Quantitative Multi-Objective Verification for Probabilistic Systems

Vojtěch Forejt\textsuperscript{1}, Marta Kwiatkowska\textsuperscript{1},
Gethin Norman\textsuperscript{2}, David Parker\textsuperscript{1}
and Hongyang Qu\textsuperscript{1}

\textsuperscript{1} University of Oxford
\textsuperscript{2} University of Glasgow

TACAS’11, Saarbrücken, March 2011
Overview

- **Verification of probabilistic systems**
  - probabilistic automata (or Markov decision processes)
  - quantitative verification of temporal logic specifications

- **Multi-objective quantitative verification**
  - formalise trade-offs between several different objectives
  - probabilistic \( \omega \)-regular properties & expected reward (or cost)
  - verification, achievability & numerical queries
  - flexible property specification, efficient model checking

- **Controller synthesis**
  - synthesis of optimal adversaries/schedulers for MDPs (or PAs)

- **Compositional probabilistic verification**
  - assume-guarantee framework for probabilistic automata
Overview

- **Verification of probabilistic systems**
  - probabilistic automata (or Markov decision processes)
  - quantitative verification of temporal logic specifications

- **Multi-objective quantitative verification**
  - formalise trade-offs between several different objectives
  - probabilistic ω-regular properties & expected total reward
  - verification, achievability & numerical queries
  - flexible property specification, efficient model checking

- **Controller synthesis**
  - synthesis of optimal adversaries/schedulers for PAs/MDPs

- **Compositional probabilistic verification**
  - assume-guarantee framework for probabilistic automata
Probabilistic automata (PAs)

- Model nondeterministic as well as probabilistic behaviour
  - very similar to Markov decision processes (MDPs)

- A probabilistic automaton is a tuple \( M = (S, s_{\text{init}}, \alpha_M, \delta_M) \):
  - \( S \) is the state space
  - \( s_{\text{init}} \in S \) is the initial state
  - \( \alpha_M \) is the action alphabet
  - \( \delta_M \subseteq S \times \alpha_M \times \text{Dist}(S) \) is the transition probability relation

- Augment with (action-based) reward structures \( \rho \)
  - \( \rho : \alpha_\rho \rightarrow \mathbb{R}_{>0} \) (where \( \alpha_\rho \subseteq \alpha_M \))
  - (to model time, energy consumption, …)

- Parallel composition: \( M_1 \parallel M_2 \)
  - CSP style – synchronise over common actions [Segala]
Running example

- **Two components, each a probabilistic automaton:**
  - $M_m$: a machine executing 2 tasks
  - $M_c$: a possible controller for the machine

- **Example reward structures:**
  - $\rho_{\text{time}}$ = {fast $\rightarrow$ 1, slow $\rightarrow$ 3, on $\rightarrow$ 5}
  - $\rho_{\text{pow}}$ = {fast $\rightarrow$ 20, slow $\rightarrow$ 10, on $\rightarrow$ 2}
To reason formally about PAs, we use adversaries
- also called “schedulers”, “strategies”, “policies”, ...
- an adversary $\sigma$ resolves nondeterminism in a PA $M$
- makes a (possibly randomised) choice, based on history
- induces probability measure $Pr_{M|\sigma}$ over (infinite) paths in $M$

Probabilistic (linear-time) properties
- we focus on action-based, $\omega$-regular (e.g. LTL) properties
- e.g. $\Diamond done \land \Box \neg off$ – “eventually finish, without switching off”
- $Pr_{M|\sigma}(\phi) =$ probability of $\phi$ being true under adversary $\sigma$

Reward-based properties
- we focus on expected total reward properties
- $ExpTot_{M|\sigma}(\rho) =$ expected sum of rewards $\rho$ over paths wrt. $Pr_{M|\sigma}$
- e.g. “expected total time/energy/cost/... to complete”
• **Probabilistic model checking** (e.g. using LTL, …)
  – usually quantify over all adversaries \( \sigma \in \text{Adv}_M \)
  – e.g. \( M \models P_{\geq p} [\phi] \iff \Pr_{M}^{\sigma} (\phi) \geq p \) for all \( \sigma \in \text{Adv}_M \)
  – corresponds to best-/worst-case behaviour analysis

• **Or, in a more quantitative fashion, just compute:**
  – e.g. \( \Pr_{M}^{\text{min}} (\phi) = \inf \{ \Pr_{M}^{\sigma} (\phi) \mid \sigma \in \text{Adv}_M \} \)
  – similarly for \( \Pr_{M}^{\text{max}} (\phi), \text{ExpTot}_{M}^{\text{min}} (\rho), \text{ExpTot}_{M}^{\text{max}} (\rho) \)

• **Reduces to:** graph-based analysis + linear program
  – for case of LTL \( \phi \), on (synchronous) PA-automaton product
  – only need to consider deterministic (pure) adversaries
  – efficient: complexity is polynomial in \( |M| \) (but 2EXP in \( |\phi| \))
    • in practice, for scalability, often approximate (e.g. value iter.)
  – tools available: PRISM, Liquor, ProbDiVinE, RAPTURE, PASS, …
Running example

