Information and Computation 210 (2012) 87–110

Contents lists available at SciVerse ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Space-efficient scheduling of stochastically generated tasks ☆

Tomáš Brázdil [a,1], Javier Esparza [b,*], Stefan Kiefer [c,2], Michael Luttenberger [b]

[a] Faculty of Informatics, Masaryk University, Brno, Czech Republic
[b] Institut für Informatik, Technische Universität München, Germany
[c] Department of Computer Science, University of Oxford, UK

ARTICLE INFO

Article history:
Received 2 November 2010
Revised 10 June 2011
Available online 18 October 2011

Keywords:
Stochastic models
Space-efficient scheduling
Multithreaded programs
Branching processes

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 other tasks. 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 investigate the expected value $\mathbb{E}[S^\sigma]$.

1. Introduction

We study the problem of scheduling tasks where every task can stochastically generate a set of new subtasks. Tasks can be of different types, and each type has a fixed, known probability of generating new subtasks.

Systems of tasks can be described using a notation similar to that of stochastic context-free grammars. For instance

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

describes a system with two types of tasks. Tasks of type X can generate two 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. Readers familiar with process algebra will identify this notation as a probabilistic version of Basic Parallel Processes [1–3].

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, and the next task to be processed is selected by a *scheduler*. We study 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 one 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

☆ A preliminary version of this work appeared at the 37th International Colloquium on Automata, Languages and Programming, ICALP 2010.

* Corresponding author.
E-mail addresses: xbrazdil@fi.muni.cz (T. Brázdil), esparza@in.tum.de (J. Esparza), stefan.kiefer@cs.ox.ac.uk (S. Kiefer), luttenbe@model.in.tum.de (M. Luttenberger).

[1] Supported by Czech Science Foundation, grant No. P202/10/1469.
[2] Supported by the EPSRC project *Automated Verification of Probabilistic Programs* and by a postdoctoral fellowship of the German Academic Exchange Service (DAAD).

0890-5401/$ – see front matter
doi:10.1016/j.ic.2011.10.005

the pool during the computation. The completion time has been studied in [4], 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. [5]). In the more general context of multithreaded computations, a task models a thread, which may generate new threads. The problem of scheduling multithreaded computations space-efficiently on *multi*processor machines has been extensively studied (see e.g. [6–9]). These papers assume that schedulers know nothing about the program, while we consider the case in which stochastic information on the program behaviour is available (obtained from sampling). We restrict ourselves to the case in which a task has at most two children, i.e., all rules $X \xrightarrow{p} \langle X_1, \ldots, X_n \rangle$ satisfy $n \leqslant 2$. This case already allows to model the forking-mechanism underlying many multithreaded operating systems, e.g. Unix-like systems.

We study the performance of *online* schedulers, which know only the past of the computation, and compare them with the *optimal offline* scheduler, which has complete information about the future. 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 (in the context of search problems or multithreaded computations, a termination probability different from 1 usually indicates the presence of an error). These can be further divided into those with finite and infinite expected completion time, often called *subcritical* and *critical*. 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 $f_Y(x, y) = 0.7x + 0.3y$, and the reader can easily guess the formal definition. The completion probability is the least fixed point of the system of pgfs [10].

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 very 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 $\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{x} = \boldsymbol{0}$ with seed $\boldsymbol{0}$). This connection allows us to apply recent results on the convergence speed of Newton's method [11,12], leading to tail bounds of S^{op}, i.e., bounds on $\Pr[S^{op} \geqslant k]$. We then study (Section 4) the distribution of S^{σ} for an online scheduler σ, and obtain upper and lower bounds for the performance of *any* online scheduler in subcritical systems. The proof of this result suggests two ways of improving the bounds for special classes of task systems, and special classes of schedulers. We study *continuing* task systems, which are particularly natural in the context of multithreaded computation and queueing theory, and *light-first* schedulers, in which "light" tasks (loosely speaking, tasks whose progeny becomes extinguished in a short time) are chosen before "heavy" tasks, and obtain improved tail bounds.

Related work. Space-efficient scheduling for search problems or multithreaded computations has been studied in [5–9]. These papers assume that nothing is known about the program generating the computations. We study the case in which statistical information is available on the probability that computations split or die.

The theory of *branching processes* studies stochastic processes modeling populations whose members can reproduce or die [10,13]. In computer science terminology, all existing work on branching processes assumes that the number of processors is *unbounded* [14–19]. We study the 1-processor case, and to our knowledge we are the first to do so. The authors of [7] study, in a non-probabilistic setting, so-called *strict computations*, in which a task can only terminate after all the tasks it has (recursively) spawned have terminated. The optimal scheduler in this case is the *depth-first* scheduler, i.e., the one that completely executes the child task before its parent, resulting in the familiar stack-based execution. Under this scheduler our tasks are equivalent to special classes of recursive state machines [20] and probabilistic pushdown automata [21]. Recent results [22] can be used to analyze the completion space for such systems.

Last but not least, our results are strongly related to the area of probabilistic verification [23,24]. They provide techniques and fast algorithms for the verification of properties of the form: the available memory (for storing tasks) suffices to carry out the computation with probability at least p (for some given bound p).

Structure of the paper. The rest of the paper is structured as follows. The preliminaries in Section 2 formalize the notions from the introduction and summarize known results on which we build. In Section 3 we study the performance of optimal offline schedulers. Section 4 is dedicated to online schedulers. First we prove performance bounds that hold uniformly for all online schedulers, then we provide improved upper bounds for certain task systems, and then for certain schedulers. In Section 5 we obtain several results on the expected space consumption under different schedulers. Section 6 contains conclusions.

Proofs. The main body of the paper provides proof sketches of a number of theorems. Detailed proofs of all theorems can be found in Appendices A–C.

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 $\boldsymbol{0}$ (resp. $\boldsymbol{1}$). We use angular brackets to denote multisets and often

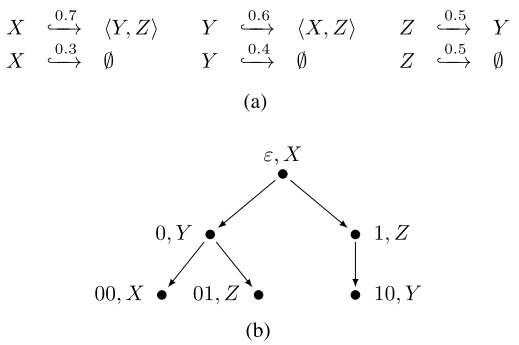

Fig. 1. (a) A task system. (b) A family tree.

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. The set of multisets over A containing at most 2 elements is denoted by $M_A^{\leqslant 2}$.

Definition 1. A *task system* is a tuple $\Delta = (\Gamma, \hookrightarrow, \mathit{Prob}, X_0)$ where Γ is a finite set of *task types*, $\hookrightarrow \subseteq \Gamma \times M_\Gamma^{\leqslant 2}$ 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*.

Fig. 1(a) shows a task system with $\Gamma = \{X, Y, Z\}$. 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. Intuitively, a family tree is a tree whose nodes are tasks; a node is labeled with the type of its task. The initial task is the root, and the children of a task are the tasks generated by it, sorted according to some fixed total order on the task types. Formally, 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 $L(w) \hookrightarrow \emptyset$, 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 $L(w0) \preccurlyeq L(w1)$, where $\prec$ is an arbitrary total order on Γ. 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.) Fig. 1 shows on the bottom a family tree of the task system on the top of the figure.

We define a function Pr which, loosely speaking, assigns to a family tree $t = (N, L)$ its probability (see *Assumptions* below). Assume that the root of t is labeled by X. If t consists only of the root, and $X \overset{p}{\hookrightarrow} \emptyset$, 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.

For every task system Δ, we define its *probability generating function* (*pgf*) as the function $\boldsymbol{f} : \mathbb{R}^\Gamma \to \mathbb{R}^\Gamma$ where for every $X \in \Gamma$

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

Example 1. Fig. 1 shows (a) a task system with $\Gamma = \{X, Y, Z\}$; and (b) a family tree t of the system with probability $\Pr[t] = 0.7 \cdot 0.6 \cdot 0.3 \cdot 0.5 \cdot 0.5 \cdot 0.4$. The name and label of a node are written next to it. The pgf is given by

$$\boldsymbol{f}(X, Y, Z) = \begin{pmatrix} \boldsymbol{f}_X(X, Y, Z) \\ \boldsymbol{f}_Y(X, Y, Z) \\ \boldsymbol{f}_Z(X, Y, Z) \end{pmatrix} = \begin{pmatrix} 0.7 \cdot Y \cdot Z + 0.3 \\ 0.6 \cdot X \cdot Z + 0.4 \\ 0.5 \cdot Y + 0.5 \end{pmatrix}$$

with task types used as corresponding variables on $\mathbb{R}$.

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) $\Pr[\mathcal{T}_X] = \sum_{t \in \mathcal{T}_X} \Pr[t] = 1$. So 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 1. *It can be decided in polynomial time whether assumptions* (1) *and* (2) *are satisfied.*

Proof. (1) is trivial. It is well known (see e.g. [10]) that (2) holds if and only if the least nonnegative fixed point of $\boldsymbol{f}$ equals $\mathbf{1}$, which is decidable in polynomial time [20,25]. □

Derivations and schedulers. Let $t = (N, L)$ be a family tree. A *state* of t is a 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 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 $s_1 \Rightarrow \cdots \Rightarrow s_k$ of states such that $s_1 = \{\epsilon\}$ and $s_k = \emptyset$. A *scheduler* is a mapping σ that assigns to a family tree t a derivation $\sigma(t)$ of t. If $\sigma(t) = (s_1 \Rightarrow \cdots \Rightarrow s_k)$, then for every $1 \leqslant i < k$ 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. This definition allows for schedulers that know the tree, and so how future tasks will behave. In Section 4 we define and study online schedulers which only know the past of the computation. Notice that schedulers are deterministic (non-randomized).

Example 2. A scheduler may schedule the tree t in Fig. 1 as follows: $\{\varepsilon\} \Rightarrow \{0, 1\} \Rightarrow \{0, 10\} \Rightarrow \{0\} \Rightarrow \{00, 01\} \Rightarrow \{01\} \Rightarrow \{\}$. The scheduler which always picks the least unprocessed task w.r.t. the lexicographical order on $\{0, 1\}^*$ (an online scheduler), schedules t differently, 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*, which assigns to a tree $t \in \mathcal{T}_X$ its number of nodes. Assuming that tasks are executed for one time unit before its generated subtasks are returned to the pool, T_X corresponds to the time required to completely execute X. Our assumption (2) guarantees that T_X is finite with probability 1, but its 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 [4]. The notion of criticality comes from the theory of branching processes, see e.g. [10,13]. Here we only recall the following results:

Proposition 2. *(See [10,20].) 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.*

The proposition essentially follows from statements in [10,20], but we provide an explicit proof, in order to make the paper more self-contained.

Proof. One can show (see e.g. [21]) 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. [26]) 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 $\rho(\boldsymbol{f}'(\mathbf{1})) \geqslant 1$. On the other hand, in [12,20] it is shown that $\rho(\boldsymbol{f}'(\mathbf{1})) \leqslant 1$. (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, $\rho(\boldsymbol{f}'(\boldsymbol{y})) \leqslant 1$ also holds for the least fixed point of $\boldsymbol{f}$ which is $\mathbf{1}$ according to the proof of Proposition 1.) 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 $\geqslant 1$. This condition holds iff $\boldsymbol{f}'(\mathbf{1})\boldsymbol{x} \geqslant \boldsymbol{x}$ holds for a nonnegative, nonzero vector $\boldsymbol{x}$ (see e.g. Theorem 2.1.11 of [27] and cf. [20]). This can be checked in polynomial time with linear programming. □

Example 3. For the task system of Fig. 1, the spectral radius of $\boldsymbol{f}'(\mathbf{1})$ is ≈ 0.97. Hence it is subcritical.

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 *completion space* S_X^σ as follows. If $\sigma(t) = (s_1 \Rightarrow \cdots \Rightarrow s_k)$, then $S_X^\sigma(t) := \max\{|s_1|, \ldots, |s_k|\}$, where $|s_i|$ is the cardinality of s_i. Sometimes we write $S^\sigma(t)$, meaning $S_X^\sigma(t)$ for the type X labelling the root of t. If we write S^σ without specifying the application to any tree, then we mean $S_{X_0}^\sigma$.

Example 4. The schedulers of Example 2 have completion space 2 and 3, respectively.

Running examples. Throughout the paper we use two task systems as running examples. The first one is the task system of Fig. 1, which we analyze numerically. The second one is actually a family of task systems: the family $X \xrightarrow{p} \langle X, X \rangle$, $X \xrightarrow{q} \emptyset$ with pgf $f(x) = px^2 + q$, where $0 < p \leqslant 1/2$ is a parameter and $q = 1 - p$. This family is very simple, which allows to interpret our results analytically. Notice that the probability p satisfies $p \leqslant 1/2$: For $1/2 < p \leqslant 1$ the least fixed point of f

is $q/p < 1$, and so the system does not terminate with probability 1. For $p \leqslant 1/2$ the least fixed point of f is 1, and the system is critical for $p = 1/2$, and subcritical for $p < 1/2$.

3. Optimal (offline) schedulers

Let S^{op} be the random variable that assigns to a family tree the minimal completion space of its derivations. We call $S^{op}(t)$ the *optimal completion space* of t. The optimal scheduler assigns to each tree a derivation with optimal completion space. 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. Note that, although the optimal scheduler "knows" how the stochastic choices are resolved, the optimal completion space $S^{op}(t)$ is still a random variable, because it depends on a random tree. The following proposition characterizes the optimal completion space of a tree in terms of the optimal completion space of its children.

Proposition 3. *Let t be a family tree. Then*

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

Proof. 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_{1-i}, 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 $s(i) = \max\{S^{op}(t_i)+1, S^{op}(t_{1-i})\}$. Since the optimal scheduler can choose the value of i that minimizes $s(i)$, we have $S^{op}(t) \leqslant \min\{s(0), s(1)\}$.

