Space-efficient scheduling of stochastically generated tasks *

Tomáš Brázdil Javier Esparza
Stefan Kiefer Michael Luttenberger

Institut für Informatik, Technische Universität München, Germany
{brazdil,esparza,kiefer,luttenbe}@model.in.tum.de

Abstract. We study the problem of scheduling tasks for execution by a processor when the tasks can stochastically generate new tasks. Tasks can be of different types, and each type has a fixed, known probability of generating d tasks for each number d. We present results on the random variable S^σ modeling the maximal space needed by the processor to store the currently active tasks when acting under the scheduler σ. We obtain tail bounds for the distribution of S^σ for both offline and online schedulers, and also bounds on the expected value $\mathbb{E}[S^\sigma]$.

* This work was partially supported by the DFG project *Algorithms for Software Model Checking*.

1 Introduction

We study the problem of scheduling tasks that can stochastically generate new tasks. We assume that the execution of a task τ can generate a set of subtasks $\tau_1, \tau_2, \ldots, \tau_d$, where $d \geq 0$. Tasks can be of different types, and each type has a fixed, known probability of generating d subtasks for each number d. Systems of tasks can be described using a notation similar to that of stochastic grammars. For instance

$$X \xhookrightarrow{0.2} \langle X, X\rangle \qquad X \xhookrightarrow{0.3} \langle X, Y\rangle \qquad X \xhookrightarrow{0.5} \emptyset$$
$$Y \xhookrightarrow{0.7} \langle X\rangle \qquad Y \xhookrightarrow{0.3} \langle Y\rangle$$

describes a system with two types of tasks. Tasks of type X can generate 2 tasks of type X, one task of each type, or zero tasks with probabilities 0.2, 0.3, and 0.5, respectively (angular brackets denote multisets). Tasks of type Y can generate one task, of type X or Y, with probability 0.7 and 0.3. Tasks are executed by one processor. The processor repeatedly selects a task from a pool of unprocessed tasks, processes it, and puts the generated subtasks (if any) back into the pool. The pool initially contains one task of type X_0, and the next task to be processed is selected by a *scheduler*.

We are interested in the random variables modeling the time and space needed to *completely* execute a task τ, i.e., to empty the pool of unprocessed tasks assuming that initially the pool only contains task τ. We assume that processing a task takes one time unit, and storing it in the pool takes a unit of memory. So the *completion time* is given by the total number of tasks processed, and the *completion space* by the maximum size reached by the pool during the computation. It is easy to see that the distribution of the completion time is independent of the scheduler, but that of the completion space is not. The completion time has been studied in [12], and so the bulk of the paper is devoted to studying the distribution of the completion space for different classes of schedulers.

Our computational model is abstract, but relevant for different scenarios. In the context of search problems, a task is a problem instance, and the scheduler is part of a branch-and-bound algorithm (see e.g. [22]). The processor either directly solves the instance ($d = 0$), or extracts from it a set of sub-instances ($d > 0$). In the more general context of multithreaded computations, a task models a thread, which, executed for at most one unit of time, either terminates ($d = 0$), generates a new thread ($d = 2$), or none of the two ($d = 1$).[1] The problem of scheduling multithreaded computations space-efficiently on *multi*processor machines has been extensively studied (see e.g. [27, 6, 2, 1]). However, these papers study the worst-case performance of certain schedulers over all possible computations generated by all possible programs, when the schedulers know nothing about the program. We initiate the study of a different problem: schedule computations when stochastic information on the programs generating them is available (for instance, by collecting statistics on the behavior of the programs), and obtain stochastic performance bounds. We consider the single-processor case, which is trivial in the setting of [27, 6, 2, 1], but hard in our setting, and leading to a rich theory.

We study the performance of *online* schedulers that know the past of the computation, but not its future. As a measure for their performance, we also study the *optimal offline* scheduler, which has complete information about the future of the computation. Intuitively, this scheduler has access to an oracle that knows how the stochastic choices will be resolved. The oracle can be replaced by a machine that inspects the code of a task and determines which subtasks it will generate (if any).

We consider task systems with completion probability 1, which can be further divided into those with finite and infinite expected completion time, often called *subcritical* and *critical*. Whether a system has completion probability 1, and if so whether it is critical or subcritical, can be determined in polynomial time [14]. Many of our results are related to the probability generating functions (pgfs) associated to a task system. The functions for the example above are $f_X(x, y) = 0.2x^2 + 0.3xy + 0.5$ and $f_Y(x, y) = 0.7x + 0.3y$,

[1] Notice that we do not model dependencies between threads, but see the point below on depth-first schedulers.

and the reader can easily guess the formal definition. The completion probability is the least fixed point of the system of pgfs [18].

Our first results (Section 3) concern the distribution of the completion space S^{op} of the optimal offline scheduler op on a fixed but arbitrary task system with $\boldsymbol{f}(\boldsymbol{x})$ as pgfs (in vector form). We exhibit a surprising connection between the probabilities $\Pr[S^{op} = k]$ and the *Newton approximants* to the least fixed point of $\boldsymbol{f}(\boldsymbol{x})$ (the approximations to the least fixed point obtained by applying Newton's method for approximating a zero of a differentiable function to $\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{x} = \mathbf{0}$ with seed $\mathbf{0}$). This connection allows us to apply recent results on the convergence speed of Newton's method [23, 11], leading to bounds for $\Pr[S^{op} \geq k]$, and to an efficient algorithm for approximating $\mathbb{E}[S^{op}]$. We then study (Section 4) the distribution of S^{σ} for an online scheduler σ. Using a martingale argument we obtain upper and lower bounds for the performance of *any* online scheduler σ in subcritical systems. These bounds suggest a way of assigning weights to task types reflecting how likely they are to require large space. We study *light-first* schedulers, in which "light" tasks are chosen before "heavy" tasks with larger components, and obtain an improved tail bound.

So far we have assumed that there are no dependencies between tasks, requiring a task to be executed before another. We study in Section 4.3 the case in which a task can only terminate after all the tasks it has (recursively) spawned have terminated. These are the *strict* computations studied in [6]. The optimal scheduler in this case is the one that completely executes the child task before its parent, resulting in the familiar stack-based execution. We determine the exact asymptotic performance of depth-first schedulers.

We finish the paper by presenting some results on minimizing the expected completion space (Section 5). It is easy to see that in a subcritical system every online scheduler has finite expected completion space. We show that in a critical system they all have infinite expected value; that is, a scheduler can only achieve a finite expected value if it has information about the future (loosely speaking, it must look into the code). We also show that schedulers minimizing the expected completion space exist but require unbounded memory.

Related work. Space-efficient scheduling for search problems or multithreaded computations has been studied in [22, 27, 6, 2, 1]. However, these papers only study the worst-case: they provide schedulers with a guaranteed space-consumption for any computation. In this paper we assume that statistical information is available on the probability that a computation splits or dies.

Our paper is related to the theory of *urn models* [21, 26] and *branching processes*, stochastic processes modeling the evolution of populations whose members can reproduce or die [18, 4]. However, branching processes have been studied as models of biological or physical systems, and, in computer science terminology, the assumption is made that the number of processors is *unbounded*. The maximum population in this setting has been studied in [3, 7, 25, 28, 30, 32]. We study the 1-processor case, which to our knowledge has not been previously studied. Some urn models studied in the literature exactly match our 1-processor model [20, 24], but the random variable modeling space consumption does not seem to have been studied in the setting of multiple types. In the single-type case, the space consumption corresponds to the maximum of a particular random walk associated with the Gambler's-Ruin problem [8, 16, 31].

Recursive state machines [14] and probabilistic pushdown automata [13] can be seen as instances of our model for schedulers satisfying the following constraint: if thread A spawns thread B, then B is executed before A. For these schedulers, the completion space corresponds to the maximal recursion depth or stack height, which has not been studied so far.

2 Preliminaries

Let A be a finite set. We regard elements of $\mathbb{N}^A$ and $\mathbb{R}^A$ as *vectors* and use boldface (like $\boldsymbol{u}, \boldsymbol{v}$) to denote vectors. The vector whose components are all 0 (resp. 1) is denoted by $\mathbf{0}$ (resp. $\mathbf{1}$). We use angular brackets to denote multisets and often identify multisets over A and vectors indexed by A. For instance, if $A = \{X, Y\}$

and $\boldsymbol{v} \in \mathbb{N}^A$ with $\boldsymbol{v}_X = 1$ and $\boldsymbol{v}_Y = 2$, then $\boldsymbol{v} = \langle X, Y, Y \rangle$. We often shorten $\langle a \rangle$ to a. $M_A^{\leq 2}$ denotes the multisets over A containing at most 2 elements.

Definition 2.1. *A* task system *is a tuple* $\Delta = (\Gamma, \hookrightarrow, \mathit{Prob}, X_0)$ *where* Γ *is a finite set of* task types, $\hookrightarrow \subseteq \Gamma \times M_\Gamma^{\leq 2}$ *is a set of* transition rules, Prob *is a function assigning positive probabilities to transition rules so that for every* $X \in \Gamma$ *we have* $\sum_{X \hookrightarrow \alpha} \mathit{Prob}((X, \alpha)) = 1$*, and* $X_0 \in \Gamma$ *is the* initial type.

We write $X \overset{p}{\hookrightarrow} \alpha$ whenever $X \hookrightarrow \alpha$ and $\mathit{Prob}((X, \alpha)) = p$. Executions of a task system are modeled as family trees, defined as follows. Fix an arbitrary total order $\preceq$ on Γ. A *family tree* t is a pair (N, L) where $N \subseteq \{0,1\}^*$ is a finite binary tree (i.e. a prefix-closed finite set of words over $\{0,1\}$) and $L : N \hookrightarrow \Gamma$ is a labelling such that every node $w \in N$ satisfies one of the following conditions: w is a leaf and $L(w) \hookrightarrow \varepsilon$, or w has a unique child $w0$, and $L(w)$ satisfies $L(w) \hookrightarrow L(w0)$, or w has two children $w0$ and $w1$, and $L(w0)$, $L(w1)$ satisfy $L(w) \hookrightarrow \langle L(w0), L(w1) \rangle$ and $L(w0) \preceq L(w1)$. Given a node $w \in N$, the subtree of t rooted at w, denoted by t_w, is the family tree (N', L') such that $w' \in N'$ iff $ww' \in N$ and $L'(w') = L(ww')$ for every $w' \in N'$. If a tree t has a subtree t_0 or t_1, we call this subtree a *child* of t. (So, the term *child* can refer to a node or a tree, but there will be no confusion.)

We define a function $\Pr$ which, loosely speaking, assigns to a family tree t its probability (see the assumption below). Let $t = (N, L)$ be a family tree. Assume that the root of t is labeled by X. If t consists only of the root, and $X \overset{p}{\hookrightarrow} \varepsilon$, then $\Pr[t] = p$; if the root has only one child (the node 0) labeled by Y, and $X \overset{p}{\hookrightarrow} Y$, then $\Pr[t] = p \cdot \Pr[t_0]$; if the root has two children (the nodes 0 and 1) labeled by Y and Z, and $X \overset{p}{\hookrightarrow} \langle Y, Z \rangle$, then $\Pr[t] = p \cdot \Pr[t_0] \cdot \Pr[t_1]$. We denote by $\mathcal{T}_X$ the set of all family trees whose root is labeled by X, and by $\Pr_X$ the restriction of $\Pr$ to $\mathcal{T}_X$. We drop the subscript of $\Pr_X$ if X is understood.

Example 2.2. Figure 1 shows (a) a task system with $\Gamma = \{X, Y, Z\}$; and (b) a family tree t of the system with probability $\Pr[t] = 0.25 \cdot 0.1 \cdot 0.75 \cdot 0.6 \cdot 0.4 \cdot 0.9$. The name and label of a node are written close to it.

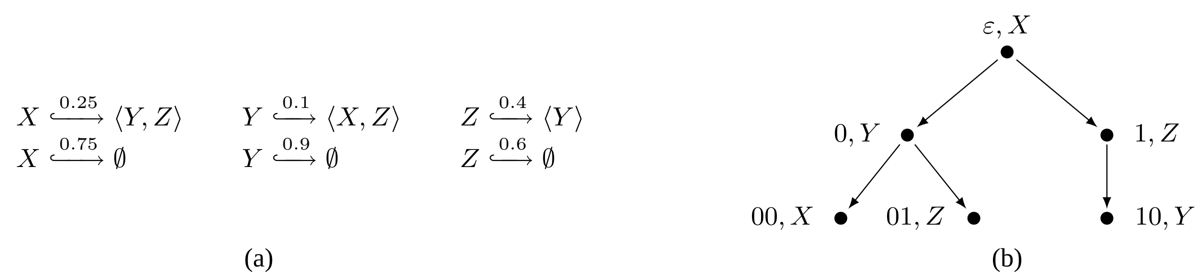

Fig. 1. (a) A task system. (b) A family tree.

Assumptions. Throughout the paper we assume that a task system $\Delta = (\Gamma, \hookrightarrow, \mathit{Prob}, X_0)$ satisfies the following two conditions for every type $X \in \Gamma$: (1) X is *reachable* from X_0, meaning that some tree in $\mathcal{T}_{X_0}$ contains a node labeled by X, and (2) $\Pr[\mathcal{T}_X] = \sum_{t \in \mathcal{T}_X} \Pr[t] = 1$. In other words, we assume that $(\mathcal{T}_X, \Pr_X)$ is a discrete probability space with $\mathcal{T}_X$ as set of elementary events and $\Pr_X$ as probability function. This is the formal counterpart to assuming that every task is completed with probability 1.

Proposition 2.3. *It can be decided in polynomial time whether assumptions (1) and (2) are satisfied.*

Proof. The statement on assumption (1) is trivial. For assumption (2) let the *probability generating function* (pgf) of the task system be defined as the function $\boldsymbol{f} : \mathbb{R}^\Gamma \to \mathbb{R}^\Gamma$ of Δ where for every $X \in \Gamma$

$$\boldsymbol{f}_X(\boldsymbol{v}) = \sum_{X \overset{p}{\hookrightarrow} \langle Y, Z \rangle} p \cdot \boldsymbol{v}_Y \cdot \boldsymbol{v}_Z + \sum_{X \overset{p}{\hookrightarrow} \langle Y \rangle} p \cdot \boldsymbol{v}_Y + \sum_{X \overset{p}{\hookrightarrow} \emptyset} p\,.$$

It is well known (see e.g. [18]) that assumption (2) holds iff the least fixed point of $\boldsymbol{f}$ equals $\mathbf{1}$. This condition is decidable in polynomial time [14]. The pgf $\boldsymbol{f}$ will play a crucial role in the following. $\square$

Derivations and schedulers. Let $t = (N, L)$ be a family tree. A *state* of t is a maximal subset of N in which no node is a proper prefix of another node (graphically, no node is a proper descendant of another node). The elements of a state s are called *tasks*. If s is a state and $w \in s$, then the *w-successor of s* is the uniquely determined state s' defined as follows: if w is a leaf of N, then $s' = s \setminus \{w\}$; if w has one child $w0$, then $s' = (s \setminus \{w\}) \cup \{w0\}$; if w has two children $w0$ and $w1$, then $s' = (s \setminus \{w\}) \cup \{w0, w1\}$. We write $s \Rightarrow s'$ if s' is the w-successor of s for some w. A *derivation of t* is a sequence $s_1 \Rightarrow \ldots \Rightarrow s_k$ of states such that $s_1 = \{\epsilon\}$ and $s_k = \emptyset$. Observe that a tree may have multiple derivations. A *scheduler* is a mapping σ that assigns to a family tree t a derivation $\sigma(t)$ of t. If $\sigma(t) = (s_1 \Rightarrow \ldots \Rightarrow s_k)$, then for every $1 \le i < k$ we denote by $\sigma(t)[i]$ a task of s_i such that s_{i+1} is the $\sigma(t)[i]$-successor of s_i. Intuitively, $\sigma(t)[i]$ is the task of s_i scheduled by σ. Notice that this definition allows for schedulers that know the tree, and so how the tasks will behave. In Section 4 we define and study online schedulers which only know the past of the computation.

Example 2.4. A scheduler σ_1 may schedule the tree t in Figure 1 as follows: $\{\varepsilon\} \Rightarrow \{0, 1\} \Rightarrow \{0, 10\} \Rightarrow \{0\} \Rightarrow \{00, 01\} \Rightarrow \{01\} \Rightarrow \{\}$. Let σ_2 be the scheduler which always picks the least unprocessed task w.r.t. the lexicographical order on $\{0, 1\}^*$. (This is an example of an online scheduler.) It schedules t as follows: $\{\varepsilon\} \Rightarrow \{0, 1\} \Rightarrow \{00, 01, 1\} \Rightarrow \{01, 1\} \Rightarrow \{1\} \Rightarrow \{10\} \Rightarrow \{\}$.

Time and space. Given $X \in \Gamma$, we define a random variable T_X, the *completion time of X*, that assigns to a tree $t \in \mathcal{T}_X$ its number of nodes. If we assume that each task is executed during one time unit before its generated subtasks are returned to the pool, then T_X corresponds indeed to the time the processor needs to completely execute X. Notice that T_X does not depend on a scheduler, and that our assumptions guarantee that T_X is always finite. However, the expectation $\mathbb{E}[T_X]$ may or may not be finite. A task system Δ is called *subcritical* if $\mathbb{E}[T_X]$ is finite for every $X \in \Gamma$. Otherwise it is called *critical*. If Δ is subcritical, then $\mathbb{E}[T_X]$ can be easily computed by solving a system of linear equations [12]. The notion of criticality comes from the theory of branching processes, see e.g. [18, 4]. Here we only recall the following results:

Proposition 2.5 ([18, 14]). *Let Δ be a task system with pgf $\boldsymbol{f}$. Denote by $\boldsymbol{f}'(\mathbf{1})$ the Jacobian matrix of partial derivatives of $\boldsymbol{f}$ evaluated at $\mathbf{1}$. If Δ is critical, then the spectral radius of $\boldsymbol{f}'(\mathbf{1})$ is equal to 1; otherwise it is strictly less than 1. It can be decided in polynomial time whether Δ is critical.*

A state models a pool of tasks awaiting to be scheduled. We are interested in the maximal size of the pool during the execution of a derivation. So we define the *random width* S^σ_X as follows. If $\sigma(t) = (s_1 \Rightarrow \ldots \Rightarrow s_k)$, then $S^\sigma_X(t) := \max\{|s_1|, \ldots, |s_k|\}$, where $|s_i|$ is the cardinality of s_i. Sometimes we write $S^\sigma(t)$, meaning $S^\sigma_X(t)$ for the type X labelling the root of t. If we write S^σ without specifying the application to any tree, then we mean $S^\sigma_{X_0}$.

Example 2.6. For the schedulers of Example 2.4 we have $S^{\sigma_1}(t) = 2$ and $S^{\sigma_2}(t) = 3$.

3 Optimal (Offline) Schedulers

Let S^{op} be the random variable that assigns to a family tree the minimal width of its derivations. We call $S^{op}(t)$ the *optimal width* of t. The optimal scheduler assigns to each tree a derivation with optimal width. In the multithreading scenario, it corresponds to a scheduler that can inspect the code of a thread and decide whether it will spawn a new thread or not. While in most scenarios the optimal scheduler is not realizable or computationally too expensive, it provides an absolute lower bound for the space resources. The following proposition characterizes the optimal width of a tree in terms of the optimal width of its children.

Proposition 3.1. *Let t be a family tree. Then*

$$S^{op}(t) = \begin{cases} \min\{\max\{S^{op}(t_0)+1, S^{op}(t_1)\}, \max\{S^{op}(t_0), S^{op}(t_1)+1\}\} & \text{if } t \text{ has two children } t_0, t_1 \\ S^{op}(t_0) & \text{if } t \text{ has exactly one child } t_0 \\ 1 & \text{if } t \text{ has no children.} \end{cases}$$

Proof sketch (see the appendix for more details). The only nontrivial case is when t has two children t_0 and t_1. Consider the following schedulings for t, where $i \in \{0,1\}$: Execute first all tasks of t_i and then all tasks of t_{1-i}; within both t_i and t_{i-1}, execute tasks in optimal order. While executing t_i, the root task of t_{1-i} remains in the pool, and so the completion space is $s(i) = \max\{S^{op}(t_i)+1, S^{op}(t_{1-i})\}$. The optimal scheduler chooses the value of i that minimizes $s(i)$. $\square$

Given a type X, we are interested in the probabilities $\Pr\left[S_X^{op} \le k\right]$ for $k \ge 1$. Proposition 3.1 yields a recurrence relation which at first sight seems difficult to handle. However, using results of [10, 9] we can exhibit a surprising connection between these probabilities and the pgf $\boldsymbol{f}$.

Let $\boldsymbol{\mu}$ denote the least fixed point of $\boldsymbol{f}$ and recall from Section 2 that $\boldsymbol{\mu} = \mathbf{1}$. Clearly, $\mathbf{1}$ is a zero of $\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{x}$. It has recently been shown that $\boldsymbol{\mu}$ can be computed by applying to $\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{x}$ Newton's method for approximating a zero of a differentiable function [14, 23]. More precisely, $\boldsymbol{\mu} = \lim_{k\to\infty} \boldsymbol{\nu}^{(k)}$ where

$$\boldsymbol{\nu}^{(0)} = \mathbf{0} \quad \text{and} \quad \boldsymbol{\nu}^{(k+1)} = \boldsymbol{\nu}^{(k)} + (I - \boldsymbol{f}'(\boldsymbol{\nu}^{(k)}))^{-1}\left(\boldsymbol{f}(\boldsymbol{\nu}^{(k)}) - \boldsymbol{\nu}^{(k)}\right)$$

and $\boldsymbol{f}'(\boldsymbol{\nu}^{(k)})$ denotes the Jacobian matrix of partial derivatives of $\boldsymbol{f}$ evaluated at $\boldsymbol{\nu}^{(k)}$ and I the identity matrix. Computing $\boldsymbol{\mu}$, however, is in our case uninteresting: Recall that we assume $\Pr[\mathcal{T}_X] = 1$ for every type X, and in this case it is well-known that $\boldsymbol{\mu} = \mathbf{1}$ [18]. So, why do we need Newton's method? Because the sequence of Newton approximants provides exactly the information we are looking for:

Theorem 3.2. $\Pr\left[S_X^{op} \le k\right] = \boldsymbol{\nu}_X^{(k)}$ *for every type X and every $k \ge 0$.*

Proof sketch (see the appendix full proofs). We illustrate the proof idea on the one-type task system with pgf $f(x) = px^2 + q$, where $q = 1 - p$. Let $\mathcal{T}_{\le k}$ and $\mathcal{T}_{=k}$ denote the sets of trees t with $S^{op}(t) \le k$ and $S^{op}(t) = k$, respectively. We show $\Pr[\mathcal{T}_{\le k}] = \nu^{(k)}$ for all k by induction on k. The case $k = 0$ is trivial. Assume that $\nu^{(k)} = \Pr[\mathcal{T}_{\le k}]$ holds for some $k \ge 0$. We prove $\Pr[\mathcal{T}_{\le k+1}] = \nu^{(k+1)}$. Notice that

$$\nu^{(k+1)} := \nu^{(k)} + \frac{f(\nu^{(k)}) - \nu^{(k)}}{1 - f'(\nu^{(k)}} = \nu^{(k)} + (f(\nu^{(k)}) - \nu^{(k)}) \cdot \sum_{i=0}^{\infty} f'(\nu^{(k)})^i \,.$$

Let $\mathcal{B}_{k+1}^{(0)}$ be the set of trees that have two children both of which belong to $\mathcal{T}_{=k}$, and, for every $i \ge 0$, let $\mathcal{B}_{k+1}^{(i+1)}$ be the set of trees with two children, one belonging to $\mathcal{T}_{\le k}$, the other one to $\mathcal{B}_{k+1}^{(i)}$. By Proposition 3.1 we have $\mathcal{T}_{\le k+1} = \bigcup_{i\ge 0} \mathcal{B}_{k+1}^{(i)}$. We prove $\Pr\left[\mathcal{B}_{k+1}^{(i)}\right] = f'(\nu^{(k)})^i \, (f(\nu^{(k)} - \nu^{(k)})$ by an (inner) induction on i, which completes the proof. For the base $i = 0$, let $\mathcal{A}_{\le k}$ be the set of trees with two children in $\mathcal{T}_{\le k}$; by induction hypothesis we have $\Pr[\mathcal{A}_{\le k}] = p\nu^{(k)}\nu^{(k)}$. In a tree of $\mathcal{A}_{\le k}$ either (a) both children belong to $\mathcal{T}_{=k}$, and so $t \in \mathcal{B}_{k+1}^{(0)}$, or (b) at most one child belongs to $\mathcal{T}_{=k}$. By Proposition 3.1, the trees satisfying (b) belong to $\mathcal{T}_{\le k}$. In fact, a stronger property holds: a tree of $\mathcal{T}_{\le k}$ either satisfies (b) or it has one single node. Since the probability of the tree with one node is q, we get $\Pr[\mathcal{A}_{\le k}] = \Pr\left[\mathcal{B}_{k+1}^{(0)}\right] + \Pr[\mathcal{T}_{\le k}] - q$. Applying the induction hypothesis again we obtain $\Pr\left[\mathcal{B}_{k+1}^{(0)}\right] = p\nu^{(k)}\nu^{(k)} + q - \nu^{(k)} = f(\nu^{(k)}) - \nu^{(k)}$. For the induction step, let $i > 0$. Divide $\mathcal{B}_{k+1}^{(i)}$ into two sets, one containing the trees whose left (right) child belongs to $\mathcal{B}_{k+1}^{(i)}$ (to $\mathcal{T}_{\le k}$), and the other the trees whose left (right) child belongs to $\mathcal{T}_{\le k}$ (to $\mathcal{B}_{k+1}^{(i)}$).

