# Provably Optimal Scheduling of Similar Tasks

Dissertation
zur Erlangung des Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
der Technischen Fakultät der
Universität des Saarlandes

von

Hannah Bast

Saarbrücken
2000

| | |
|---|---|
| Tag des Kolloquiums: | 4. Februar 2000 |
| Dekan: | Prof. Dr. Wolfgang Paul |
| Berichterstatter: | Prof. Dr. Kurt Mehlhorn<br>Prof. Dr. Susanne Albers |

## Abstract

We consider the problem of processing a given number of tasks on a given number of processors as quickly as possible when the processing times of the tasks are variable and not known in advance. The tasks are assigned to the processors in *chunks* consisting of several tasks at a time, and the difficulty lies in finding the optimal tradeoff between the processors' load balance, which is favoured by having small chunks, and the total scheduling overhead, which will be the lower the fewer chunks there are. Our studies are motivated by a practical problem from high-performance computing, namely parallel-loop scheduling, for which a large variety of heuristics have been proposed in the past, but hardly any rigorous analysis has been presented to date. Our work is based on a generic approach that covers the whole spectrum of processing time irregularity. This approach does not make any assumptions about task processing times, but instead works with estimated ranges for processing times, one for each chunk size, and a measure for the overall deviation of the actual processing times from these estimates. Our analysis provides a general upper bound applicable for every conceivable setting of these parameters, together with lower bounds showing that no algorithm can do significantly better than the ones we propose. Our general result implies optimal bounds for a whole variety of specific settings, including the modelling of task processing times as independent, identically distributed random variables, which underlies most of the previously existing heuristics. Our results confirm the practicability of certain well-established techniques for parallel-loop scheduling, while, on the other hand, revealing major flaws in other approaches.

## Kurzzusammenfassung

Wir betrachten das Problem, eine gegebene Anzahl von Aufgaben (engl. *tasks*) möglichst schnell auf einer gegebenen Anzahl von Prozessoren zu bearbeiten, wenn die Bearbeitungszeiten der Aufgaben nicht im Vorhinein bekannt sind. Die Zuweisung der Aufgaben an die Prozessoren geschieht in Stücken (engl. *chunks*) von mehreren Aufgaben auf einmal, und die Schwierigkeit liegt darin, das optimale Verhältnis zu finden zwischen der Güte der Lastverteilung, die durch das Wählen kleiner Stücke begünstigt wird, und der Summe der mit den einzelnen Zuweisungsoperationen verbundenen Wartezeiten, die umso kleiner ist je weniger Stücke es gibt. Unsere Untersuchungen sind durch ein praktisches Problem aus dem Bereich des Hochgeschwindigkeitsrechnens, das sogenannte Scheduling paralleler Schleifen, motiviert, für das in der Vergangenheit zwar eine Vielzahl von Heuristiken vorgestellt wurde, es aber bisher keine vollständige mathematische Analyse gab. Unsere Arbeit basiert auf einem generischen Ansatz, der das gesamte Spektrum möglicher Unregelmäßigkeiten in den Bearbeitungszeiten der Aufgaben umfasst. Dieser Ansatz macht keinerlei Annahmen über die Bearbeitungszeiten, sondern arbeitet stattdessen mit geschätzten Bereichen für die Bearbeitungszeiten, einem für jede Stückgröße, und einem Maß für die Abweichung der tatsächlichen Bearbeitungszeiten von dieser Schätzung. Wir beweisen eine generische, für alle Werte dieser Parameter gültige obere Schranke, sowie eine dazu passende untere Schranke. Dieses sehr allgemeine Resultat impliziert Schranken für eine Vielzahl spezieller Modelle, darunter dasjenige, wonach die Bearbeitungszeiten der Aufgaben identisch verteilte, unabhängige Zufallsvariablen sind; dieses Modell liegt den meisten der oben erwähnten Heuristiken zugrunde. Teils bestätigen unsere Ergebnisse den Sinn und Zweck gewisser wohletablierter Techniken für das Scheduling paralleler Schleifen, teils weisen sie auf erhebliche Schwächen bekannter Verfahren hin.

# Preface

This thesis is about tradeoffs . . . in all respects. At first sight, it seems this work is merely concerned with achieving good load balancing with little overhead. A closer look reveals that, at a more abstract level, the goal was to make interesting theory with practical relevance. The real challenge underlying this work, however, was that of living a happy life as a computer scientist.

There is a basic truth about tradeoffs: you can easily have one extreme, but you just cannot have both. So easy to understand, yet nobody wants to understand. The scheduling problem considered in this thesis had to bow to this insight: I proved a lower bound on the sum of the two quantities in question (to demonstrate my goodwill, I also proved an upper bound).

When it comes to the balancing act between theoretical depth and practical relevance, irrationality starts. Though it is pointed out very clearly that we address a practical problem, a theorist is bound to come (and did come) arguing that our theory is too specific, and is subsumed by his work anyway. On the other hand, regardless of our indicating the obvious weaknesses of purely experimental research, a more practically inclined colleague will wonder what the fuss of proving all these theorems is about. However, given the increasing acceptance of the need for mediating between theory and practice, there is hope that these will remain exceptions.

But what of the tradeoff between the length of your publication list and the development of your human potential? It is well known that, for your scientific career, it is the results that count. An equally common though somewhat less appreciated experience is that, for what kind of human being you are, all that matters is *how* you get to these results. What kind of result is one that you achieved out of an inferiority complex? What kind of result is one that further inflates your ego? What kind of result is one for which others had to suffer? What kind of load balancing strategy is one that incurs enormous overheads? Yes, the answers are similar. Making a career as a computer scientist and at the same time becoming a better human being is a very delicate job indeed, and little we are taught in this respect. But it should be possible, shouldn't it?

## Acknowledgements

Above all, I feel very fortunate for the extraordinary freedom that I was granted during the years of my graduate studies in the working group of Kurt Mehlhorn at the Max-Planck-Institut für Informatik. Further, I would like to thank the Deutsche Forschungsgemeinschaft for supporting my research with a graduate fellowship awarded within the Graduiertenkolleg Informatik at the Universität des Saarlandes. I am greatly indebted to Volker Priebe for his fantastic and incredibly thorough proof-reading, and also for an enthusiasm about my work that easily made up for the ignorance of the rest of the world. Many thanks also to Susan Hert, who did a wonderful job in bringing my writing closer to what is commonly known as English, to Peter Müller, for having been one of the most agreeable room-mates I ever had, to Susanne Albers for having accepted the burden of co-refereeing this thesis, to Stefan Schirra for having willingly written the protocol of the defense, and to the countless others who have helped me over the past years. More than thank you, finally, to Petra-Elisa, with whom three years ago a wonderful new chapter in my life began ...

# Contents

# Chapter 1

# Introduction

This thesis deals with a particular multiprocessor scheduling problem that arises naturally in the context of high-performance parallel computing. Since classical scheduling theory does not adequately address the pecularities of this problem, numerous heuristic solutions have been proposed in the literature, all of which, however, lack a solid theoretical underpinning. We address exactly this deficit, by providing, for the first time for this problem, a combination of precise modelling, practical methods, and rigorous analysis. In Section 1.1 of this introduction, we will first describe the problem and work towards a meaningful abstraction by carefully considering alternatives and discussing related work. In Section 1.2, we will describe our new theoretical approach, outline our results, and interpret them with respect to their practical relevance. Section 1.3, finally, describes the organization of the rest of the thesis.

## 1.1 Motivation and background

Parallel computing, that is, having several processors working in concert, is one of the most promising approaches to coping with even the largest computational problems in a reasonable time. Compared to ordinary sequential computing, parallel computing, in principle, offers the possibility of an unlimited speed-up of running times, namely by a factor on the order of the number of processors employed. However, taking advantage of this huge potential is a challenging task. Given a specific problem, we must first devise an algorithmic solution that exhibits subproblems which may be solved concurrently. This issue is the subject of the theory of parallel algorithms, whose abstract machine models allow for a focussing on the inherent parallelism in a computational problem. Second, given a parallel algorithm, it must be coded in such a form that a compiler is able to recognize the parallelism contained. This issue comprises the development of parallel-programming languages as well as the automatic detection of parallelism in programs written in an ordinary, sequential style. Third, the parallelism must actually be mapped, or *scheduled*, on the target machine. 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 here is effective *loop scheduling*, which determines which iteration is executed on which processor at which time. This problem shall be the starting point for our work.

We distinguish between two types of loops with respect to the contained parallelism: *serial* loops, whose iterations must be executed one after the other, and (fully) *parallel* loops, whose iterations may be executed in any order, hence also concurrently with each other. A simple example for each of these two types is provided in Figure 1.1. Of course, loops may also

```
DO I = 1,N                    DO J = 1,M
   A[I] = A[I-1]*I               A[J] = B[J]
ENDDO                         ENDDO
```

FIGURE 1.1: A serial and a parallel loop.

be in between these two extremes in the sense that only iterations within (usually relatively small) subgroups can be executed independently from each other—so-called *do across* loops. The exploitation of such extremely fine-grained parallelism will not be our concern here; for an overview of techniques, we refer the reader to (Polychronopoulos, 1988) or (Wolfe, 1996). For compute-intensive applications, the richest source of parallelism is (often deeply) nested loops, containing a mixture of both parallel and serial loops. In view of (automatable) code-restructuring techniques such as *loop interchanging* or *loop coalescing* (Polychronopoulos, 1987; Polychronopoulos, 1988; Wolfe, 1996), we may assume that such a loop nest is given in the "normal form" of one or several fully parallel loops nested inside a serial loop. A simple example (modelling a successive over-relaxation procedure) is given in Figure 1.2. Given such code, our

```
DO I = 1, MAXITERATIONS
   DO J = 1, N
      A[J] = UPDATE(A[J])
   ENDDO
ENDDO
```

FIGURE 1.2: A loop nest in normal form.

task becomes that of scheduling a sequence of fully parallel loops, where all the iterations of one loop have to be completed before the execution of the next loop can begin. The challenge therefore lies in the scheduling of a single parallel loop so as to minimize the completion time of the last iteration, a quantity known as the *length* or *makespan* of the schedule. We also note that, since a parallel loop might be iterated a large number of times, even small deviations in the makespan can accumulate as considerable amounts of running time; achieving optimal or close-to-optimal solutions 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. We next survey this theory with respect to its applicability to the parallel-loop scheduling problem, as described above. In particular, the survey will point out the key features of this scheduling problem, which will be the base for our abstract problem formulation. In the context of the parallel-loop scheduling problem, a task will always refer to the execution of one iteration; task processing times then correspond to execution times of iterations.

### 1.1.1 Static scheduling

In what is known as *static scheduling*, the tasks' processing times are known in advance, so that an optimal schedule can be computed before the computation starts. Let us 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 applications like parallel-loop scheduling, where the processing time of a single task (an iteration) is small compared to 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 (Coffman, 1976; Garey and Johnson, 1979). For many practical applications, however, sufficiently accurate approximations do equally well. So, for instance, in parallel-loop scheduling it would clearly be satisfactory to divide the loop into blocks or chunks of consecutive iterations, one chunk for each processor, such that the total processing times of the chunks differ at most by the execution time of a single iteration. This scheme, known as *block scheduling* or *static chunking*, can obviously be implemented in linear time.

Of course, it is unrealistic that a compiler will be given iteration execution times as part of its input. Provided that execution times are deterministic, however, they can usually be obtained by general-purpose performance-prediction tools like those presented in (Fahringer and Zima, 1993) or (Sarkar, 1989). A specific scheme that addresses the issue of execution-time profiling for the particular case of a parallel loop repeated within a serial loop was presented in Bull (1998).

A more serious problem for static scheduling schemes is when the tasks to be scheduled are *irregular* in the sense that their processing times vary widely and in an unpredictable manner. In that case, performance-prediction tools may at best serve to provide estimates, from which the actual processing times may deviate considerably. Processing times may vary for two types of reasons. First, there might be *algorithmic irregularity*, which in the case of loops refers to different (machine) code being executed for different iterations. For example, an expression might be evaluated whose complexity depends somehow on the loop index, or the loop body might contain an if-statement or yet another (serial) loop with possibly variable bounds. Secondly, each computation has some *systemic irregularity* caused by the various interactions between application program, system software, and hardware. A characteristic example here is a memory access, whose latency may vary by orders of magnitudes depending on where in the memory hierarchy the data is actually read or written. While algorithmic irregularity can sometimes be figured out by the compiler or might be low altogether, systemic irregularity is generally both considerable and very hard to predict. Indeed, this effect can be observed already in an ordinary sequential computer, and, not surprisingly, gets multiplied by having several such machines working in concert. Experimental evidence for this is provided in (Hummel *et al.*, 1995; Durand *et al.*, 1996; Hummel *et al.*, 1997). We notice that in view of considerable irregularity, due to whatever reason, static scheduling becomes inapproriate, as it cannot avoid that some processors finish long before others, thus wasting their time for the rest of the computation.

### 1.1.2 Online scheduling

In appreciation of the fact that parts or all of the relevant information might not be available to an algorithm, much attention in scheduling theory has been devoted to the study of so-called *online* or *dynamic* problems. The optimal online algorithm for scheduling a given number of tasks with unknown processing times appears to be the straightforward *list* scheduling method (Graham, 1966). Here tasks are given in a list, with an arbitrary but fixed order, and at runtime, whenever a processor is (in the beginning) or becomes idle, the first task from the list is removed and scheduled to that processor. Clearly, this produces a schedule where the finishing times of the processors differ by at most the execution time of a single iteration—a result that we have deemed satisfactory above. In fact, the first dynamic parallel-loop scheduling schemes were trickily optimized implementations of list scheduling on shared-memory multiprocessor machines (Smith, 1981; Lusk and Overbeek, 1983; Tang *et al.*, 1985; Tang and Yew, 1987), exploiting the feature that processors may "schedule themselves" the tasks from the list, thus doing away with a (notoriously inefficient) central scheduling unit. To emphasize this, in the context of parallel-loop scheduling, it is therefore rather spoken of *self-scheduling* instead of list scheduling.

As it turns out, however, the straighforward list or self-scheduling scheme often performs rather poorly in practice, for the following reason. The problem with every dynamic approach 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 list 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 list 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 a sense, this fully dynamic approach is therefore just the extreme opposite to static scheduling, which minimizes overheads at the risk of a large load imbalance. It seems that in order to achieve both a small overhead and a small load imbalance, a more satisfactory scheme should rather schedule *chunks* of several tasks at a time.

So what is the optimal algorithm for scheduling tasks in chunks when there is a constant overhead for each scheduling operation? The common approach to answer such questions in online theory is *competitive analysis* (Sleator and Tarjan, 1985; Karlin *et al.*, 1988), which compares the results produced by an online algorithm to the optimal result that could have been produced if all the relevant information had been available beforehand. The quality of an online algorithm is expressed in its *competitive ratio*, that quantifies by how much, in the worst case, the online solution deviates from the optimal offline solution. For a more detailed introduction into this field, we refer the reader to the recent survey of (Sgall, 1998). As an immediate consequence of the analysis given by Graham (1966), the competitive ratio of list scheduling is bounded by $2 - 1/p$, where $p$ is the number of processors, and according to a result by (Shmoys *et al.*, 1995), no deterministic algorithm can do better than list scheduling in this respect (and even randomization does not help much). But this implies that the makespan achieved by list scheduling can be almost twice as large as the optimum, which is a very poor performance indeed for a loop scheduling scheme. This seems to contradict our result from above that list scheduling would indeed be a perfect scheme if scheduling overheads could be ignored. The reason for this negative result is that competitiveness is measured with respect to the *worst-case* input. For our scheduling problem, this worst case is when all iterations have the same short execution time except for one iteration which takes much longer, irrespective of whether overheads are taken into account or not.

As a consequence, even when there is overhead, list scheduling comes out as the optimal algorithm with respect to competitiveness. Of course, worst cases like the above are extremely unlikely to occur for realistic inputs, and it becomes clear that a meaningful analysis of the tradeoff between load balance and scheduling overhead must be based on some notion of *similarity* between tasks. Then larger chunks, which help to keep the overhead low, will also tend to have larger processing times, which complicates load balancing.

### 1.1.3 Stochastic scheduling

The traditional way to analyze a computational problem theoretically under the assumption of somehow "realistic" inputs 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* (Pinedo, 1995). Unfortunately, average-case analysis is usually much harder than worst-case analysis, and quickly becomes infeasible when dealing with dynamic problems. Indeed, compared to 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 seems to be known (Kruskal and Weiss, 1985). We will next describe this result, 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 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 results were promising: the chunk size can be chosen such that the expected makespan is within a factor of $1 + \epsilon$ of the optimum, where $\epsilon \to 0$ as the number of tasks goes to infinity. However, this result was proven under a number of very idealized assumptions; in fact, the authors themselves explained that their results should be viewed as "lower bounds for real machines". First, their analysis was asymptotic, so that it remained unclear when $\epsilon$ 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 for the best ad-hoc choice of chunk size, the schedule produced is never even close to optimal. And fourth, a suitable—not to mention optimal—chunk size is very hard to determine, and no intuitive default setting exists. Roughly speaking, the bottom line is that fixed-size chunking is a useful but "quick-and-dirty" heuristic. In fact, already Kruskal and Weiss noted that an optimal scheme should rather have "the chunk size decrease as the scheduling process evolves". This is quite natural, since smaller chunks are required only towards the end, in order to achieve even finishing times, whereas earlier chunks may be larger thus helping to keep the scheduling overhead small. We will come back to fixed-size chunking in Section 6.1.

### 1.1.4 Heuristics for parallel-loop scheduling

Having surveyed the field of scheduling theory (without finding any satisfactory approaches), let us next turn our attention to the large body of work that explicitly addresses the scheduling of parallel loops. As we have seen, the basic challenge in designing a parallel-loop scheduling scheme is to determine chunk sizes so as to achieve an optimal tradeoff between load balance and scheduling overhead. For this task, a large variety of heuristics have been proposed to date (Polychronopoulos and Kuck, 1987; Tzen and Ni, 1991; Flynn *et al.*, 1992; Lucco, 1992;

Liu *et al.*, 1994; Hagerup, 1997), each of which will be described and discussed in detail in Section 7. Let us mention here that most of these heuristics were actually implemented, and it seems that at least two of them, the *guided self-scheduling* of (Polychronopoulos and Kuck, 1987) and a variant of the *factoring* strategy by (Flynn *et al.*, 1992) have been embodied in a variety of serious (that is, not specially created) applications. On top of these basic strategies, numerous further schemes were constructed, addressing issues such as affinity and data locality (Markatos and LeBlanc, 1994; Eager and Subramaniam, 1994; Hummel *et al.*, 1995; Orlando and Perego, 1998a), self-adaptiveness (Eager and Zahorjan, 1992; Yan *et al.*, 1997), distributed memory (Rudolph and Polychronopoulos, 1989; Liu and Saletore, 1993; Liu *et al.*, 1993), and heterogeneous, time-shared environments (Hummel *et al.*, 1996; Orlando and Perego, 1998b). For our purposes here, we note that for all of these, a clever heuristic for determining chunk sizes is an essential component.

It is conspicious that none of the numerous schemes that have been presented for parallel-loop scheduling to date are supported by any rigorous analysis. This is unsatifactory in several respects. First, and most obviously, there is a risk that such schemes perform poorly even under circumstances that conform well to the underlying model. Indeed, we will show this to be the case for several of the previous schemes. But even if one is willing to accept sufficient amounts of empirical data as a kind of guarantee, it still remains obscure in which way which parts of the heuristics actually effected the claimed performance. So, for example, Hagerup (1997) 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". Similarly, the FAC scheme of Flynn *et al.* (1992) was designed on the basis of a very intricate heuristic but tends to perform rather poorly compared to its "quick-and-dirty" variant FAC2, which does away with all the intricacy. 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

Let us briefly summarize our findings from the last section. On the one hand, we have pointed out two characteristic features of parallel-loop scheduling: first, that task processing times, while similar, vary in an unpredictable manner, and second, that each scheduling operation has a significant overhead, which suggests the scheduling of chunks of tasks, instead of one task at a time. We have surveyed the wide field of scheduling theory with regard to these features, finding only a single, specialized result. On the other hand, an abundance of heuristics for parallel-loop scheduling, which all deal with the tradeoff between load balance and scheduling overhead, exist. All of these, however, lack a solid theoretical underpinning. Our work aims at closing this gap between theory and practice.

### 1.2.1 Generic approach

Let us briefly restate our problem in general terms. We are given $n$ tasks with previously unknown processing times, to be scheduled on $p$ processors. The tasks are scheduled on processors in groups, called *chunks*, at the price of a certain overhead per chunk. Here, as well as later, we will work with the assumption of a fixed overhead $h$. As will be shown in Chapter 3, this (of course unrealistic) restriction is not for technical neccessity 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 to 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.

As we have learned from the previous section, a meaningful analysis of the tradeoff between load balance and scheduling overhead, and hence of the wasted time, must be built on the assumption of somehow similar processing times. This is of course a fairly general concept, which could be modelled in various ways. One example is what we call the *independent-tasks setting*, which underlies the theoretical investigations of (Kruskal and Weiss, 1985) as well as most existing heuristic schemes; here the task processing times are independent, identically distributed random variables. Note that, since the independence assumption implies a very sharp concentration of chunk processing times, this is a particularly well-behaved setting. Hagerup (1997) 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 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 considered in (Liu *et al.*, 1994).

But even if we could prove tight bounds on the makespan for each of these particular scenarios, we would gain only partial insight into the role that irregularity plays for our scheduling problem. For the stochastic modellings, this problem gets amplified by the fact 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 distribution function can express that some task takes somewhere between ten and twenty 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 fifteen 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. This is in fact a well-known phenomenon in the field of stochastic scheduling. For example, the basic problem of scheduling $n$ tasks on two processors with minimal makespan is NP-hard for arbitrary given processing times, but becomes polynomially solvable when task processing times are assumed to be exponentially distributed with arbitrary given parameters, and the goal is to minimize the makespan in expectation (Pinedo, 1995).

While it is certainly instructive to study any one of the aforementioned particular settings, we are more ambitious than that. Namely, our goal is to parameterize the entire spectrum of irregularity, with the static and the online setting at its respective endpoints, and for every parameter setting determine the corresponding optimal algorithm together with a performance guarantee. Such a generic result would, in particular, settle our scheduling problem for a variety of specific settings, but, what is more, also provide comprehensive insight into how irregularity affects scheduling efficiency in general.

To realize the mentioned parameterization, we introduce two concepts: a *variance estimator*, which estimates the variability of chunk processing times and is provided as part of the input, and a *deviation*, which measures the deviation of the actual processing times from their estimated behaviour and is not known until after the event. The variance estimator is specified by two functions $\alpha, \beta : \mathbb{R}^+ \rightarrow \mathbb{R}^+$, 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 $\varepsilon$ that measures the average distance of an actual chunk processing time to its respective range $[\alpha(w), \beta(w)]$. Naturally, $\varepsilon$ 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 $\varepsilon$ will be. This approach will be described and explained in more detail in Chapter 2.

### 1.2.2 Theoretical results

Our main result quantifies for each setting of the parameters $n$, $p$, $h$, $\alpha$, $\beta$ and $\varepsilon$, 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 $\varepsilon$. To establish this result, we will 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 $\gamma^*(n/p) = \min\{\, i : \gamma^{(i)}(n/p) \le 0 \,\}$, and $\gamma$ is approximately $\mathrm{id} - \alpha \circ \beta^{-1}$, where id denotes the identity function. Unfortunately, the underlying intuition cannot be easily conveyed in a few lines; indeed, the whole of Section 2.3 will be dedicated to this task. Instead we provide the following table 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.

| | $\left[\, w - \sqrt{w}\,,\, w + \sqrt{w} \,\right]$ | $\left[\, w/2\,,\, 2w \,\right]$ | $\left[\, w/2\,,\, w \log w \,\right]$ | $\left[\, w/2\,,\, w^2 \,\right]$ |
|---|---|---|---|---|
| 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 | $H \cdot N$ | $H \cdot N$ | $H \cdot N$ | $H \cdot N$ |

TABLE 1.1: The order of the optimal wasted time compared to that of fixed-size chunking, static chunking, and self-scheduling for a variety of estimated ranges, where $H = h + \varepsilon$ and $N = n/p$.

Equipped with the magic formula for OPT, it becomes easy to prove upper bounds for a whole variety of task processing time 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 $\sigma$ 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 $O(h \cdot \log(n/p))$.