It remains to argue why the scheduler cannot save space by interleaving the schedulings for t_0 and t_1. Consider an optimal scheduling of t. Assume that the derivation of the t_0-tree is completed first. Then at least one task from t_1 terminates only after the derivation of t_0 is completed, so this scheduling needs space of at least $S^{op}(t_0)+1$. Since any scheduling of t needs space of at least $S^{op}(t_1)$, we have $S^{op}(t) \geqslant s(0)$. Assume now that the derivation of the t_1-tree is completed first. Then, similarly, we have $S^{op}(t) \geqslant s(1)$. So, we obtain $S^{op}(t) \geqslant \min\{s(0), s(1)\}$, and, with the inequality above, $S^{op}(t) = \min\{s(0), s(1)\}$. □

Given a type X, we are interested in the probabilities $\Pr[S^{op}_X \leqslant k]$ for $k \geqslant 1$. Proposition 3 yields a recurrence relation which at first sight seems difficult to handle. However, using results of [28,29] 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 the proof of Proposition 1 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 [20,11]. More precisely, $\boldsymbol{\mu} = \lim_{k\to\infty} \boldsymbol{\nu}^{(k)}$ where

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

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. Surprisingly, the sequence of approximants computed by Newton's method provides exactly the information we are looking for:

Theorem 1. $\Pr[S^{op}_X \leqslant k] = \boldsymbol{\nu}^{(k)}_X$ *for every type X and every $k \geqslant 0$.*

Proof sketch. We illustrate the proof idea on the one-type task system with the pgf $f(x) = px^2 + q$, where $q = 1-p$. Notice that the "vectors" f and $\nu^{(k)}$ and the "matrix" f' have a single entry only, so they can be treated as numbers. Let $\mathcal{T}_{\leqslant k}$ and $\mathcal{T}_{=k}$ denote the sets of trees t with $S^{op}(t) \leqslant k$ and $S^{op}(t) = k$, respectively. We show $\Pr[\mathcal{T}_{\leqslant k}] = \nu^{(k)}$ for all k by induction on k. The case $k = 0$ is trivial. Assume that $\nu^{(k)} = \Pr[\mathcal{T}_{\leqslant k}]$ holds for some $k \geqslant 0$. We prove $\Pr[\mathcal{T}_{\leqslant k+1}] = \nu^{(k+1)}$. Notice that

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

Let $\mathcal{B}^0_{k+1}$ be the set of trees that have two children both of which belong to $\mathcal{T}_{=k}$, and, for every $i \geqslant 0$, let $\mathcal{B}^{i+1}_{k+1}$ be the set of trees with two children, one belonging to $\mathcal{T}_{\leqslant k}$, the other one to $\mathcal{B}^i_{k+1}$. By Proposition 3 we have $\mathcal{T}_{\leqslant k+1} = \bigcup_{i\geqslant 0} \mathcal{B}^i_{k+1}$. We prove $\Pr[\mathcal{B}^i_{k+1}] = f'(\nu^{(k)})^i \ (f(\nu^{(k)}) - \nu^{(k)})$ by an (inner) induction on i, which completes the proof. For the base $i = 0$, let $\mathcal{A}_{\leqslant k}$ be the set of trees with two children in $\mathcal{T}_{\leqslant k}$; by induction hypothesis we have $\Pr[\mathcal{A}_{\leqslant k}] = p\nu^{(k)}\nu^{(k)}$. In a tree of $\mathcal{A}_{\leqslant k}$ either (a) both children belong to $\mathcal{T}_{=k}$, and so $t \in \mathcal{B}^0_{k+1}$, or (b) at most one child belongs to $\mathcal{T}_{=k}$. By Proposition 3, the trees satisfying (b) belong to $\mathcal{T}_{\leqslant k}$. In fact, a tree of $\mathcal{T}_{\leqslant k}$ either satisfies (b) or it has one single node. Since the probability

of the tree with one node is q, we get $\Pr[\mathcal{A}_{\leqslant k}] = \Pr[\mathcal{B}^0_{k+1}] + \Pr[\mathcal{T}_{\leqslant k}] - q$. Applying the induction hypothesis again we obtain $\Pr[\mathcal{B}^0_{k+1}] = p\nu^{(k)}\nu^{(k)} + q - \nu^{(k)} = f(\nu^{(k)}) - \nu^{(k)}$. For the induction step, let $i \geqslant 0$. Divide $\mathcal{B}^{i+1}_{k+1}$ into two sets, one containing the trees whose left (right) child belongs to $\mathcal{B}^i_{k+1}$ (to $\mathcal{T}_{\leqslant k}$), and the other the trees whose left (right) child belongs to $\mathcal{T}_{\leqslant k}$ (to $\mathcal{B}^i_{k+1}$). 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 $\Pr[\mathcal{B}^{i+1}_{k+1}] = (2p\nu^{(k)}) \cdot f'(\nu^{(k)})^i (f(\nu^{(k)}) - \nu^{(k)})$. Since $f(x) = px^2 + q$ we have $f'(\nu^{(k)}) = 2p\nu^{(k)}$, and so $\Pr[\mathcal{B}^{i+1}_{k+1}] = f'(\nu^{(k)})^{i+1}(f(\nu^{(k)}) - \nu^{(k)})$ as desired. □

Example 5. Computing Newton approximants for the task system from Fig. 1 and applying Theorem 1 yields

$$\Pr\big[S^{op}_X \leqslant 0\big] = 0 \qquad \Pr\big[S^{op}_X \leqslant 3\big] \approx 0.88$$
$$\Pr\big[S^{op}_X \leqslant 1\big] = 0.3 \qquad \Pr\big[S^{op}_X \leqslant 4\big] \approx 0.96$$
$$\Pr\big[S^{op}_X \leqslant 2\big] \approx 0.70 \qquad \Pr\big[S^{op}_X \leqslant 5\big] \approx 0.99.$$

Example 6. 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 $0 < p \leqslant 1/2$ and $q = 1 - p$. Using Newton approximants we obtain the recurrence relation

$$\Pr\big[S^{op} \geqslant k+1\big] = \frac{p \cdot \Pr[S^{op} \geqslant k]^2}{1 - 2p + 2p \cdot \Pr[S^{op} \geqslant k]}$$

for the distribution of the optimal scheduler. In particular, for the critical value $p = 1/2$ we get $\Pr[S^{op} \geqslant k = 2^{1-k}$.

Theorem 1 allows to compute the probability mass function of S^{op}. As a Newton iteration requires $\mathcal{O}(|\Gamma|^3)$ arithmetical operations for a task system with set of types Γ, we obtain the following corollary, where by the unit cost model we refer to the model in which arithmetic operations have cost 1, independently of the size of the operands [30].

Corollary 1. *$\Pr[S^{op}_X = k]$ 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. [31]), meaning, roughly speaking, that each iteration doubles the number of accurate bits of $\Pr[S^{op}_X \geqslant k]$ computed so far. For critical systems, it has recently been proved that Newton's method still converges linearly [11,12]. These results lead to tail bounds for S^{op}_X:

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

Proof. By Theorem 1 we have $\Pr[S^{op} \geqslant k] = 1 - \boldsymbol{\nu}^{(k-1)}_{X_0}$. 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 [20].

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, contradict the assumption that the task system is subcritical. For nonsingular systems, it is a standard fact (see e.g. [31]) that Newton's method converges quadratically. As $\Pr[S^{op} \geqslant k] \leqslant 1 - \boldsymbol{\nu}^{(k-1)}_{X_0}$, 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 [11, 12] and it follows from there that there is a $c_1 \in (0, \infty)$ such that $1 - \boldsymbol{\nu}^{(k)}_X \leqslant c_1 \cdot 2^{-k/(n2^n)}$ where $n = |\Gamma|$, implying the statement. □

4. Online schedulers

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

For our results in this section it is convenient to assume that the task system is in a certain normal form, which we call *compact*.

Compact task systems. Any task system can be transformed into a so-called *compact* task system such that for every scheduler of the compact task system we can construct a scheduler of the original system yielding the same completion space up to an increase of at most $|\Gamma|$. 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 iteratively removing all rules with non-compact types on the left-hand side, and all occurrences of non-compact types on the right-hand side.

Example 7. Consider the following task system:

$$X \xrightarrow{0.5} \langle X, Y\rangle \qquad Y \xrightarrow{0.5} \langle Y, Z\rangle \qquad Z \xrightarrow{0.5} Z$$
$$X \xrightarrow{0.5} \langle X, X\rangle \qquad Y \xrightarrow{0.5} Z \qquad Z \xrightarrow{0.5} \emptyset.$$

Here, only type Z is not compact. After removing it, we obtain the new system:

$$X \xrightarrow{0.5} \langle X, Y\rangle \qquad Y \xrightarrow{0.5} Y$$
$$X \xrightarrow{0.5} \langle X, X\rangle \qquad Y \xrightarrow{0.5} \emptyset.$$

In the reduced system, type Y is not compact anymore, which necessitates a second iteration:

$$X \xrightarrow{0.5} X$$
$$X \xrightarrow{0.5} \langle X, X\rangle.$$

Note that if a scheduler schedules Z-tasks before Y-tasks and Y-tasks before X-tasks, then the pool has, at any time, at most two non-X-tasks.

The following proposition allows us to concentrate on compact task systems.

Proposition 4. *Let Γ' denote 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 $S^{\sigma} \leqslant \ell$. Assume that $X_0 \notin \Gamma'$. Let Δ' be the compacted version of Δ. Every scheduler σ' for Δ' can be transformed into a scheduler σ for Δ such that for all k*

$$\Pr[S^{\sigma',\Delta'} \geqslant k] \leqslant \Pr[S^{\sigma,\Delta} \geqslant k] \leqslant \Pr[S^{\sigma',\Delta'} \geqslant k - \ell].$$

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

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. Now we prove Proposition 4.

Proof. Let Δ_1 be a non-compact task system with 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 completion space of the derivation is 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 completion space. 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 completion space of a derivation by at most 1.

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

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

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 $S_X^{\sigma,\Delta} \leqslant \ell$ for all $X \in \Gamma'$.

It remains to show the inequality $\Pr[S_X^{\sigma',\Delta'} \geqslant k] \leqslant \Pr[S_X^{\sigma,\Delta} \geqslant k]$; but this follows directly from the fact that Δ' is obtained by deleting rules and types from Δ and σ is obtained by extending σ'. □

Assumption. In light of Proposition 4 we assume for the rest of the section that task systems are compact.

The following main theorem gives lower and upper bound on $\Pr[S^{\sigma} \geqslant k]$ which hold uniformly for all online schedulers σ. The bounds are given in terms of vectors $\boldsymbol{v}, \boldsymbol{w}$ with $\boldsymbol{f}(\boldsymbol{v}) \leqslant \boldsymbol{v}$ and $\boldsymbol{f}(\boldsymbol{w}) \geqslant \boldsymbol{w}$. We will show later (Proposition 5) how to compute such vectors.

Theorem 2. *Let Δ be subcritical. Let $\boldsymbol{v}, \boldsymbol{w} \in (1,\infty)^{\Gamma}$ be vectors with $\boldsymbol{f}(\boldsymbol{v}) \leqslant \boldsymbol{v}$ and $\boldsymbol{f}(\boldsymbol{w}) \geqslant \boldsymbol{w}$. Denote by $\boldsymbol{v}_{min}$ and $\boldsymbol{w}_{max}$ the least component of $\boldsymbol{v}$ and the greatest component of $\boldsymbol{w}$, respectively. Then*

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

Proof sketch. The proof adapts a technique for the analysis of random walks going back to Feller (see [32,33]), which uses the vector $\boldsymbol{v}$ to derive a supermartingale (and proceeds analogously with $\boldsymbol{w}$). More precisely, we use $\boldsymbol{v}$ to assign a *weight* $m^{(i)}$ to the pool of tasks at time i and to choose a constant h, so that the following property holds: at any stopping time τ of the process, the expected value of $h^{m^{(\tau)}}$ is at most $\boldsymbol{v}_{X_0}$. This provides a "stochastic invariant" of the process, from which we can extract an upper bound for $\Pr[S^{\sigma} \geqslant k]$.

We now give some more details. For every $i \geqslant 1$, let $\boldsymbol{z}^{(i)}$ be the vector of random variables that measures the number of tasks of each type at time i. We choose $h > 1$ and $\boldsymbol{u} \in (0,\infty)^{\Gamma}$ such that $h^{\boldsymbol{u}_X} = \boldsymbol{v}_X$ for all $X \in \Gamma$. Now, let $m^{(i)} = \boldsymbol{z}^{(i)} \bullet \boldsymbol{u} \stackrel{def.}{=} \sum_{X\in\Gamma} \boldsymbol{z}_X^{(i)} \boldsymbol{u}_X$, i.e., $m^{(i)}$ is the random variable that measures the weight of the tasks at time i, when the weight of a task of type X is given by $\boldsymbol{u}_X$. One can show that $h^{m^{(1)}}, h^{m^{(2)}}, \ldots$ is a supermartingale for any online scheduler σ. Define a stopping time $\tau_k := \inf\{i \geqslant 1 \mid m^{(i)} \in \{0\} \cup [k,\infty)\}$; i.e., τ_k is the time when the task system either terminates or has at least k tasks in the pool. Using the Optional–Stopping Theorem [34], we obtain

$$\mathbb{E}\big[h^{m^{(\tau_k)}}\big] \leqslant h^{\boldsymbol{u}_{X_0}} = \boldsymbol{v}_{X_0}.$$

Set $p_k := \Pr[m^{(\tau_k)} \geqslant k]$. Then we have

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

which gives

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

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

$$\Pr\big[S^{\sigma} \geqslant k\big] = \Pr\Big[\sup_i \big|\boldsymbol{z}^{(i)}\big| \geqslant k\Big] \leqslant \Pr\Big[\sup_i m^{(i)} \geqslant k\boldsymbol{u}_{min}\Big] = p_{k\boldsymbol{u}_{min}} \leqslant \frac{\boldsymbol{v}_{X_0}-1}{\boldsymbol{v}_{min}^{k}-1}.$$

The lower bound is shown similarly. □

All online schedulers perform within the bounds of Theorem 2. For an application of the upper bound, assume one wants to provide as much space as is necessary to guarantee that, say, 99.9% of the executions of a task system can run without needing additional memory. This can be accomplished, regardless of the scheduler, by providing k space units, where k is chosen such that the upper bound of Theorem 2 is at most 0.001.

