Bisimulation by Partitioning is $\Omega((m+n)\log n)$

Jan Friso Groote
Eindhoven University of Technology, The Netherlands

Jan Martens
Eindhoven University of Technology, The Netherlands

Erik de Vink
Eindhoven University of Technology, The Netherlands

Abstract

An asymptotic lowerbound of $\Omega((m+n)\log n)$ is established for partition refinement algorithms that decide bisimilarity on labeled transition systems. The lowerbound is obtained by subsequently analysing two families of deterministic transition systems — one with a growing action set and another with a fixed action set.

For deterministic transition systems with a one-letter action set, bisimilarity can be decided with fundamentally different techniques than partition refinement. In particular, Paige, Tarjan, and Bonic give a linear algorithm for this specific situation. We show, exploiting the concept of an oracle, that the approach of Paige, Tarjan, and Bonic is not of help to develop a generic algorithm for deciding bisimilarity on labeled transition systems that is faster than the established lowerbound of $\Omega((m+n)\log n)$.

2012 ACM Subject Classification Theory of computation; Theory of computation → Interactive computation; Theory of computation → Design and analysis of algorithms

Keywords and phrases Bisimimilarity, partition refinement, labeled transition system, lowerbound

Digital Object Identifier 10.4230/LIPIcs...

Funding *Jan Martens*: AVVA project NWO 612.001.751/TOP1.17.002

1 Introduction

Strong bisimulation [16, 13] is the gold standard for equivalence on labeled transition systems (LTSs). Deciding bisimulation equivalence among the states of an LTS is a crucial step for tool-supported analysis and model checking of LTSs. The well-known and widely-used partition refinement algorithm of Paige and Tarjan [14] has a worst-case upperbound $\mathrm{O}(m\log n)$ for establishing the bisimulation equivalence classes. Here, m is the number of transitions and n is the number of states in an LTS. The algorithm of Paige and Tarjan seeks to find, starting from an initial partition, via refinement steps, the coarsest stable partition, that in fact is built from the bisimulation equivalence classes that are looked for. The algorithm achieves the complexity of the logarithm of the number of states n by restricting the amount of work for refining blocks and moving states. Refining blocks is carried out by only investigating the smaller splitting blocks, using an intricate bookkeeping trick. Only the smaller parts of a block that are to be moved to a new block are split off, leaving the bulk of the original block at its place. These specific ideas go back to [8] and make the difference with the earlier $\mathrm{O}(mn)$ algorithm of Kanellakis and Smolka [11].

The Paige-Tarjan algorithm, with its format of successive refinements of an initial partition till a fixpoint is reached, has been leading for variations and generalizations for deciding specific forms of (strong) bisimilarities, see e.g. [4, 6, 7, 18, 10]. We are interested in the question whether the Paige-Tarjan algorithm is computationally optimal. A result is provided by Berkholz et al. in a paper [2] that studies stable colourings of (coloured) graphs. More specifically, they show that for an undirected graph with n nodes and m edges, canonical coarsest bi-stable colouring is in $\Omega((m+n)\log n)$. Translated to LTSs, the result of [2] builds on the assumption that LTSs are essentially non-deterministic, i.e., every state has multiple outgoing transitions for the same label. A first contribution of the present paper is a lowerbound of the class of partition refinement algorithms

for deciding bisimilarity of deterministic LTSs. We define what a partition refinement algorithm is and articulate the complexity in terms of the number of states that are moved. Then, a particular family of (deterministic) LTSs, called bisplitters, is shown to require $n \log n$ work. This strengthens the result of [2], actually answering an open question in it.

We obtain our lowerbound results assuming that algorithms use partition refinement. However, one may wonder if a different approach than partition refinement can lead to a faster decision procedure for bisimulation. For the specific case of deterministic LTSs with a singleton action set and a state labelling, Robert Paige, Robert Tarjan and Robert Bonic propose a linear algorithm [15], which we will refer to as Roberts' algorithm. In [5] it is proven that partition refinement à la Hopcroft has a lowerbound of $\Omega(n \log n)$ in this case. Concretely, this means that Roberts' algorithm achieves the essentially better performance by using a completely different technique than partition refinement to determine the bisimulation equivalence classes.

Crucial for Roberts' algorithm is the ability to identify, in linear time, the bisimilarity classes of cycles. In this paper we show that if the alphabet consists of at least two actions a rapid decision on 'cycles' as in [15] will not be of help to improve on the Paige-Tarjan algorithm for general LTSs. We argue that the specialty in the algorithm of [15], viz. to be able to quickly decide the bisimilarity of the states on cycles, can be captured by means of a stronger notion, namely an oracle, that provides the bisimulation classes of the states of a so-called 'end structure', the counterpart in the multiple action setting of a cycle in the single action setting. The oracle can be consulted to refine the initial partition with respect to the bisimilarity on the end structures of the LTS for free. We show that for the class of partition refinement algorithms enhanced with such an oracle, thus encompassing the algorithm of [15], the $n \log n$ lowerbound persists for non-degenerate action sets.

The family of $n \log n$-hard LTSs we use to establish the lowerbound, involve an action set of $\log n$ actions. Building on the two results already mentioned, and exploiting ideas borrowed from [15] to extend the bisimulation classes for the states in the end structures, i.e. cycles, to the states of the complete LTS, we provide another family of (deterministic) LTSs that have only two actions. Led by these LTSs we argue that for the two-action case the complexity of deciding bisimulation is $\Omega((m+n)\log n)$, whether we use an oracle or not.

The document is structured as follows. In Section 2 we give the necessary preliminaries on the problem. A recap of the linear algorithm of [15] is provided in Section 3. Next, we introduce the family of deterministic LTSs $\mathcal{B}_k$ for which we show in Section 4 that deciding bisimilarity is $\Omega(n \log n)$ for the class of partition refinement algorithms and for which we establish in Section 5 an $\Omega(n \log n)$ lowerbound for the class of partition refinement algorithms enhanced with an oracle for end structures. In Section 6 we introduce the family of deterministic LTSs $\mathcal{C}_k$, each involving two actions only, to take the number of transitions m into account and establish an $\Omega((m+n)\log n)$ lowerbound for partition refinement with and without oracle for end structures. We wrap up with concluding remarks.

2 Preliminaries

Given a set of states S, a *partition* of S is a set of sets of states $\pi \subseteq 2^S$ such that for all $B, B' \in \pi$ it holds that $B \neq \emptyset$, $B \cap B' = \emptyset$, and $\bigcup_{B \in \pi} B = S$. The elements of a partition are refered to as blocks. A partition π of S induces an equivalence relation $=_\pi \subseteq S \times S$, where for two states $s, t \in S$, $s =_\pi t$ iff the states are in the same block, i.e. there is a block $B \in \pi$ such that $s, t \in B$. A partition π of S is a *refinement* of a partition π' of S iff for every block $B \in \pi$ there is a block $B' \in \pi'$ such that $B \subseteq B'$. It follows that each block of π' is the union of blocks of π. The refinement is *strict* if $\pi \neq \pi'$. The common refinement of two partitions π and π' is the partition with blocks $\{ B \cap B' \mid B \in \pi,\ B' \in \pi' \}$. A sequence of partitions $\pi_0, \ldots, \pi_n$ is called a refinement sequence iff

π_{i+1} is a refinement of π_i, for all $0 \leqslant i < n$.

▶ **Definition 1.** *A labeled transition system with initial partition (LTS) $L = (S, \mathcal{A}, \rightarrow, \pi_0)$ is given by a finite set of states S, a finite alphabet of actions $\mathcal{A}$, a transition relation $\rightarrow \subseteq S \times \mathcal{A} \times S$, and a partition π_0 of S. A labeled transition system with initial partition is called deterministic (dLTS) if the transition relation is a total function $S \times \mathcal{A} \rightarrow S$.*

Note that we omit an initial state, as it is not relevant in this article. Note also that in the presence of an initial partition, an LTS with one action label represents a Kripke structure. For a dLTS with a set of states S and the initial partition $\pi_0 = \{S\}$ we have that π_0 itself already represents bisimilarity, contrary to LTSs in general.

Given an LTS $L = (S, \mathcal{A}, \rightarrow, \pi_0)$, states $s, t \in S$, and an action $a \in \mathcal{A}$, we write $s \xrightarrow{a} t$ instead of $(s, a, t) \in \rightarrow$. For dLTSs we occasionally write $L(s, a)$ for t, i.e., t is the image of the pair (s, a) of the function $\rightarrow$. We say that s *reaches* t via a iff $s \xrightarrow{a} t$. A state s reaches a set $U \subseteq S$ iff there is a state in U that is reached by s. A set of states $V \subseteq S$ is called *stable* under a set of states $U \subseteq S$ iff for all actions a either all states in V reach U via a, or no state in V reaches U via a. A partition π is stable under a set of states U iff each block $B \in \pi$ is stable under U. A partition π is called stable iff it is stable under all its blocks.

Following [16, 13], for an LTS L a symmetric relation $R \subseteq S \times S$ is called a bisimulation relation iff for all $(s, t) \in R$ and $a \in \mathcal{A}$, we have that $s \xrightarrow{a} s'$ for some $s' \in S$ implies that $t \xrightarrow{a} t'$ for some $t' \in S$ such that $(s', t') \in R$. In the setting of the present paper, as we incorporate the initial partition in the definition of an LTS, bisimilarity is slightly non-standard. For a bisimulation relation R, we additionally require that it respects the initial partition π_0 of L, i.e. $(s, t) \in R$ implies $s =_{\pi_0} t$. Two states $s, t \in S$ are called (strongly) bisimilar for L iff a bisimulation relation R exists with $(s, t) \in R$, notation $s \leftrightarrows_L t$. Bisimilarity is an equivalence relation on the set of states of L. We write $[s]_L^{\leftrightarrows}$ for the bisimulation equivalence class of the state s in L.

