Theory Comput. Systems **33**, 489–563 (2000)
DOI: 10.1007/s002240010013

Theory of Computing Systems

On Scheduling Parallel Tasks at Twilight*

H. Bast

Max-Planck-Institut für Informatik,
66123 Saarbrücken, Germany
hannah@mpi-sb.mpg.de

Abstract. We consider the problem of processing a given number of tasks on a given number of processors as quickly as possible when only vague information about the processing time of a task is available before it is completed. Whenever a processor is idle, it can be assigned, at the price of a certain overhead, a portion, called a chunk, of the unassigned tasks. The goal is to minimize the makespan, that is, the time that passes until all the tasks are completed. The difficulty then is to find the optimal tradeoff between the processors' load balance, which is favoured by having small, and therefore many, chunks, and the total scheduling overhead, which is lower when there are fewer chunks. This scheduling problem has been the subject of intensive research in the past, and a large variety of heuristics have been proposed. Its mathematical analysis, however, turned out to be difficult even for simplistic models of the vague-information issue, and little theoretical work has been presented to date. In this work we present a novel theoretical model that covers a multitude of natural vague-information scenarios, and for which we can prove general upper and lower bounds on the achievable makespan. From this we derive optimal bounds and algorithms for a whole variety of specific scenarios, including the modelling of task processing times as independent, identically distributed random variables, which guided the design of most of the previously existing heuristics. Unlike traditional approaches, our model neither ignores a priori knowledge of the input (the processing times) nor does it restrict the distribution of the input, but instead works with the concepts of an a priori estimate of the processing times, which is implicit in every algorithm, and a measure for the deviation of this estimate from the actual processing times, which is not known until all the tasks are completed.

* This research was supported in part by a Graduiertenkolleg graduate fellowship of the Deutsche Forschungsgemeinschaft (DFG), and in part by ESPRIT LTR Project No. 20244 - ALCOM-IT.

1. Introduction

Most information is vague, that is, incomplete and possibly imprecise, but modelling this concept in a rigorous yet practically useful way is very difficult. Indeed, virtually all theoretical studies of computational problems proceed from one of the three following simplifying assumptions: the relevant information is completely known a priori, it is completely unknown a priori, or its distribution is in some way restricted. In this article we study a multiprocessor scheduling scenario where none of these simplifications leads to satisfactory results, and for which, correspondingly, hardly any theoretical work exists. On the other hand, we will see that this scenario has a very practical background, and that many heuristics have been devised in the past; all of these, however, lack a solid theoretical underpinning. We address this deficit by providing, for the first time for such a vague-information setting, a precise yet general mathematical model, together with a comprehensive theoretical analysis.

The following subsection will guide the reader through the vast scheduling literature, and step by step familiarize him or her with the various aspects and subtleties of our scheduling scenario. Impatient readers, who want to know about this scenario and about our results right away, should consider jumping to Section 1.2 now. Very impatient readers, who are looking for a more formal and complete description of our setting, might even want to jump to Section 2, which technically does not rely on the Introduction.

1.1. *Motivation and Background*

One of the challenges in exploiting the power of parallel computers is to map, or *schedule*, the parallelism contained in a program onto a set of processors such that these processors are utilized as effectively as possible. Since the bulk of the running time of a program is spent in its repetitive parts, and these are typically coded as loops, one of the key issues in this context is effective *loop scheduling*, that is, determining which iteration of a loop should be executed on which processor at which time. Of special interest here are so-called *parallel loops*, whose iterations may be executed in any order, hence also concurrently with each other. Parallel loops are prevalent especially in scientific and numeric code, and there exists an abundance of techniques and tools for detecting when iterations are independent as well as for transforming suitable loops to establish that property [35].

In general, processors need to synchronize after the parallel execution of a loop in order to ensure that all iterations are completed before the execution of the next program statement is begun. The goal of effective loop scheduling is therefore to minimize the completion time of the last iteration, a quantity known as the *length* or *makespan* of the schedule. We also remark that a parallel loop might be executed a large number of times (as part of an outer serial loop), in which case even small deviations in the makespan can accumulate as considerable amounts of running time; achieving an optimal or close-to-optimal makespan is therefore of utmost importance.

Problems that involve the scheduling of certain tasks on parallel machines or processors have been the subject of very intensive theoretical research. In the remainder of this subsection, we give an overview over the various lines of this research and discuss their applicability to problems like parallel-loop scheduling. This will naturally lead to our abstract problem formulation, given in Section 1.2.

1.1.1. *Static Scheduling.* When the processing times of the tasks are known in advance, optimal schedules can be computed *statically*, that is, before the actual computation starts. We assume that *preemptions* are not allowed, meaning that once a task is scheduled on a processor, it must be run to completion on that processor. This is a realistic assumption for fine-grained applications like parallel-loop scheduling, where the processing time of a single task (an iteration) is small compared with the overall execution time. We observe that nonpreemptive static scheduling with minimal makespan is tantamount to distributing the tasks on the processors so as to achieve an optimal *load balance*, that is, processor finishing times which differ as little as possible from each other. The exact version of this problem is easily seen to be NP-complete even when the number of processors is restricted to two [3], [8]. For many practical applications, however, sufficiently accurate approximations do equally well. Given for instance the above-mentioned property of relatively small task processing times, it would clearly be satisfactory if the finishing times of the processors differed by no more than a single task's processing time. This, however, can be easily achieved by a linear-time algorithm, which divides the total work into blocks or chunks of consecutive iterations, one chunk for each processor. In the context of loop scheduling, the optimal such scheme is known as *block scheduling* or *static chunking*.

What complicates the application of static scheduling schemes in practice is that in order to know the processing times of the tasks in advance, we must find a way to predict them. This is difficult because certain low-level details of the actual process that determines processing times are practically impossible to foresee. For example, while a compiler might be able to figure out which variables a program will manipulate at a particular point, it usually cannot predict with certainty where in the memory hierarchy (cache, etc.) a particular memory access will take place. In so-called *regular* applications, the outcomes of such low-level events follow a well-predictable pattern. In that case, static scheduling coupled with some kind of performance prediction is the method of choice. A specific scheme that addresses this issue for the particular case of a parallel loop repeated within a serial loop was presented by Bull [2]. We here consider so-called *irregular* applications, where the processing times vary widely and in an unpredictable manner. Given the complex interplay of several processors working in concert, this is clearly a realistic assumption in the context of parallel computing in general, and, as various experimental studies have shown, for loop scheduling in particular [15], [4], [16]. Under such circumstances, static scheduling schemes are clearly inappropriate, as they cannot avoid that some processors finish long before others, thus wasting their time for the rest of the computation.

1.1.2. *Dynamic Scheduling.* In appreciation of the problems pointed out above, much attention in scheduling theory has been devoted to the study of what today are called *online* problems, where parts or all of the relevant information are not available to an algorithm beforehand. A basic method for the setting where nothing is known about the processing time of a task until it is completed is as follows. The tasks are queued in an arbitrary order, and at runtime, whenever a processor becomes idle, the task at the head of the queue is removed and scheduled to that processor. Unlike for static schemes, a task here does not know a priori on which processor it will be processed, which is referred to as *dynamic* scheduling. We remark that for scheduling the iterations of a parallel loop on a shared-memory machine, this scheme can be implemented without a (notoriously inefficient) central scheduling unit: via a shared memory variable the processors can instead “schedule themselves” the tasks from the queue whenever they become idle [32], [23], [33]. In the context of parallel computing, the described method is therefore referred to as *self-scheduling*.

In a sense, self-scheduling is an efficient scheduling scheme. First, by construction it produces a schedule in which the finishing times of the processors differ by at most the processing time of a single task—a result that we have deemed satisfactory above. Second, self-scheduling is provably optimal with respect to the so-called *competitive ratio*, defined as the maximum factor (over all possible inputs) by which the solution deviates from the optimal so-called offline solution, which could have been produced if all the relevant information had been available beforehand. This optimality result is a combination of the upper bound implied by Graham’s analysis [9] and a lower bound proven by Shmoys et al. [31].

As it turns out, however, self-scheduling sometimes performs quite poorly in practice, and at times even worse than simple static schemes. The reason is that each scheduling decision that is postponed to runtime is associated with a certain *overhead* which *adds* to the parallel execution time. Examples for possible sources of such overhead are *synchronization*, employed to handle concurrent requests to a common task pool (for example, the queue above), *communication*, needed to retrieve data required for processing a task, and, of course, *computation*, necessary to make the scheduling decision itself. Especially in the case of fine-grained applications, such as a typical parallel loop, this overhead may account for a significant portion of the total running time, so great care has to be taken so as not to outweigh the gains of an improved load balance. In fact, this outweighing is likely to happen for the described self-scheduling algorithm, which, scheduling one iteration at a time, achieves its near-optimal balance of processor loads at the price of a maximal number of scheduling operations. In this sense, self-scheduling is just the extreme opposite to static scheduling, which minimizes overheads at the risk of a large load imbalance. To achieve better tradeoffs between the load imbalance and the scheduling overhead, we must therefore consider *hybrid* schemes which schedule not all, but several tasks at a time.

In the literature, such hybrid scheduling schemes have hardly been investigated. One obvious reason for this is the implicit assumption that the overhead entailed by a dynamic scheduling operation (which is always there) be negligible compared with the processing time of a single task. A subtler point is that in the scenario where nothing is known about the processing times in advance, self-scheduling comes out as an optimal algorithm with respect to the competitive ratio, even when incorporating overheads into the analysis. This should not be too surprising since when the processing times can be arbitrary, then, intuitively speaking, no advantage lies in scheduling several tasks at a time (a single task might take an equally long time). A meaningful analysis of hybrid scheduling schemes that considers the tradeoff between load imbalance and scheduling overhead must hence be based on some notion of a *bounded irregularity* of the task processing times, by which it would be implied that a larger number of tasks indeed tends to have a larger total processing time than a smaller number of tasks.

1.1.3. *Stochastic Scheduling.* The traditional approach to make such a notion of a bounded irregularity precise is *average-case* analysis. In the context of scheduling problems, an average-case analysis will assume that the processing times behave randomly according to some probability distribution, certain properties of which are provided as part of the input; in the operations-research community, this setting is better known under the heading of *stochastic scheduling* [28]. Unfortunately, average-case analysis is usually much harder than worst-case analysis, and quickly becomes intractable when dealing with dynamic problems. Indeed, compared with the abundance of results built on worst-case analysis, theoretical analyses of scheduling problems with random processing times are few and typically concerned with only the most basic settings. When it comes to considering both random processing times *and* scheduling overheads, only a single, specialized result, due to Kruskal and Weiss [18], seems to be known. We describe this result next, and come back to average-case analysis in Section 1.2.

The object of the investigations by Kruskal and Weiss was the simple *fixed-size chunking* heuristic, which instead of scheduling only one task at a time, schedules so-called *chunks* of a fixed number of tasks at a time. Kruskal and Weiss modelled task processing times, as well as the per-chunk overheads, as independent, identically distributed (and sufficiently well behaved) random variables. In a sense, their result was promising: they showed that the chunk size can be chosen such that the expected makespan is within a factor of $1 + \varepsilon$ of the optimum, where $\varepsilon \to 0$ as the number of tasks goes to infinity. However, this result has a number of weaknesses, and the authors themselves explained that it should be viewed as "a lower bound for real machines". First, their analysis was asymptotic, so that it remained unclear when ε would actually become small. Second, the assumption of independent, identically distributed task processing times leads to chunk processing times that are very sharply concentrated around their mean—indeed, for this setting even the aforementioned naive static chunking strategy gives reasonable results. It was left entirely open what would happen for more realistic, less well-behaved processing times. Third, even within this limited setting, fixed-size chunking is not the asymptotically optimal scheduling scheme. Fourth, a suitable—not to mention optimal—chunk size is very hard to determine, and no intuitive default setting exists. The bottom line is that, roughly speaking, fixed-size chunking is a useful but "quick-and-dirty" heuristic. Indeed, Kruskal and Weiss already noted that an optimal scheme should rather have "the chunk size decrease as the scheduling process evolves". This is in fact quite natural, since smaller chunks are required only towards the end, in order to achieve balanced finishing times, whereas earlier chunks should be larger in order to help keep the scheduling overhead small.

1.1.4. *Decreasing-Size Heuristics.* Following the work by Kruskal and Weiss, a multitude of heuristics for scheduling tasks in chunks of decreasing sizes have been presented, among others by Polychronopoulos and Kuck [29], Tzen and Ni [34], Flynn et al. [7], Lucco [22], Liu et al. [21], and Hagerup [13]. A survey of most of these can be found in the comparative experimental study of Hagerup [13] or in [1]. We only mention here that most of these heuristics were actually implemented, and at least two of them, the *guided self-scheduling* of Polychronopoulos and Kuck [29] and a variant of the *factoring* strategy due to Flynn et al. [7] have been embodied in a variety of serious (that is, not specially created) applications. On top of these basic strategies, numerous more complex schemes were constructed, addressing other important practical issues such as affinity and data locality [24], [5], [15], [25], self-adaptiveness [6], [36], distributed memory [30], [19], [20], and heterogeneous, time-shared environments [17], [26]. For our purposes here, one should note that for all of these, a clever heuristic for partitioning the tasks into chunks is an essential component.

It is conspicious that not even the most basic decreasing-size schemes are supported by any rigorous analysis. This is unsatisfactory in several respects. First, and most obviously, there is the risk of a poor performance even under circumstances that conform well to the underlying model. As is shown in [1], this is indeed the case for several of the named schemes. Second, experiments are in general hardly suitable to assess the appropriateness of the complexity of a scheme relative to its performance. So, for example, the FAC scheme of Flynn et al. [7] was designed on the basis of a very intricate heuristic but tends to perform rather poorly compared with its "quick-and-dirty" variant FAC2, which does away with all the intricacy. Similarly, Hagerup [13] honestly judged his BOLD scheme by saying that "considerations that are at least as logical lead to different variants of BOLD that just happen not to perform as well". A third issue is that many of the previous schemes are controlled by one or more parameters, whose tuning has a great impact on the performance. Concrete guidance for setting these parameters properly was often not given.

1.2. *Our Work*

We summarize the various elements that constitute our scheduling scenario. We are given n tasks, the processing time of each of which is not known until it is completed, to be scheduled in an arbitrary order on p processors, initially idle. Whenever a processor is idle, it may be scheduled an arbitrary portion, called a *chunk*, of the so far unscheduled tasks, at the price of a certain overhead per chunk. Here, as well as later, we work with the assumption of a fixed overhead h. As is shown in Section 3, this (of course unrealistic) restriction is not for technical necessity but rather for convenience, to simplify our presentation. Our goal is to minimize the makespan of the schedule, which we will see to be tantamount to minimizing the sum of the idle times of processors finishing early plus the sum of all overheads, a quantity called the *wasted time* of the schedule. Using this objective, achieving near-optimal makespan corresponds to achieving a wasted time that is negligible compared with the total processing time of the tasks (on which a scheduling algorithm has no influence). It should be noted that when considering wasted times, small constant-factor changes are not very significant: for instance, wasted times amounting to 1% and 5% of the total processing time correspond to a makespan that is off the optimum by a factor of 1.01 and 1.05, respectively.

1.2.1. *A Generic Approach.*

As we have learned from Section 1.1, a meaningful analysis of the tradeoff between load imbalance and scheduling overhead, and hence of the wasted time, must consider some notion of a bounded irregularity of the task processing times. This is of course a fairly vague concept, which can be concretized in many ways. One example is what we call the *independent-tasks setting*, which underlies the theoretical investigations of Kruskal and Weiss [18] as well as most existing heuristic schemes; here the task processing times are independent, identically distributed random variables. Note that this is a particularly well-behaved setting, since the independence assumption implies a very sharp concentration of chunk processing times. Hagerup [13] also considered a special instance of what we call the *coupled-tasks* setting, where task processing times are again identically distributed random variables; however, tasks are now divided arbitrarily into groups, and independence only holds between pairs of tasks from different groups, while the processing times of all tasks in the same group are equal with probability one. This models, for example, an image processing application, where, naturally, processing costs vary from region to region rather than from pixel to pixel. Yet another model of bounded irregularity is that in which thresholds $T_{\min}$ and $T_{\max}$ are known such that each task processing time is guaranteed to lie within $[T_{\min}, T_{\max}]$. We call this the *bounded-tasks setting*, instances of which were previously also considered by Liu et al. [21].

While it is certainly instructive to study any of these particular settings, it would of course be more desirable to have a meta-model that comprises all of the approaches named above. Results for such a model would provide more comprehensive insight into how exactly the issue of vagueness of information affects scheduling efficiency. Moreover, each of the settings described above hinges on the fact that the input distribution is restricted, so that it is not clear whether results could be extrapolated to distributions that deviate (maybe only slightly) from the imposed requirements. More convincing performance bounds should do away with such restrictions, and instead have the property that they degrade gracefully as inputs become less well-behaved. A third concern, particular to stochastic modellings in general, is that a probability distribution is always in danger of overspecifying the modelled behaviour, by having to assign a probabilitiy to each and every event. So, for example, no probability distribution function can express that some task takes somewhere between 10 and 20 milliseconds—which is certainly an informative statement by itself—without also having to commit to some average time, or to specify how likely it is that less than 15 milliseconds suffice. As a result, probabilistic assumptions may—and usually do—add an artifical regularity to the studied problem, which it did not possess originally. Algorithms and analyses that, perhaps even unknowingly, exploit this structure are hence of somewhat limited use.

A year's (or two) contemplation over the desires expressed above leads us to the following two concepts: a *variance estimator*, which models what an algorithm implicitly assumes on the processing times, and a *deviation*, which measures the deviation of the actual processing times from their estimated behaviour and is not known until all the tasks are completed. The variance estimator is specified by two functions $\alpha, \beta\colon \mathbb{R}^+ \to \mathbb{R}^+$, with the meaning that $[\alpha(w), \beta(w)]$ is an *estimated range* of typical processing times for chunks of size w. The deviation will be defined as a nonnegative real quantity ε that measures the average distance of an actual chunk processing time to its respective range $[\alpha(w), \beta(w)]$. Naturally, ε will be zero if the processing times of all chunks are within the estimated ranges, and the more the processing times deviate from these ranges, the larger the value of ε will be. This approach is described and explained in great detail in Section 2.

Table 1. The optimal wasted time compared with that of fixed-size chunking, static chunking, and self-scheduling for a variety of estimated ranges, where $H = h + \varepsilon$ and $N = n/p$. The entries are exact up to a constant factor, provided that N is sufficiently large compared with H.

	$[w - \sqrt{w}, w + \sqrt{w}]$	$[w/2, 2w]$	$[w/2, w \log w]$	$[w/2, w^2]$
OPT	$H \cdot \log\log N$	$H \cdot \log N$	$H \cdot \log^2 N$	$H \cdot \sqrt{N}$
FIX	$\sqrt{H \cdot N}$	$\sqrt{H \cdot N}$	$\sqrt{H \cdot N \log N}$	$(H \cdot N)^{2/3}$
SC	$H + \sqrt{N}$	$H + N$	$H + N \cdot \log N$	$H + N^2$
SS	$H \cdot N$	$H \cdot N$	$H \cdot N$	$H \cdot N$

1.2.2. *Theoretical Results.* Our main result quantifies for each setting of the parameters n, p, h, α, β, and ε, in a closed formula $\mathrm{OPT}(n, p, h, \alpha, \beta, \varepsilon)$, the optimal wasted time that can be achieved for scheduling n tasks on p processors with overhead h, when the average deviation of the processing time of a chunk with respect to $[\alpha, \beta]$ is at most ε. To establish this result, we present a single generic algorithm, the *balancing* strategy, establish upper bounds on its wasted time, and subsequently prove that no other algorithm can do better. This powerful result is in sharp contrast with aforementioned previous work that, even for the particular independent-tasks setting, could not provide any nontrivial performance bounds. Our formula for OPT is extremely sexy, namely,

$$(h + \varepsilon) \cdot \gamma^*(n/p),$$

where $\gamma^*(n/p) = \min\{\, i\colon\ \gamma^{(i)}(n/p) \le 0\,\}$, and γ is approximately $\mathrm{id} - \alpha \circ \beta^{-1}$, that is, $\gamma(x) \approx x - \alpha(\beta^{-1}(x))$. Unfortunately, the underlying intuition cannot be easily conveyed in a few lines; indeed, the whole of Section 2.3 is dedicated to this task. Instead we provide Table 1 as an appetizer. It states the order of magnitude of $\mathrm{OPT}(n, p, h, \alpha, \beta, \varepsilon)$ for a number of selected ranges $[\alpha(w), \beta(w)]$. For the sake of comparison, the last three rows provide the corresponding performance of an optimal fixed-size scheme, and of the aforementioned static-chunking and self-scheduling schemes, respectively. The previously existing decreasing-size schemes are missing from the table because all of them were designed for the special independent-tasks setting, so that they do not easily adapt to more irregular scheduling problems.

Equipped with the magic formula for OPT, it becomes easy to prove upper bounds for a whole variety of input models. So, for instance, for the independent-tasks setting we can show that the expected deviation of a chunk with respect to $[w - \sigma\sqrt{\ln w} \cdot w^{1/2}, w + \sigma\sqrt{p + \ln w} \cdot w^{1/2}]$ is on the order of the standard deviation σ of a single task's processing time. This correspondence, which will be established by a careful approximation of the convergence rate of the central limit theorem, immediately implies an upper bound of $O((h + \sigma) \cdot \log\log(n/p))$ on the *expected* wasted time achievable in the independent-tasks setting; as will also be shown, no algorithm can do significantly better than this. Similary, the (much more poorly behaved) coupled-tasks setting corresponds to ranges $[w, pw^2]$ and $\varepsilon \le \sigma^2$, which implies an $O((h + \sigma^2) \cdot \sqrt{n})$ upper bound on the expected wasted time. This can be improved in special cases, for example to $O(h \cdot \log n \cdot \log(n/p))$, when the chunk processing times have exponentially small tails. For the bounded-tasks setting, finally, the formula for OPT immediately implies an upper and lower bound of $\Theta(h \cdot \log(n/p))$.

1.2.3. *Practical Significance.* Apart from yielding tight performance bounds for a whole variety of concrete settings, our analysis also provides valuable insights on how to design practical schemes for scheduling tasks on parallel processors.

For example, we will see that by underestimating the irregularity, that is, by choosing too narrow estimated ranges, the average deviation ε may grow as large as n/p, resulting in disastrous performance. On the other hand, the first row of Table 1 gives an indication that even a considerable widening of the estimated ranges has a much less dramatic effect. This suggests that overestimating the irregularity is always preferable to risking large deviations. Since large variances call for smaller chunks, it follows that in case of doubt a chunk size should be chosen too small rather than too large—a guideline that was not adequately followed in many previous papers.

Further, it turns out that for small to moderate degrees of irregularity, in particular for the independent-tasks and the bounded-tasks setting, very simple scheduling schemes suffice to achieve a wasted time that is logarithmic in n/p, which should be good enough for all practical purposes. This insight was missing from most previous work, where much more complicated strategies did not achieve a significantly better performance (see also [1]). We show that in order to achieve sublogarithmic wasted times, a strategy must consider the processing times of already completed chunks, which inevitably leads to more complicated algorithms (and implementations).

Another practically relevant outcome of our analysis is that the decreasing of chunk sizes should stop at some minimal chunk size that should ideally be a small constant factor times the scheduling overhead. In fact, actual implementations of dynamic loop scheduling schemes have been applying this principle for a long time, but so far no theoretical explanation could be given. In previous works, only Lucco [22] and Liu et al. [21] took this issue into account.

1.2.4. *Beyond Scheduling.* We would finally like to highlight two contributions of this work that we expect to be of interest beyond the particular problem studied.

One such contribution is our idea of modelling the issue of vague information in computational problems by an *estimate*, which models the implicit assumptions an algorithm makes on the input, together with a *deviation*, which is not known until after the problem has been solved. This approach is an alternative to the traditional probabilistic approach, where the quantities in question are modelled as random variables. In the context of this work, the deterministic approach turns out to be simpler, more direct, closer to reality, and more general than its probabilistic counterpart. As a matter of fact, the deterministic approach was our key to solving a problem that in previous work, using probabilistic arguments, appeared to be mathematically intractable. We would expect a similar approach to yield new results and insights also for other problems involving quantities that vary in an unpredictable but somehow limited manner.

A second contribution that we expect to be of more general interest is our *master theorem for the* * *operator*. For a function $\gamma\colon \mathbb{R} \to \mathbb{R}$, the * operator "counts" the number of iterations of γ required to get from some x to some y; formally, $\gamma^*(x, y) = \min\{ i \in \mathbb{N}\colon \gamma^{(i)}(x) \leq y \}$. Just as in our work, deriving closed formulas for $\gamma^*(x, y)$ for a given function γ frequently occurs as a subtask in the analysis of all kinds of algorithms, where it is typically solved in some ad hoc manner. Our master theorem, stated and proven in Section 4.1, provides a surprising approximation of $\gamma^*(x, y)$ in terms of the integral $\int dz/(z - \gamma(z))$.

1.3. *Overview*

The remainder of this article is organized as follows. The next section sets the framework for our theoretical investigations. In particular, we make precise the concepts of our generic approach and carefully explain the intuition behind them. Following that, Section 3 is dedicated to the proof of our generic upper bound, which we formulate in what we call our *Main Theorem*. We first state the theorem, give a few explanations, and then proceed to the (quite involved) proof. In the course of the proof, our new *balancing* strategy is described and explained. Section 4 is devoted to specific bounds for our scheduling problem. We show how to instantiate our Main Theorem for a variety of settings, including the aforementioned bounded-tasks, independent-tasks, and coupled-tasks setting. Apart from yielding specific results, this section also provides valuable intuition on the functional relation described by the magic OPT formula. In particular, this section presents our *master theorem for the* * *operator*. Section 5, finally, is concerned with various lower bounds. We first show that no algorithm can do significantly better than what is stated in our Main Theorem, and then extend this result to randomly distributed processing times.

2. Framework

This section sets the formal framework for our scheduling scenario. In Section 2.1 we first recapitulate the scenario and introduce some basic terminology. Section 2.2 provides the definitions laying the ground work for our generic analyis. Section 2.3 serves to clarify the intuition behind our formalization.

2.1. *Basic Setting and Definitions*

Given are n tasks, ordered in a queue, to be processed on p processors, initially idle. Whenever a processor is idle, it may remove an arbitrary number of tasks, called a *chunk*, from the head of the queue. The processor is then working for a period of time, which consists of the *overhead* and the *processing time* of the chunk, where the latter is just the sum of the processing times of the contained tasks. For the sake of clarity, all our results are stated for a fixed overhead h per chunk; as we will see in Section 3, however, these results can be easily extended to variable overheads. Since the processing time of a chunk is not known in advance, at any time all that is known about a scheduled chunk is whether it is completed or not. Once a chunk has been assigned to a processor it may not be preempted, but has to be run to completion on that processor.

A *scheduling algorithm* is a (deterministic) algorithm that determines how many tasks an idle processor removes from the queue at which time. We say that a chunk is *scheduled* (synonymously: *assigned*, *allocated*) by an algorithm, and we refer to the number of tasks in a chunk as the *size* of that chunk. According to the above description, for determining a chunk size an algorithm may employ knowledge of the processing times of already completed chunks. If it ignores this information, the partitioning of the

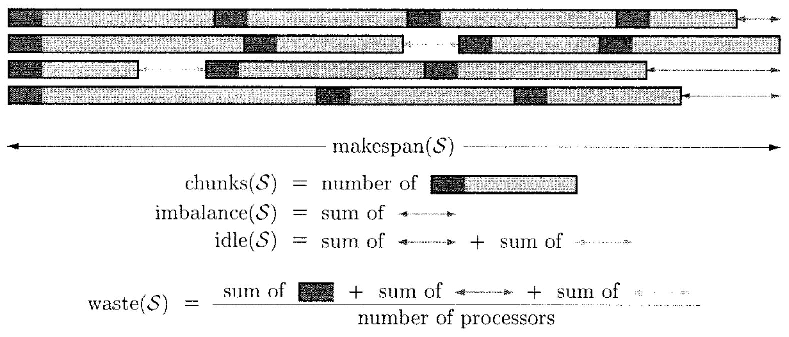
makespan($\mathcal{S}$)
chunks($\mathcal{S}$) = number of
imbalance($\mathcal{S}$) = sum of
idle($\mathcal{S}$) = sum of + sum of
waste($\mathcal{S}$) = sum of + sum of + sum of / number of processors

Fig. 1. A schedule $\mathcal{S}$ on four processors and some associated quantities.

tasks into chunks will be independent of the task processing times; the algorithm is then said to have *fixed partition.*

Given n tasks and p processors, a scheduling algorithm produces a *schedule*, defined as the partitioning of the tasks into chunks together with a mapping that determines for each chunk the time when it is scheduled, its completion time, its overhead, and the (index of the) processor to which it is assigned. For our analysis, it is convenient to understand a chunk as a collection of tasks *plus* its image under the mapping of the schedule of which it is a part. In view of this convention, we often denote schedules by a collection of chunks, and, in particular, write $\mathcal{C} \in \mathcal{S}$ to denote that chunk $\mathcal{C}$ belongs to the partitioning of schedule $\mathcal{S}$. The following definition names the characteristic properties of a schedule, which Figure 1 illustrates by an example.

Definition. For a schedule $\mathcal{S}$ on p processors and with overhead h per chunk, denote by c_k the number of chunks assigned to the kth processor, by T_k their total processing time, and by t_k^{fin} the finishing time of the last such chunk, for $k = 1, \ldots, p$. Then define

$$\text{chunks}(\mathcal{S}) = \textstyle\sum_{k=1}^{p} c_k,$$

$$\text{makespan}(\mathcal{S}) = \max\{t_1^{\text{fin}}, \ldots, t_p^{\text{fin}}\},$$

$$\text{imbalance}(\mathcal{S}) = \textstyle\sum_{k=1}^{p}(\text{makespan}(\mathcal{S}) - t_k^{\text{fin}}),$$

$$\text{idle}(\mathcal{S}) = \textstyle\sum_{k=1}^{p}(\text{makespan}(\mathcal{S}) - T_k - h \cdot c_k),$$

$$\text{waste}(\mathcal{S}) = (h \cdot \text{chunks}(\mathcal{S}) + \text{idle}(\mathcal{S}))\,/p.$$

The last two quantities are referred to as the *idle time* and *wasted time*, respectively, of $\mathcal{S}$.

From the definition above, or more easily from Figure 1, it is straightforward to deduce that p times the makespan of a schedule is just p times its wasted time plus the total processing time of all tasks. A scheduling algorithm has no influence on the latter, so in order to achieve a schedule with near-optimal makespan, it must take care to incur as little wasted time as possible.

2.2. *Modelling Processing Time Irregularity*

We now provide the basic ingredients for our generic cake, as tasted in the Introduction: the *variance estimator*, which represents an algorithm's a priori estimate of chunk processing times, the *deviation*, which measures the deviation of the actual processing times from these estimates, and the *progress rate*, which specifies for a particular estimate the optimal "pace" of the scheduling process. We first give a precise definition for each of these terms, and afterwards provide extensive intuition in the form of a simplified analysis. In the following definition, as well as later in the paper, id is used to denote the identity function $x \mapsto x$.

Definition. For continuous and strictly increasing functions $\alpha, \beta\colon \mathbb{R}^+ \to \mathbb{R}^+$ such that, for some constant $A \geq 1$, $\mathrm{id}/A \leq \alpha \leq \mathrm{id} \leq \beta$ on $\mathbb{R}^+$, and such that $\beta - \alpha$ is increasing, the function

$$[\alpha, \beta]\colon\ w \mapsto [\alpha(w), \beta(w)]$$

is called a *variance estimator*. The function $\beta - \alpha$ is referred to as the *width* of $[\alpha, \beta]$, and we say that $[\alpha, \beta]$ has *sublinear, linear,* or *superlinear* width if, for $w \to \infty$, the quotient $(\beta(w) - \alpha(w))/w$ tends to zero, to a positive constant, or to infinity, respectively.

We briefly comment on the finer points of this first definition. As described in the Introduction, the intended meaning of $[\alpha(w), \beta(w)]$ is that it estimates the range of processing times for chunks of size w. The fact that α and β are defined over the positive reals instead of over the positive integers is merely a technicality, which will be convenient later, in the analysis. The condition $\alpha \leq \mathrm{id} \leq \beta$ reflects the concept of a similarity between task processing times, which we found to be a prerequisite to a meaningful analysis. Note that our definition relates to a time scale, where the processing time of a single task is "around" 1. The condition $\mathrm{id}/A \leq \alpha$, finally, is essential to ensure that a variance estimator represents meaningful information, because assuming bounds on the processing times from above but none from below amounts to an almost complete online setting. If, for example, all chunks assigned after the very first one had (close to) zero processing time, then the wasted time of the schedule would be proportional to the processing time of that first, and typically large, chunk.

