Electronic Communications of the EASST
Volume X (2013)

Proceedings of the Automated Verification of Critical Systems (AVoCS 2013)

On the Efficiency of Deciding Probabilistic Automata Weak Bisimulation

Vahid Hashemi, Holger Hermanns and Andrea Turrini

15 pages

Guest Editors: Steve Schneider, Helen Treharne
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

On the Efficiency of Deciding Probabilistic Automata Weak Bisimulation

Vahid Hashemi[1], **Holger Hermanns**[2] **and Andrea Turrini**[3]

[1] hashemi@mpi-inf.mpg.de
Max Planck Institute for Informatics, Saarbrücken, Germany

[1] vhashemi@cs.uni-saarland.de
[2] hermanns@cs.uni-saarland.de
Department of Computer Science, Saarland University, Saarbrücken, Germany

[3] turrini@ios.ac.cn
State Key Laboratory of Computer Science, Institute of Software,
Chinese Academy of Sciences, Beijing, China

Abstract: Weak probabilistic bisimulation on probabilistic automata can be decided by an algorithm that needs to check a polynomial number of linear programming problems encoding weak transitions. It is hence polynomial, but not guaranteed to be strongly polynomial. In this paper we show that for polynomial rational probabilistic automata strong polynomial complexity can be ensured. We further discuss complexity bounds for generic probabilistic automata. Then we consider several practical algorithms and LP transformations that enable an efficient solution for the concrete weak transition problem. This sets the ground for effective compositional minimisation approaches for probabilistic automata and Markov decision processes.

Keywords: Complexity, Concurrency, Efficiency, Linear Programming, Markov Decision Processes, Weak Bisimulation

1 Introduction

Probability and nondeterminism are core aspects of concurrent systems. *Probability* for instance arises when a system, performing an action, is able to switch to more than one state and the likelihood of each of these states can be faithfully estimated. Probability can model both specific system choices (such as flipping a coin, commonly used in randomised distributed algorithms) and general system properties (such as message loss probabilities when sending a message over a wireless medium). *Nondeterminism* represents behaviours that we can not or do not want to attach a precise (possibly probabilistic) interpretation to. This might reflect the concurrent execution of several components at unknown (relative) speeds or behaviours we decide to keep undetermined for simplifying the system model or allowing for different implementations.

Several models have been proposed in the literature to study formally systems where a combination of probability and nondeterminism is considered: among others, there are Markov Decision Processes (*MDP*) [Der70], Labelled Concurrent Markov Chains (*LCMC*), Alternating Probabilistic Models [Var85, Han91, PLS00], and Probabilistic Automata (*PA*) [Seg95].

Probabilistic automata extend classical concurrency models in a simple yet conservative fashion. In probabilistic automata, there is no global notion of time, and concurrent processes may perform probabilistic experiments inside a transition. This is represented by transitions of the form $s \xrightarrow{a} \mu$, where s is a state, a is an action label, and μ is a probability measure on states. Labelled transition systems are instances of this model family, obtained by restricting to Dirac measures (assigning full probability to single states). Thus, foundational concepts and results of standard concurrency theory are retained in full and extend smoothly to the *PA* model. Since the *PA* model is akin to *MDP* model, its fundamental beauty can be paired with powerful model checking techniques, as implemented for instance in the PRISM tool [HKNP06]. We refer the interested reader to [Seg06] for a survey on this and other models.

Given a real system, we can conceive several different probabilistic automata models to reflect its behavior. *Bisimulation relations* provide a powerful tool to check whether two models describe essentially the same system. They are then called bisimilar. The bisimilarity of two systems can be viewed in terms of a game played between a challenger and a defender. In each step of the possibly infinite bisimulation game, the challenger chooses one automaton, makes a step, and the defender matches it with a step of the other automaton. Depending on how we want to treat internal computations, this leads to *strong* and *weak* bisimulations: the former requires that each single step of the challenger automaton is matched by an equally labelled single step of the defender automaton, the latter allows the matching up to internal computation steps. On the other hand, depending on how nondeterminism is resolved, probabilistic bisimulation can be varied by allowing the defender to match the challenger's step by a convex combination of enabled probabilistic transitions. This results in a spectrum of four bisimulations: strong [Seg95, Han91, Var85], strong probabilistic [Seg95], weak [PLS00, Seg95, EHZ10a, EHZ10b], and weak probabilistic [Seg95, EHZ10b, EHZ10a] bisimulation.

Besides comparing automata, bisimulation relations allow us to reduce the size of an automaton without changing its properties (i.e., with respect to logic formulae satisfied by it). This is particularly useful to alleviate the state explosion problem notoriously encountered in model checking. If the bisimulation is a congruence with respect to the operators of a process calculus used to build up the automata out of smaller ones, this can give rise to a compositional strategy to associate a small automaton model to a large system without intermediate state space explosion. In several related settings, this strategy has been proven very effective [CGM+96, HK00, KKZJ07, BHH+09, CHLS09]; it can speed up the overall model analysis or turn a too large problem into a tractable one. Both, strong and weak bisimilarity are used in practice, with weaker relations leading to greater reduction. However, this approach has thus far not been explored in the context of *MDP*s or probabilistic automata. A striking reason is that until recently no effective decision algorithm was at hand for weak bisimilarity on *PA*; a polynomial time decision algorithm has been proposed only recently [HT12], based on linear programming problems. That algorithm can be embedded into a procedure to compress a given *PA* to its canonical minimal representative [EHS+13]. Since weak bisimilarity is a congruence for parallel composition, hiding, and other operators on *PA*, this paves the way for compositional strategies to associate a small *PA* model to a large system without intermediate state space explosion.

The weak bisimilarity decision algorithm follows the usual partition-refinement approach, and thereby induces a polynomial number of linear programming problems that can be solved in

polynomial time [Kar84, Kha79]. The algorithm is hence polynomial, but is not guaranteed to be *strongly* polynomial. In other words, it is as yet unclear if there are hidden factors in the complexity, such as data dependencies that lead to a non-polynomial worst case if properly considered. In this paper, we discuss the efficiency of solving the specific LP problems from both theoretical and practical viewpoints. This establishes an interesting connection between the concurrency theory world and combinatorial optimisation world.

We first consider the theoretical efficiency of solving the problem. By restricting the class of considered models to rational probabilistic automata where probabilities are polynomially representable, we show that the feasibility of the LP problem can be checked in strongly polynomial time. We are not aware of any non-polynomially representable probabilistic automata appearing in any application study. So, this is indeed a prominent result.

