Alcom-FT Technical Report Series
ALCOMFT-TR-02-36

Scheduling at Twilight the Easy Way*

Hannah Bast

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

Abstract. We investigate particularly simple algorithms for optimizing the tradeoff between load imbalance and assignment overheads in dynamic multiprocessor scheduling scenarios, when the information that is available about the processing time of a task before it is completed is vague. We describe a simple and elegant generic algorithm that, in a very general model, always comes surprisingly close to the theoretical optimum, and the performance of which we can analyze exactly with respect to constant factors. In contrast, we prove that algorithms that assign tasks in equal-sized portions perform far from optimal in general. In fact, we give evidence that the performance of our generic scheme cannot be improved by any constant factor without sacrificing the simplicity of the algorithm. We also give lower bounds on the performance of the various decreasing-size heuristics that have typically been used so far in concrete applications.

1 Introduction

Many scheduling problems fit into the following abstract scenario: n tasks of similar complexity are to be scheduled in arbitrary order on p identical processors such that the completion time of the last task, the so-called makespan of the schedule, becomes as small as possible. Assignments may be at runtime, in *chunks* of several tasks at a time, and for each such assignment an overhead is incurred, at least part of which is independent of the size of the chunk, that is, the number of tasks in it. The larger these chunks are chosen, the fewer assignments are necessary, and the lower the total assignment overhead will be. On the other hand, only vague information on the processing times of the tasks is available in

* Partially supported by the Future and Emerging Technologies programme of the EU under contract number IST-1999-14186 (ALCOM-FT).

advance, and the larger a chunk, the harder it is to predict its processing time. To avoid that some processors finish long before others, chunks should therefore be as small as possible, especially towards the end of the scheduling process. It is not hard to see (and is explained in Section 2) that the optimal makespan is achieved by minimizing the *total wasted time* of the schedule, which is the sum of all the overheads plus the idle times of those processors waiting for the last processor to finish.

Previous work concerning this basic scheduling problem can be roughly classified as follows: on the one hand, a large number of heuristics have been designed and investigated [7][12][4][9][13][5], on the other hand, many extensions of the basic scenario and also real systems have been developed, all of which rely on a good heuristic for the basic problem [8][3][10][6][11]. It is only relatively recently, however, that a rigorous theoretical study was presented [1], which, in a sense, does not leave much to be desired: in a general model, capturing all kinds of vague-information scenarios, matching upper and lower bounds are presented, together with a generic strategy achieving these bounds. This result is stated and explained in Section 2.

This paper puts emphasis on two more practical aspects, which played only a secondary role in [1]: the simplicity of the scheduling algorithm, and the constant factors in the performance bounds. Concerning simplicity, it is a matter of fact that most existing heuristics are quite elaborate, while, as far as we know, only the two simplest ones are actually in use: the *fixed-size* (FIX) heuristics, where the chunk size is kept fixed throughout the scheduling process, and what we call *geometric* (GEO) heuristics, where chunk sizes decrease at an exponential rate. It is exactly these two heuristics that we investigate in this paper.

The remainder of this introduction gives an account of our results. In short summary, our results suggest that, for any scenario within our framework, a suitably chosen geometric heuristic yields a hardly beatable combination of simplicity and efficiency, while, in comparison, the efficieny of any fixed-size heuristic is always off by an order of magnitude. Another, maybe important, aspect of this paper is that it provides a fast and easy access to the main result of the extensive study undertaken in [1].

We give our analysis in the general model of [1], and from this derive bounds for four specific settings: (1) the *independent-tasks* setting, where the processing times of the tasks are independent, identically distributed random variables; (2) the *bounded-ranges* setting, where the processing times of all tasks are inside of a given range $[T_{\min}, T_{\max}]$; (3) the *exponential-tails* setting, where for the processing time T of a single task, $T \geq T_{\min}$ and $\Pr(T \geq t) \sim e^{-t/\lambda}$, for some $T_{\min}, \lambda > 0$ (common for all tasks), but the processing times of different task may arbitrarily depend on each other; (4) the *heavy-tails* setting, where for the processing time T_w of a chunk of size w, $T \geq w \cdot T_{\min}$ and $\Pr(T_w \geq t) \sim (t/w)^{-\kappa}$, for some $T_{\min}, \kappa > 0$, and, again, dependencies may be arbitrary. The general model, and how it covers each of these settings, will be explained in Section 2.