Definition. Let $[\alpha, \beta]$ be a variance estimator. Then we define, for a chunk $\mathcal{C}$ of size w with processing time T that is part of a schedule on p processors,

$$\begin{aligned}
\mathrm{early}_\alpha(\mathcal{C}) &= \max\{0, \alpha(w) - T\},\\
\mathrm{late}_\beta(\mathcal{C}) &= \max\{0, T - \beta(w)\},\\
\mathrm{dev}_{\alpha,\beta}(\mathcal{C}) &= \mathrm{early}_\alpha(\mathcal{C}) + (p-1)\cdot \mathrm{late}_\beta(\mathcal{C}),
\end{aligned}$$

called the *earliness*, *lateness*, and *deviation*, respectively, of $\mathcal{C}$ with respect to $[\alpha, \beta]$. From that we define, for a schedule $\mathcal{S}$ on p processors,

$$\begin{aligned}
\text{sum-early}_\alpha(\mathcal{S}) &= \sum_{\mathcal{C}\in\mathcal{S}} \text{early}_\alpha(\mathcal{C}),\\
\text{sum-late}_\beta(\mathcal{S}) &= \sum_{\mathcal{C}\in\mathcal{S}} \text{late}_\beta(\mathcal{C}),\\
\text{max-late}_\beta(\mathcal{S}) &= \max_{\mathcal{C}\in\mathcal{S}} \text{late}_\beta(\mathcal{C}),
\end{aligned}$$

and

$$\begin{aligned}
\text{av-dev}_{\alpha,\beta}(\mathcal{S}) &= (\text{sum-early}_\alpha(\mathcal{S}) + (p-1)\cdot \text{sum-late}_\beta(\mathcal{S}))/\text{chunks}(\mathcal{S}),\\
\text{am-dev}_{\alpha,\beta}(\mathcal{S}) &= (\text{sum-early}_\alpha(\mathcal{S}) + (p-1)\cdot \text{max-late}_\beta(\mathcal{S}))/\text{chunks}(\mathcal{S}).
\end{aligned}$$

The latter quantities are referred to as the *average deviation* and *amortized deviation*, respectively, of $\mathcal{S}$ with respect to $[\alpha, \beta]$.

We briefly explain why we have defined two measures for the deviation of a schedule. First observe that both of them are zero if and only if the processing times of all chunks are within the estimated ranges according to $[\alpha, \beta]$. Also, in both definitions, finishing the processing of a chunk earlier or later than estimated is weighted differently, the intuitive reason being that the earliness of a chunk merely affects the processor working on it, while all processors may have to wait for a chunk that finishes late; this becomes clearer in Section 2.3. The two measures differ in that the average deviation accounts for the lateness of *every* chunk, while only the chunk with maximal lateness contributes to the amortized deviation. In particular, we always have

$$\text{am-dev}_{\alpha,\beta}(\mathcal{S}) \le \text{av-dev}_{\alpha,\beta}(\mathcal{S}),$$

and the two measures are equal if and only if all the lateness of the schedule is concentrated on one chunk. Our main result will be expressed in terms of the average deviation, which is easier to handle, while some of our more specific results, dealt with later in the paper, call for the more precise (and actually more natural) amortized-deviation measure. Note that the definition of the deviation of a chunk is consistent with those of the deviation of a schedule, in the sense that for an arbitrary chunk $\mathcal{C}$, $\text{dev}_{\alpha,\beta}(\mathcal{C}) = \text{av-dev}_{\alpha,\beta}(\{\mathcal{C}\}) = \text{am-dev}_{\alpha,\beta}(\{\mathcal{C}\})$, where $\{\mathcal{C}\}$ denotes the (sub)schedule consisting only of the chunk $\mathcal{C}$.

In the definition of the progress rate, given next, the $\circ$ operator denotes the composition of two functions f and g, that is, $f \circ g \colon x \mapsto f(g(x))$. The inverse of a function $f \colon \mathbb{R}^+ \to \mathbb{R}^+$ that is strictly increasing and unbounded (but not necessarily surjective), is defined as $f^{-1}\colon y \mapsto \inf\{x \ge 0 : f(x) \ge y\}$. If f is a bijection, this is just the usual inverse of f. If f is a bijection between $\mathbb{R}^+$ and (y_0, ∞), for some $y_0 > 0$, then $f^{-1}(y) = 0$, for $y \le y_0$. The $\circ$ as well as the $^{-1}$ notation is used extensively throughout the paper.

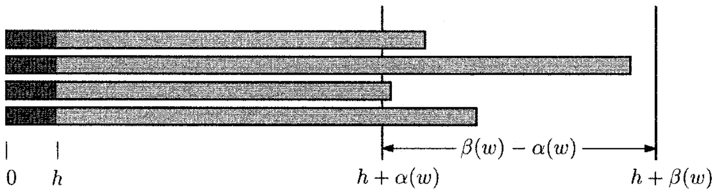

Fig. 2. The imbalance of the initial chunks is at most $p \cdot (\beta(w) - \alpha(w))$.

Definition. For an arbitrary variance estimator $[\alpha, \beta]$ and for arbitrary $M > 0$, the *progress rate* associated with $[\alpha, \beta]$ and M is defined as

$$\gamma_M = \max\left\{0, \mathrm{id} - \max\{M, (\mathrm{id} + \delta)^{-1}\}\right\},$$

where $\delta = \alpha^{-1} \circ (\beta - \alpha)$.

It will become clear in the following section that this complicated function has in fact a very natural interpretation in the context of our scheduling problem.

2.3. *Intuitive Analysis*

As promised, we next provide intuition for the above definitions, by investigating, under extremely simplifying (and formally inadmissable) assumptions, the properties of an optimal scheduling process. Let $[\alpha, \beta]$ be a variance estimator, let p be the number of processors, and let us first proceed under the assumption that the deviation is zero. We begin by considering the first p chunks to be scheduled; since they are all scheduled at the same time, it seems natural to have them of a common size w. Doing this, the imbalance of the (partial) schedule constituted by these p chunks is certainly at most $(p - 1) \cdot (\beta(w) - \alpha(w))$; this is illustrated in Figure 2.

Now, for the sake of simplicity, we assume that α is a linear function, that is, $\alpha = \mathrm{id}/A$, for some $A \geq 1$. To be able to "catch up" with the processor finishing last, and thus to achieve an even processor utilization, each other processor must process at least $A \cdot (\beta(w) - \alpha(w))$ more tasks. Since, in view of the scheduling overhead, it is desirable that we schedule as few chunks as possible, w should be chosen maximal with respect to this constraint. This suggests a value for w that guarantees that when $p \cdot w$ tasks are assigned, $(p - 1) \cdot A \cdot (\beta(w) - \alpha(w)) \approx p \cdot A \cdot (\beta(w) - \alpha(w))$ tasks will be left. Writing δ for $A \circ (\beta - \alpha) = \alpha^{-1} \circ (\beta - \alpha)$, like in the definition of progress rate above, w should hence satisfy

$$p \cdot w + p \cdot \delta(w) = n,$$

where n is the total number of tasks. Under the assumption that $\mathrm{id} + \delta$ is a bijection of $\mathbb{R}^+$ (which it indeed is if $\lim_{w \to 0} \beta(w) = 0$) this equation has a unique solution

$$w = (\mathrm{id} + \delta)^{-1}(n/p).$$

Unfortunately, matters become really complicated after the first p chunks since from then on the assignment of chunks will most likely occur in a completely asynchronous

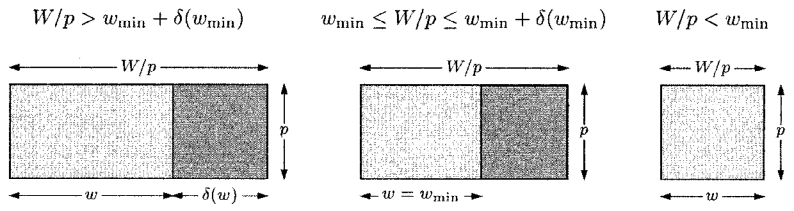
$W/p > w_{\min} + \delta(w_{\min})$
$w_{\min} \le W/p \le w_{\min} + \delta(w_{\min})$
$W/p < w_{\min}$
W/p
p
w
$\delta(w)$
$w = w_{\min}$

Fig. 3. $w = \min\left\{W/p, \max\{w_{\min}, (\mathrm{id}+\delta)^{-1}(W/p)\}\right\}$.

manner. However, for the purpose of our providing intuition here, we assume that, throughout the scheduling process, chunks are scheduled in *rounds* of p chunks of a common size each, determined according to the rule formulated above. However, we consider that, as our true analysis will show, an optimal algorithm should not assign chunks smaller than a certain *minimal chunk size* $w_{\min}$. Taking this into account, the common chunk size for a round should be chosen as

$$w = \min\left\{W/p, \max\{w_{\min}, (\mathrm{id}+\delta)^{-1}(W/p)\}\right\},$$

where W is the number of tasks unassigned before the first chunk of that round is scheduled. Here the minimum ensures that for the very last chunk we do not assign more tasks than are actually left. An illustration of this formula is given in Figure 3.

We measure the progress of an algorithm by the number of unassigned tasks divided by p. Then a round of p chunks, with common size determined according to the above formula, reduces this quantity from some W/p to

$$\left(W - p \cdot \min\left\{W/p, \max\{w_{\min}, (\mathrm{id}+\delta)^{-1}(W/p)\}\right\}\right)/p.$$

According to the definition of progress rate, this is just $\gamma_{w_{\min}}(W/p)$. Note that, in Figure 3, $\gamma_{w_{\min}}(W/p)$ is just the width of the dark grey rectangle(s). Clearly, the larger δ is, the closer $\gamma_{w_{\min}}$ is to the identity function, which corresponds to a (literally) small progress made by a single round. Table 2 gives a feeling for how the progress rate is related to δ, where, for simplicity, it is assumed that $w_{\min} = 1$.

It is now easy to see why it is natural to express bounds on the wasted time in terms of the progress rate. To this end, consider the following illustration of a scheduling process evolving in rounds as described above:

$$n/p \longrightarrow \gamma_{w_{\min}}(n/p) \longrightarrow \gamma^{(2)}_{w_{\min}}(n/p) \longrightarrow \cdots \longrightarrow 0.$$

Table 2. Examples of variance estimators and their associated progress rate.

$[\alpha(w), \beta(w)]$	$\delta(x)$	$(\mathrm{id}+\delta)^{-1}(x)$	$\gamma_1(x)$
$[w, w+\sqrt{w}]$	$\sqrt{x}$	$\approx x - \sqrt{x}$	$\approx \sqrt{x}$
$[w, 2w]$	x	$x/2$	$x/2$
$[w, w \cdot \log w]$	$\approx x \log x$	$\approx x/\log x$	$\approx x - x/\log x$
$[w, w^2]$	$\approx w^2$	$\approx \sqrt{x}$	$\approx x - \sqrt{x}$

The number of rounds in this process can be concisely expressed as $\gamma^*_{w_{\min}}(n/p)$, where for an arbitrary function $f\colon \mathbb{R} \to \mathbb{R}$, f^* is defined as

$$f^*(x) = \min\{i \in \mathbb{N}\colon\ f^{(i)}(x) \le 0\}.$$

Denoting our schedule by $\mathcal{S}$, we hence have

$$\text{chunks}(\mathcal{S}) = p \cdot \gamma^*_{w_{\min}}(n/p).$$

Note that for the variance estimators from Table 2, the function γ^*_1 is approximately $\log\log$, $\log$, $\log^2$, and $\sqrt{\ }$, respectively.

For a bound on the wasted time, it remains to investigate the idle time of $\mathcal{S}$, which, provided that waiting between two chunks never occurs (as is the case for most, though not all, of the algorithms studied in this article), is equal to the imbalance of $\mathcal{S}$. We again simplify matters here, making the seemingly natural assumption that the last chunks to finish are also those which were scheduled last. Then, still in the absence of deviations, the imbalance of $\mathcal{S}$ is certainly bounded by $p \cdot (h + \beta(w_{\min}))$. Now also taking deviations into account, we assume, again for simplicity, that only the very last chunk, we call it $\mathcal{C}_1$, is late, while the last chunks of the other processors, we call them $\mathcal{C}_2, \ldots, \mathcal{C}_p$, are all early. In this seemingly worst case the deviations increase the imbalance by exactly $(p-1) \cdot \text{late}_\beta(\mathcal{C}_1) + \text{early}_\alpha(\mathcal{C}_2) + \cdots + \text{early}_\alpha(\mathcal{C}_p)$. According to the definition given in the previous section, this quantity is just $\text{sum-early}_\alpha(\mathcal{S}) + (p-1) \cdot \text{max-late}_\beta(\mathcal{S})$, and we obtain

$$\text{imbalance}(\mathcal{S}) \le p \cdot h + p \cdot \beta(w_{\min}) + \text{sum-early}_\alpha(\mathcal{S}) + (p-1) \cdot \text{max-late}_\beta(\mathcal{S}).$$

Writing ε for the average deviation of $\mathcal{S}$, which in the considered case is equal to the amortized deviation, we may conclude that

$$\begin{aligned}\text{waste}(\mathcal{S}) &= (h \cdot \text{chunks}(\mathcal{S}) + \text{imbalance}(\mathcal{S}))/p \\ &= \mathcal{O}\left((h+\varepsilon) \cdot \gamma^*_{w_{\min}}(n/p) + \beta(w_{\min})\right).\end{aligned}$$

This is exactly the bound we prove in the next section.

3. Generic Upper Bound

This section is devoted to the proof of our Main Theorem, which, using the formalism introduced in the previous section, provides a generic upper bound that covers a wide spectrum of possible irregularities in the task's processing times. We first state the theorem, and then make a few remarks.

Main Theorem. *Let task processing times be arbitrary, let the overhead be $h \ge 1$, and let $[\alpha, \beta]$ be a variance estimator such that both id/α and $\min\{\beta/\text{id}, 2\}$ are decreasing functions. Then for all $w_{\min} \in \mathbb{N}$, $w_{\min} \ge h$, there exists an algorithm that for all $n, p \in \mathbb{N}$, given n tasks and p processors, produces a schedule $\mathcal{S}$ with*

$$\text{waste}(\mathcal{S}) = \mathcal{O}\left((h+\varepsilon) \cdot \gamma^*_{w_{\min}}(n/p) + \beta(w_{\min})\right),$$

where $\varepsilon = \text{av-dev}_{\alpha,\beta}(\mathcal{S})$ and $\gamma_{w_{\min}}$ is the progress rate associated with $[\alpha, \beta]$ and $w_{\min}$.

We first remark that the conditions imposed on α and β are a technicality which stems from our proofs. For the theorem above, we chose a convenient formulation, while the actual weaker requirements are detailed in Theorems 3.2 and 3.3. All variance estimators considered in this paper have these (for a variance estimator natural) properties.

In Section 5 we prove a lower bound showing that no algorithm can do better than what is stated in the theorem above. This lower bound implies that the above bound is optimal for $w_{\min} = \lceil \alpha^{-1}(h+\varepsilon) \rceil$. To verify this, let $\delta = \alpha^{-1} \circ (\beta - \alpha)$ and observe that for all $x \leq (\mathrm{id} + \delta)(w_{\min})$, $(\mathrm{id} + \delta)^{-1}(x) \leq w_{\min}$, so that owing to $\beta \leq \mathrm{id} + \delta$, $\gamma^*_{w_{\min}}(\beta(w_{\min}))$ is just $\lceil \beta(w_{\min})/w_{\min} \rceil$. This implies that for $n/p \geq \beta(w_{\min})$,

$$\begin{aligned}(h+\varepsilon) \cdot \gamma^*_{w_{\min}}(n/p) &\geq (h+\varepsilon) \cdot (\beta(w_{\min})/w_{\min}) \\ &= ((h+\varepsilon)/\lceil \alpha^{-1}(h+\varepsilon) \rceil) \cdot \beta(w_{\min}) = \Omega(\beta(w_{\min})).\end{aligned}$$

For $w_{\min} = \lceil \alpha^{-1}(h+\varepsilon) \rceil$—and hence actually for all $w_{\min}$ in the order of $\alpha^{-1}(h+\varepsilon)$—the bound from the above theorem therefore becomes

$$\mathrm{waste}(\mathcal{S}) = \mathcal{O}\left((h+\varepsilon) \cdot \gamma^*_{\lceil \alpha^{-1}(h+\varepsilon) \rceil}(n/p)\right),$$

which exactly matches the lower bound stated in Theorem 5.1. Note, however, that a scheduling algorithm does not know ε in advance, which is why we formulated the above theorem for general $w_{\min}$.

At this point, we also comment on the role of the overhead in our scheduling problem. In the above theorem, as well as for all the other results stated in this paper, the per-chunk overhead is assumed to be a fixed constant h. As is clear from our problem definition, however, for bounds on the makespan it is irrelevant which part of the total time consumed by a chunk is overhead and which is processing times of the tasks. As a consequence, all our results therefore continue to hold for arbitrary overheads, with the meaning of h re-interpreted as the *average* overhead incurred for a chunk, that is, the total overhead divided by the number of chunks. This will become clearer in the forthcoming analysis.

A final remark is concerned with the somewhat peculiar role of ε in the bound above, which is not, as one might expect, a property of the set of tasks alone, but of the schedule produced by some algorithm on these tasks. In particular, for one and the same input different algorithms might incur different values of ε. It should be clear that this anomaly is not an artefact of our modelling but that it is inherent in a scheduling scenario involving vague information. Since our algorithms cannot find out in advance which tasks are going to take a long time and which a short time, one algorithm might, by chance, group together tasks with high and low processing times in the same chunk, while another algorithm might schedule all long tasks in one chunk and all short tasks in another chunk. Obviously, the second algorithm will then incur a larger deviation than the first. As we will see in the following section though, this effect disappears when considering concrete settings that make somehow "symmetric" assumptions on the task's processing times.

The remainder of this section is organized as follows. Section 3.1 first establishes a number of abstract properties of the * operator, which will be used on various occasions in the analysis. In Section 3.2 we then consider the class of fixed-partition scheduling algorithms, and show that they can achieve the above stated bound for all variance estimators of at least linear width. Following that, Section 3.3 provides a description of the generic balancing (BAL) strategy, parameterized by $[\alpha, \beta]$, together with a complete analysis. The final section, Section 3.4, is dedicated to a variant of BAL, named BAL′, whose analysis will establish the Main Theorem stated above. The reason that we investigate both schemes is that BAL is more natural and simpler than BAL′, and also more efficient for small to moderate deviations, while for very large deviations only BAL′ is asymptotically optimal.

3.1. *Properties of the Star Operator*

While most of the properties expressed in the lemmas below are quite obvious and easy to prove, it took an exceptional effort from us (we could not resist mentioning it) to establish Lemma 3.5 in its current form. Translated to our scheduling context, the simple but somewhat amazing message of this lemma is that increasing the width of a variance estimator by a constant factor increases the wasted-time bound stated in the Main Theorem by at most the same factor. To avoid any misunderstandings, we first restate our definition of the * operator.

Definition. For an arbitrary function $\gamma\colon \mathbb{R} \to \mathbb{R}$, we define

$$\gamma^*\colon x \mapsto \min\{i \in \mathbb{N}\colon \gamma^{(i)}(x) \le 0\}.$$

Remark. Throughout the paper, we apply the * operator only to functions such that the value assigned above is finite for all x.

Lemma 3.1. *Let $\gamma, \tilde{\gamma}\colon \mathbb{R} \to \mathbb{R}$ such that γ is increasing. Then $\gamma \le \tilde{\gamma}$ implies $\gamma^* \le \tilde{\gamma}^*$.*

Proof. It suffices to check that, by a simple induction,

$$\gamma^{(i)}(x) = \gamma(\gamma^{(i-1)}(x)) \le \gamma(\tilde{\gamma}^{(i-1)}(x)) \le \tilde{\gamma}(\tilde{\gamma}^{(i-1)}(x)) = \tilde{\gamma}^{(i)}(x),$$

for all $i \in \mathbb{N}$ and for all x. □

Lemma 3.2. *Let $\gamma\colon \mathbb{R} \to \mathbb{R}$ be increasing. Then for all $x, y > 0$, and for all $i \in \mathbb{N}_0$,*

$$\gamma^{(i)}(x) \ge y \quad \Rightarrow \quad \gamma^*(x) - \gamma^*(y) \ge i.$$

Proof. For $i' = \gamma^*(y) > 0$, $\gamma^{(i'-1)}(y) > 0$, hence by the assumption on y, and because γ is increasing, $\gamma^{(i'-1+i)}(x) \ge \gamma^{(i'-1)}(y) > 0$. This in turn implies that $\gamma^*(x) > i'-1+i$ and thus $\gamma^*(x) \ge i' + i$. □

Lemma 3.3. *Let $\gamma\colon \mathbb{R} \to \mathbb{R}$ with $\gamma \le \mathrm{id} - M$, for some $M > 0$. Then for all $x, y \ge 0$ with $x \ge y$,*

$$\gamma^*(x) - \gamma^*(y) \le \lceil (x - y)/M \rceil.$$

Proof. Let i be the smallest nonnegative integer with the property that $\gamma^{(i)}(x) \le y$. Then $\gamma^*(x) - \gamma^*(y) \le i$, and because each application of γ decreases its argument by at least M, $i \le \lceil (x-y)/M \rceil$. □

Lemma 3.4. *For increasing* $\delta\colon \mathbb{R}^+ \to \mathbb{R}^+$ *and for arbitrary* $M > 0$, *the function* $\mathrm{id} - \max\{M, (\mathrm{id}+\delta)^{-1}\}$ *is well-defined and increasing.*

Proof. Since $\mathrm{id}+\delta$ is strictly increasing and unbounded, the well-definedness follows by our definition of the inverse given in Section 2.2. For a proof of the monotonicity property, assume that for $x, y \ge 0$, $x - (\mathrm{id}+\delta)^{-1}(x) < y - (\mathrm{id}+\delta)^{-1}(y)$. Then with $x' = (\mathrm{id}+\delta)^{-1}(x)$ and $y' = (\mathrm{id}+\delta)^{-1}(y)$, we have $\delta(x') = x - x' < y - y' = \delta(y')$ and hence, because δ is increasing, $x' < y'$, so that also $x = x' + \delta(x') < y' + \delta(y') = y$. This proves that $\mathrm{id} - (\mathrm{id}+\delta)^{-1}$ is increasing, which continues to hold when the minimum with $\mathrm{id} - M$ is formed. □

Lemma 3.5. *For increasing continuous* $\delta\colon \mathbb{R}^+ \to \mathbb{R}^+$ *and for arbitrary* $M > 0$ *and* $K \in \mathbb{N}$, *let* $\gamma = \mathrm{id} - \max\{M, (\mathrm{id}+\delta)^{-1}\}$ *and* $\tilde{\gamma} = \mathrm{id} - \max\{M, (\mathrm{id}+K\delta)^{-1}\}$. *Then, for all* $x > 0$,

$$\tilde{\gamma}^*(x) \le K \cdot \gamma^*(x).$$

Proof. The key to the proof is showing that for all $x > 0$,

$$\tilde{\gamma}^{(K)}(Kx) \le K \cdot \gamma(x)$$

(we mention that the simpler statement $\tilde{\gamma}^{(K)}(x) \le \gamma(x)$, which would also imply the lemma, is wrong). For that, define $w = \max\{M, (\mathrm{id}+\delta)^{-1}(x)\}$ as the portion that γ subtracts from an arbitrary fixed argument $x > 0$. In case $w = M$, we very simply have $\gamma(x) = x - M$, so that $\tilde{\gamma}^{(K)}(Kx) \le K \cdot x - K \cdot M = K \cdot \gamma(x)$. Otherwise, we have $x = w + \delta(w)$ and thus $\delta(w) = x - w = \gamma(x)$, and for all $y \ge 0$ it holds that

$$y \ge w + K\delta(w) \quad \Longleftrightarrow \quad (\mathrm{id}+K\delta)^{-1}(y) \ge w,$$

that is, $\tilde{\gamma}$ subtracts at least w from any argument $\ge w + K\delta(w)$. Since $Kx = Kw + K\delta(w)$, we conclude that $\tilde{\gamma}^{(K-1)}(Kx) \le w + K\delta(w)$, and in the same way, $\tilde{\gamma}(w + K\delta(w)) \le K\delta(w)$. By the previous lemma, γ is increasing, so that

$$\tilde{\gamma}^{(K)}(Kx) = \tilde{\gamma}(\tilde{\gamma}^{(K-1)}(Kx)) \le \tilde{\gamma}(w + K\delta(w)) \le K\delta(w) = K \cdot \gamma(x),$$

as claimed above. Iterative application of this statement yields that for all $i \in \mathbb{N}_0$,

$$\tilde{\gamma}^{(Ki)}(Kx) \le K \cdot \gamma^{(i)}(x),$$

so that for $i = \gamma^*(x)$, we have

$$\tilde{\gamma}^{(Ki)}(x) \le \tilde{\gamma}^{(Ki)}(Kx) \le K \cdot \gamma^{(i)}(x) \le 0,$$

which, by the definition of the star operator, proves that $\tilde{\gamma}^*(x) \le Ki = K \cdot \gamma^*(x)$. □

Lemma 3.6. *For increasing continuous $\delta\colon \mathbb{R}^+ \to \mathbb{R}^+$ and for arbitrary $M > 0$, let $\gamma = \mathrm{id} - \max\{M, (\mathrm{id}+\delta)^{-1}\}$, and for $\tilde{\delta} = \max\{M, \delta\}$, let $\tilde{\gamma} = \mathrm{id} - \max\{M, (\mathrm{id}+\tilde{\delta})^{-1}\}$. Then, for all $x > 0$, $\gamma^*(x) = \tilde{\gamma}^*(x)$.*

Proof. The proof is by induction on $\gamma^*(x)$, making use of the equivalence $\gamma^*(x) = 1 \iff 0 < x \le M \iff \tilde{\gamma}^*(x) = 1$ several times; in particular, it immediately settles the base case. For $\gamma^*(x) = 2$, we must have $0 < \gamma(x) \le M$. Since $\tilde{\gamma} \ge \gamma$ and $\max\{M, \gamma\} = \max\{M, \tilde{\gamma}\}$, this implies $0 < \tilde{\gamma}(x) \le M$, which in turn proves $\tilde{\gamma}^*(x) = 2$. For $\gamma^*(x) > 2$, finally, let $w = (\mathrm{id} + \delta)^{-1}(x)$, and verify that $\delta(w) = \gamma(x) > M$. Then $\delta(w) = \tilde{\delta}(w)$ and hence $x = w + \delta(w) = w + \tilde{\delta}(w)$, which implies $\gamma(x) = \tilde{\gamma}(x)$, and it follows by way of induction that

$$\gamma^*(x) = \gamma^*(\gamma(x)) + 1 = \tilde{\gamma}^*(\gamma(x)) + 1 = \tilde{\gamma}^*(\tilde{\gamma}(x)) + 1 = \tilde{\gamma}^*(x).$$ □

3.2. *Fixed-Partition Scheduling*

In this section we explore the power of fixed-partition scheduling algorithms, that is, algorithms whose division of the tasks into chunks does not depend on the tasks' processing times. One should note here that of the numerous known heuristics, which we mentioned in the Introduction, all but the most recent one [13] are of the fixed-partition type. For our purposes, it is useful to think of a particular fixed-partition algorithm as being specified by a function $\varrho\colon \mathbb{R}^+ \to \mathbb{N}$ such that, when W tasks are unassigned, the size of the next chunk scheduled is $\min\{W, \varrho(W/p)\}$. Note that the minimum with W is just to ensure that the chunk size is never greater than the total number of remaining tasks. In the following we denote an algorithm defined in this way by $\mathrm{FP}(\varrho)$. Note that, naturally, $\mathrm{FP}(\varrho)$ never inserts waiting time before scheduling a chunk to an idle processor.

We next observe that it is natural for a fixed-partition algorithm to have $\varrho(x) \le x$ unless x is small. This is because when all processors request at roughly the same time—as they indeed do in the beginning—all of them should be assigned a chunk of about the same size. Given that $\varrho(x) \le x$, a scheduling operation by $\mathrm{FP}(\varrho)$ cannot decrease the number of unassigned tasks by more than a factor of $1 - 1/p$, where p is the number of processors, and p successive scheduling operations therefore cannot decrease it by more than a factor of $(1-1/p)^p \ge \frac{1}{4}$. A reasonable fixed-partition algorithm is therefore bound to have a number of scheduling operations that are logarithmic in n/p, the number of tasks per processor.

However, variance estimators of sublinear width have a progress rate γ with $\gamma(x)/x = o(1)$, in which case the bound claimed in the Main Theorem becomes sublogarithmic in n/p. As demonstrated by the following theorem, the class of fixed-partition algorithms is however sufficiently powerful for all variance estimators of at least linear width. With an eye towards a future application, the theorem is formulated for a slightly generalized setting, where the p processors are not all idle initially, but start the computation at arbitrary times $t_1, \dots, t_p$. The quantity

$$\max\{t_1, \dots, t_p\} - \sum_{k=1}^{p} t_k$$

is referred to as the *initial imbalance* of the respective schedule.

Theorem 3.1. *Let task processing times be arbitrary, and let the overhead be* $h \geq 1$. *Let* $[\alpha, \beta]$ *be a variance estimator, and let* $A \geq 1$ *with* $\alpha \geq \mathrm{id}/A$. *Then for all* $w_{\min} \in \mathbb{N}$, $w_{\min} \geq h$, *and for all* $n, p \in \mathbb{N}$, *given* n *tasks and* p *processors, the algorithm* $\mathrm{FP}(x \mapsto \lfloor \beta^{-1}(x/A + \beta(w_{\min}))\rfloor)$ *produces a schedule* $\mathcal{S}$ *with*

$$\begin{aligned}
&\mathrm{chunks}(\mathcal{S}) \leq p \cdot \gamma^*(n/p),\\
&\mathrm{idle}(\mathcal{S}) \leq p \cdot h + p \cdot \beta(w_{\min}) + \max\{0, I - n/A - ph\} + \mathcal{E},\\
&\mathrm{waste}(\mathcal{S}) \leq (h + \varepsilon) \cdot \gamma^*(n/p) + h + \beta(w_{\min}) + \max\{0, I - n/A - ph\}/p,
\end{aligned}$$

where $\mathcal{E} = \text{sum-early}_\alpha(\mathcal{S}) + (p-1)\cdot \text{max-late}_\beta(\mathcal{S})$, $\varepsilon = \text{am-dev}_{\alpha,\beta}(\mathcal{S}) = \mathcal{E}/\mathrm{chunks}(\mathcal{S})$, $\gamma = \max\{0, \mathrm{id} - \max\{w_{\min}, \lfloor (3A\beta)^{-1}\rfloor\}\}$, *and* I *is the initial imbalance of* $\mathcal{S}$.

Addendum. *For any* $\tilde{\delta}\colon \mathbb{R}^+ \to \mathbb{R}^+$ *such that* $\tilde{\delta} \geq 6A \cdot \max\{\beta - \alpha, \mathrm{id}\}$ *on the interval* $[w_{\min}, \lfloor (\mathrm{id} + \tilde{\delta})^{-1}(n/p)\rfloor]$, $\tilde{\gamma} = \mathrm{id} - \max\{w_{\min}, \lfloor (\mathrm{id} + \tilde{\delta})^{-1}\rfloor\}$ *has the property that* $\gamma^*(n/p) \leq \tilde{\gamma}^*(n/p)$.

That a fixed-partition algorithm can achieve the bound stated in the Main Theorem for variance estimators of at least linear width is implied by the addendum of Thereom 3.1 as follows. Given a variance estimator $[\alpha, \beta]$ such that $\beta - \alpha \geq \mathrm{id}/D$ for some $D \geq 1$, $\tilde{\delta} = 6DA \cdot (\beta - \alpha)$ is easily seen to fulfill the condition $\tilde{\delta} \geq 6A \cdot \max\{\beta - \alpha, \mathrm{id}\}$. On the other hand, since $\tilde{\delta}$ is within a constant factor of $\delta = \alpha^{-1} \circ (\beta - \alpha)$, Lemma 3.5 implies that, with $\tilde{\gamma} = \mathrm{id} - \max\{w_{\min}, \lfloor (\mathrm{id} + \tilde{\delta})^{-1}\rfloor\}$ and $\gamma_{w_{\min}} = \mathrm{id} - \max\{w_{\min}, (\mathrm{id} + \delta)^{-1}\}$, $\tilde{\gamma}^*$ is within a constant factor of $\gamma^*_{w_{\min}}$. Plugging this into the bound of Theorem 3.1 we obtain that, under the assumptions of that theorem and without initial imbalance,

$$\mathrm{waste}(\mathcal{S}) = \mathcal{O}\Big((h + \varepsilon) \cdot \gamma^*_{w_{\min}}(n/p) + \beta(w_{\min})\Big),$$

which is exactly the bound stated in the Main Theorem.

The proof of Theorem 3.1 is organized as follows. We first give a complete proof for the case $A = 1$, that is, for $\alpha = \mathrm{id}$, and subsequently extend our findings to the general case by means of a simple time-scaling argument. In the analysis for $A = 1$, we first investigate how the imbalance of the schedule produced by $\mathrm{FP}(x \mapsto \lfloor \beta^{-1}(x + \beta(w_{\min}))\rfloor)$ develops over time. Then we estimate the total number of scheduling operations. A final paragraph is dedicated to the proof of the addendum. Throughout the proof, let l denote the number of chunks in $\mathcal{S}$, and for $j = 1, \dots, l$, we use w_j and W_j for the size of the jth chunk and the number of tasks unassigned before the scheduling of that chunk, respectively. In particular then, $w_j = \min\{W_j, \lfloor \beta^{-1}(W_j/p + \beta(w_{\min}))\rfloor\}$. For conventional purposes, we also take $W_{l+1} = 0$.