We then consider the general rational case, and study the complexity of the decision problem together with several optimisations. We reformulate the original LP problem [HT12] in order to simplify the construction of the dual LP [BT97] which is smaller in size than the original. By using a state-of-the-art preconditioned conjugate gradient (PCG) algorithm combined with a partial updating procedure [Ans99] we show that the dual LP can be solved efficiently. On the other hand, taking advantage of the small-sized dual LP, we give an upper bound on the complexity of checking the feasibility of the original LP problem. As a matter of fact, to the best of our knowledge, this provides the tightest available worst case complexity for the verification problem at hand.

We furthermore discuss the practical efficiency of solving the decision problem. In practice one would usually opt for the notoriously efficient simplex method [Sha87] to solve the LP problems. But a small modification of the underlying network [HT12] enables us to adapt the corresponding LP problem into a variant of a minimum cost flow problem [AMO93] with flow proportional sets. This is a special class of linear programming problems where the underlying network structure can be exploited, in particular if it is sparse. Sparsity is indeed frequently observed in practical applications of probabilistic automata. We therefore compare the simplex method with a very efficient state-of-the-art network simplex algorithm [BF12] specialised for the minimum cost flow problem with additional side constraints. This is known to outperform the simplex method [MSJ11, HK95, Cal02] when the number of nodes is an order of magnitude larger than the number of side constraints.

Organisation of the paper. After the preliminaries in Section 2, we recall the LP problem construction in Section 3 and in Section 4 we focus on the efficiency of solving the LP problem from both a theoretical and a practical point of view. Section 5 concludes the paper. An appendix includes detailed proofs.

2 Mathematical Preliminaries and Probabilistic Automata

A σ-*field* over a set Ω is a subset $\mathscr{F} \subseteq 2^{\Omega}$ that contains the empty set $\emptyset$ and it is closed under complement and countable union. We say that $(\Omega, \mathscr{F})$ is a *measurable space* and we call the σ-field 2^{Ω} the *discrete* σ-field over Ω. Given $C \subseteq 2^{\Omega}$, we denote by $\sigma(C)$ the smallest σ-field containing C and we call it the σ-field generated by C.

A *measure* over a measurable space $(\Omega, \mathscr{F})$ is a function $\mu : \mathscr{F} \to \mathbb{R}^{\geq 0}$ such that $\mu(\emptyset) = 0$ and,

for each countable family $\{\Omega_i\}_{i\in I}$ of pairwise disjoint elements of $\mathscr{F}$, $\mu(\cup_{i\in I}\Omega_i) = \sum_{i\in I}\mu(\Omega_i)$. If $\mu(\Omega) \leq 1$, then we call μ a *sub-probability measure* and, if $\mu(\Omega) = 1$, then we call μ a *probability measure*. We say that μ is *discrete* measure over Ω if $\mathscr{F}$ is discrete. In this case, for each $X \subseteq \Omega$, $\mu(X) = \sum_{x\in X}\mu(\{x\})$ and we drop brackets whenever possible. For a set Ω, denote by $\mathrm{Disc}(\Omega)$ the set of discrete probability measures over Ω, and by $\mathrm{SubDisc}(\Omega)$ the set of discrete sub-probability measures over Ω. We call $X \subseteq \Omega$ the *support* of a measure μ if $\mu(\Omega \setminus X) = 0$; in particular, if μ is discrete, we denote by $\mathrm{Supp}(\mu)$ the minimum support set $\{\, x \in \Omega \mid \mu(x) > 0 \,\}$. Moreover, we denote by $\mu(\bot)$ the value $1 - \mu(\Omega)$ where $\bot \notin \Omega$, and by δ_x, for $x \in \Omega \cup \{\bot\}$, the *Dirac* measure such that for each $X \subseteq \Omega$, $\mu(X) = 1$ if $x \in X$, 0 otherwise. For a sub-probability measure μ, we also write $\mu = \{\, (x, p_x) \mid x \in \Omega, p_x = \mu(x) > 0 \,\}$.

A function $f\colon \Omega_1 \to \Omega_2$ is called a *measurable function* from $(\Omega_1, \mathscr{F}_1)$ to $(\Omega_2, \mathscr{F}_2)$ if the inverse image of f of any element of $\mathscr{F}_2$ is an element of $\mathscr{F}_1$. In this case, for a measure μ on $(\Omega_1, \mathscr{F}_1)$, we define the *image measure* of μ under f on $(\Omega_2, \mathscr{F}_2)$, denoted by $f(\mu)$, by: for each $X \in \mathscr{F}_2$, $f(\mu)(X) = \mu(f^{-1}(X))$, that is, the measure of X in $\mathscr{F}_2$ is the measure of the elements in $\mathscr{F}_1$ whose f-image is in X. Since f is measurable, then $f(\mu)$ is a well defined measure.

The lifting $\mathscr{L}(\mathscr{R})$ [JL91] of a relation $\mathscr{R} \subseteq X \times Y$ is defined as follows: for $\rho_X \in \mathrm{Disc}(X)$ and $\rho_Y \in \mathrm{Disc}(Y)$, $\rho_X\ \mathscr{L}(\mathscr{R})\ \rho_Y$ holds if there exists a *weighting function* $w\colon X \times Y \to [0,1]$ such that (1) $w(x,y) > 0$ implies $x\ \mathscr{R}\ y$, (2) $\sum_{y\in Y} w(x,y) = \rho_X(x)$, and (3) $\sum_{x\in X} w(x,y) = \rho_Y(y)$. When $\mathscr{R}$ is an equivalence relation on X, $\rho_1\ \mathscr{L}(\mathscr{R})\ \rho_2$ holds if for each $\mathscr{C} \in X/\mathscr{R}$, $\rho_1(\mathscr{C}) = \rho_2(\mathscr{C})$.

A Probabilistic Automaton (*PA*) $\mathscr{A}$ is a tuple $\mathscr{A} = (S, \bar{s}, \Sigma, D)$, where S is a countable set of *states*, $\bar{s} \in S$ is the *start state*, Σ is a countable set of *actions*, and $D \subseteq S \times \Sigma \times \mathrm{Disc}(S)$ is a *probabilistic transition relation*. The set Σ is divided in two sets H and E of internal (hidden) and external actions, respectively; we let s,t,u,v, and their variants with indices range over S, a, b range over actions, and τ range over hidden actions. We refer to the elements of $\mathscr{A}$ by S, $\bar{s}$, Σ, and D and we propagate indices and primes as expected: S'_i is the state set of the automaton $\mathscr{A}'_i$. In this work we consider only finite *PA*s, i.e., *PA*s such that sets S and D are finite.

A transition $tr = (s, a, \mu) \in D$, also denoted by $s \xrightarrow{a} \mu$, is said to *leave* from state s, to be *labelled* by a, and to *lead* to the *target* probability measure μ, also denoted by μ_{tr}. We denote by $src(tr)$ the *source* state s and by $act(tr)$ the *action* a. We also say that s enables action a, that action a is enabled from s, and that (s, a, μ) is enabled from s. Finally, we denote by $D(a)$ the set of transitions with action a, i.e., $D(a) = \{\, tr \in D \mid act(tr) = a \,\}$.