A comparison of the lower bound with Corollary 2 proves that for subcritical task systems the asymptotic performance of any online scheduler σ is far away from that of the optimal offline scheduler: the ratio $\Pr[S^{\sigma} \geqslant k]/\Pr[S^{op} \geqslant k]$ is unbounded.

Example 8. Consider again the task system from Fig. 1. Its pgf has two fixed points, $(1,1,1)^{\top}$ and $\boldsymbol{v} \approx (1.142, 1.130, 1.065)^{\top}$, so $\boldsymbol{v}$ can be used to obtain both an upper and a lower bound for online schedulers. Theorem 2 yields for all online schedulers σ:

$$\begin{array}{lllll} 0.29 \leqslant \Pr[S^{\sigma} \geqslant 1] \leqslant 2.18 & \qquad & 0.11 \leqslant \Pr[S^{\sigma} \geqslant 4] \leqslant 0.50 \\ 0.20 \leqslant \Pr[S^{\sigma} \geqslant 2] \leqslant 1.06 & & 0.09 \leqslant \Pr[S^{\sigma} \geqslant 5] \leqslant 0.39 \\ 0.15 \leqslant \Pr[S^{\sigma} \geqslant 3] \leqslant 0.69 & & 0.07 \leqslant \Pr[S^{\sigma} \geqslant 6] \leqslant 0.31. \end{array}$$

Example 9. Consider again the one-type 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 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, an inspection of the proof shows that in this particular case the exponent $k+2$ of the lower bound can be improved to k (basically a consequence of the fact that if the completion space is at least k, then the derivation must encounter a state with exactly k tasks). So the upper and lower bounds coincide, and we get

$$\Pr[S^{\sigma} \geqslant k] = \frac{q/p-1}{(q/p)^k-1}$$

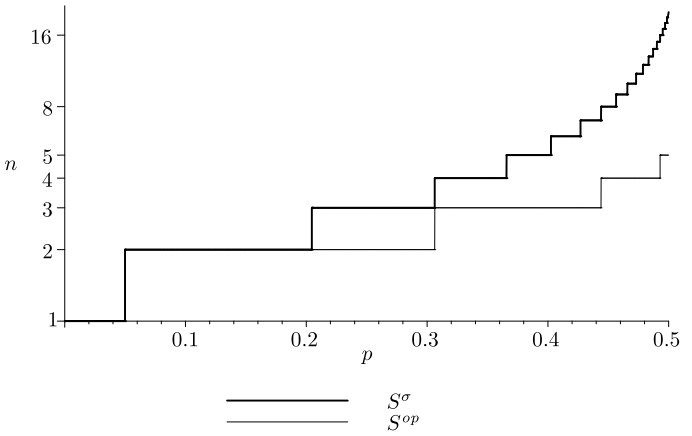

Fig. 2. The least number n such that $\Pr[S \leqslant n] \geqslant 0.95$, where $S \in \{S^{op}, S^{\sigma}\}$ is the completion space of the optimal and the online schedulers. Notice the logarithmic scale.

for every online scheduler σ. In particular, as one intuitively expects, all online schedulers are equivalent.[3] Fig. 2 plots the function that assigns to every $p \in [0, 0.5]$ the 95% quantile of the variables S^{op} and S^{σ}, where σ is an arbitrary online scheduler. Loosely speaking, this is the least number of tasks one must be able to store in order to have 95% confidence that the execution of the task system will be successful.

Theorem 2 assumes the existence of vectors $\boldsymbol{v}, \boldsymbol{w} \in (1, \infty)^{\Gamma}$ with $\boldsymbol{f}(\boldsymbol{v}) \leqslant \boldsymbol{v}$ and $\boldsymbol{f}(\boldsymbol{w}) \geqslant \boldsymbol{w}$. The following proposition provides such vectors, and shows that they can be computed in polynomial time.

Proposition 5. *We have*

$$\boldsymbol{f}\left(\mathbf{1} + \frac{\boldsymbol{u}}{q_{max}}\right) \leqslant \mathbf{1} + \frac{\boldsymbol{u}}{q_{max}}$$

where $\boldsymbol{u} \in [1, \infty)^{\Gamma}$ is the vector of expected completion times, i.e., $\boldsymbol{u}_Y = \mathbb{E}[T_Y]$ for all $Y \in \Gamma$, and $q_{max} = \max_{X \in \Gamma} \sum_{X \xhookrightarrow{p} YZ} p \cdot \boldsymbol{u}_Y \cdot \boldsymbol{u}_Z$.

Moreover, we have

$$\boldsymbol{f}\left(\mathbf{1} + \frac{\boldsymbol{x}}{c \cdot \boldsymbol{x}_{min}^2}\right) \geqslant -\mathbf{1} + \frac{\boldsymbol{x}}{c \cdot \boldsymbol{x}_{min}^2}$$

where c is the smallest nonzero coefficient of $\boldsymbol{f}$, and $\boldsymbol{x} := (I - \boldsymbol{f}'(\mathbf{1}))^{-1}\mathbf{y}$, and $\boldsymbol{x}_{min} > 0$ is the least component of $\boldsymbol{x}$, and $\mathbf{y} \in \{0, 1\}^{\Gamma}$ is the vector such that $\mathbf{y}_X = 1$ for the components X with $X \xhookrightarrow{p} \langle Y, Z \rangle$ for some $Y, Z \in \Gamma$, and $\mathbf{y}_X = 0$ otherwise.

Theorem 2 hints at two ways to improve the upper bound on $\Pr[S^{\sigma} \geqslant k]$:

1. For particular task systems, compute a vector $\boldsymbol{v} \in (1, \infty)^{\Gamma}$ such that the upper bound of Theorem 2 is as good as possible.
2. For particular online schedulers, improve the upper bound of Theorem 2.

We follow these ways in Sections 4.1 and 4.2.

4.1. Optimizing the upper bound for continuing task systems

Theorem 2 shows that any vector $\boldsymbol{v}$ satisfying $\boldsymbol{f}(\boldsymbol{v}) \leqslant \boldsymbol{v}$ leads to an upper bound on the performance of the scheduler. For large k, the best bound is obtained by maximizing $\boldsymbol{v}_{min}$. Define

$$\boldsymbol{v}_{min}^{opt} := \sup\{\boldsymbol{v}_{min} \mid \boldsymbol{f}(\boldsymbol{v}) \leqslant \boldsymbol{v}\}.$$

Computing or even approximating $\boldsymbol{v}_{min}^{opt}$ is a difficult optimization problem, due to the nonlinearity of $\boldsymbol{f}$. We show that for the class of *continuing* systems, an ϵ-approximation of $\boldsymbol{v}_{min}^{opt}$, i.e., a number d with $|d - \boldsymbol{v}_{min}^{opt}| \leqslant \epsilon$ can be computed in polynomial time.

[3] For this example $\Pr[S^{\sigma} \geqslant k]$ can also be computed by elementary means, see, e.g., Chapter XIV of [32] on "ruin" problems.

A task system is continuing if (1) there are no rules of the form $X \xhookrightarrow{p} Y$ and (2) 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. Continuing task systems appear in two common scenarios. The first one are multithreaded programs in which a thread repeatedly executes a loop that generates a new thread, until it exits the loop and terminates. This behaviour is modeled by rules of the form $X \xhookrightarrow{p} \langle X, Y\rangle$. The second scenario is a variant of the well known $M/M/1$ queue of queueing theory (see e.g. [35]). Customers are served by one server, and both arrival and service times are exponentially distributed. However, customers can have different types, and arrival and service rates of the customers *may depend on their type*. Assume for simplicity that there are two types of customers X and Y with arrival rates a_X, a_Y and service rates s_X, s_Y. If the server is currently serving a customer of type X, then the probability that the service terminates before a new customer arrives is $s_X/(a_X + a_Y + s_X)$, and the probabilities that new customers of type X (resp. Y) arrives before the service terminates is $a_X/(a_X + a_Y + s_X)$ (resp. $a_Y/(a_X + a_Y + s_X)$). We model customers as tasks. Since the exponential distribution is memoryless, the arrival of a new customer, say of type Y, while the server is serving a customer c of type X, is equivalent to the server finishing the service of c, and the queue receiving *two* new customers of type X and Y, respectively. So the queue is modeled by the continuing task system

$$\begin{array}{ll} X \xrightarrow{s_X/(a_X+a_Y+s_X)} \emptyset & Y \xrightarrow{s_Y/(a_X+a_Y+s_Y)} \emptyset \\ X \xrightarrow{a_X/(a_X+a_Y+s_X)} \langle X, X\rangle & Y \xrightarrow{a_X/(a_X+a_Y+s_Y)} \langle X, Y\rangle \\ X \xrightarrow{a_Y/(a_X+a_Y+s_X)} \langle X, Y\rangle & Y \xrightarrow{a_Y/(a_X+a_Y+s_Y)} \langle Y, Y\rangle. \end{array}$$

Example 10. An editor of two journals, say X and Y, receives manuscripts for journal X and journal Y at a rate of $a_X = 2$ and $a_Y = 1$ per month, respectively. She spends a fixed time per month on dealing with manuscripts, provided there are manuscripts waiting; if she works exclusively on X-manuscripts (resp. Y-manuscripts), she can handle them at a rate of $s_X = 4$ (resp. $s_Y = 6$) per month. According to the discussion above, this situation is modeled by the following continuing task system.

$$\begin{array}{ll} X \xrightarrow{4/7} \emptyset & Y \xrightarrow{2/3} \emptyset \\ X \xrightarrow{2/7} \langle X, X\rangle & Y \xrightarrow{2/9} \langle X, Y\rangle \\ X \xrightarrow{1/7} \langle X, Y\rangle & Y \xrightarrow{1/9} \langle Y, Y\rangle. \end{array}$$

In particular, if the editor is working on an X-manuscript, she finishes handling it before any other manuscript arrives with probability 4/7, and she receives a new Y-manuscript before with probability 1/7. When a new manuscript arrives, the editor can choose to continue with the current one, or switch to the new one. Let $\boldsymbol{f}$ denote the pgf. The spectral radius of $\boldsymbol{f}'(\mathbf{1})$ is about 0.80; i.e., the task system is subcritical. In other words, the editor can always clear the pool of waiting manuscripts in finite expected time, no matter how she distributes her time among X- and Y-manuscripts. The function $\boldsymbol{f}$ has two nonnegative fixed points: $(1, 1)^\top$ and $\boldsymbol{v} \approx (1.56, 1.32)^\top$. Applying Theorem 2 with $\boldsymbol{v}$ yields $\boldsymbol{v}_{min} \approx 1.32$, and so, for instance, $\Pr[S_X^{\sigma'} \geqslant 10] \leqslant 0.039$. We can obtain a better bound with $\boldsymbol{v} = (1.40, 1.33)^\top$, which satisfies $\boldsymbol{f}(\boldsymbol{v}) \leqslant \boldsymbol{v}$, and leads to $\Pr[S_X^{\sigma'} \geqslant 10] \leqslant 0.025$.

Theorem 3. *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*:

$$\textit{maximize } d \textit{ such that } \forall Y \in \Gamma\colon \quad 0 \leqslant d \leqslant \boldsymbol{v}_Y \geqslant \boldsymbol{f}_Y(\boldsymbol{v}).$$

Proof sketch. The proof follows a proof of [36]. Notice that $\boldsymbol{v}_{min}^{opt} = \sup\{d \in \mathbb{R} \mid \exists \boldsymbol{v} \in [0, \infty)^\Gamma\colon\ d\mathbf{1} \leqslant \boldsymbol{v} \geqslant \boldsymbol{f}(\boldsymbol{v})\}$. As we consider continuing task systems we can, for all $Y \in \Gamma$, write $\boldsymbol{f}_Y$ as $\boldsymbol{f}_Y(\boldsymbol{x}) = \boldsymbol{x}_Y \cdot A_Y \boldsymbol{x} + c_Y$, where $A \in [0, 1]^{\Gamma \times \Gamma}$ is a matrix and A_Y denotes its row vector corresponding to Y. We claim that the following systems (1) and (2) are equivalent:

$$\text{maximize } d \text{ such that } \forall Y \in \Gamma\colon \quad 0 \leqslant d \leqslant \boldsymbol{v}_Y \geqslant \boldsymbol{f}_Y(\boldsymbol{v}) \tag{1}$$

$$\text{maximize } d \text{ such that } \forall Y \in \Gamma\colon \quad \left\{ \begin{array}{l} 0 \leqslant d \leqslant \boldsymbol{v}_Y \\ s_Y = 1 - A_Y \boldsymbol{v} \\ \begin{pmatrix} \boldsymbol{v}_Y & \sqrt{c_Y} \\ \sqrt{c_Y} & s_Y \end{pmatrix} \quad \text{positive semidefinite} \end{array} \right\} \tag{2}$$

For the equivalence of (1) and (2) note that the condition on the matrices being positive semidefinite is equivalent to $\boldsymbol{v}_Y \cdot s_Y \geqslant c_Y$. Substituting $s_Y = 1 - A_Y \boldsymbol{v}$ yields $\boldsymbol{v}_Y \cdot (1 - A_Y \boldsymbol{v}) \geqslant c_Y$ which is, using $\boldsymbol{f}_Y(\boldsymbol{v}) = \boldsymbol{v}_Y \cdot A_Y \boldsymbol{v} + c_Y$, equivalent to $\boldsymbol{v}_Y \geqslant \boldsymbol{f}_Y(\boldsymbol{v})$. So (1) and (2) are equivalent.