Despite a careful analysis that cares about constant factors and small input values, the generality of our main result brings along a certain inaccuracy in its bounds, which complicates a proper parameter tuning in practice. Also, not all instances of the generic balancing strategy are as simple as may be desirable in practical applications. In recognition of these facts, we investigate in further depth two classes of particularly simple scheduling algorithms. The first class is the *fixed-size* schemes, which divide the given set of tasks into chunks of equal size. The second class is the so-called *geometric* schemes, which have the property that chunk sizes decrease by a fixed factor, that is, the sizes of successively allocated chunks form a geometric sequence. For both classes we give a general *and* exact analysis based on the parameters $\alpha$, $\beta$, and $\varepsilon$ of our generic approach; as for our main result, this will imply corresponding bounds for each of the specific independent-tasks, bounded-tasks, and coupled-tasks setting. As indicated by Table 1.2, our results will convincingly demonstrate that comitting to a fixed chunk size is inevitably coupled with a significant performance loss, while the geometric schemes can achieve wasted times surprisingly close to the theoretical optimum. We also provide evidence that no other comparably simple scheme shares this property. Moreover, by exploiting the special structure of geometric schedules, we are able to remove the constant factor entailed by our general analysis of the balancing scheme, which leads to a very strong performance guarantee. We conclude that the geometric schemes achieve a hardly surpassable combination of simplicity, efficiency, and reliability, which makes them prime candidates for implementation in a real system.

| | independent tasks | bounded tasks | coupled tasks |
|---|---|---|---|
| FIX | $\sqrt{h \cdot n/p}$ | $\sqrt{h \cdot n/p}$ | $(\, h \cdot n\,)^{2/3} \;/\; \sqrt[3]{p}$ |
| GEO | $h \cdot \log(n/p)$ | $h \cdot \log(n/p)$ | $h \cdot \sqrt{n \cdot \log(n/p)}$ |
| BAL | $h \cdot \log\log(n/p)$ | $h \cdot \log(n/p)$ | $h \cdot \sqrt{n}$ |

TABLE 1.2: Bounds on the order of the (expected) wasted times in various settings for the respectively optimal fixed-size, geometric, and balancing scheme.

The final part of this thesis is concerned with the mathematical analysis of previous heuristics for scheduling tasks in chunks of decreasing sizes. As we have mentioned earlier, all of these had so far been assessed merely on an experimental basis. Rigorous analysis existed only for a fixed-size scheme, where, however, it was limited to the particular independent-tasks setting. For all except one of the previous heuristics, we are able to prove either an upper or a lower bound on the achieved wasted time, assuming the setting from which it was actually derived. In some cases our results nicely match previous empirical findings, while for other schemes, our analysis reveals major flaws, which apparently had not become evident by the respective experiments. So, for example, we prove that the FAC2 scheme proposed by (Flynn *et al.*, 1992), and used in a variety of applications (Banicescu and Hummel, 1995; Banicescu, 1996) achieves an expected wasted time of $h \cdot \log_2(n/p) + O(\sigma^2 \cdot p^{5/3})$ in the independent-tasks setting with variance $\sigma^2$. This supports the claim of its inventors that FAC2 is indeed a sound scheduling scheme for the independent-tasks setting. On the other hand, we show that the original factoring scheme (FAC) of Flynn and Hummel and the tapering strategy (Taper) due to (Lucco, 1992), both of which are rather intricate, as well as the guided self-scheduling scheme (GSS) proposed by (Polychronopoulos and Kuck, 1987), incur an expected wasted time of $\Omega(\sqrt{n/p})$ in the independent-tasks setting. According to Table 1.2, both schemes are therefore asymptotically no better than what can already be achieved by a simple fixed-size scheme.

### 1.2.3 Practical significance

Apart from their contributing to scheduling theory in general, our results also have a number of interesting implications relevant to the problem of scheduling parallel loops in practice, which indeed was the starting point for our studies.

One consequence from our analysis is that by underestimating the irregularity, that is, by choosing too narrow estimated ranges, the average deviation $\varepsilon$ may grow as large as $n/p$, resulting in disastrous performance. On the other hand, the first row of Table 1.1 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 realized in most previous work. In the terminology of (Hagerup, 1997), who classified dynamic chunk scheduling strategies on a scale ranging from *timid* (aiming primarily at a small load imbalance) to *bold* (aiming primarily at a small number of chunks), this guideline could be casually formulated as not to be too bold in dynamic loop scheduling.

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. We show that in order to achieve sublogarithmic wasted times, a strategy must consider the processing times of already completed chunks, which adds a significant complication to the implementation.

Another practically useful 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 evidence for its meaning was given. In previous theoretical work, only (Lucco, 1992) and (Liu *et al.*, 1994) 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.

The first is our idea of *modelling variability* of a real quantity by estimated ranges, that may be used for solving the problem, together with a deviation that is not known until the problem has been solved. This approach is an alternative to the more common probabilistic approach, where the quantity in question is modelled as a random variable with known characteristics. In the context of this work, the deterministic approach turned 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 the key to solving a problem that in previous work, based on probabilistic assumptions, appeared to be mathematically intractable. We would expect a similar approach to be helpful also for the solution of other problems involving quantities that vary in an unpredictable but somehow limited manner. In fact, a loosely related idea underlies the work of (Kleinberg *et al.*, 1997); they prove an average-case bound by replacing the random input, namely a bandwidth having either a low or a high value, by a fixed value, the *effective* bandwidth, and plugging this into an algorithm originally designed for deterministic inputs.

A second contribution that we expect to be of more general interest is our *master theorem for the* $^*$ *operator*. For a function $\gamma : \mathbb{R} \to \mathbb{R}$, the $^*$ operator "counts" the number of iterations of $\gamma$ required to get from some $x$ to some $y$; formally, $\gamma^*(x, y) = \min\{ i \in \mathbb{N} : \gamma^{(i)}(x) \leq y \}$. Just as in our work, deriving closed formulas for $\gamma^*(x, y)$ for a given function $\gamma$ frequently occurs as a subtask in the analysis 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 integrals of the functions $z \mapsto 1/(z - \gamma(z))$ and $z \mapsto 1/(\gamma^{-1}(z) - z)$.

## 1.3 Overview

On the level of chapters, the thesis is organized as follows. The next chapter will set the framework for our theoretical investigations. In particular, we will make precise the concepts of our generic approach and carefully explain the intution behind them.

Following that, Chapter 3 is dedicated to the proof of our generic upper bound, which we formulate in what we call our *Main Theorem*. We will 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 will be described and explained.

Chapter 4 is devoted to specific bounds for our scheduling problem. We will 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 chapter will also provide valuable intuition on the mathematical relationship described by the OPT formula decribed earlier. In particular, this chapter will present our *master theorem for the* $*$ *operator.*

Chapter 5 will be concerned with various lower bounds. We will first show that no algorithm can do significantly better than what is stated in our Main Theorem. Then we will extend this result to random processing times, and, in particular, prove a very specific, strong lower bound for the independent-tasks setting.

Chapter 6 will be concerned with simple scheduling strategies, and investigate in depth the class of fixed-size and geometric scheduling strategies, mentioned earlier.

Chapter 7 deals with previously existing heuristics for our scheduling problem.

We will close with some conclusions and pointers to directions for future research in Chapter 8, which is subsequently translated into Deutsch.

# Chapter 2

# Framework

This chapter sets the formal framework for our scheduling problem. Though parallel-loop scheduling is our primary motivation, we formulate the problem in somewhat more general terms, as is usually done in scheduling theory. This is both to emphasize the theoretical rigor of our work, as well as to stress our belief in a wider applicability of our theory. In Section 2.1, we first restate the scenario of the scheduling problem and give some basic definitions. Section 2.2 will provide 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

Our scheduling scenario is as follows. 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 an *overhead* time that is independent of the number of tasks in the chunk, plus the *processing time* of the chunk, which is just the sum of the processing times of the contained tasks. For the sake of clarity, all our results will be stated for a fixed overhead $h$ per chunk; as we will see in Chapter 3, however, the results can be easily extended to variable overheads. The processing time of a chunk is not known in advance, so at any time all that is known about a scheduled chunk is whether it is completed yet 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 will say that a chunk is *scheduled* (synonymously: *assigned*, *allocated*) by an algorithm, and we will 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 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 will be 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 will often denote schedules by a collection of tasks, and, in particular, write $\mathcal{C} \in \mathcal{S}$ to denote that the chunk $\mathcal{C}$ belongs to the partitioning of the schedule $\mathcal{S}$. The following definition names the characteristic properties of a schedule, which, afterwards, Figure 2.1 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 $k$th processor, by $T_k$ their total processing time, and by $t_k^{\text{fin}}$ the finishing time of the last such chunk, for $k = 1, \ldots, p$. Then define

$$\begin{aligned}
\text{chunks}(\mathcal{S}) &= \sum\nolimits_{k=1}^{p} c_k, \\
\text{makespan}(\mathcal{S}) &= \max\{\, t_1^{\text{fin}}, \ldots, t_p^{\text{fin}} \,\}, \\
\text{imbalance}(\mathcal{S}) &= \sum\nolimits_{k=1}^{p} (\text{makespan}(\mathcal{S}) - t_k^{\text{fin}}), \\
\text{idle}(\mathcal{S}) &= \sum\nolimits_{k=1}^{p} (\text{makespan}(\mathcal{S}) - T_k - h \cdot c_k), \\
\text{waste}(\mathcal{S}) &= (\, h \cdot \text{chunks}(\mathcal{S}) + \text{idle}(\mathcal{S}) \,)/p.
\end{aligned}$$

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

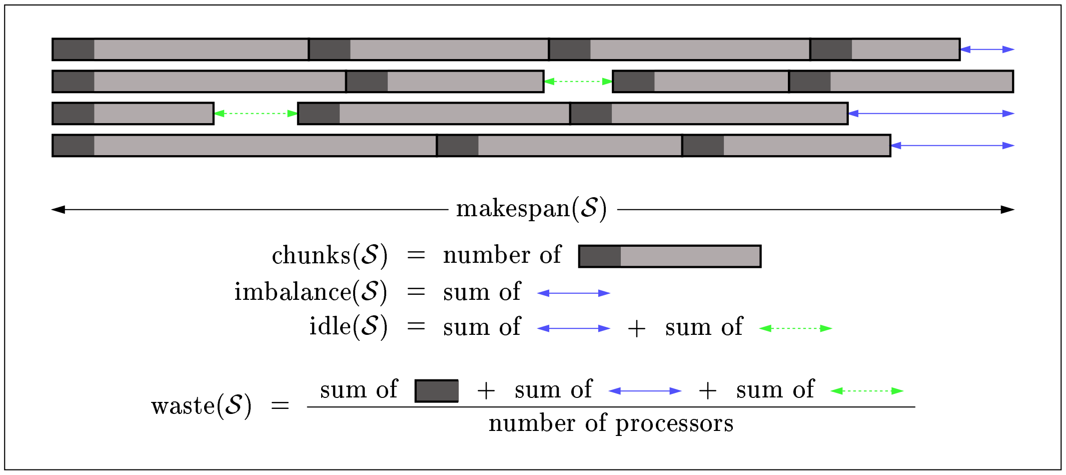

makespan($\mathcal{S}$)
chunks($\mathcal{S}$) = number of
imbalance($\mathcal{S}$) = sum of
idle($\mathcal{S}$) = sum of + sum of
waste($\mathcal{S}$) = sum of + sum of + sum of / number of processors

FIGURE 2.1: A schedule $\mathcal{S}$ on four processors and some associated quantities.

From the definition above, or more easily from Figure 2.1, 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

As outlined in the introduction, we aim at a parameterization of the entire spectrum of processing time irregularity together with an analysis that yields, for every setting of these parameters, bounds on the then best possible performance and a corresponding optimal algorithm. This section provides the basic ingredients for our generic cake: 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 characterises the optimal "pace" at which an algorithm working with a specific variance estimator should proceed. We will 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 will be used to denote the identity function $x \mapsto x$.

**Definition:** For continuous and strictly increasing functions $\alpha, \beta : \mathbb{R}^+ \to \mathbb{R}^+$ 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

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

is called a *variance estimator*. The function $\beta - \alpha$ will be referred to as the *width* of $[\,\alpha, \beta\,]$, and we will 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.

Let us 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 $\alpha$ and $\beta$ 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 for the value of a variance estimator as a decision base, 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 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,

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

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,

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

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

Let us 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 late chunk to finish; 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, $f \circ g : x \mapsto f(g(x))$. The inverse of a function $f : \mathbb{R}^+ \to \mathbb{R}^+$ that is strictly increasing and unbounded (but not necessarily surjective), is defined as $f^{-1} : y \mapsto \inf\{ x \geq 0 : f(x) \geq y \}$. If $f$ is a bijection, this is just the usual inverse of $f$. If $f$ is a bijection between $\mathbb{R}^+$ and $(y_0, \infty)$, for some $y_0 > 0$, then $f^{-1}(y) = 0$, for $y \leq y_0$. The $\circ$ as well as the $^{-1}$ notation will be be used extensively throughout the paper.

**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

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

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 2.2 below.

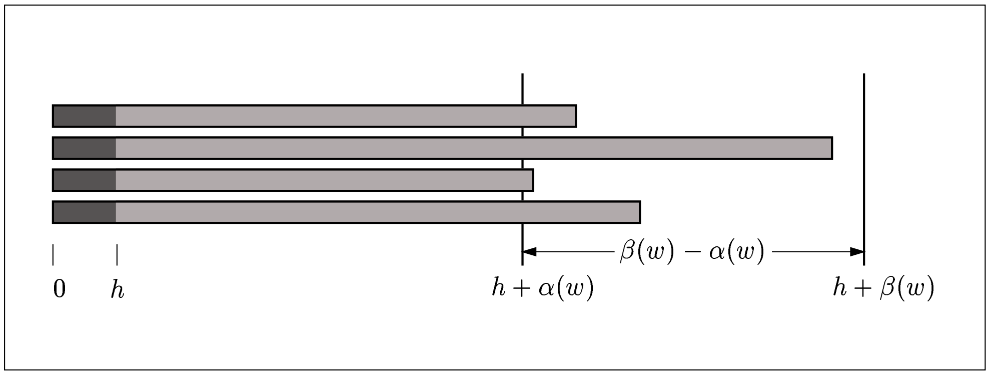


FIGURE 2.2: The imbalance of the initial chunks is at most $p \cdot (\beta(w) - \alpha(w))$.

Now, for the sake of simplicity, let us assume that $\alpha$ 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 latest, 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 $\delta$ for $A \circ (\beta - \alpha) = \alpha^{-1} \circ (\beta - \alpha)$, like in the definition of progress rate above, $w$ should hence satisfy

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

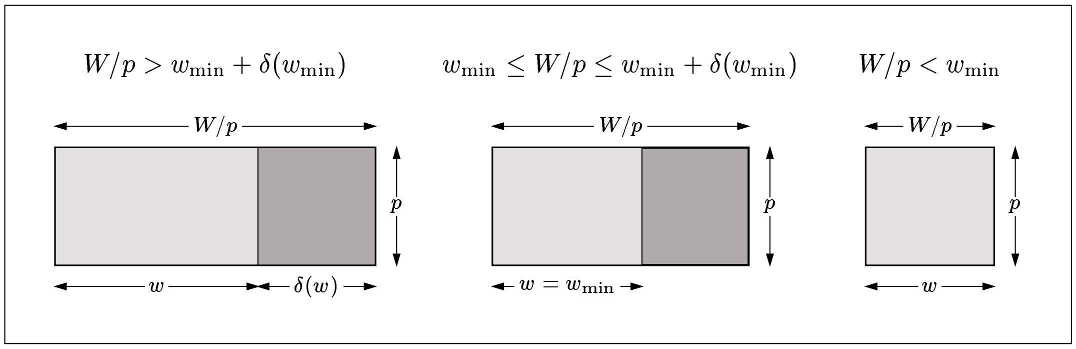
where $n$ is the total number of tasks. Under the assumption that $\mathrm{id} + \delta$ is a bijection of $\mathbb{R}^+$ (which it indeed is if $\lim_{w \to 0} \beta(w) = 0$) this equation has a unique solution

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

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 manner. But for the purpose of our providing intuition here, let us 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, let us 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

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

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 2.3 below.



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

FIGURE 2.3: $w = \min\Big\{\, W/p,\ \max\{\, w_{\min}, (\mathrm{id} + \delta)^{-1}(W/p)\,\}\,\Big\}$.

Let us 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

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

According to the definition of progress rate, this is just $\gamma_{w_{\min}}(W/p)$. Note that, in Figure 2.3 above, $\gamma_{w_{\min}}(W/p)$ is just the width of the dark grey rectangle(s). Clearly, the larger $\delta$ is, the closer $\gamma_{w_{\min}}$ is to the identity function, which corresponds to a (literally) small progress made by a single round. Table 2.1 below gives a feeling for how the progress rate is related to $\delta$, where, for simplicity, it is assumed that $w_{\min} = 1$.

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

TABLE 2.1: Examples of variance estimators and their associated progress rate.

It is now easy to see why it is natural to express bounds on the wasted time in terms of the progress rate. Namely, using the heuristic described above, the whole scheduling process can be illustrated as

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

The number of rounds is just $\gamma^{*}_{w_{\min}}(n/p)$, where for an arbitrary function $f : \mathbb{R} \to \mathbb{R}$, $f^*$ is defined as

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

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

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

Note that for the variance estimators from Table 2.1 above, the function $\gamma_1^*$ is approximately $\log\log$, $\log$, $\log^2$, and $\sqrt{\phantom{x}}$, 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 thesis), 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, let us assume, again for simplicity, that only the very last chunk—let us call it $\mathcal{C}_1$—is late, while the last chunks of the other processors—we call them $\mathcal{C}_2, \dots, \mathcal{C}_p$—are all early. In this seemingly worst case the deviations increase the imbalance by exactly $(p-1) \cdot \mathrm{late}_\beta(\mathcal{C}_1) + \mathrm{early}_\alpha(\mathcal{C}_2) + \cdots + \mathrm{early}_\alpha(\mathcal{C}_p)$. 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

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

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

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

This is exactly the bound proven in the next chapter.

# Chapter 3

# Generic upper bound

This chapter is devoted to the proof of our main theorem, which, using the formalism introduced in the previous chapter, provides a generic upper bound on the wasted time that covers the entire spectrum of processing-time irregularity. Let us first state the theorem, and then make a few remarks.

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

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

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

**Remark:** The conditions imposed on $\alpha$ and $\beta$ 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 thesis have these (for a variance estimator natural) properties.

**Remark:** In Chapter 5, we will prove a lower bound showing that no algorithm can do better than what is stated in the theorem above. This lower bound will imply that the above bound is optimal for $w_{\min} = \lceil \alpha^{-1}(h+\varepsilon) \rceil$. To verify this, 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 for $n/p \geq \beta(w_{\min})$,

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

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

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

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

**Remark:** At this point, let us 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 $h$ reinterpreted 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.

**Remark:** A final remark is concerned with the somewhat peculiar role of $\varepsilon$ 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, different algorithms might incur different values of $\varepsilon$. It should be clear that this is an inherent feature of our scheduling problem. Since our algorithms cannot find out in advance which tasks are going to take long and which short, 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 chapter though, this effect disappears when considering concrete settings that make somehow "symmetric" assumptions on the task's processing times.

The remainder of this chapter is organized as follows. Section 3.1 will first establish a number of abstract properties of the * operator, which will be used on various occasions in the analysis. In Section 3.2, we will 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 will provide a description of the generic balancing (BAL) strategy, parameterized by $[\alpha, \beta]$, together with a complete analysis. The final 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 us an exceptional effort (we could not resist mentioning) 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, let us first restate our definition of the $^*$ operator.

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

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

**Remark:** In this thesis, we will apply the $^*$ operator only to functions such that the value assigned above is finite for all $x$.

**Lemma 3.1.** Let $\gamma, \tilde{\gamma} : \mathbb{R} \to \mathbb{R}$ such that $\gamma$ 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 $\gamma : \mathbb{R} \to \mathbb{R}$ be increasing. Then for all $x, y > 0$, and for all $i \in \mathbb{N}_0$,

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

**Proof:** For $i' = \gamma^*(y) > 0$, $\gamma^{(i'-1)}(y) > 0$, hence by the assumption on $y$, and because $\gamma$ 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 $\gamma : \mathbb{R} \to \mathbb{R}$ 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 $\gamma$ decreases its argument by at least $M$, $i \le \lceil (x-y)/M \rceil$. □

**Lemma 3.4.** For increasing $\delta : \mathbb{R}^+ \to \mathbb{R}^+$ 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 $\delta$ 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 $\delta : \mathbb{R}^+ \to \mathbb{R}^+$ 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 would like to 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 $\gamma$ 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

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

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, $\gamma$ 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 $\delta : \mathbb{R}^+ \to \mathbb{R}^+$ 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 several times use of the equivalence $\gamma^*(x) = 1 \iff 0 < x \le M \iff \tilde{\gamma}^*(x) = 1$, which, in particular, 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. As we will see in Section 7, all but one of the many previously existing heuristics are of the fixed-partition type. For our purposes, it will be useful to think of a particular fixed-partition algorithm as being defined by a function $\varrho : \mathbb{R}^+ \to \mathbb{N}$ 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 will denote an algorithm defined in this way by $\mathrm{FP}(\varrho)$. Note that, naturally, $\mathrm{FP}(\varrho)$ does never insert 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 $(1 - 1/p)^p \ge 1/4$. A reasonable fixed-partition algorithm is therefore bound to have a number of scheduling operations that is logarithmic in $n/p$, the number of tasks per processor.

In contrast to this, variance estimators of sublinear width have a progress rate $\gamma$ with $\gamma(x)/x = o(1)$, in which case the bound claimed in the Main Theorem becomes sublogarithmic in $n/p$. However, as demonstrated by the following theorem, the class of fixed-partition algorithms is 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 $t_1, \ldots, t_p$. The quantity

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

will be referred to as the *initial imbalance* of the respective schedule.

**Theorem 3.1.** Let task processing times be arbitrary, and let the overhead be $h \ge 1$. Let $[\,\alpha, \beta\,]$ be a variance estimator, and let $A \ge 1$ with $\alpha \ge \mathrm{id}/A$. Then for all $w_{\min} \in \mathbb{N}$, $w_{\min} \ge h$, 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

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

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

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

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 $\beta-\alpha \ge \text{id}/D$ for some $D \ge 1$, $\tilde{\delta} = 6DA\cdot(\,\beta-\alpha\,)$ is easily seen to fulfill the condition $\tilde{\delta} \ge 6A\cdot\max\{\,\beta-\alpha,\text{id}\,\}$. 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 $\tilde{\gamma} = \text{id} - \max\{\,w_{\min}, \lfloor(\text{id}+\tilde{\delta})^{-1}\rfloor\,\}$ and $\gamma_{w_{\min}} = \text{id} - \max\{\,w_{\min}, (\text{id}+\delta)^{-1}\,\}$, $\tilde{\gamma}^*$ 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,

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

which is exactly the bound stated in the Main Theorem.

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

### 3.2.1 The idle time

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

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

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

$$\text{imbalance}(\{\mathcal{C}_1,\ldots,\mathcal{C}_j\}) \;\le\; (p-1)\cdot(h+T_j)$$

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

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

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

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 $j$th through $(j+p-1)$th chunk 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

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

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

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

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

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 $\tilde{\delta} \geq 6A \cdot \max\{\,\beta - \alpha, \mathrm{id}\,\}$ on $[\,w_{\min}, \lfloor (\mathrm{id} + \tilde{\delta})^{-1}(n/p) \rfloor\,]$, claims a bound on the wasted time in terms of the function $\tilde{\gamma} = \mathrm{id} - \max\{\,w_{\min}, \lfloor (\mathrm{id} + \tilde{\delta})^{-1} \rfloor\,\}$. To this end, let $x \leq n/p$ and $w = \max\{\,w_{\min}, \lfloor (\mathrm{id} + \tilde{\delta})^{-1}(x) \rfloor\,\}$, for which owing to the assumption on $\tilde{\delta}$,

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

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

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

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

$$(3A\beta)^{-1}(x) \geq \left\lfloor (\mathrm{id} + \tilde{\delta})^{-1}(x) \right\rfloor,$$

and thus also

$$\left\lfloor (3A\beta)^{-1}(x) \right\rfloor \geq \left\lfloor (\mathrm{id} + \tilde{\delta})^{-1}(x) \right\rfloor.$$

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

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

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. □

## 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 is ignored by all fixed-partition algorithms.

Let us 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 $\varrho : \mathbb{R}^+ \to \mathbb{R}^+$, an instance of BAL is obtained by implementing two functions $\varrho_1, \varrho_2 : \mathbb{R}^+ \to \mathbb{R}^+$, one for each phase; we will denote such an instance by $\text{BAL}(\varrho_1, \varrho_2)$. 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 $\max\{\, w_{\min}, (\text{id} + \delta)^{-1}(W/p)\,\}$, where $\delta = \alpha^{-1} \circ (\beta - \alpha)$ and $w_{\min}$ is the minimal chunk size. For technical reasons, we will actually take $\varrho_1 = \lfloor (\text{id} + \tilde{\delta})^{-1} \rfloor$, where for some $K \geq 6$,

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

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 will be 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 $\varrho_1 = \lfloor (\text{id} + \tilde{\delta})^{-1} \rfloor$ as above, we have $w = \lfloor (\text{id} + \tilde{\delta})^{-1}(W/p) \rfloor$. Hence, if $W/p < K \cdot w_{\min}$, we have $w = 0$ and $d = W/p/K$, while in the opposite case, we have $(\text{id} + \tilde{\delta})(w) \leq W/p$ 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 large compared to $w$. 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 will assume 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 will be referred to as the *tolerance interval* of that round. The quantities $t - d/2$ and $t + h + d/2$ will be called the *lower* and *upper tolerance (threshold)* of the round, respectively. Figure 3.1 below gives an illustration of what has been described so far.