An *execution fragment* of a *PA* $\mathscr{A}$ is a finite or infinite sequence of alternating states and actions $\alpha = s_0 a_1 s_1 a_2 s_2 \ldots$ starting from a state s_0, also denoted by $first(\alpha)$, and, if the sequence is finite, ending with a state denoted by $last(\alpha)$, such that for each $i > 0$ there exists a transition $(s_{i-1}, a_i, \mu_i) \in D$ such that $\mu_i(s_i) > 0$. The *length* of α, denoted by $|\alpha|$, is the number of occurrences of actions in α. If α is infinite, then $|\alpha| = \infty$. Denote by $frags(\mathscr{A})$ the set of execution fragments of $\mathscr{A}$ and by $frags^*(\mathscr{A})$ the set of finite execution fragments of $\mathscr{A}$. An execution fragment α is a *prefix* of an execution fragment α', denoted by $\alpha \leqslant \alpha'$, if the sequence α is a prefix of the sequence α'. The *trace* of α, denoted by $trace(\alpha)$, is the sub-sequence of external actions of α; we denote by ε the empty trace. Similarly, we define $trace(a) = a$ for $a \in E$ and $trace(\tau) = \varepsilon$.

A *scheduler* for a *PA* $\mathscr{A}$ is a function $\sigma\colon frags^*(\mathscr{A}) \to \mathrm{SubDisc}(D)$ such that for each finite execution fragment α, $\mathrm{Supp}(\sigma(\alpha)) \subseteq \{\, tr \in D \mid src(tr) = last(\alpha) \,\}$. Given a scheduler σ and a finite execution fragment α, the probability measure $\sigma(\alpha)$ describes how transitions are chosen

to move on from $last(\alpha)$; with probability $\sigma(\alpha)(\bot)$ the computation terminates after α. A scheduler σ and a state s induce a probability measure $\mu_{\sigma,s}$ over execution fragments as follows. The basic measurable events are the cones of finite execution fragments, where the cone of α, denoted by C_α, is the set $\{\alpha' \in frags(\mathscr{A}) \mid \alpha \leqslant \alpha'\}$. The probability $\mu_{\sigma,s}$ of a cone C_α is recursively defined as:

$$\mu_{\sigma,s}(C_\alpha) = \begin{cases} 0 & \text{if } \alpha = t \text{ for a state } t \neq s, \\ 1 & \text{if } \alpha = s, \\ \mu_{\sigma,s}(C_{\alpha'}) \cdot \sum_{tr \in D(a)} \sigma(\alpha')(tr) \cdot \mu_{tr}(t) & \text{if } \alpha = \alpha' a t. \end{cases}$$

Standard measure theoretical arguments ensure that $\mu_{\sigma,s}$ extends uniquely to the σ-field generated by cones. We call the resulting measure $\mu_{\sigma,s}$ a *probabilistic execution fragment* of $\mathscr{A}$ and we say that it is generated by σ from s. Given a finite execution fragment α, we define $\mu_{\sigma,s}(\alpha)$ as $\mu_{\sigma,s}(\alpha) = \mu_{\sigma,s}(C_\alpha) \cdot \sigma(\alpha)(\bot)$, where $\sigma(\alpha)(\bot)$ is the probability of terminating the computation after α has occurred.

We say that there is a *weak combined transition* from $s \in S$ to $\mu \in \mathrm{Disc}(S)$ labelled by $a \in \Sigma$, denoted by $s \stackrel{a}{\Longrightarrow}_{\mathrm{c}} \mu$, if there exists a scheduler σ such that the following holds for the induced probabilistic execution fragment $\mu_{\sigma,s}$: (1) $\mu_{\sigma,s}(frags^*(\mathscr{A})) = 1$; (2) $trace(\mu_{\sigma,s}) = \delta_{trace(a)}$; and (3) $last(\mu_{\sigma,s}) = \mu$. In this case, we say that the weak combined transition $s \stackrel{a}{\Longrightarrow}_{\mathrm{c}} \mu$ is induced by σ. Note that we are using the fact that $trace(\cdot)$ and $last(\cdot)$ are measurable functions.

Albeit the definition of weak combined transitions is somewhat intricate, this definition is just the obvious extension of weak transitions on labelled transition systems to the setting with probabilities. See [Seg06] for more details on weak combined transitions.

Weak probabilistic bisimilarity [Seg95, Seg06] is of central importance for our considerations.

Definition 1 Let $\mathscr{A}_1, \mathscr{A}_2$ be two *PA*s. An equivalence relation $\mathscr{R}$ on the disjoint union $S_1 \uplus S_2$ is a *weak probabilistic bisimulation* if, for each pair of states $s,t \in S_1 \uplus S_2$ such that $s \mathscr{R} t$, if $s \stackrel{a}{\longrightarrow} \mu_s$ for some probability measure μ_s, then there exists μ_t such that $t \stackrel{a}{\Longrightarrow}_{\mathrm{c}} \mu_t$ and $\mu_s \mathscr{L}(\mathscr{R}) \mu_t$.

We say that $\mathscr{A}_1$ and $\mathscr{A}_2$ are weak probabilistic bisimilar if there exists a weak probabilistic bisimulation $\mathscr{R}$ on $S_1 \uplus S_2$ such that $\bar{s}_1 \mathscr{R} \bar{s}_2$. We denote the coarsest weak probabilistic bisimulation, called weak probabilistic bisimilarity, by $\approx$.

3 Weak Transition Construction as a Linear Programming Problem: An Overview

The main source of the worst case theoretical complexity of the decision algorithms [HT12, CS02] for *PA* weak probabilistic bisimulation is the recurring need to check for the existence of the weak combined transition. This is solved with an exponential algorithm in [CS02] and a polynomial algorithm in [HT12]. The latter approach takes inspiration from network flow problems: a weak combined transition $t \stackrel{a}{\Longrightarrow}_{\mathrm{c}} \mu_t$ of a *PA* $\mathscr{A}$ is described as an enriched flow problem in which the initial probability mass δ_t splits along internal transitions, and precisely one external transition with label $a \neq \tau$ for every stream, in order to reach μ_t. The enriched flow problem is then translated into a linear programming problem extended with *balancing constraints* that encode the need to respect transition probability measure. To describe the structure of the enriched

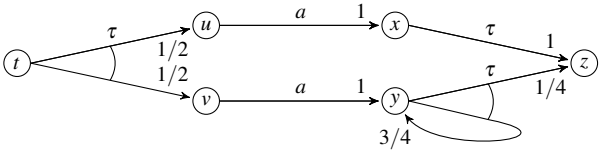

Figure 1: The Probabilistic Automaton $\mathscr{E}$

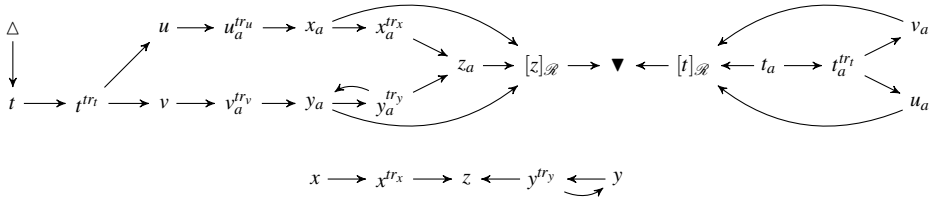

Figure 2: The network graph $G(t,a,\delta_z,\mathscr{R})$ for the automaton $\mathscr{E}$

linear programming problem, we first recall the definition of the network graph corresponding to a weak combined transition.