Partition refinement algorithms for deciding bisimilarity on LTSs start with an initial partition π_0, which is then repeatedly refined until a stable partition is reached. This stable partition is then the coarsest stable partition of the LTS refining π_0 and coincides with bisimilarity [11, 14].

We define that an algorithm is a partition refinement algorithm if it constructs a valid sequence of partitions. Concretely this means that a state s in a block B is only assigned to another block if there is reason to do so; there is a splitter block B' in the current partition to which s has a transition and some other state in B does not, or the other way around. Thus, the block B is not stable under the block B'. Moreover, states that cannot be distinguished based on the current partition, i.e. there is no such block B' such that only one of the states reaches B', cannot be a part of different blocks. (I think obsolete) Finally, we insist that each subsequent partition reflects some progress, i.e., π_{i+1} is a strict refinement of π_i. This leads to the following notion of a valid refinement and a valid partition sequence.

▶ **Definition 2.** *Let $L = (S, \mathcal{A}, \rightarrow, \pi_0)$ be an LTS, and π a partition of S. We call a refinement π' of π a* valid refinement *with respect to L, if the following criteria hold.*

(a) π' is a strict refinement of π;
(b) if $s \neq_{\pi'} t$ for $s, t \in S$, then (i) $s \neq_\pi t$ or (ii) $s' \in S$ exist such that for some $a \in A$, $s \xrightarrow{a} s'$ and for all $t' \in S$ such that $t \xrightarrow{a} t'$, we have $s' \neq_\pi t'$ or vice versa.

A sequence of partitions $\Pi = (\pi_0, \ldots, \pi_n)$ is called valid iff every successive partition π_i, for $0 < i \leqslant n$, is a valid refinement of π_{i-1}, and, moreover, the partition π_n is stable.

When a partition π is refined into a partition π', states that are in the same block but can reach different blocks can lead to a split of the block into smaller ones, each holding one of the states. This means

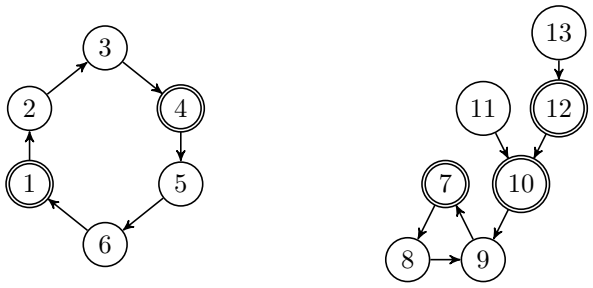

Figure 1 dLTS with one action label (not shown) and initial partition distinguishing states $1, 4, 7, 10, 12$ from states $2, 3, 5, 6, 8, 9, 11, 13$.

that a block $B \in \pi$ is split into k blocks $B_1, \ldots, B_k \in \pi'$. The least amount of work is done for this operation, by creating new blocks for the least number of states possible. Thus, $B \in \pi$ is transformed into $B_1 \in \pi'$, say, the biggest block among $B_1, \ldots, B_k$. Therefore, the so-called refinement cost rc of the refinement π' of π is given by

$$rc(\pi, \pi') = \textstyle\sum_{B\in\pi} |B| - \max_{B'\in\pi' :\, B'\subseteq B} |B'| \,.$$

For a sequence of refinements $\Pi = (\pi_0, \ldots, \pi_n)$ we write $rc(\Pi)$ for $\sum_{i=1}^{n} rc(\pi_{i-1}, \pi_i)$. For an LTS L, we write $rc(L)$ for $\min\{\, rc(\Pi) \mid \Pi \text{ a valid refinement sequence for } L \,\}$. Note that this complexity measure is more general than the one used in [2]. In that complexity measure the total number of transitions between the splitting block and the witnesses block thereof is counted, which is always greater or equal than our refinement costs.this sentence

We characterise the states of LTSs by sequences of bits. The set of bits is denoted as $\mathbb{B} = \{\mathtt{0}, \mathtt{1}\}$. Bit sequences of length up to and including k are written as $\mathbb{B}^{\leqslant k}$. The inverse of a bit b is denoted by $\overline{b}$. Thus $\overline{\mathtt{0}} = \mathtt{1}$ and $\overline{\mathtt{1}} = \mathtt{0}$. For two bit sequences σ, σ', we write $\sigma \preccurlyeq \sigma'$ to indicate that σ is a prefix of σ' and write $\sigma \prec \sigma'$ if σ is a strict prefix of σ'. For a bit sequence $\sigma \in \mathbb{B}^k$, for any $i, j \leqslant k$ we write $\sigma[i]$ to indicate the bit at position i starting from position 1. We write $\sigma[i{:}j] = \sigma[i]\sigma[i{+}1]\cdots\sigma[j]$ to indicate the subword from position i to position j.

3 Roberts' algorithm

Most algorithms to determine bisimulation on an LTS use partition refinement. However, there is one notable exception. On the class of dLTSs with a singleton action alphabet, deciding the coarsest stable partition, i.e. bisimilarity, requires linear time only. This is due to the algorithm of Robert Paige, Robert Tarjan, and Robert Bonic [15], which we therefore aptly call Roberts' algorithm.

The algorithm exploits the structure of dLTSs with one label, of which examples are depicted in Figure 1 where the initial partition is indicated by single/double circles around the states. For each state in such a dLTS we can assign a unique cycle (also referred to as 'end structure' in the sequel), and that state is either on this cycle or following outgoing edges lead to that (unique) cycle. Below, we roughly sketch how Roberts' algorithm works. See [15] for details.

1. Build lassos and mark states to identify the cycles of the dLTS.
2. For each state there is exactly one word of specific bounded length of labels of the initial partitions of the states on the (unique) run from this state.this sentence Identify for each cycle the states with the lexicographic least such word. This can be done in linear time in the size of the cycle. If there are bisimilar states on the cycle, then the algorithm will identify them. States on different cycles can only be bisimilar if the number of non-bisimilar states on these cycles is the same. By

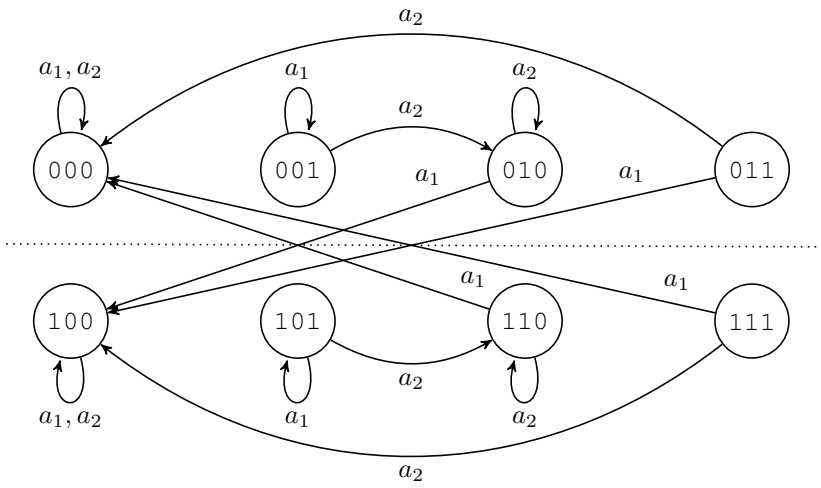

Figure 2 Bisplitter $\mathcal{B}_3$ with initial partition $\{\{000, 001, 010, 011\}, \{100, 101, 110, 111\}\}$.

comparing cycles with the same number of bisimulation equivalence classes, starting with the lexicographic least state, it is then determined linearly for all states on final cycles whether they are bisimilar.

3. By a backward calculation along the paths leading to the cycles the bisimilarity equivalence classes for the states not on cycles can then be determined in linear time as well.

A striking observation is that any partition refinement algorithm, using a valid sequence of partitions, requires a refinement cost of $\Omega(n \log n)$ to calculate which states are bisimilar for the dLTSs to which Roberts' algorithm applies. This follows from results in [3, 5] where it is shown that Hopcroft's algorithm [8] cannot have a better running time than $\Omega(n \log n)$. Below we come back to this observation, showing that the ideas in Roberts' algorithm cannot be exploited to come up with a linear algorithm for bisimulation if the LTS is either nondeterministic, or has more than one action label.

4 $\mathcal{B}_k$ is $\Omega(n \log n)$ for partition refinement

In this section we introduce a family of deterministic LTSs called bisplitters on which the cost of any partition refinement algorithm is $\Omega(n \log n)$ where n is the number of states. With some modification we obtain in Section 6 a family of LTSs that has the bound $\Omega((n + m) \log n)$ where m is the number of transitions.

▶ **Definition 3.** *For $k > 1$, the bisplitter $\mathcal{B}_k = (S, \mathcal{A}_k, \rightarrow, \pi_0)$ is defined as the dLTS that has the set $S = \{\, \sigma \mid \sigma \in \mathbb{B}^k \,\}$ as its set of states, the set $\mathcal{A}_k = \{a_1, \ldots, a_{k-1}\}$ as its set of actions, the relation*

$$\{\, \sigma \xrightarrow{a_i} \sigma \mid \sigma \in S,\ 1 \leqslant i < k \colon \sigma[i{+}1] = 0 \,\} \cup$$
$$\{\, \sigma \xrightarrow{a_i} \sigma[1{:}i{-}1]\,\overline{\sigma[i]}\,0^{k-i} \mid \sigma \in S,\ 1 \leqslant i < k \colon \sigma[i{+}1] = 1 \,\}$$

as its transition function, and the set $\pi_0 = \{\, \{\, \sigma \in S \mid \sigma[1] = 0 \,\}, \{\, \sigma \in S \mid \sigma[1] = 1 \,\} \,\}$ as its initial partition.