Using both induction hypotheses, we get that the probability of each set is $p\nu^{(k)} f'(\nu^{(k)})^i (f(\nu^{(k)}) - \nu^{(k)})$. So $\Pr\left[\mathcal{B}_{k+1}^{(i+1)}\right] = (2p\nu^{(k)}) \cdot f'(\nu^{(k)})^i (f(\nu^{(k)}) - \nu^{(k)})$. Since $f(x) = px^2 + q$ we have $f'(\nu^{(k)}) = 2p\nu^{(k)}$, and so $\Pr\left[\mathcal{B}_{k+1}^{(i+1)}\right] = f'(\nu^{(k)})^{i+1} (f(\nu^{(k)} - \nu^{(k)})$ as desired. □

Example 3.3. Consider the task system $X \overset{p}{\hookrightarrow} \langle X, X \rangle,\ X \overset{q}{\hookrightarrow} \emptyset$ with pgf $f(x) = px^2 + q$, where p is a parameter and $q = 1 - p$. The least fixed point of f is 1 if $p \leq 1/2$ and q/p otherwise. So we consider only the case $p \leq 1/2$. The system is critical for $p = 1/2$ and subcritical for $p < 1/2$. Using Newton approximants we obtain the following recurrence relation for the distribution of the optimal scheduler, where $p_k := \Pr[S^{op} \geq k]$: $p_{k+1} = (pp_k^2)/(1 - 2p + 2pp_k)$. In particular, for the critical value $p = 1/2$ we get $p_k = 2^{1-k}$ and $\mathbb{E}[S^{op}] = \sum_{i \geq 1} \Pr[S^{op} \geq k] = 2$.

Theorem 3.2 allows to compute the probability mass function of S^{op}. As a Newton iteration requires $\mathcal{O}(|\Gamma|^3)$ arithmetical operations, we obtain the following corollary, where by the unit cost model we refer to the cost in the Blum-Shub-Smale model, in which arithmetic operations have cost 1 independently of the size of the operands.

Corollary 3.4. $\Pr\left[S_X^{op} = k\right]$ *can be computed in time* $\mathcal{O}(k \cdot |\Gamma|^3)$ *in the unit cost model.*

It is easy to see that Newton's method converges quadratically for subcritical systems (see e.g. [29]). For critical systems, it has recently been proved that Newton's method still converges linearly [23, 11]. These results lead to tail bounds for S_X^{op}:

Corollary 3.5. *For any task system* Δ *there are real numbers* $c > 0$ *and* $0 < d < 1$ *such that* $\Pr\left[S_X^{op} \geq k\right] \leq c \cdot d^k$ *for all* $k \in \mathbb{N}$. *If* Δ *is subcritical, then there are real numbers* $c > 0$ *and* $0 < d < 1$ *such that* $\Pr\left[S_X^{op} \geq k\right] \leq c \cdot d^{2^k}$ *for all* $k \in \mathbb{N}$.

4 Online Schedulers

From this section on we concentrate on online schedulers that only know the past of the computation. Formally, a scheduler σ is *online* if for every tree t with $\sigma(t) = (s_1 \Rightarrow \ldots \Rightarrow s_k)$ and for every $1 \leq i < k$, the task $\sigma(t)[i]$ depends only on $s_1 \Rightarrow \ldots \Rightarrow s_i$ and on the restriction of the labelling function L to $\bigcup_{j=1}^{i} s_j$.

Fix an online scheduler σ. For every tree t with $\sigma(t) = (s_1 \Rightarrow \ldots \Rightarrow s_k)$ and for every $j \geq 0$, let $\boldsymbol{z}^{(j)}(t)$ denote the multiset of types labelling the tasks of s_j if $j \leq k$ (i.e., $\boldsymbol{z}^{(j)}(t) = \langle L(w) \mid w \in s_j \rangle$), and the empty multiset otherwise. One can show that an online scheduler σ induces a partial function $\Lambda_\sigma \colon (\mathbb{N}^\Gamma)^* \to \Gamma$ defined as follows: $\Lambda_\sigma(\boldsymbol{c}^{(1)} \ldots \boldsymbol{c}^{(i)})$ is defined if there is a tree t such that $\sigma(t) = (s_1 \Rightarrow \ldots \Rightarrow s_k)$ with $k \geq i$ and $\boldsymbol{c}^{(1)} = \boldsymbol{z}^{(1)}(t), \ldots, \boldsymbol{c}^{(i)} = \boldsymbol{z}^{(i)}(t)$; in this case $\Lambda_\sigma(\boldsymbol{c}^{(1)} \ldots \boldsymbol{c}^{(i)}) = L(\sigma(t)[i])$. Intuitively, if Λ_σ receives as input the multisets of types of the states $s_1, \ldots, s_i$, then it returns the type of the task of s_i picked up by the scheduler. The following lemma, an easy consequence of the definitions, allows us to identify an online scheduler σ with the function Λ_σ.

Lemma 4.1. *Let* σ_1, σ_2 *be online schedulers. If* $\Lambda_{\sigma_1} = \Lambda_{\sigma_2}$, *then* $\Pr[S^{\sigma_1} = k] = \Pr[S^{\sigma_2} = k]$ *for all* $k \geq 1$.

Let $X^{(i)} = \Lambda_\sigma(\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(i)})$, i.e., $X^{(i)}$ is the type picked up at the i-th step. Then $X^{(i)}$ is randomly replaced by new types according to the distribution on the transition rules. More precisely, if $\boldsymbol{r}^{(i)} := \boldsymbol{z}^{(i+1)} - \boldsymbol{z}^{(i)} + X^{(i)}$, then $\Pr\left[\boldsymbol{r}^{(i)} = \alpha \mid X^{(i)} = X\right] = \sum_{X \overset{p}{\hookrightarrow} \alpha} p$.

A Normal Form for Task Systems. It is convenient to introduce a normal form for task systems, which allows us to formulate our results more succinctly and clearly. For every scheduler of the normal form we can find a scheduler of the original system with nearly the same properties. A type is called compact if, loosely speaking, it can eventually reproduce. Formally, a type W is *compact* if there is a rule $X \hookrightarrow \langle Y, Z \rangle$ such that X is reachable from W. A task system is *compact* if all its types are compact. A non-compact task system can be compacted by iterating the following procedure: remove all rules with non-compact types on the left hand side, and remove all occurrences of non-compact types on the right hand side of all rules.

Proposition 4.2. *Let us denote by Γ' the set of all task types removed from Δ by the above compacting procedure and let $|\Gamma'| = \ell$. If $X_0 \in \Gamma'$, then there is a scheduler σ such that $S^\sigma \leq \ell$.*

Assume that $X_0 \notin \Gamma'$. Let Δ' be the compacted version of Δ (i.e., $\Gamma \setminus \Gamma'$ is the set of task types of Δ'). Every scheduler σ' for Δ' can be transformed into a scheduler σ for Δ such that for all k

$$\mathrm{Pr}\left[S^{\sigma',\Delta'} \geq k\right] \leq \mathrm{Pr}\left[S^{\sigma,\Delta} \geq k\right] \leq \mathrm{Pr}\left[S^{\sigma',\Delta'} \geq k - \ell\right] .$$

(The second superscript of S indicates the task system on which the scheduler operates.)

Notice that computing σ from σ' is easy: σ acts like σ' but gives preferences to the types that have been (first) eliminated during the compacting procedure.
Further assumption: From now on we assume that task systems are compact.

4.1 Tail Bounds for Online Schedulers

The following main theorem gives computable lower and upper bounds which hold uniformly for all online schedulers σ.

Theorem 4.3. *Let Δ be subcritical. Let $\boldsymbol{v}, \boldsymbol{w} \in (1, \infty)^\Gamma$ be vectors with $\boldsymbol{f}(\boldsymbol{v}) \leq \boldsymbol{v}$ and $\boldsymbol{f}(\boldsymbol{w}) \geq \boldsymbol{w}$. Such vectors exist and can be computed in polynomial time. Denote by $\boldsymbol{v}_{min}$ and $\boldsymbol{w}_{max}$ the least component of $\boldsymbol{v}$ and the greatest component of $\boldsymbol{w}$, respectively. Then*

$$\frac{\boldsymbol{w}_{X_0} - 1}{\boldsymbol{w}_{max}^{k+2} - 1} \leq \mathrm{Pr}[S^\sigma \geq k] \leq \frac{\boldsymbol{v}_{X_0} - 1}{\boldsymbol{v}_{min}^{k} - 1} \text{ for all online schedulers } \sigma.$$

Proof sketch. Choose $h > 1$ and $\boldsymbol{u} \in (0, \infty)^\Gamma$ such that $h^{\boldsymbol{u}_X} = \boldsymbol{v}_X$ for all $X \in \Gamma$. Define for all $i \geq 1$ the variable $m^{(i)} = \boldsymbol{z}^{(i)} \bullet \boldsymbol{u}$ where "$\bullet$" denotes the scalar product, i.e., $m^{(i)}$ measures the number of tasks at time i weighted by types according to $\boldsymbol{u}$. One can show that $h^{m^{(1)}}, h^{m^{(2)}}, \ldots$ is a supermartingale for any online scheduler σ, and, using the Optional Stopping Theorem [33], that $\mathrm{Pr}\left[\sup_i m^{(i)} \geq x\right] \leq (\boldsymbol{v}_{X_0} - 1)/(h^x - 1)$ for all x (see the appendix for the details and [16, 31] for a similar argument on random walks). As each type has at least weight $\boldsymbol{u}_{min}$, we have that $S^\sigma \geq k$ implies $\sup_i m^{(i)} \geq k\boldsymbol{u}_{min}$. Hence $\mathrm{Pr}[S^\sigma \geq k] \leq \mathrm{Pr}\left[\sup_i m^{(i)} \geq k\boldsymbol{u}_{min}\right] \leq (\boldsymbol{v}_{X_0} - 1)/(\boldsymbol{v}_{min}^k - 1)$. The lower bound is shown similarly. $\square$

Theorem 4.3 stakes out the "playing field" in which all online schedulers perform. A comparison of the lower bound with Corollary 3.5 proves that the asymptotic performance of any online scheduler σ is far away from that of the optimal offline scheduler: the ratio $\mathrm{Pr}[S^\sigma \geq k] / \mathrm{Pr}[S^{op} \geq k]$ is unbounded.
Example 4.4. Consider again the task system with pgf $f(x) = px^2 + q$. For $p < 1/2$ the pgf has two fixed points, 1 and q/p. In particular, $q/p > 1$, so q/p can be used to obtain both an the upper and a lower bound for online schedulers. Since there is only one type of tasks, vectors have only one component, and the maximal and minimal components coincide; moreover, in this case the exponent $k + 2$ of the lower bound can be improved to k. So the upper and lower bounds coincide, and we get $\mathrm{Pr}[S^\sigma \geq k] = \frac{q/p-1}{(q/p)^k-1}$ for every online scheduler σ. In particular, as one intuitively expects, all online schedulers are equivalent.[2]

[2] For this example $\mathrm{Pr}[S^\sigma \geq k]$ can also be computed by elementary means.

Notice that any vector $\boldsymbol{v}$ satisfying $\boldsymbol{f}(\boldsymbol{v}) \leq \boldsymbol{v}$ leads to an upper bound on the performance of the scheduler. So we can try to compute the vector $\boldsymbol{v}$ leading to the tightest bound. In Appendix E we show how to compute, in polynomial time, an ϵ-approximation of $\sup\{\boldsymbol{v}_{min} \mid \boldsymbol{f}(\boldsymbol{v}) \leq \boldsymbol{v}\}$ for the class of *continuing task systems*. A task system is continuing if for every rule $X \xhookrightarrow{p} \langle Y, Z\rangle$ we have $Y = X$ or $Z = X$. Intuitively, in a continuing task system a task does not change its type when it spawns a new task.

4.2 Tail Bounds for Light-First Schedulers

We present a class of online schedulers for which a sharper upper bound than the one given by Theorem 4.3 can be proved. Intuitively, a good heuristic is to pick the task with the smallest expected completion time. If we compute a vector $\boldsymbol{v}$ with $\boldsymbol{f}(\boldsymbol{v}) \leq \boldsymbol{v}$ in polynomial time according to the proof of Theorem 4.3, then the type X_{min} for which $\boldsymbol{v}_{X_{min}} = \boldsymbol{v}_{min}$ holds turns out to be the type with smallest expected completion time. This suggests choosing the active type X with smallest component in $\boldsymbol{v}$. So we look at $\boldsymbol{v}$ as a vector of weights, and always choose the lightest active type. For this scheduler we obtain two different upper bounds.

Given a vector $\boldsymbol{v}$ with $\boldsymbol{f}(\boldsymbol{v}) \leq \boldsymbol{v}$ we denote by $\sqsubseteq$ a total order on Γ such that whenever $X \sqsubseteq Y$ then $\boldsymbol{v}_X \leq \boldsymbol{v}_Y$. If $X \sqsubseteq Y$, then we say that X is lighter than Y. The *$\boldsymbol{v}$-light-first scheduler* is an online scheduler that, in each step, picks a task of the lightest type available in the pool according to $\boldsymbol{v}$. Theorem 4.5 below strengthens the upper bound of Theorem 4.3 for light-first schedulers. For the second part of Theorem 4.5 we use the notion of *$\boldsymbol{v}$-accumulating types*. A type $X \in \Gamma$ is $\boldsymbol{v}$-accumulating if for every $k \geq 0$ the $\boldsymbol{v}$-light-first scheduler has a nonzero probability of reaching a state with at least k tasks of type X in the pool.

Theorem 4.5. *Let Δ be subcritical and $\boldsymbol{v} \in (1,\infty)^{\Gamma}$ with $\boldsymbol{f}(\boldsymbol{v}) \leq \boldsymbol{v}$. Let σ be a $\boldsymbol{v}$-light-first scheduler. Let $\boldsymbol{v}_{minmax} := \min_{X \hookrightarrow \langle Y,Z\rangle} \max\{\boldsymbol{v}_Y, \boldsymbol{v}_Z\}$ (here the minimum is taken over all transition rules with two types on the right hand side). Then $\boldsymbol{v}_{minmax} \geq \boldsymbol{v}_{min}$ and for all $k \geq 1$*

$$\Pr[S^\sigma \geq k] \leq \frac{\boldsymbol{v}_{X_0} - 1}{\boldsymbol{v}_{min}\boldsymbol{v}_{minmax}^{k-1} - 1}\,.$$

Moreover, let $\boldsymbol{v}_{minacc} := \min\{\boldsymbol{v}_X \mid X \in \Gamma,\ X \text{is } \boldsymbol{v}\text{-accumulating}\}$. Then $\boldsymbol{v}_{minacc} \geq \boldsymbol{v}_{minmax}$, $\boldsymbol{v}_{minacc}$ can be computed in polynomial time, and there is an integer ℓ such that for all $k \geq \ell$

$$\Pr[S^\sigma \geq k] \leq \frac{\boldsymbol{v}_{X_0} - 1}{\boldsymbol{v}_{min}^{\ell}\boldsymbol{v}_{minacc}^{k-\ell} - 1}\,.$$

Proof sketch. Recall the proof sketch of Theorem 4.3 where we used that $S^\sigma \geq k$ implies $\sup_i m^{(i)} \geq k\boldsymbol{u}_{min}$, as each type has at least weight $\boldsymbol{u}_{min}$. Let ℓ be such that no more than ℓ tasks of non-accumulating type can be in the pool at the same time. Then $S^\sigma \geq k$ implies $\sup_i m^{(i)} \geq \ell\boldsymbol{u}_{min} + (k-\ell)\boldsymbol{u}_{minacc}$ which leads to the final inequality of Theorem 4.5 in a way analogous to the proof sketch of Theorem 4.3. □

Intuitively, a light-first scheduler "works against" light tasks by picking them as soon as possible. In this way it may be able to avoid the accumulation of some light types, so it may achieve $\boldsymbol{v}_{minacc} > \boldsymbol{v}_{min}$. This is illustrated in the following example.

Example 4.6. Consider the task system with two types of tasks and pgfs $x = a_2xy + a_1y + a_0$ and $y = b_2xy + b_1y + b_0$, where $a_2 + a_1 + a_0 = 1 = b_2 + b_1 + b_0 = 1$. The system is subcritical if $a_1b_2 < a_2b_1 - a_2 + b_0$. The pgfs have a greatest fixed point $\boldsymbol{v}$ with

$$\boldsymbol{v}_X = (1 - a_2 - b_1 - a_1b_2 + a_2b_1)/b_2 \quad \text{and} \quad \boldsymbol{v}_Y = (1 - b_1 - b_2)/(a_2 + a_1b_2 - a_2b_1)\,.$$

We have $\boldsymbol{v}_X \leq \boldsymbol{v}_Y$ iff $a_2 - b_2 \leq a_2b_1 - a_1b_2$, and so the light-first scheduler chooses X before Y if this condition holds, and Y before X otherwise. We show that the light-first scheduler is asymptotically optimal. Assume w.l.o.g. $\boldsymbol{v}_X \leq \boldsymbol{v}_Y$. Then X is not accumulating (because X-tasks are picked as soon as they are created), and so $\boldsymbol{v}_{minacc} = \boldsymbol{v}_Y$. So the upper bound for the light-weight scheduler yields a constant c_2 such that $\Pr[S^\sigma \geq k] \leq c_2/\boldsymbol{v}_Y^k$. But the general lower bound for arbitrary online schedulers states that there is a constant c_1 such that $\Pr[S^\sigma \geq k] \geq c_1/\boldsymbol{v}_Y^k$, so we are done.

4.3 Tail Bounds for Depth-first Schedulers

Space-efficient scheduling of multithreaded computations has received considerable attention [27, 6, 2, 1]. The setting of these papers is slightly different from ours, because they assume data dependencies among the threads, which may cause a thread to wait for a result from another thread. In this sense our setting is similar to that of [22], where, in thread terminology, the threads can execute independently. Most results of [27, 6, 2, 1] are for *depth-first* computations, in which, loosely speaking, if thread A has to wait for thread B, then B is a descendant of A (i.e., B was spawned by A or by a descendant of A). As observed in [6, 27], the optimal scheduler for this class of computations is the one that, when A spawns B, interrupts the execution of A and continues with B; this scheduler (which is online) produces the familiar stack-based execution.

In this section we study the performance of this scheduler. In our setting, this corresponds to studying *depth-first schedulers*. A depth-first scheduler σ_λ is given in terms of a function λ that assigns to each rule $X \hookrightarrow \langle Y, Z\rangle$ either YZ or ZY, i.e., λ fixes an order on the tasks of the right-hand side. Intuitively, if the function assigns YZ to X, this means that Z models the continuation of the thread X, while Y models a new thread for whose termination Z waits. Formally, if $X \hookrightarrow \alpha$ is a rule in the task system, then $\lambda(X \hookrightarrow \alpha) = \beta$ where $\beta \in \Gamma^*$ and α is the Parikh image of β (i.e., a multiset of task types occurring in β such that the number of occurrences of any task type X in β is the same as in α).

The depth-first scheduler σ_λ keeps as an internal data structure a word $w \in \Gamma^*$, a "stack", such that the Parikh image of w is the multiset of the task types in the pool. If $w = Xw'$ for some $w' \in \Gamma^*$, then σ picks X. Assume that a transition rule $X \hookrightarrow \alpha$ "fires". Then σ_λ replaces Xw' by $\beta w'$ where $\beta = \lambda(X \hookrightarrow \alpha)$.