Definition 2 (cf. [HT12, Sect. 3.2]) Given a *PA* $\mathscr{A}$, a state t, an action a, a probability measure μ, and an equivalence relation $\mathscr{R}$ on S, the network graph $G(t,a,\mu,\mathscr{R}) = (V,E)$ relative to the weak combined transition $t \stackrel{a}{\Longrightarrow}_{\mathrm{c}} \mu_t$ is defined as follows: for $a \neq \tau$, the set of vertices is $V = \{\triangle,\blacktriangledown\} \cup S \cup S^{tr} \cup S_a \cup S_a^{tr} \cup (S/\mathscr{R})$ where $S^{tr} = \{\, v^{tr} \mid tr = v \stackrel{b}{\longrightarrow} \rho \in D, b \in \{a,\tau\} \,\}$, $S_a = \{\, v_a \mid v \in S \,\}$, and $S_a^{tr} = \{\, v_a^{tr} \mid v^{tr} \in S^{tr} \,\}$ and the set of arcs is $E = \{(\triangle,t)\} \cup \{\, (v_a,\mathscr{C}),(\mathscr{C},\blacktriangledown) \mid \mathscr{C} \in S/\mathscr{R}, v \in \mathscr{C} \,\} \cup \{\, (v,v^{tr}),(v^{tr},v'),(v_a,v_a^{tr}),(v_a^{tr},v_a') \mid tr = v \stackrel{\tau}{\longrightarrow} \rho \in D, v' \in \mathrm{Supp}(\rho) \,\} \cup \{\, (v,v_a^{tr}),(v_a^{tr},v_a') \mid tr = v \stackrel{a}{\longrightarrow} \rho \in D, v' \in \mathrm{Supp}(\rho) \,\}$.

We refer to the elements of $S \cup S_a$ as state nodes, of $\mathscr{T} = S^{tr} \cup S_a^{tr}$ as transition nodes, and of $S/\mathscr{R}$ as class nodes. We denote by $\mathscr{V}$ the set $V \setminus \{\triangle,\blacktriangledown\}$. When $a = \tau$ the definition of $G(t,\tau,\mu,\mathscr{R}) = (V,E)$ is similar: $V = \{\triangle,\blacktriangledown\} \cup S \cup S^{tr} \cup (S/\mathscr{R})$ and $E = \{(\triangle,t)\} \cup \{\, (v,\mathscr{C}),(\mathscr{C},\blacktriangledown) \mid \mathscr{C} \in S/\mathscr{R}, v \in \mathscr{C} \,\} \cup \{\, (v,v^{tr}),(v^{tr},v') \mid tr = v \stackrel{\tau}{\longrightarrow} \rho \in D, v' \in \mathrm{Supp}(\rho) \,\}$.

As an example of the construction of the network, consider the probabilistic automaton $\mathscr{E}$ given in Fig. 1 and the network graph $G(t,a,\delta_z,\mathscr{R})$ depicted in Fig. 2, where $\mathscr{R}$ is the equivalence relation inducing classes $\{t,u,v\}$ and $\{x,y,z\}$.

As pointed out in [HT12], the fact that the network admits a flow that respects the probability measure μ_t does not imply the existence of a corresponding weak combined transition, because the flow may not respect probability ratios. Therefore, the network is converted into a linear programming problem for which the feasibility is shown to be equivalent to the existence of the desired weak combined transition. The idea is to convert the flow network into the canonical LP problem and then add the balancing constraints that force the "flow" to split according to transition probability measures.

Definition 3 (cf. [HT12, Def. 6]) For $a \neq \tau$, the $LP(t,a,\mu,\mathscr{R})$ Linear Programming problem

associated to the network graph $(V,E) = G(t,a,\mu,\mathscr{R})$ is defined as follows:

$$
\begin{array}{ll}
\max \sum_{(x,y)\in E} -f_{x,y} & \\
\text{subject to} & \\
f_{u,v} \geq 0 & \text{for each } (u,v) \in E \\
f_{\triangle,t} = 1 & \\
f_{\mathscr{C},\blacktriangledown} = \mu(\mathscr{C}) & \text{for each } \mathscr{C} \in S/\mathscr{R} \\
\sum_{u\in\{x|(x,v)\in E\}} f_{u,v} - \sum_{u\in\{y|(v,y)\in E\}} f_{v,u} = 0 & \text{for each } v \in V \setminus \{\triangle,\blacktriangledown\} \\
f_{v^{tr},v'} - \rho(v') \cdot f_{v,v^{tr}} = 0 & \text{for each } tr = v \xrightarrow{\tau} \rho \in D \text{ and } v' \in \mathrm{Supp}(\rho) \\
f_{v_a^{tr},v'_a} - \rho(v') \cdot f_{v_a,v_a^{tr}} = 0 & \text{for each } tr = v \xrightarrow{\tau} \rho \in D \text{ and } v' \in \mathrm{Supp}(\rho) \\
f_{v_a^{tr},v'_a} - \rho(v') \cdot f_{v,v_a^{tr}} = 0 & \text{for each } tr = v \xrightarrow{a} \rho \in D \text{ and } v' \in \mathrm{Supp}(\rho)
\end{array}
$$

In the LP problem described in Def. 3, the objective function maximises the total sum of negated flow routed along the arcs of the network. In fact, the total flow is described as the sum of negated flow variables which are positive themselves and this prevents sending a large amount of flow over disconnected components of the network or over cycles that can be ignored. Furthermore, in the LP, there are two different sets of constraints. The first set is the ordinary set of flow conservation constraints which require the total flow incoming and outgoing a node of the network to be equal. The second set is the set of balancing constraints that require the entering amount of flow to a transition node to be distributed based on probabilities assigned to the outgoing arcs.