In Section 3, we investigate the generic algorithm $\text{GEO}(C, w_{\min})$, which, when W tasks are left, assigns a chunk of $\lceil W/(Cp) + w_{\min} \rceil$ tasks to the next available

Table 1. Asymptotic wasted times that are achieved by the optimal algorithm, by an optimal geometric algorithm, and by an optimal fixed-size algorithm, respectively, for a variety of settings, where n is the number of tasks ($\kappa = 2$ for the heavy-tails setting here)

	Independent	Bounded	Exponential	Heavy-tails
OPT	$\log\log n$	$\log n$	$\log^2 n$	$\sqrt{n}$
GEO	$\log n$	$\log n$	$\log^2 n$	$\sqrt{n}\log n$
FIX	$\sqrt{n}$	$\sqrt{n}$	$\sqrt{n\log n}$	$n^{2/3}$

processor. As is easy to see, instances of this scheme have the nice property that, without the rounding, the chunk sizes exactly form a geometric series (as our analysis will show, rounding is not a serious issue in the context of this scheduling problem). We prove that for every setting of the parameters of the general model of [1], there is an instance of GEO, whose wasted time is within a logarithmic factor of the theoretical optimum; see Table 1. Moreover, our analysis is exact with respect to constant factors, which, from a practical perspective, somewhat compensates for the asymptotic deviation from the theoretical optimum. It should be clear at this point, that our actual objective function is the makespan, which, as explained in the next section, is one pth of the sum of the total processing time and the total wasted time. The magnitude of the wasted time should therefore always be seen in relation to the total processing time, which, by our choice of time scale, is always about n.

As a by-product of our analysis of GEO, we also obtain a tight bound for the performance of the popular FAC2 scheme of [4], which essentially equals GEO$(2,0)$ except that it recomputes the chunk size only every pth chunk (keeping the last computed size for the intermediate chunks); this principle is called *factoring* in [4]. Our bound on the wasted time of FAC2 is a factor of $1/\ln 2 \approx 1.44$ off what can be achieved by a GEO heuristic, which suggests that the factoring principle is not meaningful regarding efficiency.

In Section 4, we investigate FIX(w), which, whenever a processor is available, assigns a chunk of w tasks to it (possibly less for the very last chunk). We prove that even for an optimal choice of w, the deviation of the wasted time from the optimum is always by an order of magnitude larger for FIX than for a suitable GEO scheme; in particular, FIX(w) is *never* optimal; again see Table 1. This adds to the known disadvantages of the fixed-size heuristic: that it is hard to compute the optimal chunk size for a given distribution of processing times, and that no natural default setting with a generally reasonable performance exists.

In Section 5, we give lower bounds for the GSS [12], FAC [4], and TAPER [9] heuristics, all of which are variants of GEO that have actually been used in concrete applications. We show that if the processing times of the tasks are independent, identically distributed random variables (which were indeed the assumptions underlying the design of these heuristics), for each of these heuristics the expected difference of the makespan from the optimum is on the order of $\sqrt{n/p}$. Asymptotically, this is as bad as the performance of a static scheme which simply assigns a single chunk of size n/p (suitably rounded) to each processor.

2 Preliminaries

We first make our scheduling scenario precise. Given are n tasks, ordered in a queue, to be processed on p identical 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, at least part of which 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. The processing time of a chunk is not known in advance, and 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 has to be run to completion on that processor (nonpreemptive scheduling).

A *scheduling algorithm* is an 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. However, all algorithms considered in this paper ignore this information, so that for them the partitioning of the tasks into chunks is independent of the task processing times.

An assignment of a collection of chunks to a set of processors will be called a *schedule*. We will often have to sum over all chunks $\mathcal{C}$ in a schedule $\mathcal{S}$, for which we will use the notation $\sum_{\mathcal{C}\in\mathcal{S}}$. The following definition names some characteristic properties of a schedule, which Figure 1 illustrates by an example.

Definition 1. *For a schedule $\mathcal{S}$ on p processors, let l_k be the number of chunks assigned to the kth processor, let h_k be the sum of their overheads, and let T_k be the sum of their processing times. Then define*

$$\begin{aligned}
\mathrm{chunks}(\mathcal{S}) &= \sum\nolimits_{k=1}^{p} l_k, \\
\mathrm{imbalance}(\mathcal{S}) &= \sum\nolimits_{k=1}^{p} \big(\max\{h_1+T_1,\ldots,h_p+T_p\} - (h_k+T_k)\big), \\
\mathrm{waste}(\mathcal{S}) &= \big(\sum\nolimits_{k=1}^{p} h_k + \mathrm{imbalance}(\mathcal{S})\big)/p.
\end{aligned}$$

The last quantity is called the average wasted time *of $\mathcal{S}$; the (total) wasted time is just p times this quantity. In the context of a schedule $\mathcal{S}$, the* average overhead per chunk *is the quantity $\sum_{k=1}^{p} h_k/\mathrm{chunks}(\mathcal{S})$.*

From the definition above, or more easily from Figure 1, it is straightforward to deduce that p times the makespan of a schedule is just its total wasted time

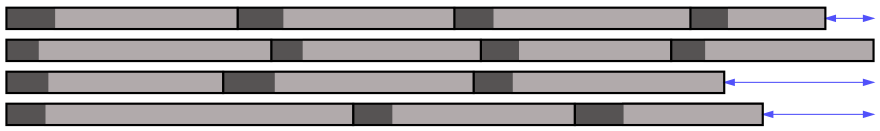

Fig. 1. A schedule on four processors and with fourteen chunks. Overheads are drawn in dark grey, processing times in light grey. The sum of the lengths of the arrows is the imbalance

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 (total or average) as possible.