Thus the bisplitter $\mathcal{B}_k$ has 2^k states, viz. the bitstrings of length k, and $\mathcal{B}_k$ has $k{-}1$ action labels. It has $(k{-}1)2^k$ transitions: (i) a self-loop for bitstring σ with label a_i if the $i{+}1$-th bit of σ equals 0;

(ii) otherwise, i.e. when $i{+}1$-th bit of σ equals $\mathtt{1}$, the bitstring σ has for label a_i a transition to the bitstring that equals the first $i{-}1$ bits of σ, flips the i-th bit of σ, and has $k{-}i$-many $\mathtt{0}$'s following. The initial partition π_0 distinguishes the bitstrings starting with $\mathtt{0}$ from those starting with $\mathtt{1}$. A drawing of bisplitter $\mathcal{B}_3$ is given in Figure 2. We see, e.g., for the bitstring $\sigma = \mathtt{101}$ an a_1-transition to itself, as $\sigma[2] = 0$, and an a_2-transition to $\mathtt{110}$, as $\sigma[3] = 1$.

▶ **Definition 4.** *For any string $\sigma \in \mathbb{B}^{\leqslant k}$, we define the prefix block B_σ of $\mathcal{B}_k$ to be the block $B_\sigma = \{\, \sigma' \in \mathbb{B}^k \mid \sigma \preccurlyeq \sigma' \,\}$.*

The following lemma collects a number of results related to prefix blocks.

▶ **Lemma 5.** *Let $k \geqslant 2$ and let the dLTS $\mathcal{B}_k = (\mathbb{B}^k, \mathcal{A}_k, \rightarrow, \pi_0)$ be the k-th bisplitter. Let the sequence $\Pi = (\pi_0, \ldots, \pi_n)$ be a valid refinement sequence. Then it holds that*

(a) *Every partition π_i of Π contains prefix blocks only.*

(b) *If partition π_i of Π contains a prefix block B_σ with $|\sigma| < k$, then π_i is not stable.*

(c) *If B_σ is in π_i, for $0 \leqslant i < n$, then either $B_\sigma \in \pi_{i+1}$, or $B_{\sigma 1} \in \pi_{i+1}$ and $B_{\sigma 0} \in \pi_{i+1}$.*

Proof (a). Initially, for $\pi_0 = \{B_0, B_1\}$ both blocks are prefix blocks by definition. We prove, if partition π_i, for $0 \leqslant i < n$, has only prefix blocks then all blocks in π_{i+1} are prefix blocks as well.

Assume, to arrive at a contradiction, that there is a block $B \in \pi_{i+1}$ that is not a prefix block. Because π_{i+1} is a refinement of π_i, we have $B \subseteq B_\sigma$ for some prefix block $B_\sigma \in \pi_i$. This means that σ is a common prefix of all elements of B. We can choose θ such that $\sigma\theta$ is the longest common prefix of all elements of B. Since every singleton of $\mathbb{B}^k$ is a prefix block, B is not a singleton. This means that $|\sigma\theta| < k$, and that there are some elements σ_1 and σ_2 of B such that $\sigma\theta 0$ is a prefix of σ_1 and $\sigma\theta 1$ is a prefix of σ_2. Because B is not a prefix block, there must exist at least one $\tau \in \mathbb{B}^k$ with a prefix $\sigma\theta$ such that $\tau \notin B$. Obviously, we have either (i) $\sigma\theta 0$ is a prefix of τ, or (ii) $\sigma\theta 1$ is a prefix of τ. We will show that in both these cases τ in fact belongs to B in π_{i+1}, which is a contradiction. This also means that for any $B \in \pi_{i+1}$, where $B \subseteq B_\sigma$ for some prefix block $B_\sigma \in \pi_i$, we have that B is a prefix block for the prefix $\sigma\theta$ (i.e. $B = B_{\sigma\theta}$) where $\sigma\theta$ is the longest common prefix of all elements of B.

(i) Suppose $\sigma\theta 0$ is a prefix of τ. We will show that τ and σ_1 belong to the same block in π_{i+1} because for each a_j (where $1 \leqslant j < k$) the states σ_1' and τ', such that $\sigma_1 \xrightarrow{a_j} \sigma_1'$ and $\tau \xrightarrow{a_j} \tau'$, belong to the same block in π_i. There are three cases:

- $j < |\sigma\theta|$: Since $\sigma\theta$ is a prefix of both σ_1 and τ, we have $\sigma_1[j+1] = \tau[j+1]$.
  - If $\sigma_1[j+1] = \tau[j+1] = 0$, then $\sigma_1' = \sigma_1$ and $\tau' = \tau$. Obviously, both σ_1' and τ' belong to B_σ (since σ_1 and τ belong to B_σ).
  - If $\sigma_1[j+1] = \tau[j+1] = 1$, then both σ_1' and τ' are of the form $\rho[1:j-1]\overline{\rho[j]}0^{k-j}$ where $\rho = \sigma\theta$, and we have $\sigma_1' = \tau'$, so they clearly belong to the same block of π_i.
- $j = |\sigma\theta|$: Since $\sigma_1[j+1] = \tau[j+1] = 0$, we have $\sigma_1' = \sigma_1$ and $\tau' = \tau$, and obviously both σ_1' and τ' belong to B_σ (since σ_1 and τ belong to B_σ).
- $j > |\sigma\theta|$: In fact, for arbitrary ρ, performing a_j with $j > |\sigma\theta|$ in $\sigma\theta\rho$ leads to $\sigma\theta\rho'$ (for both cases, where $(\sigma\theta\rho)[j+1]$ is 0 or 1). In particular this means that if $j > |\sigma\theta|$ and $\sigma_1 \xrightarrow{a_j} \sigma_1'$ and $\tau \xrightarrow{a_j} \tau'$, then $\sigma\theta$ is a prefix of both σ_1' and τ', and σ_1' and τ' belong to B_σ in π_i.

(ii) Now, suppose $\sigma\theta 1$ is a prefix of τ. We will show that τ and σ_2 belong to the same block in π_{i+1} because for each a_j (where $1 \leqslant j < k$) the states σ_2' and τ', such that $\sigma_2 \xrightarrow{a_j} \sigma_2'$ and $\tau \xrightarrow{a_j} \tau'$, belong to the same block in π_i. There are three cases:

- $j < |\sigma\theta|$: Similar as in (i).
- $j = |\sigma\theta|$: Since $\sigma_1[j+1] = \tau[j+1] = 1$, we have $\sigma_1' = \tau' = \rho[1:j-1]\overline{\rho[j]}0^{k-1}$ where $\rho = \sigma\theta$, so clearly σ_1' and τ' are in a same block in π_i.

- $j > |\sigma\theta|$: Similar as in (i).

◀

Proof (b). Suppose $B_\sigma \in \pi_i$ and $|\sigma| = \ell < k$. Let $\theta \in \mathbb{B}^*$ be such that $\sigma_1 = \sigma 0\theta$ and $\sigma_2 = \sigma 1\theta$. Then we have $\sigma_1 \xrightarrow{a_\ell} \sigma_1 \in B_\sigma$ and $\sigma_2 \xrightarrow{a_\ell} \sigma[1{:}\ell{-}1]\overline{\sigma[\ell]}\,0^{k-\ell} \notin B_\sigma$. Thus B_σ isn't stable, and hence π_i isn't either. ◀

Proof (c). We show that for a prefix block $B_\sigma \in \pi_i$, a bit $b \in \mathbb{B}$ and all $\theta, \theta' \in \mathbb{B}^{k-(|\sigma|+1)}$ the states $\sigma_1 = \sigma b\theta$ and $\sigma_2 = \sigma b\theta'$ are not split by any action a_j, for $1 \leqslant j < k$, and thus are in the same block of π_{i+1}. Pick j, $1 \leqslant j < k$, and suppose $\sigma_1 \xrightarrow{a_j} \sigma_1'$, $\sigma_2 \xrightarrow{a_j} \sigma_2'$, i.e., $\sigma_1' = \mathcal{B}_k(\sigma_1, a_j)$ and $\sigma_2' = \mathcal{B}_k(\sigma_2, a_j)$. If $j \leqslant |\sigma|$ and $\sigma[j] = 0$ then $\sigma_1' = \sigma_1$ and $\sigma_2' = \sigma_2$ hence both $\sigma_1', \sigma_2' \in B_\sigma$ don't split for a_j. If $j \leqslant |\sigma|$ and $\sigma[j] = 1$ then $\sigma_1' = \sigma_2'$ and don't split for a_j either. If $j > |\sigma|$ then both $\sigma_1', \sigma_2' \in B_\sigma$ and don't split for a_j either. ◀

With the help of the above lemma, clarifying the form of the partitions in a valid refinement sequence for the bisplitter family, we are able to obtain a lowerbound for any partition refinement algorithm acting on it.

▶ **Theorem 6.** *For any $k > 1$, application of partition refinement to the bisplitter $\mathcal{B}_k$ has refinement costs $rc(\mathcal{B}_k) \in \Omega(n \log n)$ where $n = 2^k$ is the number of states of $\mathcal{B}_k$.*

Proof. Let $\Pi = \pi_0, \ldots, \pi_n$ be a valid refinement sequence for $\mathcal{B}_k$. By items a and b of Lemma 5, we have $\pi_n = \{\,\{s\} \mid s \in \mathbb{B}^k\,\}$ since π_n is stable. Item c of Lemma 5 implies that in every refinement step (π_i, π_{i+1}) a block is kept or it is refined in two blocks of equal size. The cost of refining the block B_σ, for $|\sigma| < k$, into $B_{\sigma 0}$ and $B_{\sigma 1}$ is the number of states in $B_{\sigma 0}$ or $B_{\sigma 1}$, which are the same and equal to $\frac{1}{2}2^{k-|\sigma|}$. Therefore, we have $rc(\mathcal{B}_k, \Pi) = \sum_{\ell=1}^{k-1} 2^\ell \frac{1}{2}2^{k-\ell} = \sum_{\ell=1}^{k-1} \frac{1}{2}2^k = (k-1)2^{k-1}$. If n is the number of states of $\mathcal{B}_k$, it holds that $n = 2^k$, thus $k - 1 = \log \frac{1}{2}n$. Hence, $rc(\mathcal{B}_k, \Pi) = \frac{1}{2}n \log \frac{1}{2}n$ which is in $\Omega(n \log n)$. ◀

5 $\mathcal{B}_k$ is $\Omega(n \log n)$ for partition refinement with an oracle

One may wonder whether the approach of calculating bisimulation equivalence classes will work on transition systems with non-degenerate action sets as well as the Roberts' algorithm guarantees a linear performance for the degenerate case. In order to capture the approach of [15], we augment the class of partition refinement algorithms with an oracle. At the start of the algorithm the oracle can be consulted to identify the bisimulation classes for designated states, viz. for those that are in an 'end structure', the counterpart of the cycles in [15]. This results in a refinement of the initial partition; partition refinement then starts from the updated partition.