By this equivalence and the fact that (2) is a semidefinite program, it follows that (1) describes a convex program. Hence one can solve (1) approximately using the ellipsoid algorithm [37]. Following [36], 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 $\log \frac{R}{\delta}$. A separation oracle can be obtained from the equivalence of (1) and (2). The points (b) and (c) are dealt with in Appendix B. □

4.2. Optimizing the upper bound for light-first schedulers

We present a class of online schedulers for which sharper upper bounds than the one given by Theorem 2 can be proved. It may be intuitive that a good heuristic is to pick the task with the smallest expected completion time. If the vector $\boldsymbol{v}$ with $\boldsymbol{f}(\boldsymbol{v}) \leqslant \boldsymbol{v}$ which is needed for the upper bound of Theorem 2 is computed according to Proposition 5, then the type X_{min} for which $\boldsymbol{v}_{X_{min}} = \boldsymbol{v}_{min}$ holds is the type with the smallest expected completion time. This suggests regarding $\boldsymbol{v}$ as a vector of weights, and always choosing the lightest active type. This scheduler leads in fact to better upper bounds.

Given a vector $\boldsymbol{v}$ with $\boldsymbol{f}(\boldsymbol{v}) \leqslant \boldsymbol{v}$ we denote by $\leqslant$ a total order on Γ such that whenever $X \leqslant Y$ then $\boldsymbol{v}_X \leqslant \boldsymbol{v}_Y$. If $X \leqslant Y$, 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 below strengthens the upper bound of Theorem 2 for light-first schedulers. For the second part of Theorem 4 we use the notion of *$\boldsymbol{v}$-accumulating types*. A type $X \in \Gamma$ is $\boldsymbol{v}$-accumulating if for every $k \geqslant 0$ 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. *Let Δ be subcritical and $\boldsymbol{v} \in (1,\infty)^{\Gamma}$ with $\boldsymbol{f}(\boldsymbol{v}) \leqslant \boldsymbol{v}$. Let σ be a $\boldsymbol{v}$-light-first scheduler. Let $\boldsymbol{v}_{min\,max} := \min_{X \hookrightarrow \langle Y,Z\rangle} \max\{\boldsymbol{v}_Y, \boldsymbol{v}_Z\}$ (here the minimum is taken over all transition rules with two types on the right-hand side). Then $\boldsymbol{v}_{min\,max} \geqslant \boldsymbol{v}_{min}$ and for all $k \geqslant 1$*

$$\Pr\bigl[S^{\sigma} \geqslant k\bigr] \leqslant \frac{\boldsymbol{v}_{X_0} - 1}{\boldsymbol{v}_{min}\,\boldsymbol{v}_{min\,max}^{k-1} - 1}.$$

Moreover, let $\boldsymbol{v}_{min\,acc} := \min\{\boldsymbol{v}_X \mid X \in \Gamma,\ X \text{ is } \boldsymbol{v}\text{-accumulating}\}$. Then (i) *$\boldsymbol{v}_{min\,acc} \geqslant \boldsymbol{v}_{min\,max}$,* (ii) *$\boldsymbol{v}_{min\,acc}$ can be computed in polynomial time, and* (iii) *there is an integer ℓ such that for all $k \geqslant \ell$*

$$\Pr\bigl[S^{\sigma} \geqslant k\bigr] \leqslant \frac{\boldsymbol{v}_{X_0} - 1}{\boldsymbol{v}_{min}^{\ell}\,\boldsymbol{v}_{min\,acc}^{k-\ell} - 1}.$$

Proof sketch. For the sketch we only outline proofs of the assertion that $\boldsymbol{v}_{min\,acc}$ can be computed in polynomial time, and of the final inequality.

We prove in Appendix B the following characterization: X is $\boldsymbol{v}$-accumulating if and only if there is Y such that (1) a multiset containing Y can be derived from X, and (2) a multiset containing $\{X, Y\}$ can be derived from Y using only rules $Z \hookrightarrow \beta$ with $Z \leqslant Y$. This immediately leads to a polynomial algorithm.

For the final inequality, recall the proof sketch of Theorem 2 where we used that $S^{\sigma} \geqslant k$ implies $\sup_i m^{(i)} \geqslant k\boldsymbol{u}_{min}$, 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 $S^{\sigma} \geqslant k$ implies $\sup_i m^{(i)} \geqslant \ell\boldsymbol{u}_{min} + (k-\ell)\boldsymbol{u}_{min\,acc}$ which leads to the final inequality of Theorem 4 in a way analogous to the proof sketch of Theorem 2. □

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 $\boldsymbol{v}_{min\,acc} > \boldsymbol{v}_{min}$. This is illustrated in the following examples.

Example 11. Consider again the task system from Fig. 1. with the vector $\boldsymbol{v} \approx (1.142, 1.130, 1.065)^{\top}$ from Example 8. Hence, Z is the lightest task. For the first part of Theorem 4 we have $\boldsymbol{v}_{min\,max} = \boldsymbol{v}_Y \approx 1.130$. Therefore, for the $\boldsymbol{v}$-light-first scheduler σ, we obtain the following upper bounds: $\Pr[S^{\sigma} \geqslant 1] \leqslant 2.18$, $\Pr[S^{\sigma} \geqslant 2] \leqslant 0.70$, $\Pr[S^{\sigma} \geqslant 3] \leqslant 0.40$, $\Pr[S^{\sigma} \geqslant 4] \leqslant 0.27$, $\Pr[S^{\sigma} \geqslant 5] \leqslant 0.20$. Notice the improvement over Example 8. The upper bounds are further improved asymptotically using the second part of Theorem 4 as follows: The type X is $\boldsymbol{v}$-accumulating, but Y and Z are not. It follows $\boldsymbol{v}_{min\,acc} = \boldsymbol{v}_X \approx 1.142$. Further, it is not hard to show that, starting with a single X-symbol, there are, at any time, at most two non-X-tasks in the pool. It follows from the proof of Theorem 4 that one can then take $\ell = 2$. Consequently, we have

$$\Pr\bigl[S^{\sigma} \geqslant k\bigr] \leqslant \frac{\boldsymbol{v}_X - 1}{\boldsymbol{v}_Z^2\,\boldsymbol{v}_X^{k-2} - 1} \in \mathcal{O}\bigl(1/\boldsymbol{v}_X^k\bigr);$$

i.e., the upper bound for the $\boldsymbol{v}$-light-first scheduler coincides, up to a constant factor, with the lower bound for any online scheduler.

Example 12. Consider the task system with 2 task types 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 $\boldsymbol{v}_X = (1-a_2-b_1-a_1b_2+a_2b_1)/b_2$ and $\boldsymbol{v}_Y = (1-b_1-b_2)/(a_2+a_1b_2-a_2b_1)$. We have $\boldsymbol{v}_X \leqslant \boldsymbol{v}_Y$ iff $a_2-b_2 \leqslant a_2b_1-a_1b_2$, 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. $\boldsymbol{v}_X \leqslant \boldsymbol{v}_Y$. Then X is not accumulating (because X-tasks are picked as soon as they are created), and so $\boldsymbol{v}_{min\,acc} = \boldsymbol{v}_Y$. So the upper bound for the light-weight scheduler yields a constant c_2 such that $\Pr[S^\sigma \geqslant k] \leqslant c_2/\boldsymbol{v}_Y^k$. But the general lower bound for arbitrary online schedulers states that there is a constant c_1 such that $\Pr[S^\sigma \geqslant k] \geqslant c_1/\boldsymbol{v}_Y^k$, so we are done.

5. Expectations

We finish our study of the completion space by presenting some results about its expectation $\mathbb{E}[S^\sigma]$. In Section 5.1 we show that the expectation of the completion space of the optimal offline scheduler is always finite, and can be efficiently approximated. In Section 5.2, we obtain two results for online schedulers. The first one states that, while $\mathbb{E}[S^\sigma]$ depends on σ, whether $\mathbb{E}[S^\sigma]$ is finite or infinite does not: $\mathbb{E}[S^\sigma]$ is always finite for subcritical systems, and infinite for critical systems. The second result deals with the question of finding an online scheduler σ such that $\mathbb{E}[S^\sigma]$ is as small as possible. Loosely speaking, we show how to efficiently construct a sequence of schedulers whose expected completion spaces converge to the optimal value.

For the rest of the section we fix a task system $\Delta = (\Gamma, \hookrightarrow, Prob, X_0)$.

5.1. 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 $\mathbb{E}[R] = \sum_{i=1}^{\infty} \Pr[R \geqslant i]$. So we can (under-) approximate $\mathbb{E}[R]$ by $\sum_{i=1}^{k} \Pr[R \geqslant i]$ for finite k. We say that *k terms compute b bits of* $\mathbb{E}[S^{op}]$ if $\mathbb{E}[S^{op}] - \sum_{i=0}^{k-1}(1 - \boldsymbol{v}^{(i)}_{X_0}) \leqslant 2^{-b}$.

Theorem 5. *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 $e^{(i)} := 1 - \boldsymbol{v}^{(i)}_{X_0}$. Then we have $\mathbb{E}[S^{op}] - \sum_{i=0}^{k-1}(1 - \boldsymbol{v}^{(i)}_{X_0}) = \sum_{i=k}^{\infty} e^{(i)}$. It follows from [12] that there is a $c_1 \in (0, \infty)$ such that for all $i \in \mathbb{N}$ we have $e^{(i)} \leqslant c_1 \cdot 2^{-i/(n2^n)}$ where $n = |\Gamma|$. Using this inequality we get

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

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

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

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

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

The final statement follows from Corollary 1. □

Example 13. Using Newton approximants for the task system from Fig. 1 we obtain $\mathbb{E}[S^{op}] \approx 2.17$.

Example 14. Consider again the task system from Example 6 for the case $p = 1/2$, where we had $\Pr[S^{op} \geqslant k] = 2^{1-k}$. It follows that $\mathbb{E}[S^{op}] = \sum_{k=1}^{\infty} \Pr[S^{op} \geqslant k] = \sum_{k=1}^{\infty} 2^{1-k} = 2$.

5.2. Online schedulers

We show for online schedulers that the finiteness of $\mathbb{E}[S^\sigma]$ does not depend on the choice of the online scheduler σ.

Theorem 6. *If Δ is subcritical, then $\mathbb{E}[S^\sigma]$ is finite for every online scheduler σ. If Δ is critical, then $\mathbb{E}[S^\sigma]$ is infinite for every online scheduler σ.*

Proof sketch. The first assertion follows from Theorem 2. Let Δ be critical. For this sketch we focus on the case where X_0 is reachable from every type. By Proposition 2 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 2, 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 $\Pr[S^\sigma \geqslant k] \geqslant \boldsymbol{u}_{X_0}/(k+2)$. So we have $\mathbb{E}[S^\sigma] = \sum_{k=1}^{\infty} \Pr[S^\sigma \geqslant k] \geqslant \sum_{k=1}^{\infty} \boldsymbol{u}_{X_0}/(k+2) = \infty$. □

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 space.

We close this section by showing how an optimal online scheduler can be effectively approximated.

Theorem 7. *Let Δ be compact and subcritical, and let $\boldsymbol{v} \in (1,\infty)^\Gamma$ with $\boldsymbol{f}(\boldsymbol{v}) \leqslant \boldsymbol{v}$. Then there is a sequence of online schedulers $\sigma_1, \sigma_2, \ldots$ such that for all $m \in \mathbb{N}$:*

$$\mathbb{E}\big[S^{\sigma_m}\big] \leqslant \inf_{\sigma \text{ is online}} \mathbb{E}\big[S^\sigma\big] + \frac{\boldsymbol{v}_{X_0}}{\boldsymbol{v}_{min}^m(\boldsymbol{v}_{min}-1)}.$$

Moreover, a finite representation of each scheduler σ_m can be computed in time polynomial in $m^{|\Delta|}$.

Proof sketch. Here we only sketch the main argument. A full proof is given in Appendix C.2. By Theorem 2, for all online schedulers σ we have $\Pr[S^\sigma \geqslant k] \leqslant (\boldsymbol{v}_{X_0}-1)/(\boldsymbol{v}_{min}^k - 1)$. Since $\Pr[S^\sigma \geqslant k] \leqslant 1$, this implies $\Pr[S^\sigma \geqslant k]\boldsymbol{v}_{min}^k \leqslant \boldsymbol{v}_{X_0}$, or equivalently $\Pr[S^\sigma \geqslant k] \leqslant \boldsymbol{v}_{X_0}/\boldsymbol{v}_{min}^k$. Consequently, we have for all $m \in \mathbb{N}$:

$$\begin{aligned}
\mathbb{E}\big[S^\sigma\big] &= \sum_{k=1}^{\infty} \Pr\big[S^\sigma \geqslant k\big] = \sum_{k=1}^{m} \Pr\big[S^\sigma \geqslant k\big] + \sum_{k=m+1}^{\infty} \Pr\big[S^\sigma \geqslant k\big] \\
&\leqslant \sum_{k=1}^{m} \Pr\big[S^\sigma \geqslant k\big] + \sum_{k=m+1}^{\infty} \frac{\boldsymbol{v}_{X_0}}{\boldsymbol{v}_{min}^k} \\
&= \sum_{k=1}^{m} \Pr\big[S^\sigma \geqslant k\big] + \frac{\boldsymbol{v}_{X_0}}{\boldsymbol{v}_{min}^m(\boldsymbol{v}_{min}-1)}.
\end{aligned}$$

By choice of $\boldsymbol{v}$ we have $\boldsymbol{v}_{min} > 1$ which means that in order to obtain a σ_m it suffices to minimize the finite sum $F_\sigma := \sum_{k=1}^{m} \Pr[S^\sigma \geqslant k]$ (note that F_σ is the expectation of the random variable which assigns the completion space to trees with the completion space less than m and assigns m to others).

In order to minimize F_σ we may ignore all derivations containing a pool of tasks larger than m. Also, observe that the only information which an online scheduler needs to decide on a next step is the current pool of tasks and the maximal size of the pools in the history. These observations allow us to reduce the problem of minimizing F_σ to the problem of minimizing an expected total reward in a finite Markov decision process M of size $\mathcal{O}(m^{|\Delta|})$.