It is easy to observe that the $LP(t,a,\mu,\mathscr{R})$ LP problem has size that is quadratic in the size $N = \max\{|S|,|D|\}$ of $\mathscr{A}$: the number of variables is at most $3N^2+5N+1$ while the number of constraints is at most $6N^2+11N+2$. It is worthwhile to spell out the amount of transition, state, and class nodes of the network $G(t,a,\mu,\mathscr{R})$: there are at most $2|D|$ transition nodes, at most $2|S|$ state nodes, and at most $|S|$ class nodes.

The equivalence of LP problem and weak transition is formalised by Theorem 8 and Corollary 9(1) of [HT12]:

Proposition 1 *A weak combined transition $t \stackrel{a}{\Longrightarrow}_{\mathrm{c}} \mu_t$ such that $\mu\ \mathscr{L}(\mathscr{R})\ \mu_t$ exists if and only if the LP problem $LP(t,a,\mu,\mathscr{R})$ has a feasible solution.*

4 Efficiency of Solving the LP Problem

The analysis of the $LP(t,a,\mu,\mathscr{R})$ LP problem formalised in [HT12, Thm. 7] considers only the theoretical complexity class the problem belongs to while it does not address how efficiently the LP problem can indeed be solved. In order to arrive at a precise characterisation of the problem efficiency, we establish a connection between the probabilistic verification community and the combinatorial optimisation community. We first intend to make precise the theoretical worst case running time needed to solve the LP problem. Then we focus on the practical aspects and, by modifying the LP problem as a flow network problem, we show that the underlying network structure can be exploited to solve the LP efficiently via an algorithm in the network optimisation setting.

4.1 Efficiency of Solving the LP Problem: Theory

Deciding the existence of a weak combined transition in a probabilistic automaton can be done in polynomial time [HT12, Thm. 7 and 8]. With the aim to refine this result, for instance finding the sufficient conditions to have strong polynomiality, we discuss the problem in the context of some restricted classes of probabilistic automata.

Definition 4 An algorithm runs in *strongly polynomial time* if the number of operations in the arithmetic model of computation is not dependent on the size of the input instances and it can be bounded by a polynomial of the number of input instances.

The linear programming problem is a prominent problem for which it is not known if it admits a strongly polynomial time algorithm. Using polynomial algorithms for solving linear programming problems, e.g., ellipsoid method [Kha79] or interior point method [Kar84] as a subroutine, Tardos [Tar86] showed that combinatorial LPs can be solved strongly polynomially. Further, Adler et al. [AC91] generalized this class of LPs and showed that for the LPs whose constraint set is a pre-Leontief substitution system, the optimal solution can be computed in strongly polynomial time.

We consider these two sub-classes of probabilistic automata:

Definition 5 Given a family of *PA*s $\{\mathscr{A}_k\}_{k\in\mathbb{N}}$, we say that $\{\mathscr{A}_k\}_{k\in\mathbb{N}}$ is

- *rational* if for each $k \in \mathbb{N}$ it holds that for each $(s,a,\mu) \in D_k$ and $v \in \mathrm{Supp}(\mu)$, $\mu(v) \in \mathbb{Q}$,
- *polynomially representable* if $\{\mathscr{A}_k\}_{k\in\mathbb{N}}$ is rational and there exists a polynomial P such that for each $k \in \mathbb{N}$ and each transition $s \xrightarrow{a} \mu \in D_k$, it holds that $\mathrm{lcm}\{\, d_v \mid v \in \mathrm{Supp}(\mu), \mu(v) = \frac{n_v}{d_v} \in \mathbb{Q}, \frac{n_v}{d_v} \text{ is irreducible} \,\} \leq P(N_k)$ where $N_k = \max\{|S_k|, |D_k|\}$.

We extend the above notions to each $\mathscr{A} \in \{\mathscr{A}_k\}_{k\in\mathbb{N}}$ in the obvious way; for instance, we say that $\mathscr{A}$ is polynomially representable if $\mathscr{A}$ is part of a family that is polynomially representable.

Rational automata We start our analysis with the class of rational *PA*s and we look for the best achievable worst case complexity of solving the LP problem $LP(t,a,\mu,\mathscr{R})$ via a reformulation that reduces the size of $LP(t,a,\mu,\mathscr{R})$. This is important because the worst case complexity of any LP solver depends on the number of variables and constraints. Therefore, having a smaller LP assists us to reach a better worst case complexity in particular for the LP problems corresponding to large scale probabilistic automata. To reach our goal, we modify the network provided in Def. 2 and we reformulate the original LP problem on the basis of these changes.

Consider the network $G(t,a,\mu,\mathscr{R})$ and let $\mathscr{G}(t,a,\mu,\mathscr{R})$ be a directed network which is generated from the network $G(t,a,\mu,\mathscr{R})$ by removing the source node $\triangle$ and the sink node $\blacktriangledown$; let $\mathscr{V} = V \setminus \{\triangle, \blacktriangledown\}$ and $\mathscr{E} = E \setminus (\{(\triangle,t)\} \cup \{\,(\mathscr{C}, \blacktriangledown) \mid \mathscr{C} \in S/\mathscr{R}\,\})$ be the set of vertices and directed arcs of $\mathscr{G}(t,a,\mu,\mathscr{R})$, respectively. Moreover, let $\bar{\mathscr{E}} \subseteq \mathscr{E}$ be the set $\bar{\mathscr{E}} = \{\,(v^{tr}, v'), (v^{tr}_a, v'_a) \mid tr = v \xrightarrow{\tau} \rho \in D, v' \in \mathrm{Supp}(\rho)\,\} \cup \{\,(v^{tr}_a, v'_a) \mid tr = v \xrightarrow{a} \rho \in D, v' \in \mathrm{Supp}(\rho)\,\}$. Then, we define $\rho_{ij} = \mu_{tr}(v')$ as the proportionality coefficient corresponding to the arc $(i,j) \in \bar{\mathscr{E}}$ where $(i,j) = (v^{tr}, v')$ or $(i,j) = (v^{tr}_a, v'_a)$. Since in both original and modified networks each arc in $\bar{\mathscr{E}}$ belongs to a single transition, the corresponding proportional coefficient is uniquely determined.