Following [1], we model the unpredictability of processing times by two parameters: the *variance estimator*, which represents an algorithm's a priori estimate of the processing times of chunks, and the *deviation*, which measures the deviation of the actual processing times from these estimates (and is not known in advance).

Definition 2. *For strictly increasing and bijective $\alpha, \beta : \mathbb{R}^+ \to \mathbb{R}^+$ such that for all $w > 0$, $w/2 \le \alpha(w) \le w \le \beta(w)$, and such that either both $\beta(w)/w$ and $w/\alpha(w)$ are nonincreasing in w, or $\beta(w)/w$ is nondecreasing and $w/\alpha(w)$ is constant,*[1] *the function*

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

is called a variance estimator. *With respect to a variance estimator $[\alpha, \beta]$, 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 $[\alpha, \beta]$. In the context of a variance estimator $[\alpha, \beta]$ and a schedule $\mathcal{S}$, the* average deviation per chunk *is the quantity $\sum_{\mathcal{C} \in \mathcal{S}} \text{dev}_{\alpha,\beta}(\mathcal{C}) / \text{chunks}(\mathcal{S})$.*

For the fine points of this definition, we refer the reader to [1]. In particular, it is clarified there why the monotonicity conditions are natural for a variance estimator, that the condition $\alpha(w) \le w \le \beta(w)$ is just a convention regarding the time scale, and that the 2 may be replaced by an arbitrary larger constant.

In the terminology thus introduced, we now state the Main Theorem of [1]. As the intuition behind the bound is not quickly explained, we refer the reader

[1] The conditions on α and β in [1] are slightly weaker, but we do not need this generality here.

to Section 2.3 of [1], dedicated to exactly this task. The first row of Table 1 gives the asymptotic behaviour in n for a selection of variance estimators. The * operator used below is defined as $f^*(x) = \min\{i : f^{(i)}(x) \leq 0\}$, i.e., it "counts" how often a function has to be applied to reduce its argument to zero or below (always a finite number of times in the following).

Theorem 1 ([1]). *For every variance estimator $[\alpha, \beta]$, there is an algorithm that for all $n, p \in \mathbb{N}$, given n tasks (with arbitrary, unknown processing times) and p processors, produces a schedule $\mathcal{S}$ with*

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

where h and ε are the average overhead and the average deviation per chunk, and $\gamma = \mathrm{id} - \max\{\, h+\varepsilon, \beta^{-1} \,\}$. This bound is optimal in the sense that no algorithm can improve on it by more than a constant factor (for a precise statement of the lower bound, see [1]).

To obtain from this theorem bounds for a specific setting, we have to provide a bound on the average deviation with respect to a suitably chosen variance estimator. As the bound of Theorem 1 is more sensitive to an increase in ε than to a widening of $[\alpha, \beta]$, this variance estimator should be chosen such that ε will be relatively small, that is, intuitively, such that it is unlikely that the processing time of a chunk of size w lies outside of $[\alpha(w), \beta(w)]$. The following table gives a variance estimator and the corresponding expected average deviation and the order of growth of γ^* for each of the four settings described in the introduction. The first row has already been established in [1], the second row is trivial, and the third and fourth row are proven in the full paper [2].

Independent	$[\alpha,\beta] : w \mapsto [w - \Theta(\sqrt{w \ln w}), w + \Theta(\sqrt{w \ln w})]$	$\varepsilon = O(1)$	$\gamma^* \approx \log\log$
Bounded	$[\alpha,\beta] : w \mapsto [T_{\min} \cdot w, T_{\max} \cdot w]$	$\varepsilon = 0$	$\gamma^* \approx \log$
Exponential	$[\alpha,\beta] : w \mapsto [T_{\min} \cdot w, w \cdot \ln(pw^2)]$	$\varepsilon = O(1)$	$\gamma^* \approx \log^2$
Heavy	$[\alpha,\beta] : w \mapsto [T_{\min} \cdot w, pw^2]$	$\varepsilon = O(1)$	$\gamma^* \approx \sqrt{}$

When dealing with the independent-tasks setting, we will make frequent use of the following lemma, which is a generalization of a result in [1]; a proof can be found in the full paper [2].

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

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

If $\mu_Z = 0$, the bound holds with $1 - \eta$ instead of $2/3 - \eta$. If Z is normal, it holds with $\eta = 0$.