3.2.1. *The Idle Time.* For $j = 1, \ldots, l$, let $\mathcal{C}_j$ denote the jth chunk assigned. As is shown next by a simple induction, for all $j = 0, \ldots, l$,

$$\begin{aligned}&\text{imbalance}(\{\mathcal{C}_1, \ldots, \mathcal{C}_j\})\\ &\quad\le p \cdot h + p \cdot \beta(w_{\min}) + W_{j+1} + \max\{0, I - n - ph\}\\ &\qquad + \text{sum-early}_{\text{id}}(\{\mathcal{C}_1, \ldots, \mathcal{C}_j\}) + (p-1) \cdot \text{max-late}_\beta(\{\mathcal{C}_1, \ldots, \mathcal{C}_j\}).\end{aligned}$$

For the base case $j = 0$, note that all the terms on the right-hand size are nonnegative. For the induction step $j - 1 \to j$, we distinguish between two cases, depending on the processing time T_j of $\mathcal{C}_j$. In case this chunk is the last to finish among those in $\{\mathcal{C}_1, \ldots, \mathcal{C}_j\}$, we have

$$\begin{aligned}&\text{imbalance}(\{\mathcal{C}_1, \ldots, \mathcal{C}_j\})\\ &\quad\le (p-1) \cdot (h + T_j)\\ &\quad\le (p-1) \cdot (h + \beta(w_j) + \text{late}_\beta(\mathcal{C}_j))\\ &\quad\le p \cdot h + p \cdot \beta(w_j) - \beta(w_j) + (p-1) \cdot \text{max-late}_\beta(\{\mathcal{C}_1, \ldots, \mathcal{C}_j\})\end{aligned}$$

and it remains to verify that, owing to $w_j = \min\{W_j, \lfloor \beta^{-1}(W_j/p + \beta(w_{\min}))\rfloor\}$ and $\beta \ge \text{id}$, $p \cdot \beta(w_j) - \beta(w_j) \le W_j + p \cdot \beta(w_{\min}) - w_j = W_{j+1} + p \cdot \beta(w_{\min})$. In the opposite case, if $\mathcal{C}_j$ is not the chunk of $\{\mathcal{C}_1, \ldots, \mathcal{C}_j\}$ to finish last, we simply have

$$\begin{aligned}\text{imbalance}(\{\mathcal{C}_1, \ldots, \mathcal{C}_j\}) &= \text{imbalance}(\{\mathcal{C}_1, \ldots, \mathcal{C}_{j-1}\}) - (h + T_j)\\ &\le \text{imbalance}(\{\mathcal{C}_1, \ldots, \mathcal{C}_{j-1}\}) - w_j + \text{early}_{\text{id}}(\mathcal{C}_j),\end{aligned}$$

and the desired bound follows by the induction hypotheses. This completes the induction, and we have thus proven that

$$\text{idle}(\mathcal{S}) \le p \cdot h + p \cdot \beta(w_{\min}) + \max\{0, I - n - ph\} + \mathcal{E},$$

where $\mathcal{E} = \text{sum-early}_{\text{id}}(\mathcal{S}) + (p-1) \cdot \text{max-late}_\beta(\mathcal{S})$.

3.2.2. *The Scheduling Overhead.* In order to bound the total number of chunks scheduled, first observe that for all $j = 1, \ldots, l$, $w_j = \min\{W_j, \lfloor \beta^{-1}(W_j/p + \beta(w_{\min}))\rfloor\} \le W_j/p + \beta(w_{\min})$, and thus $W_{j+1} = W_j - w_j \ge (1 - 1/p) \cdot W_j - \beta(w_{\min})$. Hence for all $j = 1, \ldots, l - p + 1$,

$$W_{j+p-1} \ge (1 - 1/p)^{p-1} \cdot W_j - (p-1) \cdot \beta(w_{\min}) \ge W_j/3 - p \cdot \beta(w_{\min}),$$

which implies that, provided $W_{j+p} > 0$, each of the jth through $(j + p - 1)$th chunks is of size at least

$$\left\lfloor \beta^{-1}(W_{j+p-1}/p + \beta(w_{\min}))\right\rfloor \ge \left\lfloor \beta^{-1}(W_j/(3p))\right\rfloor = \left\lfloor (3\beta)^{-1}(W_j/p)\right\rfloor.$$

In combination with the fact that each chunk, except maybe the very last, has size at least $w_{\min}$, we thus obtain that for all $j = 1, \ldots, l - p$,

$$W_{j+p} \le \max\left\{0, W_j - p \cdot \max\{w_{\min}, \lfloor (3\beta)^{-1}(W_j/p)\rfloor\}\right\} = p \cdot \gamma(W_j/p).$$

By the definition of the * operator this immediately implies the desired bound

$$\text{chunks}(\mathcal{S}) \le p \cdot \gamma^*(n/p).$$

3.2.3. *The Wasted Time.* Combining the bounds from Sections 3.2.1 and 3.2.2, using that $\varepsilon = \text{am-dev}_{\alpha,\beta}(\mathcal{S}) = \mathcal{E}/\text{chunks}(\mathcal{S})$, we obtain

$$\begin{aligned}\text{waste}(\mathcal{S}) &= (h \cdot \text{chunks}(\mathcal{S}) + \text{idle}(\mathcal{S}))/p \\ &\le (h+\varepsilon)\cdot \text{chunks}(\mathcal{S})/p + h + \beta(w_{\min}) + \max\{0, I - n - ph\}/p \\ &\le (h+\varepsilon)\cdot \gamma^*(n/p) + h + \beta(w_{\min}) + \max\{0, I - n - ph\}/p.\end{aligned}$$

This proves Theorem 3.1 for the case $A = 1$.

The extension to the general case is straightforward. Given an arbitrary $A \ge 1$ with $\alpha \ge \text{id}/A$, first observe that, since in the bounds claimed in the theorem α only occurs in the deviations ε and $\mathcal{E}$ (and not in the definition of γ), and because these deviations can only become smaller for a wider variance estimator, we can assume that $\alpha = \text{id}/A$ without loss of generality. We then rescale our time unit by a factor of A, such that the variance estimator becomes $[\text{id}, A\beta]$, and all quantities measured in time, namely, I, h, $\mathcal{E}$, and ε increase by a factor of A. We can now apply the analysis from above, obtaining that for $\gamma = \max\{0, \text{id} - \max\{w_{\min}, \lfloor (3A\beta)^{-1} \rfloor\}\}$,

$$\begin{aligned}&\text{chunks}(\mathcal{S}) \le p \cdot \gamma^*(n/p),\\ &\text{idle}(\mathcal{S}) \le A \cdot p \cdot h + A \cdot p \cdot \beta(w_{\min}) + \max\{0, A \cdot I - n - A \cdot ph\} + A \cdot \mathcal{E},\\ &\text{waste}(\mathcal{S}) \le (Ah + A\varepsilon)\cdot \gamma^*(n/p) + A \cdot h + A \cdot \beta(w_{\min})\\ &\qquad\qquad + A \cdot \max\{0, I - n/A - ph\}/p.\end{aligned}$$

Measured in the original time unit, that is, multiplied by $1/A$, these are exactly the bounds stated in Theorem 3.1.

3.2.4. *The Addendum.* It remains to prove the addendum, which, under the aditional assumption that there exists $\tilde{\delta}$: $\mathbb{R}^+ \to \mathbb{R}^+$ such that $\tilde{\delta} \ge 6A \cdot \max\{\beta - \alpha, \text{id}\}$ on $[w_{\min}, \lfloor (\text{id} + \tilde{\delta})^{-1}(n/p) \rfloor]$, claims a bound on the wasted time in terms of the function $\tilde{\gamma} = \text{id} - \max\{w_{\min}, \lfloor (\text{id} + \tilde{\delta})^{-1} \rfloor\}$. To this end, let $x \le n/p$ and $w = \max\{w_{\min}, \lfloor (\text{id} + \tilde{\delta})^{-1}(x) \rfloor\}$, for which owing to the assumption on $\tilde{\delta}$,

$$w + \tilde{\delta}(w) \ge 3A \cdot w + \tilde{\delta}(w)/2 \ge 3A \cdot w + 3A \cdot (\beta(w) - \alpha(w)) \ge 3A\beta(w).$$

Now if $w > w_{\min}$, we have

$$x \ge (\text{id} + \tilde{\delta})(w) \ge 3A\beta(w) = 3A\beta(\lfloor (\text{id} + \tilde{\delta})^{-1}(x) \rfloor),$$

and since $3A\beta$ and hence also its inverse is increasing, we obtain

$$(3A\beta)^{-1}(x) \ge \lfloor (\text{id} + \tilde{\delta})x \rfloor,$$

and thus also

$$\lfloor (3A\beta)^{-1}(x) \rfloor \ge \lfloor (\text{id} + \tilde{\delta})x \rfloor.$$

For $w = w_{\min}$, we have $\lfloor (\text{id} + \tilde{\delta})^{-1}(x) \rfloor \le w_{\min}$, and we conclude that for arbitrary $x \le n/p$,

$$\max\{w_{\min}, \lfloor (3A\beta)^{-1}(x) \rfloor\} \ge \max\{w_{\min}, \lfloor (\text{id} + \tilde{\delta})^{-1}(x) \rfloor\}.$$

Therefore $\gamma(x) \leq \tilde{\gamma}(x)$, so that by the monotonicity property of the * operator established in Lemma 3.1, $\gamma^*(n/p) \leq \tilde{\gamma}^*(n/p)$. We have thus, finally, proven Theorem 3.1 in its entirety. $\square$

3.3. *The Balancing Strategy*

As we have seen, Theorem 3.1 from the previous section implies the bound stated in the Main Theorem only for variance estimators of at least linear width. This section is dedicated to the generic *balancing* (BAL) strategy, which provides optimal algorithms for every given variance estimator. To achieve this, for (some of) its scheduling decisions the balancing strategy considers the time when a chunk is scheduled, which by definition is ignored by any fixed-partition algorithm.

We first describe the workings of BAL on a high level, from which the strategy may be viewed as working in two phases. In the first phase, BAL groups a number of consecutive processor requests, serving them in what we call a *round*, and trying to maintain the invariant that all processors finish their chunks of a round at roughly the same time. Naturally, chunk sizes will decrease over the rounds, until a point where the width $\beta(w) - \alpha(w)$ of the estimated ranges $[\alpha(w), \beta(w)]$ becomes large relative to the chunk sizes. Then the second phase begins, where the remaining tasks are scheduled by a fixed-partition algorithm, selected according to Theorem 3.1.

We now give a detailed description of BAL. Like a particular fixed-partition algorithm is specified by a function $\varrho\colon \mathbb{R}^+ \to \mathbb{R}^+$, an instance of BAL is obtained by implementing two functions $\varrho_1, \varrho_2\colon \mathbb{R}^+ \to \mathbb{R}^+$, one for each phase; we denote such an instance by BAL(ϱ_1, ϱ_2). The first of these functions is used to determine the size w of the first chunk assigned in a round, namely, $w = \varrho_1(W/p)$, where W is the number of tasks unassigned at the beginning of the round. If exactly p chunks were assigned in the round, each of size w, then according to our heuristic considerations in Section 2.3, w should be chosen as approximately $\max\{w_{\min}, (\mathrm{id}+\delta)^{-1}(W/p)\}$, where $\delta = \alpha^{-1} \circ (\beta - \alpha)$ and $w_{\min}$ is the minimal chunk size. For technical reasons, we actually take $\varrho_1 = \lfloor (\mathrm{id} + \tilde{\delta})^{-1} \rfloor$, where for some $K \geq 6$,

$$\tilde{\delta}(w) = K \cdot \max\{w_{\min}, 2 \cdot \max\{\beta(w) - w, w - \alpha(w)\}\}.$$

This amounts to pretending a slightly larger width, which, as Lemmas 3.5 and 3.6 will ensure, does not affect our final result by more than a constant factor. Concerning the constant K, our analysis will actually choose a relatively large value in order to avoid tedious complications. However, as is pointed out in Section 3.3.4, a smaller value would also work.

After having computed w, which will be the size of the first chunk in the round, BAL next sets $d = (W/p - w)/K$ as the *tolerance* of the round. Note that, for a variance estimator $[\alpha, \beta]$, and for $\varrho_1 = \lfloor (\mathrm{id} + \tilde{\delta})^{-1} \rfloor$ as above, we have $w = \lfloor (\mathrm{id} + \tilde{\delta})^{-1}(W/p) \rfloor$. Hence, if $W/p < K \cdot w_{\min}$, we have $w = 0$ and $d = W/p\big/K$, while in the opposite case, we have $(\mathrm{id} + \tilde{\delta})(w) \leq W/p$ and $d \geq \tilde{\delta}(w)/K = \max\{w_{\min}, 2 \cdot \max\{\beta(w) - w, w - \alpha(w)\}\}$, which implies $[\alpha(w), \beta(w)] \subseteq [w - d/2, w + d/2]$.

Having computed w and d, BAL next tests the condition $d > w/6$. If it is fulfilled, the first phase is finished. According to the above, this happens when either few tasks remain, namely, if $W/p < K \cdot w_{\min}$, or if the width of $[\alpha(w), \beta(w)]$ is relatively

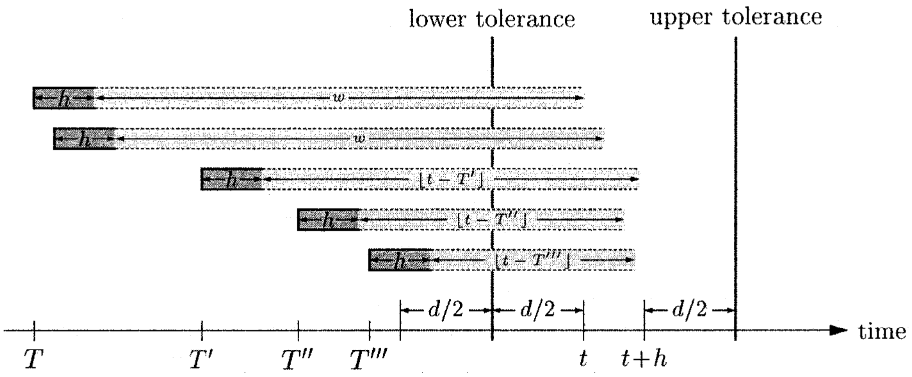

Fig. 4. Chunk assignment in a round started at time T, with target t and tolerance d. The light gray rectangles indicate chunk sizes and not processing times.

large compared with w. If $d \le w/6$, the round is continued and it then holds that $d/2 \ge \max\{\beta(w) - w, w - \alpha(w)\}$, $d \ge w_{\min}$, and $w \ge 6 \cdot w_{\min}$.

Having computed w and d, and having checked that $d \le w/6$, Bal next sets the *target* $t = T + h + w$, where T is the actual time. To achieve a finishing time of approximately t for all chunks assigned in the round, Bal serves each request arriving at a time T' with $T \le T' \le t - d$, by a chunk of size $w' = \min\{w, \lfloor t - T' \rfloor\}$. Note that since $T' \le t - d \le t - w_{\min}$, w' is guaranteed to be at least $w_{\min}$, and that for $T' = T$, indeed $w' = w$. Our analysis assumes that $h \ge 1$ and that $\tilde{\delta}$ is an increasing function, in which case $t \le T' + h + w' \le t + h$ and $\max\{\beta(w') - w', w' - \alpha(w')\} \le d/2$. The estimated finishing time of each chunk assigned in the round is therefore contained in the interval $[t - d/2, t + h + d/2]$, which is referred to as the *tolerance interval* of that round. The quantities $t - d/2$ and $t + h + d/2$ are called the *lower* and *upper tolerance* (*threshold*) of the round, respectively. Figure 4 gives an illustration of what has been described so far.

The round ends at time $t - d$, and with the arrival of the first request at or after time $t - d$, a new round is started in the same manner as just described. This process continues until at the beginning of a potential new round, the condition $d > w/6$ is fulfilled for the first time, in which case the first phase ends. In the second phase, the remaining tasks are scheduled by the fixed-partition algorithm specified by ϱ_2. According to Theorem 3.1, for a given variance estimator $[\alpha, \beta]$ and minimal chunk size $w_{\min}$, we take ϱ_2: $x \mapsto \lfloor \beta^{-1}(x/A + \beta(w_{\min})) \rfloor$, for some $A \ge 1$ with $\alpha \ge \mathrm{id}/A$.

Figure 5 gives a pseudo C-code implementation of the function that computes, for a given request, the chunk size according to Bal(ϱ_1, ϱ_2). The code involves a number of global variables, where (the constants) p and h hold the number of processors and the scheduling overhead, respectively, W is initialized to the total number of tasks, and PHASE is initially set to 1. All the other variables are initialized to zero, to ensure that a new round is started right in the beginning (at time 0).

For a better understanding of the particularities of Bal, we next take a look at the schedule produced in its first phase. Here, as well as later in the analysis, it is convenient to denote by W_i, w_i, d_i, t_i the values of the program variables W, w, d, t just after the

```
(1)   if (PHASE == 1 && $T \geq t - d$) { (* new round *)
(2)       $w = \varrho_1(W/p)$;
(3)       $d = (W/p - w)/K$;
(4)       if ($d > w/6$) PHASE = 2;
(5)       $t = T + h + w$;
(6)   }
(7)   if (PHASE == 1)  $s = \min\{w, \lfloor t - T \rfloor\}$;
(8)   if (PHASE == 2)  $s = \varrho_2(W/p)$;
(9)   $s = \min\{W, s\}$;
(10)  $W = W - s$;
(11)  return $s$;
```

Fig. 5. The size computed by BAL(ϱ_1, ϱ_2) for a chunk scheduled at time T.

ith execution of lines (2)–(5), that is, during the ith round of the first phase. Note that according to line (3), $W_i/p = w_i + Kd_i$, and by line (4), $w_i \geq 6d_i$, which owing to $K \geq 6$ implies that $w_i \geq 3/K \cdot W_i/p$. The tolerance thresholds of round i are $t_i - d_i/2$ and $t_i + h + d_i/2$, and will be denoted by t_i^{low} and t_i^{upp}, respectively. Round i ends at time $t_i - d_i$, which will be denoted by t_i^{end}.

For simplicity, we first restrict our attention to the deviationless case, where the processing times of all chunks are within the estimated ranges. As was shown already in the description of BAL, each chunk then finishes within the tolerance thresholds of its round, which, by the condition in line (1), implies that at most one chunk is assigned to each processor in each round. Hence at most $p \cdot w_i$ tasks are assigned in round i, so that at least $p \cdot Kd_i$ tasks are left for the next round, that is,

$$W_{i+1}/p \geq Kd_i.$$

This in turn implies that $w_{i+1} \geq 3/K \cdot W_{i+1}/p \geq 3d_i$, and since the $(i+1)$th round does not start before $t_i^{\text{end}} = t_i - d_i$, we have

$$t_{i+1}^{\text{end}} = t_{i+1} - d_{i+1} \geq t_i^{\text{end}} + h + w_{i+1} - d_{i+1} > t_i^{\text{end}} + h + 1.5d_i = t_i^{\text{upp}}.$$

This proves the valuable property that the tolerance intervals of successive rounds do not intersect, and that in every round the previously assigned chunks finish before the round ends. Therefore each processor is guaranteed to be assigned at least one chunk in every round, so that actually, by what was already shown above, exactly one chunk is assigned to each processor in each round. We conclude that in the deviationless case, the schedule produced in BAL's first phase exhibits a very regular structure, which is illustrated in an example in Figure 6.

Such a structure makes it very easy to bound the wasted time of the schedule: the idle time is at most $p - 1$ times the width of the last tolerance interval, and the number of scheduling operations is just p times the number of rounds. In the general case, however, with arbitrary and unpredictable deviations, this structure is not preserved. Chunks might then be processed very quickly, causing the *busyness* of a

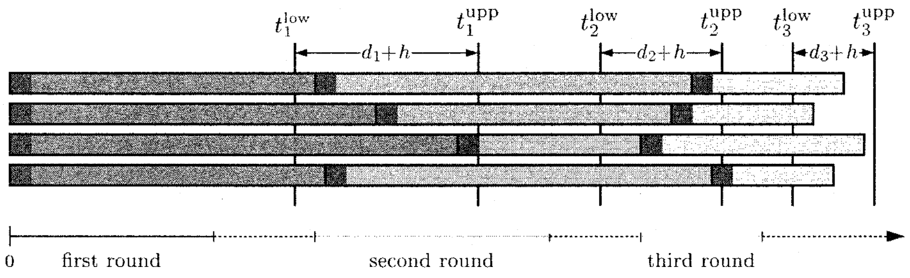

Fig. 6. Three rounds in the deviationless case.

round, defined as the additional number of tasks assigned in this round to processors after their first chunks were completed. Too much busyness is of course bad, leaving the next round with fewer tasks than would be required to reduce the imbalance further. In fact, in the general case it is not even guaranteed that the upper tolerance thresholds of successive rounds form an increasing sequence. Equally bad, processors may also enter a round very late or not at all, in case they are still occupied with chunks of previous rounds. This accounts for the *laziness* of a round, to be defined later as the resulting decrease in the number of tasks scheduled in that round. In an extreme case, only a single chunk might be scheduled in a whole round, and a fast decrease of the unassigned tasks over the rounds, as for the deviationless case, cannot be proven.

Note that busyness and laziness are side effects of the philosophy behind BAL to compensate for an unexpected behaviour of a chunk by adjusting the size of the next chunk assigned to the affected processor accordingly. This behaviour turns out to give good results in practice, but, unfortunately, causes major technical complications in the analysis. In extremely bad cases, when deviations are very large, we will see that BAL is not even asymptotically optimal in the strict theoretical sense. As an alternative, we present in Section 3.4 a variant of BAL that avoids the difficulties mentioned, at the price, however, of a more involved implementation and a considerably worse performance in the case of moderate deviations. The remainder of this section is dedicated to the complete analysis of the BAL strategy, and will culminate in the following result.

Theorem 3.2. *Let task processing times be arbitrary, and let the overhead be $h \geq 1$. Let $[\alpha, \beta]$ be a variance estimator, let $A \geq 1$ with $\alpha \geq \mathrm{id}/A$, and let $w_{\min} \in \mathbb{N}$, $w_{\min} \geq h$ such that, for $K = 49A$, $\tilde{\delta}\colon w \mapsto K \cdot \max\{w_{\min}, 2 \cdot \max\{\beta(w) - w, w - \alpha(w)\}\}$ is increasing, and the function $w \mapsto \tilde{\delta}(w)/w - 6A$ has at most one zero. Then for all n, $p \in \mathbb{N}$, given n tasks and p processors, the algorithm* BAL(ϱ_1, ϱ_2) *with $\varrho_1\colon x \mapsto \lfloor (\mathrm{id} + \tilde{\delta})^{-1}(x) \rfloor$ and $\varrho_2\colon x \mapsto \lfloor \beta^{-1}(x/A + \beta(w_{\min})) \rfloor$ produces a schedule $\mathcal{S}$ with the property that*

$$\mathrm{waste}(\mathcal{S}) = \mathcal{O}\left((h + \varepsilon) \cdot \gamma^*_{w_{\min}}(n/p) + \beta(w_{\min})\right),$$

where $\gamma_{w_{\min}}$ is the progess rate associated with $[\alpha, \beta]$ and $w_{\min}$, and for some partition

$\mathcal{S} = \mathcal{S}_1 \dot\cup \mathcal{S}_2 \dot\cup \mathcal{S}_3$, $\varepsilon = (h+\varepsilon_1)\cdot(h+\varepsilon_2)\cdot(h+\varepsilon_3)/h^2 - h \le (h+\varepsilon_1+\varepsilon_2+\varepsilon_3)^3/h^2 - h$, *where for* $i = 1, 2, 3$,

$$\varepsilon_i = (\text{sum-early}_\alpha(\mathcal{S}_i) + \text{sum-late}_\beta(\mathcal{S}_i) + (p-2)\cdot \text{max-late}_\beta(\mathcal{S}_i))/\text{chunks}(\mathcal{S}_i).$$

Remark. Note that each of the ε_i is somewhere between am-dev$_{\alpha,\beta}(\mathcal{S}_i)$ and av-dev$_{\alpha,\beta}$ $(\mathcal{S}_i)$, but typically closer to the former. Formally, the term for ε is incomparable with either of am-dev$_{\alpha,\beta}(\mathcal{S})$ or av-dev$_{\alpha,\beta}(\mathcal{S})$. However, as will become clear in the analysis, for practical purposes we can assume that $\varepsilon \approx (\text{am-dev}_{\alpha,\beta}(\mathcal{S}))^3/h^2$.

The proof of Theorem 3.2 is quite involved, so that we organized it into a number of self-contained modules. As a preparation, Section 3.3.1 introduces the symbols used in the proof. Section 3.3.2 establishes a number of basic properties of the schedule produced in the first phase, corresponding to what was shown above in the absence of busyness and laziness. Bounds on the latter are provided in Section 3.3.3. Building on this, Sections 3.3.4 and 3.3.5 derive bounds on the total overhead and idle time, from which the final section, Section 3.3.6, derives the desired bound on the wasted time.

3.3.1. *Terminology.* Let $\mathcal{S}_\text{I}$ and $\mathcal{S}_\text{II}$ denote the two parts of $\mathcal{S}$ pertaining to the chunks scheduled in the first and second phase, respectively. Let r denote the number of rounds in the first phase, that is, the number of executions of lines (2)–(5) except the last. Then, as before, W_i, w_i, d_i, t_i denote the values of BAL's program variables W, w, d, t in the various rounds, and for $i = 1, \dots, r$, $t_i^{\text{end}} = t_i - d_i$, $t_i^{\text{low}} = t_i - d_i/2$, and $t_i^{\text{upp}} = t_i + h + d_i/2$. It will further be convenient to have $d_0 = t_0^{\text{upp}} = 0$, and to denote by both W_{r+1} and n' the number of unassigned tasks when the second phase begins. Besides, we define $\tilde\gamma = \max\{0, \text{id} - \max\{w_{\min}, \varrho_1\}\} = \max\{0, \text{id} - \max\{w_{\min}, \lfloor(\text{id} + \tilde\delta)^{-1}\rfloor\}\}$, and recall that, by the definition of ϱ_1 and because of lines (2)–(4), w_i and d_i are at least $w_{\min}$, and thus $\tilde\gamma(W_i/p) = W_i/p - w_i = Kd_i$, for $i = 1, \dots, r$. Also note that, by Lemma 3.4, $\tilde\gamma$ is an increasing function.

In order to make the notions of *busyness* and *laziness* precise, we write $\mathcal{R}_i$ for the subschedule pertaining to the chunks assigned in round i; clearly, then $\mathcal{S}_\text{I} = \mathcal{R}_1 \cup \cdots \cup \mathcal{R}_r$. If a processor is assigned chunks $\mathcal{C}_1, \dots, \mathcal{C}_l$ in round i, its busyness in that round is defined as the total size of $\mathcal{C}_2, \dots, \mathcal{C}_l$, which is zero for $l \le 1$. The total busyness of all processors in round i is denoted by busy$(\mathcal{R}_i)$, and busy$(\mathcal{S}_\text{I})$ = busy$(\mathcal{R}_1) + \cdots +$ busy$(\mathcal{R}_r)$. The laziness of a processor in a round is zero for the first round, and $\max\{0, w_i - 2.5d_{i-1} - s\}$ for round i, $2 \le i \le r$, where s is the size of the first chunk assigned to that processor in round i, that is, $s = \min\{w_i, \lfloor t_i - T'\rfloor\}$ if that chunk is scheduled at time T' (see line (7)), or $s = 0$ if the processor does not request any chunk at all in that round. The total laziness of all processors in round i is denoted by lazy$(\mathcal{R}_i)$, for $i = 1, \dots, r$, and lazy$(\mathcal{S}_\text{I})$ = lazy$(\mathcal{R}_1) + \cdots +$ lazy$(\mathcal{R}_r)$.

3.3.2. *Local Properties of a Round.* This section provides a number of simple properties of $\mathcal{S}_\text{I}$, which constitute the basic building blocks of the further analysis. In fact, Lemmas 3.8–3.10 below correspond to what was shown above in our informal description of BAL for the deviationless case, except that there are now correcting terms involving $\text{busy}(\mathcal{R}_i)$ and $\text{lazy}(\mathcal{R}_i)$ for round i. Lemma 3.7 following says that the amount of time that the finishing time of a chunk $\mathcal{C}$, denoted by $\text{finish}(\mathcal{C})$, deviates from the tolerance thresholds of its round is bounded by what we defined as the chunk's earliness or lateness, respectively.

Lemma 3.7. *For every chunk $\mathcal{C}$ assigned in round i, with* $\text{finish}(\mathcal{C})$ *denoting the time when it is completed,*

$$t_i^{\text{low}} - \text{early}_\alpha(\mathcal{C}) \le \text{finish}(\mathcal{C}) \le t_i^{\text{upp}} + \text{late}_\beta(\mathcal{C}),$$

except that the first inequality does not necessarily hold if $\mathcal{C}$ is the very last chunk of $\mathcal{S}$.

Proof. We have already seen in the description of BAL that for a chunk $\mathcal{C}$ scheduled at time T' in round i and of size $w' = \min\{w_i, \lfloor t_i - T' \rfloor\}$,

$$t_i^{\text{low}} \le T' + h + \alpha(w') \le T' + h + \beta(w') \le t_i^{\text{upp}}.$$

Hence, with $\text{proc-time}(\mathcal{C})$ denoting the processing time of $\mathcal{C}$,

$$\begin{aligned} t_i^{\text{low}} - \alpha(w') + \text{proc-time}(\mathcal{C}) &\le T' + h + \text{proc-time}(\mathcal{C}) \\ &\le t_i^{\text{upp}} - \beta(w') + \text{proc-time}(\mathcal{C}), \end{aligned}$$

and according to the definitions made in Section 2.2, $\text{early}_\alpha(\mathcal{C}) = \max\{0, \alpha(w') - \text{proc-time}(\mathcal{C})\}$, $\text{late}_\beta(\mathcal{C}) = \max\{0, \text{proc-time}(\mathcal{C}) - \beta(w')\}$, and $\text{finish}(\mathcal{C}) = T' + h + \text{proc-time}(\mathcal{C})$. Finally, verify that in case $\mathcal{C}$ is the very last chunk of $\mathcal{S}$, thus scheduled in round r, its size might be smaller than $\min\{w_r, \lfloor t_r - T' \rfloor\}$ due to line (9), in which case the upper bound on $\text{finish}(\mathcal{C})$ holds all the more but not necessarily so the lower bound. □

Lemma 3.8. *For all $i \in [1..r]$, $W_i = p \cdot w_i + p \cdot Kd_i$, and it holds that $w_i \ge 5/K \cdot W_i/p$ and $Kd_i \le (1 - 5/K) \cdot W_i/p$.*

Proof. According to line (3) of the BAL code, $d_i = (W_i/p - w_i)/K$, and hence $W_i = p \cdot w_i + p \cdot Kd_i$. According to line (4) and because $K \ge 30$, $d_i \le w_i/6 \le (\frac{1}{5} - 1/K) \cdot w_i$, which is equivalent to $(W_i/p - w_i)/K \le (\frac{1}{5} - 1/K) \cdot w_i$, which implies the two inequalities stated in the lemma. □

Lemma 3.9. *For all $i \in [1..r]$, $W_{i+1} \ge p \cdot Kd_i - \text{busy}(\mathcal{R}_i)$.*

Proof. The total size of all chunks assigned to a processor in round i is the size of the first chunk, which is at most w_i, plus the busyness of that processor in round i. The total size of all chunks assigned in round i is hence at most $p \cdot w_i + \text{busy}(\mathcal{R}_i)$, and since, by the previous lemma, $W_i = p \cdot w_i + p \cdot Kd_i$, it follows that $W_{i+1} \ge p \cdot Kd_i - \text{busy}(\mathcal{R}_i)$. □

Lemma 3.10. *For all $i \in [1\,..\,r-1]$,*

(a) $t_{i+1}^{\text{end}} - t_i^{\text{end}} \geq 0$;
(b) $t_{i+1}^{\text{end}} - t_i^{\text{upp}} \geq w_{i+1} - 2.5d_i$;
(c) $t_{i+1}^{\text{upp}} - t_i^{\text{upp}} \geq d_i - \text{busy}(\mathcal{R}_i)/(3p)$.

Proof. Because of line (1), round $i+1$ cannot begin before the end of round i, so that

$$t_{i+1} \geq t_i^{\text{end}} + h + w_{i+1}.$$

Concerning (a), we have $t_{i+1}^{\text{end}} = t_{i+1} - d_{i+1} \geq t_i^{\text{end}} + h + w_{i+1} - d_{i+1}$, and according to line (4), $w_{i+1} \geq 6d_{i+1}$.

Concerning (b), it holds that $t_{i+1}^{\text{end}} \geq t_i^{\text{end}} + h + w_{i+1} - d_{i+1} = t_i^{\text{upp}} - 1.5d_i + w_{i+1} - d_{i+1}$, and, by the monotonicity of $\tilde{\gamma}$, $d_{i+1} = \tilde{\gamma}(W_{i+1}/p)/K \leq \tilde{\gamma}(W_i/p)/K = d_i$.