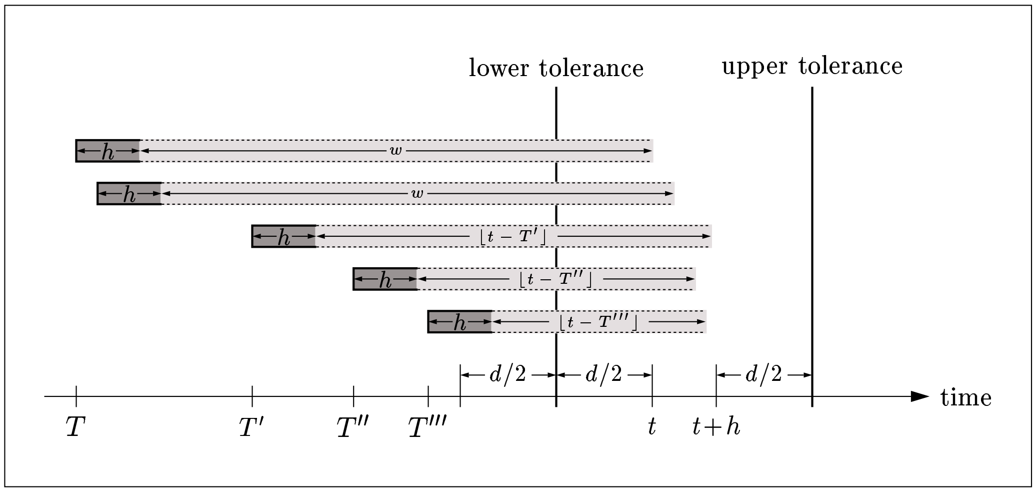


FIGURE 3.1: Chunk assignment in a round started at time $T$, with target $t$ and tolerance $d$. The light gray rectangles indicate chunk sizes and not processing times.

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 $\varrho_2$. According to Theorem 3.1, for a given variance estimator $[\,\alpha, \beta\,]$ and minimal chunk size $w_{\min}$, we take $\varrho_2 : x \mapsto \lfloor \beta^{-1}(x/A + \beta(w_{\min})) \rfloor$, for some $A \ge 1$ with $\alpha \ge \mathrm{id}/A$.

Figure 3.2 below gives a pseudo C-code implementation of the function that computes, for a given request, the chunk size according to BAL($\varrho_1, \varrho_2$). 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).

```
(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 $s$;
```

FIGURE 3.2: The size computed by BAL($\varrho_1, \varrho_2$) for a chunk scheduled at time $T$

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 will be 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 $i$th execution of lines (2)–(5), that is, during the $i$th 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 $t_i^{\text{low}}$ and $t_i^{\text{upp}}$, respectively. Round $i$ ends at time $t_i - d_i$, which will be denoted by $t_i^{\text{end}}$.

For simplicity, let us 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 3.3.

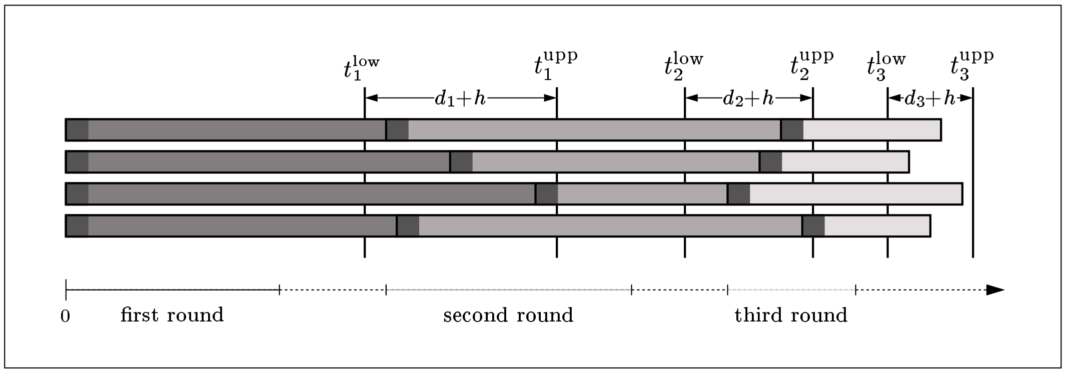


FIGURE 3.3: Three rounds in the deviationless case.

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 *busyness* of a 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 further reduce the imbalance. 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 will 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 $K = 49A$, $\tilde{\delta} : w \mapsto K \cdot \max\{\, w_{\min}, 2 \cdot \max\{\, \beta(w) - w, w - \alpha(w)\,\}\,\}$ 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$(\varrho_1, \varrho_2)$ with $\varrho_1 : x \mapsto \lfloor (\mathrm{id} + \tilde{\delta})^{-1}(x) \rfloor$ and $\varrho_2 : x \mapsto \lfloor \beta^{-1}(x/A + \beta(w_{\min})) \rfloor$ produces a schedule $\mathcal{S}$ with the property that

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

where $\gamma_{w_{\min}}$ is the progess rate associated with $[\,\alpha, \beta\,]$ and $w_{\min}$, and for some partition $\mathcal{S} = \mathcal{S}_1 \mathbin{\dot\cup} \mathcal{S}_2 \mathbin{\dot\cup} \mathcal{S}_3$, $\varepsilon = (h + \varepsilon_1) \cdot (h + \varepsilon_2) \cdot (h + \varepsilon_3)/h^2 - h \leq (h + \varepsilon_1 + \varepsilon_2 + \varepsilon_3)^3/h^2 - h$, 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 $\varepsilon_i$ 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 $\varepsilon$ 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 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 3.3.6 derives the desired bound on the wasted time.

### 3.3.1 Terminology

Let $\mathcal{S}_{\mathrm{I}}$ and $\mathcal{S}_{\mathrm{II}}$ 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, $W_i$, $w_i$, $d_i$, $t_i$ will denote the values of BAL's program variables $W$, $w$, $d$, $t$ in the various rounds, and for $i = 1, \ldots, r$, $t_i^{\mathrm{end}} = t_i - d_i$, $t_i^{\mathrm{low}} = t_i - d_i/2$, and $t_i^{\mathrm{upp}} = t_i + h + d_i/2$. It will further be convenient to have $d_0 = t_0^{\mathrm{upp}} = 0$, and to denote by both $W_{r+1}$ and $n'$ the number of unassigned tasks when the second phase begins. Besides, let us define $\tilde{\gamma} = \max\{\,0\,, \mathrm{id} - \max\{\, w_{\min}, \varrho_1\,\}\,\} =$

$\max\{\,0\,,\,\mathrm{id} - \max\{\,w_{\min}, \lfloor(\mathrm{id}+\tilde{\delta})^{-1}\rfloor\,\}\,\}$, and recall that, by the definition of $\varrho_1$ and because of lines (2),(3) and (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, let us write $\mathcal{R}_i$ for the subschedule pertaining to the chunks assigned in round $i$; clearly then $\mathcal{S}_{\mathrm{I}} = \mathcal{R}_1 \cup \cdots \cup \mathcal{R}_r$. 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 $i$ will be denoted by $\mathrm{busy}(\mathcal{R}_i)$, and $\mathrm{busy}(\mathcal{S}_{\mathrm{I}}) = \mathrm{busy}(\mathcal{R}_1) + \cdots + \mathrm{busy}(\mathcal{R}_r)$. 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 $T'$ (cf. 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 $i$ will be denoted by $\mathrm{lazy}(\mathcal{R}_i)$, for $i = 1,\dots,r$, and $\mathrm{lazy}(\mathcal{S}_{\mathrm{I}}) = \mathrm{lazy}(\mathcal{R}_1) + \cdots + \mathrm{lazy}(\mathcal{R}_r)$.

### 3.3.2 Local properties of a round

This section provides a number of simple properties of $\mathcal{S}_{\mathrm{I}}$, which will constitute the basic building blocks of the further analysis. In fact, Lemmas 3.8 through 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 $\mathrm{busy}(\mathcal{R}_i)$ and $\mathrm{lazy}(\mathcal{R}_i)$ for round $i$. The following Lemma 3.7 says that the amount of time that the finishing time of a chunk $\mathcal{C}$, denoted by $\mathrm{finish}(\mathcal{C})$, 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 $\mathrm{finish}(\mathcal{C})$ denoting the time when it is completed,

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

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

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

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

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

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

and according to the definitions made in Section 2.2, $\mathrm{early}_\alpha(\mathcal{C}) = \max\{\,0, \alpha(w') - \text{proc-time}(\mathcal{C})\,\}$, $\mathrm{late}_\beta(\mathcal{C}) = \max\{\,0, \text{proc-time}(\mathcal{C}) - \beta(w')\,\}$, and $\mathrm{finish}(\mathcal{C}) = T' + h + \text{proc-time}(\mathcal{C})$. 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. $\square$

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

**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 $K \geq 30$, $d_i \leq w_i/6 \leq (1/5 - 1/K) \cdot w_i$, which is equivalent to $(W_i/p - w_i)/K \leq (1/5 - 1/K) \cdot w_i$, which implies the two inequalities stated in the lemma. $\square$

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

**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 $W_{i+1} \geq p \cdot Kd_i - \text{busy}(\mathcal{R}_i)$. $\square$

**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, $w_{i+1} \geq 5/K \cdot W_{i+1}/p \geq 5d_i - \text{busy}(\mathcal{R}_i)/(3p)$. $\square$

**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 $W_{r+1} = 0$, 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 $W_{i+1} \leq p \cdot Kd_i + p \cdot 2.5d_{i-1} + \text{lazy}(\mathcal{R}_i)$. $\square$

**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,

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

□

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

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

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

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 $\mathcal{S}_{\mathrm{I}}$ to its total earliness and lateness.

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

**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 $j = 1, \ldots, l-1$, $\mathrm{early}_\alpha(\mathcal{C}_j) \geq t_i^{\mathrm{low}} - \mathrm{finish}(\mathcal{C}_j)$. 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 $t_i^{\mathrm{end}} = t_i^{\mathrm{low}} - d_i/2$. Hence, for all $j = 1, \ldots, l-1$,

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

where, according to line (7), and since $\mathrm{finish}(\mathcal{C}_j)$ 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.** $\mathrm{lazy}(\mathcal{S}_{\mathrm{I}}) \leq \text{sum-late}_\beta(\mathcal{S}_{\mathrm{I}}) + \text{sum-early}_\alpha(\mathcal{S}_{\mathrm{I}})$.

**Proof:** This proof is a bit longer, so let us give a plan. We will 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 let us consider a fixed processor in a round $i$, where $2 \leq i \leq r$. Let $T$ denote the finishing time of the last chunk assigned to the processor before round $i$, and write $L$ for the processor's its 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

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

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

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

In any case therefore

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

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 emtpy). 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, $\text{finish}(\mathcal{C}) - t_j^{\text{upp}} \leq \text{late}_\beta(\mathcal{C})$, while by Lemma 3.10(c),

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

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

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

$\square$

### 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 will 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 much 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 $\mathcal{S}_3 = \mathcal{S}_{\text{II}} \ne \varnothing$, 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,

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

We further have to bound $\text{chunks}(\mathcal{S}_{\text{II}})$, and for that purpose recall that in the second phase of BAL chunks are scheduled according to $\text{FP}(x \mapsto \lfloor \beta^{-1}(x/A + \beta(w_{\min})) \rfloor)$. 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, $\tilde{\delta}(w) \geq 6A \cdot w$, for all $w \leq \lfloor (\mathrm{id} + \tilde{\delta})^{-1}(n'/p) \rfloor$. On the other hand, we easily verify by means of the definition of $\tilde{\delta}$ that $\tilde{\delta} \geq K \cdot (\beta - \alpha) \geq 6A \cdot (\beta - \alpha)$. Therefore $\tilde{\delta}$ fulfills the requirements of (the addendum to) Theorem 3.1, and we obtain that

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

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,

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

**The irregular case:** $r' < r$ or $W_{r+1} \leq \mathrm{lazy}(\mathcal{S}_{\mathrm{I}})$

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

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

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,

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

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

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

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

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

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. But 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

Let us first bound the idle time of $\mathcal{S}_{\mathrm{I}}$, which is also the initial imbalance of the schedule $\mathcal{S}_{\mathrm{II}}$ produced in the second phase of BAL. Let $r'$ be the index of a round with maximal upper limit, that is, $t_{r'}^{\mathrm{upp}} \geq t_i^{\mathrm{upp}}$, 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$(\mathcal{S}_{\mathrm{I}})$ into three parts $I'$, $I''$ and $I'''$, namely the amount of idle time of $\mathcal{S}_{\mathrm{I}}$ spent before $t_{r'}^{\mathrm{end}}$, between $t_{r'}^{\mathrm{end}}$ and $t_{r'}^{\mathrm{upp}}$, and after $t_{r'}^{\mathrm{upp}}$, respectively. We will first bound each of these quantities separately.

Let us start with $I'$, which is the hard part. By Lemma 3.10(a), $t_{r'}^{\mathrm{end}} \leq t_r^{\mathrm{end}} \leq t_r^{\mathrm{low}}$, so that it suffices to bound the total amount of time that processors finish before $t_r^{\mathrm{low}}$, 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

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

Concerning $I'''$, let $\mathcal{C}$ denote the last chunk to finish in $\mathcal{S}_{\mathrm{I}}$, and note that its finishing time is just the makespan of $\mathcal{S}_{\mathrm{I}}$. Now if $i$ denotes the round in which $\mathcal{C}$ was scheduled, then, since round $r'$ has maximal upper tolerance threshold, $\text{finish}(\mathcal{C}) - t_{r'}^{\mathrm{upp}} \leq \text{finish}(\mathcal{C}) - t_i^{\mathrm{upp}}$, which by Lemma 3.7 is at most $\text{late}_\beta(\mathcal{C})$. The makespan of $\mathcal{S}_{\mathrm{I}}$ is thus at most $t_{r'}^{\mathrm{upp}} + \text{late}_\beta(\mathcal{C})$, and we conclude that

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

Finally, we may trivially bound the idle time $I''$ spent between $t_{r'}^{\mathrm{end}}$ and $t_{r'}^{\mathrm{upp}}$ by $p \cdot (t_{r'}^{\mathrm{upp}} - t_{r'}^{\mathrm{end}}) = p \cdot h + p \cdot 1.5 d_{r'}$. 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 maximal upper tolerance threshold, $t_{r'}^{\mathrm{upp}} \geq t_{r'+1}^{\mathrm{upp}}$, 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

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

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

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

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

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

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

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

### 3.3.6 The wasted time

In the last two sections, we have proven that

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

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

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

This almost proves Theorem 3.2, except that it remains to bound $\mathcal{E}$ in terms of quantities $\varepsilon_1$, $\varepsilon_2$, and $\varepsilon_3$ according to the theorem. Note, at this point, that even though $\mathcal{E}\le \text{av-dev}_{\alpha,\beta}(\mathcal{S})\cdot \mathrm{chunks}(\mathcal{S})$, we cannot bound $\mathcal{E}$ in terms of av-dev$_{\alpha,\beta}(\mathcal{S})$ (not to mention 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 $\varepsilon_i = \mathcal{E}_i/\mathrm{chunks}(\mathcal{S}_i)$, and note that $\mathcal{E}\le\mathcal{E}_1+\mathcal{E}_2+\mathcal{E}_3$.

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

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

In the following, we want to bound $\mathcal{E}_2$ in terms of $\varepsilon_1$ and $\varepsilon_2$, and $\mathcal{E}_3$ in terms of $\varepsilon_1$, $\varepsilon_2$, and $\varepsilon_3$. 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 $\text{chunks}(\mathcal{S}_2) \leq \lceil \text{sum-early}_\alpha(\mathcal{S}_1)/h \rceil \leq \mathcal{E}_1/h + 1 \leq (1 + \varepsilon_1/h) \cdot pR$, and hence

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

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

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

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

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

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

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

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 $\varepsilon \leq (h + \varepsilon')^3/h^2 - h$, 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 to 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 $i$th chunk after the upper tolerance threshold $t_{i-1}^{\mathrm{upp}}$ 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 $t_{i-1}^{\mathrm{upp}}$ the request is issued, the processor is assigned a chunk as if it had requested exactly at time $t_{i-1}^{\mathrm{upp}}$, and also the corresponding update of $W$ is performed at that time. It will be convenient to say that the $i$th chunk of a processor *belongs* to round $i$, which makes sense, because even though such a chunk might be assigned long after the $i$th round is over, its size is computed from the settings of that round. The effect of the described modifications, compared to the orginal BAL strategy, is illustrated in Figure 3.4 below. Note that without deviations, BAL′ and BAL behave identically.

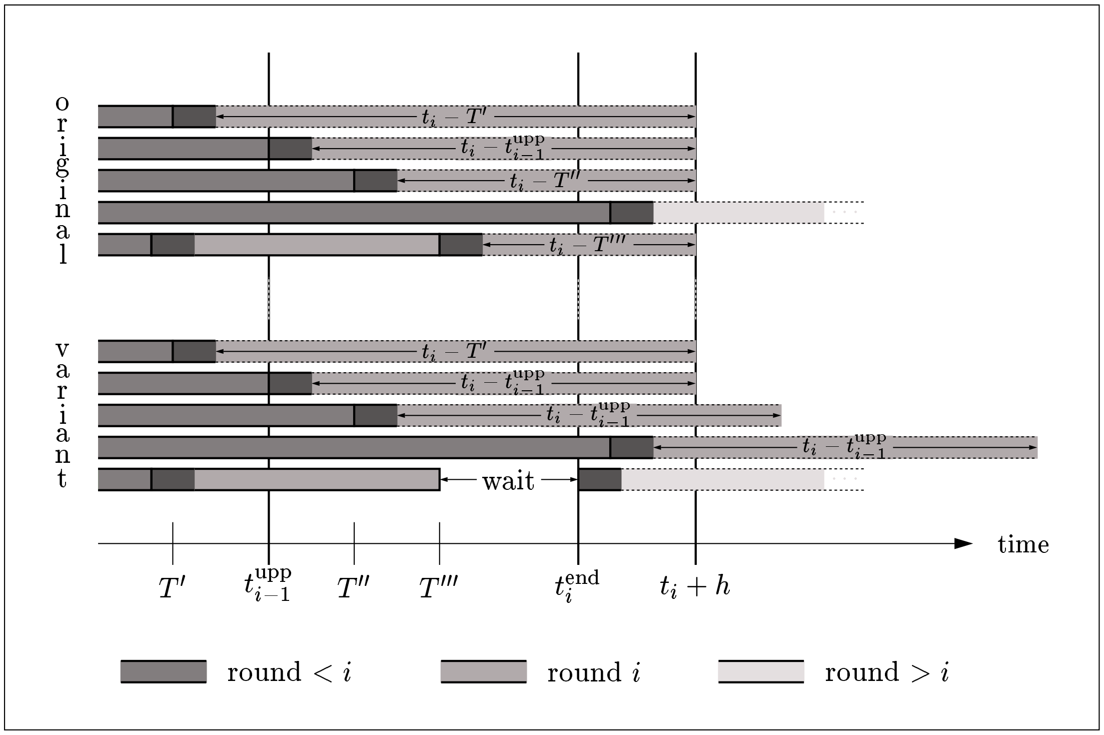


FIGURE 3.4: The original BAL compared to its variant BAL′ in a round $i$.

In analogy to the piece of code given for BAL, Figure 3.5 below 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, respectively, with the latter three being initialized to zero. And 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 zeroes—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 $t_{i-1}^{\text{upp}}$, 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.

```
$r[k] = r[k] + 1;$
**if** $($ PHASE $== 1$ && $r[k] > R\,)$ { (* new round *)
    **if** $(\,T < t - d\,)$ **wait until** $t - d;$
    $T = \max\{\,t - d, \min\{\,t + h + d/2, T\,\}\,\};$
    **if** $(\,R > 0\,)$ $W = W - p' \cdot s[R];$
    $p' = p;$
    $w = \varrho_1(W/p);$
    $s[R+1] = \min\{\,w, \lfloor T + w - t - d/2 \rfloor\,\};$
    $d = (W/p - w)/K;$
    **if** $(\,d > w/6\,)$ PHASE $= 2$ **else** $R = R + 1;$
    $t = T + h + w;$
}
**if** $(\,r[k] < R\,)$ $\{\,s = s[r[k]];\ W = W + s;\,\}$
**if** $(\,r[k] == R\,)$ $\{\,s = \min\{\,w, \max\{\,s[R], \lfloor t - T \rfloor\,\}\,\};\ p' = p' - 1;\,\}$
**if** $(\,r[k] > R\,)$ $\{\,s = \varrho_2(W/p);\,\}$
$s = \min\{\,W, s\,\};$
$W = W - s;$
**return** $s;$
```

FIGURE 3.5: The chunk size computed by BAL$'(\varrho_1, \varrho_2)$ for a request of the $k$th processor at time $T$

For most parts, the code of Figure 3.5 is a straightforward implementation of BAL$'$ as described above, but let us 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 $K = 49A$, $\tilde{\delta} : w \mapsto K \cdot \max\{\, w_{\min}, 2 \cdot \max\{\, \beta(w) - w, w - \alpha(w)\,\}\,\}$ 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$'(\varrho_1, \varrho_2)$ with $\varrho_1 : x \mapsto \lfloor (\mathrm{id} + \tilde{\delta})^{-1}(x) \rfloor$ and $\varrho_2 : x \mapsto \lfloor \beta^{-1}(x/A + \beta(w_{\min})) \rfloor$ produces a schedule $\mathcal{S}$ with the property that

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

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 will derive 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 will 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 $i$th 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 will be 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 the 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 will demonstrate 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 of all 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 $i$th chunk assigned to the $k$th processor, for $k = 1, \ldots, p$, and denote by $P'$ and $P''$ the sets of indices of processors for whose $i$th 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 $t_i^{\text{end}}$, and, by the previous two lemmas, $w_{i+1} \geq 5/K \cdot W_{i+1}/p \geq 5d_i$,

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

□

**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 the 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 $t_{i-1}^{\text{upp}}$ comes before the end of round $i$, we know that $w = \min\{ w_i, \lfloor t_i - T \rfloor \}$, for $T \le t_{i-1}^{\text{upp}}$, 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

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

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 $i \in [1\,..\,r]$, $W_{i+1} \le p \cdot Kd_i + p \cdot 2.5d_{i-1}$.

**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 $t_i \ge t_{i-1}^{\text{end}} + h + w_i = t_{i-1}^{\text{upp}} - 1.5d_{i-1} + w_i$, is at least $\lfloor w_i - 1.5d_{i-1} \rfloor \ge w_i - 2.5d_{i-1}$. By Lemmas 3.16 and 3.17 therefore, $W_{i+1} \le W_i - p \cdot w_i + p \cdot 2.5d_{i-1} = p \cdot Kd_i + p \cdot 2.5d_{i-1}$. □

**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

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

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 will therefore have to separately bound the imbalance and the waiting time of $\mathcal{S}_{\text{I}}$.

To this end, let us 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 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 finish$(\mathcal{C}_{i-1}) > t_{i-1}^{\text{upp}}$, $\mathcal{C}_i$ is scheduled at time finish$(\mathcal{C}_{i-1})$ and Lemma 3.20 says that 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 idle$(\mathcal{S}_{\text{I}}) = I' + I'' + I'''$, where $I'$, $I''$, $I'''$ denote the amount of idle time spent before $t_r^{\text{low}}$, between $t_r^{\text{low}}$ and $t_r^{\text{upp}}$, and after $t_r^{\text{upp}}$, 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 $t_r^{\text{low}}$, we also have

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

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

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

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

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

so that finally

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

### 3.4.4 The wasted time

The last two sections have shown that

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

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

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

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

# Chapter 4

# Specific upper bounds

In this chapter, we will apply our Main Theorem, proven over the course of the last chapter, to the particular independent-tasks, bounded-tasks, and coupled-tasks settings, which we already mentioned in the introduction, and which will be defined properly later in this chapter. These applications turn out to be challenging tasks on their own; since already the proof of the generic result was 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 will 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 relationship between processing time irregularity and scheduling performance expressed by our Main Theorem. Section 4.2 provides general-purpose probabilistic preliminaries. Sections 4.3, 4.4, and 4.5 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 $\gamma : \mathbb{R} \to \mathbb{R}$, the following theorem provides general-purpose approximations for the values of $\gamma^*$ from above as well as from below. The addendum says that unless $\gamma$ grows very slowly, namely with slope tending to zero, the stated bounds are tight up to a constant factor. For convenience, let us agree to write $\gamma^*(x, y)$ for $\min\{\, i : \gamma^{(i)}(x) \leq y \,\}$ in the following; in particular, then $\gamma^*(x) = \gamma^*(x, 0)$.