3 Geometric schemes

The scheme GEO($C, w_{\min}$) assigns $\lceil W/(Cp) + w_{\min} \rceil$ tasks to an idle processor, when W tasks are unassigned (in the end, when fewer tasks are left than specified by this number, simply all the remaining tasks are assigned). Without the rounding, the chunk sizes would exactly form a geometric sequence[2]; this follows from

$$(W - (W/(Cp) + w_{\min}))/(Cp) + w_{\min} = (W/(Cp) + w_{\min}) \cdot (1 - 1/(Cp)).$$

Theorem 2 below names the optimal instance of GEO for an arbitrary given variance estimator, and states a bound on its average wasted time that is specified exactly up to some lower order additive term. As explained in the remark, the bound is always within a logarithmic (and thus, as explained in the introduction, small) factor of the optimum.

For the (standard) independent-tasks setting, we can further strengthen our result as indicated in Theorem 3. As a by-product, we obtain a result for a variant of GEO where the chunk size is recomputed only every pth assignment (and kept for the intermediate chunks); the *factoring* principle introduced by Flynn *et al.* [4]. Note that the constant factor for $C = 2$ (which corresponds to the FAC2 scheme advocated by [4]), is $1/\ln 2 \approx 1.44$, to be compared with 1 for the optimal GEO scheme; see also the remark following Theorem 3.

According to the theorems, an optimal choice for C for each of the independent-tasks, bounded ranges, exponential-tails, and heavy-tails setting is as indicated in the following table; a good choice for $w_{\min}$ for each of these settings would be h. However, the bound of Theorem 2 is quite robust against deviations from these optimal parameter settings.

independent	bounded	exponential	heavy
arbitrary $C > 1$	$C = T_{\max}/T_{\min}$	$C = 2\ln n/T_{\min}$	$C = 2\sqrt{n}/T_{\min}$

Theorem 2. *For a given variance estimator $[\alpha, \beta]$, and for $n, p \in \mathbb{N}$, let $w_{\min} \geq 1$ be arbitrary, and let $C = \beta(w_{\min})/\alpha(w_{\min})$ when $\beta(w)/w$ is nonincreasing, and $C = 2 \cdot \alpha^{-1}(n/p)/\beta^{-1}(n/p)$ otherwise. Then GEO(C,$w_{\min}$), given n tasks (with arbitrary, unknown processing times) and p processors, produces a schedule $\mathcal{S}$ with*

$$\mathrm{waste}(\mathcal{S}) = (h + \varepsilon) \cdot C \cdot \bigl(\ln(n/p) + O(1)\bigr),$$

where h and ε are the average overhead and the average deviation per chunk.

[2] Note that rounding the elements of this geometric series does not give the same series, as if the original formula (with rounding) were applied for each assignment. For our analyses, however, the difference turns out to be insignificant, so that an implementation should simply choose the for it most convenient rounding (one could even round downwards).

Remark 1. For $w_{\min} = h + \varepsilon = O(1)$, it holds that $C = O(\gamma^*(n/p))$, where $\gamma = \mathrm{id} - \max\{\, h + \varepsilon, \beta^{-1} \,\}$ as in Theorem 1. When $C = O(1)$ this is trivial, otherwise this follows from the simple observation that for an arbitrary increasing function $f : \mathbb{R} \to \mathbb{R}^+$, $(\mathrm{id} - f)^*(y) \geq y/f(y)$.

Theorem 3. *For $n, p \in \mathbb{N}$, $w_{\min} \geq 1$, and $C > 1$, let $\mathcal{S}$ be the schedule produced by GEO($C, w_{\min}$) given n tasks and p processors, when the task processing times are independent, identically distributed random variables with mean* 1 *and finite third moment, and let h be the average overhead per chunk. Then,*

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

Employing the factoring principle, i.e., keeping the chunk size for p successive chunks, and taking $C = 2$, it holds that

$$\mathbf{E}\,\mathrm{waste}(\mathcal{S}) = \frac{1}{\ln 2} \cdot h \cdot \ln(n/p) + O(\, p^{5/3}\,).$$

Remark 2. The first bound is minimized by taking $C = 1 + \Theta(p^{1/3}/\sqrt{\ln(n/p)})$; then $\mathbf{E}\,\mathrm{waste}(\mathcal{S}) = (1 + o(1)) \cdot h \cdot \ln(n/p)$. For $C = 1$, no logarithmic bound holds, but instead (by Lemma 4 from Section 5) the lower bound $\mathbf{E}\,\mathrm{waste}(\mathcal{S}) \geq 1/3 \cdot \sigma \cdot \sqrt{n/p}$. Given the tightness of our analysis, this provides strong evidence (though does not prove) that no simple algorithm can achieve an average wasted time below $h \cdot \ln(n/p)$.

When factoring is applied, $C = 2$ is the smallest value for which our analysis yields a logarithmic bound. This goes a long way towards explaining why $C = 2$ turned out to give the best results in the experimental study of [4].

The following two lemmas are the keys to the proofs of Theorems 2 and 3. Throughout this section, in the context of a schedule $\mathcal{S}$, let w_j denote the size of the jth chunk, and W_j the number of unassigned tasks (W for work) before that chunk is assigned.