Thus, we can ask the oracle to provide the bisimulation classes of all elements in an end structure of the LTS at hand. This yields a new partition, viz. the common refinement of the initial partition, on the one hand, and the partition induced by the bisimulation equivalence classes as given by the oracle and the complement of their union, on the other hand. The work that remains to be done is establishing the bisimulation equivalence classes, with respect to the initial partition, for the states not in any end structure. We will establish that a partition refinement algorithm strengthened with such an oracle will not improve upon partition refinement.

We first define the notion of an end structure of an LTS and the associated notion of an end structure partition.

▶ **Definition 7.** *Given an LTS $L = (S, \mathcal{A}, \to, \pi_0)$, a non-empty subset $S' \subseteq S$ is called an* end structure *of L, if S' is a minimimal set of states closed under all transitions. Moreover, $es(L) = \{\, S' \subseteq S \mid S'$ end structure of $L \,\}$ and $\pi_{es} = \{\, [s]_L^{\leftrightarrow} \mid s \in \bigcup es(L) \,\} \cup \{S \backslash \{\, [s]_L^{\leftrightarrow} \mid s \in \bigcup es(L) \,\}\} \setminus \{\emptyset\}$ is called the end structure partition of L.*

Like the cycles of [15], an LTS can have multiple end structures. The end structure partition π_{es} consists of the bisimilarity equivalence classes of L containing a state of an end structure, completed with a block of the remaining states that are not in an end structure, and not bisimilar to any state in an end structure (if not empty).

▶ **Lemma 8.** *Let $L = (S, \mathcal{A}, \to, \pi_0)$ be a dLTS.*
(a) *If $|\mathcal{A}| = 1$ then $es(L)$ consists of all cycles in L.*
(b) *Every $s \in S$ has a path to an end structure of L.*

Proof. (a) Since an end structure S' is closed under transitions, S' is a lasso. Because S' is minimal and non-empty, it follows that S' is a cycle.

(b) Let $U = \{\, t \in S \mid s \xrightarrow{w}{}^{*} t,\ w \in \mathcal{A}^* \,\}$ be the set of states reachable from state s. Then U is closed under all transitions. The minimal subset $U' \subseteq U$ which is still closed under all transitions is an end structure of L and reachable by s. ◀

Next we enhance the notion of a partition refinement algorithm. An oracle can be consulted for the states in the end structures. In this approach, the initial partition is replaced by a partition in which all bisimilarity equivalence classes of states in end structures are separated split off from the original blocks.

▶ **Definition 9.** *A partition refinement algorithm with end structure oracle yields for an LTS $L = (S, \mathcal{A}, \to, \pi_0)$ a valid refinement sequence $\Pi = (\pi_0', \pi_1, \ldots, \pi_n)$ where π_0' is the common refinement of the initial partition π_0 and the end structure partition π_{es} of L. The partition π_0' is called the updated initial partition of L.*

As Roberts' algorithm shows, in the case of a singleton action set the availability of an end structure oracle yields the asymptotic performance of a linear algorithm. In the remainder of this section we confirm that in the case of more letters the end structure does not help either. The next lemma states that the amount of work required for the dLTS $\mathcal{B}_k$ by a partition refinement algorithm enhanced with an oracle dealing with end structures is at least the amount of work needed by a partition refinement algorithm without oracle for the dLTS $\mathcal{B}_{k-2}$.

▶ **Lemma 10.** *For the bisplitter $\mathcal{B}_k = (S, \mathcal{A}, \to, \pi_0)$, for some $k > 2$, let π_0' be the updated initial partition. Then every valid refinement sequence $\Pi = (\pi_0', \pi_2, \ldots, \pi_n)$ for the updated bisplitter $\mathcal{B}_k' = (S, \mathcal{A}, \to, \pi_0')$ satisfies $rc(\Pi) \geqslant rc(\mathcal{B}_{k-2})$.*

Proof. Observe that there are only two end structures in $\mathcal{B}_k$, viz. the singletons of the two states with $\mathtt{0}^k$ and $\mathtt{1}\mathtt{0}^{k-1}$. Since all other states can reach $\mathtt{0}^k$ or $\mathtt{1}\mathtt{0}^{k-1}$, these states are not in an end structure: Choose $\sigma \in \mathbb{B}^k$, $\sigma \neq \mathtt{0}^k, \mathtt{1}\mathtt{0}^{k-1}$. Then σ is of the form $b\mathtt{0}^j\mathtt{1}\theta$ for some $b \in \mathbb{B}$, $j \geqslant 0$ and $\theta \in \mathbb{B}^*$. For $j = 0$ we have $\sigma \xrightarrow{a_1} \overline{b}\mathtt{0}^{k-1}$ which is either $\mathtt{0}^k$ or $\mathtt{1}\mathtt{0}^{k-1}$; for $j > 0$ we have $\sigma \xrightarrow{a_{j+1}} b\mathtt{0}^{j-1}\mathtt{1}\mathtt{0}^{k-(j+1)}$ while $b\mathtt{0}^{j-1}\mathtt{1}\mathtt{0}^{k-(j+1)}$ reaches $\mathtt{0}^k$ or $\mathtt{1}\mathtt{0}^{k-1}$ by induction.

By Lemma 5, every state $\sigma \in \mathbb{B}^k$ of $\mathcal{B}_k$ is in its own bisimulation equivalence class $\{\sigma\}$. It follows that the updated initial partition π_0' equals $\{\, \{\mathtt{0}^k\}, \{\mathtt{1}\mathtt{0}^{k-1}\}, B_0 \backslash \{\mathtt{0}^k\}, B_1 \backslash \{\mathtt{1}\mathtt{0}^{k-1}\} \,\}$. We claim that if a sequence $\Pi = (\pi_0', \pi_1, \ldots, \pi_n)$ for $\mathcal{B}_k$ exists with costs $rc(\Pi) \leqslant rc(\mathcal{B}_{k-2})$, then also a valid refinement sequence Π' for $\mathcal{B}_{k-2}$ exists with costs smaller than $rc(\mathcal{B}_{k-2})$ which yields a contradiction, since, by definition, $rc(B_{k-2})$ are the minimum costs over all valid refinement sequence for $\mathcal{B}_{k-2}$.

So, assume $\Pi = (\pi'_0, \pi_1, \ldots \pi_n)$ is a valid refinement sequence for $\mathcal{B}_k$ and $rc(\Pi) \leqslant rc(\mathcal{B}_{k-2})$. We obtain a valid refinement sequence Π' for $\mathcal{B}_{k-2}$ in two steps. First, we use the projection function p from partitions on $\mathbb{B}^k$ to partitions on $\mathbb{B}_{k-2}$ that removes the prefix $\mathtt{11}$ from strings in a block (or ignores the block if such string is absent), i.e. $p(\pi) = \{\, \{\, \sigma \mid \mathtt{11}\sigma \in B \,\} \mid B \in \pi \,\}\backslash\{\emptyset\}$. In particular, $p(\pi_0) = \{\{\, \sigma \mid \sigma \in \mathbb{B}^{k-2} \,\}\}$. Second, the function P from refinement sequences of $\mathcal{B}_k$ to refinement sequences of $\mathcal{B}_{k-2}$, removes, in addition to application of p to each constituent partition, duplicate partitions from the sequence. Then $\Pi' = P(\Pi)$, say $\Pi' = (\varrho_0, \varrho_1, \ldots, \varrho_\ell)$.

We have $\varrho_0 = p(\pi'_0) = \{B_\varepsilon\}$. Next we observe that $\varrho_1 = \pi_0^{k-2} = \{B_0, B_1\}$ the initial partition of $\mathcal{B}_{k-2}$, containing the prefix blocks of $\mathtt{0}$ and $\mathtt{1}$: Take any two different states $b\theta, b\theta' \in \mathbb{B}^{k-2}$, for a bit $b \in \mathbb{B}$ and strings $\theta, \theta' \in \mathbb{B}^{k-3}$ that are not in the same block of ϱ_1. Let i, $0 \leqslant i < n$ be such that $p(\pi_i) = \varrho_0$ and $p(\pi_{i+1}) = \varrho_1$. Then $\mathtt{11}b\theta$ and $\mathtt{11}b\theta'$ have been separated when refining π_i into π_{i+1}. But no action a_j witnesses such a split: (i) $\mathcal{B}_k(\mathtt{11}b\theta, a_1) = \mathcal{B}_k(\mathtt{11}b\theta', a_1)$ as both equal $\mathtt{0}^k$; (ii) $\mathcal{B}_k(\mathtt{110}\theta, a_2) = \mathtt{110}\theta \in B_1\backslash\{\mathtt{10}^{k-1}\}$ and $\mathcal{B}_k(\mathtt{110}\theta', a_2) = \mathtt{110}\theta' \in B_1\backslash\{\mathtt{10}^{k-1}\}$; (iii) $\mathcal{B}_k(\mathtt{111}\theta, a_2) = \mathcal{B}_k(\mathtt{111}\theta', a_2)$, viz. are equal to 10^{k-1}; (iv) for $j > 2$ it holds that $\mathcal{B}_k(\mathtt{11}b\theta, a_j), \mathcal{B}_k(\mathtt{11}b\theta', a_j) \in B_1\backslash\{\mathtt{10}^{k-1}\}$. Since $\varrho_1 \neq \varrho_0$, ϱ_1 has at least two blocks. Hence these must be B_0 and B_1.