In the rest of the section we analyze S^σ for a fixed depth-first scheduler σ. Define for all vectors $\boldsymbol{u}, \boldsymbol{v}$ the vectors $L(\boldsymbol{u})$ and $Q(\boldsymbol{u}, \boldsymbol{v})$ such that for all $X \in \Gamma$

$$L(\boldsymbol{u})_X := \sum_{X \overset{p}{\hookrightarrow} Y} p\boldsymbol{u}_Y \qquad \text{and} \qquad Q(\boldsymbol{u}, \boldsymbol{v})_X := \sum_{X \overset{p}{\hookrightarrow} YZ} p\boldsymbol{u}_Y\boldsymbol{u}_Z \,.$$

Note that the sums extend over the rules after applying λ. Also note that L is a linear vector function and we view it as a matrix whose rows and columns are indexed with Γ. Furthermore, we write $Q(\cdot, \boldsymbol{v})$ and $Q(\boldsymbol{u}, \cdot)$ for the matrices with $Q(\cdot, \boldsymbol{v})\boldsymbol{u} = Q(\boldsymbol{u}, \boldsymbol{v}) = Q(\boldsymbol{u}, \cdot)\boldsymbol{v}$.
Our main theorem determines the exact asymptotic behavior of $\Pr[S^\sigma \geq k]$ for a depth-first scheduler σ:

Theorem 4.7. *Let Δ be subcritical and σ be any depth-first scheduler. Let ρ be the spectral radius of $(I - L - Q(\mathbf{1}, \cdot))^{-1}Q(\cdot, \mathbf{1})$. Then $0 < \rho < 1$ and $\Pr[S^\sigma \geq k] \in \Theta(\rho^k)$, i.e, there are $c, C > 0$ such that $c\rho^k \leq \Pr[S^\sigma \geq k] \leq C\rho^k$ for all k.*

Proof sketch. The proof idea is to compute $\Pr[S^\sigma_X \geq k]$ for all $X \in \Gamma$ at the same time. To this end, we define, for all $k \geq 1$, the vector $\boldsymbol{s}[k] \in [0, 1]^\Gamma$ such that $\boldsymbol{s}[k]_X = \Pr[S^\sigma_X \geq k]$ for all X. The following recurrence holds for $\boldsymbol{s}[k]$:

Lemma 4.8. *Let $A[k] := L + Q(\mathbf{1} - \boldsymbol{s}[k], \cdot)$. Then $(I - A[k])^{-1}$ exists and for all $k \geq 1$*

$$\boldsymbol{s}[k+1] = A[k]\boldsymbol{s}[k+1] + Q(\cdot, \mathbf{1})\boldsymbol{s}[k] = (I - A[k])^{-1}Q(\cdot, \mathbf{1})\boldsymbol{s}[k] \,.$$

The upper bound of Theorem 4.7 follows by iterating Lemma 4.8. The lower bound is involved, and heavily relies on the Perron-Frobenius theorem for nonnegative matrices. See the appendix for a full proof. □

Note that one can approximate the spectral radius in polynomial time using a binary search which uses the fact that the spectral radius of a nonnegative matrix M is at least r if and only if $M\boldsymbol{x} \geq r\boldsymbol{x}$ holds for a nonnegative, nonzero vector $\boldsymbol{x}$ (see e.g. Thm. 2.1.11 of [5] and cf. [14]), a condition that can be checked in polynomial time with linear programming. Observe also that Lemma 4.8 shows that $\boldsymbol{s}[k+1]$ can be computed from $\boldsymbol{s}[k]$ by solving a linear equation system. This requires $\mathcal{O}(|\Gamma|^3)$ arithmetical operations, so one can compute $\Pr[S^\sigma = k]$ in time $\mathcal{O}(k \cdot |\Gamma|^3)$ in the unit-cost model, cf. Corollary 3.4.

5 Expectations

In this section we study the expected completion space, i.e., the expectation $\mathbb{E}[S^\sigma]$ for both offline and online schedulers. Fix a task system $\Delta = (\Gamma, \hookrightarrow, Prob, X_0)$.

Optimal (Offline) Schedulers. The results of Section 3 allow to efficiently approximate the expectation $\mathbb{E}[S^{op}]$. Recall that for any random variable R with values in the natural numbers we have $\mathbb{E}[R] = \sum_{i=1}^{\infty} \Pr[R \geq i]$. So we can (under-) approximate $\mathbb{E}[R]$ by $\sum_{i=1}^{k} \Pr[R \geq i]$ for finite k. We say that k *terms compute* b *bits of* $\mathbb{E}[S^{op}]$ if $\mathbb{E}[S^{op}] - \sum_{i=0}^{k-1}(1 - \boldsymbol{\nu}_{X_0}^{(i)}) \leq 2^{-b}$.

Theorem 5.1. *The expectation $\mathbb{E}[S^{op}]$ is finite (no matter whether Δ is critical or subcritical). Moreover, $\mathcal{O}(b)$ terms compute b bits of $\mathbb{E}[S^{op}]$. If the task system Δ is subcritical, then $\log_2 b + \mathcal{O}(1)$ terms compute b bits of $\mathbb{E}[S^{op}]$. Finally, computing k terms takes time $\mathcal{O}(k \cdot |\Gamma|^3)$ in the unit cost model.*

Online Schedulers. The main result for online schedulers states that the finiteness of $\mathbb{E}[S^\sigma]$ does not depend on the choice of the online scheduler σ. It is easy to see that if Δ is subcritical, then every online scheduler has finite expected completion time. We show:

Theorem 5.2. *If Δ is critical, then $\mathbb{E}[S^\sigma]$ is infinite for every online scheduler σ.*

Proof sketch. For this sketch we focus on the case where X_0 is reachable from every type. By Proposition 2.5 the spectral radius of $\boldsymbol{f}'(\mathbf{1})$ equals 1. Then Perron-Frobenius theory guarantees the existence of a vector $\boldsymbol{u}$ with $\boldsymbol{f}'(\mathbf{1})\boldsymbol{u} = \boldsymbol{u}$ and $\boldsymbol{u}_X > 0$ for all X. Using a martingale argument, similar to the one of Theorem 4.3, one can show that the sequence $m^{(1)}, m^{(2)}, \ldots$ with $m^{(i)} := \boldsymbol{z}^{(i)} \bullet \boldsymbol{u}$ is a martingale for every scheduler σ, and, using the Optional-Stopping Theorem, that $\Pr[S^\sigma \geq k] \geq \boldsymbol{u}_{X_0}/(k+2)$. So we have $\mathbb{E}[S^\sigma] = \sum_{k=1}^{\infty} \Pr[S^\sigma \geq k] \geq \sum_{k=1}^{\infty} \boldsymbol{u}_{X_0}/(k+2) = \infty$. □

Since we can decide in polynomial time whether a system is subcritical or critical, we can do the same to decide on the finiteness of the expected completion time.

Depth-first Schedulers. We show how to approximate $\mathbb{E}[S^\sigma]$ for a given depth-first scheduler σ and a subcritical Δ. Again, we approximate $\mathbb{E}[S^\sigma]$ by $\sum_{i=1}^{k} \Pr[S^\sigma \geq i]$ for finite k. The following theorem shows that this is efficient. (Recall for the following statement that the 1-norm $\|\boldsymbol{v}\|_1$ of a vector $\boldsymbol{v}$ is the sum of the absolute values of its components, and the norm $\|M\|_1$ of a matrix M is the maximal 1-norm of its columns.)

Theorem 5.3. *Let Δ be subcritical, and let $B := (L + Q(\mathbf{1}, \cdot))^* Q(\cdot, \mathbf{1})$. Then $(I - B)^{-1}$ exists and $\mathbb{E}[S^\sigma] - u[k] \leq \left\|(I - B)^{-1}\right\|_1 \|\boldsymbol{s}[k]\|_1$ for all $k \geq 1$, where $u[k] := \sum_{i=1}^{k} \boldsymbol{s}[i]_{X_0} = \sum_{i=1}^{k} \Pr[S^\sigma \geq i]$. Hence, $\mathcal{O}(b)$ terms compute b bits of $\mathbb{E}[S^\sigma]$. Finally, computing k terms takes time $\mathcal{O}(k \cdot |\Gamma|^3)$ in the unit cost model.*

Online Schedulers minimizing expected completion space. We conclude the section with some results about online schedulers that minimize the expected completion space. First we prove that they always exist. Then we show that, however, they require infinite memory.

Theorem 5.4. *There is an online scheduler σ such that $\mathbb{E}[S^\sigma] = \inf_{\{\pi | \pi \text{ is online}\}} \mathbb{E}[S^\pi]$.*

An online scheduler σ requires *finite memory* if there is a deterministic finite state automaton $\mathcal{A}$ over an alphabet Σ and a function $h\colon \mathbb{N}^\Gamma \to \Sigma$ such that the value of $\Lambda_\sigma(\boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(i)})$ depends only on $\boldsymbol{c}^{(i)}$ and on the state of $\mathcal{A}$ after reading $h(\boldsymbol{c}^{(1)}) \cdots h(\boldsymbol{c}^{(i)})$.

Theorem 5.5. *For sufficiently small p and r (it suffices to choose $r := 10^{-5}$ and $p := \frac{1}{2}r$), any online scheduler that minimizes the expected completion space of the following task system requires infinite memory:*

$$X \xhookrightarrow{1/8} \langle X, X \rangle \quad X \xhookrightarrow{1/8} \langle Y, Z \rangle \quad X \xhookrightarrow{3/4} \emptyset \qquad Z \xhookrightarrow{r} \langle U, U \rangle \quad Z \xhookrightarrow{1-r} \emptyset$$
$$Y \xhookrightarrow{p} \langle Z, Z \rangle \qquad Y \xhookrightarrow{1-p} \emptyset \qquad\qquad\qquad U \xhookrightarrow{1} \emptyset$$

6 Conclusions

We have initiated the study of scheduling tasks that can stochastically generate other tasks. We have provided strong results on the performance of both online and offline schedulers for the case of one processor and task systems with completion probability 1. While we profited from the theory of branching processes, the theory considers (in computer science terms) systems with an unbounded number of processors, and therefore many questions had not been addressed before or even posed.

References

1. K. Agrawal, C.E. Leiserson, Y. He, and W.J. Hsu. Adaptive work-stealing with parallelism feedback. *ACM TOCS*, 26(3), 2008.
2. N.S. Arora, R.D. Blumofe, and C.G. Plaxton. Thread scheduling for multiprogrammed microprocessors. *Theory of Computing Systems*, 34:115–144, 2001.
3. K.B. Athreya. On the maximum sequence of a critical branching process. *Annals of Probability*, 16:502–507, 1988.
4. K.B. Athreya and P.E. Ney. *Branching Processes*. Springer, 1972.
5. A. Berman and R.J. Plemmons. *Nonnegative matrices in the mathematical sciences*. Academic Press, 1979.
6. R.D. Blumofe and C.E. Leiserson. Scheduling multithreaded computations by work stealing. *Journal of the ACM*, 46(5):720–748, 1999.
7. K.A. Borovkov and V.A. Vatutin. On distribution tails and expectations of maxima in critical branching processes. *Journal of Applied Probability*, 33(3):614–622, 1996.
8. J.L. Coolidge. The gambler's ruin. *Annals of Mathematics*, 10(4):181–192, 1909.
9. J. Esparza, S. Kiefer, and M. Luttenberger. An extension of Newton's method to ω-continuous semirings. In *DLT'07*, LNCS 4588, pages 157–168. Springer, 2007.
10. J. Esparza, S. Kiefer, and M. Luttenberger. On fixed point equations over commutative semirings. In *STACS'07*, LNCS 4397, pages 296–307. Springer, 2007.
11. J. Esparza, S. Kiefer, and M. Luttenberger. Convergence thresholds of Newton's method for monotone polynomial equations. In *STACS 2008*, pages 289–300, 2008.
12. J. Esparza, A. Kučera, and R. Mayr. Quantitative analysis of probabilistic pushdown automata: Expectations and variances. In *LICS 2005*, pages 117–126. IEEE Computer Society, 2005.
13. J. Esparza, A. Kučera, and R. Mayr. Model checking probabilistic pushdown automata. In *LICS 2004*, pages 12–21. IEEE Computer Society, 2004.
14. K. Etessami and M. Yannakakis. Recursive markov chains, stochastic grammars, and monotone systems of nonlinear equations. *Journal of the ACM*, 56(1):1–66, 2009. Earlier version appeared in STACS'05, pp. 340–352.
15. R. Fagin, A.R. Karlin, J. Kleinberg, P. Raghavan, S. Rajagopalan, R. Rubinfeld, M. Sudan, and A. Tomkins. Random walks with "back buttons". *Annals of Applied Probability*, 11(3):810–862, 2001.
16. W. Feller. *An introduction to probability theory and its applications*, volume I. John Wiley & Sons, 1968.
17. M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer, second edition, 1993.
18. T.E. Harris. *The Theory of Branching Processes*. Springer, 1963.
19. R.A. Horn and C.A. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.
20. S. Janson. Functional limit theorems for multitype branching processes and generalized Pólya urns. *Stochastic Processes and their Applications*, 110:177–245, 2004.
21. N.L. Johnson and S. Kotz. *Urn Models and Their Application*. John Wiley & Sons, 1977.
22. R.M. Karp and Y. Zhang. Randomized parallel algorithms for backtrack search and branch-and-bound computation. *Journal of the ACM*, 40(3):765–789, 1993.
23. S. Kiefer, M. Luttenberger, and J. Esparza. On the convergence of Newton's method for monotone systems of polynomial equations. In *STOC 2007*, pages 217–226. ACM, 2007.
24. S. Kotz, H. Mahmoud, and P. Robert. On generalized Pólya urn models. *Statistics & Probability Letters*, 49:163–173, 2000.
25. T. Lindvall. On the maximum of a branching process. *Scandinavian Journal of Statistics*, 3:209–214, 1976.
26. H. Mahmoud. *Pólya Urn Models*. CRC Press, 2008.
27. G.J. Narlikar and G.E. Belloch. Space-efficient scheduling of nested parallelism. *ACM TOPLAS*, 21(1):138–173, 1999.
28. O. Nerman. On the maximal generation size of a non-critical galton-watson process. *Scandinavian Journal of Statistics*, 4(3):131–135, 1977.
29. J.M. Ortega and W.C. Rheinboldt. *Iterative solution of nonlinear equations in several variables*. Academic Press, 1970.
30. A.G. Pakes. A limit theorem for the maxima of the para-critical simple branching process. *Advances in Applied Probability*, 30:740–756, 1998.
31. F. Spitzer. *Principles of Random Walk*. Springer, 1976.
32. A. Spătaru. A maximum sequence in a critical multitype branching process. *Journal of Applied Probability*, 28(4):893–897, 1991.
33. D. Williams. *Probability with Martingales*. Cambridge University Press, 1995.

A Proofs of Section 2

A.1 Proof of Proposition 2.5

Proposition 2.5 ([18, 14]). *Let Δ be a task system with pgf $\boldsymbol{f}$. Denote by $\boldsymbol{f}'(\mathbf{1})$ the Jacobian matrix of partial derivatives of $\boldsymbol{f}$ evaluated at $\mathbf{1}$. If Δ is critical, then the spectral radius of $\boldsymbol{f}'(\mathbf{1})$ is equal to 1; otherwise it is strictly less than 1. It can be decided in polynomial time whether Δ is critical.*

Proof. One can show (see e.g. [13]) that $\mathbb{E}[T_X]$ is the X-component of the least nonnegative fixed point of $\boldsymbol{f}'(\mathbf{1})\boldsymbol{x} + \mathbf{1}$, i.e., the X-component of the (componentwise) least vector $\boldsymbol{x} \in [0, \infty]^\Gamma$ with $\boldsymbol{x} = \boldsymbol{f}'(\mathbf{1})\boldsymbol{x} + \mathbf{1}$. This least fixed point is given by $\sum_{i=0}^{\infty}(\boldsymbol{f}'(\mathbf{1}))^i\mathbf{1}$, a series that may or may not converge. It is a standard fact (see e.g. [19]) that the series converges iff $\rho(\boldsymbol{f}'(\mathbf{1})) < 1$ holds for the spectral radius $\rho(\boldsymbol{f}'(\mathbf{1}))$ of $\boldsymbol{f}'(\mathbf{1})$.

Assume first that Δ is subcritical. Then the above series must converge, so we have $\rho(\boldsymbol{f}'(\mathbf{1})) < 1$ in this case. Now assume that Δ is critical. Then the above series must diverge, so we have $\rho(\boldsymbol{f}'(\mathbf{1})) \geq 1$. On the other hand, in [11, 14] it is shown that $\rho(\boldsymbol{f}'(\mathbf{1})) \leq 1$. (More precisely, it is shown there that $\rho(\boldsymbol{f}'(\boldsymbol{y})) < 1$ holds for $\boldsymbol{y}$ that are strictly less than the least fixed point of $\boldsymbol{f}$. By continuity of eigenvalues, $\rho(\boldsymbol{f}'(\boldsymbol{y})) \leq 1$ also holds for the least fixed point of $\boldsymbol{f}$ which is $\mathbf{1}$ according to the proof of Proposition 2.3.) Hence we have $\rho(\boldsymbol{f}'(\mathbf{1})) = 1$.

In order to decide on the criticality, it thus suffices to decide whether the spectral radius of $\boldsymbol{f}'(\mathbf{1})$ is ≥ 1. This condition holds iff $\boldsymbol{f}'(\mathbf{1})\boldsymbol{x} \geq \boldsymbol{x}$ holds for a nonnegative, nonzero vector $\boldsymbol{x}$ (see e.g. Thm. 2.1.11 of [5] and cf. [14]). This can be checked in polynomial time with linear programming. □

B Proofs of Section 3

B.1 Proof of Proposition 3.1

Proposition 3.1. *Let t be a family tree. Then*

$$S^{op}(t) = \begin{cases} \min\left\{\begin{array}{l}\max\{S^{op}(t_0)+1, S^{op}(t_1)\},\\ \max\{S^{op}(t_0), S^{op}(t_1)+1\}\end{array}\right\} & \textit{if } t \textit{ has two children } t_0, t_1 \\ S^{op}(t_0) & \textit{if } t \textit{ has exactly one child } t_0 \\ 1 & \textit{if } t \textit{ has no children.} \end{cases}$$

Proof. Recall the proof sketch from the main body of the paper. We detail the argument why one of the two given scheduling strategies is optimal, i.e., we argue why the scheduler cannot save space by interleaving the schedulings for t_0 and t_1.

Consider an optimal scheduling of t. W.l.o.g. the task t_0 terminates first. Then at least one t_1-task sticks around during the whole derivation of t_0. So this scheduling needs space of at least $S^{op}(t_0) + 1$. Obviously, any scheduling of t needs space of at least $S^{op}(t_1)$. So the optimal scheduler needs space of at least $\max\{S^{op}(t_0) + 1, S^{op}(t_1)\}$. But this lower bound is matched by the scheduling strategy given in the main body of the paper. □

B.2 Proof of Theorem 3.2

Theorem 3.2. $\Pr\left[S_X^{op} \leq k\right] = \boldsymbol{\nu}_X^{(k)}$ *for every type X and every $k \geq 0$.*

Proof. Let us inductively define the function ℓ on trees as follows.

$$\ell(t) := \begin{cases} 0 & \text{if } t \text{ has no children} \\ \ell(t_0)+1 & \text{if } t \text{ has one child} \\ \ell(t_0)+1 & \text{if } t \text{ has two children and } S^{op}(t_0) > S^{op}(t_1) \\ \ell(t_1)+1 & \text{if } t \text{ has two children and } S^{op}(t_0) < S^{op}(t_1) \\ 0 & \text{if } t \text{ has two children and } S^{op}(t_0) = S^{op}(t_1)\,. \end{cases}$$

With Proposition 3.1, $\ell(t)$ is the length of a longest path from the root to a descendant with the same S^{op}-value.

We proceed by induction on k. The base case $k=0$ is trivial. Let $k \geq 0$ and let t be an X-tree with $S^{op}(t) = k+1$. We have to show $\Pr\left[S^{op}_X = k+1\right] = \boldsymbol{\Delta}^{(k+1)}_X$ where

$$\boldsymbol{\Delta}^{(k+1)} = \sum_{i=0}^{\infty} \boldsymbol{f}'(\boldsymbol{\nu}^{(k)})^i \left(\boldsymbol{f}(\boldsymbol{\nu}^{(k)}) - \boldsymbol{\nu}^{(k)}\right)\,.$$

We show the following stronger claim:

$$\Pr\left[S^{op}_X(t) = k+1,\ \ell(t) = i\right] = \left(\boldsymbol{f}'(\boldsymbol{\nu}^{(k)})^i \left(\boldsymbol{f}(\boldsymbol{\nu}^{(k)}) - \boldsymbol{\nu}^{(k)}\right)\right)_X\,.$$

We proceed by an (inner) induction on i. For the induction base $i=0$ we first dispense with the case $k=0$. We have

$$\Pr\left[S^{op}_X(t) = 1,\ \ell(t) = 0\right] = \Pr[t \text{ has no children}]$$

because if t has one child then $\ell(t) \geq 1$ and if t has two children, then $S^{op}_X(t) \geq 2$. With the definition of $\boldsymbol{f}$ we obtain

$$\Pr\left[S^{op}_X(t) = 1,\ \ell(t) = 0\right] = \sum_{X \overset{p}{\hookrightarrow} \epsilon} p = \boldsymbol{f}_X(\boldsymbol{0}) = \boldsymbol{f}_X(\boldsymbol{\nu}^{(0)}) - \boldsymbol{\nu}^{(0)}_X\,.$$

Now we complete the induction base $i=0$ with the case $k \geq 1$. We have

$$\Pr\left[S^{op}_X(t) = k+1,\ \ell(t) = 0\right] = \Pr[t \text{ has two children},\ S^{op}(t_0) = S^{op}(t_1) = k] \tag{1}$$

because if t has one child, then $\ell(t) \geq 1$, and if t has no children, then $S^{op}_X(t) = 1$. Further we have by Proposition 3.1

$$\begin{aligned} \Pr\left[S^{op}_X(t) \leq k\right] = \sum_{X \overset{p}{\hookrightarrow} \langle Y,Z\rangle} & p \cdot \Big(\Pr\left[S^{op}_Y(t_0) \leq k\right] \Pr\left[S^{op}_Z(t_1) \leq k\right] \\ & \qquad - \Pr\left[S^{op}_Y(t_0) = k\right] \Pr\left[S^{op}_Z(t_1) = k\right]\Big) \\ & + \sum_{X \overset{p}{\hookrightarrow} Y} p \cdot \Pr\left[S^{op}_Y(t_0) \leq k\right] \\ & + \sum_{X \overset{p}{\hookrightarrow} \emptyset} p\,. \end{aligned} \tag{2}$$

Combining these equations we obtain

$$\Pr\left[S^{op}_X(t) = k+1,\ \ell(t) = 0\right] = \sum_{X \overset{p}{\hookrightarrow} \langle Y,Z\rangle} p \cdot \Pr\left[S^{op}_Y(t_0) = k\right] \Pr\left[S^{op}_Z(t_1) = k\right] \qquad \text{(by (1))}$$

$$
\begin{aligned}
&= \sum_{X \xhookrightarrow{p} \langle Y,Z\rangle} p \cdot \Pr\left[S_Y^{op}(t_0) \le k\right] \Pr\left[S_Z^{op}(t_1) \le k\right] && \text{(by (2))}\\
&\quad + \sum_{X \xhookrightarrow{p} Y} p \cdot \Pr\left[S_Y^{op}(t_0) \le k\right] + \sum_{X \xhookrightarrow{p} \epsilon} p \\
&\quad - \Pr\left[S_X^{op}(t) \le k\right] \\
&= \sum_{X \xhookrightarrow{p} \langle Y,Z\rangle} p \cdot \boldsymbol{\nu}_Y^{(k)} \boldsymbol{\nu}_Z^{(k)} && \text{(ind. hyp. on } k\text{)}\\
&\quad + \sum_{X \xhookrightarrow{p} Y} p \cdot \boldsymbol{\nu}_Y^{(k)} + \sum_{X \xhookrightarrow{p} \epsilon} p \\
&\quad - \boldsymbol{\nu}_X^{(k)} \\
&= \boldsymbol{f}_X(\boldsymbol{\nu}^{(k)}) - \boldsymbol{\nu}_X^{(k)} && \text{(def. of } \boldsymbol{f}\text{)}
\end{aligned}
$$

For the induction step, let $i \ge 0$. Then by Proposition 3.1 and the definition of ℓ

$$
\begin{aligned}
&\Pr\left[S_X^{op}(t) = k+1,\ \ell(t) = i+1\right] \\
&= \sum_{X \xhookrightarrow{p} \langle Y,Z\rangle} p \cdot \Big(\Pr\left[S_Y^{op}(t_0) \le k\right] \Pr\left[S_Z^{op}(t_1) = k+1,\ \ell(t_1) = i\right] \\
&\qquad\qquad + \Pr\left[S_Y^{op}(t_0) = k+1,\ \ell(t_0) = i\right] \Pr\left[S_Z^{op}(t_1) \le k\right]\Big) \\
&\quad + \sum_{X \xhookrightarrow{p} Y} p \cdot \Pr\left[S_Y^{op}(t_0) = k+1,\ \ell(t_0) = i\right] \\
&= \sum_{X \xhookrightarrow{p} \langle Y,Z\rangle} p \cdot \left(\boldsymbol{\nu}_Y^{(k)} \left(\boldsymbol{f}'(\boldsymbol{\nu}^{(k)})^i \left(\boldsymbol{f}(\boldsymbol{\nu}^{(k)}) - \boldsymbol{\nu}^{(k)}\right)\right)_Z \right.\\
&\qquad\qquad \left. + \left(\boldsymbol{f}'(\boldsymbol{\nu}^{(k)})^i \left(\boldsymbol{f}(\boldsymbol{\nu}^{(k)}) - \boldsymbol{\nu}^{(k)}\right)\right)_Y \boldsymbol{\nu}_Z^{(k)}\right) && \text{(ind. hyp. on } k, i\text{)}\\
&\quad + \sum_{X \xhookrightarrow{p} Y} p \cdot \left(\boldsymbol{f}'(\boldsymbol{\nu}^{(k)})^i \left(\boldsymbol{f}(\boldsymbol{\nu}^{(k)}) - \boldsymbol{\nu}^{(k)}\right)\right)_Y \\
&= \sum_{Y \in \Gamma} \boldsymbol{f}'_{XY}(\boldsymbol{\nu}^{(k)}) \left(\boldsymbol{f}'(\boldsymbol{\nu}^{(k)})^i \left(\boldsymbol{f}(\boldsymbol{\nu}^{(k)}) - \boldsymbol{\nu}^{(k)}\right)\right)_Y && \text{(def. of } \boldsymbol{f}\text{)}\\
&= \boldsymbol{f}'_X(\boldsymbol{\nu}^{(k)}) \boldsymbol{f}'(\boldsymbol{\nu}^{(k)})^i \left(\boldsymbol{f}(\boldsymbol{\nu}^{(k)}) - \boldsymbol{\nu}^{(k)}\right) \\
&= \left(\boldsymbol{f}'(\boldsymbol{\nu}^{(k)})^{i+1} \left(\boldsymbol{f}(\boldsymbol{\nu}^{(k)}) - \boldsymbol{\nu}^{(k)}\right)\right)_X .
\end{aligned}
$$

□

B.3 Proof of Corollary 3.5

Corollary 3.5. *For any task system Δ there are real numbers $c > 0$ and $0 < d < 1$ such that $\Pr\left[S_X^{op} \ge k\right] \le c \cdot d^k$ for all $k \in \mathbb{N}$. If Δ is subcritical, then there are real numbers $c > 0$ and $0 < d < 1$ such that $\Pr\left[S_X^{op} \ge k\right] \le c \cdot d^{2^k}$ for all $k \in \mathbb{N}$.*

Proof. By Theorem 3.2 we have $\Pr[S^{op} \geq k] = 1 - \boldsymbol{\nu}_{X_0}^{(k-1)} \leq 1 - \boldsymbol{\nu}_{X_0}^{(k)}$. So the corollary can be understood as a statement on the convergence speed of Newton's method for solving $\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{x})$. The fact that Newton's method started at $\mathbf{0}$ converges to $\mathbf{1}$ (the least fixed point of $\boldsymbol{f}$) is shown in [14].