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

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

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

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

In particular, this property holds if the (piecewise) derivative of $\gamma$ 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 $\gamma$ is an increasing bijection. Using that we easily verify that for arbitrary $x$,

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

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 $i \in \mathbb{N}$,

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

For intuition behind this approximation, see Figure 4.1, 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.

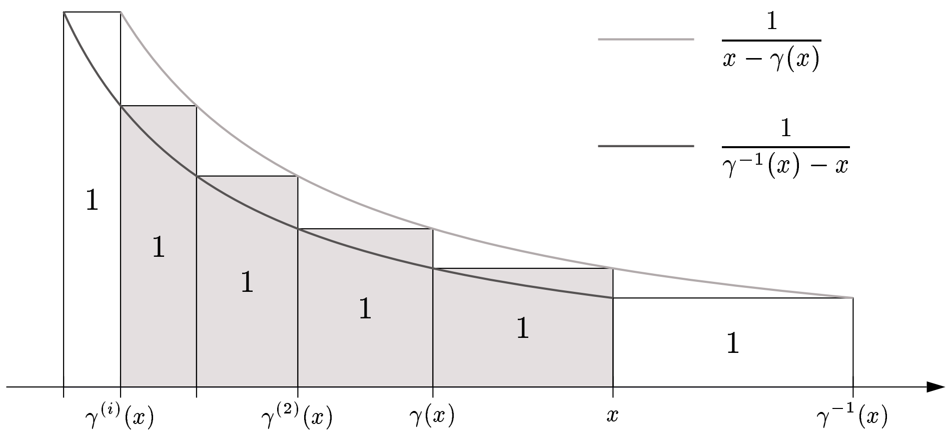


FIGURE 4.1: How $\gamma^*(\,x, \cdot\,)$ is bounded by two integrals.

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

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

and

$$\int_y^x \frac{dz}{z-\gamma(z)} \;>\; \int_{\gamma^{(i-1)}(x)}^x \frac{dz}{z-\gamma(z)} \;\geq\; i-1,$$

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 $\gamma$,

$$\frac{z-\gamma(z)}{\gamma^{-1}(z)-z} \;=\; \frac{\gamma(z')-\gamma(\,\gamma(z')\,)}{z'-\gamma(z')} \;\geq\; Q.$$

□

In the following, we will apply the above theorem to instantiate the generic bound from the previous chapter for a variety of variance estimators. Our results are summarized in the following table, where $\kappa$ 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 the corresponding Lemmas 4.1, 4.2, 4.3, and 4.4 below.

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

TABLE 4.1: The bound from the Main Theorem for four types of variance estimators.

For better readability, in the following proofs we will 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 $(\beta(w) - \alpha(w))/w$ 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

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

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 $N \geq AB \cdot M^\kappa$,

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

**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 $\gamma$ 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, let us first check whether the preconditions on $\gamma$ are satisfied. Since $\beta$ 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 $\beta^{-1}$ on $[\beta(M), \infty)$ is bounded from above by $1/(B\kappa) < 1$. This is easily seen to imply that all difference quotients of $\gamma$ are bounded from below by $1 - 1/(B\kappa) > 0$, and, in particular, that $\gamma$ 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

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

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 $\gamma$, is independent of the choice of our time unit, that is, it is invariant under the simultaneous multiplication of $\alpha$ and $\beta$ 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 $N \geq AB \cdot M^\kappa$,

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

as desired. □

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

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

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 $N \geq (\, AB\, M \ln^{\kappa}(C\, M)\,)^2$,

$$\gamma^*(N) = \Theta\Big(\, AB \cdot \ln N \cdot \ln^{\kappa}(CN)\,\Big).$$

**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 $\gamma$ are bounded from below by $1/2$ so that by Theorem 4.1,

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

Unfortunately, with $\beta(w) = B \cdot w \cdot \ln^{\kappa}(Cw)$, no closed form for the inverse $\beta^{-1}$ exists. But 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

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

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. $\square$

### 4.1.2 Linear width

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

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

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 $N \geq (\, AB \cdot M \,)^2$,

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

**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 $N \geq (AB\, M)^2$ is within a constant factor of $AB \cdot \ln N$. □

This result implies the following interesting property of linear-width variance estimators, which, as is implied by Theorem 3.1, give rise to particularly simple scheduling schemes, namely $\mathrm{FP}(x \mapsto x/(AB) + w_{\min})$ when $\alpha = \mathrm{id}/A$ and $\beta = B \cdot \mathrm{id}$. Consider the polylogarithmically superlinear variance estimator

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

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

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

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

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

But 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

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

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, that is, of the form $[\, \mathrm{id}/A, B \cdot \mathrm{id} \,]$, are optimal for a wide variety of settings. As we will see in Section 6.2, these schemes are particularly simple in that chunk sizes decrease geometrically, that is, each chunk has one $C$th of the size of the previously assigned chunk, where $C$ is just $A \cdot B$. 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

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

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 $N \geq \max\{\, M, (2AC)^{\frac{\kappa}{\kappa-1}}\,\}$,

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

**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 $\beta$, as it has been in the cases considered before. This proof is therefore going to be more involved than its predecessors.

Since $\alpha(w) = \max\{\, w/A, w - Cw^{1/\kappa}\,\}$, $\delta$ may have a sharp (that is, not differentiable) bend, which turns out to be somewhat unhandy 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 $\tilde{N} = \max\{\, \delta(M), (2AC)^{\frac{\kappa}{\kappa-1}}\,\}$; note that the second term is just the unique positive fixpoint of $\delta$, 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 $(-\infty\,,\, \delta(M)\,]$, $\gamma^{-1}$ describes a straight line with slope 1, while the derivative of $\delta^{-1}$ : $z \mapsto z^{\kappa}/(2AC)^{\kappa}$ at an arbitrary $z \in (0\,,\, (2AC)^{\frac{\kappa}{\kappa-1}}\,]$ is

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

We may therefore conclude that $\gamma^{-1}$ has slope bounded by $\kappa + 1$ everywhere on $(-\infty, \tilde{N}]$, and hence that $\gamma$ 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

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

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 $\gamma^*(\,N, \tilde{N}\,) = \min\{\, i : \gamma^{(i)}(N) \leq \tilde{N}\,\}$, that is, the number of iterations of $\gamma$ required to get from $N$ to $\tilde{N}$. In fact, we will actually bound $\gamma^*(N, \tilde{N}+M)$, which differs by at most one from $\gamma^*(\,N, \tilde{N}\,)$. Unfortunately, the derivative of $\delta^{-1}$ 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 $\gamma^*$ 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 $T : z \mapsto \ln(z/\hat{N})$ with inverse $T^{-1} : z \mapsto \hat{N} \cdot e^z$, where $\hat{N} = (2AC)^{\frac{\kappa}{\kappa-1}}$ denotes the fixpoint of $\delta$; 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

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

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

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

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

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

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

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

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

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

This finishes the analysis of variance estimators with sublinear width, and we have finally proven all the bounds claimed in Table 4.1. □

## 4.2 Probabilistic preliminaries

An application of the Main Theorem to a stochastic setting requires bounds on the expected deviations of the processing times of chunks with respect to some variance estimator. As a preparation for proving such bounds, the following lemmas provide approximations for $\mathbf{E}\max\{0, Z\}$, for general $Z$ as well as for the specific case when $Z$ is the sum of $m$ independent (not necessarily identically distributed) random variables. For the latter case, we also provide an approximation from below, which will be used several times in Chapter 7. Note that we have taken care to formulate the bounds of the two lemmas below analogously, which explains the somewhat peculiar formulation of Lemma 4.5.

**Lemma 4.5.** For an arbitrary random variable $Z$ with mean $\mu_Z < 0$ and finite variance $\sigma_Z^2 > 0$, it holds that for $t = -\mu_Z/\sigma_Z$,

$$\mathbf{E}\max\{0,\ Z\} \le \sigma_Z/t.$$

**Proof:** Since the mean of a nonnegative random variable $X$ can be expressed as $\int_0^\infty \Pr(X > x)\,dx$ (see, for example, (Grimmett and Stirzaker, 1992)), it holds that

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

Since $(Z-\mu_Z)/\sigma_Z$ has mean zero and variance one, it then follows by Chebyshev's inequality that

$$\mathbf{E}\max\{0,\ Z\} \le \sigma_Z\cdot\int_t^\infty \frac{dx}{x^2} = \sigma_Z/t.$$

□

**Lemma 4.6.** Let $Z$ be the sum of $m$ independent random variables with finite variances $\sigma_1^2, \dots, \sigma_m^2$ and finite third central moments $\varrho_1^3, \dots, \varrho_m^3$, and assume that $Z$ has mean $\mu_Z \le 0$ and variance $\sigma_Z^2 > 0$. Then for some constant $\vartheta > 0$, with $t = -\mu_Z/\sigma_Z$ and $\eta = \vartheta \cdot e^{t^2/2} \cdot \sum_{i=1}^m \varrho_i^3 \,/\, \big(\sum_{i=1}^m \sigma_i^2\big)^{3/2}$,

$$2/3 - \eta \;\le\; \frac{\mathbf{E}\max\{\,0,\,Z\,\}}{\sigma_Z \cdot \dfrac{1}{\sqrt{2\pi}}\,\dfrac{1}{1+t^2}\,e^{-t^2/2}} \;\le\; 1 + \eta.$$

If $Z$ is normal, $\eta$ may be replaced by zero. For $t = 0$, the $2/3$ may be replaced by 1.

**Proof:** 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,

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

where $\Phi$ denotes the standard normal distribution function. If $t = 0$, we are already done. Otherwise, the lemma is implied by the approximation

$$\frac{1}{\sqrt{2\pi}}\,\frac{t}{1+t^2}\,e^{-t^2/2} \;\le\; 1 - \Phi(t) \;\le\; \frac{1}{\sqrt{2\pi}}\,\frac{1/(3t)+t}{1+t^2}\,e^{-t^2/2},$$

using that $1 - t \cdot \frac{t}{1+t^2} = \frac{1}{1+t^2}$ and $1 - t \cdot \frac{1/(3t)+t}{1+t^2} = \frac{2}{3}\,\frac{1}{1+t^2}$. The proof of this approximation is analogous to that given, for example, in (Grimmett and Stirzaker, 1992) for a slightly weaker bound. Namely it suffices to check that for all $x$,

$$\frac{d}{dx}\,\frac{1/(3x)+x}{1+x^2}\,e^{-x^2/2} \;\le\; -e^{-x^2/2}$$

and

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

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

Now assume that $Z$ is the sum of $m$ independent random variables, and define $t$ and $\eta$ as done in the lemma. 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, just as in the proof of Lemma 4.5 above,

$$\mathbf{E}\max\{0, Z\} \;=\; \sigma_Z \cdot \int_t^\infty \Pr\Big((Z-\mu_Z)/\sigma_Z > x\Big)\,dx,$$

and

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

By Theorem 5.17 of (Petrov, 1995), there exists a constant $\vartheta > 0$, such that for all $x > 0$,

$$\Big|\Pr\Big((Z-\mu_Z)/\sigma_Z > x\Big) - \Pr\Big((\tilde{Z}-\mu_Z)/\sigma_Z > x\Big)\Big| \le \frac{\vartheta}{\sqrt{2\pi}} \cdot \frac{\sum_{i=1}^m \varrho_i^3}{\big(\sum_{i=1}^m \sigma_i^2\big)^{3/2}} \cdot \frac{1}{(1+x)^3},$$

and hence, with $\eta = \vartheta \cdot e^{t^2/2} \cdot \sum_{i=1}^m \varrho_i^3 \,/\, \big(\sum_{i=1}^m \sigma_i^2\big)^{3/2}$,

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

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

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

and we may conclude that

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

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

## 4.3 Bounded tasks

If for some $T_{\min}, T_{\max} > 0$ with $T_{\min} \le 1 \le T_{\max}$, 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 $T_{\min} \le 1 \le T_{\max}$ is not a restriction but merely reflects a comittment 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 4.7.** 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

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

**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 $T_{\max} \cdot w$. □

**Corollary 4.1.** 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

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

**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

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

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

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

and $w_{\min}$. According to our comments concerning the choice of $w_{\min}$ at the beginning of Chapter 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 $\overset{*}{\gamma}_{w_{\min}}(n/p) = O(T_{\max}/T_{\min} \cdot \log(n/p))$, and we finally obtain

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

□

## 4.4 Independent tasks

If for some $\sigma > 0$, the processing times of the tasks are independent, identically distributed, nonnegative random variables with mean 1, variance $\sigma^2$, and finite third moment, we will say that task processing times are *independent, with variance* $\sigma^2$. The application of our Main Theorem to this particular setting will be somewhat tricky, since the number of tasks that an optimal algorithm (namely BAL) selects for a chunk is not independent of the processing times of previously assigned chunks. Here as well as later in the paper, all bounds concerning an independent-tasks setting contain an implicit factor of $\varrho^3/\sigma^3$, where $\sigma^2$ is the variance and $\varrho^3$ 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 4.8.** Let task processing times be independent, with variance $\sigma^2$. 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

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

**Remark:** If task processing times are normal, $\sqrt{p+\ln w}$ may be replaced by $\sqrt{2\ln p + \ln w}$.

**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 $\mathcal{C}$,

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

Let $s = \vartheta\cdot\varrho^3/\sigma^3$, where $\varrho^3$ is the absolute third central moment of a single task's processing time, and $\vartheta$ 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

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

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

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

It follows immediately that

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

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 $\varepsilon$.

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 $j = 1,\ldots,n$,

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

Since

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

proving the lemma reduces to bounding the expectation of $\frac{1}{n}\sum_{j=1}^{n} Y_j$ by $\varepsilon$.

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 $j=1,\ldots,n$,

$$\mathbf{E}\Big[\frac{1}{j}\sum\nolimits_{i=1}^{j} Y_i \;\Big|\; Y_1,\ldots,Y_{j-1}\Big] \;=\; \frac{1}{j}\sum_{i=1}^{j-1} Y_i + \frac{1}{j}\cdot\mathbf{E}[\,Y_j \mid Y_1,\ldots,Y_{j-1}\,] \;\le\; \frac{1}{j}\sum_{i=1}^{j-1} Y_i + \frac{\varepsilon}{j},$$

which, by taking expectation on both sides, implies that

$$\mathbf{E}\Big[\frac{1}{j}\sum\nolimits_{i=1}^{j} Y_i\Big] \le \frac{j-1}{j}\cdot\mathbf{E}\Big[\frac{1}{j-1}\sum\nolimits_{i=1}^{j-1} Y_i\Big] + \frac{\varepsilon}{j}.$$

A simple induction now shows that

$$\mathbf{E}\Big[\frac{1}{n}\sum\nolimits_{j=1}^{n} Y_j\Big] \le \varepsilon,$$

which immediately implies the desired bound

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

$\square$

With the help of Lemma 4.8, it is now easy to translate the Main Theorem to the independent-tasks setting. We remark that Corollary 4.2 below implies a doubly logarithmic asymptotic bound as $n$ grows large, which settles a conjecture put forward in (Hagerup, 1996). 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 $O((h+\sigma^2)\,+\,\sigma\sqrt{p}\cdot(h+\sigma^2)^{1/2})$ (see, for example, (Gumbel, 1954) or (Hartley and David, 1954)).

**Corollary 4.2.** Let task processing times be independent, with variance $\sigma^2$, and let the overhead be $h\ge 1$. 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

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

**Proof:** For every $w_{\min}\in\mathbb{N}$ with $w_{\min}\ge h$, 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

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

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

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

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 $\lambda$, such that for all $w \in \mathbb{N}$, $\sqrt{p+\ln w} \cdot w^{1/2} \leq C \cdot \sqrt{p} \cdot w^{1/2+\lambda/2}$, 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

$$[\,\tilde{\alpha},\tilde{\beta}\,] : w \mapsto \Big[\, \max\{\, w/2 \,,\; w - C \cdot \sigma \cdot \sqrt{p} \cdot w^{1/2+\lambda/2} \,\} \,,\; w + C \cdot \sigma \cdot \sqrt{p} \cdot w^{1/2+\lambda/2} \,\Big].$$

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

$$\tilde{\gamma}^{*}_{w_{\min}}(n/p) = O\Big(\, \log\log\frac{n}{p} + \sigma\sqrt{p} \cdot w_{\min}^{\lambda/2-1/2} \,\Big) = O\Big(\, \log\log\frac{n}{p} + \sigma\sqrt{p} \cdot (h+\sigma^2)^{\lambda-1/2} \,\Big).$$

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

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

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

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

□

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 4.2 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 4.3.** Let task processing times be independent, with variance $\sigma^2$, 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

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

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

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

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

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

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 4.8 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

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

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

## 4.5 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 $\sigma^2$, 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* $\sigma^2$. 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 4.9.** Let task processing times be coupled, with minimum $T_{\min}$ and variance $\sigma^2$. 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

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

where

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

**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 time, 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 Lemma 4.5 applied with $\sigma_Z \le \sigma w$ and $\mu_Z \le -(p-1) \cdot w^2$,

$$\mathbf{E}\, \mathrm{late}_\beta(\mathcal{C}) = \mathbf{E} \max\{\, 0,\, T - pw^2 \,\} \;\le\; \frac{\sigma^2 w^2}{(p-1) \cdot w^2} \;\le\; \sigma^2/(p-1).$$

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

□

**Corollary 4.4.** Let task processing times be coupled, with minimum $T_{\min}$ and variance $\sigma^2$, 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

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

**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

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

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,

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

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

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

□

# Chapter 5

# Lower Bounds

This chapter complements our findings from the previous two chapters 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 to 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 chapter. For the important independent-tasks setting, we strengthen our lower bound by providing proof that an efficient strategy must not choose the initial chunk sizes significantly larger than the corresponding optimal instance of BAL.

## 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 remarked already at the beginning of Chapter 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 $\mathrm{id}/\alpha$ be a decreasing function, the (addendum to the) theorem below makes do with the superadditivity of $\alpha$; this is indeed a weaker requirement, since for arbitrary $w \geq v > 0$, provided that $\alpha(w)/w \geq \alpha(v)/v$,

$$\alpha(w+v) \geq (w+v)\cdot\alpha(w)/w = \alpha(w) + v\cdot\alpha(w)/w \geq \alpha(w) + v\cdot\alpha(v)/v = \alpha(w)+\alpha(v).$$

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

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

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

**Addendum:** If, additionally, $\alpha$ is superadditive, that is, for all $w, v > 0$, $\alpha(w+v) \geq \alpha(w) + \alpha(v)$, and provided that $\beta$ 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 will first prove the lower bound under the assumption that the given algorithm does not incur any *waiting time*. Section 5.1.2 will show how to extend this proof to the general case. The final 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 waitingless case

The basic and rather obvious idea of the proof is to play an adversary and fix the chunk processing times (and hence the $T_1, \ldots, T_n$) incrementally, along with the scheduling decisions made by our algorithm. Let us 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 to 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 right away 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 $\text{av-dev}_{\alpha,\beta}(\mathcal{S}) = \text{am-dev}_{\alpha,\beta}(\mathcal{S}) = \varepsilon$. The remainder of this proof will derive 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 $\mathcal{C}_{\text{peak}}$ 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 right at the time of allocation. For an illustration, see Figure 5.1.

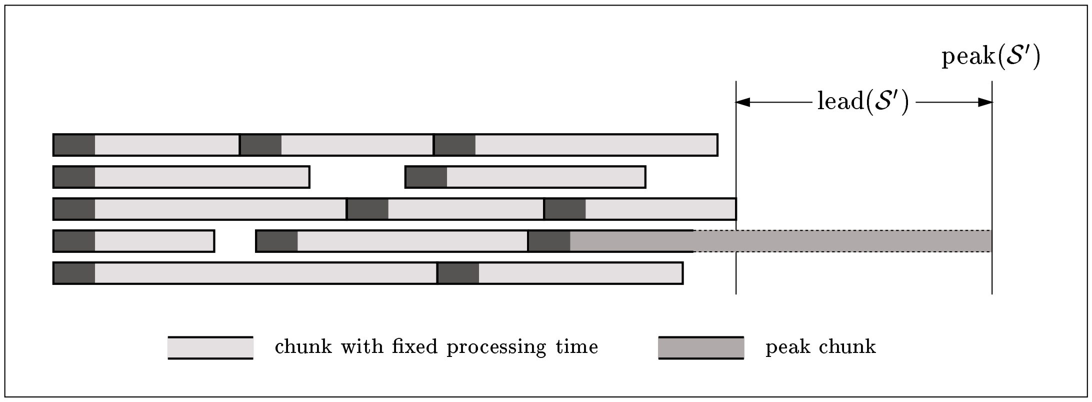


FIGURE 5.1: The peak chunk, peak, and lead of a schedule $\mathcal{S}'$.

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 imbalance$(\mathcal{S}') \geq (p-1)\cdot$lead$(\mathcal{S}')$, for every prefix $\mathcal{S}'$ of $\mathcal{S}$, and, for the complete schedule,

$$\text{imbalance}(\mathcal{S}) \geq (p-1)\cdot(\,\text{lead}(\mathcal{S}) + \varepsilon\cdot\text{chunks}(\mathcal{S})/(p-1)\,) = (p-1)\cdot\text{lead}(\mathcal{S}) + \varepsilon\cdot\text{chunks}(\mathcal{S}).$$

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

**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, $\text{peak}(\mathcal{S}') \geq T+h+\beta(w)$. 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 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

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

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

**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 waitingless case 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 5.1. 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 next to last 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 $\text{lead}(\mathcal{S}'') \geq \tilde{\gamma}(\,\text{lead}(\mathcal{S}')\,)$. □

**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

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

To get rid of the $r$, 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

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

Two items remain in order to prove Theorem 5.1. First, to relate $\tilde{\gamma}^{*}$ to $\gamma^{*}$, 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 $x \geq 0$,

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

which immediately implies that $\gamma^{*}(x) \leq 2 \cdot \tilde{\gamma}^{*}(x)$. In order to get rid of the $w_{\max}$, 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

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

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

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

**Proof:** The proof is trivial if either $\beta(w_{\max}) \geq \alpha(n/p)$ or $\alpha(w_{\max}) \leq h + \varepsilon$, so let us 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 $\gamma^{(i)}(\alpha(n/p)) \leq \beta(w_{\max})$, so that, in particular,

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

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

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

In combination with the above we thus obtain

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

□

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 will 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 occured 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 let us 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 $\mathrm{chunks}(\mathcal{S}') < \mathrm{chunks}(\mathcal{S}'') \leq \mathrm{chunks}(\mathcal{S}') + \lfloor p/2 \rfloor$,

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

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

**Proof:** Let $T$ denote the time of the latest allocation in $\mathcal{S}''$, so that 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,

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

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 $x, y \geq 0$. $\square$

**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,

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

□

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

$$\mathrm{waste}(\mathcal{S}) = \frac{1}{p}\cdot\Big(\,h\cdot\mathrm{chunks}(\mathcal{S}) + \mathrm{idle}(\mathcal{S})\,\Big) \geq \frac{p-1}{2p}\cdot\Big(\,(h+\varepsilon)\cdot(r-1) + \tilde{\gamma}^{(r+1)}(\beta(w_{\max}))\,\Big),$$

which differs by a factor of exactly 2 from the bound obtained after Lemma 5.3 in the proof for the waitingless case. 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 $\mathrm{id} + (\beta-\alpha)\circ\alpha^{-1} = \beta\circ\alpha^{-1}$,

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

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

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

and therefore

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

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

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

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

$$\begin{aligned}\min\Big\{ \alpha \circ \Big( \mathrm{id} - (\mathrm{id} + \delta)^{-1} \Big) \,,\, \alpha \circ (\mathrm{id} - \alpha^{-1}(h+\varepsilon)) \Big\} \\ \leq \quad \min\Big\{ \Big( \mathrm{id} - \alpha \circ \beta^{-1} \Big) \circ \alpha \,,\, (\mathrm{id} - (h+\varepsilon)) \circ \alpha \Big\}.\end{aligned}$$

By the monotonicity of $\alpha$, $\min\{ \alpha \circ f, \alpha \circ g \} = \alpha \circ \min\{ f, g \}$ 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

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

We have thus proven that

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

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

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

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

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

and we have finally proven that

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

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 approach, so that we cannot expect the obtained result to translate easily to a setting where processing times are randomly distributed. However, as will be 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 will derive from this general result lower bounds for two specific instances of the independent-tasks and the coupled-tasks setting.