For each node $u \in \mathscr{V}$, let b_u be a supply/demand value, that is, if $b_u > 0$ the node u is a supply node and if $b_u < 0$ the node u is a demand node. For the network $\mathscr{G}(t,a,\mu,\mathscr{R})$, we define b_u for each node $u \in \mathscr{V}$ so as to take value 1 if $u = t$, value $-\mu(\mathscr{C})$ if $u = \mathscr{C} \in S/\mathscr{R}$ and 0 otherwise. It is immediate to see that $\sum_{u\in\mathscr{V}} b_u = 0$. This fact can be seen as a feasibility condition in the corresponding flow network [AMO93]. For $s \in \mathscr{T}$, assume A_s to be the set of all arcs in the node-arc incidence matrix A that should have proportional flow. We define $\tilde{A}$ to be the subset of arcs in A that do not belong to any set A_s for $s \in \mathscr{T}$. More precisely, $\tilde{A} = A \setminus \bigcup_{s\in\mathscr{T}} A_s$. Based on the definitions, the $LP(t,a,\mu,\mathscr{R})$ LP problem can be reformulated as follows:

$$
\begin{array}{lll}
\text{LP1: min} & \sum_{(i,j)\in\mathscr{E}} f_{ij} & \\
\text{s.t.} & \sum_{(i,j)\in\mathscr{E}} f_{ij} - \sum_{(j,i)\in\mathscr{E}} f_{ji} = b_i & \text{for each } i \in \mathscr{V} \\
& \frac{f_{ij}}{\rho_{ij}} \text{ are all equal} & (i,j) \in A_s, s \in \mathscr{T} \\
& f_{ij} \geq 0 & \text{for each } (i,j) \in \mathscr{E}
\end{array}
$$

Lemma 1 *The $LP(t,a,\mu,\mathscr{R})$ LP problem and LP1 are equivalent.*

By assuming the unit flow cost $c_{ij} = 1$ for each arc $(i,j) \in \mathscr{E}$, the objective of this problem is to minimise the total cost of routing the flow on network arcs subject to the ordinary flow conservation constraints, the proportional flow constraints corresponding to the balancing constraints of the original LP problem, and the arc flow lower bounds.

It is worthwhile to note that there exists a proportional flow set for each transition node in the network and that each arc may belong to at most one proportional flow set. The flow on the arcs in each of these flow proportional sets can be regarded as a single decision variable. Using this intuition, let a_{ij} denotes the column corresponding to the arc (i,j) in the node-arc incidence matrix of the network $\mathscr{G}(t,a,\mu,\mathscr{R})$ and let $a_s = \sum_{(i,j)\in A_s} \rho_{ij} a_{ij}$ for each $s \in \mathscr{T}$. We denote by a_{ij}^k and a_s^k the k-th component of the vectors a_{ij} and a_s, respectively. By using the new notations, LP1 can be reformulated as the following LP problem which can be seen as an adaption of the LP in [BF12].

$$
\begin{array}{lll}
\text{LP2: min} & \sum_{(i,j)\in\tilde{A}} f_{ij} + \sum_{s\in\mathscr{T}} f_s & \\
\text{s.t.} & \sum_{(i,j)\in\tilde{A}} f_{ij} - \sum_{(j,i)\in\tilde{A}} f_{ji} + \sum_{s\in\mathscr{T}} a_s^i f_s = b_i & \text{for each } i \in \mathscr{V} \\
& f_{ij} \geq 0 & \text{for each } (i,j) \in \tilde{A} \\
& f_s \geq 0 & \text{for each } s \in \mathscr{T}
\end{array}
$$

Lemma 2 *LP1 and LP2 are equivalent.*

Since both LP1 and LP2 are equivalent to the $LP(t,a,\mu,\mathscr{R})$ LP problem, we exploit the structure of LP2 to improve the efficiency of checking for a solution of $LP(t,a,\mu,\mathscr{R})$, so we improve as well the complexity of deciding probabilistic weak bisimulation since, similarly to the other combinatorial optimisation problems, it is very important to have a clear intuition about the best worst case complexity of the problem under consideration. Amongst all available versions of polynomial algorithms for solving a linear programming problem, we use a state-of-the-art polynomial interior point method [Ans99] which does perform well in practice and also, to the best of the knowledge of the authors, it has the best worst case complexity. Implementing the mentioned interior point method directly on the original LP problem would not give us the best worst

case complexity since the running time of this method depends on the number of variables as well as on the bit size of the problem: the number of variables in $LP(t,a,\mu,\mathscr{R})$ is $\mathscr{O}(N^2)$ while the number of variables in LP2 is instead a linear order of the number of nodes in the network which is $\mathscr{O}(N)$, where N is the size of the automaton $\mathscr{A}$. This is an outstanding reduction in the worst case complexity especially for large probabilistic automata which is achieved by taking advantage of the LP2 formulation of the original $LP(t,a,\mu,\mathscr{R})$ LP problem.

Theorem 1 *Consider a rational PA $\mathscr{A}$, the action a, the probability measure $\mu \in \mathrm{Disc}(S)$, the equivalence relation $\mathscr{R}$ on S and a state $t \in S$. Let $N = \max\{|S|,|D|\}$. Then, checking the feasibility of the $LP(t,a,\mu,\mathscr{R})$ LP problem can be done in $\mathscr{O}(\frac{N^3}{\ln N}L)$ where L is the bit size of the problem.*

Since the best worst case running time essentially depends on the type of the polynomial algorithm used to solve the LP, any improvement in the LP solution leads to a better worst case complexity of the weak bisimulation decision problem.

It is worthwhile to note that, when the rational *PA* $\mathscr{A}$ is polynomially representable, then each probability value can be represented with a polynomial number of bits, thus the complexity stated in Theorem 1 reduces to $\mathscr{O}(\frac{N^3}{\ln N}P(N))$ for some polynomial P in N, i.e., it is strongly polynomial.

Non-rational automata The class of rational probabilistic automata, as far as the authors know, encompasses all *PA*s that have appeared in practical applications. One may nevertheless consider relevant also the analysis of *PA*s with real valued probabilities.

One possible way to represent LP problems with real data is to use a model of computation that can perform any elementary arithmetic operation in constant time, regardless of the type of the operand. Another option is to encode reals as finite precision rationals. For a survey on the theory of computation over real numbers we refer the reader to [BSS89, Bel01].

When using finite precision rationals, the representation of the *PA* must become approximate, and still the size needed for this can no longer be guaranteed to be bounded by a polynomial. If assuming the rational approximation scheme being employed by the user, we are back to the rational setting for the LP solution process, and it is left to the user to interpret the outcome on the real valued *PA*. If instead the algorithm performs the approximation prior to solving the induced LP problem, the user may in general be lacking knowledge on how to extend the result back to the original real valued *PA*.

4.2 Efficiency of Solving the LP Problem: Practice

We now consider the practical efficiency of deciding probabilistic automata weak bisimulation. To this aim, we first concentrate on the available algorithms that can be employed to solve the problem and we show that, by using the underlying structure of the problem, checking the feasibility of the LP problem can be done more efficiently than just hiring a general purpose LP solver. Afterwards we discuss other methods that are more efficient but that are not suitable for solving the $LP(t,a,\mu,\mathscr{R})$ LP problem.