For the subcritical case, observe that the matrix $I - \boldsymbol{f}'(\mathbf{1})$ is nonsingular because otherwise 1 would be an eigenvalue of $\boldsymbol{f}'(\mathbf{1})$ which would, together with Proposition 2.5, contradict the assumption that the task system is subcritical. For nonsingular systems, it is a standard fact (see e.g. [29]) that Newton's method converges quadratically. As $\Pr[S^{op} \geq k] \leq 1 - \boldsymbol{\nu}_{X_0}^{(k)}$, the statement follows.

For the general case (subcritical or critical) Newton's method for solving $\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{x})$ has been extensively studied in [23, 11] and it follows from there that there is a $c_1 \in (0, \infty)$ such that $1 - \boldsymbol{\nu}_X^{(k)} \leq c_1 \cdot 2^{-k/(n2^n)}$ where $n = |\Gamma|$, implying the statement. $\square$

C Proofs of Section 4

Lemma C.1. *Let σ be an online scheduler. For every family tree t the first $i \geq 1$ states of $\sigma(t)$ are uniquely determined by $\boldsymbol{z}^{(1)}(t), \ldots, \boldsymbol{z}^{(i)}(t)$. In particular, the function Λ_σ is well-defined.*

Proof. We proceed by induction on i. The case $i = 1$ is trivial. Let us consider $\boldsymbol{z}^{(1)}(t), \ldots, \boldsymbol{z}^{(i+1)}(t)$, and let $d = (s_1 \Rightarrow \cdots \Rightarrow s_i \Rightarrow s_{i+1})$ be a prefix of the derivation $\sigma(t)$. By induction, $s_1 \Rightarrow \cdots \Rightarrow s_i$ is completely determined by $\boldsymbol{z}^{(1)}(t), \ldots, \boldsymbol{z}^{(i)}(t)$. By the definition of online scheduler, $\sigma(t)[i]$ is completely determined by $s_1 \Rightarrow \cdots \Rightarrow s_i$ and $\boldsymbol{z}^{(1)}(t), \ldots, \boldsymbol{z}^{(i)}(t)$. Finally, there is a unique transition rule $L(\sigma(t)[i]) \hookrightarrow \alpha$ where $\alpha = \boldsymbol{z}^{(i+1)}(t) - \boldsymbol{z}^{(i)}(t) + \langle L(\sigma(t)[i])\rangle$. But then s_{i+1} is also uniquely determined.

Lemma C.2. *Let $\boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(i)} \in (\mathbb{N}^\Gamma)^+$ such that for every $1 \leq j < i$ the value $\Lambda_\sigma(\boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(j)})$ is defined. Then $\Pr\left[\bigwedge_{j=1}^{i} \boldsymbol{z}^{(j)} = \boldsymbol{c}^{(j)}\right] = \prod_{j=1}^{i-1} Prob(\Lambda_\sigma(\boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(j)}) \hookrightarrow \alpha_j)$ where for every $1 \leq j < i$ we have $\alpha_j = \boldsymbol{c}^{(j+1)} - \boldsymbol{c}^{(j)} + \langle \Lambda_\sigma(\boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(j)})\rangle$.*

Proof. Let us denote by $\mathcal{R}$ the set of all family trees t such that $\boldsymbol{z}^{(j)}(t) = \boldsymbol{c}^{(j)}$ for $1 \leq j \leq i$. By Lemma C.1, there is a derivation $d = s_1 \Rightarrow \cdots \Rightarrow s_i$ and a function $l : \bigcup_{j=1}^{i} s_j \to \Gamma$ such that for every $t = (N, L) \in \mathcal{R}$ we have that d is a prefix of $\sigma(t)$ and l coincides with l on the subtree $\bigcup_{j=1}^{i} s_j$. Let us denote by t^s the tree $\bigcup_{j=1}^{i} s_j$. Note that t^s is a subtree of every tree of $\mathcal{R}$ rooted in ϵ. Let us denote by $\mathcal{I}$ the set of all inner nodes of t^s. For every $v \in \mathcal{I}$, we denote by $child(v) := \langle l(va) \mid a \in \{0,1\}, va \in t^s\rangle$ the multiset of labels of children of the node v in t^s. Let us denote by $\mathcal{L}$ the set of all leaves of t^s. It follows directly from the definition of $\Pr$, that for all $t \in \mathcal{R}$ we have

$$\Pr[t] = \prod_{v \in \mathcal{I}} Prob(L(v) \hookrightarrow child(v)) \cdot \prod_{v \in \mathcal{L}} \Pr[t_v]$$

However, it follows directly from definitions that for every $v \in \mathcal{I}$ there is precisely one $1 \leq j < i$ such that $\sigma(t)[j] = v$, and then $L(v) = \Lambda_\sigma(\boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(j)})$ and $child(v) = \alpha_j$. Therefore,

$$\Pr[t] = \prod_{j=1}^{i-1} Prob(\Lambda_\sigma(\boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(j)}) \hookrightarrow \alpha_j) \cdot \prod_{v \in \mathcal{L}} \Pr[t_v]$$

Finally,

$$\sum_{t \in \mathcal{R}} \Pr[t] = \prod_{j=1}^{i-1} Prob(\Lambda_\sigma(\boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(j)}) \hookrightarrow \alpha_j) \cdot \prod_{v \in \mathcal{L}} \sum_{t' \in \mathcal{T}_{L(v)}} \Pr[t'] = \prod_{j=1}^{i-1} Prob(\Lambda_\sigma(\boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(j)}) \hookrightarrow \alpha_j)$$

Proof of Lemma 4.1

Lemma 4.1. *Let σ_1, σ_2 be online schedulers. If $\Lambda_{\sigma_1} = \Lambda_{\sigma_2}$, then $\Pr[S^{\sigma_1} = k] = \Pr[S^{\sigma_2} = k]$ for every $k \geq 1$.*

Proof. We denote by $\boldsymbol{z}_\lambda^{(i)}$ the variable $\boldsymbol{z}^{(i)}$ evaluated with respect to a given scheduler λ. Let us denote by A_{def} the set of all $\boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(i)} \in (\mathbb{N}^\Gamma)^+$ such that $\Lambda_{\sigma_1}(\boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(j)}) = \Lambda_{\sigma_2}(\boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(j)})$ is defined for all $1 \leq j \leq i-1$, and $\boldsymbol{c}^{(i)} = \mathbf{0}$. By Lemma C.2, for every $\boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(i)} \in A_{def}$ we have

$$\begin{aligned}
\Pr\left[\bigwedge_{j=1}^{i} \boldsymbol{z}_{\sigma_1}^{(j)} = \boldsymbol{c}^{(j)}\right] &= \prod_{j=1}^{i-1} Prob(\Lambda_{\sigma_1}(\boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(j)}) \hookrightarrow \alpha_j) \\
&= \prod_{j=1}^{i-1} Prob(\Lambda_{\sigma_2}(\boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(j)}) \hookrightarrow \alpha_j) \\
&= \Pr\left[\bigwedge_{j=1}^{i} \boldsymbol{z}_{\sigma_2}^{(i)} = \boldsymbol{c}^{(j)}\right]
\end{aligned}$$

However, then $\Pr[S^{\sigma_1} = k] = \Pr[S^{\sigma_2} = k]$ because the values of S^{σ_1} and S^{σ_2} are determined by the values of $\boldsymbol{z}_{\sigma_1}^{(1)}, \boldsymbol{z}_{\sigma_1}^{(2)}, \ldots$ and $\boldsymbol{z}_{\sigma_2}^{(1)}, \boldsymbol{z}_{\sigma_2}^{(2)}, \ldots$, and for all family trees t we have that a prefix of $\boldsymbol{z}_{\sigma_1}^{(1)}(t), \boldsymbol{z}_{\sigma_1}^{(2)}(t), \ldots$ and a prefix of $\boldsymbol{z}_{\sigma_2}^{(1)}(t), \boldsymbol{z}_{\sigma_2}^{(2)}(t), \ldots$ are in A_{def}.

C.1 Proof of Proposition 4.2

Proposition 4.2. *Let us denote by Γ' the set of all task types removed from Δ by the above compacting procedure and let $|\Gamma'| = \ell$. If $X_0 \in \Gamma'$, then there is a scheduler σ such that $S^\sigma \leq \ell$.*

Assume that $X_0 \notin \Gamma'$. Let Δ' be the compacted version of Δ (i.e., $\Gamma \setminus \Gamma'$ is the set of task types of Δ'). Every scheduler σ' for Δ' can be transformed into a scheduler σ for Δ such that for all k

$$\Pr\left[S^{\sigma',\Delta'} \geq k\right] \leq \Pr\left[S^{\sigma,\Delta} \geq k\right] \leq \Pr\left[S^{\sigma',\Delta'} \geq k - \ell\right] .$$

(The second superscript of S indicates the task system on which the scheduler operates.)

Proof. Let Δ_1 be a non-compact task system with a non-compact types Γ_{non}, and let Δ_0 be the (possibly non-compact) task system obtained from Δ_1 by removing all rules with non-compact types on the left hand side and all occurrences of non-compact types on the right hand side of all rules, i.e., Δ_0 is obtained from Δ_1 by performing the first iteration of the compacting procedure. Let σ_0 be a scheduler for Δ_0. Construct a scheduler σ_1 for Δ_1 as follows:

> The scheduler σ_1 acts exactly like σ_0 until one or two Γ_{non}-tasks are created at which point the width of the derivation may be increased by at most 1. Then σ_1 picks a Γ_{non}-task, say τ_1. Since the Γ_{non}-types are non-compact, σ_1 can complete τ_1 without further increasing the width. After τ_1 has been finished, there may be another Γ_{non}-task left, say τ_2, that was created at the time when τ_1 was created. If there is such a τ_2, then σ_1 completes τ_2 in the same way it has completed τ_1. After τ_1 (and possibly τ_2) have been completed, σ_1 resumes to act like σ_0.

It follows from this construction that the incorporation of the non-compact type Γ_{non} increases the width of a derivation by at most 1.

A straightforward induction on this construction shows for the statement of the proposition:

$$\Pr\left[S_X^{\sigma',\Delta'} \le k\right] \le \Pr\left[S_X^{\sigma,\Delta} \le k+\ell\right] \text{ for all } X \in \Gamma \setminus \Gamma'.$$

If $X_0 \in \Gamma'$, then the above construction also works. (It extends a scheduler operating on a possibly empty task system, but this poses no problems.) So, again by induction, we obtain a scheduler σ for Δ with $S_X^{\sigma,\Delta} \le \ell$ for all $X \in \Gamma'$.

It remains to show the inequality $\Pr\left[S_X^{\sigma',\Delta'} \ge k\right] \le \Pr\left[S_X^{\sigma,\Delta} \ge k\right]$, but this is clear because Δ' is obtained from deleting rules and types from Δ and σ is obtained by extending σ'. □

C.2 Proof of Theorem 4.3

We split the proof in several lemmata. With regard to the computation of a suitable vector $\boldsymbol{v}$ we first prove the following lemma.

Lemma C.3. *Let $\boldsymbol{u} \in [1,\infty)^\Gamma$ denote the vector of expected completion times, i.e., $\boldsymbol{u}_Y = \mathbb{E}[T_Y]$ for all $Y \in \Gamma$. Then $\boldsymbol{u}$ exists and is the unique solution of $\boldsymbol{x} = \boldsymbol{f}'(\mathbf{1})\boldsymbol{x} + \mathbf{1}$. Let $Q(\boldsymbol{u},\boldsymbol{u})$ denote the "quadratic part" of $\boldsymbol{f}(\boldsymbol{u})$, i.e., $(Q(\boldsymbol{u},\boldsymbol{u}))_X = \sum_{X \xhookrightarrow{p} YZ} p \cdot \boldsymbol{u}_Y \cdot \boldsymbol{u}_Z$ for all $X,Y,Z \in \Gamma$. Let $s := 1/q_{max} > 0$ where q_{max} is the largest component of $Q(\boldsymbol{u},\boldsymbol{u})$. Then for all $r \ge 0$ we have $\boldsymbol{f}(\mathbf{1}+r\boldsymbol{u}) \le \mathbf{1}+r\boldsymbol{u}$ iff $r \le s$.*

Using this lemma a suitable $\boldsymbol{v}$ can be found as follows: First compute $\boldsymbol{u}$ by solving $\boldsymbol{x} = \boldsymbol{f}'(\mathbf{1})\boldsymbol{x} + \mathbf{1}$. This yields $Q(\boldsymbol{u},\boldsymbol{u})$, and, consequently, s. With regard to the upper bound of the theorem we are interested in a $\boldsymbol{v}$ which is as large as possible, so pick $\boldsymbol{v} := \mathbf{1} + s\boldsymbol{u}$ (or $\boldsymbol{v} := \mathbf{1} + \frac{1}{2}s\boldsymbol{u}$ to be on the safe side). All steps can be performed in polynomial time.

Proof of the lemma. The fact that $\boldsymbol{u} = \boldsymbol{f}'(\mathbf{1})\boldsymbol{u} + \mathbf{1}$ exists and is the vector of expected completion times follows from the remarks made at the beginning of the proof of Proposition 2.5. Recall that the pgf $\boldsymbol{f}$ is a vector of polynomials of degree 2 with positive coefficients. So it can be written as

$$\boldsymbol{f}(\boldsymbol{x}) = Q(\boldsymbol{x},\boldsymbol{x}) + L\boldsymbol{x} + \boldsymbol{c}$$

where $Q(\boldsymbol{x},\boldsymbol{x})$ is the quadratic part of $\boldsymbol{f}(\boldsymbol{x})$. A straightforward calculation shows for all $r \in \mathbb{R}$ and $\boldsymbol{x} \in \mathbb{R}^\Gamma$

$$\begin{aligned} \boldsymbol{f}(\mathbf{1}+r\boldsymbol{x}) &= \boldsymbol{f}(\mathbf{1}) + r\boldsymbol{f}'(\mathbf{1})\boldsymbol{x} + r^2 Q(\boldsymbol{x},\boldsymbol{x}) && \text{(Taylor expansion)} \\ &= \mathbf{1} + r\boldsymbol{f}'(\mathbf{1})\boldsymbol{x} + r^2 Q(\boldsymbol{x},\boldsymbol{x}) && \text{(as } \boldsymbol{f}(\mathbf{1}) = \mathbf{1}\text{)}\,. \end{aligned}$$

For $\boldsymbol{u} = \boldsymbol{f}'(\mathbf{1})\boldsymbol{u} + \mathbf{1}$ it follows

$$\boldsymbol{f}(\mathbf{1}+r\boldsymbol{u}) = \mathbf{1} + r(\boldsymbol{u}-\mathbf{1}) + r^2 Q(\boldsymbol{u},\boldsymbol{u})\,,$$

so we have $\boldsymbol{f}(\mathbf{1}+r\boldsymbol{u}) \le \mathbf{1}+r\boldsymbol{u}$ iff $rQ(\boldsymbol{u},\boldsymbol{u}) \le \mathbf{1}$. The statement follows. □

Next we show how a suitable $\boldsymbol{w}$ can be found.

Lemma C.4. *One can compute in polynomial time a vector $\boldsymbol{w} \in (1,\infty)^\Gamma$ with $\boldsymbol{f}(\boldsymbol{w}) \ge \boldsymbol{w}$.*

Proof. Using the Taylor expansion of $\boldsymbol{f}(\mathbf{1}+r\boldsymbol{x})$ as in the previous lemma, we obtain $\boldsymbol{f}(\mathbf{1}+r\boldsymbol{x}) \ge \mathbf{1}+r\boldsymbol{x}$ iff

$$rQ(\boldsymbol{x},\boldsymbol{x}) \ge (I - \boldsymbol{f}'(\mathbf{1}))\boldsymbol{x}\,. \tag{3}$$

We will choose $\boldsymbol{w} := \mathbf{1} + r\boldsymbol{x}$, so we need to find suitable r and $\boldsymbol{x}$ such that (3) holds. Define $\boldsymbol{y} \in \{0,1\}^\Gamma$ such that $\boldsymbol{y}_X = 1$ if the X-component of $Q(\boldsymbol{x},\boldsymbol{x})$ is not constant zero (or, equivalently, if there is a rule

$X \overset{p}{\hookrightarrow} \langle Y, Z\rangle$ for some $Y, Z \in \Gamma$). Otherwise, i.e., if $\boldsymbol{f}_X(\boldsymbol{x})$ has degree 1, set $\boldsymbol{y}_X = 0$. Define $\boldsymbol{x} := \boldsymbol{f}'(\mathbf{1})^* \boldsymbol{y} = (I - \boldsymbol{f}'(\mathbf{1}))^{-1}\boldsymbol{y}$. By the compactness of the task system, all types can reach a type X with $\boldsymbol{y}_X = 1$. It follows that $\boldsymbol{f}'(\mathbf{1})^*\boldsymbol{y}$ is positive in all components. Hence, $\boldsymbol{x}_{min} > 0$ where $\boldsymbol{x}_{min}$ is the smallest component of $\boldsymbol{x}$.

Observe that $(I - \boldsymbol{f}'(\mathbf{1}))\boldsymbol{x} = \boldsymbol{y}$, so (3) holds at least for the components X with $\boldsymbol{y}_X = 0$. Let c denote the smallest nonzero coefficient of $\boldsymbol{f}$. Equation (3) holds also for the components X with $\boldsymbol{y}_X = 1$ if we set $r > 1/(c \cdot \boldsymbol{x}_{min})$. The statement follows. □

To complete the proof of Theorem 4.3 it remains to show the claimed bounds on $\Pr[S^\sigma \geq k]$.

Theorem 4.3. *Let Δ be subcritical. Let $\boldsymbol{v}, \boldsymbol{w} \in (1,\infty)^\Gamma$ be vectors with $\boldsymbol{f}(\boldsymbol{v}) \leq \boldsymbol{v}$ and $\boldsymbol{f}(\boldsymbol{w}) \geq \boldsymbol{w}$. Such vectors exist and can be computed in polynomial time. Denote by $\boldsymbol{v}_{min}$ and $\boldsymbol{w}_{max}$ the least component of $\boldsymbol{v}$ and the greatest component of $\boldsymbol{w}$, respectively. Then*

$$\frac{\boldsymbol{w}_{X_0} - 1}{\boldsymbol{w}_{max}^{k+2} - 1} \leq \Pr[S^\sigma \geq k] \leq \frac{\boldsymbol{v}_{X_0} - 1}{\boldsymbol{v}_{min}^{k} - 1} \textit{ for all online schedulers } \sigma.$$

Proof. Let $h > 1$ and $\boldsymbol{u} \in (0,\infty)^\Gamma$ such that $h^{\boldsymbol{u}_Y} = \boldsymbol{v}_Y$ for all $Y \in \Gamma$. Define $m^{(i)} := \boldsymbol{z}^{(i)} \bullet \boldsymbol{u}$ where "$\bullet$" denotes the scalar product. Not that $m^{(1)} = \boldsymbol{u}_{X_0}$.

Let us consider $i \geq 1$. Let $y = \boldsymbol{c}^{(1)}, \cdots, \boldsymbol{c}^{(i)}$ be a sequence of elements of $\mathbb{N}^\Gamma$ with $\boldsymbol{c}^{(i)} \neq \mathbf{0}$, and let T_y be the set of all family trees t satisfying $\boldsymbol{z}^{(j)}(t) = \boldsymbol{c}^{(j)}$ for every $1 \leq j \leq i$. Note that $m^{(i)}(t) \neq 0$. Observe that $m^{(i)}$ is constant over T_y, we denote by $m^{(i)}(T_y)$ its value over T_y.