### 5.2.1 General bound

Like Theorem 5.1 from the previous section, also Theorem 5.2 below is formulated in terms of the progress rate of a variance estimator. The intuition behind the requirement to $[\,\alpha, \beta\,]$ in Theorem 5.2 is that the processing times of a chunk of size $w$ should be likely to be below $\alpha(w)$, as well as to be 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,

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

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

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

where $\gamma$ 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. Namely, both proofs shows 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). As we already remarked above, the proof of the last assertion made use of the full power of the variance-estimator based approach. The approach taken in the following proof is that, very intuitively spoken, a large chunk is likely to cause an imbalance too large to be rebalanced by the remaining work. In fact, Theorem 5.1 could have also been proven along this line of argumentation, however, with somewhat more efforts. While the proof of Theorem 5.1 considered batches of $\Theta(p)$ scheduling operations, the proof below will take a more simplistic approach by coarsly quantifying the effect of individual scheduling operations. This will account 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 non-probabilistic) 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 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 $T$,

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

**Proof:** For $k = 1, \ldots, p$, let us denote by $H_k$ the sum of all overheads and waiting times of the $k$th 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

$$p \cdot \mathrm{waste}(\mathcal{S}')$$

$$\begin{aligned}
&= h \cdot \text{chunks}(\mathcal{S}') + \text{idle}(\mathcal{S}') \\
&= \sum\nolimits_{k=1}^{p} H_k + p \cdot \max\{ t_1 + H_1 + T_1, \ldots, t_p + H_p + T_p \} - \sum\nolimits_{k=1}^{p} (t_k + H_k + T_k) \\
&\geq p \cdot \max\{ t_1, \ldots, t_p \} - \sum\nolimits_{k=1}^{p} t_k - \sum\nolimits_{k=1}^{p} T_k \\
&= I - T.
\end{aligned}$$

□

**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 $\tilde{\gamma} = \text{id} - (\text{id} + (\beta - \alpha)/(4K))^{-1}$,

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

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 will first establish a lower bound on $\mathbf{E}\,\text{imbalance}(\{\mathcal{C}\})$, that is, the expected imbalance incurred by $\mathcal{C}$, for which we consider two cases. Either 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 $h + (\alpha(w) + \beta(w))/2$. But 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. Let us 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

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

note that $W_{j+1} = W_j - (W_j - W_{j+1})$, where $W_j - W_{j+1}$ is just the size of the $j$th chunk. But 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

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

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

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

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

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

At this point, recall that all 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 appropriately relate $\min\{ \tilde{\gamma}, \mathrm{id} - h \} = \mathrm{id} - \max\{ h, (\mathrm{id} + (\beta - \alpha)/(4K))^{-1} \}$ to $\gamma = \mathrm{id} - \max\{ h, (\mathrm{id} + \alpha^{-1} \circ (\beta - \alpha))^{-1} \}$. But 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

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

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 estimating 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 a particular coupling. We remark that Corollary 5.5 below settles one of the open problems put forward in (Hagerup, 1996).

**Corollary 5.5.** Let task processing times be independent, normal and with variance $\sigma^2$, 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 produced by an arbitrary scheduling algorithm given $n$ tasks and $p$ processors satisfies

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

**Remark:** For the purists: we strongly conjecture that by exploiting the specific properties of the normal distribution this bound can be improved to $\Omega(h \cdot \log_h \log(n/p) + \sigma\sqrt{h})$

**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 (those from (Grimmett and Stirzaker, 1992) do equally well), there exists a constant $K \geq 1$ such that

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

as well as

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

The precondition to Theorem 5.2 is hence fulfilled with

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

and the theorem gives us

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

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

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

□

**Corollary 5.6.** Let task processing time 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

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

**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 $\Pr\left(T \le (T_{\min} + \Delta T)\cdot w\right) \ge 1/4$ as well as $\Pr\left(T \ge (T_{\max} - \Delta T)\cdot w\right) \ge 1/4$. The preconditions to Theorem 5.2 are therefore satisfied with

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

so that we obtain

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

where $\gamma$ is the progress rate associated with $[\,\alpha,\beta\,]$ and $h$. Since $1 \le (T_{\min} + 3T_{\max})/(3T_{\min} + T_{\max}) \le 3$, Lemma 4.3 implies that for $n/p \ge (3h)^2$, $\gamma^*(n) = \Omega(\log n)$, which proves the corollary. □

## 5.2.3 The independent-tasks setting

This final section of Chapter 5 is concerned with a very specific lower bound, which shows that in the independent-tasks setting great care has to be taken so as not to choose the size of the initial $p$ chunks too large. To understand the precise meaning of the lemma below, recall that Lemma 4.8 associated with the independent-tasks setting with variance $\sigma^2$ the variance estimator

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

According to our description in Section 3.3, the instance of Bal corresponding to this variance estimator, for $n$ tasks and $p$ processors, chooses the size $w_1$ of the first $p$ chunks such that for some constant $K$,

$$p\cdot w_1\ +\ p\cdot K\cdot\sqrt{p + \ln w_1}\cdot \sigma w_1^{1/2} = n.$$

Recall that this choice of $w_1$ guarantees that after the initial $p$ chunks have been scheduled, a sufficient number of tasks is left to compensate for the imbalance caused by these chunks. What Lemma 5.9 then says is that the term $\sqrt{p + \ln w_1}$ in this inequality may not be replaced by a constant without the effect of a wasted time in the order of $\sqrt{n/p}$.

The lemma is significant for two reasons. First, it gives an indication that, even though the lower bound stated in Corollary 5.5 differs from the upper bound of the corresponding Corollary 4.2 by a factor of approximately $p$, the balancing strategy yields optimal results for the independent-tasks setting. Second, it implies lower bounds on the performance of various existing scheduling heuristics, which indeed do not choose the initial chunk sizes careful enough; we will elaborate on this in Chapter 7.

**Lemma 5.9.** *Let task processing times be independent, with variance $\sigma^2$, and let the overhead be $h$. Then for all $n, p \in \mathbb{N}$, any algorithm that given $n$ tasks and $p$ processors chooses for one of the first $p$ chunks a size of $w_1$ such that $w_1 + C\cdot\sigma\sqrt{w_1} = n/p$, for a fixed $C \ge 0$, produces a schedule $\mathcal{S}$ with*

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

**Remark:** More precisely, for sufficiently large $p$ and $n/p$ it holds that

$$\mathbf{E}\,\text{waste}(\mathcal{S}) \;\geq\; \frac{1}{4}\,\frac{1}{1+C^2}\,e^{-C^2/2}\cdot\sigma\cdot\sqrt{n/p}.$$

**Proof:** Let $\mathcal{C}_1$ denote any one of the first $p$ chunks with a size $w_1$ satisfying $w_1 + C\cdot\sigma\sqrt{w_1} = n/p$. Also denote by $T_{\mathcal{C}_1}$ its processing time, and by $T_{\mathcal{S}\setminus\mathcal{C}_1}$ the total processing time of the remaining chunks. Since the imbalance of the schedule consisting of $\mathcal{C}_1$ is just $(p-1)\cdot T_{\mathcal{C}_1}$, we may deduce from Lemma 5.7 that

$$p\cdot\text{waste}(\mathcal{S}) \;\geq\; \text{imbalance}(\{\mathcal{C}_1\}) - T_{\mathcal{S}\setminus\mathcal{C}_1} \;=\; (p-1)\cdot T_{\mathcal{C}_1} - T_{\mathcal{S}\setminus\mathcal{C}_1}.$$

Thus, with $Z = (p-1)\cdot T_{\mathcal{C}_1} - T_{\mathcal{S}\setminus\mathcal{C}_1}$,

$$\mathbf{E}\,\text{waste}(\mathcal{S}) \;\geq\; \mathbf{E}\max\{\,0, Z\,\}\;/\;p.$$

To prove the desired lower bound, it therefore remains to establish an appropriate lower bound on $\mathbf{E}\max\{\,0, Z\,\}$, which we will obtain by an application of Lemma 4.6.

Namely, since $T_{\mathcal{C}_1}$ and $T_{\mathcal{S}\setminus\mathcal{C}_1}$ are the sum of $w_1$ and $n - w_1$ random variables, respectively, which are all independent (and identically distributed), $Z$ is the sum of $n$ independent (though not identically distributed) random variables. Let $\sigma_Z^2$ denote the sum of the variances of these random variables, let $\varrho_Z^3$ denote the sum of their third absolute central moments, and write $\mu_Z$ for the mean of $Z$. Then Lemma 4.6 says that for some constant $\vartheta > 0$, for $t = -\mu_Z/\sigma_Z$ and $\eta = \vartheta\cdot e^{-t^2/2}\cdot\varrho_Z^3/\sigma_Z^3$,

$$\mathbf{E}\max\{\,0, Z\,\} \geq (2/3-\eta)\cdot\sigma_Z\cdot\frac{1}{\sqrt{2\pi}}\,\frac{1}{1+t^2}\,e^{-t^2/2}.$$

In the remainder of the proof, we will show that for $p$ and $n/p$ tending to infinity, $\eta\to 0$, $t\to C$, and $p^{-1}\cdot\sigma_Z/\sqrt{n/p}\to\sigma$. Owing to $2/3\cdot(2\pi)^{-1/2} > 1/4$, this will prove the (remark and hence the) lemma.

Since $Z = (p-1)\cdot T_{\mathcal{C}_1} - T_{\mathcal{S}\setminus\mathcal{C}_1}$, we have that, with $\sigma^2$ and $\varrho^3$ denoting the variance and the third absolute central moment, respectively, of a single task's processing time,

$$\begin{aligned}
\mu_Z &= (p-1)\cdot w_1 - n + w_1 = p\cdot w_1 - n;\\
\sigma_Z^2 &= (p-1)^2\cdot w_1\cdot\sigma^2 + (n-w_1)\cdot\sigma^2;\\
\varrho_Z^3 &= (p-1)^3\cdot w_1\cdot\varrho^3 + (n-w_1)\cdot\varrho^3.
\end{aligned}$$

By the assumption on $w_1$, $n = p\cdot w_1 + p\cdot C\cdot\sigma\sqrt{w_1} \geq p\cdot w_1$, which implies that $\mu_Z = -p\cdot C\cdot\sigma\sqrt{w_1}$, and, under the additional assumption that $w_1 \geq n/(2p)$, also that

$$p^2\cdot\sigma^2\,w_1 \;\geq\; \sigma_Z^2 \;\geq\; p(p-1)\cdot\sigma^2\,w_1.$$

Therefore, if only $w_1 \geq n/(2p)$,

$$C \leq t = -\frac{\mu_Z}{\sigma_Z} \leq \frac{p}{\sqrt{p(p-1)}} \cdot C,$$

and thus $t \to C$ as $p \to \infty$. Furthermore, with $s = \varrho^3/\sigma^3$,

$$\frac{\varrho_Z^3}{\sigma_Z^3} = \frac{(w_1 \cdot (p-1)^3 + n - w_1) \cdot \varrho^3}{\sigma_Z \cdot (w_1 \cdot (p-1)^2 + n - w_1) \cdot \sigma^2} \leq \sigma/\sigma_Z \cdot (p-1) \cdot \varrho^3/\sigma^3 \leq s/\sqrt{w_1},$$

which proves that $\eta \to 0$ as $w_1$ tends to infinity. Concerning the asymptotic behaviour of $w_1$, check that according to the (implicit) definition of $w_1$,

$$w_1 = n/p - C^2\sigma^2/2 \cdot \left( \sqrt{1 + 4 \cdot n/p \cdot C^{-2}\sigma^{-2}} - 1 \right),$$

which implies that for arbitrary $K \geq 1$, and $n/p \geq 5K^2/4 \cdot C^2\sigma^2$, $w_1 \geq (1 - 1/K) \cdot n/p$. Altogether, we have thus proven that for $p$ and $n/p$ tending to infinity, $t \to C$, $\eta \to 0$, and $p^{-1} \cdot \sigma_Z/\sqrt{n/p} \to \sigma$, from which we may finally conclude that for sufficiently large $p$ and $n/p$,

$$\mathbf{E}\,\mathrm{waste}(\mathcal{S}) \geq \mathbf{E}\max\{0, Z\} / p \geq \frac{1}{4}\frac{1}{1+C^2}e^{-C^2/2} \cdot \sigma \cdot \sqrt{n/p}.$$

□

# Chapter 6

# Simple Strategies

In view of the upper bounds proven in Chapters 3 and 4, and the matching or almost matching lower bounds proven in Chapter 5, we have, in a sense, exhaustively explored our scheduling problem at this point. This chapter puts emphasis on a more practical aspect of the problem, which so far has only played a secondary role: the simplicity of the scheduling algorithm. We will address this issue by investigating in depth two classes of particularly simple scheduling algorithms. The first are the *fixed-size* algorithms which take each chunk of the same, fixed size; note that these include the two extremes of self-scheduling (one task at a time) and static chunking (one large chunk for each processor). The second class are the so-called *geometric* algorithms, which assign a sequence of chunks of geometrically decreasing sizes. For both classes, we will provide general bounds according to our generic approach, as well as specific bounds for each of the bounded-tasks, independent-tasks, and coupled-tasks setting. Our results will demonstrate that comitting to a fixed chunk size is inevitably coupled with a significant performance loss, while for every reasonable parameter setting, geometric algorithms can achieve wasted times surprisingly close to the theoretical optimum. In fact, we also give evidence that no similarly simple algorithms can yield better results. Moreover, our bounds for the geometric schemes have an exactly specified, very small constant factor, thus allowing for more accurate performance guarantees than those obtained from our general analysis.

## 6.1 Fixed chunk size

As opposed to all of the more involved heuristics, schemes that schedule a fixed number of tasks at a time have been the subject of theoretical investigations before, namely in the work of Kruskal and Weiss (1985). However, their results were confined to the independent-tasks setting and even there turned out to be suboptimal in general; this will be detailed in Section 7.3.

### 6.1.1 Generic bound

The following theorem states an upper and a lower bound on the wasted time of an arbitrary fixed-size scheme with respect to an arbitrary variance estimator $[\,\alpha,\beta\,]$. According to the theorem, the bounds match each other up to a factor of 4; however, as will become evident by the proof, as $n,p\to\infty$ the matching factor actually approaches 1. To simplify our presentation, we will assume throughout this section that the number $p$ of processors divides the number $n$ of tasks and that the chunk size divides $n/p$; this will spare us tedious, and not very instructive considerations of special cases.

**Theorem 6.1.** Let task processing times be arbitrary, and let the overhead be $h\geq 1$. Let $[\,\alpha,\beta\,]$ be a variance estimator, and let $w\in\mathbb{N}$. Then for all $n,p\in\mathbb{N}$ with $n/(pw)\in\mathbb{N}$, given $n$ tasks and $p$ processors, $\mathrm{FP}(x\mapsto w)$ produces a schedule $\mathcal{S}$ with

$$\mathrm{waste}(\mathcal{S})\leq(h+\varepsilon)\cdot n/(pw)+\min\{\,h+\beta(w),n/(pw)\cdot(\beta(w)-\alpha(w))\,\},$$

where $\varepsilon=\text{av-dev}_{\alpha,\beta}(\mathcal{S})$. On the other hand, for every $\varepsilon\geq 0$, and for all $n,p\in\mathbb{N}$ with $n/(pw)\in\mathbb{N}$, 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, $\mathrm{FP}(x\mapsto w)$ produces a schedule with $\text{av-dev}_{\alpha,\beta}(\mathcal{S})=\varepsilon$ and

$$\mathrm{waste}(\mathcal{S})\geq(h+\varepsilon)\cdot n/(pw)+\min\{\,h+\beta(w),n/(pw)\cdot(\beta(w)-\alpha(w))\,\}\;/\;4.$$

**Proof:** We begin with the proof of the upper bound, for which we define $\mathcal{E}=\text{sum-early}_\alpha(\mathcal{S})+(p-1)\cdot\text{sum-late}_\beta(\mathcal{S})$ as the sum of the deviations of all the chunks in $\mathcal{S}$. Let $T$ denote the time of the last scheduling operation in $\mathcal{S}$, and let $\mathcal{C}$ denote the last chunk to finish. Since $\mathcal{C}$ is scheduled at or before $T$, the makespan of the schedule $\mathcal{S}$ is bounded by $T$ plus the scheduling overhead plus the processing time of $\mathcal{C}$, and hence by $T+h+\beta(w)+\text{late}_\beta(\mathcal{C})$. On the other hand, by the definition of $T$, no processor finishes before $T$, so that the total imbalance of $\mathcal{S}$ is bounded as

$$\mathrm{imbalance}(\mathcal{S})\leq(p-1)\cdot(h+\beta(w)+\text{late}_\beta(\mathcal{C}))\leq(p-1)\cdot(h+\beta(w))+\mathcal{E}.$$

When $w$ is large, we can prove a better bound, as is shown next. Consider $p$ successively allocated chunks $\mathcal{C}_1,\ldots,\mathcal{C}_p$ with processing times $T_1,\ldots,T_p$ (note that these chunks are not necessarily allocated to different processors). If all of these chunks finish before the last one of the previously assigned chunks, these allocations can only decrease the imbalance. In the opposite case, assume without loss of generality that $\mathcal{C}_1$ is the chunk to finish last. It is then easy to see that the imbalance increases by no more than $(p-1)\cdot(h+T_1)-\sum_{j=2}^{p}(h+T_j)=(p-1)\cdot T_1-\sum_{j=2}^{p}T_j$. Since all chunks are of size $w$, the last term is clearly bounded by

$$(p-1)\cdot(\beta(w)+\text{late}_\beta(\mathcal{C}_1))-\sum_{j=2}^{p}(\alpha(w)-\text{early}_\alpha(\mathcal{C}_j)),$$

which is at most

$$(p-1)\cdot(\beta(w)-\alpha(w)) + \text{sum-early}_\alpha(\{\mathcal{C}_1,\ldots,\mathcal{C}_p\}) + (p-1)\cdot\text{sum-late}_\beta(\{\mathcal{C}_1,\ldots,\mathcal{C}_p\}).$$

This proves that $p$ allocations can increase the imbalance by at most $(p-1)\cdot(\beta(w)-\alpha(w))$ plus the deviations of the corresponding chunks. Via a simple inductive argument, this immediately implies that

$$\text{imbalance}(\mathcal{S}) \le n/w\cdot(\beta(w)-\alpha(w)) + \mathcal{E},$$

and together with the first bound, we have thus shown that

$$\text{imbalance}(\mathcal{S})/p \le \min\{\, h+\beta(w), n/(pw)\cdot(\beta(w)-\alpha(w))\,\} + \mathcal{E}/p.$$

Using that $\text{chunks}(\mathcal{S}) = n/w$ and $\varepsilon = \mathcal{E}/\text{chunks}(\mathcal{S})$, we now easily obtain the desired upper bound

$$\text{waste}(\mathcal{S}) \le (h+\varepsilon)\cdot n/(pw) + \min\{\, h+\beta(w), n/(pw)\cdot(\beta(w)-\alpha(w))\,\}.$$

For the lower bound, given some $\varepsilon \ge 0$, we construct the following sequence of chunk processing times. Let $m = n/(pw)$ and $m' = \min\left\{ m, \left\lfloor \frac{h+\beta(w)}{\beta(w)-\alpha(w)} \right\rfloor \right\} \le m$. Then the processing times of the first $m-m'$ chunks of each processor are all fixed to $w$. For the remaining chunks, we distinguish between the $p-1$ first (say) processors and the $p$th processor. For each of the $p-1$ first processors, the processing time of each chunk after the $(m-m')$th is fixed to $\alpha(w)$. For the $p$th processor the processing time of each chunk after the $(m-m')$th is fixed to $\beta(w)$, except for the last chunk, whose processing time is fixed to $\beta(w)+\varepsilon\cdot n/w \;/\; (p-1)$. This construction concentrates all the deviation on a single chunk and obviously induces a schedule $\mathcal{S}$ with both average and amortized deviation $\varepsilon$. On the other hand, our construction guarantees that each processor is assigned exactly $m$ chunks, which can be seen as follows. The first $m-m'$ chunks of each processor all have processing time $w$, so that the first $p\cdot(m-m')$ chunks of the schedule are equally distributed among the processors. Now assume, by way of induction, that the first $p\cdot(m-m'+i)$ chunks of the schedule are equally distributed among the processors, for some $i$ between 0 and $m'-1$. By construction, the finishing time of these chunks is $(m-m')\cdot(h+w)+i\cdot(h+\alpha(w))$ for each of the first $p-1$ processors, whereas for the $p$th processor it is $(m-m')\cdot(h+w)+i\cdot(h+\beta(w))$. The difference is just $i\cdot(\beta(w)-\alpha(w))$, which for $i \le m'-1$, according to our definition of $m'$, is certainly bounded by

$$(m'-1)\cdot(\beta(w)-\alpha(w)) \;\le\; \frac{h+\beta(w)}{\beta(w)-\alpha(w)}\cdot(\beta(w)-\alpha(w)) \;-\; (\beta(w)-\alpha(w)) \;\le\; h+\alpha(w).$$

But this inequality is just the guarantee that of the next $p$ chunks, each of which takes time at least $h+\alpha(w)$, exactly one will be assigned to each processor (assuming, without loss of generality, that when two processors become idle at the same time, the processor with the larger index is served first). In particular, we have thus shown that the difference between the finishing time of the $p$th processor to that of any one of the other processors is exactly $m' \cdot (\beta(w) - \alpha(w)) + \varepsilon \cdot n/w \;/\; (p-1)$, therefore

$$\text{imbalance}(\mathcal{S}) \;=\; (p-1) \cdot m' \cdot (\beta(w) - \alpha(w)) + p \cdot \varepsilon \cdot m,$$

and hence, since $m' = \min\left\{ m, \left\lfloor \frac{h+\beta(w)}{\beta(w)-\alpha(w)} \right\rfloor \right\}$ and $\lfloor x \rfloor \geq x/2$ for $x \geq 1$,

$$\text{imbalance}(\mathcal{S})/p \;\geq\; \min\{\, m \cdot (\beta(w) - \alpha(w)), h + \beta(w)\,\} \;/\; 4 + m \cdot \varepsilon.$$

Since the number of scheduling operations per processor is exactly $m = n/(pw)$, this gives us the lower bound

$$\text{waste}(\mathcal{S}) \geq (h + \varepsilon) \cdot m + \min\{\, m \cdot (\beta(w) - \alpha(w)), h + \beta(w)\,\} \;/\; 4.$$

□

To convey a feeling for the very general bound stated in Theorem 6.1 above, Table 6.1 below shows special cases for a selection of variance estimators and chunk sizes. Actually, the selection of variance estimators is exactly the same as used in Chapter 4 for exemplary instantiations of our Main Theorem, except that, for simplicity, in Table 6.1 below it is assumed that $\alpha = \text{id}$. The first two rows pertain to the self-scheduling (SS) and static chunking (SC) scheme, respectively. The third row, marked FIX, states the performance for the respectively optimal fixed chunk size; note that from the bound of Theorem 6.1 a closed form for this optimal size and for the corresponding bound cannot be derived in general, but very easily so for the considered special instances. For the sake of comparison, finally, the fourth row states the optimal wasted time according to our Main Theorem. The table makes very clear that, even for an optimal choice of chunk size, the wasted time performance of a fixed-size algorithm is never even close to the optimum.

| $\beta(w)$ | $w + C \cdot w^{1/\kappa}$ | $B \cdot w$ | $B \cdot w \log^\kappa(Cw)$ | $B \cdot w^\kappa$ |
|---|---|---|---|---|
| SS | $H \cdot N$ | $H \cdot N$ | $H \cdot N$ | $H \cdot N$ |
| SC | $H + C \cdot N^{1/\kappa}$ | $H + B \cdot N$ | $H + B \cdot N \cdot \log^\kappa(CN)$ | $H + B \cdot N^\kappa$ |
| FIX | $\sqrt{H \cdot N}$ | $\sqrt{B} \cdot \sqrt{H \cdot N}$ | $\sqrt{B \log^\kappa(CN)} \cdot \sqrt{H \cdot N}$ | $B^{\frac{1}{\kappa+1}} \cdot (HN)^{1-\frac{1}{\kappa+1}}$ |
| OPT | $H \cdot \log\log N$ | $H \cdot B \cdot \log N$ | $H \cdot B \cdot \log^\kappa(CN)N$ | $H \cdot B^{1/\kappa} \cdot N^{1-1/\kappa}$ |

TABLE 6.1: The order of the wasted time of the fixed-size schemes compared to the optimal strategy for a variety of variance estimators, where $H = h + \varepsilon$ and $N = n/p$.