Next we prove that every refinement of ϱ_i into ϱ_{i+1} of Π', for i, $1 \leqslant i < \ell$, is valid for $\mathcal{B}_{k-2}$. We first observe that, for all $\sigma, \sigma' \in \mathbb{B}^{k-2}$, $a_j \in \mathcal{A}$, it holds that $\mathcal{B}_{k-2}(\sigma, a_j) = \sigma'$ iff $\mathcal{B}_k(\mathtt{11}\sigma, a_{j+2}) = \mathtt{11}\sigma'$, a direct consequence of the definition of the transition functions of $\mathcal{B}_{k-2}$ and $\mathcal{B}_k$. From this we obtain

$$\sigma =_{\varrho_i} \sigma' \iff \mathtt{11}\sigma =_{\pi_h} \mathtt{11}\sigma' \tag{1}$$

provided $\varrho_i = p(\pi_h)$, for $0 \leqslant i \leqslant \ell$, via the definition of the projection function p. Now, consider subsequent partitions ϱ_i and ϱ_i in Π'. Let h, $0 \leqslant h < n$, be such that $\varrho_i = p(\pi_h)$ and $\varrho_{i+1} = p(\pi_{h+1})$. Clearly, ϱ_{i+1} is a refinement of ϱ_i; if for $B \in \pi_{h+1}$ we have $B = \bigcup_{\alpha\in I} B_\alpha$ with $B_\alpha \in \pi_h$ for $\alpha \in I$, then for $p[B] \in \varrho_{i+1}$ we have $p[B] = \bigcup_{\alpha\in I} p[B_\alpha]$ with $p[B_\alpha] \in \varrho_i$ for $\alpha \in I$. The validity of the refinement of ϱ_i into ϱ_{i+1} is justified by the validity of π_{h+1} into π_h. If $\sigma =_{\varrho_i} \sigma'$ and $\sigma \neq_{\varrho_{i+1}} \sigma'$ for $\sigma, \sigma' \in \mathbb{B}^{k-2}$, then $\sigma, \sigma' \in B_0$ or $\sigma, \sigma' \in B_1$ since ϱ_i is a refinement of ϱ_0. Moreover, $\mathtt{11}\sigma =_{\pi_h} \mathtt{11}\sigma'$ and $\mathtt{11}\sigma \neq_{\pi_{h+1}} 11\sigma'$ by (1). Hence, by validity, $\mathcal{B}_k(\mathtt{11}\sigma, a_j) \neq_{\pi_h} \mathcal{B}_k(\mathtt{11}\sigma', a_j)$. Clearly $j \neq 1$. Also, $j \neq 2$, since $\sigma[1] = \sigma'[1]$ we have $(\mathtt{11}\sigma)[3] = (\mathtt{11}\sigma')[3]$. Therefore, $\mathcal{B}_{k-2}(\sigma, a_{j-2}) \neq_{\pi_h} \mathcal{B}_{k-2}(\sigma', a_{j-2})$, showing the refinement of ϱ_i into ϱ_{i+1} to be valid.

Finally, since every block in π_n is a singleton, this is also the case for ϱ_ℓ. Thus, ϱ_ℓ is indeed the coarsest partition as required for Π' to be a valid refinement sequence for $\mathcal{B}_{k-2}$. Every refinement of ϱ_i into ϱ_{i+1} of Π' is projected from a refinement of some π_h into π_{h+1} of Π as argued above. Therefore, since $\varrho_i = p(\pi_i)$ and $\varrho_{i+1} = p(\pi_{i+1})$, we have $rc(\varrho_i, \varrho_{i+1}) \leqslant rc(\pi_h, \pi_{h+1})$, and hence $rc(\Pi) = \sum_{h=1}^{n} rc(\pi_{h-1}, \pi_h) \geqslant \sum_{i=}^{\ell} rc(\varrho_{i-1}, \varrho_i) = rc(\Pi') \geqslant rc(\mathcal{B}_{k-2})$, as was to be shown. ◀

Next we combine the above lemma with the lowerbound provided by Theorem 6 in order to prove the main result of this section.

▶ **Theorem 11.** *Any partition refinement algorithm with end structure oracle to decide bisimilarity for a dLTS is $\Omega(n \log n)$.*

Proof. Let $\mathcal{B}'_k$ be the updated bisplitter (with the initial partition π'_0 containing $\{\mathtt{0}^k\}$, $B_0\backslash\{\mathtt{0}^k\}$, $\{\mathtt{10}^{k-1}\}$, and $B_1\backslash\{\mathtt{10}^{k-1}\}$ as given by the oracle for end structures rather than the partition π_0 containing B_0 and B_1). By Lemma 10 we have, for $k > 2$, that $rc(\mathcal{B}'_k) \geqslant rc(\mathcal{B}_{k-2})$. By Theorem 6 we know that $rc(\mathcal{B}_{k-2}) \geqslant \frac{1}{2}n' \log \frac{1}{2}n'$ where $n' = 2^{k-2}$ is the number of states of $\mathcal{B}_{k-2}$. It holds that $n' = \frac{2^{k-2}}{2^k}n = \frac{1}{4}n$. So $rc(\mathcal{B}'_k) \geqslant \frac{1}{8}n \log \frac{1}{8}n$ from which we conclude that deciding bisimilarity for $\mathcal{B}_k$ with the help of an oracle for the end structures is $\Omega(n \log n)$. ◀

6 $\mathcal{C}_k$ is $\Omega((m+n)\log n)$ for partition refinement

We modify the bisplitter $\mathcal{B}_k$, that has an action alphabet of $k-1$ actions, to obtain a dLTS with two actions only. The resulting dLTS $\mathcal{C}_k$ has the action alphabet $\{a,b\}$, for each $k>1$, and is referred to as the k-th *layered bisplitter*. We use $\mathcal{C}_k$ to obtain a $\Omega((n+m)\log n)$ lowerbound for deciding bisimilarity for LTSs with only two actions, where n is the number of states and m is de number of transitions.

To this end we adapt the construction of $\mathcal{B}_k$ at two places. Given an action alphabet $\mathcal{A}$ of $\mathcal{B}_k$ of $k-1$ actions, we introduce for each $\sigma\in\mathbb{B}^k$, a stake of 2^k states. Moreover, for each stake we add a tree gadget. These gadgets have height $\lceil\log(\frac{k-1}{2})\rceil$ to accommodate $\lceil(k-1)/2\rceil$ leaves.

▶ **Definition 12.** *Let $k>1$, $\mathcal{B}_k$ be the k-th bisplitter, and $\mathbb{A}=\{a,b\}$. The dLTS $\mathcal{C}_k=(S_k^{\mathcal{C}},\mathbb{A},\to_{\mathcal{C}},\pi_0^{\mathcal{C}})$, over the action set $\mathbb{A}$,*

(a) *has the set of states $S_k^{\mathcal{C}}$ defined as*

$$\begin{aligned} S_k^{\mathcal{C}} \;=\; & \{\,[\sigma,\ell]\in\mathbb{B}^k\times\mathbb{N}\mid 1\leqslant\ell\leqslant 2^k\,\}\cup\\ & \{\,\langle\sigma,w\rangle\in\mathbb{B}^k\times\mathbb{A}^*\mid 0\leqslant|w|\leqslant\lceil\log(\tfrac{k-1}{2})\rceil\,\},\end{aligned}$$

(b) *has the transition relation $\to_{\mathcal{C}}$ given by*

$$\begin{array}{rcll} [\sigma,\ell] & \xrightarrow{\alpha}_{\mathcal{C}} & [\sigma,\ell+1] & \text{for } \sigma\in\mathbb{B}^k,\ 1\leqslant\ell\leqslant 2^k,\ \alpha\in\mathbb{A}\\ [\sigma,2^k] & \xrightarrow{\alpha}_{\mathcal{C}} & \langle\sigma,\varepsilon\rangle & \text{for } \sigma\in\mathbb{B}^k,\ \alpha\in\mathbb{A}\\ \langle\sigma,w\rangle & \xrightarrow{\alpha}_{\mathcal{C}} & \langle\sigma,w\alpha\rangle & \text{for } \sigma\in\mathbb{B}^k,\ |w|<\lceil\log(\frac{k-1}{2})\rceil,\ \alpha\in\mathbb{A}\\ \langle\sigma,w\rangle & \xrightarrow{\alpha}_{\mathcal{C}} & [\sigma',1] & \text{for } \sigma\in\mathbb{B}^k,\ |w|=\lceil\log(\frac{k-1}{2})\rceil,\ \mathrm{bin}(w\alpha)=j,\ \mathcal{B}_k(\sigma,a_j)=\sigma', \end{array}$$

(c) *and has the initial partition $\pi_0^{\mathcal{C}}$ defined as*

$$\begin{aligned}\pi_0^{\mathcal{C}} \;=\; & \big\{\,\{\,[\sigma,\ell]\mid\sigma\in B_0\,\},\{\,[\sigma,\ell]\mid\sigma\in B_1\,\}\;\big|\;1\leqslant\ell\leqslant 2^k\,\big\}\cup\\ & \{\,\langle\sigma,w\rangle\in S_k^{\mathcal{C}}\mid\sigma\in\mathbb{B}^k,\ w\in\mathbb{A}^*\,\}.\end{aligned}$$

The auxilliary function $\mathrm{bin}:\mathbb{A}^{\leqslant\lceil\log(k-1)\rceil}\to\mathbb{N}$, *used in item b is inductively defined by* $\mathrm{bin}(\varepsilon)=0$, $\mathrm{bin}(wa)=\min\{2*\mathrm{bin}(w),k-1\}$, *and* $\mathrm{bin}(wb)=\min\{2*\mathrm{bin}(w)+1,k-1\}$.