An easy computation reveals that for $Y := \Lambda_\sigma(y)$ we have

$$\mathbb{E}\Big[h^{\boldsymbol{r}^{(i)} \bullet \boldsymbol{u}} \mid T_y\Big] = \mathbb{E}\left[\prod_{Z\in\Gamma} h^{\boldsymbol{u}_Z \cdot \boldsymbol{r}_Z^{(i)}} \mid T_y\right] = \mathbb{E}\left[\prod_{Z\in\Gamma} \boldsymbol{v}_Z^{\boldsymbol{r}_Z^{(i)}} \mid T_y\right] = \boldsymbol{f}_Y(\boldsymbol{v}) \leq \boldsymbol{v}_Y = h^{\boldsymbol{u}_Y} \quad \text{(as } \boldsymbol{f}(\boldsymbol{v}) \leq \boldsymbol{v}\text{)}. \tag{4}$$

Consequently, we have

$$\begin{aligned}
\mathbb{E}\Big[h^{m^{(i+1)}} \mid T_y\Big] &= \mathbb{E}\Big[h^{\boldsymbol{z}^{(i+1)} \bullet \boldsymbol{u}} \mid T_y\Big] && \text{(def. of } m^{(i+1)}\text{)}\\
&= \mathbb{E}\Big[h^{(\boldsymbol{z}^{(i)} + \boldsymbol{r}^{(i)} - \langle \Lambda_\sigma(y)\rangle) \bullet \boldsymbol{u}} \mid T_y\Big] && \text{(def. of } \boldsymbol{r}^{(i)}\text{)}\\
&= \mathbb{E}\Big[h^{\boldsymbol{z}^{(i)} \bullet \boldsymbol{u}} \mid T_y\Big] \cdot \mathbb{E}\Big[h^{\boldsymbol{r}^{(i)}} \mid T_y\Big] \cdot \mathbb{E}\Big[h^{-\langle \Lambda_\sigma(y)\rangle \bullet \boldsymbol{u}} \mid T_y\Big] && \begin{pmatrix} h^{\boldsymbol{z}^{(i)} \bullet \boldsymbol{u}}, \quad h^{-\langle \Lambda_\sigma(y)\rangle \bullet \boldsymbol{u}} \\ \text{const. on } T_y \end{pmatrix}\\
&= h^{m^{(i)}(T_y)} \cdot \mathbb{E}\Big[h^{\boldsymbol{r}^{(i)} \bullet \boldsymbol{u}} \mid T_y\Big] \cdot h^{-\boldsymbol{u}_Y} && \text{(def. of } m^{(i)}\text{)}.\\
&\leq h^{m^{(i)}(T_y)} && \text{(Equation (4))}
\end{aligned}$$

As this is true for all online schedulers σ and also $\mathbb{E}\big[m^{(i+1)} \mid m^{(i)} = 0\big] = 0$ we have

$$\mathbb{E}\Big[h^{m^{(i+1)}} \mid h^{m^{(1)}}, \ldots, h^{m^{(i)}}\Big] \leq h^{m^{(i)}},$$

i.e., the sequence $h^{m^{(1)}}, h^{m^{(2)}}, \ldots$ is a supermartingale.

Define the stopping time $\tau_k := \inf\{i \geq 1 \mid m^{(i)} \in \{0\} \cup [k,\infty)\}$. Note that $m^{(\tau_k)} \leq k + 2\boldsymbol{u}_{max}$, and hence that $m^{(\tau_k)} \in \{0\} \cup [k, k+2\boldsymbol{u}_{max}]$. We wish to apply Doob's Optional-Stopping Theorem [33] (sometimes called Optional-Sampling Theorem) to infer that $\mathbb{E}\Big[h^{m^{(\tau_k)}}\Big] \leq \mathbb{E}\Big[h^{m^{(1)}}\Big] = \boldsymbol{v}_{X_0}$. To this end we define the sequence $\widehat{m}^{(1)}, \widehat{m}^{(2)}, \ldots$ by setting $\widehat{m}^{(i)} := m^{(i)}$ for $i \le \tau_k$ and $\widehat{m}^{(i)} := m^{(\tau_k)}$ for $i \ge \tau_k$. The sequence $h^{\widehat{m}^{(1)}}, h^{\widehat{m}^{(2)}}, \ldots$ is a martingale as $h^{m^{(1)}}, h^{m^{(2)}}, \ldots$ is a martingale. To apply the Optional-Stopping Theorem we also need to make sure that $|h^{\widehat{m}^{(i+1)}} - h^{\widehat{m}^{(i)}}|$ is bounded by a constant, which is the case as $\widehat{m}^{(i)} \in [0, k + 2\boldsymbol{u}_{max}]$ for all i. Define the stopping time $\tau_k := \inf\{i \ge 1 \mid m^{(i)} \in \{0\} \cup [k, \infty)\}$. Doob's Optional-Stopping Theorem now yields

$$\mathbb{E}\left[h^{m^{(\tau_k)}}\right] = \mathbb{E}\left[h^{\widehat{m}^{(\tau_k)}}\right] \le \mathbb{E}\left[h^{\widehat{m}^{(1)}}\right] = \mathbb{E}\left[h^{m^{(1)}}\right] = h^{\boldsymbol{u}_{X_0}} = \boldsymbol{v}_{X_0} .$$

Let, as an abbreviation, $p_k := \Pr\left[m^{(\tau_k)} \ge k\right]$. Then we have

$$\boldsymbol{v}_{X_0} \ge \mathbb{E}\left[h^{m^{(\tau_k)}}\right] \ge h^0 \cdot (1 - p_k) + h^k \cdot p_k = 1 - p_k + h^k \cdot p_k$$

which gives

$$p_k \le \frac{\boldsymbol{v}_{X_0} - 1}{h^k - 1} .$$

Letting $|\boldsymbol{z}^{(i)}|$ denote the sum of the components of $\boldsymbol{z}^{(i)}$, and $\boldsymbol{u}_{min}$ the smallest component of $\boldsymbol{u}$, we have

$$\Pr[S^\sigma \ge k] = \Pr\left[\sup_i |\boldsymbol{z}^{(i)}| \ge k\right] \le \Pr\left[\sup_i m^{(i)} \ge k\boldsymbol{u}_{min}\right] = p_{k\boldsymbol{u}_{min}} \le \frac{\boldsymbol{v}_{X_0} - 1}{\boldsymbol{v}_{min} - 1} . \tag{5}$$

So we have shown the upper bound.

For the lower bound we redefine h and $\boldsymbol{u}$ such that $h^{\boldsymbol{u}_Y} = \boldsymbol{w}_Y$ for all $Y \in \Gamma$ which allows to show in an analogous way that

$$\mathbb{E}\left[h^{m^{(i+1)}} \mid h^{m^{(1)}}, \ldots, h^{m^{(i)}}\right] \ge h^{m^{(i)}} ,$$

i.e., the sequence $h^{m^{(1)}}, h^{m^{(2)}}, \ldots$ is now a submartingale. The Optional-Stopping Theorem now yields $\mathbb{E}\left[h^{m^{(\tau_k)}}\right] \ge \boldsymbol{w}_{X_0}$. Further we now have

$$\boldsymbol{w}_{X_0} \le \mathbb{E}\left[h^{m^{(\tau_k)}}\right] \le h^0 \cdot (1 - p_k) + h^{k+2\boldsymbol{u}_{max}} \cdot p_k = 1 - p_k + h^{k+2\boldsymbol{u}_{max}} \cdot p_k$$

which gives

$$p_k \ge \frac{\boldsymbol{w}_{X_0} - 1}{h^{k+2\boldsymbol{u}_{max}} - 1}$$

and thus

$$\Pr[S^\sigma \ge k] = \Pr\left[\sup_i |\boldsymbol{z}^{(i)}| \ge k\right] \ge \Pr\left[\sup_i m^{(i)} \ge k\boldsymbol{u}_{max}\right] = p_{k\boldsymbol{u}_{max}} \ge \frac{\boldsymbol{w}_{X_0} - 1}{\boldsymbol{w}_{max}^{k+2} - 1} .$$

□

C.3 Proof of Theorem 4.5

We first prove the following proposition.

Proposition C.5. *The set of $\boldsymbol{v}$-accumulating types can be computed in polynomial time.*

Proof. We start with some notations. By $\Rightarrow^*$ we denote the reflexive and transitive closure of $\Rightarrow$. We use "$+$" for multiset union. We say that X *can generate* a multiset α, denoted by $X \stackrel{\bullet}{\Longrightarrow} \alpha$, if some multiset containing α can be derived from X , i.e., if $X \Rightarrow^* \alpha + \beta$ for some multiset β. We write $Y \stackrel{\bullet}{\Longrightarrow}_X \alpha$ if Y can generate α using only X-bounded rules, i.e., rules $Z \hookrightarrow \beta$ such that $Z \le X$, and $Y \stackrel{\bullet}{\Longrightarrow}_{lf} \alpha$ to denote that the light-first scheduler can generate α. Finally, we denote by $\alpha^{\geq X}$ ($\alpha^{>X}$) the restriction of α to types $Y \geq X$ ($Y > X$).

We prove the following characterization: X is $\boldsymbol{v}$-accumulating iff there is Y such that $X_0 \overset{\bullet}{\Longrightarrow} Y$ and $Y \overset{\bullet}{\Longrightarrow}_Y X + Y$. This immediately leads to a polynomial algorithm.

($\Rightarrow$): Assume X is $\boldsymbol{v}$-accumulating. Then $X_0 \overset{\bullet}{\Longrightarrow}_{\mathit{lf}} n \cdot X$ holds for infinitely many $n \geq 1$. We claim that there exists a type W such that $W \overset{\bullet}{\Longrightarrow}_X n \cdot X$ for infinitely many $n \geq 1$. For the claim, take the longest suffixes of the witnesses for $X_0 \overset{\bullet}{\Longrightarrow}_{\mathit{lf}} n \cdot X$ that only use rules X-bounded rules, and let α_n be their corresponding initial multisets. These suffixes are then witnesses for $\alpha_n \overset{\bullet}{\Longrightarrow}_X n \cdot X$. By the maximality of the suffixes, either $\alpha_n = X_0$ holds for infinitely many $n \geq 1$, or $\alpha_n = \alpha_n^{\geq X}$ does. In the first case, we take $W := X_0$. In the second case, let $Z_n \hookrightarrow \beta_n$ be the rule applied to obtain α_n. Then

$$X_0 \Rightarrow_{\mathit{lf}}^* (\alpha_n - \beta_n) + Z_n \Rightarrow_{\mathit{lf}} (\alpha_n - \beta_n) + \beta_n \overset{\bullet}{\Longrightarrow}_X n \cdot X$$

where $X < Z_n$. Since the step $(\alpha_n - \beta_n) + Z_n \Rightarrow_{\mathit{lf}} (\alpha_n - \beta_n) + \beta_n$ is light-first and $X < Z_n$, we have $(\alpha_n - \beta_n) = (\alpha_n - \beta_n)^{>X}$, and so there are infinitely many $n \geq 1$ such that $\beta_n \overset{\bullet}{\Longrightarrow}_X n \cdot X$. Since $|\beta_n| \leq 2$ for all n, the type W exists, and the claim is proved.

Consider now a witness of $W \overset{\bullet}{\Longrightarrow}_X n \cdot X$ for some $n \geq 2^k + 1$, where k is the number of types. The corresponding tree has depth at least $k+1$, and so it contains a path in which some type Y appears twice. This easily leads to $Y \overset{\bullet}{\Longrightarrow}_X X + Y$ for some type Y such that $X_0 \overset{\bullet}{\Longrightarrow} Y$.

($\Leftarrow$): We start with some simple properties of the relations $\Rightarrow_X^*$ and $\Rightarrow_{\mathit{lf}}^*$.

(1) If $Y \overset{\bullet}{\Longrightarrow}_X \alpha$ and $\alpha = \alpha^{\geq X}$, then $Y \overset{\bullet}{\Longrightarrow}_{\mathit{lf}} \alpha$.
Consider a family tree having a (prefix of a) derivation that witnesses $Y \overset{\bullet}{\Longrightarrow}_X \alpha$. So all ancestors of the nodes corresponding to α are labeled by symbols that are $\leq X$. It follows that a light-first scheduler may select all ancestors of the α-nodes before selecting any α-node. Hence $Y \overset{\bullet}{\Longrightarrow}_{\mathit{lf}} \alpha$.

(2) If $X \overset{\bullet}{\Longrightarrow} Y$ and $Y \overset{\bullet}{\Longrightarrow}_{\mathit{lf}} \beta$, then $X \overset{\bullet}{\Longrightarrow}_{\mathit{lf}} \beta$.
$X \overset{\bullet}{\Longrightarrow} Y$ implies $X \Rightarrow_{\mathit{lf}}^* Y + \alpha$ for some α, and $Y \overset{\bullet}{\Longrightarrow}_{\mathit{lf}} \beta$ implies $Y \Rightarrow_{\mathit{lf}}^* \beta + \beta_1$ for some β_1. As $X \Rightarrow_{\mathit{lf}}^* Y + \alpha$, it suffices to find a derivation witnessing $Y + \alpha \Rightarrow_{\mathit{lf}}^* \emptyset$ that reaches a multiset of the form $\beta + \gamma$ for some γ. Such a derivation is obtained by interleaving the witnesses for $Y \Rightarrow_{\mathit{lf}}^* \beta + \beta_1 \Rightarrow_{\mathit{lf}}^* \emptyset$ and $\alpha \Rightarrow_{\mathit{lf}}^* \emptyset$.

Assume now that $X_0 \overset{\bullet}{\Longrightarrow} Y$ and $Y \overset{\bullet}{\Longrightarrow}_X X + Y$ hold. Then $Y \overset{\bullet}{\Longrightarrow}_X n \cdot X$ for every $n \geq 1$. Now (1) yields $Y \overset{\bullet}{\Longrightarrow}_{\mathit{lf}} n \cdot X$, and (2) leads to $X_0 \overset{\bullet}{\Longrightarrow}_{\mathit{lf}} n \cdot X$, also for every $n \geq 1$. So X is $\boldsymbol{v}$-accumulating. $\square$

Now we complete the proof of Theorem 4.5.

Theorem 4.5. *Let Δ be subcritical and $\boldsymbol{v} \in (1, \infty)^{\Gamma}$ with $\boldsymbol{f}(\boldsymbol{v}) \leq \boldsymbol{v}$. Let σ be a $\boldsymbol{v}$-light-first scheduler. Let $\boldsymbol{v}_{minmax} := \min_{X \hookrightarrow \langle Y,Z \rangle} \max\{\boldsymbol{v}_Y, \boldsymbol{v}_Z\}$ (here the minimum is taken over all transition rules with two types on the right hand side). Then $\boldsymbol{v}_{minmax} \geq \boldsymbol{v}_{min}$ and for all $k \geq 1$*

$$\Pr[S^\sigma \geq k] \leq \frac{\boldsymbol{v}_{X_0} - 1}{\boldsymbol{v}_{min} \boldsymbol{v}_{minmax}^{k-1} - 1} \,.$$

Moreover, let $\boldsymbol{v}_{minacc} := \min\{\boldsymbol{v}_X \mid X \in \Gamma,\ X$ is $\boldsymbol{v}$-accumulating$\}$. Then $\boldsymbol{v}_{minacc} \geq \boldsymbol{v}_{minmax}$, $\boldsymbol{v}_{minacc}$ can be computed in polynomial time, and there is an integer ℓ such that for all $k \geq \ell$

$$\Pr[S^\sigma \geq k] \leq \frac{\boldsymbol{v}_{X_0} - 1}{\boldsymbol{v}_{min}^{\ell} \boldsymbol{v}_{minacc}^{k-\ell} - 1} \,.$$

Proof. The inequality $\boldsymbol{v}_{minmax} \geq \boldsymbol{v}_{min}$ is trivial. For the inequality $\boldsymbol{v}_{minacc} \geq \boldsymbol{v}_{minmax}$, let $Li := \{Y \in \Gamma \mid \boldsymbol{v}_Y < \boldsymbol{v}_{minmax}\}$ be the set of types that are strictly lighter than $\boldsymbol{v}_{minmax}$. We claim that, in each step i, there is at most one task of Li-type. More formally, if $\boldsymbol{e}^{(Li)}$ denotes the vector with $\boldsymbol{e}^{(Li)}_Y = 1$ for $Y \in Li$ and $\boldsymbol{e}^{(Li)}_Y = 0$ for $Y \notin Li$, then we have $\boldsymbol{z}^{(i)} \bullet \boldsymbol{e}^{(Li)} \leq 1$ for all i. This can be shown by a straightforward induction on the derivation length: at each step the task of Li-type (if present) is selected and replaced by at most two tasks. By definition of $\boldsymbol{v}_{minmax}$, at most one of the new tasks has Li-type. Hence, the types in Li are not accumulating. It follows $\boldsymbol{v}_{minacc} \geq \boldsymbol{v}_{minmax}$.

The rest of the proof is obtained by a small modification of the proof of Theorem 4.3: it suffices to show that, in Equation (5), we can replace $k\boldsymbol{u}_{min}$ by $\boldsymbol{u}_{min} + (k-1)\boldsymbol{u}_{minmax}$ and by $\ell\boldsymbol{u}_{min} + (k-\ell)\boldsymbol{u}_{minacc}$ for some integer ℓ. (The values $\boldsymbol{u}_{minmax}$ and $\boldsymbol{u}_{minacc}$ are defined in the obvious way, i.e., using the h from the proof of Theorem 4.3 we have $h^{\boldsymbol{u}_{minmax}} = \boldsymbol{v}_{minmax}$ and $h^{\boldsymbol{u}_{minacc}} = \boldsymbol{v}_{minacc}$.) So we need to show for the light-first scheduler σ that $|\boldsymbol{z}^{(i)}| \geq k$ implies both $m^{(i)} \geq \boldsymbol{u}_{min} + (k-1)\boldsymbol{u}_{minmax}$ and $m^{(i)} \geq \ell\boldsymbol{u}_{min} + (k-\ell)\boldsymbol{u}_{minacc}$.

For the first implication, recall that $m^{(i)} = \boldsymbol{z}^{(i)} \bullet \boldsymbol{u}$. We have argued above that $\boldsymbol{z}^{(i)} \bullet \boldsymbol{e}^{(Li)} \leq 1$. This implies $m^{(i)} \geq \boldsymbol{u}_{min} + (k-1)\boldsymbol{u}_{minmax}$.

For the second implication, let ℓ' be an integer such that $\boldsymbol{z}^{(i)}_Y \leq \ell'$ for all i and for all non-accumulating types Y. Let $\ell := |\Gamma| \cdot \ell'$. Then in each step, there are at most ℓ tasks of non-accumulating type. This implies $m^{(i)} \geq \ell\boldsymbol{u}_{min} + (k-\ell)\boldsymbol{u}_{minacc}$. □

C.4 Proof of Theorem 4.7

In the following we let $M^* := I + M + MM + \cdots$ for any square matrix M. If M^* converges, then, by basic matrix facts, it equals $(I-M)^{-1}$. Also by basic matrix facts (see e.g. [19]), M^* converges iff the spectral radius of M is less than one.
We first prove Lemma 4.8.

Lemma 4.8. *Let $A[k] := L + Q(\mathbf{1} - \boldsymbol{s}[k], \cdot)$. Then $(I - A[k])^{-1}$ exists and for all $k \geq 1$*

$$\boldsymbol{s}[k+1] = A[k]\boldsymbol{s}[k+1] + Q(\cdot, \mathbf{1})\boldsymbol{s}[k] = (I - A[k])^{-1} Q(\cdot, \mathbf{1})\boldsymbol{s}[k]\,.$$

Proof. The following equation follows from the definition of a depth-first scheduler σ.

$$\begin{aligned}\Pr[S^\sigma_X \geq k+1] = & \sum_{X \xhookrightarrow{p} Y} p \Pr[S^\sigma_Y \geq k+1] \\ & + \sum_{X \xhookrightarrow{p} YZ} p \left(\Pr[S^\sigma_Y \geq k] + \Pr[S^\sigma_Y < k] \cdot \Pr[S^\sigma_Z \geq k+1]\right)\end{aligned}$$

Using the definitions this immediately implies the equality

$$\boldsymbol{s}[k+1] = A[k]\boldsymbol{s}[k+1] + Q(\cdot, \mathbf{1})\boldsymbol{s}[k]\,.$$

For the second equality of the proposition, note that $\boldsymbol{f}'(\mathbf{1}) = L + Q(\mathbf{1}, \cdot) + Q(\cdot, \mathbf{1})$. As the task system is subcritical, the spectral radius of $\boldsymbol{f}'(\mathbf{1})$ is, by Proposition 2.5, less than one. So the spectral radius of $A[k] \leq L + Q(\mathbf{1}, \cdot) \leq \boldsymbol{f}'(\mathbf{1})$ is less than one as well. Hence, by standard matrix facts [19] the sum $A[k]^*$ converges and equals $(I - A[k])^{-1}$. The second equality follows. □

For the proof of Theorem 4.7 we will need the following auxiliary lemma.

Lemma C.6. *Let A be a nonnegative square matrix with spectral radius less than one. Let $(\epsilon_n)_{n\in\mathbb{N}}$ be a sequence with $\epsilon_n \geq \epsilon_{n+1} \geq 0$ converging to 0. Then there exists an n_1 and a nonnegative matrix K such that for all $n \geq n_1$*

$$\big((1-\epsilon_n)A\big)^* \geq (I - \epsilon_n K)A^*\,.$$

Proof. We can assume $\epsilon_n \leq 1$. Let $M = (I - A)^{-1}A$. Then by a simple computation

$$\big((1-\epsilon_n)A\big)^* = \big(I + \epsilon_n M\big)^{-1} A^* \,.$$

Choose n_1 large enough so that $\rho(\epsilon_n M) < 1$. Then $(\epsilon_n M)^*$ exists and so

$$\begin{aligned}\big(I + \epsilon_n M\big)^{-1} &= I - (\epsilon_n M) + (\epsilon_n M)^2 - (\epsilon_n M)^3 + - \cdots \\ &\geq I - (\epsilon_n M)(\epsilon_n M)^* \\ &\geq I - \epsilon_n M(\epsilon_{n_1} M)^*\end{aligned}$$

Choose $K = M(\epsilon_{n_1} M)^*$ and the claim follows. □

We also need the following lemma.

Lemma C.7. *Given a depth-first scheduler and using the notation from the main body of the paper, let* $B := (I - L - Q(\mathbf{1}, \cdot))^{-1}Q(\cdot, \mathbf{1})$. *Then the spectral radius of* B *is less than one.*

Proof. Observe that $\boldsymbol{f}'(\mathbf{1}) = L + Q(\mathbf{1}, \cdot) + Q(\cdot, \mathbf{1})$. As (Δ, X) is subcritical, Proposition 2.5 implies that the spectral radius of $\boldsymbol{f}'(\mathbf{1})$ is less than one. Then it follows that the spectral radius of B is less than one as well, using the theory of M-matrices and regular splittings, see [5], Theorem 6.2.3 part P_{48}. □

Now we prove Theorem 4.7.

Theorem 4.7. *Let* Δ *be subcritical and* σ *be any depth-first scheduler. Let* ρ *be the spectral radius of* $(I - L - Q(\mathbf{1}, \cdot))^{-1}Q(\cdot, \mathbf{1})$. *Then* $0 < \rho < 1$ *and* $\Pr[S^\sigma \geq k] \in \Theta(\rho^k)$, *i.e, there are* $c, C > 0$ *such that* $c\rho^k \leq \Pr[S^\sigma \geq k] \leq C\rho^k$ *for all* k.