Modelling the described decision problem as a linear programming problem allows practitioners to use the omnipresent simplex method as an extremely efficient computational tool. It

is worthwhile to note that the efficiency of the simplex method is measured as the number of pivots needed to solve the LP problem. Moreover, practical experiments show that although this method is highly efficient, there exist problems that require an exponential number of pivots. This means that the worst case theoretical complexity of the simplex method is exponential time [KM72]. However, computational experience on thousands of real-world problems reveals that the number of pivots is usually polynomial in the number of variables and of constraints. For a comprehensive survey on the efficiency of the simplex method, we refer the interested reader to [Sha87].

Since the LP1 problem is a minimum cost flow problem on the network $\mathscr{G}(t,a,\mu,\mathscr{R})$ extended with an additional set of proportional flow constraints, we consider the usage of efficient algorithms that solve the problem directly on the flow network itself. One such algorithm is the network simplex algorithm [BF12] for the minimum cost proportional flow problem that improves the per iteration running time considerably with respect to the simplex method, as long as the number of nodes in the network is at least an order of magnitude larger than the number of side constraints in the LP [Cal02, MSJ11, BF12, MSJ13]. So, the network simplex algorithm is a candidate for improving the running time required to solve LP1. However, the number of side constraints coincides with the number of transition nodes in the LP1 problem. Since the number of transitions in the automaton is usually larger than the number of states, we have that the number of side constraints is linear in the number of nodes, and thus the above assumption is not satisfied. Still, a more accurate analysis tells us that the per iteration running time of both methods is in the same complexity class, as shown in Table 1. Since it is known that the network simplex algorithm without side constraints is better than the simplex method [AMO93], it is worthwhile to consider its usage in an implementation.

Up to now, we have shown that the simplex method and the network simplex algorithm [BF12] are quite competitive in solving the LP1. However, the embedded flow network structure in the LP1 is the source of our motivation to emphasis on the practical efficiency of the network simplex algorithm in solving the LP1. On the other hand, taking the dual of the equivalent LP2, allows us to deal with a small sized LP which is still similar to a well known combinatorial problem by itself. To clarify the point, consider the dual DLP2 of the LP2 problem:

$$\begin{aligned}
\text{DLP2: max} \quad & \sum_{s\in\mathscr{V}} b_s\pi_s \\
\text{s.t.} \quad & \pi_i - \pi_j \leq 1 \qquad \text{for each } (i,j)\in\tilde{A} && (1)\\
& \sum_{t\in\mathscr{V}} a_s^t\pi_t \leq 1 \qquad \text{for each } s\in\mathscr{T} && (2)
\end{aligned}$$

The number of variables and constraints in DLP2 is $\mathscr{O}(N)$ as well as the number of constraints in LP1 whose number of variables is instead in $\mathscr{O}(N^2)$, so DLP2 is smaller than the original LP1. This property has a deep impact when the number of transitions is remarkably more than the number of states in the network. The dual LP can be solved very efficiently using a state-of-the-art variant of the interior point method [Ans99]. This algorithm is a preconditioned conjugate gradient (PCG) method combined with a partial updating procedure which works excellently in practice as well. The algorithm is available in the famous software tools CPLEX and LOQO. Furthermore, DLP2 has itself a combinatorial structure, that is, it is the dual of the well known shortest path problem although with additional side constraints. Taking the advantage of this

Table 1: Complexity comparison

		$LP(t,a,\mu,\mathscr{R})$	LP1	LP2	DLP2
Variables/Arcs	n	$\mathscr{O}(N^2)$	$\mathscr{O}(N^2)$	$\mathscr{O}(N)$	$\mathscr{O}(N)$
Constraints	m	$\mathscr{O}(N^2)$	$\mathscr{O}(N)$	$\mathscr{O}(N)$	$\mathscr{O}(N)$
Proportional Flow Sets	p	not applicable	$\mathscr{O}(N)$	not applicable	not applicable
Free Arcs	n'	$\mathscr{O}(N^2)$	$\mathscr{O}(N)$	$\mathscr{O}(N)$	$\mathscr{O}(N)$
Simplex Method	$\mathscr{O}(nm)$	$\mathscr{O}(N^4)$	$\mathscr{O}(N^3)$	$\mathscr{O}(N^2)$	$\mathscr{O}(N^2)$
Network Simplex Algorithm [MSJ13]	$\mathscr{O}(n'+mp+p^3)$	not applicable	$\mathscr{O}(N^3)$	not applicable	not applicable

combinatorial property may help in the design of a more efficient algorithm to solve the problem.

Table 1 summarises the size of the proposed LP problems and the per-iteration complexity of the simplex method and of the network simplex algorithm.

Since each variable in the LP problem corresponds to an arc in the network, we identify by n both variables and arcs; on networks, each arc either belongs to a proportional flow set or is a free arc. The computational comparison of three LPs is based on N, the size of the automaton $\mathscr{A}$, and the smaller problem is DLP2 that is at least one degree smaller than other LPs, making it the convenient input for the LP solver. Now, consider the LP1 problem where both the simplex method and the networks simplex algorithm can be used where the latter is faster than the former provided that $p \in \mathscr{O}(\sqrt{m})$ [MSJ11]. Although, different variants of the simplex method can terminate using the anti-cycling rules [BT97]; however, they may run in an exponential number of iterations. In spite of that, in practice, the simplex algorithm can terminate in a polynomial number of iterations [KM72]. Therefore, it makes it very important to try to reduce the per-iteration time complexity of the simplex method. Now, consider the LP1 problem where both the simplex method and the networks simplex algorithm can be used where the latter is faster than the former provided that $p \in \mathscr{O}(\sqrt{m})$ [MSJ13]. Despite the fact that this requirement is not satisfied by LP1, both algorithms are worthwhile to be considered for an implementation since the per-iteration complexity is in the same complexity class for both methods.

5 Conclusion

This paper considers deciding *PA* weak bisimulation which is known to be polynomial [HT12]. We showed that the decision problem can be solved strongly polynomially for polynomially representable *PA*s. After a survey of available polynomial algorithms to solve an LP problem, we established an upper bound on the worst case complexity of the decision problem for general *PA*. For the practical efficiency, we demonstrated that a small modification of the LP discussed in [HT12] enables taking advantage of the underlying network structure to improve the practical efficiency of solving the problem. As such, the results of this paper allow a number of directions for further research: the first and foremost next step is a comprehensive empirical evaluation and comparison of the methods discussed in this paper on real world *PA*s. Furthermore, the network simplex algorithm specialised for the minimum cost flow problem with additional side constraints can be seen itself as another research direction. In fact, designing a new data structure to be able to deal with a large number of additional side constraints has not only a very important contribution in the theoretical setting but also it improves the practical efficiency of the decision

problem under our consideration. Finally, we intend to work on the dual LP itself. The latter has similarity to the combinatorial shortest path problem [AMO93]. This may lead to a very efficient combinatorial algorithm to solve the decision problem.