By [38, Theorem 7.1.9], there is a stationary policy Λ minimizing the expected total reward in M. Moreover, using [38, Theorem 7.2.18] such a Λ is computable via linear programming in time polynomial in the size of M, and thus polynomial in $m^{|\Delta|}$. The stationary policy Λ induces an online scheduler σ_m minimizing F_σ that can be finitely represented (it suffices to remember the maximal size of the task pool in the history). It follows that $\mathbb{E}[S^{\sigma_m}] \leqslant \inf_\sigma \mathbb{E}[S^\sigma] + \frac{\boldsymbol{v}_{X_0}}{\boldsymbol{v}_{min}^m(\boldsymbol{v}_{min}-1)}$. □

6. Conclusions

We have studied the problem of scheduling tasks in a processor when the tasks can stochastically generate other tasks. The problem has been thoroughly studied in the non-stochastic case, but, to the best of our knowledge, we are the first to consider it when the probabilities of task generation are known. In particular, the extensive literature on the theory of branching processes seems to have considered only systems in which a new task is immediately assigned to a fresh processor.

We have provided tail bounds on the performance of both online and offline schedulers for the case of one processor and task systems with completion probability 1. Due to surprising connections with Newton approximants and fixed points of the probability generating functions, we have proved that the bounds can be computed very efficiently.

The question of computing tail bounds when tasks are scheduled not on one but on a fixed number of processors is still open. This problem is already difficult in the non-stochastic case, because it exhibits a trade-off between time and space: while in the one-processor case the completion time is independent of the scheduler (and equal to the number of tasks that must be executed), this no longer holds for a larger number of processors.

Acknowledgments

We thank two anonymous referees for helpful comments.

Appendix A. Proofs of Section 3

A.1. Proof of Theorem 1

Here is a restatement of Theorem 1.

Theorem 1. $\Pr[S^{op}_X \leqslant k] = \boldsymbol{\nu}^{(k)}_X$ *for every type* X *and every* $k \geqslant 0$.

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, $\ell(t)$ 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 $k \geqslant 0$ and let t be an X-tree with $S^{op}(t) = k+1$. We have to show $\Pr[S^{op}_X = k+1] = \boldsymbol{\Delta}^{(k+1)}_X$ where

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

We show the following stronger claim:

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

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

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

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

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

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

$$\Pr\big[S^{op}_X(t) = k+1,\ \ell(t) = 0\big] = \Pr\big[t \text{ has two children},\ S^{op}(t_0) = S^{op}(t_1) = k\big] \tag{A.2}$$

because if t has one child, then $\ell(t) \geqslant 1$, and if t has no children, then $S^{op}_X(t) = 1$. Further we have by Proposition 3

$$\begin{aligned} \Pr\big[S^{op}_X(t) \leqslant k\big] = {} & \sum_{X \xhookrightarrow{p} \langle Y,Z\rangle} p \cdot \big(\Pr\big[S^{op}_Y(t_0) \leqslant k\big]\Pr\big[S^{op}_Z(t_1) \leqslant k\big] - \Pr\big[S^{op}_Y(t_0) = k\big]\Pr\big[S^{op}_Z(t_1) = k\big]\big) \\ & + \sum_{X \xhookrightarrow{p} Y} p \cdot \Pr\big[S^{op}_Y(t_0) \leqslant k\big] + \sum_{X \xhookrightarrow{p} \emptyset} p. \end{aligned} \tag{A.3}$$

Combining these equations we obtain

$$\begin{aligned} & \Pr\big[S^{op}_X(t) = k+1,\ \ell(t) = 0\big] \\ & \quad = \sum_{X \xhookrightarrow{p} \langle Y,Z\rangle} p \cdot \Pr\big[S^{op}_Y(t_0) = k\big]\Pr\big[S^{op}_Z(t_1) = k\big] && \big(\text{by (A.2)}\big) \\ & \quad = \sum_{X \xhookrightarrow{p} \langle Y,Z\rangle} p \cdot \Pr\big[S^{op}_Y(t_0) \leqslant k\big]\Pr\big[S^{op}_Z(t_1) \leqslant k\big] && \big(\text{by (A.3)}\big) \\ & \qquad + \sum_{X \xhookrightarrow{p} Y} p \cdot \Pr\big[S^{op}_Y(t_0) \leqslant k\big] + \sum_{X \xhookrightarrow{p} \epsilon} p - \Pr\big[S^{op}_X(t) \leqslant k\big] \end{aligned}$$

$$
\begin{aligned}
&= \sum_{X \xhookrightarrow{p} \langle Y,Z \rangle} p \cdot \boldsymbol{v}_Y^{(k)} \boldsymbol{v}_Z^{(k)} && \text{(ind. hyp. on } k\text{)} \\
&\quad + \sum_{X \xhookrightarrow{p} Y} p \cdot \boldsymbol{v}_Y^{(k)} + \sum_{X \xhookrightarrow{p} \epsilon} p - \boldsymbol{v}_X^{(k)} \\
&= \boldsymbol{f}_X(\boldsymbol{v}^{(k)}) - \boldsymbol{v}_X^{(k)} && \text{(def. of } \boldsymbol{f}\text{)}.
\end{aligned}
$$

For the induction step, let $i \geqslant 0$. Then by Proposition 3 and the definition of ℓ

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

completing the inductive proof of (A.1). □

Appendix B. Proofs of Section 4

B.1. Proof of Theorem 2

Here is a restatement of Theorem 2.

Theorem 2. *Let Δ be subcritical. Let $\boldsymbol{v}, \boldsymbol{w} \in (1,\infty)^\Gamma$ be vectors with $\boldsymbol{f}(\boldsymbol{v}) \leqslant \boldsymbol{v}$ and $\boldsymbol{f}(\boldsymbol{w}) \geqslant \boldsymbol{w}$. Denote by $\boldsymbol{v}_{min}$ and $\boldsymbol{w}_{max}$ the least component of $\boldsymbol{v}$ and the greatest component of $\boldsymbol{w}$, respectively. Then*

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

Proof. Let us fix an online scheduler σ. For any $i \geqslant 1$, let $\boldsymbol{z}^{(i)}$ be the vector of random variables that measures the number of tasks of each type at time i. Define $m^{(i)} := \boldsymbol{z}^{(i)} \bullet \boldsymbol{u}$ where "$\bullet$" denotes the scalar product. Note that $m^{(1)} = \boldsymbol{u}_{X_0}$. Let $h > 1$ and $\boldsymbol{u} \in (0,\infty)^\Gamma$ such that $h^{\boldsymbol{u}_Y} = \boldsymbol{v}_Y$ for all $Y \in \Gamma$.

Let us consider $i \geqslant 1$. Let us fix $c \in \mathbb{R}$ and $Y \in \Gamma$. Let T be the set of all family trees t such that $m^{(i)}(t) = c$ and $L(\sigma(t)[i]) = Y$. Assume that $T \neq \emptyset$. Let $r^{(i)}$ be the vector of random variables that measures the number of tasks of each type *created* at time i. Then we have $\boldsymbol{z}^{(i+1)} = \boldsymbol{z}^{(i)} + r^{(i)} - \langle Y \rangle$ (recall that $\langle Y \rangle$ stands for the vector $\boldsymbol{e} \in \{0,1\}^\Gamma$ with $\boldsymbol{e}_Y = 1$ and $\boldsymbol{e}_X = 0$ for $X \neq Y$).

An easy computation reveals that

$$
\mathbb{E}\big[h^{r^{(i)} \bullet \boldsymbol{u}} \,\big|\, T\big] = \mathbb{E}\Big[\prod_{Z \in \Gamma} h^{\boldsymbol{u}_Z \cdot r_Z^{(i)}} \,\Big|\, T\Big] = \mathbb{E}\Big[\prod_{Z \in \Gamma} \boldsymbol{v}_Z^{r_Z^{(i)}} \,\Big|\, T\Big] = \boldsymbol{f}_Y(\boldsymbol{v}) \leqslant \boldsymbol{v}_Y = h^{\boldsymbol{u}_Y}, \tag{B.1}
$$

as $\boldsymbol{f}(\boldsymbol{v}) \leqslant \boldsymbol{v}$. Consequently, we have

$$\begin{aligned}
\mathbb{E}\big[h^{m^{(i+1)}} \mid T\big] &= \mathbb{E}\big[h^{\boldsymbol{z}^{(i+1)}\bullet\boldsymbol{u}} \mid T\big] && \big(\text{def. of } m^{(i+1)}\big)\\
&= \mathbb{E}\big[h^{(\boldsymbol{z}^{(i)}+r^{(i)}-\langle Y\rangle)\bullet\boldsymbol{u}} \mid T\big] && \big(\text{def. of } r^{(i)}\big)\\
&= \mathbb{E}\big[h^{\boldsymbol{z}^{(i)}\bullet\boldsymbol{u}} \mid T\big]\cdot\mathbb{E}\big[h^{r^{(i)}\bullet\boldsymbol{u}} \mid T\big]\cdot\mathbb{E}\big[h^{-\boldsymbol{u}_Y} \mid T\big]\\
&= h^c\cdot\mathbb{E}\big[h^{r^{(i)}\bullet\boldsymbol{u}} \mid T\big]\cdot h^{-\boldsymbol{u}_Y} && \big(\text{def. of } m^{(i)}\big)\\
&\leqslant h^c && \big(\text{by (B.1)}\big).
\end{aligned}$$

As Y was chosen arbitrarily, we obtain

$$\mathbb{E}\big[h^{m^{(i+1)}} \mid m^{(i)}=c\big]\leqslant h^c.$$

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

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

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

Define the stopping time $\tau_k := \inf\{i\geqslant 1 \mid m^{(i)}\in\{0\}\cup[k,\infty)\}$. Note that $m^{(\tau_k)}\leqslant k+2\boldsymbol{u}_{max}$, and hence that $m^{(\tau_k)}\in\{0\}\cup[k,k+2\boldsymbol{u}_{max}]$. We wish to apply Doob's Optional–Stopping Theorem [34] (sometimes called Optional–Sampling Theorem) to infer that $\mathbb{E}[h^{m^{(\tau_k)}}]\leqslant\mathbb{E}[h^{m^{(1)}}]=\boldsymbol{v}_{X_0}$. To this end we define the sequence $\hat{m}^{(1)},\hat{m}^{(2)},\ldots$ by setting $\hat{m}^{(i)}:=m^{(i)}$ for $i\leqslant\tau_k$ and $\hat{m}^{(i)}:=m^{(\tau_k)}$ for $i\geqslant\tau_k$. The sequence $h^{\hat{m}^{(1)}}, h^{\hat{m}^{(2)}},\ldots$ is a supermartingale as $h^{m^{(1)}}, h^{m^{(2)}},\ldots$ is a supermartingale. To apply the Optional–Stopping Theorem we also need to make sure that $|h^{\hat{m}^{(i+1)}}-h^{\hat{m}^{(i)}}|$ is bounded by a constant, which is the case as $\hat{m}^{(i)}\in[0,k+2\boldsymbol{u}_{max}]$ for all i. Doob's Optional–Stopping Theorem now yields

$$\mathbb{E}\big[h^{m^{(\tau_k)}}\big]=\mathbb{E}\big[h^{\hat{m}^{(\tau_k)}}\big]\leqslant\mathbb{E}\big[h^{\hat{m}^{(1)}}\big]=\mathbb{E}\big[h^{m^{(1)}}\big]=h^{\boldsymbol{u}_{X_0}}=\boldsymbol{v}_{X_0}.$$

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

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

which gives

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

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

$$\Pr\big[S^\sigma\geqslant k\big]=\Pr\Big[\sup_i|\boldsymbol{z}^{(i)}|\geqslant k\Big]\leqslant\Pr\Big[\sup_i m^{(i)}\geqslant k\boldsymbol{u}_{min}\Big]=p_{k\boldsymbol{u}_{min}}\leqslant\frac{\boldsymbol{v}_{X_0}-1}{\boldsymbol{v}_{min}^k-1}. \tag{B.2}$$

So we have shown the upper bound.

For the lower bound we redefine h and $\boldsymbol{u}$ in terms of $\boldsymbol{v}$ instead of $\boldsymbol{w}$, i.e., $h^{\boldsymbol{u}_Y}=\boldsymbol{w}_Y$ for all $Y\in\Gamma$. This allows to show in an analogous way that

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

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

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

which gives

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

and thus

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

B.2. Proof of Proposition 5

Here is a restatement of Proposition 5.

Proposition 5. *We have*

$$\boldsymbol{f}\left(\mathbf{1}+\frac{\boldsymbol{u}}{q_{max}}\right)\leqslant \mathbf{1}+\frac{\boldsymbol{u}}{q_{max}}$$

where $\boldsymbol{u}\in[1,\infty)^{\Gamma}$ *is the vector of expected completion times, i.e.,* $\boldsymbol{u}_Y=\mathbb{E}[T_Y]$ *for all* $Y\in\Gamma$ *and* $q_{max}=\max_{X\in\Gamma}\sum_{X\xhookrightarrow{p}YZ} p\cdot\boldsymbol{u}_Y\cdot\boldsymbol{u}_Z$.