We next apply Theorem 6.1 to each of the bounded-tasks, independent-tasks, and coupled-tasks settings, defined in Chapter 4. A summary of these bounds will be given in Table 6.2, in the next section.

### 6.1.2 Bounded tasks

**Corollary 6.7.** Let task processing times be bounded in $[\,T_{\min}, T_{\max}\,]$, and let the overhead be $h \geq 1$. Then for all $n, p \in \mathbb{N}$, and for any $w \in \mathbb{N}$ that is within a constant factor of $\sqrt{h \cdot T_{\max} \cdot n/p}$ and for which $n/(pw) \in \mathbb{N}$, given $n$ tasks and $p$ processors, $\mathrm{FP}(x \mapsto w)$ produces a schedule $\mathcal{S}$ with

$$\mathrm{waste}(\mathcal{S}) = O\Big(\,\sqrt{h \cdot T_{\max} \cdot n/p}\,\Big).$$

**Proof:** By (the obvious) Lemma 4.7, the average deviation of $\mathcal{S}$ with respect to $[\,\alpha, \beta\,] : w \mapsto [\,T_{\min} \cdot w, T_{\max} \cdot w\,]$ is always zero, so that by Theorem 6.1,

$$\mathrm{waste}(\mathcal{S}) \leq h \cdot n/p \;/\; w + h + \beta(w) = O\Big(\,h \cdot n/p \;/\; w + T_{\max} \cdot w\,\Big).$$

The last term would obviously be minimized for $w$ equal to $\sqrt{h \cdot n/p/T_{\max}}$, and we easily verify that plugging in a term in that order yields the desired bound. $\square$

### 6.1.3 Independent tasks

**Corollary 6.8.** Let task processing times be independent, with variance $\sigma^2$, and let the overhead be $h \geq 1$. Then for all $n, p \in \mathbb{N}$, and for any $w \in \mathbb{N}$ that is within a constant factor of

$$\min\left\{\,\sqrt{(h+\sigma)\cdot n/p}\,,\,\left(\frac{(1+h/\sigma)\cdot n/p}{\sqrt{p+\ln(n/p)}}\right)^{2/3}\,\right\},$$

and for which $n/(pw) \in \mathbb{N}$, given $n$ tasks and $p$ processors, $\mathrm{FP}(x \mapsto w)$ produces a schedule $\mathcal{S}$ with

$$\mathbf{E}\,\mathrm{waste}(\mathcal{S}) = O\left(\,\sqrt{(h+\sigma)\cdot n/p}\cdot\left(\,1+\frac{\sqrt[3]{\sigma^2(p+\ln(n/p))}}{\sqrt[6]{(h+\sigma)\cdot n/p}}\,\right)\right).$$

**Proof:** By Lemma 4.8, the expected average deviation of $\mathcal{S}$ with respect to

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

is bounded by some $\varepsilon$ with $\varepsilon = O(\sigma)$. According to Theorem 6.1 therefore

$$\mathbf{E}\,\mathrm{waste}(\mathcal{S}) = O\Big(\,(h+\sigma)\cdot n/p \;/\; w + \beta(w)\,\Big) = O\Big(\,(h+\sigma)\cdot n/p \;/\; w + w + \sigma\sqrt{p+\ln w}\cdot w^{1/2}\,\Big),$$

and let us be brave and try to derive the value of $w$ that minimizes the last term. To that end, let us introduce $Q = (h+\sigma)\cdot n/p$, $R = \sigma\sqrt{p+\ln(n/p)}$, $f_1 : w \mapsto Q/w + w$, and $f_2 : w \mapsto Q/w + R\cdot\sqrt{w}$, using which we can write

$$\mathbf{E}\,\mathrm{waste}(\mathcal{S}) = O\Big(\,\max\{\,f_1(w), f_2(w)\,\}\,\Big).$$

Now it is easy to check that $f_1$ and $f_2$ have their absolute minima at $w_1 = \sqrt{Q}$ and $w_2 = (2Q/R)^{2/3}$, respectively, and the two functions intersect exactly once on the positive reals, namely at $w_\times = R^2$. Next, verify that if $w_1 \geq w_\times$, the absolute minimum of $\max\{\, f_1, f_2 \,\}$ lies at $w_1$, if $w_2 \leq w_\times$, it lies at $w_2$, while if $w_1 < w_\times < w_2$, it lies at $w_\times$. This can be seen to imply that the absolute minimum of $\max\{\, f_1, f_2 \,\}$ lies at $\min\{\, w_2, \max\{\, w_1, w_\times \,\}\,\}$ with a value of at most $\max\{\, f_1(w_1), f_2(w_2) \,\}$. For an arbitrary $w$ in the order of $\min\{\, w_2, \max\{\, w_1, w_\times \,\}\,\}$, which in fact is in the order of $\min\{\, w_1, w_2 \,\}$, we thus obtain

$$\mathbf{E}\,\mathrm{waste}(\mathcal{S}) = O\Big(\, f_1(w_1) + f_2(w_2) \,\Big) = O\Big(\, \sqrt{Q} + Q^{1/3} \cdot R^{2/3} \,\Big) = O\Big(\, \sqrt{Q} \cdot (1 + R^{2/3}/Q^{1/6}) \,\Big),$$

which is just the bound stated in the corollary. □

### 6.1.4 Coupled tasks

**Corollary 6.9.** Let task processing times be coupled, with minimum $T_{\min}$ and variance $\sigma^2$, and let the overhead be $h \geq 1$. Then for all $n, p \in \mathbb{N}$, and for any $w \in \mathbb{N}$ that is within a constant factor of $\sqrt[3]{(h + \sigma^2) \cdot n \;/\; p^2}$ and for which $n/(pw) \in \mathbb{N}$, given $n$ tasks and $p$ processors, $\mathrm{FP}(x \mapsto w)$ produces a schedule $\mathcal{S}$ with

$$\mathbf{E}\,\mathrm{waste}(\mathcal{S}) = O\Big(\, (\, h + \sigma^2 \,)^{2/3} \cdot n^{2/3} \;/\; p^{1/3} \,\Big).$$

**Proof:** By Lemma 4.9, the expected average deviation of $\mathcal{S}$ with respect to

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

is bounded by $\varepsilon = \sigma^2$, so that according to Theorem 6.1

$$\mathbf{E}\,\mathrm{waste}(\mathcal{S}) = O\Big(\, (h + \varepsilon) \cdot n/p \;/\; w + \beta(w) \,\Big) = O\Big(\, (h + \sigma^2) \cdot n/p \;/\; w + p \cdot w^2 \,\Big).$$

The last term would obviously be minimized for $w$ equal to $\sqrt[3]{(h + \sigma^2) \cdot n \;/\; (2p^2)}$, and we easily verify that plugging in a term in that order yields

$$\mathbf{E}\,\mathrm{waste}(\mathcal{S}) = O\Big(\, ((h + \sigma^2) \cdot n/p)^{2/3} \cdot p^{1/3} \,\Big) = O\Big(\, (h + \sigma^2)^{2/3} \cdot n^{2/3} \;/\; p^{1/3} \,\Big).$$

□

## 6.2 Geometrically decreasing chunk sizes

In Section 4.1.2 we have learned that linear-width variance estimators of the type $[\,\mathrm{id}/A, B \cdot \mathrm{id}\,]$ are suited for a wide variety of settings. According to Theorem 3.1, the optimal algorithms pertaining to these kind of variance estimators are of the simple form $\mathrm{FP}(x \mapsto \lfloor x/C + w_{\min} \rfloor)$. We call such a scheduling algorithm *geometric*, because, at least when the rounding issue is neglected, the chunk size decreases by a fixed factor from each scheduling operation to the next; this follows by $(W - (W/(Cp) + w_{\min}))/(Cp) + w_{\min} = (W/(Cp) + w_{\min}) \cdot (1 - 1/(Cp))$. In Section 6.2.1, we show how our bounds from Section 3.2 can be improved for geometric algorithms; from this, we will also infer an "intuitive lower bound" on the wasted time that can be achieved by fixed-partition scheduling. In the following sections we then translate the upper bound to each of the bounded-tasks, independent-tasks, and coupled-tasks settings. In particular, it will be shown that in the independent-tasks setting geometric algorithms can achieve bounds that asymptotically match our intuitive lower bound. This is in sharp contrast with several previously existing scheduling heuristics of the fixed-partition type, which despite a considerably larger intricacy do not share this property; details on this will be provided in Chapter 7.

### 6.2.1 Generic bound

Our first step will be to strengthen the general bound of Theorem 3.1 for the special case of linear-width variance estimators. We here exploit that the number of chunks scheduled by a geometric scheme can be estimated more accurately (namely by a factor of 3) than for our general result in Section 3.2.

**Theorem 6.2.** Let task processing times be arbitrary, and let the overhead be $h \geq 1$. Let $\alpha = \mathrm{id}/A$, for some $A \geq 1$, let $\beta = B \cdot \mathrm{id}$, for some $B > 1$. 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, $\mathrm{FP}(x \mapsto \lfloor x/C + w_{\min} \rfloor)$ with $C = A \cdot B$ produces a schedule $\mathcal{S}$ with

$$\mathrm{waste}(\mathcal{S}) \leq C \cdot \left( h \cdot \left\lceil 1 + \ln \frac{n/p}{C w_{\min}} \right\rceil + w_{\min} \right) + h + \text{sum-early}_\alpha(\mathcal{S})/p + \text{max-late}_\beta(\mathcal{S}).$$

**Proof:** According to Theorem 3.1,

$$\mathrm{waste}(\mathcal{S}) \leq h \cdot \mathrm{chunks}(\mathcal{S})/p + h + \beta(w_{\min}) + \mathcal{E}/p,$$

where $\mathcal{E} = \text{sum-early}_\alpha(\mathcal{S}) + (p-1) \cdot \text{max-late}_\beta(\mathcal{S})$. To prove the theorem it therefore suffices to prove the strengthened (compared to Theorem 3.1) bound

$$\mathrm{chunks}(\mathcal{S}) \leq C \cdot p \cdot \left\lceil 1 + \ln \frac{n/p}{C w_{\min}} \right\rceil.$$

To this end, let $l = \mathrm{chunks}(\mathcal{S})$ and for $j = 1, \ldots, l$, denote by $W_j$ the amount of work unassigned just before the $j$th chunk is scheduled. Then $W_1 = n$, and for $j = 1, \ldots, l-1$,

$$W_{j+1} = W_j - \lfloor (W_j/p)/C + w_{\min} \rfloor \leq W_j \cdot (1 - 1/(Cp)) \leq n \cdot c^{-j},$$

where $c = 1/(1 - 1/(Cp))$. Since each chunk has size at least $w_{\min}$, except maybe the very last, this is easily seen to imply that the total number of chunks is bounded by

$$\left\lceil \ln_c \frac{n}{C p w_{\min}} \right\rceil + Cp.$$

It remains to check that for all $x > 0$, $e^{-1/x} \geq 1 - 1/x$, or equivalently, $\ln \frac{1}{1-1/x} \geq 1/x$, so that $1/\ln c \leq Cp$, which finally gives us

$$\text{chunks}(\mathcal{S}) \;\leq\; Cp \cdot \left\lceil \ln \frac{n/p}{Cw_{\min}} \right\rceil + Cp.$$

□

We next apply Theorem 6.2 to each of the bounded-tasks, independent-tasks, and coupled-tasks settings. Our results are summarized in Table 6.2 below, which, for the sake of comparison, also shows the respective bounds for the optimal fixed-size strategy, according to Corollaries 6.7, 6.8, and 6.9, as well as those inferred from our Main Theorem in Corollaries 4.1, 4.2, and 4.4.

| | independent tasks | bounded tasks | coupled tasks |
|---|---|---|---|
| FIX | $\sqrt{h+\sigma} \cdot \sqrt{n/p}$ | $\sqrt{h \cdot T_{\max} \cdot n/p}$ | $(\,h + \sigma^2\,)^{2/3} \cdot n^{2/3} \;/\; \sqrt[3]{p}$ |
| GEO | $h \cdot \log(n/p)$ | $h \cdot T_{\max}/T_{\min} \cdot \log(n/p)$ | $(h+\sigma^2) \cdot \sqrt{n \cdot \log(n/p)}$ |
| BAL | $(h+\sigma) \cdot \log\log(n/p)$ | $h \cdot T_{\max}/T_{\min} \cdot \log(n/p)$ | $(h+\sigma^2) \cdot \sqrt{n}$ |

TABLE 6.2: Asymptotic (expected) wasted times in three specific settings, for the respectively optimal instance of the fixed-size, geometric, and balancing strategy.

As an additional yardstick for the results derived in the following, let us define

$$L_{n,p,h} \overset{\text{def}}{=} h \cdot \lceil \ln(n/(ph)) \rceil + 3h,$$

which is just the minimal bound that can be obtained from Theorem 6.2 for $n$ tasks, $p$ processors, and overhead $h$, namely by having $C = 1$, $w_{\min} = h$, and no deviations. The corresponding strategy is $\text{FP}(x \mapsto \lfloor x + h \rfloor)$, which would indeed achieve this bound, given that the processing time of each chunk exactly equaled its size. In fact, Kuck and Polychronopoulus (1987) have shown that the almost identical scheme $\text{FP}(x \mapsto \lceil x \rceil)$, introduced in their paper as *guided self scheduling* (cf. Section 7.4), is optimal in such a setting under the additional assumption that there is initial imbalance (of processor starting times) which is not known a priori. It is therefore natural to assume that $L_{n,p,h}$ is the best wasted time that can be achieved by a fixed-partition algorithm for $n$ tasks, $p$ processors, and scheduling overhead $h$.

### 6.2.2 Bounded tasks

**Corollary 6.10.** Let task processing times be bounded in $[T_{\min}, T_{\max}]$, and let the scheduling overhead be $h \geq 1$. Then for all $n, p \in \mathbb{N}$, given $n$ tasks and $p$ processors, $\text{FP}(x \mapsto \lfloor x/C + h \rfloor)$

with $C = T_{\max}/T_{\min}$ produces a schedule $\mathcal{S}$ with

$$\mathbf{E}\,\text{waste}(\mathcal{S}) \;\leq\; T_{\max}/T_{\min} \cdot L_{n,p,h} \;=\; O\Big(\, h \cdot T_{\max}/T_{\min} \cdot \log(n/p) \,\Big).$$

**Proof:** This follows by a simple combination of Theorem 6.2 with (the obvious) Lemma 4.7. □

### 6.2.3 Independent tasks

Lemma 6.1 below exploits that in the independent-tasks setting, with respect to a suitably chosen linear-width variance estimator, the expected deviations of a sequence of chunks with geometrically decreasing sizes also decreases geometrically. This results in a sharper bound than what could be proven for the corresponding Lemma 4.8 in Section 4.4 for arbitrary schedules. Together with our strengthened (compared to Theorem 3.2) bound from Theorem 6.2, this will allow us to prove a bound on the wasted time that asymptotically matches our intuitive lower bound $L_{n,p,h}$.

**Lemma 6.1.** Let task processing times be independent, with variance $\sigma^2$, and let the overhead be $h \geq 1$. Then for all $C > 1$, there exist $A, B > 1$ such that for all $n, p, w_{\min} \in \mathbb{N}$, the schedule produced by $\text{FP}(x \mapsto \lfloor x/C + w_{\min} \rfloor)$ given $n$ tasks and $p$ processors satifies

$$\mathbf{E}[\,\text{sum-early}_\alpha(\mathcal{S}) + (p-1)\cdot\text{max-late}_\beta(\mathcal{S})\,] = O\left(\, p\cdot\sigma^2 \cdot \frac{(C + (Cp)^{1/3})^2}{C-1} \,\right),$$

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

**Proof:** Let $\mathcal{C}_1, \ldots, \mathcal{C}_l$ denote the sequence of chunks in $\mathcal{S}$ (sorted by size with the largest chunk first), and let $w_i$ and $T_i$ denote the size and processing time of $\mathcal{C}_i$, respectively. We first show that the chunk size at least halves every $\lceil Cp \rceil$ allocations. For a proof, verify that for all $i = 1, \ldots, l - \lceil Cp \rceil$, using that $w_1 \geq w_2 \geq \cdots \geq w_l$,

$$\begin{aligned}
w_{i+\lceil Cp \rceil} &= \lfloor\, W_{i+\lceil Cp \rceil}/(Cp) + w_{\min} \,\rfloor \\
&\leq \lfloor\, (W_i - \lceil Cp \rceil \cdot w_{i+\lceil Cp \rceil - 1})/(Cp) + w_{\min} \,\rfloor \\
&\leq \lfloor\, W_i/(Cp) + w_{\min} \,\rfloor - w_{i+\lceil Cp \rceil - 1} \\
&= w_i - w_{i+\lceil Cp \rceil - 1},
\end{aligned}$$

and since certainly $w_{i+\lceil Cp \rceil} \leq w_{i+\lceil Cp \rceil - 1}$, we have that

$$w_{i+\lceil Cp \rceil} \leq \min\{\, w_i - w_{i+\lceil Cp \rceil - 1} \,,\, w_{i+\lceil Cp \rceil - 1} \,\} \leq w_i/2.$$

Let us next bound the earliness of $\mathcal{S}$ with respect to $\text{id}/A$, for an arbitrary $A > 1$. To that end, take $a = 1 - 1/A$, so that

$$\text{early}_{\text{id}/A}(\mathcal{C}_i) = \max\{\, 0,\, w_i/A - T_i \,\} = \max\{\, 0,\, w_i - a w_i - T_i \,\},$$

and verify that $w_i - aw_i - T_i$ is a random variable with mean $-aw_i$ and standard deviation $\sigma\sqrt{w_i}$. Concerning this random variable, the parameters $t$ and $\eta$ of Lemma 4.6 are just $t_i = a/\sigma \cdot \sqrt{w_i}$ and $\eta_i = \vartheta \cdot e^{t_i^2/2} \cdot \varrho^3/\sigma^3 \;/\; \sqrt{w_i}$, where $\varrho^3$ is the third absolute central moment of a single task's processing time and $\vartheta > 0$ is the constant according to Lemma 4.6. Applying that lemma we thus obtain that, for $K = 1 + \vartheta \cdot \varrho^3/\sigma^3 \;/\; \sqrt{w_{\min}}$,

$$\begin{aligned} \mathbf{E}\,\text{early}_{\text{id}/A}(\mathcal{C}_i) &\leq (1+\eta_i)\cdot\sigma\sqrt{w_i}\cdot\frac{1}{\sqrt{2\pi}}\,\frac{1}{1+t_i^2}\,e^{-t_i^2/2} \\ &\leq K\cdot\sigma\sqrt{w_i}\cdot\frac{1}{\sqrt{2\pi}}\,\frac{1}{1+t_i^2} \\ &= K\cdot\sigma^2/a\cdot\frac{1}{\sqrt{2\pi}}\,\frac{1}{t_i+1/t_i}, \end{aligned}$$

and summation over all chunks gives us

$$\mathbf{E}\,\text{sum-early}_{\text{id}/A}(\mathcal{S}) \leq K\cdot\sigma^2/a\cdot\frac{1}{\sqrt{2\pi}}\sum_{i=1}^{l}\frac{1}{t_i+1/t_i}.$$

In order to bound the sum, observe that $t_{i+\lceil Cp\rceil} = a/\sigma\cdot\sqrt{w_{i+\lceil Cp\rceil}} \leq a/\sigma\cdot\sqrt{w_i/2} = t_i/\sqrt{2}$, that is, the sequence $t_1,\ldots,t_l$ decreases by a factor of at least $\sqrt{2}$ over $\lceil Cp\rceil$ elements. With $I_1 = \{i : t_i < 1\}$ and $I_2 = \{i : t_i \geq 1\}$, we hence obtain that

$$\sum_{i=1}^{l}\frac{1}{t_i+1/t_i} \leq \sum_{i\in I_1} t_i + \sum_{i\in I_2} t_i^{-1} \leq 2\cdot\lceil Cp\rceil\cdot\sum_{j=0}^{\infty}\sqrt{2}^{-j} = \frac{2\lceil Cp\rceil}{1-2^{-1/2}} \leq 7\,\lceil Cp\rceil.$$

This gives us the following bound on the expected total earliness

$$\mathbf{E}\,\text{sum-early}_{\text{id}/A}(\mathcal{S}) \leq 3K\cdot\sigma^2/a\cdot\lceil Cp\rceil.$$

Next we would like to bound the maximal lateness of a chunk with respect to $B\,\text{id}$, for an arbitrary $B > 1$. In analogy to the above, let $b = B - 1$ so that

$$\text{late}_{B\,\text{id}}(\mathcal{C}_i) = \max\{\,0,\,T_i - B\,w_i\,\} = \max\{\,0,\,T_i - w_i - b\,w_i\,\}.$$

By symmetry, calculations analogous to those used for the bound on $\mathbf{E}\,\text{sum-early}_{\text{id}/A}(\mathcal{S})$ would now show that

$$\mathbf{E}\,\text{max-late}_{B\,\text{id}}(\mathcal{S}) \leq \sum_{\mathcal{C}\in\mathcal{S}}\mathbf{E}\,\text{late}_{B\,\text{id}}(\mathcal{C}) \leq 3K\cdot\sigma^2/b\cdot\lceil Cp\rceil,$$

but it turns out that we can do better. Namely, in order to avoid the loss of exactness entailed by bounding the maximum of a set of relatively large reals by their sum, we instead use that for arbitrary fixed $M > 0$,

$$\text{max-late}_{B\,\text{id}}(\mathcal{S}) \leq M + \sum_{i=1}^{l}\max\{\,0,\,T_i - w_i - b\,w_i - M\,\}.$$

This would in fact be an equality if $M = \text{max-late}_{B\,\text{id}}(\mathcal{S})$, but note that $\text{max-late}_{B\,\text{id}}(\mathcal{S})$ is a random variable here. Introducing $t_i = b/\sigma \cdot \sqrt{w_i}$ and $Q$ with $M = \sigma^2/b \cdot Q = Q/t_i \cdot \sigma\sqrt{w_i}$, we get

$$\text{max-late}_{B\,\text{id}}(\mathcal{S}) \le \sigma^2/b \cdot Q + \sum_{i=1}^{l} \max\{\,0,\, T_i - w_i - (t_i + Q/t_i)\cdot\sigma\sqrt{w_i}\,\},$$

so that by an application of Lemma 4.6 with $\eta_i = \vartheta \cdot e^{t_i^2/2} \cdot \varrho^3/\sigma^3 \,/\, \sqrt{w_i}$, similar to the one in the previous paragraph,

$$\begin{aligned}
\mathbf{E}\,\text{max-late}_{B\,\text{id}}(\mathcal{S}) \;&\le\; \sigma^2/b\cdot Q \;+\; (1+\eta_i)\cdot\sigma\sqrt{w_i}\cdot\frac{1}{\sqrt{2\pi}}\sum_{i=1}^{l}\frac{1}{1+(t_i+Q/t_i)^2}\,e^{-(t_i+Q/t_i)^2/2}\\
&\le\; \sigma^2/b\cdot Q \;+\; K\cdot\sigma^2/b\cdot\frac{1}{\sqrt{2\pi}}\sum_{i=1}^{l}\frac{t_i}{(t_i+Q/t_i)^2},
\end{aligned}$$

with the same constant $K$ as above. Now letting $I_1 = \{i : t_i < \sqrt{Q}\}$ and $I_2 = \{i : t_i \ge \sqrt{Q}\}$, it is easily seen that

$$\sum_{i=1}^{l}\frac{t_i}{(t_i+Q/t_i)^2} \le \sum_{i=1}^{l}\frac{1}{t_i+Q^2/t_i^3} \le \sum_{i\in I_1} t_i^3/Q^2 + \sum_{i\in I_2} t_i^{-1},$$

with each of the last two sums being bounded by $Q^{-1/2}\cdot\lceil Cp\rceil\sum_{j=0}^{\infty}\sqrt{2}^{-j} \le Q^{-1/2}\cdot 3.5\cdot\lceil Cp\rceil$. We therefore have

$$\mathbf{E}\,\text{max-late}_{B\,\text{id}}(\mathcal{S}) \le 3K\cdot\sigma^2/b\cdot(Q + \lceil Cp\rceil\cdot Q^{-1/2}),$$

and by setting $Q = \lceil Cp\rceil^{2/3}$,

$$\mathbf{E}\,\text{max-late}_{B\,\text{id}}(\mathcal{S}) \le 6K\cdot\sigma^2/b\cdot\lceil Cp\rceil^{2/3}.$$