Proof. Let $B := (L + Q(\mathbf{1}, \cdot))^* \, Q(\cdot, \mathbf{1})$ and ρ the spectral radius of B. We have $\rho < 1$ by Lemma C.7. To show $\rho > 0$, it suffices (by Perron-Frobenius theory [5]) to show that all row sums of B are (strictly) positive. For this, let $Y \in \Gamma$ be the index of an arbitrary row. Then, by compactness of the task system, there are types $X_0, \ldots, X_i$ $(0 \leq i \leq n-1)$ such that $Y = X_i$ and $X_i \xhookrightarrow{p_i} X_{i-1}, \ldots, X_1 \xhookrightarrow{p_1} X_0$ and $X_0 \xhookrightarrow{p_0} ZW$ for some $Z, W \in \Gamma$. It is straightforward to show by induction on i that the (Y, Z)-entry of $L^i Q(\cdot, \mathbf{1})$ is positive. It follows that the (Y, Z)-entry of B is positive, so $\rho > 1$.

For the upper bound, observe that with Lemma 4.8 we have

$$\boldsymbol{s}[k+1] = (L + Q(\mathbf{1} - \boldsymbol{s}[k], \cdot))^* \, Q(\cdot, \mathbf{1})\boldsymbol{s}[k] \leq B\boldsymbol{s}[k] \,. \tag{6}$$

By a simple induction it follows $\boldsymbol{s}[k+i] \leq B^i \boldsymbol{s}[k]$. As the absolute values of the eigenvalues of B are bounded by ρ we get $\|\boldsymbol{s}[k+i]\| \leq C_1\rho^i$ for some $C_1 > 0$, which implies the claimed upper bound.

For the lower bound, observe that there is a real number $0 < r \leq 1$ such that for all types $Y \in \Gamma$, the probability that X reaches Y is at least r. So it suffices to find any $Y \in \Gamma$ such that there is a $c_1 > 0$ with $\Pr[S_Y^\sigma \geq k] \geq c_1\rho^k$ for all k.

Recall that ρ is the spectral radius of B. It is a corollary (Corollary 2.1.6 of [5]) of Perron-Frobenius theory that B has a principal submatrix B' which is irreducible and also has spectral radius ρ. We write $\Gamma_\uparrow$ for the subset of Γ such that B' is obtained from B by deleting all rows and columns that are not indexed by $\Gamma_\uparrow$. Also by Perron-Frobenius theory, B' has an eigenvector $\boldsymbol{u}' \in (0, \infty)^{\Gamma_\uparrow}$ with $B'\boldsymbol{u}' = \rho\boldsymbol{u}'$ so that $\boldsymbol{u}'$ is positive in all components. Define $\boldsymbol{u} \in [0, \infty)^\Gamma$ as the vector with $\boldsymbol{u}_Y = \boldsymbol{u}'_Y > 0$ for $Y \in \Gamma_\uparrow$ and $\boldsymbol{u}_Y = 0$ for $Y \notin \Gamma_\uparrow$. Hence we have $B\boldsymbol{u} \geq \rho\boldsymbol{u}$. By the already proven upper bound there is a $t > 0$ such that $\boldsymbol{s}[k] \leq t\rho^k$ for all k. We abbreviate $\epsilon_k := t\rho^k$ so that $\boldsymbol{s}[k] \leq \epsilon_k\mathbf{1}$.

Now we show that there is a natural number k and a real number $d > 0$ with $\epsilon_k d < 1$ such that for all $i \geq 0$

$$\boldsymbol{s}[k+i] \geq \rho^i \left(\prod_{j=1}^{i} (1 - \epsilon_{k+j-1} d) \right) \boldsymbol{u} \,. \tag{7}$$

As $\boldsymbol{u}_Y = 0$ for $Y \notin \Gamma_\uparrow$ it suffices to show $\boldsymbol{s}[k+i] \geq_\uparrow \rho^i \left(\prod_{j=1}^{i} (1 - \epsilon_{k+j-1} d) \right) \boldsymbol{u}$ where by the notation $\boldsymbol{v} \geq_\uparrow \boldsymbol{w}$ we mean $\boldsymbol{v}_Y \geq \boldsymbol{w}_Y$ for all $Y \in \Gamma_\uparrow$. We proceed by induction on i and determine the constants on the fly. For the induction base ($i = 0$) observe that, as $\boldsymbol{s}[k]$ is positive by compactness of the task system, we can enforce $\boldsymbol{s}[k] \geq \boldsymbol{u}$ by scaling down $\boldsymbol{u}$ by multiplying it with a small constant. This does not affect the stated properties of $\boldsymbol{u}$. For the step, let $i \geq 0$. We have

$$\begin{aligned}
\boldsymbol{s}[k+i+1] &= (L + Q(\mathbf{1} - \boldsymbol{s}[k+i], \cdot))^* \, Q(\cdot, \mathbf{1}) \boldsymbol{s}[k+i] && \text{(by (6))} \\
&\geq ((1 - \epsilon_{k+i})(L + Q(\mathbf{1}, \cdot)))^* \, Q(\cdot, \mathbf{1}) \boldsymbol{s}[k+i] && (\text{as } \boldsymbol{s}[k+i] \leq \epsilon_{k+i} \mathbf{1}) \\
&\geq ((1 - \epsilon_{k+i})(L + Q(\mathbf{1}, \cdot)))^* \, Q(\cdot, \mathbf{1}) \rho^i \left(\prod_{j=1}^{i} (1 - \epsilon_{k+j-1} d) \right) \boldsymbol{u} && \text{(ind. hypothesis)} \\
&\geq (I - \epsilon_{k+i} K) B \rho^i \left(\prod_{j=1}^{i} (1 - \epsilon_{k+j-1} d) \right) \boldsymbol{u} && \left(\begin{array}{l} \text{for a large } k \text{ and} \\ \text{some matrix } K \text{ by} \\ \text{Lemma C.6} \end{array} \right) \\
&\geq \rho^i \left(\prod_{j=1}^{i} (1 - \epsilon_{k+j-1} d) \right) (\rho \boldsymbol{u} - \epsilon_{k+i} K B \boldsymbol{u}) && (\text{as } B\boldsymbol{u} \geq \rho \boldsymbol{u}) \\
&\geq_\uparrow \rho^i \left(\prod_{j=1}^{i} (1 - \epsilon_{k+j-1} d) \right) (\rho \boldsymbol{u} - \epsilon_{k+i} \rho d \boldsymbol{u}) && \left(\begin{array}{l} \text{for a large } d \text{ with} \\ KB\boldsymbol{u} \leq_\uparrow \rho d \boldsymbol{u} \end{array} \right) \\
&= \rho^{i+1} \left(\prod_{j=1}^{i+1} (1 - \epsilon_{k+j-1} d) \right) \boldsymbol{u}
\end{aligned}$$

This proves (7). So, denoting by $\boldsymbol{u}_{min} > 0$ the smallest nonzero component of $\boldsymbol{u}$, we have

$$\boldsymbol{s}[k+i]_Y \geq \rho^i \left(\prod_{j=1}^{i+1} (1 - \epsilon_{k+j-1} d) \right) \boldsymbol{u}_{min} \qquad \text{for all } Y \in \Gamma_\uparrow \text{ and all } i \geq 0.$$

Thus the proof is completed if $\prod_{j=k}^{\infty} (1 - \epsilon_j d) > 0$. To see that this inequality holds, observe that $1 - \epsilon_j d = 1 - t\rho^j d \geq 1 - \frac{1}{j^2}$ is true for almost all j and that $\prod_{j=2}^{\infty} (1 - \frac{1}{j^2}) = \frac{1}{2} > 0$. This completes the proof. $\square$

D Proofs of Section 5

D.1 Proof of Theorem 5.1

Theorem 5.1. *The expectation $\mathbb{E}[S^{op}]$ is finite (no matter whether Δ is critical or subcritical). Moreover, $\mathcal{O}(b)$ terms compute b bits of $\mathbb{E}[S^{op}]$. If the task system Δ is subcritical, then $\log_2 b + \mathcal{O}(1)$ terms compute b bits of $\mathbb{E}[S^{op}]$. Finally, computing k terms takes time $\mathcal{O}(k \cdot |\Gamma|^3)$ in the unit cost model.*

Proof. Note that the second statement implies the first one. Let $e^{(i)} := 1 - \boldsymbol{\nu}^{(i)}_{X_0}$. Then we have $\mathbb{E}[S^{op}] - \sum_{i=0}^{k-1}(1-\boldsymbol{\nu}^{(i)}_{X_0}) = \sum_{i=k}^{\infty} e^{(i)}$. It follows from [11] that there is a $c_1 \in (0,\infty)$ such that for all $i \in \mathbb{N}$ we have $e^{(i)} \le c_1 \cdot 2^{-i/(n2^n)}$ where $n = |\mathit{\Gamma}|$. Using this inequality we get

$$\sum_{i=k}^{\infty} e^{(i)} \le c_1 \sum_{i=k}^{\infty} 2^{-i/(n2^n)} \le c_2 \cdot 2^{-k/(n2^n)}$$

with $c_2 = c_1/(1-2^{-1/(n2^n)})$. Choosing $k = \lceil (b + \log_2 c_2) n2^n \rceil$ we obtain $\sum_{i=k}^{\infty} e^{(i)} \le 2^{-b}$ which proves the second statement.

For the third statement (about subcritical systems) recall from Corollary 3.5 that there are $c > 0$ and $0 < d < 1$ such that $e^{(i)} \le c \cdot d^{2^i}$ for all $i \in \mathbb{N}$. So

$$\sum_{i=k}^{\infty} e^{(i)} \le \sum_{i=k}^{\infty} c \cdot d^{2^i} \le c \cdot \sum_{i=0}^{\infty} d^{2^k+i} = \frac{c}{1-d} \cdot d^{2^k} .$$

By choosing a natural number k with $k \ge -\log_2(-\log_2 d) + \log_2 b + 1$ we obtain for all $b \ge \log \frac{c}{1-d}$ that $\frac{c}{1-d} \cdot d^{2^k} \le 2^{-b}$ which proves the third statement.

The final statement follows from Corollary 3.4. $\square$

D.2 Proof of Theorem 5.2

Theorem 5.2. *If $\mathit{\Delta}$ is critical, then $\mathbb{E}[S^\sigma]$ is infinite for every online scheduler σ.*

Proof. The proof follows the lines of the proof of Theorem 4.3. Let $\mathit{\Delta}$ be critical. By Proposition 2.5 we have $\rho(\boldsymbol{f}'(\boldsymbol{1})) = 1$ for the spectral radius of $\boldsymbol{f}'(\boldsymbol{1})$.

Let us fix an online scheduler σ. First we prove $\mathbb{E}[S^\sigma] = \infty$ for the case in which X_0 is reachable from every type $X \in \mathit{\Gamma}$. Later we will show how to drop this assumption. If X_0 is reachable from every X, it follows that $\boldsymbol{f}'(\boldsymbol{1})$ is an irreducible matrix. Then Perron-Frobenius theory [5] guarantees the existence of an eigenvector $\boldsymbol{u} \in \mathbb{R}^{\mathit{\Gamma}}$ of $\boldsymbol{f}'(\boldsymbol{1})$ which is positive in all components, i.e., $\boldsymbol{f}'(\boldsymbol{1})\boldsymbol{u} = \boldsymbol{u}$ and $\boldsymbol{u}_X > 0$ for all $X \in \mathit{\Gamma}$. W.l.o.g. we can choose $\boldsymbol{u}$ such that its largest component is 1. Let again $m^{(i)} := \boldsymbol{z}^{(i)} \bullet \boldsymbol{u}$. Note that $m^{(1)} = \boldsymbol{u}_{X_0} > 0$ and $m^{(i)} \le |\boldsymbol{z}^{(i)}|$ where $|\boldsymbol{z}^{(i)}|$ denotes the sum of the components of $\boldsymbol{z}^{(i)}$. Also note that $m^{(i)}$ returns a weighted sum of the components of $\boldsymbol{z}^{(i)}$. Loosely speaking, we will show that its expectation remains constant.

Let us consider $i \ge 1$. Let $y = \boldsymbol{c}^{(1)}, \cdots, \boldsymbol{c}^{(i)}$ be a sequence of elements of $\mathbb{N}^{\mathit{\Gamma}}$ with $\boldsymbol{c}^{(i)} \ne \boldsymbol{0}$, and let T_y be the set of all family trees t satisfying $\boldsymbol{z}^{(j)}(t) = \boldsymbol{c}^{(j)}$ for every $1 \le j \le i$. Note that $m^{(i)}(t) \ne 0$. Observe that $m^{(i)}$ is constant over T_y, we denote by $m^{(i)}(T_y)$ its value over T_y.

An easy computation reveals that for every $X \in \mathit{\Gamma}$ we have

$$\mathbb{E}\left[\boldsymbol{r}^{(i)}_X \mid T_y\right] = \sum_{\mathit{\Lambda}_\sigma(y) \xhookrightarrow{p} \alpha} p \cdot \#_X(\alpha) = \boldsymbol{f}'_{\mathit{\Lambda}_\sigma(y),X}(\boldsymbol{1})$$

which gives

$$\mathbb{E}\left[\boldsymbol{r}^{(i)} \mid T_y\right] = \boldsymbol{f}'_{\mathit{\Lambda}_\sigma(y)}(\boldsymbol{1}) \tag{8}$$

(where $\boldsymbol{f}'_{\mathit{\Lambda}_\sigma(y)}(\boldsymbol{1})$ denotes the row vector indexed by $\mathit{\Lambda}_\sigma(y)$). Consequently, we have:

$$\mathbb{E}\left[m^{(i+1)} \mid T_y\right] = \mathbb{E}\left[\boldsymbol{z}^{(i+1)} \mid T_y\right] \bullet \boldsymbol{u} \qquad \text{(def. of } m^{(i+1)})$$

$$= \Big(\mathbb{E}\Big[\boldsymbol{z}^{(i)} \mid T_y\Big] + \mathbb{E}\Big[\boldsymbol{r}^{(i)} \mid T_y\Big] - \mathbb{E}\Big[\langle X^{(i)}\rangle \mid T_y\Big]\Big) \bullet \boldsymbol{u} \qquad \text{(def. of } \boldsymbol{r}^{(i)}\text{)}$$

$$= \Big(\mathbb{E}\Big[\boldsymbol{z}^{(i)} \mid T_y\Big] + \boldsymbol{f}'_{\Lambda_\sigma(y)}(\mathbf{1}) - \langle \Lambda_\sigma(y)\rangle\Big) \bullet \boldsymbol{u} \qquad \text{(by (8))}$$

$$= m^{(i)}(T_y) + \boldsymbol{f}'_{\Lambda_\sigma(y)}(\mathbf{1})\boldsymbol{u} - \langle \Lambda_\sigma(y)\rangle \bullet \boldsymbol{u} \qquad \text{(def. of } m^{(i)}(T_y)\text{)}$$

$$= m^{(i)}(T_y) \qquad \text{(as } \boldsymbol{f}'(\mathbf{1})\boldsymbol{u} = \boldsymbol{u}\text{)}$$

Also clearly $\mathbb{E}\big[m^{(i+1)} \mid m^{(i)} = 0\big] = 0$, and hence we have

$$\mathbb{E}\Big[m^{(i+1)} \mid m^{(1)}, \ldots, m^{(i)}\Big] = m^{(i)}\,,$$

i.e., the sequence $m^{(1)}, m^{(2)}, \ldots$ is a martingale.

Define the stopping time $\tau_k := \inf\{i \geq 1 \mid m^{(i)} \in \{0\} \cup [k, \infty)\}$. Note that $m^{(\tau_k)} \leq k+2$ as $\boldsymbol{u} \leq \mathbf{1}$, and hence that $m^{(\tau_k)} \in \{0\} \cup [k, k+2]$. We wish to apply Doob's Optional-Stopping Theorem [33] (sometimes called Optional-Sampling Theorem) to infer that $\mathbb{E}\big[m^{(\tau_k)}\big] = \mathbb{E}\big[m^{(1)}\big] = \boldsymbol{u}_{X_0}$. To this end we define the sequence $\widehat{m}^{(1)}, \widehat{m}^{(2)}, \ldots$ by setting $\widehat{m}^{(i)} := m^{(i)}$ for $i \leq \tau_k$ and $\widehat{m}^{(i)} := m^{(\tau_k)}$ for $i \geq \tau_k$. The sequence $\widehat{m}^{(1)}, \widehat{m}^{(2)}, \ldots$ is a martingale as $m^{(1)}, m^{(2)}, \ldots$ is a martingale. To apply the Optional-Stopping Theorem we also need to make sure that $|\widehat{m}^{(i+1)} - \widehat{m}^{(i)}|$ is bounded by a constant, which is the case as $\widehat{m}^{(i)} \in [0, k+2]$ for all i. Doob's Optional-Stopping Theorem now yields

$$\mathbb{E}\Big[m^{(\tau_k)}\Big] = \mathbb{E}\Big[\widehat{m}^{(\tau_k)}\Big] = \mathbb{E}\Big[\widehat{m}^{(1)}\Big] = \boldsymbol{u}_{X_0}\,.$$

Recall that this is > 0. Since $m^{(\tau_k)} \in \{0\} \cup [k, k+2]$,

$$\boldsymbol{u}_{X_0} = \mathbb{E}\Big[m^{(\tau_k)}\Big] \leq 0 \cdot \Pr\Big[m^{(\tau_k)} = 0\Big] + (k+2) \cdot \Pr\Big[m^{(\tau_k)} \geq k\Big] = (k+2) \cdot \Pr\Big[m^{(\tau_k)} \geq k\Big]$$

which gives

$$\Pr\Big[m^{(\tau_k)} \geq k\Big] \geq \frac{\boldsymbol{u}_{X_0}}{k+2}\,.$$

So we have

$$\Pr[S^\sigma \geq k] = \Pr\left[\sup_i |\boldsymbol{z}^{(i)}| \geq k\right] \geq \Pr\left[\sup_i m^{(i)} \geq k\right] = \Pr\Big[m^{(\tau_k)} \geq k\Big] \geq \frac{\boldsymbol{u}_{X_0}}{k+2}\,.$$

Hence,

$$\mathbb{E}[S^\sigma] = \sum_{k=1}^{\infty} \Pr[S^\sigma \geq k] \geq \sum_{k=1}^{\infty} \frac{\boldsymbol{u}_{X_0}}{k+2} = \infty$$

which completes the proof for the case where X_0 is reachable from all types.

Now we show that $\mathbb{E}[S^\sigma] = \infty$ also holds when X_0 is not reachable from all types. Recall that $\rho(\boldsymbol{f}'(\mathbf{1})) = 1$. It is a corollary (Corollary 2.1.6 of [5]) of Perron-Frobenius theory that $\boldsymbol{f}'(\mathbf{1})$ has a principal submatrix B which is irreducible and has spectral radius $\rho(B) = 1$. Let $\Gamma' \subseteq \Gamma$ denote the set of types such that B is obtained from $\boldsymbol{f}'(\mathbf{1})$ by deleting all rows and columns not indexed by Γ'. Consider the task system Δ' which is the original task system restricted to Γ'. More concretely, Δ' has types Γ' and transition rules as follows: A rule $X \xhookrightarrow{p} \alpha'$ is in Δ' iff $X \in \Gamma'$ and there is an $\alpha \in M_\Gamma^{\leq 2}$ such that $X \xhookrightarrow{p} \alpha$ is in the original task system and α' is obtained from α by deleting the types that are not in Γ'. Let $\boldsymbol{g} : \mathbb{R}^{\Gamma'} \to \mathbb{R}^{\Gamma'}$ denote the pgf for Δ'. From the construction of Δ' it is straightforward to see that $B = \boldsymbol{g}'(\mathbf{1})$. Pick an arbitrary $X \in \Gamma'$ as the initial type of Δ'. As $B = \boldsymbol{g}'(\mathbf{1})$ is irreducible, X is reachable from all types in Γ'. Hence, the first part of the proof applies and we obtain that, in Δ', we have $\mathbb{E}[S^\sigma_X] = \infty$ for all online schedulers σ. As Δ' was obtained by erasing types and rules from the original task system, it is easy to see that, also in the original task system, we have $\mathbb{E}[S^\sigma_X] = \infty$ for all online schedulers σ. As X is reachable from X_0, it follows $\mathbb{E}[S^\sigma] = \infty$ for all online schedulers σ. $\square$

D.3 Proof of Theorem 5.3

Theorem 5.3. *Let Δ be subcritical, and let $B := (L + Q(\mathbf{1}, \cdot))^* Q(\cdot, \mathbf{1})$. Then $(I - B)^{-1}$ exists and $\mathbb{E}[S^\sigma] - u[k] \leq \left\|(I - B)^{-1}\right\|_1 \|\boldsymbol{s}[k]\|_1$ for all $k \geq 1$, where $u[k] := \sum_{i=1}^{k} \boldsymbol{s}[i]_{X_0} = \sum_{i=1}^{k} \Pr[S^\sigma \geq i]$. Hence, $\mathcal{O}(b)$ terms compute b bits of $\mathbb{E}[S^\sigma]$. Finally, computing k terms takes time $\mathcal{O}(k \cdot |\Gamma|^3)$ in the unit cost model.*

Proof. By Lemma C.7 the spectral radius of B is less than one. So by standard matrix facts (see [19]) $B^* = (I - B)^{-1}$ exists. Recall from Lemma 4.8 that $\boldsymbol{s}[k + j + 1] = A[k + j]^* Q(\cdot, \mathbf{1}) \boldsymbol{s}[k + j]$ for all $j \geq 0$. As $A[k+j]^* Q(\cdot, \mathbf{1}) \leq B$, we obtain $\boldsymbol{s}[k+j] \leq B^j \boldsymbol{s}[k]$ by a simple induction. Define the "error vector" $\boldsymbol{\delta}[k]$ by

$$\boldsymbol{\delta}[k]_Y := \mathbb{E}[S^\sigma_Y] - \sum_{i=1}^{k} \boldsymbol{s}[i]_Y \qquad \text{for all } Y \in \Gamma.$$

Then we have

$$\boldsymbol{\delta}[k] = \sum_{i=k+1}^{\infty} \boldsymbol{s}[i] \leq \sum_{j=0}^{\infty} \boldsymbol{s}[k+j] \leq \sum_{j=0}^{\infty} B^j \boldsymbol{s}[k] = B^* \boldsymbol{s}[k]$$

which yields

$$\mathbb{E}[S^\sigma] - u[k] = \boldsymbol{\delta}[k]_{X_0} \leq \|\boldsymbol{\delta}[k]\|_1 \leq \|B^*\|_1 \|\boldsymbol{s}[k]\|_1 \leq \|B^*\|_1 \cdot C' \rho^k$$

where the last step is by Theorem 4.7. Recall that $\rho < 1$. Let $D > 1$ such that $D \geq \|B^*\|_1 \cdot C'$. In order to show that $\mathcal{O}(b)$ terms compute b bits of $\mathbb{E}[S^\sigma]$, we have to find a K such that $D\rho^{K \cdot b} \leq 2^{-b}$ for all b. Choosing $K := \left\lceil \frac{1 + \log D}{\log \frac{1}{\rho}} \right\rceil$ we have in fact

$$D\rho^{K \cdot b} \leq D \cdot 2^{-b} \cdot D^{-b} \leq 2^{-b}\,.$$

The final statement about the time in the unit-cost model follows from the comments at the end of Section 4.3. ◻

D.4 Proofs of Theorem 5.4 and Theorem 5.5

In this subsection we give formal proofs for Theorem 5.4, respectively Theorem 5.5.

For this we first introduce an alternative semantics of task systems which is equivalent to the original one with respect to the expected completion space (see Lemma D.2 below). Informally, we move from the set of family trees to the set of derivations (i.e. sequences of multisets) assigned to the family trees by a given online scheduler. The new semantics considerably simplifies proofs concerning the expected completion space for online schedulers.