Moreover, we have

$$\boldsymbol{f}\left(\mathbf{1}+\frac{\boldsymbol{x}}{c\cdot\boldsymbol{x}_{min}^2}\right)\geqslant -\mathbf{1}+\frac{\boldsymbol{x}}{c\cdot\boldsymbol{x}_{min}^2}$$

where c *is the smallest nonzero coefficient of* $\boldsymbol{f}$, *and* $\boldsymbol{x}:=(I-\boldsymbol{f}'(\mathbf{1}))^{-1}\boldsymbol{y}$, *and* $\boldsymbol{x}_{min}>0$ *is the least component of* $\boldsymbol{x}$, *and* $\boldsymbol{y}\in\{0,1\}^{\Gamma}$ *is the vector such that* $\boldsymbol{y}_X=1$ *for the components* X *with* $X\xhookrightarrow{p}\langle Y,Z\rangle$ *for some* $Y,Z\in\Gamma$, *and* $\boldsymbol{y}_X=0$ *otherwise.*

Proof. 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 $\boldsymbol{f}(\boldsymbol{x})$, i.e., $Q(\boldsymbol{x},\boldsymbol{x})_X=\sum_{X\xhookrightarrow{p}\langle Y,Z\rangle} p\cdot\boldsymbol{x}_Y\cdot\boldsymbol{x}_Z$ for all $X\in\Gamma$. Then for any $\boldsymbol{x},\boldsymbol{y}\in\mathbb{R}^{\Gamma}$ we have $\boldsymbol{f}'(\boldsymbol{y})\boldsymbol{x}=L\boldsymbol{x}+Q(\boldsymbol{y},\boldsymbol{x})+Q(\boldsymbol{x},\boldsymbol{y})$. Hence, for all $r\in\mathbb{R}$ and $\boldsymbol{x}\in\mathbb{R}^{\Gamma}$:

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

The vector $\boldsymbol{u}$ of expected completion times is the unique solution of $\boldsymbol{u}=\boldsymbol{f}'(\mathbf{1})\boldsymbol{u}+\mathbf{1}$ which means that

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

Observe that q_{max} is the maximum of all components of $Q(\boldsymbol{u},\boldsymbol{u})$ and thus $\frac{Q(\boldsymbol{u},\boldsymbol{u})}{q_{max}}\leqslant\mathbf{1}$. It follows that

$$\boldsymbol{f}\left(\mathbf{1}+\frac{\boldsymbol{u}}{q_{max}}\right)=\mathbf{1}+\frac{\boldsymbol{u}}{q_{max}}-\frac{\mathbf{1}}{q_{max}}+\frac{1}{q_{max}}\frac{Q(\boldsymbol{u},\boldsymbol{u})}{q_{max}}\leqslant\mathbf{1}+\frac{\boldsymbol{u}}{q_{max}},$$

which shows the first part of the proposition.

For the second part of the proposition, note that $\boldsymbol{x}=(I-\boldsymbol{f}'(\mathbf{1}))^{-1}\boldsymbol{y}=\sum_{i=0}^{\infty}\boldsymbol{f}'(\mathbf{1})^i\boldsymbol{y}$. By the compactness of the task system, all types can reach a type X with $\boldsymbol{y}_X=1$. It follows that $\boldsymbol{x}=\sum_{i=0}^{\infty}\boldsymbol{f}'(\mathbf{1})^i\boldsymbol{y}$ is positive in all components, hence $\boldsymbol{x}_{min}>0$. Let $X\in\Gamma$. If $\boldsymbol{y}_X=1$, then $Q(\boldsymbol{x},\boldsymbol{x})_X\geqslant c\boldsymbol{x}_{min}^2$ by definition of c. If $\boldsymbol{y}_X=0$, then $Q(\boldsymbol{x},\boldsymbol{x})_X=0$. It follows

$$Q(\boldsymbol{x},\boldsymbol{x})\geqslant c\boldsymbol{x}_{min}^2\boldsymbol{y}. \tag{B.3}$$

We have:

$$\begin{aligned}\boldsymbol{f}\left(\mathbf{1}+\frac{\boldsymbol{x}}{c\boldsymbol{x}_{min}^2}\right)&=\mathbf{1}+\frac{\boldsymbol{f}'(\mathbf{1})\boldsymbol{x}}{c\boldsymbol{x}_{min}^2}+\frac{Q(\boldsymbol{x},\boldsymbol{x})}{c^2\boldsymbol{x}_{min}^4} && \text{(Taylor expansion)}\\ &=\mathbf{1}+\frac{\boldsymbol{x}-\boldsymbol{y}}{c\boldsymbol{x}_{min}^2}+\frac{Q(\boldsymbol{x},\boldsymbol{x})}{c^2\boldsymbol{x}_{min}^4} && \big(\text{as } \boldsymbol{y}=\boldsymbol{x}-\boldsymbol{f}'(\mathbf{1})\boldsymbol{x}\big)\\ &\geqslant\mathbf{1}+\frac{\boldsymbol{x}-\boldsymbol{y}}{c\boldsymbol{x}_{min}^2}+\frac{\boldsymbol{y}}{c\boldsymbol{x}_{min}^2} && \big(\text{by (B.3)}\big)\\ &=\mathbf{1}+\frac{\boldsymbol{x}}{c\boldsymbol{x}_{min}^2}. && \square\end{aligned}$$

B.3. Proof of Theorem 3

Here is a restatement of Theorem 3.

Theorem 3. *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*:

$$\textit{maximize } d \textit{ such that } \forall Y\in\Gamma:\quad 0\leqslant d\leqslant\boldsymbol{v}_Y\geqslant\boldsymbol{f}_Y(\boldsymbol{v}).$$

Proof. Considering the proof sketch in the main body of the paper, it remains to provide (b) a point $\boldsymbol{v}$ inside the convex space, and (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.

For the radius R, note that all feasible points $\boldsymbol{x}$ satisfy $\boldsymbol{x}_Y \cdot A_Y\boldsymbol{x} + c_Y \leqslant \boldsymbol{x}_Y$ for all $Y \in \Gamma$, implying $A_Y\boldsymbol{x} \leqslant 1$. Also note that the compactness of the task system implies that for all $Y \in \Gamma$ the diagonal entry A_{YY} is nonzero. Together with $A_Y\boldsymbol{x} \leqslant 1$ this implies that for all $Y \in \Gamma$ and all feasible vectors $\boldsymbol{x}$ we have $\boldsymbol{x}_Y \leqslant 1/a_{min}$, where $a_{min} \geqslant 2^{-b}$ 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 $\boldsymbol{v} \geqslant \mathbf{1}$ and $\delta \geqslant 2^{-\mathrm{poly}(|\Gamma|,b)}$ such that every point $\boldsymbol{x}$ with $\|\boldsymbol{x} - \boldsymbol{v}\|_\infty \leqslant \delta$ is feasible. (Note that d poses no problem: it can be chosen as $\frac{1}{2}$.)

From $\boldsymbol{f}_Y(\boldsymbol{x}) = \boldsymbol{x}_Y \cdot A_Y\boldsymbol{x} + c_Y$ it follows that

$$\boldsymbol{f}'_Y(\mathbf{1}) = A_Y + (A_Y\mathbf{1})\boldsymbol{e}(Y), \tag{B.4}$$

where $\boldsymbol{e}(Y)$ denotes a row vector with all zeros except for the Y-component where it is 1. Recall from the remarks made at the beginning of the proof of Proposition 2 that a vector $\boldsymbol{u} \geqslant \mathbf{1}$ with $\boldsymbol{u} = \boldsymbol{f}'(\mathbf{1})\boldsymbol{u} + \mathbf{1}$ exists. We have for all $Y \in \Gamma$:

$$1 + r\boldsymbol{u}_Y - \boldsymbol{f}_Y(\mathbf{1} + r\boldsymbol{u}) = 1 + r\boldsymbol{u}_Y - (1 + r\boldsymbol{u}_Y)A_Y(\mathbf{1} + r\boldsymbol{u}) - c_Y$$

since $A_Y\mathbf{1} + c_Y = 1$:

$$= r\boldsymbol{u}_Y - rA_Y\boldsymbol{u} - r\boldsymbol{u}_Y A_Y\mathbf{1} - r^2\boldsymbol{u}_Y A_Y\boldsymbol{u}$$

by (B.4):

$$= r\boldsymbol{u}_Y - r\boldsymbol{f}'_Y(\mathbf{1})\boldsymbol{u} - r^2\boldsymbol{u}_Y A_Y\boldsymbol{u}$$

since $\boldsymbol{u} = \boldsymbol{f}'(\mathbf{1})\boldsymbol{u} + \mathbf{1}$:

$$= r(1 - r\boldsymbol{u}_Y A_Y\boldsymbol{u})$$

letting $\boldsymbol{u}_{max}$ denote the largest component of $\boldsymbol{u}$ we have $A_Y\boldsymbol{u} \leqslant \boldsymbol{u}_{max}$ and consequently:

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

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

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

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

$$= 2\delta.$$

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

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

$$\begin{aligned}
\boldsymbol{x} - \boldsymbol{f}(\boldsymbol{x}) &\geqslant \mathbf{1} + r\boldsymbol{u} - 2\delta\mathbf{1} - \boldsymbol{f}(\boldsymbol{x}) && (\text{as } \boldsymbol{x} \geqslant \mathbf{1} + r\boldsymbol{u} - 2\delta\mathbf{1})\\
&\geqslant \mathbf{1} + r\boldsymbol{u} - 2\delta\mathbf{1} - \boldsymbol{f}(\mathbf{1} + r\boldsymbol{u}) && (\text{as } \boldsymbol{x} \leqslant \mathbf{1} + r\boldsymbol{u})\\
&\geqslant 2\delta\mathbf{1} - 2\delta\mathbf{1} = \mathbf{0} && (\text{by the computation above}).
\end{aligned}$$

So if we set $\boldsymbol{v} := \mathbf{1} + r\boldsymbol{u} - \delta\mathbf{1}$, then all $\boldsymbol{x}$ with $\|\boldsymbol{x} - \boldsymbol{v}\|_\infty \leqslant \delta$ are feasible. Furthermore, $\delta = 1/(16\boldsymbol{u}_{max}^2) \geqslant 2^{-\mathrm{poly}(|\Gamma|,b)}$ 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. □

B.4. Proof of Theorem 4

Here is a restatement of Theorem 4.

Theorem 4. *Let Δ be subcritical and $\boldsymbol{v} \in (1,\infty)^\Gamma$ with $\boldsymbol{f}(\boldsymbol{v}) \leqslant \boldsymbol{v}$. Let σ be a $\boldsymbol{v}$-light-first scheduler. Let $\boldsymbol{v}_{min\,max} := \min_{X \hookrightarrow \langle Y,Z\rangle} \max\{\boldsymbol{v}_Y, \boldsymbol{v}_Z\}$ (here the minimum is taken over all transition rules with two types on the right-hand side). Then $\boldsymbol{v}_{min\,max} \geqslant \boldsymbol{v}_{min}$ and for all $k \geqslant 1$*

$$\Pr\big[S^\sigma \geqslant k\big] \leqslant \frac{\boldsymbol{v}_{X_0} - 1}{\boldsymbol{v}_{min}\boldsymbol{v}_{min\,max}^{k-1} - 1}.$$

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

$$\Pr\big[S^\sigma \geqslant k\big] \leqslant \frac{\boldsymbol{v}_{X_0} - 1}{\boldsymbol{v}_{min}^{\ell}\boldsymbol{v}_{min\,acc}^{k-\ell} - 1}.$$

Proof. The inequality $\boldsymbol{v}_{min\,max} \geqslant \boldsymbol{v}_{min}$ is trivial. For the inequality $\boldsymbol{v}_{min\,acc} \geqslant \boldsymbol{v}_{min\,max}$, let $Li := \{Y \in \Gamma \mid \boldsymbol{v}_Y < \boldsymbol{v}_{min\,max}\}$ be the set of types that are strictly lighter than $\boldsymbol{v}_{min\,max}$. 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 $\boldsymbol{z}^{(i)} \bullet \boldsymbol{e}^{(Li)} \leqslant 1$ 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 $\boldsymbol{v}_{min\,max}$, at most one of the new tasks has Li-type. Hence, the types in Li are not accumulating. It follows $\boldsymbol{v}_{min\,acc} \geqslant \boldsymbol{v}_{min\,max}$.

The inequalities in the theorem are obtained by a small modification of the proof of Theorem 2: it suffices to show that, in Eq. (B.2), we can replace $k\boldsymbol{u}_{min}$ by $\boldsymbol{u}_{min} + (k-1)\boldsymbol{u}_{min\,max}$ and by $\ell\boldsymbol{u}_{min} + (k-\ell)\boldsymbol{u}_{min\,acc}$ for some integer ℓ. (The values $\boldsymbol{u}_{min\,max}$ and $\boldsymbol{u}_{min\,acc}$ are defined in the obvious way, i.e., using the h from the proof of Theorem 2 we have $h^{\boldsymbol{u}_{min\,max}} = \boldsymbol{v}_{min\,max}$ and $h^{\boldsymbol{u}_{min\,acc}} = \boldsymbol{v}_{min\,acc}$.) So we need to show for the light-first scheduler σ that $|\boldsymbol{z}^{(i)}| \geqslant k$ implies both $m^{(i)} \geqslant \boldsymbol{u}_{min} + (k-1)\boldsymbol{u}_{min\,max}$ and $m^{(i)} \geqslant \ell\boldsymbol{u}_{min} + (k-\ell)\boldsymbol{u}_{min\,acc}$.

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

For the second implication, let ℓ' be an integer such that $\boldsymbol{z}^{(i)}_Y \leqslant \ell'$ 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 $m^{(i)} \geqslant \ell\boldsymbol{u}_{min} + (k-\ell)\boldsymbol{u}_{min\,acc}$.

It remains to show that the set of $\boldsymbol{v}$-accumulating types can be computed in polynomial time. 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 $Z \leqslant X$. Further, we write $Y \stackrel{\bullet}{\Longrightarrow}_{lf} \alpha$ to denote that the light-first scheduler can generate α. Finally, we denote by $\alpha^{\geqslant X}$ ($\alpha^{>X}$) the restriction of α to types $Y \geqslant X$ ($Y > X$).

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

($\Rightarrow$): Assume X is $\boldsymbol{v}$-accumulating. Then $X_0 \stackrel{\bullet}{\Longrightarrow}_{lf} n \cdot X$ holds for infinitely many $n \geqslant 1$. We claim that there exists a type W such that $W \stackrel{\bullet}{\Longrightarrow}_X n \cdot X$ for infinitely many $n \geqslant 1$. For the claim, take the longest suffixes of the witnesses for $X_0 \stackrel{\bullet}{\Longrightarrow}_{lf} n \cdot X$ that only use rules X-bounded rules, and let α_n be their corresponding initial multisets. These suffixes are then witnesses for $\alpha_n \stackrel{\bullet}{\Longrightarrow}_X n \cdot X$. If $\alpha_n = X_0$ holds for infinitely many $n \geqslant 1$, take $W := X_0$. Otherwise, let $Z_n \hookrightarrow \beta_n$ be the rule applied to obtain α_n. Then

$$X_0 \Rightarrow^*_{lf} (\alpha_n - \beta_n) + Z_n \Rightarrow_{lf} (\alpha_n - \beta_n) + \beta_n \stackrel{\bullet}{\Longrightarrow}_X n \cdot X\,,$$

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

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

($\Leftarrow$): We start with some simple properties of the relations $\Rightarrow^*_X$ and $\Rightarrow^*_{lf}$.