We see that with each string $\sigma\in\mathbb{B}^k$ we associate in $\mathcal{C}_k$ as many as 2^k stake states $[\sigma,1],\ldots,[\sigma,2^k]$, one for each level ℓ, $1\leqslant\ell\leqslant 2^k$. The stake states are traversed from the top $[\sigma,1]$ to bottom $[\sigma,2^k]$ on any string σ of length 2^k over $\mathbb{A}$. The tree gadget consists of a complete binary tree of height $\lceil\log(\frac{k-1}{2})\rceil$ that hence has at least $\lceil(k-1)/2\rceil$ leaves. Traversal down the tree takes a left child on action a, a right child on action b. Together with the two actions of $\mathbb{A}$, $k-1$ source-label pairs can be encoded. To simulate a transition $\sigma\xrightarrow{a_j}\sigma'$ of $\mathcal{B}_k$ in $\mathcal{C}_k$ from a leaf of a tree gadget of σ to the top of the stake of σ', we need to be at a leaf $\langle\sigma,w\rangle$ of the tree gadget of σ such that the combined string $w\alpha$ for $\alpha\in\mathbb{A}$ is the binary encoding according of bin of the index j. An α-transition thus leads from the source $\langle\sigma,w\rangle$ to the target $[\sigma',1]$ if $\sigma\xrightarrow{a_j}\sigma'$ in $\mathcal{B}_k$ and $w\alpha$ corresponds to j. The partition $\pi_0^{\mathcal{C}}=\{\,C_0^\ell,C_1^\ell\mid 1\leqslant\ell\leqslant 2^k\,\}\cup\{C_\varepsilon\}$ distinguishes, for each level ℓ, the states at level ℓ of the stakes of strings starting with 0 in C_0^ℓ, the states of the stakes at level ℓ of strings starting with 1 in C_1^ℓ, and the states of the tree gadgets collected in C_ε.

Figure 3 depicts the two-label layered 3-splitter $\mathcal{C}_3$. Because also $\mathcal{B}_k$ has an action set of size 2 the tree gadgets only consist of the root node of the form $\langle\sigma,\varepsilon\rangle$. In Figure 2 of $\mathcal{B}_3$ we see that $101\xrightarrow{a_1}101$ and $101\xrightarrow{a_2}110$. In Figure 3 we have transitions $\langle 101,\varepsilon\rangle\xrightarrow{a}[101,1]$ and $\langle 101,\varepsilon\rangle\xrightarrow{b}[110,1]$ (dotted and dashed, respectively). Coloring of nodes is used to represent the initial partition $\pi_3^{\mathcal{C}}$ that separates 8 times, once for each level ℓ, the four states of the stakes in C_0^ℓ on the left from the four stake states in C_1^ℓ on the right, and the 8 tree states in C_ε at the bottom of the picture.

The 6-th bisplitter $\mathcal{B}_6$ has five actions, a_1 to a_5. A tree gadget for the layered bisplitter $\mathcal{C}_6$ with corresponding outgoing transitions is drawn in Figure 4. The tree has height $\lceil\log((6-1)/2)\rceil=$

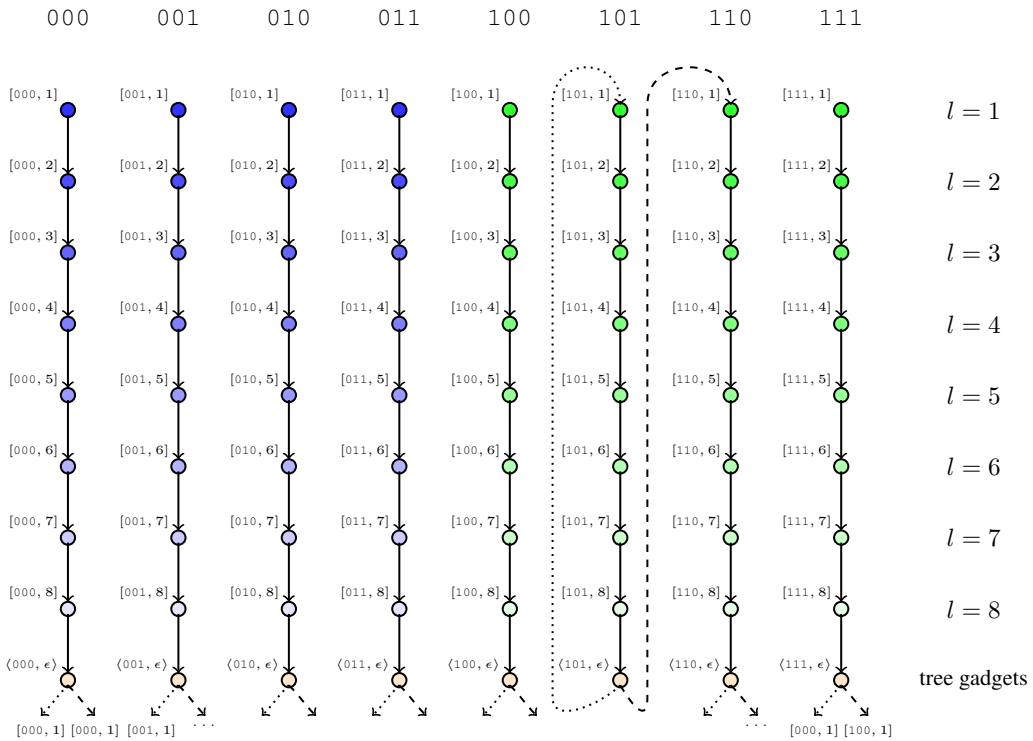

Figure 3 The partial layered bisplitter $\mathcal{C}_3$ with tree gadgets, the colors represent the initial partition

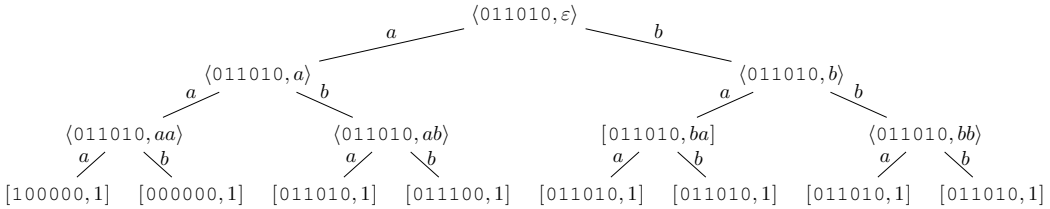

Figure 4 Example of the outgoing tree for $\mathcal{C}_6$ from the root $[011010, \varepsilon] \in S^{\mathcal{C}}_6$

$\lceil \log \frac{5}{2} \rceil = 2$, hence it has $2^2 = 4$ leaves. Since each leaf has two outgoing transitions, one labeled a and one labeled b, the two leftmost leaves $\langle \sigma, aa \rangle$ and $\langle \sigma, ab \rangle$ are used with the two labels a and b to simulate transitions for a_1 up to a_4, the two rightmost leaves $\langle \sigma, ba \rangle$ and $\langle \sigma, bb \rangle$ have together four transitions all simulating the a_5-transition of σ.

The next lemma introduces three facts for the layered bisplitter $\mathcal{C}_k$ that we need in the sequel. The first states that if two states in different stakes, but at the same level, are separated during partition refinement, then all corresponding states at lower levels are separated as well. The second fact helps to transfer witnessing transitions in $\mathcal{B}_k$ to the setting of $\mathcal{C}_k$. A transition $\sigma \xrightarrow{a_j} \sigma'$ of $\mathcal{B}_k$ is reflected by a path from $[\sigma, 2^k]$ through the tree gadget of σ from root to leaf and then to the top state $[\sigma', 1]$ of the stake of σ'. The word $w\alpha$ encountered going down and out the tree gadget corresponds to the action a_j according to the bin-function. Lastly, it is shown that no two pairs of different states within the stakes are bisimilar.

▶ **Lemma 13.** *Let Π be a valid refinement sequence for $\mathcal{C}_k$ and π a partition in Π.*

(a) *If two states* $[\sigma,\ell],[\sigma',\ell] \in S^{\mathcal{C}}_k$*, for* $1 \leqslant \ell \leqslant 2^k$*, are in a different block of* π*, then all pairs* $[\sigma,m],[\sigma',m] \in S$*, for all levels* m*,* $\ell \leqslant m \leqslant 2^k$*, are in different blocks of* π*.*

(b) *If* $[\sigma_1,2^k]$ *and* $[\sigma_2,2^k]$ *are split in* π*, then exist* $w \in \mathbb{A}^*$*,* $\alpha \in \mathbb{A}$*, and* $\sigma'_1,\sigma'_2 \in \mathbb{B}^k$ *such that*

$$[\sigma_1,2^k] \xrightarrow{w}{}^*_{\mathcal{C}} \langle\sigma_1,w\rangle \xrightarrow{\alpha}_{\mathcal{C}} [\sigma'_1,1] \quad \textit{and} \quad [\sigma_2,2^k] \xrightarrow{w}{}^*_{\mathcal{C}} \langle\sigma_2,w\rangle \xrightarrow{\alpha}_{\mathcal{C}} [\sigma'_2,1]$$

with $[\sigma'_1,1]$ *and* $[\sigma'_2,1]$ *in different blocks of* π*.*

(c) *If* π *is the last refinement in* Π*, it contains the singletons of* $[\sigma,\ell]$ *for* $\sigma \in \mathbb{B}^k$ *and* $1 \leqslant \ell \leqslant 2^k$*.*

Proof. (a) For a proof by contradiction, suppose the partition π is the first partition of Π that falsifies the statement of the lemma. So $\pi \neq \pi^{\mathcal{C}}_0$, since for the initial refinement $\pi^{\mathcal{C}}_0$ the statement holds. Thus, π is a refinement of a partition π' in Π. So, there are two states $[\sigma,\ell],[\sigma',\ell] \in S^{\mathcal{C}}_k$ in different blocks of π while the states $[\sigma,\ell{+}1],[\sigma',\ell{+}1]$ are in the same block of π and hence of π'. Since $[\sigma,\ell]$ and $[\sigma',\ell]$ only have transitions to $[\sigma,\ell{+}1]$ and $[\sigma',\ell{+}1]$, respectively, that are in the same block π', the refinement wouldn't have been valid. We conclude that no falsifying partition π in Π exists and that the lemma holds.