Concerning (c), we have $t_{i+1}^{\text{upp}} \geq t_{i+1} \geq t_i^{\text{end}} + h + w_{i+1} = t_i^{\text{upp}} - 1.5d_i + w_{i+1}$, and owing to Lemmas 3.8 and 3.9, $w_{i+1} \geq 5/K \cdot W_{i+1}/p \geq 5d_i - \text{busy}(\mathcal{R}_i)/(3p)$. □

Lemma 3.11. *For all $i \in [1\,..\,r]$, $W_{i+1} \leq p \cdot Kd_i + p \cdot 2.5d_{i-1} + \text{lazy}(\mathcal{R}_i)$.*

Proof. If $i = 1$, clearly $W_2 = W_1 - p \cdot w_1 = Kd_2$, whereas if $i = r$ and $W_{r+1} = 0$, then there is nothing to show. Otherwise, $i \geq 2$ and the size of each chunk assigned in round i is exactly the value assigned in line (7), so that, by the definition of laziness, the total size of these chunks is at least $p \cdot (w_i - 2.5d_{i-1}) - \text{lazy}(\mathcal{R}_i)$. By Lemma 3.8, we have $W_i = p \cdot w_i + p \cdot Kd_i$, and thus $W_{i+1} \leq p \cdot Kd_i + p \cdot 2.5d_{i-1} + \text{lazy}(\mathcal{R}_i)$. □

Lemma 3.12. *For $i \in [1\,..\,r-2]$, $W_{i+3}/p \leq \tilde{\gamma}(W_i/p) + \text{lazy}(\mathcal{R}_{i+1} \cup \mathcal{R}_{i+2})/p$.*

Proof. By two applications of the previous lemma and using Lemma 3.8,

$$\begin{aligned} W_{i+3}/p &\leq Kd_{i+2} + 2.5d_{i+1} + \text{lazy}(\mathcal{R}_{i+2})/p \\ &\leq (1 - 5/K) \cdot W_{i+2}/p + 2.5d_{i+1} + \text{lazy}(\mathcal{R}_{i+2})/p \\ &\leq (K-5) \cdot d_{i+1} + 2.5d_i + \text{lazy}(\mathcal{R}_{i+1})/p + 2.5d_{i+1} + \text{lazy}(\mathcal{R}_{i+2})/p \\ &\leq Kd_i + \text{lazy}(\mathcal{R}_{i+2})/p + \text{lazy}(\mathcal{R}_{i+1})/p \\ &= \tilde{\gamma}(W_i/p) + \text{lazy}(\mathcal{R}_{i+1} \cup \mathcal{R}_{i+2})/p. \end{aligned}$$ □

Lemma 3.13. *For $i \in [1\,..\,r]$, $W_{i+1}/p \leq \tilde{\gamma}^{(\lfloor i/3 \rfloor)}(n/p) + \text{lazy}(\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_i)/p$.*

Proof. First check that for all $x, y \geq 0$, because $\varrho_1 = \lfloor (\text{id} + \tilde{\delta})^{-1} \rfloor$ is an increasing function,

$$\begin{aligned} \tilde{\gamma}(x+y) &= x + y - \max\{w_{\min}, \varrho_1(x+y)\} \\ &\leq x + y - \max\{w_{\min}, \varrho_1(x)\} = \tilde{\gamma}(x) + y. \end{aligned}$$

Using this property the claim follows by a simple induction making use of the previous lemma. □

3.3.3. *Bounding Busyness and Laziness.* The following two lemmas relate the busyness and laziness of the schedule $\mathcal{S}_\text{I}$ to its total earliness and lateness.

Lemma 3.14. *For all $i \in [1\,..\,r]$*, $\text{busy}(\mathcal{R}_i) \le 2 \cdot \text{sum-early}_\alpha(\mathcal{R}_i)$.

Proof. Let $\mathcal{C}_1, \ldots, \mathcal{C}_l$ denote the chunks successively assigned to a fixed processor in round i, where possibly $l = 0$. By Lemma 3.7, we have that for $j = 1, \ldots, l-1$, $\text{early}_\alpha(\mathcal{C}_j) \ge t_i^\text{low} - \text{finish}(\mathcal{C}_j)$. Besides, the right-hand side is at least $d_i/2$, since by the condition in line (1) all of $\mathcal{C}_1, \ldots, \mathcal{C}_{l-1}$ finish before $t_i^\text{end} = t_i^\text{low} - d_i/2$. Hence, for all $j = 1, \ldots, l-1$,

$$2 \cdot \text{early}_\alpha(\mathcal{C}_j) \ge t_i^\text{low} - \text{finish}(\mathcal{C}_j) + d_i/2 = t_i - \text{finish}(\mathcal{C}_j) \ge \lfloor t_i - \text{finish}(\mathcal{C}_j) \rfloor,$$

where, according to line (7), and since $\text{finish}(\mathcal{C}_j)$ is at least h after the beginning of the round, the last term is just the size of $\mathcal{C}_{j+1}$. Consequently, the busyness of the considered processor in round i, which is just the total size of $\mathcal{C}_2, \ldots, \mathcal{C}_l$, is bounded by $2 \cdot \text{sum-early}_\alpha(\{\mathcal{C}_1, \ldots, \mathcal{C}_{l-1}\})$, and the lemma follows. ☐

Lemma 3.15. $\text{lazy}(\mathcal{S}_\text{I}) \le \text{sum-late}_\beta(\mathcal{S}_\text{I}) + \text{sum-early}_\alpha(\mathcal{S}_\text{I})$.

Proof. This proof is a bit longer, so we give a plan. We first focus on the laziness of a single processor in a single round. This will help us investigate the laziness caused by a single chunk (in possibly many rounds), after which it will be straightforward to bound the total laziness of a processor and finally of the whole schedule.

There is never laziness in the first round, so we consider a fixed processor in a round i, where $2 \le i \le r$. Let T denote the finishing time of the last chunk assigned to the processor before round i, and write L for the processor's laziness in that round, which we defined as $\max\{0, w_i - 2.5d_{i-1} - s\}$, where $s = \min\{w_i, \lfloor t_i - T \rfloor\}$ if $T < t_i^\text{end}$, and $s = 0$ otherwise. For $T < t_i^\text{end}$, therefore

$$\begin{aligned} L &= \max\{0, w_i - 2.5d_{i-1} - \min\{w_i, \lfloor t_i - T \rfloor\}\} \\ &\le \max\{0, \max\{-2.5d_{i-1}, w_i - 2.5d_{i-1} - t_i + T + 1\}\} \\ &\le \max\{0, T + w_i - 2.5d_{i-1} - t_i^\text{end}\} \\ &\le \max\{0, T - t_{i-1}^\text{upp}\}, \end{aligned}$$

where the penultimate inequality uses that $t_i^\text{end} = t_i - d_i \le t_i - 1$, and the last inequality follows by Lemma 3.10(b). For $T \ge t_i^\text{end}$, on the other hand, $L = \max\{0, w_i - 2.5d_{i-1}\}$, which, again by Lemma 3.10(b), implies that

$$L \le \max\{0, t_i^\text{end} - t_{i-1}^\text{upp}\}.$$

In any case therefore

$$L \le \max\{0, \min\{t_i^\text{end}, T\} - t_{i-1}^\text{upp}\} = |[0, T] \cap [t_{i-1}^\text{upp}, t_i^\text{end}]|,$$

that is, the laziness of the considered processor in round i is bounded by the part of T that lies in the (possibly empty) interval $[t_{i-1}^{\text{upp}}, t_i^{\text{end}}]$.

We are now ready to bound the total laziness incurred by a single fixed chunk $\mathcal{C}$ scheduled in some round j. For that purpose define j' as the index of that round after round j, in which the next chunk is assigned to the same processor, or $j' = r$, if $\mathcal{C}$ is the last chunk of that processor. Then, according to what was shown in the last paragraph, the laziness caused by $\mathcal{C}$ is at most the part of $[0, \text{finish}(\mathcal{C})]$ that lies in the intervals $[t_j^{\text{upp}}, t_{j+1}^{\text{end}}], \ldots, [t_{j'-1}^{\text{upp}}, t_{j'}^{\text{end}}]$, which, by Lemma 3.10(a), are disjoint (note that some of them may be empty). This quantity is at most $\max\{0, \text{finish}(\mathcal{C}) - \min\{t_j^{\text{upp}}, \ldots, t_{j'-1}^{\text{upp}}\}\}$, and, by Lemma 3.7, $\text{finish}(\mathcal{C}) - t_j^{\text{upp}} \le \text{late}_\beta(\mathcal{C})$, while, by Lemma 3.10(c),

$$\begin{aligned} t_j^{\text{upp}} - \min\{t_j^{\text{upp}}, \ldots, t_{j'-1}^{\text{upp}}\} &\le \sum_{i=j}^{j'-2} \max\{0, t_i^{\text{upp}} - t_{i+1}^{\text{upp}}\} \\ &\le \sum_{i=j}^{j'-2} \text{busy}(\mathcal{R}_i)/(3p). \end{aligned}$$

We conclude that the laziness of a fixed processor is bounded by the total lateness of all its chunks plus $\text{busy}(\mathcal{S}_{\text{I}})/(3p)$. Summing over all processors, and bounding busyness with the previous lemma, we obtain

$$\text{lazy}(\mathcal{S}_{\text{I}}) \le \text{sum-late}_\beta(\mathcal{S}_{\text{I}}) + \text{busy}(\mathcal{S}_{\text{I}})/3 \le \text{sum-late}_\beta(\mathcal{S}_{\text{I}}) + \text{sum-early}_\alpha(\mathcal{S}_{\text{I}}). \qquad \square$$

3.3.4. *The Scheduling Overhead.* In order to bound the total number of chunks scheduled, we consider the following partition of $\mathcal{S}$, where $r' = \min\{r, 3 \cdot \tilde{\gamma}^*(n/p)\}$:

- $\mathcal{S}_1$ contains those chunks of $\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_{r'}$ that are the first in the round assigned to their processor, so that, in particular, $\text{chunks}(\mathcal{S}_1) \le p \cdot r'$;
- $\mathcal{S}_2$ contains the chunks accounting for the busyness of the first r' rounds, that is, $\mathcal{S}_2 = (\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_{r'}) - \mathcal{S}_1$;
- $\mathcal{S}_3$ contains all the chunks scheduled after round r', that is, $\mathcal{S}_3 = \mathcal{S} - (\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_{r'}) = \mathcal{S} - (\mathcal{S}_1 \cup \mathcal{S}_2)$.

We separately bound the number of chunks in each of these subschedules, for which we have to distinguish between two cases. Note that, unless excessive lateness causes many more rounds to be executed than would be the case without deviations, we have $r' = r$, and hence $\mathcal{S}_1 \cup \mathcal{S}_2 = \mathcal{S}_{\text{I}}$ and $\mathcal{S}_3 = \mathcal{S}_{\text{II}}$.

The regular case: $r' = r$ *and* $n' > \text{lazy}(\mathcal{S}_{\text{I}})$. In the regular case, $\mathcal{S}_1 \cup \mathcal{S}_2 = \mathcal{S}_{\text{I}}$ and $\mathcal{S}_3 = \mathcal{S}_{\text{II}} \ne \emptyset$, so that

$$\text{chunks}(\mathcal{S}) \le p \cdot r + \text{busy}(\mathcal{S}_{\text{I}})/w_{\min} + \text{chunks}(\mathcal{S}_{\text{II}}).$$

In order to bound r, we can employ Lemma 3.2 to derive from Lemma 3.13 that

$$\lfloor r/3 \rfloor \le \tilde{\gamma}^*(n/p) - \tilde{\gamma}^*(n'/p - \text{lazy}(\mathcal{S}_{\text{I}})/p),$$

so that owing to $\lfloor r/3 \rfloor \ge (r-2)/3$ and $\tilde{\gamma}^*(n'/p - \text{lazy}(\mathcal{S}_{\text{I}})/p) \ge 1$, and with the help of Lemma 3.3,

$$\begin{aligned} r &\le 3 \cdot \tilde{\gamma}^*(n/p) - 3 \cdot \tilde{\gamma}^*(n'/p - \text{lazy}(\mathcal{S}_{\text{I}})/p) + 2 \\ &\le 3 \cdot \tilde{\gamma}^*(n/p) - \tilde{\gamma}^*(n'/p - \text{lazy}(\mathcal{S}_{\text{I}})/p) \\ &\le 3 \cdot \tilde{\gamma}^*(n/p) - \tilde{\gamma}^*(n'/p) + \lceil \text{lazy}(\mathcal{S}_{\text{I}})/(pw_{\min}) \rceil. \end{aligned}$$

We further have to bound chunks$(\mathcal{S}_{\text{II}})$, and for that purpose recall that in the second phase of BAL chunks are scheduled according to FP$(x \mapsto \lfloor \beta^{-1}(x/A + \beta(w_{\min})) \rfloor)$. Theorem 3.1 is therefore applicable, and we want to make use of its addendum in order to obtain a bound in terms of $\tilde{\gamma} = \text{id} - \max\{w_{\min}, \lfloor (\text{id} + \tilde{\delta})^{-1} \rfloor\}$. To this end, observe that by the condition on which the first phase is terminated, $d_{r+1} \ge w_{r+1}/6$ where $w_{r+1} = \varrho_1(W_{r+1}/p) = \lfloor (\text{id} + \tilde{\delta})^{-1}(n'/p) \rfloor$. Then take $w'_{r+1} = (\text{id} + \tilde{\delta})^{-1}(n'/p)$, for which clearly $w'_{r+1} \le w_{r+1} + 1$ and hence $\tilde{\delta}(w'_{r+1}) \ge K d_{r+1} - 1$. Using that $K \ge 49A$, we can then conclude from the inequality $K d_{r+1} \ge K/6 \cdot w_{r+1}$ derived above that $\tilde{\delta}(w'_{r+1}) \ge 6A \cdot w'_{r+1}$, and hence, by the condition on $\tilde{\delta}$ imposed by Theorem 3.2, $\tilde{\delta}(w) \ge 6A \cdot w$, for all $w \le \lfloor (\text{id} + \tilde{\delta})^{-1}(n'/p) \rfloor$. On the other hand, we easily verify by means of the definition of $\tilde{\delta}$ that $\tilde{\delta} \ge K \cdot (\beta - \alpha) \ge 6A \cdot (\beta - \alpha)$. Therefore $\tilde{\delta}$ fulfills the requirements of (the addendum to) Theorem 3.1, and we obtain that

$$\text{chunks}(\mathcal{S}_{\text{II}}) \le p \cdot \tilde{\gamma}^*(n'/p).$$

Plugging this and the bound on r derived before into the bound on chunks$(\mathcal{S})$ established at the beginning of the paragraph, we obtain that, for the regular case,

$$\text{chunks}(\mathcal{S}) \le 3p \cdot \tilde{\gamma}^*(n/p) + \text{busy}(\mathcal{S}_{\text{I}})/w_{\min} + \text{lazy}(\mathcal{S}_{\text{I}})/w_{\min} + p.$$

The irregular case: *$r' < r$ or $W_{r+1} \le \text{lazy}(\mathcal{S}_{\text{I}})$.* Intuitively, the irregular case occurs when excessive lateness of chunks causes many more rounds to be executed than in the deviationless case. If $r' = r$, then the case condition ensures that $W_{r'+1} = W_{r+1} = n' \le \text{lazy}(\mathcal{S}_{\text{I}}) = \text{lazy}(\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_{r'})$. If $r' < r$, then $r' = 3 \cdot \tilde{\gamma}^*(n/p)$ and hence $\tilde{\gamma}^{(\lfloor r'/3 \rfloor)}(n/p) \le 0$, so that, by Lemma 3.13, $W_{r'+1} \le \text{lazy}(\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_{r'})$, too. It follows that the total size of the chunks in $\mathcal{S}_2$ and $\mathcal{S}_3$ is at most busy$(\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_{r'})$ + lazy$(\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_{r'})$, and since each chunk, except maybe the very last, is of size at least $w_{\min}$, we have

$$\begin{aligned} \text{chunks}(\mathcal{S}) &\le p \cdot r' + \lceil (\text{busy}(\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_{r'}) + \text{lazy}(\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_{r'}))/w_{\min} \rceil \\ &\le 3p \cdot \tilde{\gamma}^*(n/p) + \text{busy}(\mathcal{S}_{\text{I}})/w_{\min} + \text{lazy}(\mathcal{S}_{\text{I}})/w_{\min} + p, \end{aligned}$$

just as for the regular case.

Using the bounds on busyness and laziness established in the previous section, we may finally conclude that, in any case,

$$\text{chunks}(\mathcal{S}) \le 3p \cdot \tilde{\gamma}^*(n/p) + 3 \cdot \text{sum-early}_\alpha(\mathcal{S})/w_{\min} + \text{sum-late}_\beta(\mathcal{S})/w_{\min} + p.$$

Now recall that $\tilde{\gamma} = \max\{0, \text{id} - \max\{w_{\min}, \lfloor (\text{id} + \tilde{\delta})^{-1} \rfloor\}\}$, while our goal is to bound the wasted time in terms of $\gamma_{w_{\min}} = \max\{0, \text{id} - \max\{w_{\min}, (\text{id} + \delta)^{-1}\}\}$, where $\delta = \alpha^{-1} \circ (\beta - \alpha)$. Since we defined

$$\tilde{\delta}(w) = K \cdot \max\{w_{\min}, 2 \cdot \max\{\beta(w) - w, w - \alpha(w)\}\},$$

and because

$$\max\{\beta(w) - w, w - \alpha(w)\} \leq \beta(w) - \alpha(w) \leq \delta(w),$$

an elegant sequence of applications of Lemmas 3.5, 3.6, 3.5, and 3.1, shows that $\tilde{\gamma}^* \leq 2K \cdot \gamma^*_{w_{\min}}$. We thus obtain

$$\text{chunks}(\mathcal{S}) \leq 6K \cdot p \cdot \gamma^*_{w_{\min}}(n/p) \\ + 3 \cdot \text{sum-early}_\alpha(\mathcal{S})/w_{\min} + \text{sum-late}_\beta(\mathcal{S})/w_{\min} + p.$$

At this point we feel the need to stress that here is the only place in our whole analysis where an unrealistically large constant, namely, $K = 49A$, has come into play. However, as we have mentioned before, in the description of BAL, this value of K has merely been chosen in order to avoid a number of extremely tedious technical complications, while actually $K = A$ would also suffice.

3.3.5. *The Idle Time.* We first bound the idle time of $\mathcal{S}_{\text{I}}$, which is also the initial imbalance of the schedule $\mathcal{S}_{\text{II}}$ produced in the second phase of BAL. Let r' be the index of a round with maximal upper limit, that is, $t^{\text{upp}}_{r'} \geq t^{\text{upp}}_i$, for $i = 1, \ldots, r$. Typically $r' = r$, but an extreme pattern of deviations may even cause the upper limit of the first round to be largest. We split idle($\mathcal{S}_{\text{I}}$) into three parts, I', I'', and I''', namely, the amount of idle time of $\mathcal{S}_{\text{I}}$ spent before $t^{\text{end}}_{r'}$, between $t^{\text{end}}_{r'}$ and $t^{\text{upp}}_{r'}$, and after $t^{\text{upp}}_{r'}$, respectively. We first bound each of these quantities separately.

We start with I', which is the hard part. By Lemma 3.10(a), $t^{\text{end}}_{r'} \leq t^{\text{end}}_r \leq t^{\text{low}}_r$, so that it suffices to bound the total amount of time that processors finish before t^{low}_r, the lower tolerance threshold of the last round. For processors to which a chunk that is not the very last is assigned in round r, Lemma 3.7 says that this amount is bounded by the earliness of the last such chunk. It may happen, however, that a processor is *deceived* in that it is either assigned no chunk at all in the last round, or only the very last chunk, whose size might be reduced due to line (9) of the BAL code. Since, by Lemma 3.8, $W_r > p \cdot w_r$, at least $p + 1$ chunks are scheduled in round r, and at most one of these (the very last) to a deceived processor. If there are p' deceived processors, p' of the at least $p + 1$ chunks scheduled in round r must be *intermediate*, that is, are followed by another chunk that is not the very last and assigned to the same processor. When these intermediate chunks finish, there are still enough tasks left, hence we know that all of the deceived processors finish later than the intermediate chunks. The contribution of the deceived processors to I' is hence bounded by the earliness of the intermediate chunks, and we have already seen above that the amount of I' due to the other processors is bounded by the earliness of their last chunks. This proves

$$I' \leq \text{sum-early}_\alpha(\mathcal{S}_{\text{I}}).$$

Concerning I''', let $\mathcal{C}$ denote the last chunk to finish in $\mathcal{S}_{\text{I}}$, and note that its finishing time is just the makespan of $\mathcal{S}_{\text{I}}$. Now if i denotes the round in which $\mathcal{C}$ was scheduled, then, since round r' has the maximal upper tolerance threshold, $\text{finish}(\mathcal{C}) - t^{\text{upp}}_{r'} \leq \text{finish}(\mathcal{C}) - t^{\text{upp}}_i$, which by Lemma 3.7 is at most $\text{late}_\beta(\mathcal{C})$. The makespan of $\mathcal{S}_{\text{I}}$ is thus at most $t^{\text{upp}}_{r'} + \text{late}_\beta(\mathcal{C})$, and we conclude that

$$I''' \leq (p - 1) \cdot \text{max-late}_\beta(\mathcal{S}_{\text{I}}).$$

Finally, we may trivially bound the idle time I'' spent between $t_{r'}^{\text{end}}$ and $t_{r'}^{\text{upp}}$ by $p \cdot (t_{r'}^{\text{upp}} - t_{r'}^{\text{end}}) = p \cdot h + p \cdot 1.5 d_{r'}$. Now either $r' = r$, in which case we know from Lemma 3.9 that $p \cdot d_{r'} = p \cdot d_r \leq W_{r+1}/K + \text{busy}(\mathcal{R}_r)/K \leq n'/(1.5A) + \text{busy}(\mathcal{R}_r)/3$, or $r' < r$, and because round r' has the maximal upper tolerance threshold, $t_{r'}^{\text{upp}} \geq t_{r'+1}^{\text{upp}}$, so that, by Lemma 3.10(c), $d_{r'} \leq \text{busy}(\mathcal{R}_{r'})/(3p)$. In any case, therefore, $p \cdot 1.5 d_{r'} \leq n'/A + \text{busy}(\mathcal{R}_{r'})/2$, which by Lemma 3.14 is bounded by $n'/A + \text{sum-early}_\alpha(\mathcal{R}_{r'})$, and we have proven that

$$I'' \leq n'/A + p \cdot h + \text{sum-early}_\alpha(\mathcal{S}_{\text{I}}).$$

We have finally bounded each of I', I'', and I''', and $\text{idle}(\mathcal{S}_{\text{I}})$ is just the sum of them; therefore

$$\text{idle}(\mathcal{S}_{\text{I}}) \leq n'/A + p \cdot h + 2 \cdot \text{sum-early}_\alpha(\mathcal{S}_{\text{I}}) + (p-1) \cdot \text{max-late}_\beta(\mathcal{S}_{\text{I}}).$$

Now recall that $\text{idle}(\mathcal{S}_{\text{I}})$ is just the initial imbalance of $\mathcal{S}_{\text{II}}$, and the idle time of $\mathcal{S}_{\text{II}}$ is that of the whole schedule $\mathcal{S}$. Using Theorem 3.1 we therefore obtain that

$$\begin{aligned}\text{idle}(\mathcal{S}) \leq\ & p \cdot h + p \cdot \beta(w_{\min}) \\ & + \max\{0, \text{idle}(\mathcal{S}_{\text{I}}) - n'/A - p \cdot h\} \\ & + \text{sum-early}_\alpha(\mathcal{S}_{\text{II}}) + (p-1) \cdot \text{max-late}_\beta(\mathcal{S}_{\text{II}}),\end{aligned}$$

and by the bound on $\text{idle}(\mathcal{S}_{\text{I}})$ just established,

$$\text{idle}(\mathcal{S}) \leq p \cdot h + p \cdot \beta(w_{\min}) + 2 \cdot (\text{sum-early}_\alpha(\mathcal{S}) + (p-1) \cdot \text{max-late}_\beta(\mathcal{S})).$$

3.3.6. *The Wasted Time.* In the last two sections, we have proven that

$$\begin{aligned}\text{chunks}(\mathcal{S}) &\leq 6K \cdot p \cdot \gamma^*_{w_{\min}}(n/p) \\ &\quad + 3 \cdot \text{sum-early}_\alpha(\mathcal{S})/w_{\min} + \text{sum-late}_\beta(\mathcal{S})/w_{\min} + p, \\ \text{idle}(\mathcal{S}) &\leq p \cdot h + p \cdot \beta(w_{\min}) + 2 \cdot (\text{sum-early}_\alpha(\mathcal{S}) + (p-1) \cdot \text{max-late}_\beta(\mathcal{S})),\end{aligned}$$

which, since $w_{\min} \geq h$, immediately implies that for $\mathcal{E} = \text{sum-early}_\alpha(\mathcal{S}) + \text{sum-late}_\beta(\mathcal{S}) + (p-2) \cdot \text{max-late}_\beta(\mathcal{S})$,

$$\text{waste}(\mathcal{S}) = \mathcal{O}\left(h \cdot \gamma^*_{w_{\min}}(n/p) + \beta(w_{\min}) + \mathcal{E}/p\right).$$

This almost proves Theorem 3.2, except that it remains to bound $\mathcal{E}$ in terms of quantities ε_1, ε_2, and ε_3 according to the theorem. Note, at this point, that even though $\mathcal{E} \leq \text{av-dev}_{\alpha,\beta}(\mathcal{S}) \cdot \text{chunks}(\mathcal{S})$, we cannot bound $\mathcal{E}$ in terms of $\text{av-dev}_{\alpha,\beta}(\mathcal{S})$ (not to mention $\text{am-dev}_{\alpha,\beta}(\mathcal{S})$), since the number of chunks in $\mathcal{S}$ is not bounded independently of $\mathcal{E}$. This, in turn, is due to the inherent feature of BAL that it dynamically adjusts the number of chunks it assigns to the earliness or lateness of previous chunks.

As in Section 3.3.4, define $R = 3 \cdot \tilde{\gamma}^*(n/p)$ and $r' = \min\{r, R\}$, and consider the partition $\mathcal{S}_1 \dot\cup \mathcal{S}_2 \dot\cup \mathcal{S}_3$ of $\mathcal{S}$ defined in that section. Correspondingly, define for $i = 1, 2, 3$,

$$\mathcal{E}_i = \text{sum-early}_\alpha(\mathcal{S}_i) + \text{sum-late}_\beta(\mathcal{S}_i) + (p-2) \cdot \text{max-late}_\beta(\mathcal{S}_i),$$

let $\varepsilon_i = \mathcal{E}_i/\text{chunks}(\mathcal{S}_i)$, and note that $\mathcal{E} \leq \mathcal{E}_1 + \mathcal{E}_2 + \mathcal{E}_3$.

In Section 3.3.4 we have shown that $\text{chunks}(\mathcal{S}_1) \le p \cdot r' \le pR$, which implies

$$\mathcal{E}_1 = \varepsilon_1 \cdot \text{chunks}(\mathcal{S}_1) \le \varepsilon_1 \cdot pR.$$

In the following we want to bound $\mathcal{E}_2$ in terms of ε_1 and ε_2, and $\mathcal{E}_3$ in terms of ε_1, ε_2, and ε_3. According to Section 3.3.4, the chunks of $\mathcal{S}_2$ are exactly those accounting for the busyness of rounds 1 through r'. Since obviously the scheduling times of two chunks assigned to the same processor are at least h apart, and since, by Lemma 3.7, a chunk finishes at most its earliness before the end of the round in which it was assigned, it follows that $\text{chunks}(\mathcal{S}_2) \le \lceil \text{sum-early}_\alpha(\mathcal{S}_1)/h \rceil \le \mathcal{E}_1/h + 1 \le (1 + \varepsilon_1/h) \cdot pR$, and hence

$$\mathcal{E}_2 = \varepsilon_2 \cdot \text{chunks}(\mathcal{S}_2) \le \varepsilon_2 \cdot (1 + \varepsilon_1/h) \cdot pR.$$

Concerning $\mathcal{S}_3$, we have to distinguish between the regular and the irregular case just as in Section 3.3.4. In the regular case we have seen that $\text{chunks}(\mathcal{S}_3) = \text{chunks}(\mathcal{S}_{\text{II}}) \le p \cdot \tilde{\gamma}^*(n'/p)$, which is clearly bounded by pR. For the irregular case we have proven that $\text{chunks}(\mathcal{S}_3) \le \lceil \text{lazy}(\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_{r'})/w_{\min} \rceil$, which by Lemma 3.15 is at most $(\mathcal{E}_1 + \mathcal{E}_2)/h + 1$. In any case, therefore, $\text{chunks}(\mathcal{S}_3) \le \mathcal{E}_1/h + \mathcal{E}_2/h + pR$, and hence

$$\mathcal{E}_3 = \varepsilon_3 \cdot \text{chunks}(\mathcal{S}_3) \le \varepsilon_3 \cdot (\varepsilon_1/h + \varepsilon_2/h \cdot (1 + \varepsilon_1/h) + 1) \cdot pR.$$

Having thus bounded each of $\mathcal{E}_1$, $\mathcal{E}_2$, and $\mathcal{E}_3$, we immediately obtain that

$$\begin{aligned}\mathcal{E} &\le \mathcal{E}_1 + \mathcal{E}_2 + \mathcal{E}_3 \\ &\le \left(\varepsilon_1 + \varepsilon_2 + \varepsilon_3 + \varepsilon_2 \cdot \varepsilon_1/h + \varepsilon_3 \cdot \varepsilon_1/h + \varepsilon_3 \cdot \varepsilon_2/h + \varepsilon_3 \cdot \varepsilon_2 \cdot \varepsilon_1/h^2\right) \cdot pR \\ &= \left((h + \varepsilon_1) \cdot (h + \varepsilon_2) \cdot (h + \varepsilon_3)/h^2 - h\right) \cdot pR.\end{aligned}$$

Hence with $\varepsilon = (h + \varepsilon_1) \cdot (h + \varepsilon_2) \cdot (h + \varepsilon_3)/h^2 - h$ and using that $R = 3 \cdot \tilde{\gamma}^*(n/p) \le 6K \cdot \gamma^*_{w_{\min}}(n/p)$,

$$\mathcal{E}/p = \mathcal{O}\left(\varepsilon \cdot \tilde{\gamma}^*(n/p)\right) = \mathcal{O}\left(\varepsilon \cdot \gamma^*_{w_{\min}}(n/p)\right),$$

and finally note that, since the volume of a cuboid with fixed total edge length is maximal if all edges are equally long, we also have that $\varepsilon \le (h + \varepsilon')^3/h^2 - h$, where $\varepsilon' = (\varepsilon_1 + \varepsilon_2 + \varepsilon_2)/3$. This finishes the proof of Theorem 3.2. □

3.4. *A Variant of the Balancing Strategy*

This section is concerned with a variant of the balancing scheme, named BAL′, whose analysis will close the small gap left between the performance guarantee we could prove for BAL and the upper bound claimed in our Main Theorem. Very roughly speaking, BAL′ differs from BAL in that it does not try to compensate for deviations of chunk processing times but simply aggravates them over the rounds: for each unit of earliness of a chunk, BAL′ inserts a unit of waiting time (instead of assigning an intermediate chunk), and for each unit of lateness of a chunk, it lets all subsequent chunks for that processor start one time unit later (instead of decreasing their sizes). We will be able to prove that in this manner each processor is assigned exactly as many chunks in the first phase as there are rounds, and that, compared with the deviationless case, each unit of deviation simply adds to the idle time of the schedule produced in the first phase. As mentioned before, this behaviour simplifies the analysis a lot, but lacks the so practical feature of BAL that it can nullify the effect of small to moderate deviations. The significance of the BAL′ scheme is therefore of a more theoretical nature.

More precisely, BAL′ serves a processor request exactly as BAL, with two exceptions. First, when the requesting processor has already been assigned (and finished) a chunk in the current round, the assignment is delayed until the end of the round. At that time a new round will be started and the request is served as if it were issued then. Clearly, this guarantees that at most one chunk per round is assigned to each processor. The second exception occurs when for $i \geq 2$, a processor requests its ith chunk after the upper tolerance threshold t_{i-1}^{upp} of the $(i-1)$th round; note that, by the above, this request cannot occur before the $(i-1)$th round, so that this threshold is surely known. Then, irrespective of how long after t_{i-1}^{upp} the request is issued, the processor is assigned a chunk as if it had requested exactly at time t_{i-1}^{upp}, and also the corresponding update of W is performed at that time. It is convenient to say that the ith chunk of a processor *belongs* to round i, which makes sense, because even though such a chunk might be assigned long after the ith round is over, its size is computed from the settings of that round. The effect of the described modifications, compared with the orginal BAL strategy, is illustrated in Figure 7. Note that without deviations, BAL′ and BAL behave identically.