Altogether, we have thus shown that for arbitrary $A, B > 1$,

$$\mathbf{E}[\,\text{sum-early}_{\text{id}/A}(\mathcal{S}) \;+\; (p-1)\cdot\text{max-late}_{B\,\text{id}}(\mathcal{S})\,] \le 6K\cdot p\cdot\sigma^2\cdot\left(\frac{C}{a} + \frac{\lceil Cp\rceil^{2/3}}{b}\right),$$

where $a = 1 - 1/A$ and $b = B - 1$. A simple optimization now shows that under the constraints $A, B \ge 1$ and $A\cdot B = C$, the right hand side is minimal for

$$a = \frac{C-1}{C+\lceil Cp\rceil^{1/3}} \qquad\text{and}\qquad b = \frac{C-1}{1 + C/\lceil Cp\rceil^{1/3}}.$$

Plugging these into the bound above we obtain that

$$\begin{aligned}
&\mathbf{E}[\,\text{sum-early}_{\text{id}/A}(\mathcal{S}) \;+\; (p-1)\cdot\text{max-late}_{B\,\text{id}}(\mathcal{S})\,]\\
&\qquad\le 6K\cdot p\cdot\frac{\sigma^2}{C-1}\cdot\Big(C\cdot\big(C+\lceil Cp\rceil^{1/3}\big) \;+\; \lceil Cp\rceil^{2/3}\cdot\big(1 + C/\lceil Cp\rceil^{1/3}\big)\Big)\\
&\qquad= 6K\cdot p\cdot\frac{\sigma^2}{C-1}\cdot\big(C+\lceil Cp\rceil^{1/3}\big)^2.
\end{aligned}$$

□

**Corollary 6.11.** Let task processing times be independent, with variance $\sigma^2$, and let the overhead be $h \geq 1$. Then for all $C > 1$, and for all $n, p \in \mathbb{N}$, given $n$ tasks and $p$ processors, $\mathrm{FP}(x \mapsto \lfloor x/C + h \rfloor)$ produces a schedule $\mathcal{S}$ with

$$\mathbf{E}\,\mathrm{waste}(\mathcal{S}) \leq C \cdot L_{n,p,h} \; + \; O\left( \sigma^2 \cdot \frac{(C + (Cp)^{1/3})^2}{C - 1} \right).$$

In particular, for fixed $C > 1$, as $n \to \infty$,

$$\mathbf{E}\,\mathrm{waste}(\mathcal{S}) \;/\; L_{n,p,h} \to C,$$

while for $C = 1 + p^{1/3} \cdot \left(1 + L_{n,p,h}/\sigma^2\right)^{-1/2}$, as $n \to \infty$,

$$\mathbf{E}\,\mathrm{waste}(\mathcal{S}) \;/\; L_{n,p,h} \to 1.$$

**Proof:** By a combination of Theorem 6.2 with the previous lemma,

$$\mathbf{E}\,\mathrm{waste}(\mathcal{S}) \leq C \cdot \left( h \cdot \left\lceil 1 + \ln \frac{n/p}{Ch} \right\rceil + h \right) + h \; + O\left( \sigma^2 \cdot \frac{(C + (Cp)^{1/3})^2}{C - 1} \right),$$

which immediately implies the bound claimed in the corollary. Now for fixed $C > 1$, we can write

$$\mathbf{E}\,\mathrm{waste}(\mathcal{S}) = C \cdot L_{n,p,h} + O\Big( \sigma^2 p^{2/3} \Big),$$

while for $C = 1 + p^{1/3} \cdot \left(1 + L_{n,p,h}/\sigma^2\right)^{-1/2}$, we have

$$\mathbf{E}\,\mathrm{waste}(\mathcal{S}) = L_{n,p,h} + O\Big( \sigma^2 \cdot p^{2/3} \cdot (1 + L_{n,p,h}/\sigma^2)^{1/2} \Big).$$

These bounds immediately imply the desired convergences. □

### 6.2.4 Coupled tasks

While Lemma 4.9 bounds the average deviation of a schedule in the coupled-tasks setting with respect to a variance estimator of quadratic width, Lemma 6.2 below employs a linear-width variance estimator. Note that, unlike for the independent-tasks setting in the previous section, the lemma does not rely on special properties of a geometric schedule. Plugging Lemma 6.2 into the generic bound from Theorem 6.2, Corollary 6.12 below will infer a bound on the wasted time that is off the bound according to Corollary 4.2 by only a factor of $\sqrt{\log(n/p)}$.

**Lemma 6.2.** Let task processing times be coupled, with minimum $T_{\min}$ and variance $\sigma^2$, and let $B > 1$. 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

$$\mathbf{E}[\,\mathrm{sum\text{-}early}_\alpha(\mathcal{S}) + (p-1) \cdot \mathrm{max\text{-}late}_\beta(\mathcal{S})\,] = O\left( p \cdot \sigma^2 \cdot \frac{n}{B - 1} \right),$$

where $[\,\alpha, \beta\,] : w \mapsto [\,T_{\min} \cdot w, B \cdot w\,]$.

**Proof:** For $w \in \mathbb{N}$, let $\mathcal{C}$ be a chunk of an arbitrary but fixed selection of $w$ tasks and, just as in the proof of corresponding Lemma 4.9, observe that the processing time $T$ of $\mathcal{C}$ has mean $w$ and variance at most $\sigma^2 w^2$. Hence, by Lemma 4.5 applied with $\sigma_Z \leq \sigma w$ and $\mu_Z = -(B-1) \cdot w$,

$$\mathbf{E}\,\mathrm{late}_{B\,\mathrm{id}}(\mathcal{C}) = \mathbf{E}\max\{\,0, T - B \cdot w\,\} \leq \frac{\sigma^2 w^2}{(B-1)\cdot w} = \sigma^2 \cdot \frac{w}{B-1},$$

which immediately implies

$$\mathbf{E}\,\text{max-late}_{B\,\mathrm{id}}(\mathcal{S}) \leq \mathbf{E}\,\text{sum-late}_{B\,\mathrm{id}}(\mathcal{S}) \leq \sigma^2 \cdot \frac{n}{B-1}.$$

Since there is never earliness with respect to $\alpha$, this proves the lemma. $\square$

**Corollary 6.12.** Let task processing times be coupled, with minimum $T_{\min}$ and variance $\sigma^2$, and let the overhead be $h \geq 1$. Then for all $n, p \in \mathbb{N}$, given $n$ tasks and $p$ processors, $\mathrm{FP}(x \mapsto \lfloor x/C + h \rfloor)$ with $C = (1 + \sqrt{n/\log(n/p)})/T_{\min}$ produces a schedule $\mathcal{S}$ with

$$\mathbf{E}\,\mathrm{waste}(\mathcal{S}) = O\Big(\,(h + \sigma^2) \cdot \sqrt{n} \cdot \sqrt{\log(n/p)}\,/\,T_{\min}\,\Big).$$

**Proof:** By a combination of Theorem 6.2 with the previous lemma, for $A = 1/T_{\min}$ and $B = C \cdot T_{\min}$,

$$\mathbf{E}\,\mathrm{waste}(\mathcal{S}) = O\Big(\,C \cdot h \cdot \log(n/p) + n \cdot \sigma^2\,/\,(CT_{\min} - 1)\,\Big),$$

which for $C = (1 + \sqrt{n/\log(n/p)})/T_{\min}$ yields the desired bound. $\square$

# Chapter 7

# Previous Strategies

In the introduction, we have mentioned a variety of previously existing heuristics for our scheduling problem, all of which lack a rigorous mathematical analyis. In this chapter we prove, for all except one of these heuristics, either an upper or a lower bound on the achieved wasted time. In some cases, like for the FAC2 scheme of (Flynn *et al.*, 1992), our results nicely match the empirical findings of previous work. For other strategies, however, our analysis reveals flaws which apparently had not become evident by the respective experiments; this includes the (original) FAC scheme by (Flynn *et al.*, 1992) and the TAPER scheme due to (Lucco, 1992). Each of the following sections deals with one particular heuristic, first describing it, then proving an upper or a lower bound on its performance, and finally giving a brief informal conclusion on its significance. The schemes will be considered in chronological order, which is also the order by complexity (the most complex scheme coming last). For the intuition behind the design of the more sophisticated heuristics, we refer the reader to (Hagerup, 1997).

## 7.1 Static chunking

Static chunking (SC) is the obvious algorithm for scheduling similar tasks statically: it simply distributes the tasks evenly among the processors before the computation starts. Written in our notation from Section 3.2,

$$\mathrm{SC} \;=\; \mathrm{FP}(x \mapsto n/p),$$

where $n$ is the number of tasks and $p$ is the number of processors. Obviously, SC achieves a wasted time of $o(n/p)$ as $n/p \to \infty$ only for estimated processing time ranges of sublinear width, and according to our results from Chapter 6, its performance is significantly worse than that of an optimal fixed-size scheme, and much worse than that of an optimal geometric scheme (not to mention the optimal scheme). For the independent-tasks setting with variance $\sigma^2$, Lemma

5.9 implies that the wasted time incurred by SC for $n$ tasks and $p$ processors is at least

$$1/3 \cdot \sigma \cdot \sqrt{n/p}.$$

**Conclusion:** SC gives acceptable results only for a very low irregularity, and even then is far inferior to a simple dynamic algorithm. Therefore, its use is recommended only when an environment prohibits the implementation of dynamic scheduling schemes.

## 7.2 Self-scheduling

At the other end of the spectrum, we find the self scheduling (SS) strategy, that assigns merely one task at a time; formally,

$$\mathrm{SS} \;=\; \mathrm{FP}(x \mapsto 1).$$

Since SS performs as many scheduling operations as there are tasks, the wasted time for $n$ tasks, $p$ processors and overhead $h$ is always at least $h \cdot n/p$. In contrast to this, for any variance estimator, the bound of our Main Theorem is $o(n/p)$, as far as the asymptotic behaviour for $n \to \infty$ is concerned. On the other hand, it is easy to see that SS is a special instance of our balancing strategy, namely that which is optimal when, for all other parameters kept fixed, $h \to 0$.

**Conclusion:** SS is a meaningful scheme only when the overhead of a single scheduling operation is negligible compared to the typical processing time of a single task.

## 7.3 Fixed-size chunking

The fixed-size chunking (FSC) scheme, introduced by Kruskal and Weiss (1985), compromises between SC and SS by dynamically assigning chunks of a fixed intermediate size. Kruskal and Weiss investigated the independent-tasks setting, and proposed

$$\mathrm{FSC} \;=\; \mathrm{FP}\left( x \mapsto \left( \frac{\sqrt{2} \cdot h \cdot n/p}{\sigma\sqrt{\ln p}} \right)^{2/3} \right)$$

as the optimal fixed-size algorithm for $n$ tasks, $p$ processors, overhead $h$, and variance $\sigma^2$. According to our Corollary 6.8, however, this does not achieve the lowest possible wasted time unless $n$ is relatively small. We have extensively studied fixed-size schemes in Section 6.1. As one result from that section, such schemes (for an optimal choice of chunk size, as well as for the one advocated by Kruskal and Weiss) can achieve wasted times of $o(n/p)$ as $n/p \to \infty$ for every conceivable setting, which sets it apart from both SS and SC. On the other hand, our results from Section 6.2 demonstrate that the almost equally simple geometric algorithms outperform every fixed-size scheme by an order of magnitude. We have also seen that the optimal choice of a fixed chunk size for any particular setting is neither intuitive nor easy to compute, and there is no efficient default value; this is again in contrast with the geometric scheme $\mathrm{FP}(x \mapsto \lfloor x/C + h \rfloor)$, which Corollary 6.11 has shown to perform well for an arbitrary fixed $C > 1$.

**Conclusion:** Taking its simplicity into account, FSC is reasonable for the independent-tasks setting, but even then there is no reason to prefer it to an equally simple, but much more efficient geometric algorithm.

## 7.4 Guided self-scheduling

The *guided self scheduling* (GSS) scheme, introduced by Polychronopoulus and Kuck (1987), is defined as

$$\mathrm{GSS} \;=\; \mathrm{FP}(\, x \mapsto \lceil x \rceil \,);$$

which is , in fact, very similar to the geometric scheme $\mathrm{FP}(x \mapsto \lfloor x/C + h \rfloor)$ for $C = 1$. GSS was originally designed for a setting with completely regular and known task processing times, but unknown initial imbalance, and Polychronopoulus and Kuck claimed it to be optimal under these assumptions. As the following theorem says, GSS, as defined above, cannot be meaningfully applied even for the well-behaved independent-tasks setting, let alone for more irregular settings. It should be remarked, however, that already Polychronopoulus and Kuck conjectured that pretending a larger number of processors would turn GSS into a reasonable scheme for scheduling tasks with variable processing times. This is indeed confirmed by our Corollary 6.11, according to which $\mathrm{FP}(x \mapsto \lfloor x/C + h \rfloor)$ is a very efficient scheme for the independent-tasks setting for arbitrary fixed $C > 1$ (while for $C = 1$ it is subject to the same lower bound as GSS).

**Theorem 7.1.** Let task processing times be independent, with variance $\sigma^2$, and let the overhead be $h \geq 1$. Then for all $n, p \in \mathbb{N}$, given $n$ tasks and $p$ processors, GSS produces a schedule $\mathcal{S}$ with the property that

$$\mathbf{E}\,\mathrm{waste}(\mathcal{S}) \geq 1/3 \cdot \sigma \cdot \sqrt{n/p}.$$

**Proof:** Since the first chunk assigned by GSS is of size $\lceil n/p \rceil$, the stated bound is an immediate consequence of Lemma 5.9. □

**Conclusion:** GSS, in its original form, is not suited for any nonnegligible degree of irregularity.

## 7.5 Trapezoid self-scheduling

Tzen and Ni (1993) proposed the trapezoid self scheduling (TSS) strategy that assigns chunks of sizes linearly decreasing from a first size $f$ to a last size $l$. Tzen and Ni give no concrete guidance for the choice of $f$ and $l$ but advocated the use of $f = n/(2p)$, while assigning a total number of chunks between $2p$ and $4p$. For our analysis, we will actually assume (somewhat arbitrarily) that TSS assigns exactly $4p-1$ chunks such that the size of the $i$th chunk is $n/(2p)-(i-1)\cdot\Delta w$, where $\Delta w = n \;/\; (2p\,(4p-1))$. Under these assumptions, the following theorem says that TSS does not differ much from static chunking, in the sense that unless the variance of chunk processing times is very large, namely $\Omega(w^2)$ for chunks of size $w$, the imbalances of the schedules produced by TSS and SC, respectively, converge to the same distribution, as the number of tasks to be scheduled tends to infinity. In particular, this is hence the case for the independent-tasks setting, where processing time variance is just linear in the chunk size. In the coupled-tasks setting, on the contrary, chunk processing times may have variance quadratic in the chunk size, so that the condition to Theorem 7.2 is not fulfilled.

**Theorem 7.2.** Let task processing times be randomly distributed, and assume that for all $w \in \mathbb{N}$, the total processing time of an arbitrary selection of $w$ tasks has mean $w$ and variance $O(w^2/f(w))$, for some function $f : \mathbb{R}^+ \to \mathbb{R}^+$ with $f(w) \to \infty$ as $w \to \infty$. For $n, p \in \mathbb{N}$, let $F_n^{\mathrm{TSS}}$ and $F_n^{\mathrm{SC}}$ denote the distribution functions of the imbalances of the schedules produced by TSS and SC, respectively, given $n$ tasks and $p$ processors. Then, for $n \to \infty$,

$$\sup_x \left| \, F_n^{\mathrm{TSS}}(x) - F_n^{\mathrm{SC}}(x) \, \right| \to \, 0.$$

**Proof:** Let $w_j = n/(2p) - (i-1)\cdot\Delta w$, for $j = 1, \ldots, 4p-1$, and let us agree to call round one, two, three, and four, the period of time, where the first through $p$th, the $p+1$st through $2p$th, the $2p+1$st through $3p$th, and the $3p+1$ through $4p-1$st chunks, respectively, are assigned.

We first assume that the processing time of the $j$th chunk is exactly $w_j$. It is then easy to see that, with $T_k^{(i)}$ denoting the finishing time of the $k$th processor after the $i$th round,

$$\begin{aligned}
T_k^{(1)} &= n/(2p) \;-\; (k-1)\cdot\Delta w, \\
T_k^{(2)} &= n/p \;-\; (2p-1)\cdot\Delta w, \\
T_k^{(3)} &= 3n/(2p) \;-\; (4p+k-2)\cdot\Delta w, \\
T_k^{(4)} &= 2n/p \;-\; (8p-2)\cdot\Delta w = n/p,
\end{aligned}$$

for $k = 1, \ldots, p$; for ease of notation and without loss of generality we here assumed that for simultaneous processor request, the processor with the lower index is served first. It follows that after the second and last round processors finish simultaneously, and that the amount of work assigned to each processor is exactly $n/p$.

Let us now consider the setting of the theorem, where the chunk processing times $T_1, \ldots, T_{4p-1}$ are randomly distributed, and for all $j = 1, \ldots, 4p-1$, $\mathbf{var}\, T_j = O(w_j^2/f(w_j))$, for some function $f : \mathbb{R}^+ \to \mathbb{R}^+$ with $f(w) \to \infty$ as $w \to \infty$. Chebychev's inequality therefore implies that for $j = 1, \ldots, 4p-1$,

$$\Pr\left(\mid T_j - w_j \mid > \Delta w/8\right) = O\left(\frac{w_j^2/f(w_j)}{\Delta w^2}\right) = O\left(p^2/f(n/(8p^2))\right),$$

using that $\Delta w \le w_j \le n/p$ and $\Delta w \ge n/(8p^2)$. Now let $D$ denote the event that no processing time deviates from its mean by more than $\Delta w/8$. Then by the above, and since the number of chunks is bounded by $4p$, clearly $\Pr\left(\bar{D}\right) \to 0$ as $n \to \infty$. Besides, $D$ is easily seen to imply that each processor receives the same chunks as in the fixed-time setting considered in the previous paragraph. In particular then, each processor is assigned exactly $n/p$ work, and it follows that conditional on $D$, the distributions of $I_n^{\mathrm{TSS}}$ and $I_n^{\mathrm{SC}}$, denoting the imbalances incurred by TSS and SC respectively, are identical. Therefore

$$\begin{aligned}
&\left| \Pr\left(I_n^{\mathrm{TSS}} > x\right) - \Pr\left(I_n^{\mathrm{SC}} > x\right) \right| \\
&\quad = \Big| \Pr\left(I_n^{\mathrm{TSS}} > x \mid D\right) \Pr(D) + \Pr\left(I_n^{\mathrm{TSS}} > x \mid \bar{D}\right) \Pr\left(\bar{D}\right) \\
&\qquad\qquad - \Pr\left(I_n^{\mathrm{SC}} > x \mid D\right) \Pr(D) - \Pr\left(I_n^{\mathrm{SC}} > x \mid \bar{D}\right) \Pr\left(\bar{D}\right) \Big| \\
&\quad = \left| \Pr\left(I_n^{\mathrm{TSS}} > x \mid \bar{D}\right) - \Pr\left(I_n^{\mathrm{SC}} > x \mid \bar{D}\right) \right| \cdot \Pr\left(\bar{D}\right) \\
&\quad \le \Pr\left(\bar{D}\right),
\end{aligned}$$

and since $\Pr\left(\bar{D}\right) \to 0$ as $n \to \infty$, the theorem follows. □

**Conclusion:** Though a dynamic scheme, TSS is very similar to SC and should be preferred over the latter only for very irregular computations.

## 7.6 Factoring

Flynn *et al.* (1992) proposed so-called *factoring* strategies, which work in rounds of $p$ scheduling operations each such that all chunks assigned in the same round are of equal size, and this size decreases from one round to the next. Formally, for $\varrho : \mathbb{R}^+ \to \mathbb{N}$, we define FAC($\varrho$) as the (fixed-partition) scheme that in a round, at the beginning of which $W$ tasks are unassigned, assigns $p$ chunks of size $\varrho(W/p)$. Using this notation, the FAC scheme investigated in (Flynn *et al.*, 1992; Flynn and Hummel, 1992) can be written as $\mathrm{FAC}(x \mapsto \lceil \beta^{-1}(x) \rceil)$, where $\beta(w) = 2w + \sigma\sqrt{p/2} \cdot \sqrt{w}$, except for the first round, where the chunk size is chosen according to $\beta(w) = w + \sigma\sqrt{p/2} \cdot \sqrt{w}$. Note the similarity to the fixed-partition algorithm $\mathrm{FP}(x \mapsto \lfloor \beta^{-1}(x + \beta(h)) \rfloor$, which Theorem 3.2 suggests for a variance estimator $[\,\mathrm{id}, \beta\,]$. Flynn *et al.* (1992) also considered a simplified version FAC2, defined as

$$\mathrm{FAC2} = \mathrm{FAC}(x \mapsto \lceil x/2 \rceil);$$

this correspond to replacing $\beta$ by $w \mapsto 2w$ in the definition of FAC above. The following lemma says that for the independent-tasks setting, the simple FAC2 performs generally sound, while FAC performs poorly when the number of processors is small compared to the total number of tasks. The upper bound for FAC2 will be stated in terms of our intuitive lower bound $L_{n,p,h}$, defined in Section 6.2 as $h \cdot \lceil \ln(n/(ph)) \rceil + 3h$. According to Corollary 6.11, for the independent-tasks setting this bound can be asymptotically matched by a geometric scheme, which is significantly simpler than FAC.

**Theorem 7.3.** Let task processing times be independent, with variance $\sigma^2$, and let the overhead be $h \geq 1$. Then for all $n, p \in \mathbb{N}$, given $n$ tasks and $p$ processors, FAC and FAC2 produce schedules $\mathcal{S}_{\text{FAC}}$ and $\mathcal{S}_{\text{FAC2}}$, respectively, with

$$\mathbf{E}\,\text{waste}(\mathcal{S}_{\text{FAC2}}) \leq 1.45 \cdot L_{n,p,h} \;+\; O\Big(\,\sigma^2 \cdot p^{5/3}\,\Big),$$

while, if $p \leq C$ for some arbitrary fixed constant $C$,

$$\mathbf{E}\,\text{waste}(\mathcal{S}_{\text{FAC}}) \geq C_{\text{FAC}} \cdot \sigma\sqrt{n/p},$$

where $C_{\text{FAC}}$ depends only on $C$.

**Remark:** For example, for $p = 2$ and sufficiently large $n$, the last bound holds with $C_{\text{FAC}} = 1/14$.

**Proof:** The lower bound is an immediate consequence of Lemma 5.9, since for $p \leq C$, the size $w$ of the first chunk scheduled by FAC satisfies $w + \sqrt{C/2} \cdot \sigma\sqrt{w} \geq n/p$.

Proving the upper bound is essentially a matter of translating the proofs of Theorem 3.1 and Lemma 6.1 to schemes of the factoring type. To this end, let $Q = 2 - 1/p$, and consider $\varrho : \mathbb{R}^+ \to \mathbb{N}$ such that $\varrho$ is increasing and $\varrho(x) \leq x/Q$ for all $x > 0$. Let $w$ be the size of a chunk assigned by FAC($\varrho$) when some number $W$ of tasks are left. Then, by the definition of FAC($\varrho$) and by the assumption on $\varrho$, if the chunk is the $j$th in its round, $w = \varrho(W'/p) \leq W'/(Qp) = W'/(2p-1)$, where $W' = W + (j-1) \cdot w$ is the number of tasks that were left when the round began. Therefore

$$W \geq W' - (p-1) \cdot w \geq W' - \frac{p-1}{2p-1} \cdot W' = \frac{p}{2p-1} \cdot W' = W'/Q,$$

and thus $Q \cdot W/p \geq W'/p$, which proves that $w$ is no larger than the chunk size that FP($x \mapsto \lfloor \varrho(Q\,x) + 1 \rfloor$) would compute when $W$ tasks are left.