Lemma 2. *The number of chunks assigned by GEO($C, w_{\min}$), given n tasks and p processors, is bounded by $\lceil Cp \cdot \ln(1 + n/(Cpw_{\min}) \rceil$. If factoring is employed, it is bounded by $p \cdot \lceil \log_{1/(1-1/C))}(1 + n/(Cpw_{\min})) \rceil$.*

Proof (sketch). By the equation given at the beginning of this section, the quantity $W/(Cp) + w_{\min}$, where W is the amount of unassigned work, decreases by a factor of at least $c = 1/(1-1/(Cp))$ after each assignment (except the last, which reduces W to zero). This yields $\mathrm{chunks}(\mathcal{S}) \leq \lceil \log_c((n/(Cp) + w_{\min})/w_{\min}) \rceil$, and since $\ln(1/(1 - 1/x)) \geq 1/x$, for all $x > 0$, the first bound follows. The bound for GEO with factoring follows by an analogous argument on the quantity $W/C + p \cdot w_{\min}$.

Lemma 3. *For an arbitrary schedule $\mathcal{S}$, if it holds that for all j, $w_j \leq W_j/(ABp) + K$, where $A = \max_j w_j/\alpha(w_j)$, $B = \max_j \beta(w_j)/w_j$, and $K \geq 0$ is some constant, then*

$$\mathrm{imbalance}(\mathcal{S})/p \leq h_{\max} + BK + \sum\nolimits_{\mathcal{C}\in\mathcal{S}} \mathrm{early}_\alpha(\mathcal{C})/p + \max\nolimits_{\mathcal{C}\in\mathcal{S}} \mathrm{late}_\beta(\mathcal{C}),$$

where $h_{\max}$ is the maximal overhead.

Proof (sketch). We prove by induction that for $j = 0, \ldots, l$, the imbalance of the schedule $\mathcal{S}_j$ consisting of the first j chunks is bounded by

$$ph_{\max} + pBK + W_{j+1}/A + \sum_{\mathcal{C} \in \mathcal{S}_j} \mathrm{early}_\alpha(\mathcal{C}) + (p-1) \cdot \max_{i=1,\ldots,l} \mathrm{late}_\beta(\mathcal{C}).$$

In the induction step $j-1 \to j$, two cases have to be distinguished: if the jth chunk $\mathcal{C}_j$ is the latest to finish among the first j chunks, the imbalance of $\mathcal{S}_j$ can be bounded by $(p-1) \cdot (h_{\max} + \beta(w_j) + \mathrm{late}_\beta(\mathcal{C}_j))$, and it suffices to check that $(p-1) \cdot \beta(w_j) \leq pBw_j - w_j/A \leq W_j/A + pBK - w_j/A = pBK + W_{j+1}/A$; in the opposite case, it suffices to observe that the imbalance decreases by at least the processing time of $\mathcal{C}_j$, which is at least $w_j/A - \mathrm{early}_\alpha(\mathcal{C}_j)$. For more details, see the full paper [2].

Proof (of Theorem 2). Let A and B as in Lemma 3, and check that, by the choice of C in the theorem, and using that the maximal chunk size is $\lceil n/p/C + w_{\min} \rceil$, always $C \geq A \cdot B$ (without loss of generality assuming that $n/p \geq \beta(2w_{\min})$). Since further, by the definition of GEO, $w_j \leq W_j/(Cp) + w_{\min} + 1$, Lemma 3 gives us

$$\mathrm{imbalance}(\mathcal{S})/p \leq h_{\max} + B(w_{\min} + 1) + \sum_{\mathcal{C} \in \mathcal{S}} \mathrm{early}_\alpha(\mathcal{C})/p + \max_{\mathcal{C} \in \mathcal{S}} \mathrm{late}_\beta(\mathcal{C}),$$

Bounding the sum of the last two terms by $\sum_{\mathcal{C} \in \mathcal{S}} \mathrm{dev}_{\alpha,\beta}(\mathcal{C})/p = \varepsilon \cdot \mathrm{chunks}(\mathcal{S})/p$, and adding $h \cdot \mathrm{chunks}(\mathcal{S})/p$, we obtain

$$\mathrm{waste}(\mathcal{S}) \leq (h + \varepsilon) \cdot \mathrm{chunks}(\mathcal{S})/p + C(w_{\min} + 1) + h_{\max},$$

which together with the bound from Lemma 2 proves our theorem.

Proof (of Theorem 3). In this proof it is up to us to choose a variance estimator, and we take $\alpha(w) = w/A$ and $\beta(w) = B \cdot w$. Then for arbitrary $A, B > 1$ with $A \cdot B \leq C$, by an application of Lemma 3 like that in the previous proof,

$$\mathrm{imbalance}(\mathcal{S})/p \leq h_{\max} + B(w_{\min} + 1) + \sum_{\mathcal{C} \in \mathcal{S}} \mathrm{early}_\alpha(\mathcal{C})/p + \max_{\mathcal{C} \in \mathcal{S}} \mathrm{late}_\beta(\mathcal{C}).$$

In order to prove the first part of Theorem 3 we will first bound each of the last two terms and then determine A, B such that their sum becomes as small as possible.