In analogy to the piece of code given for BAL, Figure 8 provides an implementation of the function that computes for a given request the size of the chunk to be assigned then. As before, the variables W, w, d, t keep track of the number of unassigned tasks, the size of the first chunk in the current round, and the tolerance and target of that round,

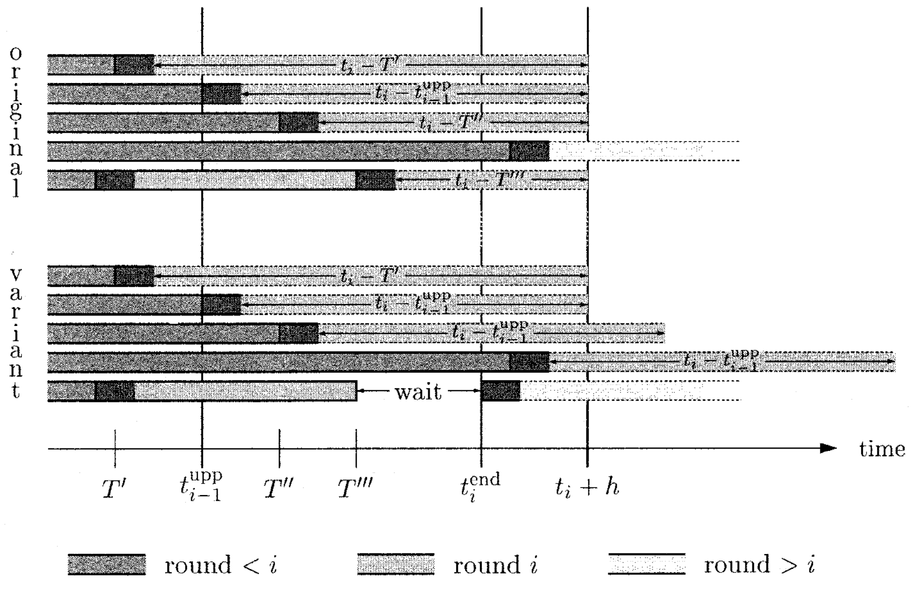

Fig. 7. The original BAL compared with its variant BAL′ in a round i.

```
(1)   $r[k] = r[k] + 1;$
(2)   if ( PHASE == 1 && $r[k] > R$ ) { (* new round *)
(3)       if ( $T < t - d$ ) wait until $t - d$;
(4)       $T = \max\{ t - d, \min\{ t + h + d/2, T \} \}$;
(5)       if ( $R > 0$ ) $W = W - p' \cdot s[R]$;
(6)       $p' = p$;
(7)       $w = \varrho_1(W/p)$;
(8)       $s[R+1] = \min\{ w, \lfloor T + w - t - d/2 \rfloor \}$;
(9)       $d = (W/p - w)/K$;
(10)      if ( $d > w/6$ ) PHASE = 2 else $R = R + 1$;
(11)      $t = T + h + w$;
(12)  }
(13)  if ( $r[k] < R$ ) { $s = s[r[k]]$; $W = W + s$; }
(14)  if ( $r[k] == R$ ) { $s = \min\{ w, \max\{ s[R], \lfloor t - T \rfloor \} \}$; $p' = p' - 1$; }
(15)  if ( $r[k] > R$ ) { $s = \varrho_2(W/p)$; }
(16)  $s = \min\{ W, s \}$;
(17)  $W = W - s$;
(18)  return $s$;
```

Fig. 8. The chunk size computed by BAL$'(\varrho_1, \varrho_2)$ for a request of the *k*th processor at time T.

respectively, with the latter three being initialized to zero. Again, the constants p and h hold the number of processors and the scheduling overhead, respectively, and the variable PHASE is initially set to one. Besides, BAL$'$ requires the following additional variables:

- An array $r[1\,..\,p]$ counting the number of chunks scheduled to each processor, initialized with zeros—this array can either be stored globally or in a distributed manner by the processors.
- A variable R keeping track of the index of the current round, initially zero.
- A dynamic array $s[\,]$, with $s[i]$ storing the size of a chunk belonging to round i and assigned after t_{i-1}^{upp}, to be used when round i is over.
- A variable p' that for each round keeps track of the number of processors that have not yet been assigned their chunk belonging to the current round.

For most parts, the code of Figure 8 is a straightforward implementation of BAL$'$ as described above, but we should comment on the subtleties. The assignment in line (4) ensures that after a waiting period the computation is continued with the appropriate time stored in T. Here the maximum construct ensures that T is at most the upper tolerance threshold of the previous round, in order to deal adequately with the special case where *all* processors finish their chunks from the previous round *after* that threshold. Concerning the assignment in line (8), note that $T + w - t - d/2$ is just the difference between the target going to be set for the current round, namely, $T + h + w$, and the upper tolerance of the previous round, namely, $t + h + d/2$. Finally, the correctness of the somewhat tricky update of W realized by lines (5), (6), (13), (14), and (17) will be implied by Lemma 3.16 below. In particular, the lemma implies that W is never assigned a negative value in line (5). The remainder of this section is concerned with proving the following result, which establishes our Main Theorem.

Theorem 3.3. *Let task processing times be arbitrary, and let the overhead be $h \geq 1$. Let $[\alpha, \beta]$ be a variance estimator, let $A \geq 1$ with $\alpha \geq \mathrm{id}/A$, and let $w_{\min} \in \mathbb{N}$, $w_{\min} \geq h$ such that, for $K = 49A$, $\tilde{\delta}\colon w \mapsto K \cdot \max\{w_{\min}, 2 \cdot \max\{\beta(w) - w, w - \alpha(w)\}\}$ is increasing, and the function $w \mapsto \tilde{\delta}(w)/w - 6A$ has at most one zero. Then for all $n, p \in \mathbb{N}$, given n tasks and p processors, the algorithm* $\textsc{Bal}'(\varrho_1, \varrho_2)$ *with $\varrho_1\colon x \mapsto \lfloor (\mathrm{id} + \tilde{\delta})^{-1}(x) \rfloor$ and $\varrho_2\colon x \mapsto \lfloor \beta^{-1}(x/A + \beta(w_{\min})) \rfloor$ produces a schedule $\mathcal{S}$ with the property that*

$$\mathrm{waste}(\mathcal{S}) = \mathcal{O}\Big((h + \varepsilon) \cdot \gamma^*_{w_{\min}}(n/p) + \beta(w_{\min})\Big),$$

where $\varepsilon = \text{av-dev}_{\alpha,\beta}(\mathcal{S})$, and $\gamma_{w_{\min}}$ is the progress rate associated with $[\alpha, \beta]$ and $w_{\min}$.

The proof is divided into four parts. Section 3.4.1 deals with the local properties of a round, Sections 3.4.2 and 3.4.3 are concerned with the number of scheduling operations and the idle time, respectively, from which Section 3.4.4 derives the desired bound on the average wasted time. With the experience of having gone through the analysis of Bal, the following should be fairly easy to follow, since the flow of argumentation is almost the same, except that now we are no longer bothered by busyness or laziness.

We make use of the very same notation as defined for the analysis of Bal in Section 3.3.1. In particular, r denotes the number of rounds, which is just the number of executions of lines (3)–(11) except the last, W_i, w_i, d_i, t_i denote the values of W, w, d, t, just after the ith execution of these lines, and $t_i^{\mathrm{low}} = t_i - d_i/2$, $t_i^{\mathrm{upp}} = t_i + d_i/2$, and $t_i^{\mathrm{end}} = t_i - d_i$, for $i = 1, \ldots, r$; by convention also $d_0 = t_0^{\mathrm{upp}} = 0$. The total number of tasks not assigned in the first phase is denoted by W_{r+1} and by n', and $\tilde{\gamma}$ is defined as $\max\{0, \mathrm{id} - \max\{w_{\min}, \lfloor (\mathrm{id} + \tilde{\delta})^{-1} \rfloor\}\}$.

3.4.1. *Local Properties of a Round.* In analogy to Section 3.3.2, this section provides the building blocks for further analysis of Bal′. Since for Bal′ there is nothing like busyness or laziness, we can now prove the valuable property that, for arbitrary deviations, a round always ends after the upper limit of the previous round. The first lemma demonstrates that the update of W in the code of Bal′ is performed correctly.

Lemma 3.16. *For $i = 1, \ldots, r$, the total size of all chunks belonging to round i is $W_i - W_{i+1}$.*

Proof. First verify that by lines (1), (2), and (10), either PHASE $= 2$ or $r[k] \leq R$ when lines (13)–(15) are executed. Therefore, if the assignment in line (15) is executed, we must have PHASE $= 2$ and hence also $R = r$, since the increment operation on R in line (10) can only be reached when PHASE $= 1$. This proves that the first r requests of a processor are scheduled according to line (13) or (14).

Now for fixed $i \in [1\,..\,r]$, let $w_{i,k}$ denote the size of the ith chunk assigned to the kth processor, for $k = 1, \ldots, p$, and denote by P' and P'' the sets of indices of processors for whose ith request the body of line (13) and the body of line (14), respectively, is executed. Then, by what was shown in the first paragraph, $P' \cup P'' = \{1, \ldots, p\}$. By lines (6) and (14), it is easy to see that at the beginning of the $(i+1)$th execution of lines (3)–(11), the value of p' is just $p - |P''| = |P'|$, and the value of W is $W_i - \sum_{k \in P''} w_{i,k}$.

Since $w_{i,k} = s[i]$ for $k \in P'$, the value of W after the $(i+1)$th execution of line (5) is hence $W_{i+1} = W_i - \sum_{k=1}^{p} w_{i,k}$, as claimed in the lemma. □

Lemma 3.17. *For all $i \in [1\,..\,r]$, $W_i = p \cdot w_i + p \cdot Kd_i$, and it holds that $w_i \geq 5/K \cdot W_i/p$ and $Kd_i \leq (1 - 5/K) \cdot W_i/p$.*

Proof. This follows by lines (9) and (10), just as for Lemma 3.8. □

Lemma 3.18. *For all $i \in [1\,..\,r]$, $W_{i+1} \geq p \cdot Kd_i$.*

Proof. Obviously, at most one chunk per processor belongs to round i, and its size is no larger than w_i. By the previous two lemmas, hence, $W_{i+1} \geq W_i - p \cdot w_i = p \cdot Kd_i$. □

Lemma 3.19. *For all $i \in [1\,..\,r-1]$, $t_i^{\text{upp}} < t_{i+1}^{\text{end}}$.*

Proof. Since round $i+1$ is never started before t_i^{end}, and, by the previous two lemmas, $w_{i+1} \geq 5/K \cdot W_{i+1}/p \geq 5d_i$,

$$t_{i+1}^{\text{end}} = t_{i+1} - d_{i+1} \geq t_i^{\text{end}} + h + w_{i+1} - d_{i+1} > t_i^{\text{end}} + h + 1.5d_i = t_i^{\text{upp}}. \qquad \square$$

Lemma 3.20. *For all $i \in [1\,..\,r]$, a chunk $\mathcal{C}$ belonging to round i and scheduled at time T has size $\min\{w_i, \lfloor t_i - \min\{T, t_{i-1}^{\text{upp}}\rfloor\}\}$, and*

$$t_i^{\text{low}} - \text{early}_\alpha(\mathcal{C}) \leq \text{finish}(\mathcal{C}) - \max\{0, T - t_{i-1}^{\text{upp}}\} \leq t_i^{\text{upp}} + \text{late}_\beta(\mathcal{C}).$$

Proof. Let T be the scheduling time of chunk $\mathcal{C}$, and denote its size by w. We first observe that, according to line (8) and because $t_0^{\text{upp}} = 0$, $s[i] = \min\{w_i, \lfloor t_i - t_{i-1}^{\text{upp}}\rfloor\}$, for $i = 1, \ldots, r$. Since the previous lemma has proven that t_{i-1}^{upp} comes before the end of round i, we know that $w = \min\{w_i, \lfloor t_i - T\rfloor\}$, for $T \leq t_{i-1}^{\text{upp}}$, and $w = \min\{w_i, \lfloor t_i - t_{i-1}^{\text{upp}}\rfloor\}$ otherwise, according to lines (13) and (14). Hence, $w = \min\{w_i, \lfloor t_i - \tilde{T}\rfloor\}$, for $\tilde{T} = \min\{T, t_{i-1}^{\text{upp}}\}$, that is, the size of $\mathcal{C}$ is exactly as if it were assigned by the original BAL at time $\tilde{T}$. Correspondingly, we can show exactly as in the proof of Lemma 3.7 that

$$t_i^{\text{low}} - \text{early}_\alpha(\mathcal{C}) \leq \tilde{T} + h + \text{proc-time}(\mathcal{C}) \leq t_i^{\text{upp}} + \text{late}_\beta(\mathcal{C}),$$

and the lemma follows owing to $\tilde{T} = T - \max\{0, T - t_{i-1}^{\text{upp}}\}$ and $T + h + \text{proc-time}(\mathcal{C}) = \text{finish}(\mathcal{C})$. □

Lemma 3.21. *For all $i \in [1\,..\,r]$, $W_{i+1} \leq p \cdot Kd_i + p \cdot 2.5d_{i-1}$.*

Proof. By the previous lemma, a chunk belonging to round i has size at least $\min\{w_i, \lfloor t_i - t_{i-1}^{\text{upp}}\rfloor\}$, which by $t_i \geq t_{i-1}^{\text{end}} + h + w_i = t_{i-1}^{\text{upp}} - 1.5d_{i-1} + w_i$, is at least $\lfloor w_i - 1.5d_{i-1}\rfloor \geq w_i - 2.5d_{i-1}$. By Lemmas 3.16 and 3.17 therefore, $W_{i+1} \leq W_i - p \cdot w_i + p \cdot 2.5d_{i-1} = p \cdot Kd_i + p \cdot 2.5d_{i-1}$. □

Lemma 3.22. *For $i \in [1\,..\,r-2]$, $W_{i+3}/p \le \tilde{\gamma}(W_i/p)$.*

Proof. By two applications of the previous lemma and using Lemma 3.17, just as done in the proof of Lemma 3.12. □

Lemma 3.23. *For $i \in [1\,..\,r]$, $W_{i+1}/p \le \tilde{\gamma}^{(\lfloor i/3 \rfloor)}(n/p)$.*

Proof. This follows easily by iterative application of the previous lemma, just as for the proof of Lemma 3.13. □

3.4.2. *The Scheduling Overhead.* With the help of Lemma 3.2, we obtain from Lemma 3.23 that

$$\lfloor r/3 \rfloor \le \tilde{\gamma}^*(n/p) - \tilde{\gamma}^*(n'/p),$$

where $n = W_1$ and $n' = W_{r+1}$. Therefore, because $\lfloor r/3 \rfloor \ge (r-2)/3$ and $\tilde{\gamma}^*(n'/p) \ge 1$,

$$r \le 3 \cdot \tilde{\gamma}^*(n/p) - 3 \cdot \tilde{\gamma}^*(n'/p) + 2 \le 3 \cdot \tilde{\gamma}^*(n/p) - \tilde{\gamma}^*(n'/p).$$

Since each processor is assigned exactly one chunk in each of the r rounds, this implies

$$\text{chunks}(\mathcal{S}_{\text{I}}) = p \cdot r \le 3p \cdot \tilde{\gamma}^*(n/p) - p \cdot \tilde{\gamma}^*(n'/p).$$

Owing to the fact that the condition for termination of the first phase and the scheduling strategy for the second phase are identical for BAL and BAL′, we can prove just as in Section 3.3.4 (in fact, by the bound above on r, only the regular case can happen now) that

$$\text{chunks}(\mathcal{S}_{\text{II}}) \le p \cdot \tilde{\gamma}^*(n'/p),$$

so that altogether

$$\text{chunks}(\mathcal{S}) = \text{chunks}(\mathcal{S}_{\text{I}}) + \text{chunks}(\mathcal{S}_{\text{II}}) \le 3p \cdot \tilde{\gamma}^*(n/p).$$

The same elegant sequence of applications of Lemmas 3.5, 3.6, 3.5, and 3.1 as in Section 3.3.4 shows that $\tilde{\gamma}^* \le 2K \cdot \gamma^*_{w_{\min}}$, so that finally

$$\text{chunks}(\mathcal{S}) \le 6K \cdot p \cdot \gamma^*_{w_{\min}}(n/p).$$

3.4.3. *The Idle Time.* It follows from Theorem 3.1, that

$$\begin{aligned}\text{idle}(\mathcal{S}_{\text{II}}) \le\ & p \cdot h + p \cdot \beta(w_{\min}) + \max\{0, \text{imbalance}(\mathcal{S}_{\text{I}}) - n'/A - p \cdot h\} \\ & + \text{sum-early}_{\alpha}(\mathcal{S}_{\text{II}}) + (p-1) \cdot \text{sum-late}_{\beta}(\mathcal{S}_{\text{II}}).\end{aligned}$$

Further, the idle time of $\mathcal{S}$ is the idle time of $\mathcal{S}_{\text{II}}$ plus the time wait($\mathcal{S}_{\text{I}}$) that processors spend waiting between two chunks in the first phase; note that such waiting is a particularity of BAL′ and did never occur with BAL. In order to bound idle($\mathcal{S}$), we therefore have to bound the imbalance and the waiting time of $\mathcal{S}_{\text{I}}$ separately.

To this end, we first verify, using simple induction, that for a fixed processor with chunks $\mathcal{C}_1, \ldots, \mathcal{C}_r$ assigned to it in the first phase,

$$\text{finish}(\mathcal{C}_i) \le t_i^{\text{upp}} + \text{sum-late}_\beta(\{\mathcal{C}_1, \ldots, \mathcal{C}_i\}),$$

for $i = 1, \ldots, r$. For $i = 1$ or if $\text{finish}(\mathcal{C}_{i-1}) \le t_{i-1}^{\text{upp}}$, the claim follows directly from Lemma 3.20. Otherwise, for $i = 2, \ldots, r$ and $\text{finish}(\mathcal{C}_{i-1}) > t_{i-1}^{\text{upp}}$, $\mathcal{C}_i$ is scheduled at time $\text{finish}(\mathcal{C}_{i-1})$ and Lemma 3.20 says that $\text{finish}(\mathcal{C}_i) \le t_i^{\text{upp}} + \text{late}_\beta(\mathcal{C}_i) + \text{finish}(\mathcal{C}_{i-1}) - t_{i-1}^{\text{upp}}$, which by the induction hypothesis is at most $t_i^{\text{upp}} + \text{sum-late}_\beta(\{\mathcal{C}_1, \ldots, \mathcal{C}_i\})$.

It is now easy to bound the idle time of $\mathcal{S}_\text{I}$. We write $\text{idle}(\mathcal{S}_\text{I}) = I' + I'' + I'''$, where I', I'', and I''' denote the amount of idle time spent before t_r^{low}, between t_r^{low} and t_r^{upp}, and after t_r^{upp}, respectively. By Lemma 3.20, each unit of I' corresponds to one unit of earliness of a chunk of $\mathcal{S}_\text{I}$, so that $I' \le \text{sum-early}_\alpha(\mathcal{S}_\text{I})$. Further, it is obvious that $I'' \le p \cdot (t_r^{\text{upp}} - t_r^{\text{low}}) = p \cdot d_r + p \cdot h$, which by Lemma 3.18 is at most $n'/A + p \cdot h$. Concerning I''' we make use of the property established in the previous paragraph implying a bound of $t_r^{\text{upp}} + \text{sum-late}_\beta(\mathcal{S}_\text{I})$ on the makespan of $\mathcal{S}_\text{I}$, so that $I''' \le (p-1) \cdot \text{sum-late}_\beta(\mathcal{S}_\text{I})$. Altogether, we have thus proven that

$$\text{idle}(\mathcal{S}_\text{I}) \le n'/A + p \cdot h + \text{sum-early}_\alpha(\mathcal{S}_\text{I}) + (p-1) \cdot \text{sum-late}_\beta(\mathcal{S}_\text{I}).$$

Since waiting can only occur before t_r^{low}, we also have

$$\text{wait}(\mathcal{S}_\text{I}) \le I' \le \text{sum-early}_\alpha(\mathcal{S}_\text{I}),$$

and hence, since $\text{imbalance}(\mathcal{S}_\text{I}) = \text{idle}(\mathcal{S}_\text{I}) - \text{wait}(\mathcal{S}_\text{I})$,

$$\begin{aligned}&\max\{0, \text{imbalance}(\mathcal{S}_\text{I}) - n'/A - p \cdot h\}\\ &\quad \le \text{sum-early}_\alpha(\mathcal{S}_\text{I}) + (p-1) \cdot \text{sum-late}_\beta(\mathcal{S}_\text{I}) - \text{wait}(\mathcal{S}_\text{I}).\end{aligned}$$

Plugged into the inequality established at the beginning of the section, this yields

$$\begin{aligned}\text{idle}(\mathcal{S}_\text{II}) \le{}& p \cdot h + p \cdot \beta(w_{\min}) + \text{sum-early}_\alpha(\mathcal{S})\\ &+ (p-1) \cdot \text{sum-late}_\beta(\mathcal{S}) - \text{wait}(\mathcal{S}_\text{I}),\end{aligned}$$

so that finally

$$\begin{aligned}\text{idle}(\mathcal{S}) = \text{wait}(\mathcal{S}_\text{I}) + \text{idle}(\mathcal{S}_\text{II}) \le{}& p \cdot h\\ &+ p \cdot \beta(w_{\min}) + \text{sum-early}_\alpha(\mathcal{S}) + (p-1) \cdot \text{sum-late}_\beta(\mathcal{S}).\end{aligned}$$

3.4.4. *The Wasted Time.* The last two sections have shown that

$$\begin{aligned}&\text{chunks}(\mathcal{S}) \le 6K \cdot p \cdot \gamma^*_{w_{\min}}(n/p),\\ &\text{idle}(\mathcal{S}) \le p \cdot h + p \cdot \beta(w_{\min}) + \text{sum-early}_\alpha(\mathcal{S}) + (p-1) \cdot \text{sum-late}_\beta(\mathcal{S}),\end{aligned}$$

which, for $\varepsilon = \text{av-dev}_{\alpha,\beta}(\mathcal{S})$, immediately implies that

$$\text{waste}(\mathcal{S}) = \mathcal{O}\left((h + \varepsilon) \cdot \gamma^*_{w_{\min}}(n/p) + \beta(w_{\min})\right).$$

We have finally proven Theorem 3.3 and hence also our Main Theorem. □

4. Specific Upper Bounds

In this section we apply our Main Theorem, proven over the course of the last section, to the particular independent-tasks, bounded-tasks, and coupled-tasks settings, which we already mentioned in the Introduction, and which are defined properly later in this section. These applications turn out to be challenging tasks on their own; since the proof of the generic result was already quite involved, this gives an indication of the complexity of the scheduling problem we study here. Two tasks need to be tackled for obtaining bounds for a specific setting. First, the setting must be related to an appropriate pair of variance estimator and deviation. This will be straightforward for the bounded-tasks setting, while for the stochastic settings, this involves the estimation of tails of probability distributions. Second, a closed formula for the * of the progress rate of that variance estimator must be determined. As a solution to this interesting stand-alone mathematical problem, we propose what we call the *master theorem for the* * *operator*.

In Section 4.1 we first state and prove this master theorem, and use it to instantiate our Main Theorem for a representative selection of variance estimators. This will in fact provide valuable intuition on the relation between processing time irregularity and scheduling performance expressed by our Main Theorem. Sections 4.2, 4.3, and 4.4 are dedicated to the bounded-tasks, independent-tasks, and coupled-tasks setting, respectively.

4.1. *A Master Theorem for the Star Operator*

For sufficiently well-behaved functions $\gamma\colon \mathbb{R} \to \mathbb{R}$, the following theorem provides general-purpose approximations for the values of γ^* from above as well as from below. The addendum says that unless γ grows very slowly, namely, with slope tending to zero, the stated bounds are tight up to a constant factor. For convenience, we write $\gamma^*(x, y)$ for $\min\{i\colon \gamma^{(i)}(x) \le y\}$ in the following; in particular, then $\gamma^*(x) = \gamma^*(x, 0)$.

Theorem 4.1. *For bijective increasing* $\gamma\colon \mathbb{R} \to \mathbb{R}$ *such that* $\mathrm{id} - \gamma$ *is positive and increasing on* $\mathbb{R}$, *it holds that for all* $x, y \ge 0$,

$$\left\lceil \int_y^x \frac{dz}{\gamma^{-1}(z) - z} \right\rceil \le \gamma^*(x, y) \le \left\lceil \int_y^x \frac{dz}{z - \gamma(z)} \right\rceil.$$

If, additionally, for all z, z' *with* $y \le z < z' \le \gamma^{-1}(x)$ *the difference quotient* $(\gamma(z') - \gamma(z))/(z' - z)$ *is bounded from below by some positive constant* Q, *then*

$$\int_y^x \frac{dz}{z - \gamma(z)} \Big/ \int_y^x \frac{dz}{\gamma^{-1}(z) - z} \le \frac{1}{Q}.$$

In particular, this property holds if the (*piecewise*) *derivative of* γ *on* $[y, \gamma^{-1}(x)]$ *exists and is at least* Q.

Proof. First check that since $\mathrm{id} - \gamma$ is positive and increasing, the same applies to $\gamma^{-1} - \mathrm{id} = (\mathrm{id} - \gamma) \circ \gamma^{-1}$, simply because γ is an increasing bijection. Using that we

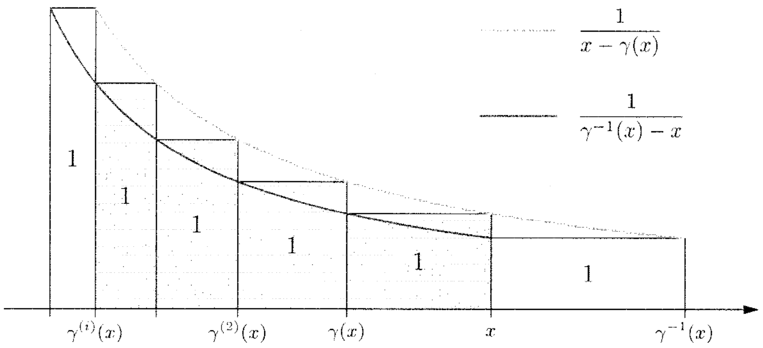

Fig. 9. How $\gamma^*(x,\cdot)$ is bounded by two integrals.

easily verify that for arbitrary x,

$$\int_{\gamma(x)}^{x} \frac{dz}{\gamma^{-1}(z)-z} \le \int_{\gamma(x)}^{x} \frac{dz}{\gamma^{-1}(\gamma(x))-\gamma(x)} = 1$$
$$= \int_{\gamma(x)}^{x} \frac{dz}{x-\gamma(x)} \le \int_{\gamma(x)}^{x} \frac{dz}{z-\gamma(z)},$$

and by analogous arguments on the intervals $[\gamma(x)^{(2)},\gamma(x)],\ldots,[\gamma^{(i)}(x),\gamma^{(i-1)}(x)]$, we obtain that for all $i\in\mathbb{N}$,

$$\int_{\gamma^{(i)}(x)}^{x} \frac{dz}{\gamma^{-1}(z)-z} \le i \le \int_{\gamma^{(i)}(x)}^{x} \frac{dz}{z-\gamma(z)}.$$

For intuition behind this approximation, see Figure 9, where i is just the area of the gray rectangles, and the integrands on the left- and right-hand side are shown in dark and light gray, respectively.

Now for arbitrary $x>y$, with $i=\gamma^*(x,y)\ge 1$, we have $\gamma^{(i)}(x)\le y$ and $\gamma^{(i-1)}(x)>y$. Therefore,

$$\int_{y}^{x} \frac{dz}{\gamma^{-1}(z)-z} \le \int_{\gamma^{(i)}(x)}^{x} \frac{dz}{\gamma^{-1}(z)-z} \le i$$

and

$$\int_{y}^{x} \frac{dz}{z-\gamma(z)} > \int_{\gamma^{(i-1)}(x)}^{x} \frac{dz}{z-\gamma(z)} \ge i-1,$$

which proves the first part of the theorem. For the second part, let z lie between y and x, $z'=\gamma^{-1}(z)$, and check that by the additional property on γ,

$$\frac{z-\gamma(z)}{\gamma^{-1}(z)-z} = \frac{\gamma(z')-\gamma(z)}{z'-z} \ge Q. \qquad \square$$

Table 3. The bound from the Main Theorem for four types of variance estimators.

$[\alpha(w), \beta(w)]$	$H \cdot \gamma^*_{AH}(N)$
$[\max\{w/A, w - Cw^{1/\kappa}\}, w + Cw^{1/\kappa}]$	$\Theta(H \cdot \log\log N + C \cdot (AH)^{1/\kappa})$
$[w/A, B \cdot w]$	$\Theta(H \cdot AB \cdot \log N)$
$[w/A, B \cdot w \log^\kappa(Cw)]$	$\Theta(H \cdot AB \cdot \log N \cdot \log^\kappa(C\,N))$
$[w/A, B \cdot w^\kappa]$	$\Theta(H \cdot (AB)^{1/\kappa} \cdot N^{1-1/\kappa})$

In the following, we apply the above theorem to instantiate the generic bound from the previous section for a variety of variance estimators. Our results are summarized in Table 3, where κ is considered a fixed constant, while for the parameters A, B, and C, all dependencies of the corresponding bound are made explicit. For the sake of clarity, we have written H for $h + \varepsilon$ and N for n/p. Strictly speaking, while the entries in the right column are always upper bounds, they are upper *and* lower bounds only for sufficiently large N; this will be made explicit in Lemmas 4.1–4.4 below.

For better readability, in the following proofs we always write $E_1 \doteqdot E_2$ instead of $E_1 = \Theta(E_2)$, for arbitrary real expressions E_1 and E_2.

4.1.1. *Superlinear Width.* We consider two types of variance estimators of superlinear width: those, for which $\big(\beta(w) - \alpha(w)\big)/w$ is polynomial, and those for which it is polylogarithmic in w. We could also consider even wider estimators, but not very meaningfully so.

Lemma 4.1. *For arbitrary fixed $A, B \geq 1$ and $\kappa > 1$, consider the variance estimator*

$$[\alpha, \beta]\colon\ w \mapsto [w/A, B \cdot w^\kappa],$$

let $\delta = \alpha^{-1} \circ (\beta - \alpha)$, and let $\gamma = \mathrm{id} - \max\{M, (\mathrm{id} + \delta)^{-1}\}$, for arbitrary fixed $M \geq 1$. Then, for all $N \geq AB \cdot M^\kappa$,

$$\gamma^*(N) = \Theta\left((AB)^{1/\kappa} \cdot N^{1-1/\kappa}\right).$$

Proof. We first give the proof for $A = 1$ and then extend it to the general case $A \geq 1$ by a simple time-scaling argument. For $A = 1$, we have $\alpha = \mathrm{id}$, which implies that $\mathrm{id} + \delta = \beta$ and hence $\gamma = \mathrm{id} - \max\{M, \beta^{-1}\}$. Here, as well as in the following three proofs, $\max\{M, \beta^{-1}\}$ and γ will be considered as functions on $\mathbb{R}$ in the obvious way by taking $\max\{M, \beta^{-1}\}(x) = M$, for all $x \leq \beta(M)$. To be able to apply Theorem 4.1, we first check whether the preconditions on γ are satisfied. Since β is a bijection $\mathbb{R}^+ \to \mathbb{R}^+$ and $\beta(w) \geq w$ for $w \geq M \geq 1$, it follows immediately that $\gamma = \mathrm{id} - \max\{M, \beta^{-1}\}$ is bijective on $\mathbb{R}$, as well as that $\mathrm{id} - \gamma = \max\{M, \beta^{-1}\}$ is positive and increasing on $\mathbb{R}$. Further, since for $w \geq M$, $\beta'(w) = B \cdot \kappa \cdot w^{\kappa-1} \geq B \cdot \kappa$, the derivative of β^{-1} on $[\beta(M), \infty)$ is bounded from above by $1/(B\kappa) < 1$. This is easily seen to imply that all difference quotients of γ are bounded from below by $1-1/(B\kappa) > 0$, and, in particular, that γ is increasing on $\mathbb{R}$. Theorem 4.1 therefore yields that

$$\gamma^*(N) \doteqdot \int_0^N \frac{dz}{\max\{M, \beta^{-1}(z)\}} = \int_0^{\beta(M)} \frac{dz}{M} + \int_{\beta(M)}^N \frac{dz}{\beta^{-1}(z)}.$$

Since $\beta^{-1}(z) = (z/B)^{1/\kappa}$, we obtain

$$\begin{aligned}\gamma^*(N) &\doteqdot \frac{\beta(M)}{M} + B^{1/\kappa} \cdot \int_{BM^\kappa}^N z^{-1/\kappa}\, dz \\ &= B \cdot M^{\kappa-1} + B^{1/\kappa} \cdot \frac{N^{1-1/\kappa} - (B \cdot M^\kappa)^{1-1/\kappa}}{1-1/\kappa} \\ &\doteqdot B^{1/\kappa} \cdot N^{1-1/\kappa},\end{aligned}$$

where the last approximation uses that $N \geq B \cdot M^\kappa$. This proves the lemma for $A = 1$.