Acknowledgements. The authors would like to appreciate professor James B. Orlin (Massachusetts Institute of Technology) and Dr. Khaled Elbassioni (Masdar Institute of Science and Technology) for their invaluable assistance to improve the results of the paper. This work has been supported by the DFG as part of the SFB/TR 14 "Automatic Verification and Analysis of Complex Systems" (AVACS), by the DFG/NWO Bilateral Research Programme ROCKS, and by the European Union Seventh Framework Programme under grant agreement no. 295261 (MEALS) and 318490 (SENSATION). Part of this work has been done when Andrea Turrini was at Saarland University supported by the Cluster of Excellence "Multimodal Computing and Interaction" (MMCI), part of the German Excellence Initiative.

Bibliography

[AC91] I. Adler, S. T. Cosares. A Strongly Polynomial Algorithm for a Special Class of Linear Programs. *Operations Research* 39(6):955–960, 1991.

[AMO93] R. K. Ahuja, T. J. Magnanti, J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[Ans99] K. M. Anstreicher. Linear Programming in $\mathcal{O}(\frac{n^3}{\ln n}L)$ Operations. *SIAM J. on Optimization* 9(4):803–812, 1999.

[Bel01] P. A. Beling. Exact Algorithms for Linear Programming over Algebraic Extensions. *Algorithmica* 31(4):459–478, 2001.

[BF12] U. Bahçeci, O. Feyzioğlu. A network simplex based algorithm for the minimum cost proportional flow problem with disconnected subnetworks. *Optimization Letters* 6:1173–1184, 2012.

[BHH+09] E. Böde, M. Herbstritt, H. Hermanns, S. Johr, T. Peikenkamp, R. Pulungan, J. Rakow, R. Wimmer, B. Becker. Compositional Dependability Evaluation for STATEMATE. *ITSE* 35(2):274–292, 2009.

[BSS89] L. Blum, M. Shub, S. Smale. On a theory of computation and complexity over the real numbers; *NP*-completeness, recursive functions and universal machines. *Bullettin of the American Mathematical Society* 21(1), 1989.

[BT97] D. Bertsimas, J. N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1997.

[Cal02] H. I. Calvete. Network simplex algorithm for the general equal flow problem. *European J. Operational Research*, pp. 585–600, 2002.

[CGM+96] G. Chehaibar, H. Garavel, L. Mounier, N. Tawbi, F. Zulian. Specification and Verification of the PowerScale® Bus Arbitration Protocol: An Industrial Experiment with LOTOS. In *FORTE*. Pp. 435–450. 1996.

[CHLS09] N. Coste, H. Hermanns, E. Lantreibecq, W. Serwe. Towards Performance Prediction of Compositional Models in Industrial GALS Designs. In *CAV*. Pp. 204–218. 2009.

[CS02] S. Cattani, R. Segala. Decision Algorithms for Probabilistic Bisimulation. In *CONCUR*. LNCS 2421, pp. 371–385. 2002.

[Der70] C. Derman. *Finite State Markovian Decision Processes*. Academic Press, Inc., 1970.

[EHS+13] C. Eisentraut, H. Hermanns, J. Schuster, A. Turrini, L. Zhang. The Quest for Minimal Quotients for Probabilistic Automata. In *TACAS*. LNCS 7795, pp. 16–31. 2013.

[EHZ10a] C. Eisentraut, H. Hermanns, L. Zhang. Concurrency and Composition in a Stochastic World. In *CONCUR*. LNCS 6269, pp. 21–39. 2010.

[EHZ10b] C. Eisentraut, H. Hermanns, L. Zhang. On Probabilistic Automata in Continuous Time. In *LICS*. Pp. 342–351. 2010.

[Han91] H. A. Hansson. *Time and Probability in Formal Design of Distributed Systems*. PhD thesis, Department of Computer Systems, Uppsala University, 1991.

[HK95] R. V. Helgason, J. L. Kennington. Primal simplex algorithms for minimum cost network flows. In *Network Models*. Handbooks in Operations Research and Management Science 7, chapter 2, pp. 85–113. Elsevier, 1995.

[HK00] H. Hermanns, J.-P. Katoen. Automated Compositional Markov Chain Generation for a Plain-old Telephone System. *Science of Computer Programming* 36(1):97–127, 2000.

[HKNP06] A. Hinton, M. Kwiatkowska, G. Norman, D. Parker. PRISM: A Tool for Automatic Verification of Probabilistic Systems. In *TACAS*. LNCS 3920, pp. 441–444. 2006.

[HT12] H. Hermanns, A. Turrini. Deciding Probabilistic Automata Weak Bisimulation in Polynomial Time. In *FSTTCS*. Pp. 435–447. 2012.

[JL91] B. Jonsson, K. G. Larsen. Specification and Refinement of Probabilistic Processes. In *LICS*. Pp. 266–277. 1991.

[Kar84] N. Karmarkar. A new polynomial-time algorithm for Linear Programming. *Combinatorica* 4(4):373–395, 1984.

[Kha79] L. G. Khachyan. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady* 20(1):191–194, 1979.

[KKZJ07] J.-P. Katoen, T. Kemna, I. S. Zapreev, D. N. Jansen. Bisimulation minimisation mostly speeds up probabilistic model checking. In *TACAS*. LNCS 4424, pp. 76–92. 2007.

[KM72] V. Klee, G. J. Minty. How good is the simplex algorithm? In *Inequalities*. Volume III, pp. 159–175. Defense Technical Information Center, 1972.

[MSJ11] D. R. Morrison, J. J. Sauppe, S. H. Jacobson. A Network Simplex Algorithm for the Equal Flow Problem on a Generalized Network. *INFORMS J. on Computing*, pp. 1–11, 2011.

[MSJ13] D. R. Morrison, J. J. Sauppe, S. H. Jacobson. An algorithm to solve the proportional network flow problem. *Optimization Letters*, 2013.

[PLS00] A. Philippou, I. Lee, O. Sokolsky. Weak Bisimulation for Probabilistic Systems. In *CONCUR*. LNCS 1877, pp. 334–349. 2000.

[Seg95] R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, MIT, 1995.

[Seg06] R. Segala. Probability and Nondeterminism in Operational Models of Concurrency. In *CONCUR*. LNCS 4137, pp. 64–78. 2006.

[Sha87] R. Shamir. The Efficiency of the Simplex Method: A Survey. *Management Science* 33(3):301–334, 1987.

[Tar86] E. Tardos. A Strongly Polynomial Algorithm to Solve Combinatorial Linear Programs. *Operations Research* 34(2):250–256, 1986.

[Var85] M. Y. Vardi. Automatic Verification of Probabilistic Concurrent Finite-State Programs. In *FOCS*. Pp. 327–338. 1985.