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

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

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

Appendix C. Proofs of Section 5

C.1. Proof of Theorem 6

Here is a restatement of Theorem 6.

Theorem 6. *If Δ is subcritical, then $\mathbb{E}[S^\sigma]$ is finite for every online scheduler σ. If Δ is critical, then $\mathbb{E}[S^\sigma]$ is infinite for every online scheduler σ.*

Proof. Let Δ be subcritical. By Theorem 2 we have for every online scheduler σ

$$\mathbb{E}\big[S^\sigma\big] = \sum_{k=1}^{\infty} \Pr\big[S^\sigma \geqslant k\big] \leqslant \sum_{k=1}^{\infty} \frac{\boldsymbol{v}_{X_0} - 1}{\boldsymbol{v}_{min}^k - 1} < \infty,$$

because it is a geometric series.

Let now Δ be critical. The proof follows the lines of the proof of Theorem 2. By Proposition 2 we have $\rho(\boldsymbol{f}'(\mathbf{1})) = 1$ for the spectral radius of $\boldsymbol{f}'(\mathbf{1})$.

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 $X \in \Gamma$. Later we will show how to drop this assumption. If X_0 is reachable from every X, it follows that $\boldsymbol{f}'(\mathbf{1})$ is an irreducible matrix. Then Perron–Frobenius theory [27] guarantees the existence of an eigenvector $\boldsymbol{u} \in \mathbb{R}^\Gamma$ of $\boldsymbol{f}'(\mathbf{1})$ which is positive in all components, i.e., $\boldsymbol{f}'(\mathbf{1})\boldsymbol{u} = \boldsymbol{u}$ and $\boldsymbol{u}_X > 0$ for all $X \in \Gamma$. W.l.o.g. we can choose $\boldsymbol{u}$ such that its largest component is 1. As in the proof of Theorem 2, let $m^{(i)} := \boldsymbol{z}^{(i)} \bullet \boldsymbol{u}$, where $\boldsymbol{z}^{(i)}$ denotes the vector of random variables that measures the number of tasks of each type at time i. Note that $m^{(1)} = \boldsymbol{u}_{X_0} > 0$ and $m^{(i)} \leqslant |\boldsymbol{z}^{(i)}|$ where $|\boldsymbol{z}^{(i)}|$ denotes the sum of the components of $\boldsymbol{z}^{(i)}$. Observe 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 $i \geqslant 1$. Let us fix $c > 0$ and $Y \in \Gamma$. Let T be the set of all family trees t such that $m^{(i)}(t) = c$ and $L(\sigma(t)[i]) = Y$. Assume that $T \neq \emptyset$. As in the proof of Theorem 2, let $r^{(i)}$ be the vector of random variables that measures the number of tasks of each type created at time i, so that we have $\boldsymbol{z}^{(i+1)} = \boldsymbol{z}^{(i)} + r^{(i)} - \langle Y \rangle$. An easy computation reveals that for every $X \in \Gamma$ we have

$$\mathbb{E}\big[r_X^{(i)} \mid T\big] = \sum_{Y \xrightarrow{p} \alpha} p \cdot \#_X(\alpha) = \boldsymbol{f}'_{Y,X}(\mathbf{1})$$

which gives

$$\mathbb{E}\big[r^{(i)} \mid T\big] = \boldsymbol{f}'_Y(\mathbf{1}) \tag{C.1}$$

(where $\boldsymbol{f}'_Y(\mathbf{1})$ denotes the row vector indexed by Y). Consequently, we have:

$$\begin{aligned}
\mathbb{E}\big[m^{(i+1)} \mid T\big] &= \mathbb{E}\big[\boldsymbol{z}^{(i+1)} \mid T\big] \bullet \boldsymbol{u} && \big(\text{def. of } m^{(i+1)}\big)\\
&= \big(\mathbb{E}\big[\boldsymbol{z}^{(i)} \mid T\big] + \mathbb{E}\big[r^{(i)} \mid T\big] - \mathbb{E}\big[\langle Y \rangle \mid T\big]\big) \bullet \boldsymbol{u} && \big(\text{def. of } r^{(i)}\big)\\
&= \big(\mathbb{E}\big[\boldsymbol{z}^{(i)} \mid T\big] + \boldsymbol{f}'_Y(\mathbf{1}) - \langle Y \rangle\big) \bullet \boldsymbol{u} && \big(\text{by (C.1)}\big)\\
&= c + \boldsymbol{f}'_Y(\mathbf{1})\boldsymbol{u} - \langle Y \rangle \bullet \boldsymbol{u} && \big(\text{def. of } m^{(i)}(T)\big)\\
&= c \quad \big(\text{as } \boldsymbol{f}'(\mathbf{1})\boldsymbol{u} = \boldsymbol{u}\big).
\end{aligned}$$

As Y was chosen arbitrarily, we obtain

$$\mathbb{E}\big[h^{m^{(i+1)}} \mid m^{(i)} = c\big] = c.$$

Also clearly $\mathbb{E}[m^{(i+1)} \mid m^{(i)} = 0] = 0$, 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 $\tau_k := \inf\{i \geqslant 1 \mid m^{(i)} \in \{0\} \cup [k, \infty)\}$. Note that $m^{(\tau_k)} \leqslant k + 2$ as $\boldsymbol{u} \leqslant \mathbf{1}$, and hence that $m^{(\tau_k)} \in \{0\} \cup [k, k+2]$. We wish to apply Doob's Optional–Stopping Theorem [34] (sometimes called Optional–Sampling Theorem) to infer that $\mathbb{E}[m^{(\tau_k)}] = \mathbb{E}[m^{(1)}] = \boldsymbol{u}_{X_0}$. To this end we define the sequence $\hat{m}^{(1)}, \hat{m}^{(2)}, \ldots$ by setting $\hat{m}^{(i)} := m^{(i)}$ for $i \leqslant \tau_k$ and $\hat{m}^{(i)} := m^{(\tau_k)}$ for $i \geqslant \tau_k$. The sequence $\hat{m}^{(1)}, \hat{m}^{(2)}, \ldots$ 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 $|\hat{m}^{(i+1)} - \hat{m}^{(i)}|$ is bounded by a constant, which is the case as $\hat{m}^{(i)} \in [0, k+2]$ for all i. Doob's Optional–Stopping Theorem now yields

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

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

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

which gives

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

So we have

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

Hence,

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

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 [27]) 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 $\alpha \in M_\Gamma^{\leqslant 2}$ 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 $\mathbb{E}[S_X^\sigma] = \infty$ 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 $\mathbb{E}[S_X^\sigma] = \infty$ for all online schedulers σ. As X is reachable from X_0, it follows $\mathbb{E}[S^\sigma] = \infty$ for all online schedulers σ. □

C.2. Proof of Theorem 7

Theorem 7. *Let Δ be compact and subcritical, and let $\boldsymbol{v} \in (1,\infty)^\Gamma$ with $\boldsymbol{f}(\boldsymbol{v}) \leqslant \boldsymbol{v}$. Then there is a sequence of online schedulers $\sigma_1, \sigma_2, \ldots$ such that for all $m \in \mathbb{N}$:*

$$\mathbb{E}\big[S^{\sigma_m}\big] \leqslant \inf_{\sigma \text{ is online}} \mathbb{E}\big[S^\sigma\big] + \frac{\boldsymbol{v}_{X_0}}{\boldsymbol{v}_{min}^m(\boldsymbol{v}_{min} - 1)}.$$

Moreover, a finite representation of each scheduler σ_m can be computed in time polynomial in $m^{|\Delta|}$.

Proof. We have already proved in the sketch that for all online schedulers σ and all $m \in \mathbb{N}$:

$$\mathbb{E}\big[S^\sigma\big] \leqslant \sum_{k=1}^{m} \Pr\big[S^\sigma \geqslant k\big] + \frac{\boldsymbol{v}_{X_0}}{\boldsymbol{v}_{min}^m(\boldsymbol{v}_{min} - 1)}.$$

So it suffices to minimize the finite sum $F_\sigma := \sum_{k=1}^m \Pr[S^\sigma \geqslant k]$. We reduce the problem of minimizing F_σ to the problem of minimizing an expected total reward in a finite Markov decision process M defined as follows:

- The set of states S of M is

  $$\big\{(\boldsymbol{c}, k) \in \mathbb{N}^\Gamma \times \{1, \ldots, m\} \;\big|\; |\boldsymbol{c}| \leqslant m\big\}.$$

  (Here the first component, $\boldsymbol{c}$, is used to simulate the pool of tasks and the second component, k, stores the maximum from the history.)
- Each $(\boldsymbol{c}, k)$ satisfying $0 < |\boldsymbol{c}| < m$ is assigned a set of actions $A_{\boldsymbol{c},k}$ which consists of all task types that occur in $\boldsymbol{c}$. (We assume that other states are absorbing and do not assign any actions to them.)
- To each pair $((\boldsymbol{c}, k), X)$, here $0 < |\boldsymbol{c}| < m$ and $X \in A_{\boldsymbol{c},k}$, we assign a probability distribution $p(\cdot \mid (\boldsymbol{c}, k), X)$ defined by

  $$p\big((\boldsymbol{c}', k') \mid (\boldsymbol{c}, k), X\big) := \begin{cases} p & \text{if } X \xhookrightarrow{p} \alpha \text{ where } \boldsymbol{c}' = (\boldsymbol{c} \setminus X) \cup \alpha \text{ and} \\ & |\boldsymbol{c}| \leqslant m-1 \text{ and } k' = \max(k, |\boldsymbol{c}'|) \\ 0 & \text{otherwise.} \end{cases}$$

- A reward function r is defined by

  $$r((\boldsymbol{c}, k)) := \begin{cases} 0 & \text{if } 0 < |\boldsymbol{c}| < m; \\ k & \text{otherwise.} \end{cases}$$

Intuitively, the MDP M simulates M in the first component of states and stores the maximal size of states in the second component. Once it enters a state of the form $(\boldsymbol{c}, k)$ where either $|\boldsymbol{c}| = 0$, or $|\boldsymbol{c}| = m$, the process stops and collects a reward equal to the maximal size of the first component of states encountered in the history. See Fig. 3 for an example.

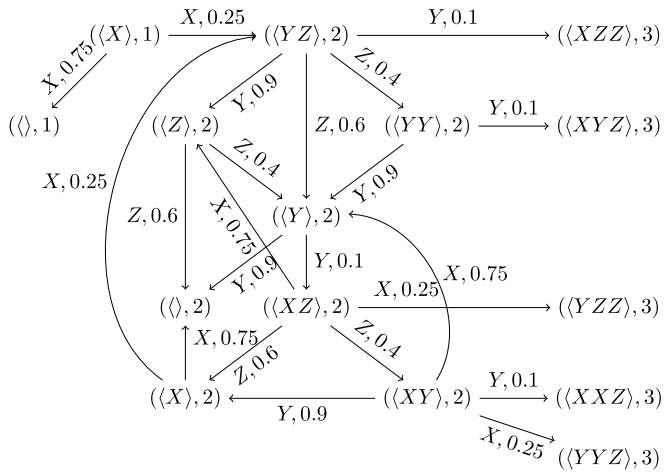

Fig. 3. The reachable part of the MDP associated with the task system of Example 1 for $m = 3$ and initial type $X_0 = X$.

As only absorbing states have positive reward, any infinite path in M has zero reward. We therefore define formally: a path π in M is a *finite* sequence $\pi = (\boldsymbol{c}^{(1)}, k_1), a_1, (\boldsymbol{c}^{(2)}, k_2), a_2, \ldots, a_{n-1}, (\boldsymbol{c}^{(n)}, k_n)$ where each $(\boldsymbol{c}^{(i)}, k_i) \in S$ and $a_i \in A_{\boldsymbol{c}^{(i)}, k_i}$ and $(\boldsymbol{c}^{(1)}, k_1) = (\langle X_0\rangle, 1)$; π is *complete* if $(\boldsymbol{c}^{(n)}, k_n)$ is absorbing, i.e., if either $|\boldsymbol{c}^{(n)}| = 0$, or $|\boldsymbol{c}^{(n)}| = m$; otherwise π is *incomplete*. A *policy* is a function Λ which for every incomplete path $\pi = (\boldsymbol{c}^{(1)}, k_1), a_1, (\boldsymbol{c}^{(2)}, k_2), a_2, \ldots, a_{n-1}, (\boldsymbol{c}^{(n)}, k_n)$ returns $\Lambda(\pi) \in A_{\boldsymbol{c}^{(n)}, k_n}$. Given a policy Λ, we define a probability, $P^{\Lambda}(\pi)$, of following the path π in M using the policy Λ as follows: If $\Lambda((\boldsymbol{c}^{(1)}, k_1), a_1, \ldots, a_{i-1}, (\boldsymbol{c}^{(i)}, k_i)) = a_i$ for all $1 \leqslant i \leqslant n$, then $P^{\Lambda}(\pi) := \prod_{i=1}^{n-1} p((\boldsymbol{c}^{(i+1)}, k_{i+1}) \mid (\boldsymbol{c}^{(i)}, k_i), a_i)$. Otherwise, we define $P^{\Lambda}(\pi) := 0$.

Each path $\pi = (\boldsymbol{c}^{(1)}, k_1), a_1, (\boldsymbol{c}^{(2)}, k_2), a_2, \ldots, a_{n-1}, (\boldsymbol{c}^{(n)}, k_n)$ is assigned an accumulated total reward $r(\pi)$ defined by $\sum_{i=1}^{n} r((\boldsymbol{c}^{(i)}, k_i))$. We define

$$E_{\Lambda} := \sum_{\pi \text{ is a complete path in } M} r(\pi) \cdot P^{\Lambda}(\pi)$$

the expected total reward on paths in M.