Let $\mathcal{C}_1, \ldots, \mathcal{C}_l$ denote the sequence of chunks assigned by GEO$(C, w_{\min})$, let w_i and T_i denote the size and processing time of $\mathcal{C}_i$, respectively, and let σ denote the standard deviation of a single task's processing time. Then, for $a = 1 - 1/A$,

$$\mathrm{early}_\alpha(\mathcal{C}_i) = \max\{\, 0,\, w_i/A - T_i \,\} = \max\{\, 0,\, w_i - aw_i - T_i \,\},$$

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 η of

Lemma 1 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 ϱ^3 is the third absolute central moment of a single task's processing time and $\vartheta > 0$ is the constant according to Lemma 1. Applying that lemma we thus obtain that, for $K = 1 + \vartheta \cdot \varrho^3/\sigma^3 \,/\, \sqrt{w_{\min}}$,

$$\mathbf{E}\,\mathrm{early}_\alpha(\mathcal{C}_i) \le (1+\eta_i) \cdot \sigma\sqrt{w_i} \cdot \frac{1}{\sqrt{2\pi}}\,\frac{1}{1+t_i^2}\,e^{-t_i^2/2} \le K \cdot \sigma\sqrt{w_i} \cdot \frac{1}{\sqrt{2\pi}}\,\frac{1}{1+t_i^2}\,,$$

and hence, writing $\sqrt{w_i}$ as $\sigma/a \cdot t_i$, and summing over all chunks,

$$\sum\nolimits_{\mathcal{C}\in\mathcal{S}} \mathbf{E}\,\mathrm{early}_\alpha(\mathcal{C}) \le K \cdot \sigma^2/a \cdot \frac{1}{\sqrt{2\pi}} \sum_{i=1}^{l} \frac{1}{t_i + 1/t_i}\,.$$

In order to bound the sum, let us make the simplifying assumption that the chunk sizes were computed without rounding (the difference to the exact analyis is negligible in the context of this proof, certainly at most a factor of two). In that case $w_{i+1} \le w_i \cdot (1-1/(Cp))$, and thus $t_{i+1} \le t_i \cdot \sqrt{1-1/(Cp)} \le t_i \cdot (1-1/(2Cp))$, and with $I_1 = \{i : t_i < 1\}$ and $I_2 = \{i : t_i \ge 1\}$, we obtain that

$$\sum_{i=1}^{l} \frac{1}{t_i + 1/t_i} \le \sum_{i\in I_1} t_i + \sum_{i\in I_2} t_i^{-1} \le 2 \cdot \sum_{j=0}^{\infty} (1-1/(2Cp))^{-j} = 4Cp.$$

This gives us the following bound on the expected total earliness

$$\mathbf{E}\left[\sum\nolimits_{\mathcal{C}\in\mathcal{S}} \mathrm{early}_\alpha(\mathcal{C})\right] \le 2K \cdot \sigma^2/a \cdot Cp = 2K \cdot \sigma^2 \cdot \frac{Cp}{1-1/A}\,.$$

By an analogous calculation, making use of an additional trick (to be found in the full paper [2]), we can further prove

$$\mathbf{E}\left[p \cdot \max\nolimits_{\mathcal{C}\in\mathcal{S}} \mathrm{late}_\beta(\mathcal{C})\right] \le 4K \cdot \sigma^2 \cdot \frac{(Cp)^{2/3}}{B-1}\,.$$

A simple optimization now shows that under the constraints $A, B > 1$ and $A \cdot B \le C$, the sum of the last two bounds is minimal for $A = \frac{(Cp)^{1/3}+C}{(Cp)^{1/3}+1}$ and $B = C \cdot \frac{(Cp)^{1/3}+1}{(Cp)^{1/3}+C}$, and plugging these into the bounds above we obtain that

$$\mathbf{E}\left[\sum\nolimits_{\mathcal{C}\in\mathcal{S}} \mathrm{early}_\alpha(\mathcal{C})/p + \max\nolimits_{\mathcal{C}\in\mathcal{S}} \mathrm{late}_\beta(\mathcal{C})\right] \le 4K \cdot \frac{\sigma^2}{C-1} \cdot \left(C + (Cp)^{1/3}\right)^2.$$

Plugging this into the bound obtained from Lemma 3 at the beginning of this proof, this together with the bound from Lemma 2 proves the first part of Theorem 3.