(b) We first prove, by induction on $|w|$, that if $\langle\sigma_1,w\rangle$ and $\langle\sigma_2,w\rangle$ are split in π, then exist $w \in \mathbb{A}^*$ and $\alpha \in \mathbb{A}$ such that $\langle\sigma_1,w\rangle \xrightarrow{v}{}^* \langle\sigma_1,wv\rangle \xrightarrow{\alpha} [\sigma'_1,1]$ and $\langle\sigma_2,w\rangle \xrightarrow{v}{}^* \langle\sigma_2,wv\rangle \xrightarrow{\alpha} [\sigma'_2,1]$ with $[\sigma'_1,1]$ and $[\sigma'_2,1]$ in different blocks of π. If w has maximal length, $|w| = \lceil\log(\frac{k-1}{2})\rceil$ this is clear. If $\langle\sigma_1,w\rangle$ and $\langle\sigma,w\rangle$ are split, for $|w| < \lceil\log(k{-}1)\rceil - 1$, then either a-transitions or b-transitions lead to split states. By the induction hypothesis, suitable paths exists from the targets of such transitions. Adding the respective transition proves the induction hypothesis. Since $[\sigma_1,2^k]$ and $[\sigma_2,2^k]$ can only reach $\langle\sigma_1,\varepsilon\rangle$ and $\langle\sigma_2,\varepsilon\rangle$ the statement follows.

(c) Choose ℓ, $1 \leqslant \ell \leqslant 2^k$ and define the relation $R \subseteq S^{\mathcal{B}}_k \times S^{\mathcal{B}}_k$ such that $(\sigma_1,\sigma_2) \in R$ iff the stake states $[\sigma_1,\ell],[\sigma_2,\ell] \in S^{\mathcal{C}}_k$ are bisimilar for $\mathcal{C}_k$. We verify that R is a bisimulation relation for $\mathcal{B}_k$. Note, that R respects $\pi^{\mathcal{B}}_k$, the initial partifion of $\mathcal{B}_k$. Now, suppose $(\sigma_1,\sigma_2) \in_\ell R$ and $\sigma_1 \xrightarrow{a_j} \sigma'_1$ for some $a_j \in \mathcal{A}_k$ and $\sigma'_1 \in S^{\mathcal{B}}_k$. By construction of $\mathcal{C}_k$ we have $[\sigma_1,\ell] \xrightarrow{a^*} [\sigma_1,2^k] \xrightarrow{a} \langle\sigma_1,\varepsilon\rangle \xrightarrow{w}{}^* \langle\sigma_1,w\rangle \xrightarrow{\alpha} [\sigma'_1,1] \xrightarrow{a^{\ell-1}}{}^* [\sigma'_1,\ell]$ where $\mathrm{bin}(w\alpha) = j$. Since $[\sigma_1,\ell]$ and $[\sigma_2,\ell]$ are bisimilar in $\mathcal{C}_k$, it follows that a corresponding path $[\sigma_2,\ell] \to^* [\sigma'_2,\ell]$ exists in $\mathcal{C}_k$ with $[\sigma'_1,\ell]$ and $[\sigma'_2,\ell]$ bisimilar in $\mathcal{C}_k$. From this we derive that $\sigma_2 \xrightarrow{a_j} \sigma'_2$ in $\mathcal{B}_k$ and $(\sigma'_1,\sigma'_2) \in R$. Hence, R is a bisimulation relation for $\mathcal{B}_k$ indeed. Now, bisimilarity of $\mathcal{B}_k$ is discrete. Thus, if two stake states $[\sigma_1,\ell]$ and $[\sigma_2,\ell]$ are bisimilar for $\mathcal{C}_k$, then σ_1 and σ_2 are bisimilar for $\mathcal{B}_k$ thus $\sigma_1 = \sigma_2$, and therefore $[\sigma_1,\ell] = [\sigma_2,\ell]$. ◀

The next lemma states that all the splitting of states $[\sigma,\ell] \in S^{\mathcal{C}}$ at some level ℓ has refinement costs that are at least that of $\mathcal{B}_k$.

▶ **Lemma 14.** *It holds that* $rc(\mathcal{C}_k) \geqslant 2^k rc(\mathcal{B}_k)$ *for all* $k > 1$*.*

Proof. Let $\Pi = (\pi^{\mathcal{C}}_0,\pi_1,\ldots,\pi_n)$ be a valid refinement sequence for $\mathcal{C}_k$. We show that for each level ℓ, the sequence Π induces a valid refinement sequence Π^ℓ for $\mathcal{B}_k$.

The mapping p_ℓ assigns to a partition π of $\mathcal{C}_k$ a partition $p_\ell(\pi)$ by putting

$$p_\ell(\pi) = \{\,\{\,\sigma \in \mathbb{B}^k \mid [\sigma,\ell] \in B\,\} \mid B \in \pi\,\} \setminus \{\emptyset\}.$$

The sequence $\Pi^\ell = (\pi^\ell_0,\ldots,\pi^\ell_m)$ is obtained from the sequence $(p_\ell(\pi^{\mathcal{C}}_0),p_\ell(\pi_1),\ldots,p_\ell(\pi_n))$ by removing possible duplicates. We verify that Π^ℓ is a valid refinement sequence for $\mathcal{B}_k$.

First, we check that π^ℓ_i is a refinement of π^ℓ_{i-1}, for $1 \leqslant i \leqslant m$. Choose such an index i arbitrary. Let the index h with $1 \leqslant h \leqslant n$ by such that $p_\ell(\pi_{h-1}) = \pi^\ell_{i-1}$ and $p_\ell(\pi_i) = \pi^\ell_h$. For each block $B' \in \pi^\ell_i$ exists a block $B \in \pi_h$ such that $B' = p_\ell(B)$. Since π_h is a refinement of π_{h-1},

thus $B = \bigcup_r B_r$ for suitable $B_r \in \pi_h$. Note, $p_\ell(B_r) \in p_\ell(\pi_{h-1})$ for each index r. We have $B' = \bigcup_r p_\ell(B_r)$ with $p_\ell(B_r) \in \pi^\ell_{i-1}$, and π^ℓ_i is a refinement of π^ℓ_{i-1}.

Next, we verify that Π^ℓ is a valid refinement sequence for $\mathcal{B}_k$. Suppose the state $\sigma_1, \sigma \in S^{\mathcal{B}}_k$ are split for the refinement of π^ℓ_{i-1} into π^ℓ_i. Then the states $[\sigma_1, \ell], [\sigma_2, \ell] \in S^{\mathcal{C}}_k$ are split for the refinement of a partition π_{h-1} into the partition π_h for a some index h, $1 \leqslant h \leqslant n$. Then either (i) $\ell = 2^k$ and $[\sigma_1, \ell]$ and $[\sigma_2, \ell]$ have α-transitions to different blocks, for some $\alpha \in \mathbb{A}$, or (ii) $\ell < 2^k$ and $[\sigma_1, \ell{+}1]$ and $[\sigma_2, \ell{+}1]$ are in different blocks of π_{h-1}. In the case of (ii), it follows by Lemma 13 that also $[\sigma_1, 2^k]$ and $[\sigma_2, 2^k]$ are in different blocks of π_{h-1}. Thus, for the refinement of some π_{g-1} into π_g, $1 \leqslant g \leqslant h \leqslant n$, splitted the two states $[\sigma_1, 2^k]$ and $[\sigma_2, 2^k]$. By Lemma 13 exist $w \in \mathbb{A}^*$, $\alpha \in \mathbb{A}$, and $\sigma'_1, \sigma'_2 \in \mathbb{B}^k$ such that

$$[\sigma_1, 2^k] \xrightarrow{w}{}^*_{\mathcal{C}} \langle \sigma_1, w\rangle \xrightarrow{\alpha}_{\mathcal{C}} [\sigma'_1, 1] \quad \text{and} \quad [\sigma_2, 2^k] \xrightarrow{w}{}^*_{\mathcal{C}} \langle \sigma_2, w\rangle \xrightarrow{\alpha}_{\mathcal{C}} [\sigma'_2, 1]$$

with $[\sigma'_1, 1]$ and $[\sigma'_2, 1]$ in different blocks of π_{g-1}. Hence, σ'_1 and σ'_2 are in different blocks of π^ℓ_{i-1} while $\sigma_1 \xrightarrow{a_j}_{\mathcal{B}} \sigma'_1$ and $\sigma_2 \xrightarrow{a_j}_{\mathcal{B}} \sigma'_2$ for $j = \mathrm{bin}(w\alpha)$, which justifies splitting σ_1 and σ_2 for π^ℓ_i. We conclude that Π^ℓ is a valid refinement sequence for $\mathcal{B}_k$.

We have established that if Π is a valid refinement sequence for $\mathcal{C}_k$, then Π^ℓ is a valid refinement sequence for $\mathcal{B}_k$. The sequence Π^ℓ is obtained from Π by sifting out the blocks of Π's partitions and removing repeated partitions. Therefore it holds that $rc(\Pi) \geqslant rc(\Pi^\ell)$. Since the mapping p_ℓ and $p_{\ell'}$ include pairswise distinct sets of stake states for $\ell \neq \ell'$, $1 \leqslant \ell \leqslant 2^k$, it follows that $rc(\Pi) \geqslant \sum_{\ell=1}^{2^k} rc(\Pi^\ell) \geqslant 2^k rc(\mathcal{B}_k)$. Taking the minimum over all valid refinement sequences for $\mathcal{C}_k$ we conclude that $rc(\mathcal{C}_k) \geqslant 2^k rc(\mathcal{B}_k)$ as was to be shown. ◀

With the above technical lemma in place, we are able to strengthen the $\Omega(n \log n)$ lowerbound of Theorem 6 to account for the number of transitions. The improved lowerbound is $\Omega((m+n)\log n)$, where m is the number of transitions and n the number of states.