Let $\Delta = (\Gamma, \hookrightarrow, Prob, X_0)$ be a task system. We denote by $\mathcal{C}$ the set $\mathbb{N}^\Gamma$, and call $\mathcal{C}$ the set of *configurations*. In this section we use the multisets over Γ and vectors of $\mathbb{N}^\Gamma$ interchangeably in the canonical way. A *multiset scheduler* κ takes a non-empty sequence $\boldsymbol{c}^{(1)} \ldots \boldsymbol{c}^{(n)}$ of configurations and chooses from the current (= last) configuration $\boldsymbol{c}^{(n)}$ the next task type which is going to be processed, more precisely, κ is a function $\kappa : \mathcal{C}^+ \to \Gamma$ such that for every $w \in \mathcal{C}^*$ and $\boldsymbol{c} \in \mathcal{C} \setminus \{\mathbf{0}\}$ we have $\boldsymbol{c}_{\kappa(w\boldsymbol{c})} > 0$. By scheduling a task of type $\kappa(w\boldsymbol{c})$, we obtain with probability $Prob((\kappa(w\boldsymbol{c}), \alpha))$ a set $\alpha \in M_\Gamma^{\leq 2}$ of new tasks, which yields the new configuration $\boldsymbol{c}' := (\boldsymbol{c} - \langle \kappa(w\boldsymbol{c}) \rangle) + \alpha$. We may think of this as moving on from $w\boldsymbol{c}$ to $w\boldsymbol{c}\boldsymbol{c}'$ with probability $Prob((\kappa(w\boldsymbol{c}), \alpha))$. We abbreviate this with $w\boldsymbol{c} \leadsto_\kappa w\boldsymbol{c}\boldsymbol{c}'$, and set $P(w\boldsymbol{c} \leadsto_\kappa w\boldsymbol{c}\boldsymbol{c}') := Prob((\kappa(w\boldsymbol{c}), \alpha))$ with $\boldsymbol{c}'$ as defined above.

Let $Path_\kappa$ be the set of all *paths* $w = \boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(n)} \in \mathcal{C}^+$ such that for every $1 \leq i < n$ we have $\boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(i)} \rightsquigarrow_\kappa \boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(i+1)}$. Given $\boldsymbol{c} \in \mathcal{C}$, we denote by $MPath_\kappa(\boldsymbol{c})$ the set of all maximal paths initiated in $\boldsymbol{c}$, i.e. $\boldsymbol{c}^{(1)} = \boldsymbol{c}$ and $\boldsymbol{c}^{(n)} = \boldsymbol{0}$. Given $w = \boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(n)} \in MPath_\kappa(\boldsymbol{c})$, we define $\Pr_\kappa[w] = \Pi_{j=1}^{n-1} P(\boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(i)} \rightsquigarrow_\kappa \boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(i+1)})$.

We write $w\boldsymbol{c}^{(1)} \rightsquigarrow_\kappa^* w\boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(i)}$ if for every $1 \leq j < i$ we have $w\boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(j)} \rightsquigarrow_\kappa w\boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(j+1)}$. We denote by $\kappa\!\downarrow$ the restriction of κ to those w that satisfy $\langle Y \rangle \rightsquigarrow_\kappa^* w$ for some $Y \in \Gamma$. In other words, $\kappa\!\downarrow$ is a partial function such that for every w satisfying $\langle Y \rangle \rightsquigarrow_\kappa^* w$ for some $Y \in \Gamma$ we have $\kappa\!\downarrow\!(w) = \kappa(w)$.

Lemma D.1. *For every $\boldsymbol{c} \in \mathcal{C} \setminus \{\boldsymbol{0}\}$ we have $\sum_{w \in MPath_\kappa(\boldsymbol{c})} \Pr_\kappa[w] = 1$, and hence $(MPath_\kappa(\boldsymbol{c}), \Pr_\kappa)$ is a discrete probability space.*

Proof. First, consider the case that $\boldsymbol{c} = \langle X \rangle$ for some $X \in \Gamma$. One can easily show that for every multiset scheduler κ the partial function $\kappa\!\downarrow$ is equal to Λ_σ for a suitable online scheduler σ. However, then for every $\boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(i)} \in MPath_\kappa(\langle X \rangle)$ there is a unique family tree $t \in \mathcal{T}_X$ (and vice versa) such that $\Pr[t] = \Pr_\kappa\left[\boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(i)}\right]$ and $\boldsymbol{z}^{(j)}(t) = \boldsymbol{c}^{(j)}$ for $1 \leq j \leq i$ (here every $\boldsymbol{z}^{(j)}$ is evaluated with respect to σ).

Consequently,

$$\sum_{w \in MPath_\kappa(\langle X \rangle)} \Pr_\kappa[w] = \sum_{w \in MPath_\kappa(\langle X \rangle)} \Pr_{\Lambda_\sigma}[w] = \sum_{t \in \mathcal{T}_X} \Pr[t] = 1$$

Consider now an arbitrary $\boldsymbol{c} \in \mathcal{C} \setminus \{\boldsymbol{0}\}$. Finally, starting in $\boldsymbol{c}$, every computation according to an arbitrary multiset scheduler κ can be considered as a parallel composition of $|\boldsymbol{c}|$ computations initiated in individual elements of $\boldsymbol{c}$. Hence, the probability of reaching $\boldsymbol{0}$ from $\boldsymbol{c}$ is equal to the probability of reaching $\boldsymbol{0}$ from every symbol in $\boldsymbol{c}$, which is one. $\square$

Given a multiset scheduler κ and $\boldsymbol{c} \in \mathcal{C}$, we define the *random width* $S^\kappa_{\boldsymbol{c}}$ on $MPath_\kappa(\boldsymbol{c})$ as follows: Given $w = \boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(i)} \in MPath_\kappa(\boldsymbol{c})$, we define $S^\kappa_{\boldsymbol{c}}(w) = \max\{|\boldsymbol{c}^{(1)}|, \ldots, |\boldsymbol{c}^{(i)}|\}$ (here each $|\boldsymbol{c}^{(i)}|$ is the size of the multiset $\boldsymbol{c}^{(i)}$).

Lemma D.2. *For every online scheduler σ there is a multiset scheduler κ such that $\mathbb{E}[S^\sigma] = \mathbb{E}\left[S^\kappa_{\langle X_0 \rangle}\right]$. For every multiset scheduler κ there is an online scheduler σ such that $\mathbb{E}\left[S^\kappa_{\langle X_0 \rangle}\right] = \mathbb{E}[S^\sigma]$.*

Proof. Let σ be an online scheduler. It follows directly from definition that $\Lambda_\sigma = \kappa\!\downarrow$ for some multiset scheduler κ. Also by Lemma C.2, for every family tree t we have $\Pr[t] = \Pr_\kappa\left[\boldsymbol{z}^{(1)}(t) \cdots \boldsymbol{z}^{(n)}(t)\right]$ where n is the least number satisfying $\boldsymbol{z}^{(n)}(t) = \boldsymbol{0}$. Because $S^\sigma(t) = S^\kappa_{\langle X_0 \rangle}(\boldsymbol{z}^{(1)}(t) \cdots \boldsymbol{z}^{(n)}(t))$, we obtain that $\mathbb{E}[S^\sigma] = \mathbb{E}\left[S^\kappa_{\langle X_0 \rangle}\right]$. On the other hand, every multiset scheduler κ satisfies $\kappa\!\downarrow = \Lambda_\sigma$ for a suitable online scheduler σ, which implies the second half of the lemma. $\square$

Let us denote by $\mathcal{S}_{mu}$ the set of all multiset schedulers. Given $\kappa \in \mathcal{S}_{mu}$, $n \geq 1$, and $\boldsymbol{c} \in \mathcal{C}$, we define $S^{\kappa,n}_{\boldsymbol{c}} = \max\{S^\kappa_{\boldsymbol{c}}, n\}$. We define $val(n, \boldsymbol{c}) = \inf_{\kappa \in \mathcal{S}_{mu}} \mathbb{E}[S^{\kappa,n}_{\boldsymbol{c}}]$. We say that κ is *optimal in* $[n, \boldsymbol{c}]$ if $\mathbb{E}[S^{\kappa,n}_{\boldsymbol{c}}] = val(n, \boldsymbol{c})$.

Lemma D.3. $val(n, \boldsymbol{c}) = \min_{\kappa \in \mathcal{S}_{mu}} \sum_{\boldsymbol{c} \rightsquigarrow_\kappa \boldsymbol{c}\boldsymbol{c}'} P(\boldsymbol{c} \rightsquigarrow_\kappa \boldsymbol{c}\boldsymbol{c}') \cdot val(\max\{n, |\boldsymbol{c}|\}, \boldsymbol{c}')$

Proof. Given a multiset scheduler κ, we denote by $\bar{\kappa}$ a multiset scheduler defined by $\bar{\kappa}(w) = \kappa(\boldsymbol{c}w)$. For every $w = \boldsymbol{c}^{(1)}\boldsymbol{c}^{(2)} \cdots \boldsymbol{c}^{(n)} \in MPath_\kappa(\boldsymbol{c})$ we have

$$S^{\kappa,n}_{\boldsymbol{c}^{(1)}}(w) = S^{\bar{\kappa},\max\{n,|\boldsymbol{c}|\}}_{\boldsymbol{c}^{(2)}}(\boldsymbol{c}^{(2)} \cdots \boldsymbol{c}^{(n)})$$

and $\Pr[w] = P(\boldsymbol{c}^{(1)} \rightsquigarrow_\kappa \boldsymbol{c}^{(1)}\boldsymbol{c}^{(2)}) \cdot \Pr_{\bar{\kappa}}\left[\boldsymbol{c}^{(2)} \cdots \boldsymbol{c}^{(n)}\right]$. It follows that

$$\begin{aligned}
\mathbb{E}[S_{\boldsymbol{c}}^{\kappa,n}] &= \sum_{\boldsymbol{c}^{(1)}\cdots\boldsymbol{c}^{(n)} \in MPath_\kappa(\boldsymbol{c})} \Pr_\kappa\left[\boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(n)}\right] \cdot S_{\boldsymbol{c}}^{\kappa,n}(\boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(n)}) \\
&= \sum_{\boldsymbol{c} \rightsquigarrow_\kappa \boldsymbol{c}\boldsymbol{c}^{(2)}} P(\boldsymbol{c} \rightsquigarrow_\kappa \boldsymbol{c}\boldsymbol{c}^{(2)}) \sum_{\boldsymbol{c}^{(2)}\cdots\boldsymbol{c}^{(n)} \in MPath_{\bar{\kappa}}(\boldsymbol{c}^{(2)})} \Pr_{\bar{\kappa}}\left[\boldsymbol{c}^{(2)} \cdots \boldsymbol{c}^{(n)}\right] \cdot S_{\boldsymbol{c}^{(2)}}^{\bar{\kappa},\max\{n,|\boldsymbol{c}|\}}(\boldsymbol{c}^{(2)} \cdots \boldsymbol{c}^{(n)}) \\
&= \sum_{\boldsymbol{c} \rightsquigarrow_\kappa \boldsymbol{c}\boldsymbol{c}^{(2)}} P(\boldsymbol{c} \rightsquigarrow_\kappa \boldsymbol{c}\boldsymbol{c}^{(2)}) \cdot \mathbb{E}\left[S_{\boldsymbol{c}^{(2)}}^{\bar{\kappa},\max\{n,|\boldsymbol{c}|\}}\right]
\end{aligned}$$

$$\inf_{\kappa \in \mathcal{S}_{mu}} \sum_{\boldsymbol{c} \rightsquigarrow_\kappa \boldsymbol{c}\boldsymbol{c}^{(2)}} P(\boldsymbol{c} \rightsquigarrow_\kappa \boldsymbol{c}\boldsymbol{c}^{(2)}) \cdot \mathbb{E}\left[S_{\boldsymbol{c}^{(2)}}^{\bar{\kappa},\max\{n,|\boldsymbol{c}|\}}\right] =$$
$$\min_{\zeta \in \mathcal{S}_{mu}} \left(\sum_{\boldsymbol{c} \rightsquigarrow_\zeta \boldsymbol{c}\boldsymbol{c}^{(2)}} P(\boldsymbol{c} \rightsquigarrow_\zeta \boldsymbol{c}\boldsymbol{c}^{(2)}) \cdot \inf_{\lambda \in \mathcal{S}_{mu}} \mathbb{E}\left[S_{\boldsymbol{c}^{(2)}}^{\lambda,\max\{n,|\boldsymbol{c}|\}}\right] \right)$$

Indeed, the inequality $\geq$ is obvious. For the opposite, observe that given a scheduler ζ from the right hand side and schedulers $\lambda_{\boldsymbol{c}^{(2)}}$ for every $\boldsymbol{c}^{(2)}$ such that $\boldsymbol{c} \rightsquigarrow_\zeta \boldsymbol{c}\boldsymbol{c}^{(2)}$, we may construct a scheduler κ on the left hand side that chooses the first step according to the ζ (reaching $\boldsymbol{c}\boldsymbol{c}^{(2)}$) and then behaves as $\lambda_{\boldsymbol{c}^{(2)}}$ (thus $\bar{\kappa}(\boldsymbol{c}^{(2)} \cdots \boldsymbol{c}^{(n)}) = \lambda_{\boldsymbol{c}^{(2)}}(\boldsymbol{c}^{(2)} \cdots \boldsymbol{c}^{(n)})$. This gives

$$\sum_{\boldsymbol{c} \rightsquigarrow_\kappa \boldsymbol{c}\boldsymbol{c}^{(2)}} P(\boldsymbol{c} \rightsquigarrow_\kappa \boldsymbol{c}\boldsymbol{c}^{(2)}) \cdot \mathbb{E}\left[S_{\boldsymbol{c}^{(2)}}^{\bar{\kappa},\max\{n,|\boldsymbol{c}|\}}\right] = \sum_{\boldsymbol{c} \rightsquigarrow_\zeta \boldsymbol{c}\boldsymbol{c}^{(2)}} P(\boldsymbol{c} \rightsquigarrow_\zeta \boldsymbol{c}\boldsymbol{c}^{(2)}) \cdot \mathbb{E}\left[S_{\boldsymbol{c}^{(2)}}^{\lambda_{\boldsymbol{c}^{(2)}},\max\{n,|\boldsymbol{c}|\}}\right]$$

Hence, $val(n, \boldsymbol{c}) = \inf_{\kappa \in \mathcal{S}_{mu}} \mathbb{E}[S_{\boldsymbol{c}}^{\kappa,n}] = \min_{\kappa \in \mathcal{S}_{mu}} \sum_{\boldsymbol{c} \rightsquigarrow_\kappa \boldsymbol{c}\boldsymbol{c}^{(2)}} P(\boldsymbol{c} \rightsquigarrow_\kappa \boldsymbol{c}\boldsymbol{c}^{(2)}) \cdot val(\max\{n, |\boldsymbol{c}|\}, \boldsymbol{c}^{(2)})$. $\square$

We are now ready to prove Theorem 5.4.

Theorem 5.4. *There is an online scheduler σ such that $\mathbb{E}[S^\sigma] = \inf_{\{\pi | \pi \text{ is online}\}} \mathbb{E}[S^\pi]$.*

Proof. We prove that for every $n \geq 1$ there is a multiset scheduler $\kappa[n]$ which is optimal in $[n, \boldsymbol{c}]$ for every $\boldsymbol{c} \in \mathcal{C}$. Then we obtain the desired result from the special case for $n = 1$ because by Lemma D.2 there is an online scheduler σ such that

$$\mathbb{E}[S^\sigma] = \mathbb{E}\left[S_{\langle X_0 \rangle}^{\kappa[1]}\right] = val(1, \langle X_0 \rangle) = \inf_{\lambda \in \mathcal{S}_{mu}} \mathbb{E}\left[S_{\langle X_0 \rangle}^{\lambda}\right] = \inf_{\pi \in \mathcal{S}_{on}} \mathbb{E}[S^\pi]$$

Given $w = \boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(i)} \in \mathcal{C}^+$, we write $wd(w) = \max\{|\boldsymbol{c}^{(1)}|, \ldots, |\boldsymbol{c}^{(i)}|\}$. For every $n \geq 1$ we define $\kappa[n]$ as follows: Given $w \in \mathcal{C}^+$, we define $\kappa[n](w)$ to *minimize* the following number

$$\sum_{w \rightsquigarrow_{\kappa[n]} w\boldsymbol{c}} P(w \rightsquigarrow_{\kappa[n]} w\boldsymbol{c}) \cdot val(\max\{n, wd(w)\}, \boldsymbol{c})$$

Given $n, m \geq 1$, $\boldsymbol{c} \in \mathcal{C}$, and $\boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(i)} \in MPath_\kappa(\boldsymbol{c})$, we define

$$S_{\boldsymbol{c},m}^{\kappa[n],n}(w) = \begin{cases} S_{\boldsymbol{c}}^{\kappa[n],n}(\boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(i)}) & \text{if } i \leq m \\ 0 & \text{otherwise.} \end{cases}$$

Clearly, for every $n \geq 1$ we have $\mathbb{E}\left[S_{\boldsymbol{c},m}^{\kappa[n],n}\right] \leq \mathbb{E}\left[S_{\boldsymbol{c},m+1}^{\kappa[n],n}\right] \leq \mathbb{E}\left[S_{\boldsymbol{c}}^{\kappa[n],n}\right]$, and by the monotone convergence theorem, $\lim_{m\to\infty} \mathbb{E}\left[S_{\boldsymbol{c},m}^{\kappa[n],n}\right] = \mathbb{E}\left[S_{\boldsymbol{c}}^{\kappa[n],n}\right]$. We prove that $\mathbb{E}\left[S_{\boldsymbol{c},m}^{\kappa[n],n}\right] \leq val(n, \boldsymbol{c})$ for all $m \geq 1$, which gives $\mathbb{E}\left[S_{\boldsymbol{c}}^{\kappa[n],n}\right] \le val(n, \boldsymbol{c})$. The case $m = 1$ is trivial. For $m > 1$ we have by Lemma D.3 and induction hypothesis

$$\begin{aligned}
val(n, \boldsymbol{c}) &= \min_{\lambda \in \mathcal{S}_{mu}} \sum_{\boldsymbol{c} \rightsquigarrow_\lambda \boldsymbol{c}\boldsymbol{c}'} P(\boldsymbol{c} \rightsquigarrow_\lambda \boldsymbol{c}\boldsymbol{c}') \cdot val(\max\{n, |\boldsymbol{c}|\}, \boldsymbol{c}') \\
&= \sum_{\boldsymbol{c} \rightsquigarrow_{\kappa[n]} \boldsymbol{c}\boldsymbol{c}'} P(\boldsymbol{c} \rightsquigarrow_{\kappa[n]} \boldsymbol{c}\boldsymbol{c}') \cdot val(\max\{n, |\boldsymbol{c}|\}, \boldsymbol{c}') \\
&\ge \sum_{\boldsymbol{c} \rightsquigarrow_{\kappa[n]} \boldsymbol{c}\boldsymbol{c}'} P(\boldsymbol{c} \rightsquigarrow_{\kappa[n]} \boldsymbol{c}\boldsymbol{c}') \cdot \mathbb{E}\left[S_{\boldsymbol{c}',m-1}^{\kappa[\max\{n,|\boldsymbol{c}|\}],\max\{n,|\boldsymbol{c}|\}}\right] \\
&= \mathbb{E}\left[S_{\boldsymbol{c},m}^{\kappa,n}\right]
\end{aligned}$$

□

We turn to the proof of Theorem 5.5. We start by recalling the result:

Theorem 5.5. *For sufficiently small p and r (it suffices to choose, e.g., $r := 10^{-5}$ and $p := \frac{1}{2}r$), any online scheduler that minimizes the expected completion space of the task system*

$$\begin{array}{llll}
X \xhookrightarrow{1/8} \langle X, X\rangle & X \xhookrightarrow{1/8} \langle Y, Z\rangle & X \xhookrightarrow{3/4} \emptyset & Z \xhookrightarrow{r} \langle U, U\rangle \quad Z \xhookrightarrow{1-r} \emptyset \\
Y \xhookrightarrow{p} \langle Z, Z\rangle & Y \xhookrightarrow{1-p} \emptyset & & U \xhookrightarrow{1} \emptyset
\end{array}$$

requires infinite memory.

Note that the values $r := 10^{-5}$ and $p := \frac{1}{2}$ are not unique. In the course of the proof we provide a series of inequalities involving p and r so that whenever these inequelities are satisfied, the example works as expected.

To simplify notation in the following, we identify any word $\alpha \in \Gamma^*$ with the multiset which counts how often a letter of Γ appears in α, e.g. the word YXY is one representation of the multiset $\langle X, Y, Y\rangle$.

The proof of Theorem 5.5 relies mainly on the following crucial proposition:

Proposition D.4.

1. *If κ is optimal in $[n + 2, YZX^n]$, then $\kappa(YZX^n) = Y$*
2. *If κ is optimal in $[n + 3, YZX^n]$, then $\kappa(YZX^n) = Z$*

Let us first explain the intuition behind this proposition, and how from this result Theorem 5.5 follows. Assume that we want to minimize the expected completion space starting in YZX^n when the maximum in the history is either $n + 2$, or $n + 3$. What type we have to choose in YZX^n? First, choosing X is not minimizing (in both cases), because this would increase the number of tasks with much higher probability than choosing either Y, or Z. Now the difference between Y and Z is that the former generates 3 tasks (via ZZ) with a very small probability, while the latter generates at most 2 tasks but with higher probability. It follows that if the maximum in the history is $n + 2$, the better choice is Y, because although both Y and Z may exceed $n + 2$, the Y exceeds with smaller probability. On the other hand, if the maximum in the history is $n + 3$, then the better choice is Z, because it never exceeds $n + 3$ before getting to X^n (as opposite to Y).

Before giving a surprisingly non-trivial proof of Proposition D.4, let us show how this proposition implies non-existence of a finite memory online scheduler that minimizes the expected completion space.

Proof. (of Theorem 5.5.) First, let us define a notion of finite memory multiset scheduler. We say that a multiset scheduler κ is *finite memory* if there is a finite state automaton $\mathcal{A}$ over an alphabet Σ and a function $h : \mathcal{C}^+ \to \Sigma$ such that for every $\boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(n)} \in \mathcal{C}^+$ the value of $\kappa(\boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(n)})$ depends only on the state of $\mathcal{A}$ after reading $h(\boldsymbol{c}^{(1)}) \cdots h(\boldsymbol{c}^{(n)})$ and on $\boldsymbol{c}^{(n)}$. It follows from the proof of Lemma D.2 that if σ is a finite memory online scheduler that minimizes the expected completion space, then Λ_σ is a finite memory multiset scheduler that minimizes the expected completion space. Hence, it suffices to show that there is no finite memory multiset scheduler that minimizes the expected completion space.

