2}}$$

One can easily demonstrate (by substituting $x = \frac{1}{\sqrt{N}}$, and expanding as Taylor series at x=0) that

$$u_c^{PKh} = 1 - \sqrt[4]{\frac{1}{1+4N^2}} = 1 - \frac{1}{\sqrt[4]{4}\sqrt{N}} + \frac{1}{16\sqrt[4]{4}N^2\sqrt{N}} + O\left(\frac{1}{N^4\sqrt{N}}\right) \qquad \textit{(Eq. 29)}$$

That is, as N increases, critical utilization u_c^{PKh} for the queueing wait time converges to 1 as $1 - \frac{1}{\sqrt[4]{4\sqrt{N}}}$, which is very similar, but faster compared to $u_c = 1 - \frac{1}{\sqrt{N}}$ in the case of discrete scheduling.

9 References

Begen, M. A., & Queyranne, M. (2011). Appointment scheduling with discrete random durations. *Mathematics of Operations Research*, 240-257.

Blake, J. T., Carter, M. W., & Richardson, S. (1996). An Analysis of Emergency Room Wait Time Issues via Computer Simulation. *INFOR: Information Systems and Operational Research, 34*(4), 263-273.

Brown, L. D., Gans, N., Mandelbaum, A., Sakov, A., & Shen, H. (2005). Statistical Analysis of a Telephone Call Center. *Journal of the American Statistical Association, 100 (469)*, 36-50.

Cayirli, T., & Veral, E. (2003). Outpatient scheduling in healthcare: A review of literature. *Production and Operations Management*, 519–549.

Chan, W., & Maa, D. (1978). The GI/Geom/N Queue In Discrete Time. *INFOR. Information systems and operational research*, 232-252.

Cooper, R. B. (1981). *Introduction to Queueing Theory.* New York: North-Holland.

Dhall, S. K., & Liu, C. L. (1978, January-February). On a Real-Time Scheduling Problem. *Operations Research, 26*(1), 127-140.

Erlang, A. K. (1909). The theory of probabilities and telephone conversations. *Nyt Tidsskrift for Matematik* , 33-40.

Gradshteĭn, I., & Ryzhik, I. (2007). *Table of integrals, series, and products.* Amsterdam: Elsevier.

Heimann, D., & Neuts, M. (1973). The single server queue in discrete time-numerical analysis IV. *Naval Research Logistics Quarterly*, 753-766.

Huggins, A., Claudio, D., & Eduardo, P. (2014). Improving Resource Utilization in a Cancer Clinic: An Optimization Model. *Proceedings of the 2014 Industrial and Systems Engineering Research Conference*, (p. 1444).

Kc, D., & Terwiesch., C. (n.d.). Impact of Workload on Service Time and Patient Safety: An Econometric Analysis of Hospital Operations. *Management Science, 55*(2009), 1486–1498.

Kleinrock, L. (1975). *Queueing Systems Volume 1: Theory.* New York, NY: Wiley.

Klimko, E., & Neuts, M. (1973). The single server queue in discrete time-numerical analysis II. *Naval Research Logistics Quarterly*, 305-319.

Kumar, S. (2001). *Analytical and metaheuristic solutions for emerging scheduling problems in e-commerce and robotics.* The University of Texas at Dallas, Naveen Jindal School of Management. ProQuest Dissertations Publishing.

LaGanga, L., & Lawrence, S. (2012). Appointment Overbooking in Health Care Clinics to Improve Patient Service and Clinic Performance. *Production and Operations Management*, 874-888.

Little, J. D. (1961, May-June). A Proof for the Queuing Formula: L= λ W. *Operations Research, 9*(3), 383-387.

López, J. M., García, M., Díaz, J. L., & García, D. F. (2000). Worst-Case Utilization Bound for EDF Scheduling on Real-Time Multiprocessor Systems. *Proceedings 12th Euromicro Conference on Real-Time Systems*, (pp. 25-33).

Marchesi, J. F., Hamacher, S., & Fleck, J. L. (2020). A stochastic programming approach to the physician staffing and scheduling problem. *Computers & Industrial Engineering*, 142.

Neuts, M. F. (1973). The single server queue in discrete time-numerical analysis I. *Naval Research Logistics Quarterly*, 297-304.

Neuts, M., & Klimko, E. (1973). The single server queue in discrete time-numerical analysis III. *Naval Research Logistics Quarterly*, 557-567.

Petrean, A. L. (1998). Application of the Queueing Theory to Discrete Event Simulation. *Buletinul ştiinţific al Universitatii Baia Mare, Seria B, Fascicola matematică-informatică, 14*(1), 81-88.

Pollaczek, F. (1930). Über eine Aufgabe der Wahrscheinlichkeitstheorie. I. *Mathematische Zeitschrift, 32*, 64-100.

Punitha, S. (2018). Design and evaluation of traffic delays in toll plaza using combination of queueing and simulation. *Journal of physics, 1139*(1), 1-9.

Rosenthal, D., & Pianykh, O. (2021). *Efficient Radiology: How to Optimize Radiology Operations.* Springer International Publishing.

Wainer, G. A. (2009). *Discrete Event Modeling and Simulation: A Practitioner's Approach.* Boca Raton: CRC Press.

Zacharias, C., & Pinedo, M. (2014). Appointment Scheduling with No-Shows and Overbooking. *Production and Operations Management, 23*(5), 788-801.

Zacharias, C., & Yunes, T. (2019). Multimodularity in the Stochastic Appointment Scheduling. *Management Science*, 744-763.