The second part assumes that the factoring principle is employed, in which case the formula for the jth chunk size becomes

$$w_j = \lceil (W_j + i \cdot w_j)/(Cp) + w_{\min} \rceil,$$

where $i \in \{0, \ldots, p-1\}$ is the number of chunks assigned since the last chunk size was computed (0 if a new chunk size is computed for $\mathcal{C}_j$). We therefore have $(Cp - p + 1) \cdot w_j \leq W_j + Cp(w_{\min} + 1)$, and hence

$$w_j \leq W_j/(Cp-p+1) + \frac{Cp}{Cp-p+1} \cdot (w_{\min}+1) = W_j/((C-1+1/p) \cdot p) + O(w_{\min}).$$

We can now apply Lemma 3 and bound total earliness and maximal lateness in the same way as above (now with $C - 1 + 1/p$ playing the role of C), to show that for $C - 1 + 1/p > 1$,

$$\mathbf{E}\,\mathrm{imbalance}(\mathcal{S})/p = h_{\max} + O\left(C \cdot w_{\min} + \frac{\sigma^2}{C - 2 + 1/p} \cdot \left(C + (Cp)^{1/3} \right)^2 \right).$$

Together with the bound from Lemma 2, and taking $C = 2$ (which is the smallest value, for which $C - 1 + 1/p > 1$ holds in general), this proves the second part of Theorem 3.

4 Fixed-size schemes

The following theorem states an upper and a matching lower bound on the average wasted time of an arbitrary fixed-size scheme with respect to an arbitrary variance estimator $[\alpha, \beta]$. For every variance estimator, this bound is by an order of magnitude larger than what can be achieved by GEO; see Table 1 and the remark below.

Let FIX(w) denote the strategy that, whenever a processor is available, assigns a chunk of w tasks to it (possibly less for the very last chunk). To simplify our presentation, we will assume throughout this section that each chunk has the same overhead, that the number p of processors divides the number n of tasks and that the chunk size w divides n/p. Without these restrictions, the bounds for FIX would rather be even worse than better.

Theorem 4. *Assume that the overhead for each chunk is h. For $n, p \in \mathbb{N}$ and $w \in \mathbb{N}$ with $n/(pw) \in \mathbb{N}$, let $\mathcal{S}$ be the schedule produced by FIX(w) given n tasks and p processors. Then, for an arbitrary given variance estimator $[\alpha, \beta]$,*

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

where ε is the average deviation per chunk.

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, FIX(w) produces a schedule with average deviation ε and

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

Remark 3. The order of growth of these bounds with respect to n/p is simply that of $\min_{w \in \mathbb{N}} \left(n/(pw) + \beta(w)\right)$, for example

$[w-\sqrt{w}, w+\sqrt{w}]$	$[w/2, 2w]$	$[w, w \log w]$	$[w, w^2]$
$\sqrt{n/p}$	$\sqrt{n/p}$	$\sqrt{n/p \cdot \log(n/p)}$	$(n/p)^{2/3}$

Proof (sketch). For the proof of the upper bound define $\mathcal{E} = \sum_{\mathcal{C} \in \mathcal{S}} \mathrm{dev}_{\alpha,\beta}(\mathcal{C})$. Since the time a processor has to wait for the last processor to finish is at most the overhead plus the processing time of the last chunk to finish, we have, denoting that chunk by $\mathcal{C}$,

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

It is also not hard to show that p allocations can increase the imbalance by at most $(p-1) \cdot (\beta(w) - \alpha(w))$ plus the deviations of the corresponding chunks (see [2] for details), which, via a simple induction, implies that

$$\mathrm{imbalance}(\mathcal{S}) \leq n/w \cdot \big(\beta(w) - \alpha(w)\big) + \mathcal{E}.$$

From these two bounds the upper bound claimed in the theorem follows easily.

For the lower bound, given some $\varepsilon \geq 0$, we construct the following sequence of chunk processing times. Let $m = n/(pw)$ and $m' = \min\{m, \lfloor (h+\beta(w))/(\beta(w) - \alpha(w)) \rfloor\} \leq m$. 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 pth 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 pth 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 $\beta(w) + \varepsilon \cdot n/w \; / \; (p-1)$. The corresponding schedule clearly has an average deviation of ε, and it is not hard to see that its wasted time can be bounded as claimed in the theorem. For more details, we again have to refer to the full paper [2].

5 Analysis of GSS, FAC, and TAPER

In this section, we will prove that in the independent-tasks setting, Guided Self-Scheduling (GSS) by Polychronopoulos and Kuck [12], the original Factoring scheme (FAC) of Flynn, Hummel, and Schonberg [4], and the Tapering scheme (TAPER) by Lucco [9], all incur a wasted time that is on the same asymptotic order as for the naive static scheme, which simply assigns a single chunk of size n/p (suitably rounded) to each processor. The constant factors, as implied by Lemma 4 below, appear rather low, but there is a nonnegligible probability that the wasted time will exceed its mean by a significant factor. Therefore each of these strategies will indeed perform poorly every once in a while (which is also what we observed in experiments).