To obtain a contradiction, let us assume that κ is a finite memory multiset scheduler that minimizes the expected completion space. Given a path $\boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(i)} \in Path_\kappa$ we denote by $h(\boldsymbol{c}^{(1)} \cdots \boldsymbol{c}^{(i)})$ the word $h(\boldsymbol{c}^{(1)}) \cdots h(\boldsymbol{c}^{(i)})$. For every $n \geq 1$ we denote by w_n the path $X \cdot X^2 \cdots X^{n+3} \cdot X^{n+2} \cdots X$. There are two numbers $n < m$ such that the automaton $\mathcal{A}$ enters the same state after reading either of the words $h(w_n)$ and $h(w_m)$. Let us consider the paths $w'_n = w_n \cdot X^2 \cdot X^3 \cdots X^{m+1} \cdot YZX^m$ and $w'_m = w_m \cdot X^2 \cdot X^3 \cdots X^{m+1} \cdot YZX^m$. The automaton enters the same state q after reading either of the words $h(w'_n)$ and $h(w'_m)$. Hence, $\kappa(w'_n) = \kappa(w'_m)$. However, if κ minimizes the completion space, then, by Proposition D.4, we have $\kappa(w'_n) = Y$ and $\kappa(w'_m) = Z$, a contradiction. This proves Theorem 5.5. □

It remains to prove Proposition D.4.

For the rest of this proof we denote $s := \frac{1}{8}$. To prove Proposition D.4 we make use of the following three technical lemmas (Lemma D.5, Lemma D.6, and Lemma D.7 below):

Lemma D.5. *There are numbers $k, \ell > 0$ such that for every $n \geq 1$ and every $i > 0$ we have*

1. $val(n+i+1, X^n) < val(n+i, X^n) + k$
2. $val(n+i, X^n) < val(n+i+1, X^n) - \ell$

[More concretely, we may choose $k := 1$, and $\ell := (\frac{3}{4})^4 \cdot \frac{6}{7}$]

Proof.

ad 1. For every multiset scheduler κ we have $\mathbb{E}\left[S^{\kappa,n+i+1}_{X^n}\right] \leq \mathbb{E}\left[S^{\sigma,n+i}_{X^n} + 1\right] = \mathbb{E}\left[S^{\kappa,n+i}_{X^n}\right] + 1$. Hence, it suffices to define $k = 1$.

ad 2. Let κ be a multiset scheduler which is optimal in $[n+i+1, X^n]$. Let κ_n be a scheduler that behaves similarly as κ except that in configurations of size at most $n-2$, κ_n prefers either Y or Z to X. Clearly, $\mathbb{E}\left[S^{\kappa_n,n+i}_{X^n}\right] = \mathbb{E}\left[S^{\kappa,n+i}_{X^n}\right]$ and $\mathbb{E}\left[S^{\kappa_n,n+i+1}_{X^n}\right] = \mathbb{E}\left[S^{\kappa,n+i+1}_{X^n}\right]$. Now using an arbitrary scheduler a configuration of the form X^{n-4} is reachable from X^n with the probability $(1-2s)^4$ via a path w_{dec} of the form $X^n X^{n-1} \cdots X^{n-4}$. Let us denote A_n the set of all paths of $Path_{\kappa_n}$ that start with w_{dec}, reach an empty pool of tasks, and after w_{dec} never reach a configuration of the form $\{Y, Z\}^* X^m$ where $m > n-4$. It is easy to see that for every $w \in A_n$ we have $S^{\kappa_n,n+i}_{X^n}(w) = n+i$ and $S^{\kappa_n,n+i+1}_{X^n}(w) = n+i+1$, which implies that

$$\begin{aligned}
\mathbb{E}\left[S^{\kappa_n,n+i}_{X^n}\right] &\leq \Pr[A_n]\left(\mathbb{E}\left[S^{\kappa_n,n+i+1}_{X^n} \mid A_n\right] - 1\right) + (1-\Pr[A_n])\mathbb{E}\left[S^{\kappa_n,n+i+1}_{X^n} \mid MPath_{\kappa_n}(X^n) \setminus A_n\right] \\
&= \mathbb{E}\left[S^{\kappa_n,n+i+1}_{X^n}\right] - \Pr[A_n] \\
&= \mathbb{E}\left[S^{\kappa,n+i+1}_{X^n}\right] - \Pr[A_n]
\end{aligned}$$

Thus every scheduler λ, which is optimal in $[n+i, X^n]$, satisfies $\mathbb{E}\left[S^{\lambda,n+i}_{X^n}\right] \leq \mathbb{E}\left[S^{\kappa,n+i+1}_{X^n}\right] - \Pr[A_n]$ and hence $val(n+i, X^n) \leq val(n+i+1, X^n) - \Pr[A_n]$. We prove that $\Pr[A_n] > (1-2s)^4(1-\frac{s}{1-s}) > 0$ (hence, it suffices to define $\ell := (1-2s)^4(1-\frac{s}{1-s})$).

Let us denote by h_n the probability that using κ_n we reach the empty pool of tasks from X^{n-4} and at the same time never reach a configuration of the form $\{Y, Z\}^* X^m$ where $m > n-4$. We prove that $h_n > 1-\frac{s}{1-s} > 0$ using basic results of the theory of random walks, and obtain $\Pr[A_n] = \Pr[w_{dec}] \cdot h_n > (1-2s)^4(1-\frac{s}{1-s})$.

Let us define a sequence of random variables $Z_1, Z_2, \cdots$ such that for every path $w \in MPath_{\kappa_n}(X^{n-2})$ the value $Z_i(w)$ is the number of X-tasks in the i-th moment when X is chosen to move (i.e. by the definition of κ_n, the i-th moment when the current configuration is of the form X^m for some m). It is easy to see that $Z_1, Z_2, \ldots$ is a random walk on the set $\mathbb{Z}^+$ of non-negative whole numbers where the probability of going from $k > 0$ to $k+1$ is s and the probability of going from $k > 0$ to $k-1$ is $1-s$. It can be easily shown that the probability of reaching 0 from $n-2$ while avoiding $n-3$ is equal to $\frac{1-\frac{s}{1-s}}{1-(\frac{s}{1-s})^{n-2+1}}$. However, this is precisely the probability h_n and hence $h_n > 1 - \frac{s}{1-s} > 0$.

□

In the rest of this section we write (m, α) instead of $val(m, \alpha)$.

Lemma D.6. *Let $c, d > 0$ such that $r\,c + (1-r)\,d = 1$ and $r < \frac{d\,\ell}{16\,c\,k}$. Then for $i \geq 2$ we have*

$$rc(n+i+1, X^n) + (1-r)d(n+i, X^n) < s(n+i+1, X^{n+1}) + (1-s)(n+i, X^{n-1})$$

Proof. Denoting $L = rc(n+i+1, X^n) + (1-r)d(n+i, X^n)$ we have

$$\begin{aligned}
L &= rcs(n+i+1, X^{n+1}) + rcs(n+i+1, YZX^{n-1}) + rc(1-2s)(n+i+1, X^{n-1}) \\
&\quad + (1-r)ds(n+i, X^{n+1}) + (1-r)ds(n+i, YZX^{n-1}) + (1-r)d(1-2s)(n+i, X^{n-1}) \\
&= rcs(n+i+1, X^{n+1}) + rcs(n+i+1, X^{n-1}) + rc(1-2s)(n+i+1, X^{n-1}) && (i \geq 2) \\
&\quad + (1-r)ds(n+i, X^{n+1}) + (1-r)ds(n+i, X^{n-1}) + (1-r)d(1-2s)(n+i, X^{n-1}) \\
&= rcs(n+i+1, X^{n+1}) + rc(1-s)(n+i+1, X^{n-1}) \\
&\quad + (1-r)ds(n+i, X^{n+1}) + (1-r)d(1-s)(n+i, X^{n-1}) \\
&< rcs(n+i, X^{n+1}) + rcsk + rc(1-s)(n+i, X^{n-1}) + rc(1-s)k && (Lemma\ D.5) \\
&\quad + (1-r)ds(n+i+1, X^{n+1}) - (1-r)ds\ell + (1-r)d(1-s)(n+i, X^{n-1})
\end{aligned}$$

Now because we have chosen $s = \frac{1}{8}$, and r so that $r < \frac{1}{2}$, we obtain

$$rcsk + rc(1-s)k = rck < \frac{d\ell}{16} = \frac{1}{2}ds\ell < (1-r)ds\ell$$

and thus

$$\begin{aligned}
L &< s(rc + (1-r)d)(n+i+1, X^{n+1}) + (1-s)(rc + (1-r)d)(n+i, X^{n-1}) \\
&= s(n+i+1, X^{n+1}) + (1-s)(n+i, X^{n-1})
\end{aligned}$$

□

For $W \in \{X, Y, Z\}$, we write

$$I^W_{n,\alpha} = \inf_{\substack{\lambda \in \mathcal{S}_{mu} \\ \lambda(\alpha) = W}} \mathbb{E}\left[S^{\lambda,n}_{\alpha}\right]$$

Lemma D.7.

1. *if κ is optimal in $[n+2, ZZX^n]$, then $\kappa(ZZX^n) = Z$*
2. *if κ is optimal in $[n+3, ZZZX^n]$, then $\kappa(ZZZX^n) = Z$*

Proof.

ad 1. We have

$$I^X_{n+2,ZZX^n} = s(n+3, ZZX^{n+1}) + s(n+3, YZZZX^{n-1}) + (1-2s)(n+2, ZZX^{n-1})$$

and

$$I^Z_{n+2,ZZX^n} = r(n+3, X^n) + (1-r)(n+2, X^n)$$

However, we have chosen r so that $r < \frac{\ell}{16k}$, and hence $r < \frac{d\ell}{16ck}$ for $c = d = 1$. Thus by Lemma D.6

$$I^Z_{n+2,ZZX^n} < s(n+3, X^{n+1}) + (1-s)(n+2, X^{n-1}) \leq I^X_{n+2,ZZX^n}$$

ad 2. We have

$$I^X_{n+3,ZZZX^n} = s(n+4, ZZZX^{n+1}) + s(n+4, YZZZZX^{n-1}) + (1-2s)(n+3, ZZZX^{n-1})$$

and

$$I^Z_{n+3,ZZZX^n} = r(n+4, X^n) + (1-r)(n+3, X^n)$$

We have chosen r so that $r < \frac{\ell}{16k}$, and hence $r < \frac{d\ell}{16ck}$ for $c = d = 1$. Thus by Lemma D.6

$$I^Z_{n+3,ZZZX^n} < s(n+4, X^{n+1}) + (1-s)(n+3, X^{n-1}) \leq I^X_{n+3,ZZZX^n}$$

□

We are now in the position to prove Proposition D.4.

Proof. (of Proposition D.4.)

ad 1. We have

$$I^X_{n+2,YZX^n} = s(n+3, YZX^{n+1}) + s(n+3, YZYZX^{n-1}) + (1-2s)(n+2, YZX^{n-1})$$

and

$$\begin{aligned}
I^Z_{n+2,YZX^n} &= r(n+3, X^n) + (1-r)(n+2, YX^n) \\
&= r(n+3, X^n) + (1-r)p(n+2, ZZX^n) + (1-r)(1-p)(n+2, X^n) \\
&= r(n+3, X^n) + (1-r)pr(n+3, X^n) && (Lemma\ D.7) \\
&\quad + (1-r)p(1-r)(n+2, X^n) + (1-r)(1-p)(n+2, X^n) \\
&= r(1+(1-r)p)(n+3, X^n) + (1-r)(1-pr)(n+2, X^n)
\end{aligned}$$

Now observe that setting $c = 1 + (1-r)p$ and $d = (1-pr)$ gives $rc + (1-r)d = 1$. Also a straightforward computation reveals that we have chosen r so that $r < \frac{d\ell}{16ck}$ Hence, by Lemma D.6, $I^Z_{n+2,YZX^n} < I^X_{n+2,YZX^n}$.

Finally, we have chosen p such that

$$p = \frac{r}{2} < \frac{r\ell}{rk + \ell - r\ell + r^2\ell} \tag{9}$$

which gives us

$$\begin{aligned}
I^Y_{n+2,YZX^n} &= p(n+3, ZZZX^n) + (1-p)(n+2, ZX^n) \\
&= pr(n+4, X^n) + p(1-r)(n+3, X^n) + (1-p)(n+2, X^n) && (Lemma\ D.7)
\end{aligned}$$

$$
\begin{aligned}
&= pr(n+4, X^n) + p(1-r)(n+3, X^n) \\
&\quad + (1-r)(1-pr)(n+2, X^n) + ((1-p) - (1-r)(1-pr))(n+2, X^n) \\
&< pr(n+3, X^n) + prk + p(1-r)(n+3, X^n) && (Lemma\ D.5) \\
&\quad + (1-r)(1-pr)(n+2, X^n) + ((1-p) - (1-r)(1-pr))(n+3, X^n) \\
&\quad - ((1-p) - (1-r)(1-pr))\ell \\
&< (p + (1-p) - (1-r)(1-pr))(n+3, X^n) + (1-r)(1-pr)(n+2, X^n) && (Eq.\ (9)) \\
&= r(1 + (1-r)p)(n+3, X^n) + (1-r)(1-pr)(n+2, X^n) \\
&= I^Z_{n+2,YZX^n}
\end{aligned}
$$

Hence, $I^Y_{n+2,YZX^n} < I^Z_{n+2,YZX^n} < I^X_{n+2,YZX^n}$, and $\kappa(YZX^n) = Y$ if κ is optimal in $[n+2, YZX^n]$.

ad 2. We have

$$
\begin{aligned}
I^Y_{n+3,YZX^n} &= p(n+3, ZZZX^n) + (1-p)(n+3, ZX^n) \\
&= pr(n+4, X^n) + p(1-r)(n+3, X^n) + (1-p)(n+3, X^n) && (Lemma\ D.7) \\
&= pr(n+4, X^n) + (1-pr)(n+3, X^n)
\end{aligned}
$$

and

$$
I^Z_{n+3,YZX^n} = (n+3, X^n)
$$

which implies $I^Z_{n+3,YZX^n} < I^Y_{n+3,YZX^n}$. Also

$$
\begin{aligned}
I^Z_{n+3,YZX^n} &= (n+3, X^n) \\
&= s(n+3, X^{n+1}) + s(n+3, YZX^{n-1}) + (1-2s)(n+3, X^{n-1}) \\
&= s(n+3, X^{n+1}) + (1-s)(n+3, X^{n-1})
\end{aligned}
$$

and

$$
\begin{aligned}
I^X_{n+3,YZX^n} &= s(n+3, YZX^{n+1}) + s(n+3, YZYZX^{n-1}) + (1-2s)(n+3, X^{n-1}) \\
&= sp(n+4, ZZZX^{n+1}) + s(1-p)(n+3, X^{n+1}) && (1.) \\
&\quad + s(n+3, YZYZX^{n-1}) + (1-2s)(n+3, X^{n-1}) \\
&\geq sp(n+4, X^{n+1}) + s(1-p)(n+3, X^{n+1}) + (1-s)(n+3, X^{n-1}) \\
&> sp(n+3, X^{n+1}) + sp\ell + s(1-p)(n+3, X^{n+1}) && (Lemma\ D.6) \\
&\quad + (1-s)(n+3, X^{n-1}) \\
&= s(n+3, X^{n+1}) + (1-s)(n+3, X^{n-1}) + sp\ell
\end{aligned}
$$

Hence $I^Z_{n+3,YZX^n} < I^X_{n+3,YZX^n}$, and thus $\kappa(YZX^n) = Z$ if κ is optimal in $[n+3, YZX^n]$.

□

E Optimizing the Bound for Continuing Task Systems

It follows from Theorem 4.3 that, for large k, the best bound is obtained by maximizing $\boldsymbol{v}_{min}$. Now we show for continuing task systems that the *best* (i.e., the largest) $\boldsymbol{v}_{min}$ can be approximated in polynomial time. More formally, define the "optimal" $\boldsymbol{v}_{min}$ by $\boldsymbol{v}^{opt}_{min} := \sup\left\{d \in \mathbb{R} \mid \exists \boldsymbol{v} \in [0,\infty)^{\Gamma} : d\mathbf{1} \leq \boldsymbol{v} \geq \boldsymbol{f}(\boldsymbol{v})\right\}$. We show that one can compute, in polynomial time, an ϵ-approximation of $\boldsymbol{v}^{opt}_{min}$, i.e., a number d with $|d - \boldsymbol{v}^{opt}_{min}| \leq \epsilon$. As we consider continuing task systems we can, for all $Y \in \Gamma$, write $\boldsymbol{f}_Y$ as $\boldsymbol{f}_Y(\boldsymbol{x}) = \boldsymbol{x}_Y \cdot q_Y(\boldsymbol{x}) + c_Y$ where $q_Y(\boldsymbol{x})$ is linear. Note that $q_Y(\mathbf{1}) + c_Y = 1$. We can show the following theorem whose proof closely follows a proof of [15]:

Theorem E.1. *Given a continuing task system whose coefficients are given as b-bit rationals, one can compute an ϵ-approximation of $\boldsymbol{v}_{min}^{opt}$ in time $\mathrm{poly}(|\Gamma|, b, \log\frac{1}{\epsilon})$ in the usual (Turing) model by solving the following system: maximize d subject to $0 \le d \le \boldsymbol{v}_Y \ge \boldsymbol{f}_Y(\boldsymbol{v})$*

Proof. The proof follows a proof of [15]. We claim that the following systems (10) and (11) are equivalent:

$$\text{maximize } d \text{ subject to } \quad 0 \le d \le \boldsymbol{v}_Y \ge \boldsymbol{f}_Y(\boldsymbol{v}) \tag{10}$$

$$\text{maximize } d \text{ subject to } \quad \left\{ \begin{array}{l} 0 \le d \le \boldsymbol{v}_Y \\ s_Y = 1 - q_Y(\boldsymbol{v}) \\ \begin{pmatrix} \boldsymbol{v}_Y & \sqrt{c_Y} \\ \sqrt{c_Y} & s_Y \end{pmatrix} \text{positive semidefinite} \end{array} \right\} \tag{11}$$

For the equivalence of (10) and (11) note that the condition on the matrices being positive semidefinite is equivalent to $\boldsymbol{v}_Y \cdot s_Y \ge c_Y$. Substituting $s_Y = 1 - q_Y(\boldsymbol{v})$ yields $\boldsymbol{v}_Y \cdot (1 - q_Y(\boldsymbol{v})) \ge c_Y$ which is, using $\boldsymbol{f}_Y(\boldsymbol{v}) = \boldsymbol{v}_Y \cdot q_Y(\boldsymbol{v}) + c_Y$, equivalent to $\boldsymbol{v}_Y \ge \boldsymbol{f}_Y(\boldsymbol{v})$. So (10) and (11) are in fact equivalent.

We solve the convex program (10) approximately using the ellipsoid algorithm [17]. Following [15], the ellipsoid algorithm can solve a convex programming problem given (a) a separation oracle describing the convex space, (b) a point $\boldsymbol{v}$ inside the convex space, (c) radii δ and R such that the ball of radius δ around $\boldsymbol{v}$ is inside the convex body, and the ball of radius R contains the convex body. The running time is polynomial in the dimension of the space and in $\log\frac{R}{\delta}$.

The fact that (10) describes a convex program follows from the fact that it is equivalent to the semidefinite program (11). A separation oracle can also be obtained due to this equivalence. For the radius R, note that all feasible points $\boldsymbol{x}$ satisfy $\boldsymbol{x}_Y \cdot q_Y(\boldsymbol{x}) + c_Y \le \boldsymbol{x}_Y$ for all $Y \in \Gamma$, implying $q_Y(\boldsymbol{x}) \le 1$. Also note that for every $Z \in \Gamma$ there is a $Y \in \Gamma$ such that $q_Y(\boldsymbol{x})$ depends on $\boldsymbol{x}_Z$, so for all $Z \in \Gamma$ and all feasible vectors $\boldsymbol{v}$ we have $\boldsymbol{v}_Z \le 1/a_{min}$ where $a_{min} \ge 2^{-b}$ denotes the smallest nonzero coefficient of the task system. So we can choose $R := |\Gamma| \cdot 2^b$.

It remains to describe a feasible vector $\boldsymbol{v} \ge \boldsymbol{1}$ and a $\delta \ge 2^{-\mathrm{poly}(|\Gamma|,b)}$ such that every point $\boldsymbol{x}$ with $\|\boldsymbol{x} - \boldsymbol{v}\|_\infty \le \delta$ is feasible. (Note that d poses no problem: it can be chosen as $\frac{1}{2}$.) For that we use the vector $\boldsymbol{u}$ from Lemma C.3, i.e., $\boldsymbol{u}$ satisfies $\boldsymbol{u} = \boldsymbol{f}'(\boldsymbol{1})\boldsymbol{u} + \boldsymbol{1}$. By a computation that is similar to the one in the proof of Lemma C.3 we have $\boldsymbol{1} + r\boldsymbol{u} - \boldsymbol{f}(\boldsymbol{1} + r\boldsymbol{u}) = r(\boldsymbol{1} - rQ(\boldsymbol{u}, \boldsymbol{u}))$. So we have for all $Y \in \Gamma$:

$$1 + r\boldsymbol{u}_Y - \boldsymbol{f}_Y(\boldsymbol{1} + r\boldsymbol{u}) = r(1 - r\boldsymbol{u}_Y q_Y(\boldsymbol{u}))$$

Letting $\boldsymbol{u}_{max}$ denote the largest component of $\boldsymbol{u}$ we have $q_Y(\boldsymbol{u}) \le \boldsymbol{u}_{max}$ and consequently:

$$\ge r(1 - r\boldsymbol{u}_{max}^2)$$

By restricting r to $1/(4\boldsymbol{u}_{max}^2) \le r \le 1/(2\boldsymbol{u}_{max}^2)$ we have $r\boldsymbol{u}_{max}^2 \le 1/2$ and so:

$$\ge r/2 \ge \frac{1}{8\boldsymbol{u}_{max}^2}$$

By setting $\delta := 1/(16\boldsymbol{u}_{max}^2)$:

$$= 2\delta$$

Summarizing we have $\boldsymbol{1} + r\boldsymbol{u} - \boldsymbol{f}(\boldsymbol{1} + r\boldsymbol{u}) \ge 2\delta\boldsymbol{1}$.

Let $\boldsymbol{x}$ be any vector with $\boldsymbol{1} + r\boldsymbol{u} \ge \boldsymbol{x} \ge \boldsymbol{1} + r\boldsymbol{u} - 2\delta\boldsymbol{1}$. Then we have:

$$\boldsymbol{x} - \boldsymbol{f}(\boldsymbol{x}) \ge \boldsymbol{1} + r\boldsymbol{u} - 2\delta\boldsymbol{1} - \boldsymbol{f}(\boldsymbol{x}) \qquad \text{(as } \boldsymbol{x} \ge \boldsymbol{1} + r\boldsymbol{u} - 2\delta\boldsymbol{1}\text{)}$$

$$\geq \mathbf{1} + r\boldsymbol{u} - 2\delta\mathbf{1} - \boldsymbol{f}(\mathbf{1} + r\boldsymbol{u}) \qquad \text{(as } \boldsymbol{x} \leq \mathbf{1} + r\boldsymbol{u}\text{)}$$
$$\geq 2\delta\mathbf{1} - 2\delta\mathbf{1} = \mathbf{0} \qquad \text{(by the computation above)}$$

So if we set $\boldsymbol{v} := \mathbf{1} + r\boldsymbol{u} - \delta\mathbf{1}$, then all $\boldsymbol{x}$ with $\|\boldsymbol{x} - \boldsymbol{v}\|_\infty \leq \delta$ are feasible. Furthermore, $\delta = 1/(16\boldsymbol{u}_{max}^2) \geq 2^{-\mathrm{poly}(|\Gamma|,b)}$ because $\boldsymbol{u}$ is the solution of the linear equation system $\boldsymbol{x} = \boldsymbol{f}'(\mathbf{1})\boldsymbol{x} + \mathbf{1}$. This completes the proof. $\square$