To deal with the general case $[\alpha(w), \beta(w)] = [w/A, B \cdot w^\kappa]$, just observe that $\alpha^{-1} \circ (\beta - \alpha)$, and hence also γ, is independent of the choice of our time unit, that is, it is invariant under the simultaneous multiplication of α and β by an arbitrary fixed constant. Instead of $w \mapsto [w/A, B \cdot w^\kappa]$, we may therefore just as well consider the variance estimator $w \mapsto [w, AB \cdot w^\kappa]$, for which the above analysis shows that if $N \geq AB \cdot M^\kappa$, then

$$\gamma^*(N) \doteqdot (AB)^{1/\kappa} \cdot N^{1-1/\kappa},$$

as desired. □

Lemma 4.2. *For arbitrary fixed $A, B, C, \kappa \geq 1$, consider the variance estimator*

$$[\alpha, \beta]\colon\ w \mapsto [w/A,\, B \cdot w \cdot \ln^\kappa(Cw)],$$

let $\delta = \alpha^{-1} \circ (\beta - \alpha)$, and let $\gamma = \mathrm{id} - \max\{M, (\mathrm{id} + \delta)^{-1}\}$, for arbitrary fixed $M \geq 3$. Then for all $N \geq (AB\,M \ln^\kappa(C\,M))^2$,

$$\gamma^*(N) = \Theta\left(AB \cdot \ln N \cdot \ln^\kappa(CN)\right).$$

Proof. We proceed just like in the proof of Lemma 4.1, first assuming that $A = 1$, for which $\gamma = \mathrm{id} - \max\{M, \beta^{-1}\}$. Since for $w \geq M \geq 3$,

$$\beta'(w) = B \cdot \ln^{\kappa-1}(Cw) \cdot (\kappa + \ln(Cw)) \geq 2,$$

we can argue just as before that all difference quotients of γ are bounded from below by

$\frac{1}{2}$ so that, by Theorem 4.1,

$$\gamma^*(N) \doteqdot \frac{\beta(M)}{M} + \int_{\beta(M)}^{N} \frac{dz}{\beta^{-1}(z)}.$$

Unfortunately, with $\beta(w) = B \cdot w \cdot \ln^\kappa(Cw)$, no closed form for the inverse β^{-1} exists. However, taking instead $\tilde{\beta}(z) = z/(B \cdot \ln^\kappa(Cz))$, we have that for $w \geq M \geq 3$,

$$\tilde{\beta}(\beta(w)) = \frac{B \cdot w \cdot \ln^\kappa(Cw)}{B \cdot (\ln(C\beta(w)))^\kappa} = \frac{w \cdot \ln^\kappa(Cw)}{(\ln(Cw) + \ln B + \kappa \ln\ln(Cw))^\kappa},$$

which is clearly at most w and at least $w/(1 + \ln B + \kappa)^\kappa$. Therefore $\tilde{\beta}(z)$ is within a constant factor of $\beta^{-1}(z)$, for all $z \geq \beta(M)$, and thus

$$\begin{aligned}
\int_{\beta(M)}^{N} \frac{dz}{\beta^{-1}(z)} &\doteqdot B \cdot \int_{\beta(M)}^{N} \frac{\ln^\kappa(Cz)}{z}\, dz \\
&= B \cdot \int_{\ln \beta(M)}^{\ln N} (\bar{z} + \ln C)^\kappa \, d\bar{z} \\
&= B \cdot \frac{\ln^{\kappa+1}(CN) - \ln^{\kappa+1}(C\beta(M))}{\kappa + 1} \\
&= B \cdot \ln\left(\frac{N}{\beta(M)}\right) \cdot \sum_{j=0}^{\kappa} \frac{\ln^j(CN) \cdot \ln^{\kappa-j}(C\beta(M))}{\kappa + 1},
\end{aligned}$$

where the sum can easily be seen to lie between $\ln^\kappa(CN)$ and $(\kappa + 1) \cdot \ln^\kappa(CN)$. Since $N \geq \beta^2(M)$, we have thus shown that for the case $A = 1$,

$$\gamma^*(N) \doteqdot B \cdot \ln N \cdot \ln^\kappa(CN).$$

For arbitrary $A \geq 1$, the same formula holds for $N \geq (A\beta)^2(M)$ and when B is replaced by AB, which is shown via the invariance of $[\alpha, \beta]$ under an arbitrary time scaling exactly as done for the previous proof. □

4.1.2. *Linear Width*

Lemma 4.3. *For arbitrary fixed $A \geq 1$ and $B > 1$, consider the variance estimator*

$$[\alpha, \beta]\colon\ w \mapsto [w/A,\, B \cdot w].$$

Let $\delta = \alpha^{-1} \circ (\beta - \alpha)$, and let $\gamma = \mathrm{id} - \max\{M, (\mathrm{id} + \delta)^{-1}\}$, for arbitrary fixed $M \geq 2$. Then for all $N \geq (AB \cdot M)^2$,

$$\gamma^*(N) = \Theta(AB \cdot \ln N).$$

Proof. Since $\gamma = \mathrm{id} - \max\{M, \mathrm{id}/(AB)\}$ has slope at least $1 - 1/(AB) > 0$ everywhere, Theorem 4.1 yields that

$$\begin{aligned}\gamma^*(N) &\doteqdot \int_0^N \frac{dz}{\max\{M, z/(AB)\}} \\ &= \int_0^{AB\cdot M} \frac{dz}{M} + AB \cdot \int_{AB\cdot M}^N \frac{dz}{z} \\ &= AB + AB \cdot \ln \frac{N}{AB \cdot M},\end{aligned}$$

which for $N \geq (AB \cdot M)^2$ is within a constant factor of $AB \cdot \ln N$. $\square$

This result implies the following interesting property of linear-width variance estimators, which, as we will see in a minute, give rise to scheduling schemes that are particularly easy to implement. Consider the polylogarithmically superlinear variance estimator

$$[\alpha, \beta]\colon\ w \mapsto \bigl[w/A,\, B \cdot w \cdot \ln^\kappa(Cw)\bigr],$$

for which the Main Theorem together with Lemma 4.2 proves the following bound on the wasted time:

$$\Theta(H \cdot AB \cdot \ln N \cdot \ln^\kappa(CN)).$$

Now for n tasks and p processors, chunk sizes are naturally bounded by $N = n/p$, and for all $w \leq N$,

$$[w/A,\, B \cdot w \ln^\kappa(Cw)] \subseteq [w/A,\, B \cdot \ln^\kappa(CN) \cdot w].$$

However, for the linear-width variance estimator corresponding to the ranges on the right-hand side, the Main Theorem in combination with Lemma 4.3 implies a bound on the wasted time of

$$\Theta(H \cdot AB \cdot \ln^\kappa(CN) \cdot \ln N),$$

which is identical to the bound obtained for the variance estimator of polylogarithmically superlinear width. This provides evidence that the class of scheduling schemes associated with variance estimators of linear width are optimal for a wide variety of settings. On the other hand, the optimal scheme pertaining to such a variance estimator $[\mathrm{id}/A,\, B \cdot \mathrm{id}]$ is of a particularly simple form, namely, $\mathrm{FP}(x \mapsto \lfloor x/C + w_{\min} \rfloor)$, where $C = A \cdot B$. It is easy to see that (ignoring the rounding issue) the sizes of the chunks assigned according to this strategy form a geometric sequence; this follows by $(W - (W/(Cp) + w_{\min}))/(Cp) + w_{\min} = (W/(Cp) + w_{\min}) \cdot (1 - 1/(Cp))$. We leave it to the reader to verify that the variance estimators of polynomially superlinear width, which were considered in Lemma 4.1, may not be replaced by variance estimators of linear width without loss.

4.1.3. *Sublinear Width*

Lemma 4.4. *For arbitrary fixed $A \geq 1$, $C > 1$, and $\kappa > 1$, consider the variance estimator*

$$[\alpha, \beta]\colon\ w \mapsto [\max\{w/A, w - Cw^{1/\kappa}\}, w + Cw^{1/\kappa}],$$

let $\delta = \alpha^{-1} \circ (\beta - \alpha)$, and let $\gamma = \mathrm{id} - \max\{M, (\mathrm{id} + \delta)^{-1}\}$, for arbitrary fixed $M > 0$. Then for all $N \geq \max\{M, (2AC)^{\kappa/(\kappa-1)}\}$,

$$\gamma^*(N) = \Theta\left(\ln(\ln N/\ln M) + C \cdot A \cdot M^{1/\kappa}/M\right).$$

Proof. As before, our goal will be to bound $\gamma^*(N)$ with the help of Theorem 4.1, which for sublinear-width variance estimators, however, turns out to be more complicated than for those of at least linear width. In particular, we cannot use the time-scaling argument here since for $A > 1$, $A\beta$ is no longer of the same form as β, as it has been in the cases considered before. This proof is therefore going to be more involved than its predecessors.

Since $\alpha(w) = \max\{w/A, w - Cw^{1/\kappa}\}$, δ may have a sharp (that is, not differentiable) bend, which turns out to be somewhat unconvenient to deal with. However, it is easy to see that $\delta(w)$ is always between $C \cdot w^{1/\kappa}$ and $2AC \cdot w^{1/\kappa}$. In view of Lemmas 3.1 and 3.5, we may hence assume without loss of generality that

$$\delta(w) = 2AC \cdot w^{1/\kappa}.$$

Our next step will be to bound $\gamma^*(\tilde{N})$, where $\tilde{N} = \max\{\delta(M), (2AC)^{\kappa/(\kappa-1)}\}$; note that the second term is just the unique positive fixpoint of δ, and that $\tilde{N} \leq N$. To this end, first verify by means of the identity $\mathrm{id} - (\mathrm{id} + \delta)^{-1} = (\mathrm{id} + \delta^{-1})^{-1}$ that the inverse of $\gamma = \mathrm{id} - \max\{M, (\mathrm{id} + \delta)^{-1}\}$ is just

$$\gamma^{-1} = \mathrm{id} + \max\{M, \delta^{-1}\}.$$

Now on $(-\infty, \delta(M)]$, γ^{-1} describes a straight line with slope 1, while the derivative of $\delta^{-1}\colon z \mapsto z^\kappa/(2AC)^\kappa$ at an arbitrary $z \in (0, (2AC)^{\kappa/(\kappa-1)}]$ is

$$\kappa \cdot z^{\kappa-1}/(2AC)^\kappa \leq \kappa.$$

We may therefore conclude that γ^{-1} has slope bounded by $\kappa + 1$ everywhere on $(-\infty, \tilde{N}]$, and hence that γ has slope at least $1/(\kappa+1)$ everywhere on $(-\infty, \gamma^{-1}(\tilde{N})]$, so that, by Theorem 4.1,

$$\gamma^*(\tilde{N}) \doteqdot \int_0^{\tilde{N}} \frac{dz}{\max\{M, \delta^{-1}(z)\}}.$$

Since $\tilde{N} \geq \delta(M)$, it holds that

$$\begin{aligned}\int_0^{\tilde{N}} \frac{dz}{\max\{M, \delta^{-1}(z)\}} &= \int_0^{\delta(M)} \frac{dz}{M} + \int_{\delta(M)}^{\tilde{N}} \frac{dz}{z^\kappa/(2AC)^\kappa} \\ &= \frac{\delta(M)}{M} + (2AC)^\kappa \frac{\tilde{N}^{1-\kappa} - \delta(M)^{1-\kappa}}{1-\kappa} \\ &\doteqdot \frac{\delta(M)}{M},\end{aligned}$$

and thus

$$\gamma^*(\tilde{N}) \doteqdot \delta(M)/M = 2C \cdot A \cdot M^{1/\kappa}/M.$$

To finish the proof, it remains to bound $\gamma^*(N, \tilde{N}) = \min\{i\colon \gamma^{(i)}(N) \leq \tilde{N}\}$, that is, the number of iterations of γ required to get from N to $\tilde{N}$. In fact, we actually bound $\gamma^*(N, \tilde{N} + M)$, which differs by at most one from $\gamma^*(N, \tilde{N})$. Unfortunately, the derivative of δ^{-1} grows beyond all bounds so that $\lim_{x\to\infty} \gamma'(x) = 0$, which invalidates a further direct application of Theorem 4.1. What comes to our rescue is that the evaluation of γ^* can be shown to be equivalent to the evaluation of $\tilde{\gamma}^*$, where $\tilde{\gamma} = T \circ \gamma \circ T^{-1}$, for an arbitrary bijective transform T. In order to prove this, use simple induction to check that

$$\tilde{\gamma}^{(i)}(T(x)) = \tilde{\gamma}^{(i-1)}(\tilde{\gamma}(T(x))) = \tilde{\gamma}^{(i-1)}(T(\gamma(x))) = \cdots = T(\gamma^{(i)}(x)),$$

for all $i \in \mathbb{N}$, which immediately implies that

$$\gamma^*(x, y) = \tilde{\gamma}^*(T(x), T(y))$$

for all x, y in the domain of T.

For our purposes, consider the transform $T\colon z \mapsto \ln(z/\hat{N})$ with inverse $T^{-1}\colon z \mapsto \hat{N} \cdot e^z$, where $\hat{N} = (2AC)^{\kappa/(\kappa-1)}$ denotes the fixpoint of δ; in particular, $\hat{N} \leq \tilde{N}$. Then, since for all $z \geq \delta(M)$, $\gamma^{-1}(z) = z + \delta^{-1}(z) = z + z^\kappa/(2AC)^\kappa$, it holds for all $z \geq T(\delta(M))$ that

$$\begin{aligned}\tilde{\gamma}^{-1}(z) = (T \circ \gamma^{-1} \circ T^{-1})(z) &= (T \circ \gamma^{-1})(\hat{N} \cdot e^z) \\ &= T(\hat{N}e^z + (\hat{N}e^z)^\kappa/(2AC)^\kappa) \\ &= T(\hat{N}e^z \cdot (1 + e^{(\kappa-1)z})) \\ &= z + \ln(1 + e^{(\kappa-1)z}).\end{aligned}$$

Using that, we easily check that for all $z \geq T(\delta(M))$,

$$\left(\tilde{\gamma}^{-1}\right)'(z) = 1 + (\kappa - 1)\frac{e^{(\kappa-1)z}}{1 + e^{(\kappa-1)z}} \leq \kappa,$$

which implies that $\tilde{\gamma}'(z) \geq 1/\kappa$, for all $z \geq \tilde{\gamma}^{-1}(T(\delta(M))) = T(\gamma^{-1}(\delta(M))) = T(M + \delta(M))$. Having verified this, we may now apply Theorem 4.1 in order to obtain the approximation

$$\tilde{\gamma}^*(T(N), T(\tilde{N}+M)) \doteqdot \int_{T(\tilde{N}+M)}^{T(N)} \frac{dz}{\tilde{\gamma}^{-1}(z) - z} = \int_{T(\tilde{N}+M)}^{T(N)} \frac{dz}{\ln(1 + e^{(\kappa-1)z})}.$$

Since the integral cannot be solved in a closed form, we resort to the approximation $(t+1)/2 \leq \ln(1+e^t) \leq t+1$ for $t \geq 0$, from which we obtain that

$$\int_{T(\tilde{N}+M)}^{T(N)} \frac{dz}{\ln(1+e^{(\kappa-1)z})} \doteqdot \int_{T(\tilde{N}+M)}^{T(N)} \frac{dz}{1+(\kappa-1)z}$$
$$= \frac{1}{\kappa-1} \cdot \ln \frac{1+(\kappa-1)\cdot T(N)}{1+(\kappa-1)\cdot T(\tilde{N}+M)}.$$

Since $T(N) = \ln(N/\hat{N})$ and $T(\tilde{N}+M) = \ln((\tilde{N}+M)/\hat{N})$, and by the bound on N assumed in the lemma, the last term can be shown to be in the order of $\ln \log_M N$, and we finally get

$$\gamma^*(N, \tilde{N}) \doteqdot \gamma^*(N, \tilde{N}+M) = \tilde{\gamma}^*(T(N), T(\tilde{N}+M)) \doteqdot \ln(\ln N/\ln M).$$

Having bounded $\gamma^*(\tilde{N})$ and $\gamma^*(N, \tilde{N})$ separately, we now easily obtain the desired result

$$\gamma^*(N) \doteqdot \gamma^*(N, \tilde{N}) + \gamma^*(\tilde{N}) \doteqdot \ln(\ln N/\ln M) + C \cdot A \cdot M^{1/\kappa}/M.$$

This finishes the analysis of variance estimators with sublinear width, and we have finally proven all the bounds claimed in Table 3. □

4.2. *Bounded Tasks*

If for some $T_{\min}, T_{\max} > 0$ with $T_{\min} \leq 1 \leq T_{\max}$, the processing time of each task is guaranteed to be in the range $[T_{\min}, T_{\max}]$, we say that task processing times are *bounded by* $[T_{\min}, T_{\max}]$. Note that, as for our definition of a variance estimator, the condition $T_{\min} \leq 1 \leq T_{\max}$ is not a restriction but merely reflects a commitment to a certain time scale. As we will see next, the application of the Main Theorem to a bounded-tasks setting is almost trivial; in particular, note that no randomness is involved here.

Lemma 4.5. *Let task processing times be bounded by* $[T_{\min}, T_{\max}]$. *Then for all* $n, p \in \mathbb{N}$, *the schedule* $\mathcal{S}$ *produced by an arbitrary algorithm given* n *tasks and* p *processors satisfies*

$$\text{av-dev}_{\alpha,\beta}(\mathcal{S}) = 0,$$

where

$$[\alpha, \beta]\colon\ w \mapsto [T_{\min} \cdot w, T_{\max} \cdot w].$$

Proof. It suffices to observe that, by the assumption on the task processing times, the sum of the processing times of any w tasks will always be at least $T_{\min} \cdot w$ and at most $T_{\max} \cdot w$. $\square$

Corollary 4.2. *Let task processing times be bounded by* $[T_{\min}, T_{\max}]$*, and let the overhead be* $h \geq 1$*. Then there exists a fixed-partition algorithm that for all* $n, p \in \mathbb{N}$*, given* n *tasks and* p *processors, produces a schedule* $\mathcal{S}$ *with*

$$\text{waste}(\mathcal{S}) = \mathcal{O}(h \cdot T_{\max}/T_{\min} \cdot \log(n/p)).$$

Proof. For every $w_{\min} \in \mathbb{N}$ with $w_{\min} \geq h$, a combination of the Main Theorem, or rather of Theorem 3.1, with the previous lemma yields a fixed partition algorithm that for n tasks and p processors produces a schedule $\mathcal{S}$ with

$$\text{waste}(\mathcal{S}) = \mathcal{O}(h \cdot \gamma^*_{w_{\min}}(n/p) + \beta(w_{\min})),$$

where $\gamma_{w_{\min}}$ is the progress rate associated with

$$[\alpha, \beta]\colon\ w \mapsto [T_{\min} \cdot w, T_{\max} \cdot w]$$

and $w_{\min}$. According to our comments concerning the choice of $w_{\min}$ at the beginning of Section 3, we choose $w_{\min} = \lceil h/T_{\min} \rceil$, in which case $\beta(w_{\min}) \leq 2 \cdot h \cdot T_{\max}/T_{\min}$. Further, Lemma 4.3 tells us that $\gamma^*_{w_{\min}}(n/p) = O(T_{\max}/T_{\min} \cdot \log(n/p))$, and we finally obtain

$$\begin{aligned}\text{waste}(\mathcal{S}) &= \mathcal{O}(h \cdot T_{\max}/T_{\min} \cdot \log(n/p) + h \cdot T_{\max}/T_{\min})\\ &= \mathcal{O}(h \cdot T_{\max}/T_{\min} \cdot \log(n/p)).\end{aligned}$$

$\square$

4.3. *Independent Tasks*

If for some $\sigma > 0$, the processing times of the tasks are independent, identically distributed, nonnegative random variables with mean 1, variance σ^2, and finite third moment, we say that task processing times are *independent, with variance* σ^2. The application of our Main Theorem to this setting is analogous to that for the bounded-tasks setting, but significantly more involved. As a prerequisite, we first prove the following assertion on the sum of independent, identically distributed random variables; a related, more general result can be found in [1].

Lemma 4.6. *Let* Z *be the sum of* m *independent random variables, identically distributed with mean* $\mu \leq 0$*, finite variance* $\sigma^2 > 0$*, and finite third central moment* $\varrho^3 > 0$*. Then for some constant* $\vartheta > 0$*, with* $t = -\mu/\sigma$ *and* $\eta = \vartheta \cdot e^{t^2/2} \cdot \varrho^3/\sigma^3 \big/ m^{1/2}$*,*

$$\mathbf{E}\ \max\{0, Z\} \leq (1 + \eta) \cdot \sigma\sqrt{m} \cdot \frac{1}{\sqrt{2\pi}} \frac{1}{1 + t^2} e^{-t^2/2}.$$

If Z *is normal, the inequality even holds for* $\eta = 0$.

Proof. Let $\mu_Z = m \cdot \mu$ and $\sigma_Z^2 = m \cdot \sigma^2$ denote the mean and variance of Z, respectively. We first consider the special case where Z is a normal random variable. Then $(Z - \mu_Z)/\sigma_Z$ has standard normal distribution, so that, by the definition of the expected value,

$$\begin{aligned}
\mathbf{E}\ \max\{0, Z\} &= \sigma_Z \cdot \mathbf{E}\ \max\{0, (Z - \mu_Z)/\sigma_Z - t\} \\
&= \sigma_Z \cdot \int_{-\infty}^{\infty} \max\{0, x - t\} \cdot \frac{1}{\sqrt{2\pi}} e^{-x^2/2}\, dx \\
&= \sigma_Z \cdot \int_{t}^{\infty} (x - t) \cdot \frac{1}{\sqrt{2\pi}} e^{-x^2/2}\, dx \\
&= \sigma_Z \cdot \left(\frac{1}{\sqrt{2\pi}} e^{-t^2/2} - t \cdot (1 - \Phi(t)) \right),
\end{aligned}$$

where Φ denotes the standard normal distribution function. The lemma now follows by the approximation

$$1 - \Phi(t) \geq \frac{1}{\sqrt{2\pi}} \frac{t}{1 + t^2} e^{-t^2/2}$$

and the fact that $1 - t \cdot t/(1 + t^2) = 1/(1 + t^2)$. The proof of this approximation is analogous to that given, for example, in the textbook of Grimmett and Stirzaker [10] for a slightly weaker bound; it suffices to check that for all x,

$$\frac{d}{dx} \frac{x}{1 + x^2} e^{-x^2/2} \geq -e^{-x^2/2},$$

which, by multiplication with $(2\pi)^{-1/2}$ and integration over $[t, \infty]$, yields the desired bound. This finishes the proof for the case of normal Z.

Now assume that Z is the sum of m independent, identically distributed random variables. Our plan is to bound the difference between $\mathbf{E}\ \max\{0, Z\}$ and $\mathbf{E}\ \max\{0, \tilde{Z}\}$, where $\tilde{Z}$ is a normal variable with the same mean and variance as Z. This will reduce the proof of the lemma to what has already been shown in the first paragraph. Bounding this difference turns out to be a matter of bounding the pointwise difference between the distribution functions of Z and $\tilde{Z}$, which we establish via a strong bound on the convergence rate of the central limit theorem. First observe that, since the mean of a nonnegative random variable X can be expressed as $\int_0^{\infty} \Pr(X > x)\, dx$ (see, for example, [10]),

$$\begin{aligned}
\mathbf{E}\ \max\{0, Z\} &= \sigma_Z \cdot \int_0^{\infty} \Pr\left(\frac{Z}{\sigma_Z} > x \right) dx \\
&= \sigma_Z \cdot \int_0^{\infty} \Pr\left(\frac{Z - \mu_Z}{\sigma_Z} > x + t \right) dx \\
&= \sigma_Z \cdot \int_t^{\infty} \Pr\left(\frac{Z - \mu_Z}{\sigma_Z} > x \right) dx,
\end{aligned}$$

and

$$\mathbf{E}\ \max\{0, \tilde{Z}\} = \sigma_Z \cdot \int_t^{\infty} \Pr\left(\frac{\tilde{Z} - \mu_Z}{\sigma_Z} > x \right) dx.$$

Now by a variant of the Berry–Esseen inequality [27, Theorem 5.17], there exists a constant $\vartheta > 0$, such that for all $x > 0$,

$$\left|\Pr\left(\frac{Z-\mu_Z}{\sigma_Z} > x\right) - \Pr\left(\frac{\tilde{Z}-\mu_Z}{\sigma_Z} > x\right)\right| \le \frac{\vartheta}{\sqrt{2\pi}} \cdot \frac{m \cdot \varrho^3}{\left(m \cdot \sigma^2\right)^{3/2}} \cdot \frac{1}{(1+x)^3},$$

and hence, with $\eta = \vartheta \cdot e^{t^2/2} \cdot \varrho^3/\sigma^3/m^{1/2}$,

$$\left|\Pr\left(\frac{Z-\mu_Z}{\sigma_Z} > x\right) - \Pr\left(\frac{\tilde{Z}-\mu_Z}{\sigma_Z} > x\right)\right| \le \eta \cdot \frac{1}{\sqrt{2\pi}} \cdot e^{-t^2/2} \cdot \frac{1}{(1+x)^3}.$$

Since $\int_t^\infty (1+x)^{-3}\,dx \le (1+t^2)^{-1}$, this implies

$$\int_t^\infty \left|\Pr\left(\frac{Z-\mu_Z}{\sigma_Z} > x\right) - \Pr\left(\frac{\tilde{Z}-\mu_Z}{\sigma_Z} > x\right)\right| dx \le \eta \cdot \frac{1}{\sqrt{2\pi}} \frac{1}{1+t^2} e^{-t^2/2},$$

and we may conclude that

$$|\mathbf{E}\,\max\{0, Z\} - \mathbf{E}\,\max\{0, \tilde{Z}\}| \le \eta \cdot \sigma_Z \cdot \frac{1}{\sqrt{2\pi}} \frac{1}{1+t^2} e^{-t^2/2}.$$

Together with what was shown in the first paragraph for the normal case, this proves our lemma in the general case. □

We are now ready to derive a bound on the average deviation incurred by some strategy in the independent-tasks setting. This bound will contain an implicit factor of ϱ^3/σ^3, where σ^2 is the variance and ϱ^3 is the absolute third central moment of a single task's processing time. We treat this factor as a fixed constant, which is justified in view of the fact that for an arbitrary random variable X, the quotient $\mathbf{E}|X - \mathbf{E}X|^3/(\mathbf{E}|X - \mathbf{E}X|^2)^{3/2}$ is invariant under the multiplication of X with an arbitrary (nonzero) factor.

Lemma 4.7. *Let task processing times be independent, with variance σ^2. Then for all $n, p \in \mathbb{N}$, the schedule $\mathcal{S}$ produced by an arbitrary algorithm given n tasks and p processors satisfies*

$$\mathbf{E}[\text{av-dev}_{\alpha,\beta}(\mathcal{S})] = O(\sigma),$$

*where*1

$$[\alpha, \beta]\colon\ w \mapsto [w - \sigma \cdot \sqrt{\ln w} \cdot w^{1/2}, w + \sigma \cdot \sqrt{p + \ln w} \cdot w^{1/2}].$$

Proof. We first prove that the expected deviation of an arbitrary fixed chunk with respect to $[\alpha, \beta]$ is bounded by $O(\sigma)$. To this end, let $w \in \mathbb{N}$, and let $\mathcal{C}$ be a chunk of an arbitrary but fixed selection of w tasks. Then, by the definitions in Section 2.2, and with T denoting the total processing time of $\mathcal{C}$,

$$\begin{aligned}
&\mathbf{E}\,\mathrm{early}_\alpha(\mathcal{C}) = \mathbf{E}\,\max\{0, w - \sigma \cdot \sqrt{\ln w} \cdot w^{1/2} - T\},\\
&\mathbf{E}\,\mathrm{late}_\beta(\mathcal{C}) = \mathbf{E}\,\max\{0, T - w - \sigma \cdot \sqrt{p + \ln w} \cdot w^{1/2}\}.
\end{aligned}$$

Let $s = \vartheta \cdot \varrho^3/\sigma^3$, where ϱ^3 is the absolute third central moment of a single task's processing time, and ϑ is the constant according to Lemma 4.6. Since T has expected value w and variance $\sigma^2 w$, we then obtain from Lemma 4.6, applied with $t_1 = \sqrt{\ln w}$ and $\eta_1 = s \cdot e^{t_1^2/2}/\sqrt{w} = s$, that

$$\mathbf{E}\,\mathrm{early}_\alpha(\mathcal{C}) \le (1 + \eta_1) \cdot \sigma\sqrt{w} \cdot \frac{1}{\sqrt{2\pi}} \frac{1}{1 + t_1^2} e^{-t_1^2/2} \le \frac{1 + s}{\sqrt{2\pi}} \cdot \sigma.$$

Similarly, with $t_2 = \sqrt{p + \ln w}$ and $\eta_2 = s \cdot e^{t_2^2/2}/\sqrt{w} = s \cdot e^{p/2}$,

$$\mathbf{E}\,\mathrm{late}_\beta(\mathcal{C}) \le (1 + \eta_2) \cdot \sigma\sqrt{w} \cdot \frac{1}{\sqrt{2\pi}} \frac{1}{1 + t_2^2} e^{-t_2^2/2} \le \frac{1 + s}{\sqrt{2\pi}} \cdot \frac{\sigma}{p}.$$

It follows immediately that

$$\mathbf{E}\,\mathrm{dev}_{\alpha,\beta}(\mathcal{C}) = \mathbf{E}\,\mathrm{early}_\alpha(\mathcal{C}) + (p - 1) \cdot \mathbf{E}\,\mathrm{late}_\beta(\mathcal{C}) \le (1 + s) \cdot \sigma,$$

and we have shown that, for $\varepsilon = (1 + s) \cdot \sigma$, the expected deviation of an arbitrary fixed chunk with respect to $[\alpha, \beta]$ is bounded by ε.

Using this property, we now prove the lemma. Let $\mathcal{C}_1, \ldots, \mathcal{C}_l$ denote the chunks of $\mathcal{S}$, in the order they were allocated, and denote by $w_1, \ldots, w_l$ their respective sizes. Now l is a random variable but certainly $l \le n$, so that we may define n random variables $Y_1, \ldots, Y_n$ such that, for $j = 1, \ldots, n$,

$$Y_j = \begin{cases} \mathrm{dev}_{\alpha,\beta}(\mathcal{C}_j), & j \le l,\\ \varepsilon, & j > l. \end{cases}$$

Since

$$\text{av-dev}_{\alpha,\beta}(\mathcal{S}) = \frac{1}{l}\sum_{j=1}^{l} Y_j = \varepsilon + \frac{n}{l} \cdot \left(\frac{1}{n}\sum_{j=1}^{n} Y_j - \varepsilon\right),$$

proving the lemma reduces to bounding the expectation of $(1/n)\sum_{j=1}^{n} Y_j$ by ε.

Since the selection of tasks belonging to $\mathcal{C}_j$ is completely determined by the algorithm together with the processing times of the previously scheduled chunks $\mathcal{C}_1, \ldots, \mathcal{C}_{j-1}$, and since the processing times of the individual tasks are independent, the property established in the first paragraph of the proof implies that for all $j = 1, \ldots, l$,

$$\mathbf{E}[Y_j \mid Y_1, \ldots, Y_{j-1}] \le \varepsilon.$$

Since this very bound holds trivially when $j > l$, it holds in fact for all $j = 1, \ldots, n$. Using this, we can show that for all $j = 1, \ldots, n$,

$$\begin{aligned}\mathbf{E}\left[\frac{1}{j}\sum_{i=1}^{j} Y_i \,\middle|\, Y_1, \ldots, Y_{j-1}\right] &= \frac{1}{j}\sum_{i=1}^{j-1} Y_i + \frac{1}{j}\cdot \mathbf{E}[Y_j | Y_1, \ldots, Y_{j-1}] \\ &\leq \frac{1}{j}\sum_{i=1}^{j-1} Y_i + \frac{\varepsilon}{j},\end{aligned}$$

which, by taking expectation on both sides, implies that

$$\mathbf{E}\left[\frac{1}{j}\sum_{i=1}^{j} Y_i\right] \leq \frac{j-1}{j}\cdot \mathbf{E}\left[\frac{1}{j-1}\sum_{i=1}^{j-1} Y_i\right] + \frac{\varepsilon}{j}.$$

A simple induction now shows that

$$\mathbf{E}\left[\frac{1}{n}\sum_{j=1}^{n} Y_j\right] \leq \varepsilon,$$

which immediately implies the desired bound

$$\mathbf{E}\,\text{av-dev}_{\alpha,\beta}(\mathcal{S}) = \varepsilon + \frac{n}{l}\cdot\left(\mathbf{E}\left[\frac{1}{n}\sum_{j=1}^{n} Y_j\right] - \varepsilon\right) \leq \varepsilon = (1+s)\cdot\sigma = O(\sigma). \qquad \square$$

With the help of Lemma 4.7, it is now easy to translate the Main Theorem to the independent-tasks setting. We remark that Corollary 4.3 below implies a doubly logarithmic asymptotic bound as n grows large, which settles a conjecture put forward in [12]. The apparently weird $\sigma\sqrt{p}\cdot(h+\sigma^2)^{1/2+\lambda}$ term becomes meaningful in light of the fact that when the minimum chunk size is in the order of $h+\sigma^2$ (as will be), then the expected maximal processing time of p chunks of such size is tightly bounded by $O((h+\sigma^2)+\sigma\sqrt{p}\cdot(h+\sigma^2)^{1/2})$ [11], [14].