Theorem 5. *For $n, p \in \mathbb{N}$, let $\mathcal{S}$ be the schedule produced by GSS, FAC, or TAPER given n tasks and p processors, when the processing times of the tasks are independent, identically distributed random variables with mean 1, variance $\sigma^2 > 0$ and finite third moment. Then*

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

To prove the theorem, it suffices to consider the size w_1 of the first chunk assigned by GSS, FAC, or TAPER. For GSS, we have $w_1 \geq n/p$, for FAC we have $w_1 + \sigma\sqrt{p/2}\sqrt{w_1} \geq n/p$, and for TAPER we have $w_1 + 1.3\sigma\sqrt{w_1} \geq n/p$. The theorem is therefore implied by the following general lemma, which, intuitively, says that one should be very careful so as not to choose the initial chunks too large.

Lemma 4. *If $w_1 + K \cdot \sigma\sqrt{w_1} \geq n/p$, for a fixed $K \geq 0$, it holds that for p and n/p sufficiently large (with respect to $K \cdot \sigma$)*

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

For $K = 0$, the $1/4$ may be replaced by $1/3$.

Proof (sketch). Let $\mathcal{C}_1$ denote any one of the first p chunks with a size w_1 satisfying $w_1 + K \cdot \sigma\sqrt{w_1} \geq n/p$. 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. It is not hard to see that $p \cdot \mathrm{waste}(\mathcal{S}) \geq (p-1) \cdot T_{\mathcal{C}_1} - T_{\mathcal{S}\setminus\mathcal{C}_1}$, and thus, with $Z = (p-1) \cdot T_{\mathcal{C}_1} - T_{\mathcal{S}\setminus\mathcal{C}_1}$, $\mathbf{E}\,\mathrm{waste}(\mathcal{S}) \;\geq\; \mathbf{E}\max\{\,0, Z\,\} \;/\; p$. A straightforward application of Lemma 1 now gives a lower bound on $\mathbf{E}\max\{\,0, Z\,\}$, from which the bound claimed in the lemma can be derived by a series of simple estimations. For the details, again see [2].

Acknowledgments: Thanks to Berthold Vöcking for prodding me to write this up. I'm also grateful to one of the referees for his detailed and knowledgeable comments on the original submission. I only wish his motivation had been to improve the final version and not to reject the paper.

References

1. Bast, H., On Scheduling Parallel Tasks at Twilight, *Theory of Computing Systems* **33**:5 (2000), pp. 489–563.
2. Bast, H., Scheduling at Twilight the Easy Way, full paper (2002), `http://www.mpi-sb.mpg.de/~hannah/papers`.
3. Eager, D. L., and Subramaniam, S., Affinity scheduling of unbalanced workloads, *in* Proceedings Supercomputing (SC 1994), pp. 214–226.
4. Flynn, L. E., Hummel, S. F., and Schonberg, E., Factoring: A method for scheduling parallel loops, *Communications of the ACM* **35**:8 (1992), pp. 90–101.
5. Hagerup, T., Allocating Independent Tasks to Parallel Processors: An Experimental Study, *Journal of Parallel and Distributed Computing* **47** (1997), pp. 185–197.
6. Hummel, S. F., Schmidt, J., Uma, R. N., and Wein, J., Load-sharing in heterogeneous systems via weighted factoring, *in* Proceedings 8th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA 1996), pp. 318–328.
7. Kruskal, C. P., and Weiss, A., Allocating independent subtasks on parallel processors, *IEEE Transactions on Software Engeneering* **11** (1985), pp. 1001–1016.

8. Liu, J., Saletore, V. A., and Lewis, T. G., Safe self scheduling—a parallel loop scheduling scheme for shared-memory multiprocessors, *International Journal of Parallel Programming* **22**:6 (1994), pp. 589–616.
9. Lucco, S., A dynamic scheduling method for irregular parallel programs, *in* Proceedings Conference on Programming Language Design and Implementation (PLDI 1992), pp. 200–211.
10. Markatos, E. P., and LeBlanc, T. J., Using processor affinity in loop scheduling on shared-memory multiprocessors, *IEEE Transactions on Parallel and Distributed Systems* **5**:4 (1994), pp. 379–400.
11. Orlando, S., and Perego, R., Scheduling data-parallel computations on heterogeneous and time-shared environments, *in* Proceedings European Conference on Parallel Computing (EURO-PAR 1998), Springer Lecture Notes in Computer Science **1470**, pp. 356–365.
12. Polychronopoulos, C. D., and Kuck, D. J., Guided self-scheduling: A practical scheduling scheme for parallel supercomputers, *IEEE Transactions on Computers* **36** (1987), pp. 1425–1439.
13. Tzen, T. H., and Ni, L. M., Trapezoid self-scheduling—a practical scheduling scheme for parallel compilers, *IEEE Transactions on Parallel and Distributed Systems* **4**:1 (1993), pp. 87–98.