It is now not hard to see that there is a natural correspondence of online schedulers and policies. This is formally stated in the following proposition:

Proposition 6.

1. *Each online scheduler σ induces a policy Λ such that $F_{\sigma} = E_{\Lambda}$.*
2. *Each policy Λ induces an online scheduler σ such that $F_{\sigma} = E_{\Lambda}$.*

We first finalize the proof of Theorem 7 and postpone the proof of Proposition 6.

It follows that

$$\inf_{\sigma \text{ is an online scheduler}} F_{\sigma} = \inf_{\Lambda \text{ is a policy in } M} E_{\Lambda}.$$

By [38, Theorem 7.1.9], there is a stationary policy Λ in M such that $E_{\Lambda} = \inf_{\Lambda \text{ is a policy in } M} E_{\Lambda}$. Moreover, using [38, Theorem 7.2.18] such a policy is computable via linear programming in time polynomial in the size of M, and thus polynomial in $m^{|\Delta|}$. This policy Λ induces an online scheduler σ minimizing F_{σ} that can be finitely represented (it suffices to remember maximal size of pools of tasks in the history). It follows that $\mathbb{E}[S^{\sigma}] \leqslant \inf_{\pi} \mathbb{E}[S^{\pi}] + \frac{\boldsymbol{v}_{X_0}}{\boldsymbol{v}_{min}(\boldsymbol{v}_{min}-1)}$. This completes the proof of Theorem 7. □

We now turn to the proof of Proposition 6.

Proposition 6.

1. *Each online scheduler σ induces a policy Λ such that $F_{\sigma} = E_{\Lambda}$.*
2. *Each policy Λ induces an online scheduler σ such that $F_{\sigma} = E_{\Lambda}$.*

Proof. We use the notations and assumptions stated in the preceding proof of Theorem 7.

ad 1. Recall that for every tree t with $\sigma(t) = (s_1 \Rightarrow \cdots \Rightarrow s_k)$ and for every $j \geqslant 0$ we denote by $\boldsymbol{z}^{(j)}(t)$ the multiset of types labelling the tasks of s_j if $j \leqslant k$ (i.e., $\boldsymbol{z}^{(j)}(t) = \langle L(w) \mid w \in s_j \rangle$), and the empty multiset otherwise.
We define a policy Λ as follows: Let $\omega = (\boldsymbol{c}^{(1)}, k_1), a_1, \ldots, a_{i-1}, (\boldsymbol{c}^{(i)}, k_i)$ be an incomplete path in M. If there is a tree t such that $\boldsymbol{c}^{(1)} = \boldsymbol{z}^{(1)}(t), \ldots, \boldsymbol{c}^{(i)} = \boldsymbol{z}^{(i)}(t)$, we define $\Lambda(\omega)$ to be $L(\sigma(t)[i])$. Otherwise, we define $\Lambda(\omega)$ to be the least type w.r.t. $\preccurlyeq$.
We show by induction on i that the first $i \geqslant 1$ states of $\sigma(t)$ are uniquely determined by $\boldsymbol{z}^{(1)}(t), \ldots, \boldsymbol{z}^{(i)}(t)$ which implies that Λ is well defined. 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.
It follows easily from definitions that for each tree t such that $S^{\sigma}(t) < m$ there is a unique complete path ω of the form

$$\big(\boldsymbol{z}^{(1)}(t), k_1\big), \sigma(t)[1], \big(\boldsymbol{z}^{(2)}(t), k_2\big), \sigma(t)[2], \ldots, \sigma(t)[n-1], \big(\boldsymbol{z}^{(n-1)}(t), k_{n-1}\big).$$

Here each k_i is equal to $\max\{|\boldsymbol{z}^{(j)}| \mid 1 \leqslant j \leqslant i\}$. It is straightforward to show that the probability $P^{\Lambda}(\omega)$ of following ω in M using Λ is equal to the probability of the family tree t. It follows that $F_{\sigma} = E_{\Lambda}$.

ad 2. Let $\omega = (\boldsymbol{c}^{(1)}, k_1), a_1, \ldots, a_{n-1}, (\boldsymbol{c}^{(n)}, k_n)$ be a complete path in M such that for each $1 \leqslant i \leqslant n-1$ we have $a_i = \Lambda((\boldsymbol{c}^{(1)}, k_1), a_1 \cdots a_{i-1}, (\boldsymbol{c}^{(i)}, k_i))$ and $p(\boldsymbol{c}^{(i)} \mid \boldsymbol{c}^{(i-1)}, a_i) > 0$.
First assume that $k_n < m$. We define a family tree $t = (N, L)$ and its derivation $s_1 \Rightarrow \cdots \Rightarrow s_n$ corresponding to ω as follows. We put $s_1 := \{\varepsilon\}$ and $L(\varepsilon) := X_0$. Assume that s_i has already been defined and that the multiset $\langle L(w) \mid w \in s_i \rangle$ is equal to $\boldsymbol{c}^{(i)}$. Denote by X the type $\Lambda((\boldsymbol{c}^{(1)}, k_1) \cdots (\boldsymbol{c}^{(i)}, k_i))$. By definition, $X \in \boldsymbol{c}^{(i)}$ and so there is $w \in s_i$ such that $L(w) = X$. Assume that w is the left most node of s_i with this property (this assumption gives us a unique online scheduler). Assume that $\boldsymbol{c}^{(i+1)} = (\boldsymbol{c}^{(i)} \setminus \langle X \rangle) \cup \alpha$ for some transition rule $X \overset{p}{\hookrightarrow} \alpha$. Then children of w in t as well as the next state s_{i+1} are defined as follows:
- If $\alpha = \langle Y, Z \rangle$, where $Y \preccurlyeq Z$, then w has two children, $w0$ and $w1$, labeled $L(w0) = Y$ and $L(w1) = Z$ and we put $s_{i+1} = (s_i \setminus \{w\}) \cup \{w0, w1\}$.
- If $\alpha = \langle Y \rangle$, then w has one child, $w0$, labeled $L(w0) = Y$ and we put $s_{i+1} = (s_i \setminus \{w\}) \cup \{w0\}$.
- If $\alpha = \emptyset$, then w is a leaf and we put $s_{i+1} = s_i \setminus \langle w \rangle$.

It is easy to verify that $s_1 \Rightarrow \cdots \Rightarrow s_n$ is indeed a derivation of t according to a fixed online scheduler σ. Moreover, the probability of t is equal to the probability of following ω in M using the policy Λ. Also $S^{\sigma}(t) = k_n = r(\omega)$.
If $k_n = m$, we define a set T of family trees $t = (N, L)$ together with their derivations of the form $\omega_t = s_1 \Rightarrow \cdots \Rightarrow s_{n-1} \Rightarrow s'_0 \Rightarrow s'_1 \Rightarrow \cdots \Rightarrow s'_k$ as follows. The sequence $s_1 \Rightarrow \cdots \Rightarrow s_{n-1} \Rightarrow s'_0$ and the restriction of L to $\bigcup s_i \cup s'_0$ is obtained from ω using the same procedure as above and $s'_1 \Rightarrow \cdots \Rightarrow s'_k$ is obtained by always choosing the left most node to expand. More precisely, let w be the left most node of s'_i. Then for some rule $L(w) \hookrightarrow \alpha$ we define children of w and s'_{i+1} as follows:
- If $\alpha = \langle Y, Z \rangle$, where $Y \preccurlyeq Z$, then w has two children $w0$ and $w1$, labeled $L(w0) = Y$ and $L(w1) = Z$ and we put $s'_{i+1} = (s'_i \setminus \{w\}) \cup \{w0, w1\}$.
- If $\alpha = \langle Y \rangle$, then w has one child, $w0$, labeled $L(w0) = Y$ and we put $s'_{i+1} = (s_i \setminus \{w\}) \cup \{w0\}$.
- If $\alpha = \emptyset$, then w is a leaf and we put $s'_{i+1} = s'_i \setminus w$.

Now clearly the probability of T is equal to the probability of following ω in M using the policy Λ. All derivations defined above are obtained using a fixed scheduler σ. For all $t \in T$ we have that $S^{\sigma}(t) \geqslant m$.
This gives us a scheduler σ such that $F_{\sigma} = E_{\Lambda}$. □

References

[1] Y. Hirshfeld, M. Jerrum, F. Moller, A polynomial algorithm for deciding bisimilarity of normed context-free processes, Theoret. Comput. Sci. 158 (1–2) (1996) 143–159.
[2] H. Hüttel, N. Kobayashi, T. Suto, Undecidable equivalences for basic parallel processes, Inform. and Comput. 207 (7) (2009) 812–829.
[3] S.B. Fröschle, P. Jancar, S. Lasota, Z. Sawa, Non-interleaving bisimulation equivalences on basic parallel processes, Inform. and Comput. 208 (1) (2010) 42–62.
[4] J. Esparza, A. Kučera, R. Mayr, Quantitative analysis of probabilistic pushdown automata: Expectations and variances, in: LICS 2005, IEEE, 2005, pp. 117–126.
[5] R. Karp, Y. Zhang, Randomized parallel algorithms for backtrack search and branch-and-bound computation, J. ACM 40 (3) (1993) 765–789.
[6] G. Narlikar, G. Belloch, Space-efficient scheduling of nested parallelism, ACM TOPLAS 21 (1) (1999) 138–173.
[7] R. Blumofe, C. Leiserson, Scheduling multithreaded computations by work stealing, J. ACM 46 (5) (1999) 720–748.
[8] N. Arora, R. Blumofe, C. Plaxton, Thread scheduling for multiprogrammed microprocessors, Theory Comput. Syst. 34 (2001) 115–144.
[9] K. Agrawal, C. Leiserson, Y. He, W. Hsu, Adaptive work-stealing with parallelism feedback, ACM Trans. Comput. Systems 26 (3) (2008).
[10] T. Harris, The Theory of Branching Processes, Springer, 1963.
[11] S. Kiefer, M. Luttenberger, J. Esparza, On the convergence of Newton's method for monotone systems of polynomial equations, in: STOC 2007, ACM, 2007, pp. 217–226.
[12] J. Esparza, S. Kiefer, M. Luttenberger, Convergence thresholds of Newton's method for monotone polynomial equations, in: STACS 2008, 2008, pp. 289–300.

[13] K. Athreya, P. Ney, Branching Processes, Springer, 1972.
[14] K. Athreya, On the maximum sequence of a critical branching process, Ann. Probab. 16 (1988) 502–507.
[15] K. Borovkov, V. Vatutin, On distribution tails and expectations of maxima in critical branching processes, J. Appl. Probab. 33 (3) (1996) 614–622.
[16] T. Lindvall, On the maximum of a branching process, Scand. J. Stat. 3 (1976) 209–214.
[17] O. Nerman, On the maximal generation size of a non-critical Galton–Watson process, Scand. J. Stat. 4 (3) (1977) 131–135.
[18] A. Pakes, A limit theorem for the maxima of the para-critical simple branching process, Adv. in Appl. Probab. 30 (1998) 740–756.
[19] A. Spătaru, A maximum sequence in a critical multitype branching process, J. Appl. Probab. 28 (4) (1991) 893–897.
[20] K. Etessami, M. Yannakakis, Recursive Markov chains, stochastic grammars, and monotone systems of nonlinear equations, J. ACM 56 (1) (2009) 1–66.
[21] J. Esparza, A. Kučera, R. Mayr, Model checking probabilistic pushdown automata, in: LICS 2004, IEEE Computer Society, 2004, pp. 12–21.
[22] T. Brázdil, J. Esparza, S. Kiefer, On the memory consumption of probabilistic pushdown automata, in: Proceedings of FSTTCS, 2009, pp. 49–60.
[23] M.Z. Kwiatkowska, G. Norman, D. Parker, Prism: Probabilistic symbolic model checker, in: T. Field, P.G. Harrison, J.T. Bradley, U. Harder (Eds.), Computer Performance Evaluation/TOOLS, in: Lecture Notes in Comput. Sci., vol. 2324, Springer, 2002, pp. 200–204.
[24] C. Baier, B.R. Haverkort, H. Hermanns, J.-P. Katoen, Performance evaluation and model checking join forces, ACM Commun. Comput. Algebra 53 (9) (2010) 76–85.
[25] J. Esparza, A. Gaiser, S. Kiefer, Computing least fixed points of probabilistic systems of polynomials, in: Proceedings of STACS, 2010, pp. 359–370.
[26] R. Horn, C. Johnson, Matrix Analysis, Cambridge University Press, 1985.
[27] A. Berman, R. Plemmons, Nonnegative Matrices in the Mathematical Sciences, Academic Press, 1979.
[28] J. Esparza, S. Kiefer, M. Luttenberger, On fixed point equations over commutative semirings, in: STACS'07, in: Lecture Notes in Comput. Sci., vol. 4397, Springer, 2007, pp. 296–307.
[29] J. Esparza, S. Kiefer, M. Luttenberger, An extension of Newton's method to ω-continuous semirings, in: DLT'07, in: Lecture Notes in Comput. Sci., vol. 4588, Springer, 2007, pp. 157–168.
[30] L. Blum, F. Cucker, M. Shub, S. Smale, Complexity and Real Computation, Springer-Verlag, 1998.
[31] J. Ortega, W. Rheinboldt, Iterative Solution of Nonlinear Equations in Several Variables, Academic Press, 1970.
[32] W. Feller, An Introduction to Probability Theory and Its Applications, vol. I, John Wiley & Sons, 1968.
[33] F. Spitzer, Principles of Random Walk, Springer, 1976.
[34] D. Williams, Probability with Martingales, Cambridge University Press, 1995.
[35] L. Kleinrock, Queueing Systems: Theory, Wiley–Interscience, 1975.
[36] R. Fagin, A. Karlin, J. Kleinberg, P. Raghavan, S. Rajagopalan, R. Rubinfeld, M. Sudan, A. Tomkins, Random walks with "back buttons", Ann. Appl. Probab. 11 (3) (2001) 810–862.
[37] M. Grötschel, L. Lovász, A. Schrijver, Geometric Algorithms and Combinatorial Optimization, second ed., Springer, 1993.
[38] M.L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming, John Wiley & Sons, 2005.