In particular, we have thus shown that the chunk assigned by FAC2 = FAC($x \mapsto \lceil x/2 \rceil$) when a certain number of tasks are left, is no larger than the chunk size that FP($x \mapsto \lfloor Q/2 \cdot x + 1 \rfloor$) would compute in the same situation. Since scheduling smaller chunks can only make the imbalance smaller, we hence obtain from Theorem 3.1 that with $C = 2/Q = 1 + 1/(2p-1)$, and for arbitrary $A, B > 1$ with $A \cdot B = C$,

$$\text{imbalance}(\mathcal{S}_{\text{FAC2}}) \leq p \cdot h + p \cdot C \;\; + \text{sum-early}_{\text{id}/A}(\mathcal{S}_{\text{FAC2}}) + (p-1) \cdot \text{max-late}_{B\,\text{id}}(\mathcal{S}_{\text{FAC2}}).$$

A calculation almost identical to that in the proof of Lemma 6.1 shows that

$$\mathbf{E}[\,\text{sum-early}_{\text{id}/A}(\mathcal{S}_{\text{FAC2}}) + (p-1) \cdot \text{max-late}_{B\,\text{id}}(\mathcal{S}_{\text{FAC2}})\,] = O\left(\; p \cdot \sigma^2 \cdot \frac{(C + (Cp)^{1/3})^2}{C-1}\;\right),$$

and hence, plugging in $C = 1 + 1/(2p-1)$,

$$\mathbf{E}\,\text{imbalance}(\mathcal{S}_{\text{FAC2}}) = p \cdot h \,+\, O\Big(\, p \cdot \sigma^2 \cdot p^{5/3} \,\Big).$$

Concerning the total number of chunks assigned, it is easily verified that

$$\text{chunks}(\mathcal{S}_{\text{FAC2}}) = p \cdot \Big(\, \lfloor \log_2(n/p) \rfloor + 1 \,\Big),$$

and we may finally conclude that

$$\mathbf{E}\,\text{waste}(\mathcal{S}_{\text{FAC2}}) \le h \cdot \log_2(n/p) + 2h + O\Big(\, \sigma^2 \cdot p^{5/3} \,\Big).$$

The bound claimed in the theorem follows since $L_{n,p,h} = h \cdot \lceil \ln(n/(ph)) \rceil + 3h$, and $1/\ln 2 \le 1.45$. □

**Conclusion:** The performance of FAC is poor when the number of processors is small compared to the total number of tasks, and always out of proportion to the complexity of the scheme. In contrast to this, the simple FAC2 is a provably sound scheme for the independent-tasks settings. As far as the factoring principle (batches of $p$ chunks of equal size) is concerned, its only effect appears to be an unnecessary increase of the total overhead by a factor of roughly 1.5.

## 7.7 Tapering

The TAPER strategy, invented by Lucco (1992), is very similar in spirit to the FAC scheme considered in the previous section. Namely, we can write TAPER as $\text{FP}(x \mapsto \lceil \beta^{-1}(x) \rceil)$, where $\beta : w \mapsto w + C \cdot \sqrt{w}$, for some constant $C > 0$ (in Lucco's paper, this constant is called $v_\alpha$). For the independent-tasks setting with variance $\sigma^2$, Lucco advocates the use of $C = 1.3\sigma$.

**Theorem 7.4.** Let task processing times be independent, with variance $\sigma^2$, and let the overhead be $h \ge 1$. Then for all $n, p \in \mathbb{N}$, given $n$ tasks and $p$ processors, TAPER produces a schedule $\mathcal{S}$ with the property that for some constant $C_{\text{TAPER}}$,

$$\text{waste}(\mathcal{S}) \ge C_{\text{TAPER}} \cdot \sigma \cdot \sqrt{n/p}.$$

**Remark:** For normal processing times and when $p$ is sufficiently large, the inequality holds with $C_{\text{TAPER}} = 0.0455 \ge 1/22$.

**Proof:** The size $w$ of the first chunk assigned by TAPER satisfies $w + 1.3\sigma \cdot \sqrt{w} \ge n/p$, and we can use Lemma 5.9 again. □

**Conclusion:** Similar to FAC, the complexity of TAPER is out of proportion to its efficiency, and it tends to perform poorly for a large number of tasks.

## 7.8 Bold

In contrast to all of the heuristics described in the previous sections, the Bold strategy of (Hagerup, 1997) is not of the fixed-partition type, but makes extensive use of the processing times of already processed chunks. Indeed, the author takes this approach to such an extreme that we would not be able to give a sound description of his scheme on less than a page here. As a consequence, Bold's formulas for the computation of chunk sizes are far more complex than those of Taper or FAC, and any kind of nontrivial analytical statement about the performance of Bold appears to be completely out of reach. Instead, the study of (Hagerup, 1997) is based on (synthetic) experiments in the independent-tasks setting, in which the Bold strategy is convincingly shown to outperform each of the heuristics described in the previous sections. Concerning mathematical analysis, the author conjectures that, in an independent-tasks setting, the excepted wasted time incurred by Bold for $n$ tasks is $O(\log\log n)$ as $n \to \infty$, and with all other parameters kept fixed. According to Corollary 4.2, this bound can indeed be achieved by an instance of our balancing strategy, and by Corollary 5.5 no other algorithm can improve on this by more than a constant factor.

**Conclusion:** The Bold scheme is extremely well-tuned to the independent-tasks setting, and outperforms all of the previously existing heuristics. Again, however, the achieved performance does not justify the outstanding complexity of the scheme.

Chapter 8

# Conclusion

In this thesis, we presented upper and lower bounds for the problem of scheduling a given number of tasks on a given number of processors with minimal makespan, that is, such that the completion time of the last task is minimized. We assumed that the tasks are assigned to the processors in chunks of several tasks at a time at the price of a fixed overhead time, which is independent of the number of tasks in the chunk. We also assumed that the processing times of the tasks, while similar, are not known in advance. The challenge then lied in minimizing the sum of the idle times of processors finishing early plus the sum of all overheads, a quantity that we called the wasted time of the schedule.

We motivated our investigations by a practical problem from the field of high-performance parallel computing, namely that of effectively scheduling the iterations of a parallel loop on a multiprocessor machine. A large number of heuristics had been proposed for this problem in the past, but hardly any rigorous analysis had been presented to date. We surveyed the large body of scheduling theory with respect to the characteristic features of the parallel-loop scheduling problem, finding only a single, very specialized result.

We took a generic approach to solving our scheduling problem, making use of two parameters: the variance estimator, consisting of two functions $\alpha, \beta : \mathbb{R}^+ \to \mathbb{R}^+$, which for each $w \in \mathbb{N}$ specify a range $[\,\alpha(w), \beta(w)\,]$ of estimated processing times for chunks of size $w$, and the deviation, a positive real quantity $\varepsilon$, which measures the deviation of the actual processing times from the estimated ranges, and which is not known until all the tasks have been processed. As we could show, there exists for all (meaningful) $\alpha, \beta : \mathbb{R}^+ \to \mathbb{R}^+$ an algorithm that, under the assumption that he average deviation of the chunk processing times from the corresponding ranges $[\,\alpha(w), \beta(w)\,]$ is bounded by $\varepsilon$, schedules $n$ tasks on $p$ processors with a wasted time bounded by

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

where $h$ is the overhead per chunk, and $\gamma \approx \mathrm{id}-\alpha\circ\beta^{-1}$ and $\gamma^*(n/p) = \min\{\, i \in \mathbb{N} : \gamma^{(i)}(n/p) \le 0 \,\}$.

We also proved a lower bound demonstrating that no algorithm can do significantly better than this.

On the way to proving this general result, we first considered a restricted class of scheduling algorithms, namely those whose scheduling decisions do not depend on information about the processing times of already completed chunks. We showed that such an algorithm can be optimal only if $\beta(w) - \alpha(w)$ grows at least linearly in $w$, that is, when we anticipate a relatively large irregularity in the processing times of the tasks. We then described the slightly more involved generic *balancing* strategy and proved that its instances can achieve the bound above for all conceivable parameter settings. We actually considered two variants of this strategy, where the first was more natural and also more practical, while the second, by specifically employing intermediate idling of processors, was more easily amenable to theoretical analysis.

We applied our general result to obtain specific bounds for a number of natural settings. These were the independent-tasks setting, where the task processing times are independent identically distributed random variables, the bounded-tasks setting, where each task has a known maximal and minimal processing time, and the coupled-tasks setting, where only groups of tasks have independent processing times, while the processing times within a group are strongly related.

With an eye towards implementation, we investigated in further depth two classes of particularly simple scheduling algorithms: those with a fixed size for each chunk, and those which decrease the chunk size by a fixed factor from one scheduling operation to the next; algorithms of the latter type were called *geometric*. Taking the same approach as for our general bounds, we obtained, for both of these classes, generic bounds, as well as specific bounds for each of the independent, bounded, and coupled-tasks setting. From these results it became evident that for every conceivable setting comitting to a fixed chunk-size is inevitably coupled with a significant efficiency loss compared to the corresponding optimal algorithm. In contrast to this, it turned out that for every reasonable setting, in particular for each of the aforementioned specific settings, a geometric algorithm with near-optimal performance exists. In view of its simplicity, this led us to recommend the scheduling of chunks of geometrically decreasing sizes as the method of choice for scheduling parallel loops.

We also provided a theoretical assessment of the previously existing parallel-loop scheduling heuristics. We found that, while each of these schemes has its particular merits and performs well for certain inputs, most of these also show a poor behaviour for certain other inputs, even when restricted to the special independent-tasks setting. Exempt from this criticism was the FAC2 scheme by Flynn *et al.* (1992), the performance of which comes close to that of an optimal geometric algorithm, and the BOLD scheme of Hagerup (1997), which is extremely well-tuned to the independent-tasks setting.

Throughout our work, we made a special effort to single out those results or techniques that appeared to have a significance beyond the particular context in which we used them. One such technique was our approach of modelling the concept of incomplete information about a real quantity deterministically, by estimated ranges and a deviation, instead of probabilistically, by a random variable. Taking this approach was probably the single largest step to solving the considered problem, and we would expect a similar success for related problems also. Another result we deemed of general interest was our master theorem for the * operator, which for a given function provides a convenient way to approximate the number of iterations required to get from some argument to another. We noted that such approximations are an integral part of the analysis of all kinds of algorithms.

Let us finally indicate directions for future research on scheduling problems of the type we have considered. Our investigations focussed on two characteristics of the problem: processing times that are unpredictable, and scheduling overheads that cannot be ignored. Of course, for many scheduling problems encountered in practice also other aspects might have a crucial effect on scheduling efficiency, so for example, processors running at different speeds, processors joining and leaving the system over time, or tasks having an affinity for certain processors on which they will hence be processed faster. As indicated in the introduction, various heuristics addressing one or more of these aspects already exist, but, just as was the case for our scheduling problem, very little is known about these problems theoretically.

# Chapter 9

# Zusammenfassung

Wir haben in dieser Arbeit obere und untere Schranken für das folgendes Problem bewiesen: gegeben eine Anzahl von Aufgaben (engl. *tasks*) sowie eine Anzahl von Prozessoren, weise die Aufgaben dynamisch so an die Prozessoren zu, dass die parallele Ausführungszeit, d.h. der Zeitpunkt zu dem die letzte Aufgabe beendet wird, minimiert wird. Wir haben dabei angenommen, dass die Zuweisung der Aufgaben an die Prozessoren in Stücken (engl. *chunks*) von mehreren Aufgaben auf einmal geschieht, wobei jede Zuweisungsoperation mit einer festen von der Stückgröße unabhängigen Wartezeit verbunden ist, und dass die Bearbeitungszeiten der Aufgaben zwar ähnlich sind, aber in unvorhersehbarer Weise schwanken können. Die Schwierigkeit lag dann darin, die Summe der Wartezeiten der Prozessoren, die vor dem letzten Prozessor fertig werden, plus die Summe der mit den Zuweisungsoperationen verbundenen Wartezeiten zu minimieren, eine Größe die wir die verschwendete Zeit (engl. *wasted time*) genannt haben.

Wir haben diese Problemstellung durch ein praktisches Problem aus dem Gebiet des parallelen Hochgeschwindigkeitsrechnens motiviert, bei dem es darum geht, eine Schleife mit voneinander unabhängigen Iterationen schnellstmöglich auf einem Parallelrechner auszuführen. Während zahlreiche Heuristiken für dieses klassische Problem bekannt und auch im Einsatz sind, gab es bislang so gut wie keine theoretischen Resultate. Wie wir gezeigt haben, wird in der Tat keiner der bekannten Ansätze aus der Scheduling-Theorie den Besonderheiten dieses Problems gerecht.

Unsere Ergebnisse basieren auf einem generischen Ansatz, der auf den folgenden zwei Parametern beruht: zum einen eine Abschätzung der Varianz (engl. *variance estimator*), die durch zwei Fuktionen $\alpha, \beta : \mathbb{R}^+ \to \mathbb{R}^+$ gegeben ist, die für jedes $w \in \mathbb{N}$ einen geschätzten Bereich $[\,\alpha(w), \beta(w)\,]$ für die Bearbeitungzeiten von Stücken der Größe $w$, d.h. die aus $w$ Aufgaben bestehen, angeben; zum anderen eine Abweichung (engl. *deviation*), eine positive reelle Größe $\varepsilon$, die ein Maß dafür ist um wieviel die tatsächlichen Bearbeitungszeiten von den geschätzten Bereichen abweichen, und die daher erst im Nachhinein bekannt ist. Wie wir zeigen konnten, gibt es für alle (sinnvollen) $\alpha, \beta : \mathbb{R}^+ \to \mathbb{R}^+$ einen Algorithmus, der, unter der Bedingung dass die durchschnittliche Bearbeitungszeit eines Stücks um höchstens $\varepsilon$ von dem seiner Größe entsprechenden Bereich $[\,\alpha(w), \beta(w)\,]$ abweicht, $n$ Aufgaben so an die $p$ Prozessoren zuweist, dass die verschwendete Zeit höchstens

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

ist, wobei $h$ die Wartezeit pro Zuweisung bezeichnet, und $\gamma \approx \mathrm{id} - \alpha \circ \beta^{-1}$ und $\gamma^*(n/p) = \min\{\,i \in \mathbb{N} : \gamma^{(i)}(n/p) \leq 0\,\}$. Mit einer dazugehörigen unteren Schranke konnten wir auch beweisen, dass kein Algorithmus wesentlich bessere Resultate erzielen kann.

Zum Beweis dieses sehr allgemeinen Ergebnisses haben wir zuerst eine beschränkte Klasse von Algorithmen betrachtet, und zwar jene, die ihre Entscheidungen nicht von den Bearbeitungszeiten bereits abgeschlossener Stücke abhängig machen. Wie wir gezeigt haben, können solche Algorithmen nur dann optimal sein wenn $\beta(w) - \alpha(w)$ wenigsten linear in $w$ wächst, was einer relativ großen Unregelmäßigkeit in den Bearbeitungszeiten der Aufgaben entspricht. Für den allgemeinen Fall haben wir das generische *balancing* Verfahren vorgestellt und bewiesen, dass es für jede (sinnvolle) Belegung unserer Parameter eine optimal Instanz liefert. Genau genommen haben wir zwei Varianten dieses Verfahrens betrachtet, wobei die erste die natürlichere und auch praktikablere war, während die zweite durch das gezielte Einsetzen von zusätzlichen Wartezeiten zwar etwas artifiziell, aber dafür leichter zu analysieren war.

Mit Hilfe unseres allgemeinen Ergebnisses konnten wir leicht Resultate für mehrere natürliche Modelle zeigen. Darunter war zum Beispiel jenes, in dem die Bearbeitungszeiten der Aufgaben identisch verteilte, unabhängige Zufallsvariablen sind, sowie jenes, in dem alle Aufgaben eine bekannte minimale und maximale Berarbeitungszeit haben, und jenes, wo nur Gruppen von Aufgaben unabhängig voneinander sind, während die Bearbeitungszeiten von Aufgaben aus derselben Gruppe stark korreliert sind.

Mit Hinblick auf eine tatsächliche Implementierung, haben wir zwei Klassen von besonders einfachen Algorithmen genauer untersucht: solche, die alle Stücke von der gleichen festen Größe wählen, und solche, die die Stückgröße von einer Zuweisung zur nächsten um einen festen konstanten Faktor reduzieren; Algorithmen mit letzterer Eigenschaft haben wir *geometrisch* genannt. In der selben Art wie für unser Hauptergebnis, haben wir für beide Klassen zuerst eine allgemeine, generische Schranke bewiesen, und aus dieser dann spezielle Schranken für die oben genannten Modelle hergeleitet. Aus diesen Ergebnissen wurde deutlich, dass die Festlegung auf eine fixe Stückgröße unter allen Umständen mit einem erheblichen Effizienzverlust verbunden ist. Im Gegensatz dazu gibt es unter fast allen Umständen, insbesondere unter denen der oben genannten speziellen Modelle, einen geometrische Algorithmus mit fast optimaler paralleler Ausführungszeit. In Anbetracht der Einfachheit der geometrischen Verfahren, haben wir diese daher für den praktischen Gebrauch besonders empfohlen.

Schließlich haben wir auch die vorgenannten existierenden Heuristiken theoretisch untersucht. Dabei kamen wir zu dem Schluß, dass die meisten dieser Heuristiken nicht einmal unter den

Umständen, für die sie ursprünglich konzipiert wurden, generell gute Ergebnisse liefern, geschweige denn für allgemeinere Modelle. Ausgenommen von dieser Kritik waren das FAC2 Verfahren aus (Flynn *et al.*., 1992), das fast genau so gut wie ein optimales geometrisches Verfahren ist, sowie BOLD (Hagerup, 1997), das genau auf das Modell der unabhängigen Bearbeitungszeiten zugeschnitten ist.

Generell haben wir uns besonders bemüht, solche Ergebnisse und Techniken herauszustellen, die über den speziellen Kontext, in dem sie in dieser Arbeit verwendet wurden, hinaus von Interesse schienen. Eine solche Technik war unser Ansatz, unvollständige Information über eine reelle Größe deterministisch, durch einen geschätzten Bereich und eine Abweichung, zu modellieren, an Stelle von probabilistisch, durch eine Zufallsvariable. Die Wahl und Formulierung dieses Ansatzes war wohl der größte einzelne Schritt zur Lösung des betrachteten Problems, und wir würden einen ähnlichen Erfolg auch für verwandte Problemstellungen erwarten. Ein anderes Resultat von wohl allgemeinerem Interesse ist unser Hauptsatz für den * Operator. Für eine gegebene Funktion liefert dieser Satz Abschätzungen für die Anzahl der Iterationen, die benötigt werden um von einem gegebenen Argument zu einem anderem zu gelangen. Wie wir angemerkt haben, werden Abschätzungen dieser Art in der Analyse aller möglichen Algorithmen benötigt.

# Bibliography

[1] Banicescu, I. (1996), Load balancing and data locality in the parallelization of the fast multipole algorithm, Ph.D. thesis, Polytechnic University, Department of Computer Science.

[2] Banicescu, I., and Hummel, S. F. (1995), Balancing processor loads and exploiting data locality in $N$-body simulations, *in* Proceedings Supercomputing (SC'95).

[3] Bast, H. (1998), Dynamic scheduling with incomplete information, *in* Proceedings Symposium on Parallel Algorithms and Architectures (SPAA'98), pp. 182–191.

[4] Bull, J. M. (1998), Feedback guided dynamic loop scheduling: algorithms and experiments, *in* Proceedings European Conference on Parallel Computing (EURO-PAR'98), Springer Lecture Notes in Computer Science **1470**, pp. 377–382.

[5] Coffman, E. G. (1976), Computer and job-shop scheduling theory, John Wiley and Sons, New York.

[6] Durand, M. D., Jalby, W., Kervella, L., Montaut, T. (1996), Impact of memory contention on dynamic scheduling on NUMA Multiprocessors, *IEEE Transactions on Parallel and Distributed Systems*, **11**, pp. 1201–1214.

[7] Eager, D. L., and Subramaniam, S. (1994), Affinity scheduling of unbalanced workloads, *in* Proceedings Supercomputing (SC'94), pp. 214–226.

[8] 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.

[9] Fahringer, T., and Zima, H. (1993), A static parameter based performance prediction tool for parallel programs, *in* Proceedings International Conference on Supercomputing (ICS'93), pp. 207–219.

[10] Fang, Z. X., Tang, P. Y., Yew, P. C., and Zhu, C. Q (1990), Dynamic processor self-scheduling for general parallel nested loops, *in* IEEE Transactions on Computers **39**:7, pp. 919-929.

[11] Flynn, L. E., and Hummel, S. F. (1992), A mathematical analysis of scheduling parallel loops in decreasing-size chunks, manuscript. A preliminary version is available as IBM Research Report No. RC 18462.

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

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

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

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

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

[17] Hagerup, T. (1996), Allocating Independent Tasks to Parallel Processors: An Experimental Study, *in* Proceedings Parallel Algorithms for Irregularly Structured Problems (IRREGULAR 1996), Springer Lecture Notes in Computer Science, **1117**, pp. 1–33.

[18] Hagerup, T. (1997), Allocating Independent Tasks to Parallel Processors: An Experimental Study, *Journal of Parallel and Distributed Computing* (JPDC) **47**, pp. 185–197.

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

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

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

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

[23] Karlin, A., Manasse, M., Rudolph, L., and Sleator, D. D. (1988), Competitive snoopy caching, *Algorithmica* **3**, pp. 79–119.

[24] Kleinberg, J., Rabani, Y., and Tardos, E. (1997), Allocating bandwidth for bursty connections, *in* Proceedings Symposium on the Theory of Computing (STOC'97), pp. 664–673.

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

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

[27] 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.

[28] 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, pp. 589–616.

[29] 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.

[30] 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.

[31] 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, pp. 379–400.

[32] Orlando, S., and Perego, R. (1997), A support for non-uniform parallel loops and its application to a flame simulation code, *in* Proceedings International Symposium on Solving

Irregularly Structured Problems in Parallel (IRREGULAR'97), Springer Lecture Notes in Computer Science **1253**, pp. 186–197.

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

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

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

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

[37] Polychronopoulos, C. D. (1987), Loop coalescing: a compiler transformation for parallel machines, *in* Proceedings International Conference on Parallel Processing (ICPP'87), pp. 235–242.

[38] Polychronopoulos, C. D. (1988), Parallel Programming and Compilers, Kluwer Academic Publishers, Boston.

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

[40] 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.

[41] Rudolph, L., Slivkin-Allalouf, M., and Upfal, E. (1991), A simple load balancing scheme for task allocation in parallel machines, *in* Proceedings Symposium on Parallel Algorithms and Architectures (SPAA'91), pp. 237–245.

[42] Sarkar, V. (1989), Determining average program execution times and their variance, *in* Proceedings Conference on Programming Language Design and Implementation (PLDI'89), pp. 298–312.

[43] Sgall, J. (1998), On-line scheduling—a survey, in *Online Algorithms: The State of the Art*, Springer Lecture Notes in Computer Science **1442**, pp. 196–231.

[44] Sleator, D., and Tarjan, R. (1985), Amortized efficiency of list update and paging rules, *Communications of the ACM* **28**:2, pp. 202–208.

[45] Smith, B. (1981), Architecture and applications of the HEP multiprocessor computer system, *in* Proceedings SPIE Symposium (Real Time Processing IV), pp. 241–248.

[46] 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.

[47] Tang, P., Yew, P. C., and Zhu, C. Q (1988), Impact of self-scheduling order on performance of multiprocessor systems, *in* Proceedings International Conference on Supercomputing (ICS'88), pp. 593–603.

[48] Tzen, T. H., and Ni, L. M. (1991), Dynamic loop scheduling for shared-memory multiprocessors, *in* Proceedings International Conference on Parallel Processing (ICPP'91), pp. II 247–II 250.

[49] 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, pp. 87–98.

[50] Wolfe, M. (1996) High Performance Compilers for Parallel Computing, Addison-Wesley, Amsterdam/Bonn/Sidney.

[51] 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, pp. 70–81.

# Curriculum Vitae

| | |
|---|---|
| **Name** | Hannah Bast |
| **Date of Birth** | April 22, 1970 |
| **Place of Birth** | Hanau, Germany |

## Education

| | |
|---|---|
| **1995 - 1999** | Ph.D. student, Max-Planck-Institut für Informatik, Saarbrücken, Germany. |
| **1994** | Diplom ($\sim$ Master's degree) in computer sciene, Universität des Saarlandes, Germany. |
| **1990** | Vordiplom ($\sim$ Bachelor's degree) in computer science, Universität des Saarlandes, Germany. |
| **1990** | Vordiplom ($\sim$ Bachelor's degree) in mathematics, Universität des Saarlandes, Germany. |
| **1988** | Abitur ($\sim$ Highschool diploma), European School of Luxembourg, Luxembourg. |

## Work and Teaching Experience

| | |
|---|---|
| **1989 - 1994** | Teaching assistant for lectures "Praxis des Programmierens", "Informatik I", "Informatik II", "Höhere Numerik", "Praktische Mathematik", "Komplexitätstheorie", "Parallele Algorithmen", "Parallele Algorithmen mit Sublogarithmischer Laufzeit", Universität des Saarlandes. |
| **1989 - 1991** | Werkstudentin at Siemens AG, München, Germany. |

## Publications

1. Bast, H. (1998), Dynamic Scheduling with Incomplete Information, *in* Proceedings 10th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA'98), pp. 182–191.

2. Bast, H., and Hagerup, T. (1995), Fast Parallel Space Allocation, Estimation and Integer Sorting, *Information and Computation* **123**, pp. 72–110.

3. Bast, H., Dietzfelbinger, M., und Hagerup, T. (1992), A Perfect Parallel Dictionary, *in* Proceedings 17th International Symposium on Mathematical Foundations of Computer Science (MFCS'92), Springer Lecture Notes in Computer Science, **629**, pp. 133–141.

4. Bast, H., und Hagerup, T. (1991), Fast and Reliable Parallel Hashing, *in* Proceedings 3rd Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA'91), pp. 50–61.