▶ **Theorem 15.** *Deciding bisimilarity for dLTSs with a partition refinement algorithm is $\Omega((m+n)\log n)$, where n is the number of states and m is the number of transitions of the dLTS.*

Proof. For the bisplitter $\mathcal{B}_k$, we know by Theorem 6 that $rc(\mathcal{B}_K) \geqslant 2^{k-1}(k{-}1)$. Thus, by Lemma 14, we obtain $rc(\mathcal{C}_K) \geqslant 2^{2k-1}(k{-}1)$. In the case of $\mathcal{C}_k$ we have $n = 2^k(2^k + 2^{\lceil \log(k-1)\rceil} - 1)$ and $m = 2n$. Hence $n + m \in \Theta(2^{2k-1})$ and $\log n \in \Theta(k-1)$, from which it follows that $rc(\mathcal{C}_k) \in \Omega((m+n)\log n)$. ◀

Underlying the proof of a lowerbound for deciding bisimilarity for the family of layered bisplitters $\mathcal{C}_k$ is the observation that each $\mathcal{C}_k$ can be seen as 2^k stacked instances of the ordinary bisplitters $\mathcal{B}_k$, augmented with tree gadgets to handle transitions properly. The other essential ingredient for the proof of Theorem 15 is the complexity of deciding bisimilarity with a partition refinement algorithm on the $\mathcal{B}_k$ family. The same reasoning applies when considering partition refinement algorithms with an oracle for end structures from Section 5. Also with an oracle the lowerbound of $\Omega((m{+}n)\log n)$ remains.

▶ **Theorem 16.** *Any partition refinement algorithm with an oracle for end structures that decides bisimilarity for dLTSs is $\Omega((m+n)\log n)$.*

Proof sketch. The proof is similar to that of Lemma 10 and Theorem 15. Consider, for some $k > 2$, the layered bisplitter $\mathcal{C}_k$ having initial partition π_0. The dLTS $\mathcal{C}_k$ has two end structures, viz. the set $S_0 \subset S^{\mathcal{C}}_k$ containing the states of the stake and accompanying tree gadget $S_0 = \{\, [\mathtt{0}^k, \ell] \mid 1 \leqslant \ell \leqslant 2^k \,\} \cup \{\, \langle \mathtt{0}^k, w\rangle \mid w \in \mathbb{A}^*,\ |w| \leqslant \lceil \log(\frac{k-1}{2})\rceil \,\}$ for $\mathtt{0}^k$ and a similar $S_1 \subseteq S^{\mathcal{C}}_k$ for $\mathtt{1}\mathtt{0}^{k-1}$. The sets S_0 and S_1 are minimally closed under the transitions of $\mathcal{C}_k$. Other states, on the stake or tree

gadget for a string σ, have a path to these sets inherited from a path from σ to $\mathtt{0}^k$ or $\mathtt{10}^k$ in $\mathcal{B}_k$. The bisimulation classes S'_0 and S'_1, say, with respect to $S^{\mathcal{C}}_k$ rather than π_0, consist of S_0 and S_1 themselves plus a part of the tree gadgets for transitions in $\mathcal{C}_k$ leading to S_0 and S_1, respectively.

The update of the initial partition π_0 with oracle information, which concerns, ignoring the tree gadgets, the common refinement of the layers on $\{\,[\sigma,\ell] \mid \sigma \in B_0\,\}$ and $\{\,[\sigma,\ell] \mid \sigma \in B_1\,\}$ on the one hand, and (a trivial refinement of) the bisimulation classes S'_0 and S'_1 on the other hand, is therefore equal to π on the stakes (and generally finer on the tree gadgets).

Then every valid refinement sequence $\Pi = (\pi'_0, \pi_2, \ldots, \pi_n)$ for the updated dLTS $\mathcal{C}'_k = (S, \mathcal{A}, \to, \pi'_0)$ satisfies $rc(\Pi) \geqslant rc(\mathcal{C}_{k-2})$. Following the lines of the proof of Lemma 10, we can show that a valid refinement sequence Π for $\mathcal{C}_k$ with updated initial partition π'_0 induces a valid refinement sequence Π' for $\mathcal{C}_{k-2}$.

The number of states in $\mathcal{C}_{k-2}$ is $\Theta(n)$ with n the number of states of $\mathcal{C}_k$, and number of transitions in $\mathcal{C}_{k-2}$ is $\Theta(m)$ with m the number of transitions of $\mathcal{C}_k$. Therefore, $rc(\Pi) \geqslant rc(\Pi') \geqslant rc(\mathcal{C}_{k-2})$, from which we derive that any partition refinement algorithm with oracle for end structures involves $\Theta((m+n)\log n)$ times moving a state for $\mathcal{C}_k$ and that hence the algorithm is $\Omega((m+n)\log n)$. ◀

7 Conclusion

We have shown that, even when restricted to deterministic LTSs, it is not possible to construct an algorithm based on partition refinement that is more efficient than $\Omega((m+n)\log n)$. This strengthens the result of [2]. The bound obtained is preserved even when the algorithm is extended with an oracle that can determine for specific states whether they are bisimilar or not in constant time. The oracle proof technique enabled us to show that the algorithmic ideas underlying Roberts' algorithm [15] for the one-letter alphabet case cannot be used to come up with a fundamentally faster enhanced partition refinement algorithm for bisimulation. Of course, this is not addressing a generic lower bound to decide bisimilarity on LTSs, nor proving the conjecture that the Paige-Tarjan algorithm is optimal for deciding bisimilarity. It is conceivable that a more efficient algorithm exists that is not based on partitioning for bisimulation. However, as it stands, no techniques are known to prove such a generic algorithmic lowerbound, and all that that do exist make assumptions on allowed operations, such as the well-known lowerbound on sorting. Further investigations to obtain a more general lowerbound may strenghten the oracle used even further, such that a wider range of algorithms is covered.

For the parallel setting, where deciding bisimilarity can be done faster indeed, a similar dichotomy between the case of a single letter alphabet and of a multiple letter alphabet occurs. For LTSs with multiple action labels a linear algorithm is proposed in [12], whereas for dLTSs with one action label it is possible to calculate bisimulation in logarithmic time, cf. [9]. The question is raised in [17], if a sub-linear parallel solution exists at all. Since this problem in a general setting is known to be P-complete as shown in [1], it is generally believed that no logarithmic algorithm is possible. It is worthwhile to transfer the results of this paper to a parallel setting, in order to better understand whether it is possible to design parallel partition based algorithms for bisimulation on LTSs that have a sub-linear complexity.

Acknowledgements We are grateful to the anonymous reviewers of CONCUR 2021 their thorough reading and constructive feedback.

References

1 J. Balcázar, J. Gabarro, and M. Santha. Deciding bisimilarity is P-complete. *Formal Aspects of Computing*, 4(1):638–648, 1992.

2 C. Berkholz, P. Bonsma, and M. Grohe. Tight lower and upper bounds for the complexity of canonical colour refinement. *Theory of Computing Systems*, 60(4):581–614, 2017.

3 J. Berstel and O. Carton. On the complexity of Hopcroft's state minimization algorithm. In M. Domaratzki et al., editor, *Proc. CIAA 2004*, pages 35–44. LNCS 3317, 2004.

4 P. Buchholz. Exact performance equivalence: An equivalence relation for stochastic automata. *Theoretical Computer Science*, 215:263–287, 1999.

5 G. Castiglione, A. Restivo, and M. Sciortino. Hopcroft's algorithm and cyclic automata. In C. Martín-Vide et al., editor, *Proc. LATA 2008*, pages 172–183. LNCS 5196, 2008.

6 A. Dovier, C. Piazza, and A. Policriti. An efficient algorithm for computing bisimulation equivalence. *Theoretical Computer Sciemce*, 311:221–256, 2004.

7 J.F. Groote, H.J. Rivera Verduzco, and E.P. de Vink. An efficient algorithm to determine probabilistic bisimulation. *Algorithms*, 11(9):131,1–22, 2018.

8 J. Hopcroft. An $n \log n$ algorithm for minimizing states in a finite automaton. In Z. Kohavi and A. Paz, editors, *Theory of Machines and Computations*, pages 189–196. Academic Press, 1971.

9 J. Jájá and Kwan Woo Ryu. An efficient parallel algorithm for the single function coarsest partition problem. *Theoretical Computer Science*, 129(2):293–307, 1994.

10 D.N. Jansen, J.F. Groote, J.J.A. Keiren, and A. Wijs. An O$(m \log n)$ algorithm for branching bisimilarity on labelled transition systems. In A. Biere and D. Parker, editors, *Proc. TACAS*, pages 3–20. LNCS 12079, 2020.

11 P.C. Kanellakis and S.A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. *Information and Computation*, 86(1):43–68, 1990.

12 J. Martens, J.F. Groote, L. van de Haak, P. Hijma, and A. Wijs. A linear parallel algorithm to compute bisimulation and relational coarsest partitions. In preparation.

13 R. Milner. *A Calculus of Communicating Systems*. LNCS 92, 1980.

14 R. Paige and R.E. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.

15 R. Paige, R.E. Tarjan, and R. Bonic. A linear time solution to the single function coarsest partition problem. *Theoretical Computer Science*, 40:67–84, 1985.

16 D.M.R. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Proc. 5th GI-Conference on Theoretical Computer Science*, pages 167–183. LNCS 104, 1981.

17 S. Rajasekaran and I. Lee. Parallel algorithms for relational coarsest partition problems. *IEEE Transactions on Parallel and Distrubuted Systems*, 9(7):687–699, 1998.

18 T. Wißmann, U. Dorsch, S. Milius, and L. Schröder. Efficient and modular coalgebraic partition refinement. *Logical Methods Computer Science*, 16(1), 2020.