Corollary 4.3. *Let task processing times be independent, with variance σ^2, and let the overhead be $h \geq 1$. Then for arbitrary fixed $\lambda > 0$, there exists an algorithm that for all $n, p \in \mathbb{N}$, given n tasks and p processors, produces a schedule $\mathcal{S}$ with*

$$\mathbf{E}\,\text{waste}(\mathcal{S}) = \mathcal{O}((h+\sigma)\cdot\log\log(n/p) + \sigma\sqrt{p}\cdot(h+\sigma^2)^{1/2+\lambda}).$$

Proof. For every $w_{\min} \in \mathbb{N}$ with $w_{\min} \geq h$, a combination of our Main Theorem with the previous lemma yields an algorithm that for n tasks and p processors yields a schedule $\mathcal{S}$ with

$$\mathbf{E}\,\text{waste}(\mathcal{S}) = \mathcal{O}((h+\sigma)\cdot\gamma^*_{w_{\min}}(n/p) + \beta(w_{\min})),$$

where $\gamma_{w_{\min}}$ is the progress rate associated with

$$[\alpha, \beta]\colon\ w \mapsto [w - \sigma \cdot \sqrt{\ln w} \cdot w^{1/2}, w + \sigma \cdot \sqrt{p + \ln w} \cdot w^{1/2}]$$

and $w_{\min}$. Now $[\alpha, \beta]$ has sublinear width but is not quite of the type considered in the corresponding Section 4.1.3. We therefore next derive a slightly wider variance estimator $[\tilde{\alpha}, \tilde{\beta}]$ that indeed suits the requirements of Lemma 4.4 from that section. To this end, observe that there exists a constant $C \geq 1$, depending only on λ, such that for all $w \in \mathbb{N}$, $\sqrt{p + \ln w} \leq C \cdot \sqrt{p} \cdot w^{\lambda/2}$, and thus

$$w + \sigma\sqrt{p + \ln w} \cdot w^{1/2} \leq w + C \cdot \sigma\sqrt{p} \cdot w^{1/2+\lambda/2}.$$

Further, for $w \geq (3 + 8\sigma^2) \cdot \ln(3 + 8\sigma^2)$, it holds that $w/\ln w \geq 4\sigma^2$, so that $\sigma\sqrt{\ln w} \cdot w^{1/2} \leq w/2$, which in turn implies that

$$w - \sigma \cdot \sqrt{\ln w} \cdot w^{1/2} \geq \max\{w/2, w - C \cdot \sigma\sqrt{p} \cdot w^{1/2+\lambda/2}\}.$$

Taking $w_{\min} = \lceil h + (3 + 8\sigma^2) \cdot \ln(3 + 8\sigma^2)\rceil$, the variance estimator

$$[\tilde{\alpha}, \tilde{\beta}]\colon\ w \mapsto \left[\max\{w/2, w - C \cdot \sigma \cdot \sqrt{p} \cdot w^{1/2+\lambda/2}\}, w + C \cdot \sigma \cdot \sqrt{p} \cdot w^{1/2+\lambda/2}\right]$$

hence satisfies $\tilde{\alpha} \leq \alpha$ and $\tilde{\beta} \geq \beta$ at least on $[w_{\min}, \infty)$, which for $\tilde{\delta} = \tilde{\alpha}^{-1} \circ (\tilde{\beta} - \tilde{\alpha})$ and $\tilde{\gamma}_{w_{\min}} = \max\{0, \mathrm{id} - \max\{w_{\min}, (\mathrm{id} + \tilde{\delta})^{-1}\}\}$ is easily seen to imply that $\gamma_{w_{\min}} \leq \tilde{\gamma}_{w_{\min}}$, and thus, by Lemma 3.1, $\gamma^*_{w_{\min}} \leq \tilde{\gamma}^*_{w_{\min}}$. Concerning $\tilde{\gamma}_{w_{\min}}$ we may now apply Lemma 4.4, from which we obtain that

$$\begin{aligned}\tilde{\gamma}^*_{w_{\min}}(n/p) &= \mathcal{O}\left(\log\log(n/p) + \sigma\sqrt{p} \cdot w_{\min}^{\lambda/2-1/2}\right)\\ &= \mathcal{O}\left(\log\log(n/p) + \sigma\sqrt{p} \cdot (h + \sigma^2)^{\lambda-1/2}\right).\end{aligned}$$

Concerning $\beta(w_{\min})$, it is easy to check that

$$\beta(w_{\min}) \leq w_{\min} + C \cdot \sigma\sqrt{p} \cdot w_{\min}^{1/2+\lambda/2} = \mathcal{O}\left(h + \sigma\sqrt{p} \cdot (h + \sigma^2)^{1/2+\lambda}\right).$$

Plugging these bounds into the bound obtained at the beginning of the proof, we finally obtain

$$\mathbf{E}\,\mathrm{waste}(\mathcal{S}) = \mathcal{O}\left((h + \sigma) \cdot \log\log(n/p) + \sigma\sqrt{p} \cdot (h + \sigma^2)^{1/2+\lambda}\right). \qquad \square$$

Since the proof of the corollary above makes use of a variance estimator of sublinear width, the corresponding algorithm is not of the fixed-partition type, but rather one of the more sophisticated instances of our balancing strategy. Since our Main Theorem, in its general form, was established by means of BAL′, the question arises whether the bound of Corollary 4.3 can also be achieved by the original BAL scheme, which we found to be more natural and easier to implement. The following corollary (to Theorem 3.2) gives a positive answer.

Corollary 4.4. *Let task processing times be independent, with variance σ^2, and let the overhead be $h \geq 1$. Then for arbitrary fixed $\lambda > 0$, there exists an instance of* BAL *that for all $n, p \in \mathbb{N}$, given n tasks and p processors, produces a schedule $\mathcal{S}$ with*

$$\mathbf{E}\,\text{waste}(\mathcal{S}) = \mathcal{O}\left((h + \sigma^3/h^2) \cdot \log\log(n/p) + \sigma\sqrt{p} \cdot (h + \sigma^2)^{1/2+\lambda}\right).$$

Proof. According to Theorem 3.2, for every $w_{\min} \in \mathbb{N}$ with $w_{\min} \geq h$, and for the variance estimator

$$[\alpha, \beta]\colon\ w \mapsto [w - \sigma \cdot \sqrt{\ln w} \cdot w^{1/2}, w + \sigma \cdot \sqrt{p + \ln w} \cdot w^{1/2}]$$

there is an instance of BAL that, given n tasks and p processors, produces a schedule $\mathcal{S}$ with

$$\text{waste}(\mathcal{S}) = \mathcal{O}\left((h + \varepsilon) \cdot \gamma^*_{w_{\min}}(n/p) + \beta(w_{\min})\right),$$

where $\gamma_{w_{\min}}$ is the progress rate associated with $[\alpha, \beta]$ and $w_{\min}$, and

$$\varepsilon = (h + \varepsilon_1) \cdot (h + \varepsilon_2) \cdot (h + \varepsilon_3)/h^2 - h,$$

for some partition $\mathcal{S} = \mathcal{S}_1 \dot\cup \mathcal{S}_2 \dot\cup \mathcal{S}_3$, and $\varepsilon_i = \text{av-dev}_{\alpha,\beta}(\mathcal{S}_i)$, for $i = 1, 2, 3$. Now Lemma 4.7 implies that all of $\mathbf{E}[\varepsilon_1]$, $\mathbf{E}[\varepsilon_2 \mid \varepsilon_1]$, and $\mathbf{E}[\varepsilon_3 \mid \varepsilon_1, \varepsilon_2]$ are bounded by $O(\sigma)$ which is easily seen to imply that $\mathbf{E}\,\varepsilon = O(\sigma + \sigma^2/h + \sigma^3/h^2) = O(h + \sigma^3/h^2)$, and hence

$$\mathbf{E}\,\text{waste}(\mathcal{S}) = \mathcal{O}\left((h + \sigma^3/h^2) \cdot \gamma^*_{w_{\min}}(n/p) + \beta(w_{\min})\right).$$

The desired bound now follows by setting $w_{\min} = \lceil h + (3 + 8\sigma^2) \cdot \ln(3 + 8\sigma^2)\rceil$ and estimating $\gamma^*_{w_{\min}}(n/p)$ and $\beta(w_{\min})$ just as done in the proof of the previous corollary. □

4.4. *Coupled Tasks*

If for some $\sigma > 0$ and $T_{\min}$ with $0 < T_{\min} \leq 1$, task processing times are identically distributed random variables with range $[T_{\min}, \infty)$, mean 1, and variance σ^2, and if for each pair of tasks it holds that their processing times are either independent or equal with probability one, then we say that task processing times are *coupled*, *with minimum $T_{\min}$ and variance σ^2*. The corollary below gives an indication that scheduling in the coupled-tasks is much harder than in the independent and bounded-tasks settings. It should be noted, however, that since the corollary below makes no assumptions on the well-behavedness of the distribution of a task's processing time, except that its variance exists, this result is really a worst-case bound. We leave it to the reader to verify that for reasonably behaved distributions (for instance, exponential), significantly better bounds can be achieved.

Lemma 4.8. *Let task processing times be coupled, with minimum $T_{\min}$ and variance σ^2. Then for all $n, p \in \mathbb{N}$, the schedule $\mathcal{S}$ produced by an arbitrary fixed-partition algorithm given n tasks and p processors satisfies*

$$\mathbf{E}[\text{av-dev}_{\alpha,\beta}(\mathcal{S})] \le \sigma^2,$$

where

$$[\alpha, \beta]\colon\ w \mapsto [T_{\min} \cdot w,\ p\,w^2].$$

Proof. Let $w \in \mathbb{N}$, and let $\mathcal{C}$ be a chunk of an arbitrary but fixed selection of w tasks. By assumption, these tasks are divided into some number l of groups such that all tasks from the same group have equal processing times, while processing times of tasks from different groups are independent. Let $w_1, \ldots, w_l \in \mathbb{N}$ be the sizes of the groups, and note that $w_1 + \cdots + w_l = w$. Clearly then the total processing time T of $\mathcal{C}$ has mean w and variance

$$\sigma^2 \cdot w_1^2 + \cdots + \sigma^2 \cdot w_l^2 \le \sigma^2 \cdot (w_1 + \cdots + w_l)^2 = \sigma^2 \cdot w^2,$$

so that, by a simple application of Chebyshev's inequality,

$$\begin{aligned}\mathbf{E}\,\text{late}_\beta(\mathcal{C}) = \mathbf{E}\,\max\{0, T - pw^2\} &= \int_0^\infty \Pr(T - pw^2 > x)\,dx \\ &\le \int_{(p-1)w^2}^\infty \Pr(T - w > x)\,dx \\ &\le \int_{(p-1)w^2}^\infty (\sigma^2 w^2/x^2)\,dx \\ &\le \sigma^2/(p-1).\end{aligned}$$

Since there is never earliness with respect to α, we have thus proven that the expected deviation of an arbitrary fixed chunk is bounded by σ^2. This immediately implies the same bound for the expected average deviation of the schedule produced by an arbitrary fixed-partition algorithm. □

Corollary 4.5. *Let task processing times be coupled, with minimum $T_{\min}$ and variance σ^2, and let the overhead be $h \ge 1$. Then there exists a fixed-partition algorithm that for all $n, p \in \mathbb{N}$ with $n/p \ge p \cdot (h + \sigma^2)^2/T_{\min}^3$, given n tasks and p processors, produces a schedule $\mathcal{S}$ with*

$$\mathbf{E}\,\text{waste}(\mathcal{S}) = \mathcal{O}\left((h + \sigma^2) \cdot \sqrt{n/T_{\min}}\right).$$

Proof. A simple application of the Main Theorem in combination with the previous lemma yields an algorithm that for n tasks and p processors produces a schedule $\mathcal{S}$ with

$$\mathbf{E}\,\text{waste}(\mathcal{S}) = \mathcal{O}\left((h + \sigma^2) \cdot \gamma^*_{w_{\min}}(n/p) + \beta(w_{\min})\right),$$

where $\gamma_{w_{\min}}$ is the progress rate associated with the variance estimator $w \mapsto [T_{\min} \cdot w,\, pw^2]$ and the minimal chunk size $w_{\min} = \lceil (h + \sigma^2)/T_{\min} \rceil$. According to Lemma 4.1,

$$\gamma^*_{w_{\min}}(n/p) = \mathcal{O}\left(p^{1/2}/T_{\min}^{1/2} \cdot (n/p)^{1/2}\right) = \mathcal{O}(n^{1/2}/T_{\min}^{1/2}),$$

so that, for $n/p \geq p \cdot (h + \sigma^2)^2/T_{\min}^3$,

$$\begin{aligned}\mathbf{E}\,\mathrm{waste}(\mathcal{S}) &= \mathcal{O}\left((h+\sigma^2) \cdot \sqrt{n/T_{\min}} + p \cdot (h+\sigma^2)^2/T_{\min}^2\right)\\ &= \mathcal{O}\left((h+\sigma^2) \cdot \sqrt{n/T_{\min}}\right). \qquad \square\end{aligned}$$

5. Lower Bounds

This section complements our findings from the previous two sections with matching or almost matching lower bounds. In Section 5.1 we show that for each variance estimator, no algorithm can improve by more than a constant factor on the wasted-time bound stated in our Main Theorem; this implies the optimality of the balancing strategy, at least within the realm of our modelling. Section 5.2 presents a general lower bound for the case when task processing times are randomly distributed. Note that, unlike for our upper bounds, we cannot hope to obtain such a lower bound via reduction from a lower bound pertaining to our general framework; namely, as was explained in the Introduction, compared with our variance-estimator based approach, probabilistic assumptions add a certain regularity to the problem, which makes proving lower bounds harder. Indeed, the results from Section 5.2 will leave a small gap to the upper bounds proven in the previous section.

5.1. *Arbitrary Processing Times*

This section is dedicated to proving the following theorem, which provides the exactly matching lower bound to our Main Theorem. As we already remarked at the beginning of Section 3, this lower bound implies that the optimal choice for the minimal chunk size $w_{\min}$ is in the order of $\alpha^{-1}(h + \varepsilon)$. Note that while the Main Theorem requires that id/α be a decreasing function, the (addendum to the) theorem below makes do with the superadditivity of α; this is indeed a weaker requirement, since for arbitrary $w \geq v > 0$, it follows from $\alpha(w)/w \geq \alpha(v)/v$ that

$$\begin{aligned}\alpha(w+v) &\geq (w+v) \cdot \alpha(w)/w = \alpha(w) + v \cdot \alpha(w)/w \geq \alpha(w) + v \cdot \alpha(v)/v\\ &= \alpha(w) + \alpha(v).\end{aligned}$$

Theorem 5.1. *Let processing times be arbitrary, let the overhead be h, and let $[\alpha, \beta]$ be an arbitrary variance estimator. Then for every $\varepsilon \geq 0$, for every scheduling algorithm $\mathcal{A}$, and for all $n, p \in \mathbb{N}$, there exist $T_1, \ldots, T_n \geq 0$ such that, given n tasks with processing times $T_1, \ldots, T_n$ and p processors, $\mathcal{A}$ produces a schedule $\mathcal{S}$ with $\varepsilon = \text{av-dev}_{\alpha,\beta}(\mathcal{S}) =$*

am-$\text{dev}_{\alpha,\beta}(\mathcal{S})$ *and*

$$\text{waste}(\mathcal{S}) = \Omega\left((h+\varepsilon)\cdot\gamma^*(\alpha(n/p))\right),$$

where $\gamma = \text{id} - \max\{h+\varepsilon, \alpha\circ\beta^{-1}\}$.

Addendum. If, additionally, α is superadditive, that is, for all $w, v > 0$, $\alpha(w+v) \geq \alpha(w)+\alpha(v)$, and provided that β is a bijection on $\mathbb{R}^+$, it holds that

$$\gamma^*(\alpha(n/p)) \geq \gamma^*_{\alpha^{-1}(h+\varepsilon)}(n/p),$$

where $\gamma_{\alpha^{-1}(h+\varepsilon)}$ denotes the progress rate associated with $[\alpha, \beta]$ and $\alpha^{-1}(h+\varepsilon)$.

The proof of Theorem 5.1 is organized as follows. In Section 5.1.1 we first prove the lower bound under the assumption that the given algorithm does not incur any *waiting time*. Section 5.1.2 shows how to extend this proof to the general case. The final section, Section 5.1.3, is concerned with the proof of the addendum that translates the proven bounds to a form compatible with our Main Theorem. Throughout the proof, the *lower* and *upper threshold* of a chunk $\mathcal{C}$ of size w and scheduled at a time T mean the times $T+h+\alpha(w)$ and $T+h+\beta(w)$, denoted by lower($\mathcal{C}$) and upper($\mathcal{C}$), respectively.

5.1.1. *The Case Without Waiting.* The basic and rather obvious idea of the proof is to play an adversary and fix the chunk processing times (and hence the $T_1,\dots,T_n$) incrementally, along with the scheduling decisions made by our algorithm. We next describe this construction in detail. Though we need not fix the processing time of a chunk right at the time of its allocation, we usually do that, except for one designated *peak chunk*, for which the decision is postponed. Initially, the very first chunk assigned becomes the (first) peak chunk. Whenever a new chunk $\mathcal{C}_{\text{new}}$ is scheduled, its upper threshold upper($\mathcal{C}_{\text{new}}$) is compared with that of the current peak chunk $\mathcal{C}_{\text{peak}}$: if upper($\mathcal{C}_{\text{new}}$) $\leq$ upper($\mathcal{C}_{\text{peak}}$), the finishing time of $\mathcal{C}_{\text{new}}$ is fixed immediately at its lower threshold lower($\mathcal{C}_{\text{new}}$); in the opposite case, $\mathcal{C}_{\text{new}}$ becomes the new peak chunk, and the finishing time of $\mathcal{C}_{\text{peak}}$ is fixed at the maximum of lower($\mathcal{C}_{\text{peak}}$) and the actual time. Note that as a consequence the upper threshold of a peak chunk is always larger than that of its predecessor. The processing time of the last peak chunk $\mathcal{C}_{\text{last}}$, finally, is fixed at $\beta(w)+\varepsilon\cdot l/(p-1)$, where w is the size of $\mathcal{C}_{\text{last}}$ and l is the total number of chunks scheduled. This finishes the description of our incremental construction, and we now have to verify that the resulting schedule $\mathcal{S}$ indeed has the properties stated in the theorem. Since all chunks have deviation zero, except the last one, whose deviation is $\varepsilon\cdot l$, we immediately see that av-$\text{dev}_{\alpha,\beta}(\mathcal{S})$ = am-$\text{dev}_{\alpha,\beta}(\mathcal{S}) = \varepsilon$. The remainder of this proof derives the desired lower bound on the wasted time of $\mathcal{S}$.

To this end, we introduce the notions of the *peak* and the *lead* of a (partial) schedule, where the former is simply the upper threshold of the peak chunk, and the latter measures the lead of this peak chunk on the other chunks. Formally, if $\mathcal{S}'$ denotes a *prefix* of $\mathcal{S}$, that is, a sequence of chunks assigned until some time in the scheduling process, and if

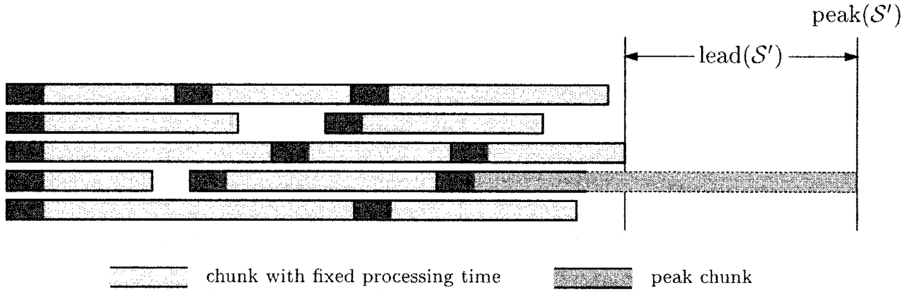

Fig. 10. The peak chunk, peak, and lead of a schedule $\mathcal{S}'$.

$\mathcal{C}_{\text{peak}}$ is the peak chunk of $\mathcal{S}'$, that is, the peak chunk at that time, we define

$$\begin{aligned}\text{peak}(\mathcal{S}') &= \text{upper}(\mathcal{C}_{\text{peak}}),\\ \text{lead}(\mathcal{S}') &= \text{peak}(\mathcal{S}') - \max_{\mathcal{C}\in\mathcal{S}'\setminus\mathcal{C}_{\text{peak}}} \text{finish}(\mathcal{C}).\end{aligned}$$

This is indeed well-defined, since, by the above construction, the finishing time of all chunks except the peak chunk are fixed immediately at the time of allocation. For an illustration, see Figure 10.

Note that since the upper threshold of the peak chunk is maximal among the upper thresholds of the chunks in $\mathcal{S}'$, the lead according to this definition is always positive. Also observe that the lead of a schedule is intimately related to its imbalance: namely, $\text{imbalance}(\mathcal{S}') \ge (p-1)\cdot \text{lead}(\mathcal{S}')$, for every prefix $\mathcal{S}'$ of $\mathcal{S}$, and, for the complete schedule,

$$\begin{aligned}\text{imbalance}(\mathcal{S}) &\ge (p-1)\cdot(\text{lead}(\mathcal{S}) + \varepsilon\cdot \text{chunks}(\mathcal{S})/(p-1))\\ &= (p-1)\cdot \text{lead}(\mathcal{S}) + \varepsilon\cdot\text{chunks}(\mathcal{S}).\end{aligned}$$

Using the above definitions, we prove a lower bound on waste($\mathcal{S}$) as follows. First, Lemma 5.1 demonstrates that the scheduling of a large chunk incurs a correspondingly large lead. Following that, Lemma 5.2 shows that the lead cannot decrease arbitrarily fast from one batch of allocations to the next. Using these two lemmas, Lemma 5.3 derives a bound on the lead of $\mathcal{S}$, proceeding from which we then argue that either many (small) chunks are assigned or the final lead is large. Throughout the proof, $\tilde{\gamma}$ denotes the function $x \mapsto (\text{id} - \alpha\circ\beta^{-1})(x - h - \varepsilon)$; check that since β^{-1} and $\beta - \alpha$ are increasing functions, the same applies to $(\beta-\alpha)\circ\beta^{-1} = \text{id} - \alpha\circ\beta^{-1}$, and hence to $\tilde{\gamma}$.

Lemma 5.1. *For an arbitrary prefix $\mathcal{S}'$ of $\mathcal{S}$, if w denotes the size of the chunk of $\mathcal{S}'$ that was scheduled last, then* $\text{lead}(\mathcal{S}') \ge \beta(w) - \alpha(w)$.

Proof. Let T denote the scheduling time of the chunk of $\mathcal{S}'$ that was scheduled last. The upper threshold of this chunk is $T + h + \beta(w)$, so that, by definition of the peak, $\text{peak}(\mathcal{S}') \ge T + h + \beta(w)$. Now for an arbitrary chunk $\mathcal{C}$ of $\mathcal{S}'$ that is not the peak chunk of $\mathcal{S}'$, the following holds. If the size of $\mathcal{C}$ is below w, its finishing time, fixed at time T at the latest, is at most

$$T + h + \alpha(w) \leq \text{peak}(\mathcal{S}') - \beta(w) + \alpha(w) = \text{peak}(\mathcal{S}') - (\beta(w) - \alpha(w)).$$

If the size of $\mathcal{C}$ is at least w, then the difference between the upper and the lower threshold of $\mathcal{C}$ is at least $\beta(w) - \alpha(w)$, since $\beta - \alpha$ is increasing. The finishing time of $\mathcal{C}$ is therefore at most

$$\max\left\{T, \text{peak}(\mathcal{S}') - (\beta(w) - \alpha(w))\right\} = \text{peak}(\mathcal{S}') - (\beta(w) - \alpha(w)).$$

This proves that the lead of $\mathcal{S}'$ is at least $\beta(w) - \alpha(w)$. $\square$

Lemma 5.2. *For two arbitrary prefixes $\mathcal{S}'$, $\mathcal{S}''$ of $\mathcal{S}$ with* $\text{chunks}(\mathcal{S}'') - \text{chunks}(\mathcal{S}') \leq p - 1$,

$$\text{lead}(\mathcal{S}'') \geq \tilde{\gamma}(\text{lead}(\mathcal{S}')).$$

Proof. The key to this proof is the simple observation that in the case without waiting each of the at most $p - 1$ chunks in $\mathcal{S}''\backslash\mathcal{S}'$ is scheduled before or at time $\text{peak}(\mathcal{S}') - \text{lead}(\mathcal{S}')$; see Figure 10. Let $\mathcal{C}$ denote an arbitrary such chunk except the peak chunk of $\mathcal{S}''$, and let w denote its size. Clearly then, its upper threshold cannot be more than $\text{peak}(\mathcal{S}'')$, and by the observation above, its lower threshold is at most $\text{peak}(\mathcal{S}') - \text{lead}(\mathcal{S}') + h + \alpha(w)$. Hence, using that $\text{peak}(\mathcal{S}') \leq \text{peak}(\mathcal{S}'')$,

$$\begin{aligned}
\text{peak}(\mathcal{S}'') - \text{lower}(\mathcal{C}) &\geq \max\{\text{lead}(\mathcal{S}') - h - \alpha(w), \text{upper}(\mathcal{C}) - \text{lower}(\mathcal{C})\}\\
&= \max\{\text{lead}(\mathcal{S}') - h - \alpha(w), \beta(w) - \alpha(w)\}\\
&\geq \text{lead}(\mathcal{S}') - h - (\alpha \circ \beta^{-1})(\text{lead}(\mathcal{S}') - h)\\
&\geq \tilde{\gamma}(\text{lead}(\mathcal{S}')).
\end{aligned}$$

Here the penultimate inequality follows from the fact that the decreasing function $w \mapsto \text{lead}(\mathcal{S}') - h - \alpha(w)$ intersects the increasing function $w \mapsto \beta(w) - \alpha(w)$ at $w = \beta^{-1}(\text{lead}(\mathcal{S}') - h)$. Since the finishing time $\text{finish}(\mathcal{C})$ of $\mathcal{C}$ is fixed at time $\max\{\text{lower}(\mathcal{C}), \text{peak}(\mathcal{S}') - \text{lead}(\mathcal{S}')\}$ at the latest, we thus obtain

$$\text{peak}(\mathcal{S}'') - \text{finish}(\mathcal{C}) \geq \min\{\text{peak}(\mathcal{S}'') - \text{lower}(\mathcal{C}), \text{lead}(\mathcal{S}')\} \geq \tilde{\gamma}(\text{lead}(\mathcal{S}')).$$

By the definition of the lead, this immediately implies that $\text{lead}(\mathcal{S}'') \geq \tilde{\gamma}(\text{lead}(\mathcal{S}'))$. $\square$

Lemma 5.3. *With $w_{\max}$ denoting the maximal size of a chunk in $\mathcal{S}$, and* $r = \lceil \text{chunks}(\mathcal{S})/(p-1)\rceil$,

$$\text{lead}(\mathcal{S}) \geq \tilde{\gamma}^{(r+1)}(\beta(w_{\max})),$$

and thus, by construction,

$$\mathrm{idle}(\mathcal{S}) \geq (p-1) \cdot \tilde{\gamma}^{(r+1)}(\beta(w_{\max})) + \varepsilon \cdot \mathrm{chunks}(\mathcal{S}).$$

Proof. Let $\mathcal{S}'$ be a prefix of $\mathcal{S}$ such that the chunk of $\mathcal{S}'$ scheduled last has size $w_{\max}$. Then, by Lemma 5.1,

$$\mathrm{lead}(\mathcal{S}') \geq \beta(w_{\max}) - \alpha(w_{\max}),$$

and since $\mathcal{S}\backslash\mathcal{S}'$ contains at most $\mathrm{chunks}(\mathcal{S}) \leq r \cdot (p-1)$ chunks, repeated application of Lemma 5.2 yields that

$$\mathrm{lead}(\mathcal{S}) \geq \tilde{\gamma}^{(r)}(\mathrm{lead}(\mathcal{S}')) \geq \tilde{\gamma}^{(r)}(\beta(w_{\max}) - \alpha(w_{\max})).$$

From that, the desired bound follows owing to $\tilde{\gamma}(\beta(w_{\max})) \leq (\mathrm{id} - \alpha \circ \beta^{-1})(\beta(w_{\max})) = \beta(w_{\max}) - \alpha(w_{\max})$. □

We are now ready to derive a lower bound on the wasted time of $\mathcal{S}$. Namely, with r and $w_{\max}$ defined as in the last lemma, the number of chunks in $\mathcal{S}$ is at least $(p-1)\cdot(r-1)$, and we immediately obtain that

$$\begin{aligned}\mathrm{waste}(\mathcal{S}) &= (1/p) \cdot (h \cdot \mathrm{chunks}(\mathcal{S}) + \mathrm{idle}(\mathcal{S})) \\ &\geq ((p-1)/p) \cdot \left((h+\varepsilon) \cdot (r-1) + \tilde{\gamma}^{(r+1)}(\beta(w_{\max}))\right).\end{aligned}$$

To eliminate r, check that because $\tilde{\gamma}$ always decreases its argument by at least $h+\varepsilon$, for all $i \in \mathbb{N}$ and $x > 0$,

$$\tilde{\gamma}^*(x) \leq i + \lceil \tilde{\gamma}^{(i)}(x)/(h+\varepsilon) \rceil \leq i + 1 + \tilde{\gamma}^{(i)}(x)/(h+\varepsilon),$$

hence with $i = r+1$ and $x = \beta(w_{\max})$,

$$\tilde{\gamma}^{(r+1)}(\beta(w_{\max})) + (h+\varepsilon) \cdot (r+2) \geq (h+\varepsilon) \cdot \tilde{\gamma}^*(\beta(w_{\max})).$$

This, in turn, implies the lower bound

$$\mathrm{waste}(\mathcal{S}) \geq ((p-1)/p) \cdot (h+\varepsilon) \cdot \left(\tilde{\gamma}^*(\beta(w_{\max})) - 3\right).$$

Two items remain in order to prove Theorem 5.1. First, to relate $\tilde{\gamma}^*$ to γ^*, where $\gamma = \mathrm{id} - \max\{h+\varepsilon, \alpha \circ \beta^{-1}\}$, and, second, to resolve the dependency on the (unknown) $w_{\max}$. For the first item, just observe that for all $x \geq 0$,

$$\begin{aligned}\gamma^{(2)}(x) &\leq \gamma(x) - h - \varepsilon \leq (\mathrm{id} - \alpha \circ \beta^{-1})(x) - h - \varepsilon \\ &\leq (\mathrm{id} - \alpha \circ \beta^{-1})(x - h - \varepsilon) = \tilde{\gamma}(x),\end{aligned}$$

which immediately implies that $\gamma^*(x) \leq 2 \cdot \tilde{\gamma}^*(x)$. In order to eliminate $w_{\max}$, we make use of the trivial lower bound on $\mathrm{chunks}(\mathcal{S})$ of $\lceil n/w_{\max} \rceil$. We then have $\mathrm{idle}(\mathcal{S}) \geq \varepsilon \cdot n/w_{\max}$ and $\mathrm{waste}(\mathcal{S}) \geq (h+\varepsilon) \cdot (n/p)/w_{\max}$, so that in combination with the bound above on $\mathrm{waste}(\mathcal{S})$ we obtain

$$\mathrm{waste}(\mathcal{S}) = \Omega\left((h+\varepsilon) \cdot \left((n/p)/w_{\max} + \gamma^*(\beta(w_{\max})) \right)\right).$$

Resolving the dependency on $w_{\max}$ is now a matter of proving the following somewhat amazing lemma. Note that since $\mathrm{id}/A \le \alpha \le \mathrm{id}$, it holds that $\alpha(n/p)/\alpha(w_{\max}) \le A \cdot (n/p)/w_{\max}$.

Lemma 5.4. $\alpha(n/p)/\alpha(w_{\max}) + \gamma^*(\beta(w_{\max})) \ge \gamma^*(\alpha(n/p))$.

Proof. The proof is trivial if either $\beta(w_{\max}) \ge \alpha(n/p)$ or $\alpha(w_{\max}) \le h + \varepsilon$, so we assume in the following that $\alpha(n/p) > \beta(w_{\max})$ and $w_{\max} > \alpha^{-1}(h+\varepsilon)$. Then we can choose $i \in \mathbb{N}$ minimal such that $\gamma^{(i)}(\alpha(n/p)) \le \beta(w_{\max})$, so that, in particular,

$$\gamma^*(\alpha(n/p)) \le i + \gamma^*(\beta(w_{\max})).$$

Besides, it holds that $I = \gamma^{(i-1)}(\alpha(n/p)) > \beta(w_{\max})$ and thus $0 \le \gamma^{(i)}(\alpha(n/p)) \le \alpha(n/p) - i \cdot \max\{h+\varepsilon, \alpha \circ \beta^{-1}(I)\}$, which, since $\alpha \circ \beta^{-1}(I) \ge \alpha(w_{\max}) \ge h + \varepsilon$ by assumption, implies that

$$0 \le \alpha(n/p) - i \cdot \alpha(w_{\max}).$$

In combination with the above we thus obtain

$$\gamma^*(\alpha(n/p)) \le i + \gamma^*(\beta(w_{\max})) \le \alpha(n/p)/\alpha(w_{\max}) + \gamma^*(\beta(w_{\max})). \qquad \square$$

This finishes the proof of Theorem 5.1 under the constraint that an algorithm may not incur any waiting time between any two chunks successively assigned to the same processor. In the next section we adapt the argumentation above to the case of arbitrary scheduling algorithms.

5.1.2. *The General Case.* In view of possible waiting times, we need to complement our incremental construction of the chunk processing times by the description of a (very) special case, which could not have occurred so far. Namely, it may now happen—even if not very meaningfully so—that after the selection of some peak chunk, the next chunk is assigned at a time T' *after* the upper threshold T of that peak chunk; in particular, then, *all* processors wait between T and T'. Our action in that case will simply be to make the new chunk the peak chunk, and to fix the finishing time of the old peak chunk at its upper threshold (and not at T').

In view of the general setting, it is easy to see that Lemma 5.1 holds without changes, while for Lemmas 5.2 and 5.3 a correcting term now has to be added. To enable a concise statement of the modified statements we agree to define, for an arbitrary schedule $\mathcal{S}'$, and for arbitrary nonnegative T', T'',

$$\mathrm{idle}_{[T',T'']}(\mathcal{S}')$$

as the total amount of idle time of $\mathcal{S}'$ spent in the time interval $[T', T'']$. This is consistent with our definition from Section 2.1 in the sense that with $T = \mathrm{makespan}(\mathcal{S}')$, $\mathrm{idle}(\mathcal{S}') = \mathrm{idle}_{[0,T]}(\mathcal{S}')$.

Lemma 5.5. *For two arbitrary prefixes $\mathcal{S}', \mathcal{S}''$ of $\mathcal{S}$ such that* $\text{chunks}(\mathcal{S}') < \text{chunks}(\mathcal{S}'') \leq \text{chunks}(\mathcal{S}') + \lfloor p/2 \rfloor$,

$$\text{lead}(\mathcal{S}'') \geq \tilde{\gamma}(\text{lead}(\mathcal{S}')) - 2 \cdot \text{idle}_{[T',T'']}(\mathcal{S})/p,$$

where $T' = \text{peak}(\mathcal{S}') - \text{lead}(\mathcal{S}')$, *and* $T'' = \text{peak}(\mathcal{S}'') - \text{lead}(\mathcal{S}'')$.

Proof. Let T denote the time of the latest allocation in $\mathcal{S}''$, so that for an arbitrary chunk $\mathcal{C} \in \mathcal{S}'' \backslash \mathcal{S}'$ with size w,

$$\text{lower}(\mathcal{C}) \leq T + h + \alpha(w).$$

Without waiting, we would have $T \leq T' = \text{peak}(\mathcal{S}') - \text{lead}(\mathcal{S}')$, as in the proof of Lemma 5.2. Now that waiting is allowed, we make use of the following argument. By the definition of the lead, there are $p - 1$ processors whose chunks of $\mathcal{S}'$ finish before or at $T' = \text{peak}(\mathcal{S}') - \text{lead}(\mathcal{S}')$, and since $|\mathcal{S}'' \backslash \mathcal{S}'| \leq \lfloor p/2 \rfloor$, and by the definition of T, at least $1 + p - 1 - \lfloor p/2 \rfloor = \lceil p/2 \rceil$ of these are not assigned another chunk before T. Therefore,

$$\text{idle}_{[T',T]}(\mathcal{S}) \geq p/2 \cdot (T - T'),$$

and thus, using that $T \leq \text{peak}(\mathcal{S}'') - \text{lead}(\mathcal{S}'') = T''$,

$$\begin{aligned}\text{lower}(\mathcal{C}) &\leq T + h + \alpha(w)\\ &\leq T' + 2 \cdot \text{idle}_{[T',T]}(\mathcal{S})/p + h + \alpha(w)\\ &\leq \text{peak}(\mathcal{S}') - \text{lead}(\mathcal{S}') + 2 \cdot \text{idle}_{[T',T'']}(\mathcal{S})/p + h + \alpha(w).\end{aligned}$$

From this we deduce, analogously to the proof of Lemma 5.2,

$$\begin{aligned}&\text{peak}(\mathcal{S}'') - \text{lower}(\mathcal{C})\\ &\quad\geq \max\left\{\text{lead}(\mathcal{S}') - h - 2 \cdot \text{idle}_{[T',T'']}(\mathcal{S})/p - \alpha(w), \beta(w) - \alpha(w)\right\}\\ &\quad\geq \tilde{\gamma}(\text{lead}(\mathcal{S}') - 2 \cdot \text{idle}_{[T',T'']}(\mathcal{S})/p),\end{aligned}$$

which implies the same bound for the lead of $\mathcal{S}''$. It remains to appeal to the sublinearity property of $\tilde{\gamma}$, according to which $\tilde{\gamma}(x - y) \geq \tilde{\gamma}(x) - y$, for all $x, y \geq 0$. □

Lemma 5.6. *With $w_{\max}$ denoting the maximal size of a chunk of $\mathcal{S}$, and* $r = \lceil \text{chunks}(\mathcal{S}) / \lfloor p/2 \rfloor \rceil$,

$$\text{lead}(\mathcal{S}) \geq \tilde{\gamma}^{(r+1)}(\beta(w_{\max})) - 2 \cdot \text{idle}_{[0,T]}(\mathcal{S})/p,$$

where $T = \text{peak}(\mathcal{S}) - \text{lead}(\mathcal{S})$, *and thus*

$$\text{idle}(\mathcal{S}) \geq (p - 1) \cdot \tilde{\gamma}^{(r+1)}(\beta(w_{\max}))/2 + \varepsilon \cdot \text{chunks}(\mathcal{S}).$$

Proof. As in the proof of corresponding Lemma 5.3, there exists a prefix $\mathcal{S}'$ of $\mathcal{S}$ for which, by Lemma 5.1,

$$\text{lead}(\mathcal{S}') \geq \beta(w_{\max}) - \alpha(w_{\max}),$$

so that, by iterated application of Lemma 5.5 making use of the sublinearity property of $\tilde{\gamma}$,

$$\begin{aligned}\text{lead}(\mathcal{S}) &\geq \tilde{\gamma}^{(r)}(\beta(w_{\max}) - \alpha(w_{\max})) - 2 \cdot \text{idle}_{[0,T]}(\mathcal{S})/p \\ &\geq \tilde{\gamma}^{(r+1)}(\beta(w_{\max})) - 2 \cdot \text{idle}_{[0,T]}(\mathcal{S})/p.\end{aligned}$$

The bound on the idle time follows, since

$$\text{idle}(\mathcal{S}) = \text{idle}_{[0,T]}(\mathcal{S}) + (p-1) \cdot \text{lead}(\mathcal{S}) + \varepsilon \cdot \text{chunks}(\mathcal{S}),$$

and, because the lead is never negative,

$$\text{lead}(\mathcal{S}) \geq \text{lead}(\mathcal{S})/2 \geq \tilde{\gamma}^{(r+1)}(\beta(w_{\max}))/2 - \text{idle}_{[0,T]}(\mathcal{S})/p. \qquad \square$$

As before, we easily obtain from this lemma a lower bound on the wasted time of $\mathcal{S}$. Namely, using that $\text{chunks}(\mathcal{S}) \geq (r-1) \cdot \lfloor p/2 \rfloor \geq (r-1) \cdot (p-1)/2$,

$$\begin{aligned}\text{waste}(\mathcal{S}) &= (1/p) \cdot (h \cdot \text{chunks}(\mathcal{S}) + \text{idle}(\mathcal{S})) \\ &\geq ((p-1)/2p) \cdot \left((h+\varepsilon) \cdot (r-1) + \tilde{\gamma}^{(r+1)}(\beta(w_{\max}))\right),\end{aligned}$$

which differs by a factor of exactly 2 from the bound obtained after Lemma 5.3 in the proof for the case without waiting. The very same manipulations as used before will therefore lead to the bound stated in the theorem.

5.1.3. *Matching the Upper Bound.* This final section is concerned with proving the addendum to Theorem 5.1, which translates the bound proven above to a form compatible with our Main Theorem. We first show, by a tricky combination of simple algebraic manipulations, that

$$\alpha \circ \gamma_{\alpha^{-1}(h+\varepsilon)} \leq \gamma \circ \alpha,$$

from which the addendum will follow easily. We start by observing that because $\text{id} + (\beta - \alpha) \circ \alpha^{-1} = \beta \circ \alpha^{-1}$,

$$\text{id} = (\text{id} + (\beta - \alpha) \circ \alpha^{-1}) \circ \alpha \circ \beta^{-1}.$$

Owing to the fact that α is superadditive and hence α^{-1} is subadditive, it holds that, with $\delta = \alpha^{-1} \circ (\beta - \alpha)$,

$$\begin{aligned}\text{id} &= \alpha^{-1} \circ (\text{id} + (\beta - \alpha) \circ \alpha^{-1}) \circ \alpha \circ \beta^{-1} \circ \alpha \\ &\leq (\alpha^{-1} \circ \text{id} + \alpha^{-1} \circ (\beta - \alpha) \circ \alpha^{-1}) \circ \alpha \circ \beta^{-1} \circ \alpha \\ &= (\text{id} + \delta) \circ \beta^{-1} \circ \alpha,\end{aligned}$$

and therefore

$$\begin{aligned}\alpha \circ (\mathrm{id} - (\mathrm{id} + \delta)^{-1}) &\le \alpha \circ (\mathrm{id} - (\mathrm{id} + \delta)^{-1}) \circ (\mathrm{id} + \delta) \circ \beta^{-1} \circ \alpha \\ &= \alpha \circ \delta \circ \beta^{-1} \circ \alpha \\ &= (\mathrm{id} - \alpha \circ \beta^{-1}) \circ \alpha.\end{aligned}$$

Similarly, $\alpha \le \mathrm{id}$ and the superadditivity property of α imply that

$$\alpha \circ (\mathrm{id} - \alpha^{-1}(h + \varepsilon)) \le \alpha - (h + \varepsilon) \le \alpha - (h + \varepsilon) \circ \alpha = (\mathrm{id} - (h + \varepsilon)) \circ \alpha.$$

Altogether, since for arbitrary functions $f_1,\ f_2, g_1, g_2,\ f_1 \le g_1$ and $f_2 \le g_2$ together imply that $\min\{f_1,\ f_2\} \le \min\{g_1, g_2\}$, we obtain that

$$\begin{aligned}&\min\left\{\alpha \circ (\mathrm{id} - (\mathrm{id} + \delta)^{-1}), \alpha \circ (\mathrm{id} - \alpha^{-1}(h + \varepsilon))\right\} \\ &\qquad \le \min\left\{\left(\mathrm{id} - \alpha \circ \beta^{-1}\right) \circ \alpha, (\mathrm{id} - (h + \varepsilon)) \circ \alpha\right\}.\end{aligned}$$

By the monotonicity of α, $\min\{\alpha \circ f, \alpha \circ g\} = \alpha \circ \min\{f, g\}$ and $\min\{f \circ \alpha, g \circ \alpha\} = \min\{f, g\} \circ \alpha$, for arbitrary functions f and g, so that the last inequality may be rewritten as

$$\alpha \circ (\mathrm{id} - \max\{\alpha^{-1}(h + \varepsilon), (\mathrm{id} + \delta)^{-1}\}) \le (\mathrm{id} - \max\{h + \varepsilon, \alpha \circ \beta^{-1}\}) \circ \alpha.$$

We have thus proven that

$$\alpha \circ \gamma_{\alpha^{-1}(h+\varepsilon)} \le \gamma \circ \alpha,$$

which, via a simple induction, is easily seen to imply that for all $i \in \mathbb{N}$,

$$\alpha \circ \gamma^{(i)}_{\alpha^{-1}(h+\varepsilon)} \le \gamma^{(i)} \circ \alpha.$$

Hence for arbitrary $x > 0$, and for all $i \in \mathbb{N}$,

$$\gamma^{(i)}(\alpha(x)) \le 0 \quad \Rightarrow \quad \alpha \circ \gamma^{(i)}_{\alpha^{-1}(h+\varepsilon)}(x) \le 0 \quad \Rightarrow \quad \gamma^{(i)}_{\alpha^{-1}(h+\varepsilon)}(x) \le 0,$$

and we have finally proven that

$$\gamma^*(\alpha(n/p)) \ge \gamma^*_{\alpha^{-1}(h+\varepsilon)}(n/p).$$

This finishes the proof of Theorem 5.1. □

5.2. *Randomly Distributed Processing Times*

For the lower bound proof given in the previous section, we fixed chunk processing times at both ends of the estimated ranges $[\alpha(w), \beta(w)]$. In a sense, the proof thus exploited the full generality of our variance-estimator-based model, so that we cannot expect the obtained result to translate easily to a setting where processing times are randomly distributed. However, as is shown in Section 5.2.1 below, a rather simple argument suffices to prove a surprisingly tight general lower bound on the expected wasted time for randomly distributed task processing times. In Section 5.2.2 we derive from this general result lower bounds for two specific instances of the independent-tasks and the coupled-tasks setting (note that for the bounded-tasks setting, which is nonprobabilistic, Theorem 5.1 applies).

5.2.1. *General Bound.* Like Theorem 5.1 from the previous section, Theorem 5.2 below is also formulated in terms of the progress rate of a variance estimator $[\alpha, \beta]$. Intuitively, the ranges $[\alpha(w), \beta(w)]$ now describe the concentration of processing times of chunks of size w around their mean; the corresponding requirement of Theorem 5.2 is that such processing times have a certain likelihood to lie below $\alpha(w)$ as well as to lie above $\beta(w)$.

Theorem 5.2. *Let task processing times be randomly distributed, with mean* 1, *let the overhead be* h, *and assume that there exist* $K \geq 1$ *and a variance estimator* $[\alpha, \beta]$ *such that for all* $w \in \mathbb{N}$, *it holds that for the total processing time* T *of* w *tasks*,

$$\min\{\Pr(T \leq \alpha(w)), \Pr(T \geq \beta(w))\} \geq 1/K.$$

Then for all $n, p \in \mathbb{N}$, *and for an arbitrary algorithm that, given* n *tasks and* p *processors, produces a schedule* $\mathcal{S}$ *such that the processing times of the chunks of* $\mathcal{S}$ *are independent, it holds that*

$$\mathbf{E}\,\text{waste}(\mathcal{S}) = \Omega\left((h \cdot \gamma^*(n)/p)/K\right),$$

where γ *denotes the progress rate associated with* $[\alpha, \beta]$ *and* h.

The proof of this theorem is somewhat akin to that of Theorem 5.1 but not analogous. The similarity is that both proofs show that either relatively small and hence many chunks are allocated, or the final imbalance is likely to be large. For Theorem 5.1 this was realized by showing first that a large chunk induces a large peak, and second that the peak can only decrease at a certain fixed rate from one batch of allocations to the next (the difference between lead and imbalance is not essential at this point). However, as we already remarked above, the proof of the last assertion made use of the full power of the variance-estimator-based model. Instead, the following proof is based on the argument that, intuitively speaking, a large chunk is likely to cause an imbalance too large to be rebalanced by the remaining work. In fact, Theorem 5.1 could also have been proven along this line of argumentation, however, with somewhat more effort. While the proof of Theorem 5.1 considered batches of $\Theta(p)$ scheduling operations, the proof below takes a more simplistic approach by coarsely quantifying the effect of individual scheduling operations. This accounts for a loss of accuracy in our bounds that is on the order of the number of processors.

Technically, the proof is organized as follows. With Lemma 5.7 we first provide a formalization of the pretty intuitive (and nonprobabilistic) fact that whenever the imbalance is large and the processing time of the remaining work is small, the wasted time is bound to be large. After that we prove the key Lemma 5.8 saying that a too large chunk causes a large expected wasted time. From this result it will be straightforward to deduce the theorem.

Lemma 5.7. *For an arbitrary schedule $\mathcal{S}'$ on p processors, with initial imbalance I and total processing time T,*

$$p \cdot \text{waste}(\mathcal{S}') \geq I - T.$$

Proof. For $k = 1, \ldots, p$, we denote by H_k the sum of all overheads and waiting times of the kth processor, by T_k its total processing time, and by t_k the time when it first becomes idle initially. Then $T = \sum_{k=1}^{p} T_k$ and, by the definition of initial imbalance given in Section 3.2 (just before Theorem 3.1), $I = p \cdot \max\{t_1, \ldots, t_p\} - \sum_{k=1}^{p} t_k$, so that

$$\begin{aligned}
&p \cdot \text{waste}(\mathcal{S}') \\
&\quad = \sum_{k=1}^{p} H_k + \text{imbalance}(\mathcal{S}') \\
&\quad = \sum_{k=1}^{p} H_k + p \cdot \max\{t_1 + H_1 + T_1, \ldots, t_p + H_p + T_p\} \\
&\qquad - \sum_{k=1}^{p} (t_k + H_k + T_k) \\
&\quad \geq p \cdot \max\{t_1, \ldots, t_p\} - \sum_{k=1}^{p} t_k - \sum_{k=1}^{p} T_k \\
&\quad = I - T.
\end{aligned}$$

□

Lemma 5.8. *Under the assumptions of the theorem, for $W, p \in \mathbb{N}$, let $\mathcal{S}'$ denote the schedule produced by an algorithm given W tasks and p processors and for an arbitrary fixed initial imbalance. Then, with $\tilde{\gamma} = \text{id} - (\text{id} + (\beta - \alpha)/(4K))^{-1}$,*

$$\max\left\{W - w,\, p \cdot \mathbf{E}\,\text{waste}(\mathcal{S}')\right\} \geq \tilde{\gamma}(W),$$

where w denotes the size of the very first chunk of $\mathcal{S}'$.

Proof. Let $\mathcal{C}$ denote the very first chunk of $\mathcal{S}'$, let T denote its processing time, and let us measure time relative to the scheduling time of $\mathcal{C}$ (which is hence 0). We first establish a lower bound on the expected imbalance incurred by $\mathcal{C}$, for which we consider two cases. In one case, there exists a processor other than the one to which $\mathcal{C}$ is assigned that becomes idle later than $h + (\alpha(w) + \beta(w))/2$. Since, under the assumptions of the theorem, $\Pr(T \leq \alpha(w)) \geq 1/K$, this gives us

$$\mathbf{E}\,\text{imbalance}(\{\mathcal{C}\}) \geq \left(\frac{\alpha(w) + \beta(w)}{2} - \alpha(w) \right) \cdot \Pr(T \leq \alpha(w)) \geq \frac{\beta(w) - \alpha(w)}{2K}.$$

In the opposite case, there certainly exists a processor other than the one to which $\mathcal{C}$ is assigned that becomes idle at or before $h + (\alpha(w) + \beta(w))/2$. However, then again, now owing to $\Pr(T \geq \beta(w)) \geq 1/K$,

$$\mathbf{E}\,\text{imbalance}(\{\mathcal{C}\}) \geq \left(\beta(w) - \frac{\alpha(w)+\beta(w)}{2} \right) \cdot \Pr(T \geq \beta(w)) \geq \frac{\beta(w)-\alpha(w)}{2K}.$$

With $\tilde{\delta}$ defined as $(\beta - \alpha)/(4K)$, it follows that in any case, the expected imbalance incurred by $\mathcal{C}$, which is just the initial imbalance of $\mathcal{S}'\backslash\{\mathcal{C}\}$, is at least $2 \cdot \tilde{\delta}(w)$. Since the expected total processing time of $\mathcal{S}'\backslash\{\mathcal{C}\}$ is just $W - w$, Lemma 5.7 hence allows us to conclude that

$$p \cdot \mathbf{E}\,\text{waste}(\mathcal{S}') \geq 2 \cdot \tilde{\delta}(w) - (W - w).$$

The lemma now follows easily. Either $W - w \geq \tilde{\gamma}(W)$, in which case we are done, or $W - w < \tilde{\gamma}(W)$, which by the identity $\tilde{\gamma} = \text{id} - (\text{id} + \tilde{\delta})^{-1} = \tilde{\delta} \circ (\text{id} + \tilde{\delta})^{-1}$ and by the monotonicity of $\tilde{\delta}$ implies $\tilde{\delta}(w) \geq \tilde{\gamma}(W)$ and hence $2 \cdot \tilde{\delta}(w) - (W - w) \geq \tilde{\gamma}(W)$. □

With Lemma 5.8 it is now easy to prove the theorem. As in that lemma, define $\tilde{\gamma} = \text{id} - (\text{id} + (\beta - \alpha)/(4K))^{-1}$, and first observe that since $\tilde{\gamma}(x) > 0$ for all $x > 0$, there must exist an integer j such that $W_{j+1} < \tilde{\gamma}^{(j)}(n)$, where W_{j+1} denotes the number of unassigned tasks after the first j scheduling operations. Let j denote the smallest such integer, and note that, since the decisions taken by a scheduling algorithm may depend on the processing times of already processed chunks, j is actually a random variable. We therefore temporarily consider a restricted probability space, where j as well as the processing times of the first $j - 1$ chunks are arbitrarily fixed. In this probability space consider the schedule $\mathcal{S}_j$ consisting of the remaining chunks. Using that the processing time of the first chunk of $\mathcal{S}_j$ is independent of the processing times of the previous chunks, Lemma 5.8, applied to $\mathcal{S}_j$, then guarantees that

$$\max\left\{W_{j+1},\, p \cdot \mathbf{E}\,\text{waste}(\mathcal{S}_j)\right\} \geq \tilde{\gamma}(W_j);$$

note that $W_{j+1} = W_j - (W_j - W_{j+1})$, where $W_j - W_{j+1}$ is just the size of the jth chunk. However, by the way j was defined, $W_{j+1} < \tilde{\gamma}^{(j)}(n)$ and $W_j \geq \tilde{\gamma}^{(j-1)}(n)$, so that we have in fact

$$p \cdot \mathbf{E}\,\text{waste}(\mathcal{S}_j) \geq \tilde{\gamma}(W_j) \geq \tilde{\gamma}^{(j)}(n).$$

By adding the overhead of the initial $j - 1$ chunks, this immediately gives us a bound for the complete schedule:

$$p \cdot \mathbf{E}\,\text{waste}(\mathcal{S}) \geq h \cdot (j - 1) + \tilde{\gamma}^{(j)}(n).$$

Now the very argument used in the proof of Theorem 5.1 (just after the proof of Lemma 5.3) can be applied to eliminate j and deduce that

$$p \cdot \mathbf{E}\,\text{waste}(\mathcal{S}) \geq h \cdot \left(\min\{\tilde{\gamma}, \text{id} - h\}^*(n) - 2\right).$$

At this point, recall that all the probabilistic assertions we have derived so far were in fact conditional on the above fixing of j and the processing times of the initial $j - 1$

chunks. However, since our last bound is independent of j, the arbitrariness of our fixing implies that the bound must hold for the complete probability space as well.

All that remains to complete the proof is now to relate $\min\{\tilde{\gamma}, \mathrm{id} - h\} = \mathrm{id} - \max\{h, (\mathrm{id}+(\beta-\alpha)/(4K))^{-1}\}$ appropriately to $\gamma = \mathrm{id}-\max\{h, (\mathrm{id}+\alpha^{-1}\circ(\beta-\alpha))^{-1}\}$. However, since, by our definition of a variance estimator, we have $\alpha \geq \mathrm{id}/A$, for some constant $A \geq 1$, Lemma 3.5 yields $\gamma^* \leq 4KA \cdot \min\{\tilde{\gamma}, \mathrm{id} - h\}^*$, and we may conclude that

$$p \cdot \mathbf{E}\,\mathrm{waste}(\mathcal{S}) = \Omega(h \cdot \gamma^*(n)/(AK)).$$

This finishes the proof of Theorem 5.2. □

5.2.2. *Specific Bounds.* In view of Theorem 5.2, obtaining lower bounds for a stochastic setting reduces to bounding from below the tails of the underlying probability distribution. In the following, this is demonstrated for two specific settings. The first is a special instance of the independent-tasks setting with the processing time of a task assumed to be normal (truncated at zero). The second is a special instance of the coupled-tasks setting, assuming a uniform distribution of the task processing times, as well as a particular coupling. Corollary 5.3 settles one of the open problems put forward in [12].

Corollary 5.3. *Let task processing times be independent, normal and with variance σ^2, and let the overhead be $h \geq 2$. Then for all $n, p \in \mathbb{N}$ with $n/p \geq \max\{h, (4\sigma)^2\}$, the schedule $\mathcal{S}$ produced by an arbitrary scheduling algorithm given n tasks and p processors satisfies*

$$\mathbf{E}\,\mathrm{waste}(\mathcal{S}) = \Omega((h \cdot \log\log_h n + \sigma\sqrt{h})/p).$$

Proof. The proof is a simple matter of combining Theorem 5.2 with well-known tail estimates for the normal distribution. Let $w \in \mathbb{N}$, and let T be the total processing time of an arbitrary fixed selection of w tasks. Then T is normal with mean w and variance $\sigma^2 w$, and by the tail estimates established in the proof of Lemma 4.6 (in fact, for our purposes here those from [10] would do equally well), there exists a constant $K \geq 1$ such that

$$\Pr(T \geq w + \sigma\sqrt{w}) \geq 1/K,$$

as well as

$$\Pr(T \leq \max\{w/2, w - \sigma\sqrt{w}\}) \geq \Pr(T \leq w - \sigma\sqrt{w}) \geq 1/K.$$

The precondition to Theorem 5.2 is hence fulfilled with

$$[\alpha, \beta]\colon\ w \mapsto [\max\{w/2, w - \sigma\sqrt{w}\}, w + \sigma\sqrt{w}],$$

and the theorem gives us

$$\mathbf{E}\,\mathrm{waste}(\mathcal{S}) = \Omega(h \cdot \gamma^*(n)/p),$$

where γ is the progress rate associated with $[\alpha, \beta]$ and h. Using Lemma 4.4 to evaluate γ^*, we have that, for $n/p \geq \max\{h, (4\sigma)^2\}$,

$$\gamma^*(n) = \Omega(\log\log_h n + \sigma/\sqrt{h}). \qquad \square$$

Corollary 5.4. *Let task processing times be coupled, and uniformly distributed in* $[T_{\min}, T_{\max}]$, *and let the overhead be* h. *Then for all* $n, p \in \mathbb{N}$ *with* $n/p \geq (3h)^2$, *and for every algorithm that produces a schedule* $\mathcal{S}$ *such that the processing times of each pair of tasks are equal if the tasks belong to the same chunk and independent otherwise, it holds that*

$$\mathbf{E}\,\mathrm{waste}(\mathcal{S}) = \Omega((h \cdot \log n)/p).$$

Proof. By assumption, the processing time T of a chunk of $\mathcal{S}$ of size w is uniformly distributed in $[T_{\min} \cdot w, T_{\max} \cdot w]$, so that for $\Delta T = (T_{\max} - T_{\min})/4$, we have $\Pr(T \leq (T_{\min} + \Delta T) \cdot w) \geq \frac{1}{4}$ as well as $\Pr(T \geq (T_{\max} - \Delta T) \cdot w) \geq \frac{1}{4}$. The preconditions to Theorem 5.2 are therefore satisfied with

$$[\alpha, \beta]\colon\ w \mapsto [(3T_{\min} + T_{\max})/4 \cdot w, (T_{\min} + 3T_{\max})/4 \cdot w],$$

so that we obtain

$$\mathbf{E}\,\mathrm{waste}(\mathcal{S}) = \Omega(h \cdot \gamma^*(n)/p),$$

where γ is the progress rate associated with $[\alpha, \beta]$ and h. Since $1 \leq (T_{\min} + 3T_{\max})/(3T_{\min} + T_{\max}) \leq 3$, Lemma 4.4 implies that for $n/p \geq (3h)^2$, $\gamma^*(n) = \Omega(\log n)$, which proves the corollary. $\square$

Acknowledgments

The author wishes to thank Torben Hagerup for introducing her to the subject, Volker Priebe for incredibly thorough proof reading and tireless praising, Susan Hert for improving the English, Pierre Fraigniaud for having been such an obliging editor, and Berthold Vöcking for comments on the Introduction.

References

[1] Bast, H. (2000), Provably Optimal Scheduling of Similar Tasks, Ph.D. thesis, Universität des Saarlandes.
[2] Bull, J. M. (1998), Feedback guided dynamic loop scheduling: algorithms and experiments, in *Proceedings, European Conference on Parallel Computing* (*EURO-PAR* '98), pp. 377–382, Lecture Notes in Computer Science 1470, Springer-Verlag, Berlin.
[3] Coffman, E. G. (1976), *Computer and Job-Shop Scheduling Theory*, Wiley, New York.
[4] Durand, M. D., Jalby, W., Kervella, L., and Montaut, T. (1996), Impact of memory contention on dynamic scheduling on NUMA multiprocessors, *IEEE Transactions on Parallel and Distributed Systems*, **11**, 1201–1214.
[5] Eager, D. L., and Subramaniam, S. (1994), Affinity scheduling of unbalanced workloads, in *Proceedings, Supercomputing* (*SC* '94), pp. 214–226.

[6] Eager, D. L., and Zahorjan, J. (1992), Adaptive Guided Self-Scheduling, Technical Report 92-01-01, Department of Computer Science and Engineering, University of Washington.

[7] Flynn, L. E., Hummel, S. F., and Schonberg, E. (1992), Factoring: a method for scheduling parallel loops, *Communications of the ACM* **35**(8), 90–101.

[8] Garey, M. R., and Johnson, D. S. (1979), *Computers and Intractability, a Guide to the Theory of NP-Completeness*, Freemann, San Francisco, California.

[9] Graham, R. L. (1966) Bounds for certain multi-processing anomalies, *Bell System Technical Journal* **45**, 1563–1581.

[10] Grimmett, G. R., and Stirzaker, D. R. (1992), *Probability and Random Processes* (2nd edn.), Oxford University Press, Oxford.

[11] Gumbel, E. J. (1954), The maxima of the mean largest value and of the range, *Annals of Mathematical Statistics* **25**, 76–84.

[12] Hagerup, T. (1996), Allocating independent tasks to parallel processors: an experimental study, in *Proceedings*, *Parallel Algorithms for Irregularly Structured Problems* (*IRREGULAR* 1996), pp. 1–33, Lecture Notes in Computer Science 1117, Springer-Verlag, Berlin.

[13] Hagerup, T. (1997), Allocating independent tasks to parallel processors: an experimental study, *Journal of Parallel and Distributed Computing* **47**, 185–197.

[14] Hartley, H. O., and David, H. A. (1954), Universal bounds for mean range and extreme observation, *Annals of Mathematical Statistics* **25**, 85–99.

[15] Hummel, S. F., Banicescu, I., Wang, C., and Wein, J. (1995), Load balancing and data locality via fractiling: an experimental study, in *Proceedings*, 3*rd Workshop on Languages, Compilers, and Run-Time Systems for Scalable Computers* (*LCR* '95), pp. 85–98, Kluwer, Dordrecht.

[16] Hummel, S. F., Kimelman, D., Schonberg, E., Tennehouse, M., and Zernik, D. (1997), Using program visualization for tuning parallel-loop scheduling, *IEEE Concurrency* **5**, 26–40.

[17] Hummel, S. F., Schmidt, J., Uma, R. N., and Wein, J. (1996), Load-sharing in heterogeneous systems via weighted factoring, in *Proceedings*, 8*th Annual ACM Symposium on Parallel Algorithms and Architectures* (*SPAA* '96), pp. 318–328.

[18] Kruskal, C. P., and Weiss, A. (1985), Allocating independent subtasks on parallel processors, *IEEE Transactions on Software Engineering* **11**, 1001–1016.

[19] Liu, J., Lam B. Y., and Saletore, V. A. (1993), Scheduling non-uniform parallel loops on distributed memory machines, in *Proceedings, Hawaii International Conference on System Sciences*, vol. 2, pp. 516–525.

[20] Liu, J., and Saletore, V. A. (1993), Self-scheduling on distributed-memory machines, in *Proceedings, Supercomputing* (*SC* '93), pp. 814–823.

[21] Liu, J., Saletore, V. A., and Lewis, T. G. (1994), Safe self scheduling—a parallel loop scheduling scheme for shared-memory multiprocessors, *International Journal of Parallel Programming* **22**(6), 589–616.

[22] Lucco, S. (1992), A dynamic scheduling method for irregular parallel programs, in *Proceedings*, *Conference on Programming Language Design and Implementation* (*PLDI* '92), pp. 200–211.

[23] Lusk, E. L., and Overbeek, R. A., (1983), Implementation of Monitors with Macros: a Programming Aid for the HEP and Other Parallel Processors, Technical Report ANL-83-97, Argonne National Laboratory, Argonne, Illinois.

[24] Markatos, E. P., and LeBlanc, T. J. (1994), Using processor affinity in loop scheduling on shared-memory multiprocessors, *IEEE Transactions on Parallel and Distributed Systems* **5**(4), 379–400.

[25] Orlando, S., and Perego, R. (1998), A comparison of implementation strategies for nonuniform data-parallel computations, *Journal of Parallel and Distributed Computing* **52**, 132–149.

[26] Orlando, S., and Perego, R. (1998), Scheduling data-parallel computations on heterogeneous and time-shared environments, in *Proceedings*, *European Conference on Parallel Computing* (*EURO-PAR* '98), pp. 356–365, Lecture Notes in Computer Science 1470, Springer-Verlag, Berlin.

[27] Petrov, V. V. (1995), *Limit Theorems of Probability Theory*, Oxford University Press, Oxford.

[28] Pinedo, M. (1995), *Scheduling*: *Theory, Algorithms, and Systems*, Prentice-Hall, Englewood Cliffs, New Jersey.

[29] Polychronopoulos, C. D., and Kuck, D. J. (1987), Guided self-scheduling: a practical scheduling scheme for parallel supercomputers, *IEEE Transactions on Computers* **36**, 1425–1439.

[30] Rudolph, D. C., and Polychronopoulos, C. D. (1989), An efficient message-passing scheduler based on guided self-scheduling, in *Proceedings International Conference on Supercomputing* (*ICS* '89), pp. 50–60.

[31] Shmoys, D. B., Wein, J., and Williamson, D. P. (1995), Scheduling parallel machines online, *SIAM Journal of Computing* **24**, 1313–1331.
[32] Smith, B. (1981), Architecture and applications of the HEP multiprocessor computer system, in *Proceedings, SPIE Symposium* (*Real Time Processing* IV), pp. 241–248.
[33] Tang, P., and Yew, P. C. (1986), Processor self-scheduling for multiple-nested parallel loops, in *Proceedings, International Conference on Parallel Processing* (*ICPP* '86), pp. 528–535.
[34] Tzen, T. H., and Ni, L. M. (1993), Trapezoid self-scheduling—a practical scheduling scheme for parallel compilers, *IEEE Transactions on Parallel and Distributed Systems* **4**(1), 87–98.
[35] Wolfe, M. (1996) *High Performance Compilers for Parallel Computing*, Addison-Wesley, Amsterdam.
[36] Yan, Y., Jin, C., and Zhang, X. (1997), Adaptively scheduling parallel loops in distributed shared-memory systems, *IEEE Transactions on Parallel and Distributed Systems* **8**(1), 70–81.

Online publication November 27, 2000