- **Two components, each a probabilistic automaton:**
  - $M_m$: a machine executing 2 tasks
  - $M_c$: a possible controller for the machine

- **Example properties for $M = M_c \parallel M_m$**
  - $Pr_M^{\text{min}}(\Diamond \text{done})$ – “minimum probability of termination”
  - $\text{ExpTot}_M^{\text{max}}(\rho_{\text{time}})$ – “maximum expected execution time”
Running example

- **Two components, each a probabilistic automaton:**
  - $M_m$: a machine executing 2 tasks
  - $M_c$: a possible controller for the machine

Example properties for $M = M_c \parallel M_m$

- $\Pr_{M}^{\text{min}}(\Diamond \text{done}) = 1$ so $M \models P_{\geq 1} [\Diamond \text{done}]
- \text{ExpTot}_{M}^{\text{max}}(\rho_{\text{time}}) = 3.1666\ldots$ so $M \models R_{<3.2} [\rho_{\text{time}}]$
Overview

• Verification of probabilistic systems
  – probabilistic automata (or Markov decision processes)
  – quantitative verification of temporal logic specifications

• Multi-objective quantitative verification
  – formalise trade-offs between several different objectives
  – probabilistic $\omega$-regular properties & expected total reward
  – verification, achievability & numerical queries
  – flexible property specification, efficient model checking

• Controller synthesis
  – synthesis of optimal adversaries/schedulers for PAs/MDPs

• Compositional probabilistic verification
  – assume-guarantee framework for probabilistic automata
Quantitative multi-objective properties

- Analyse trade-offs between multiple quantitative objectives
  - e.g. “expected send time vs. expected power consumption”

- Quantitative multi-objective (qmo) properties $\Psi$
  - boolean combinations of probabilistic/reward predicates, i.e.
    - $\Psi :: = \text{true} | \Psi \land \Psi | \Psi \lor \Psi | \neg \Psi | [\phi]_p | [\rho]_r$
      - where $\phi$ is an $\omega$-regular (e.g. LTL) property, $p \in [0,1]$, $\rho$ is a reward structure, $r \in \mathbb{R}_{\geq 0}$ and $\sim \in \{<,\leq,\geq,>\}$
    - example: $[\diamond \text{done}]_{\geq 1} \land [\rho_{\text{pow}}]_{\leq 30}$

- Satisfaction with respect to both PA $M$ and adversary $\sigma$
  - $M,\sigma \models [\phi]_p \iff \Pr_\sigma M (\phi) \sim p$
  - $M,\sigma \models [\rho]_r \iff \text{ExpTot}_\sigma M (\rho) \sim r$
  - obvious semantics for logical connectives…
    - e.g. $M,\sigma \models \Psi_1 \lor \Psi_2 \iff M,\sigma \models \Psi_1$ or $M,\sigma \models \Psi_2$
Quantitative multi-objective queries

- 3 types of queries for a qmo-property $\Psi$:
  
  - **Verification queries**: $M \models_{(\forall)} \Psi$
    - is $M,\sigma \models \Psi$ satisfied for all adversaries $\sigma$ of $M$?
    - also: $M \models_{(\forall^{\text{fair}})} \Psi$ – satisfaction for all *fair* adversaries
  
  - **Achievability queries**: $M \models_{(\exists)} \Psi$
    - does there exist an adversary $\sigma$ of $M$ such that $M,\sigma \models \Psi$?
  
  - **Numerical queries**:
    - $\min/\max$ probability/reward subject to constraint $\Psi$?
    - e.g. $\Pr_{M}^{\max}(\phi \mid \Psi) = \sup \{ \Pr_{M,\sigma}(\phi) \mid \sigma \in \text{Adv}_{M} \text{ and } M,\sigma \models \Psi \}$

- Examples…
  
  - $M \models_{(\forall)} [\Diamond \text{done}]_{>0.99} \lor [\Diamond \square \text{off}]_{\geq 1}$
  
  - $M \models_{(\exists)} [\rho_{\text{time}}]_{\leq 5} \land [\rho_{\text{pow}}]_{\leq 30} \land [\Diamond \text{done}]_{\geq 1}$
  
  - $\text{ExpTot}_{M}^{\min}(\rho_{\text{time}} \mid [\rho_{\text{pow}}]_{\leq 30} \land [\Diamond \text{done}]_{\geq 1})$
Model checking qmo-properties

- **Just consider numerical queries**
  - verification query = (negated) achievability query
  - achievability query = numerical query with dummy objective
- **Just consider conjunctions of probability/reward predicates**
  - convert to disjunctive normal form, check separately

- So let’s assume a query of the form $\text{ExpTot}_{M}^{\max}(\rho_{0} \mid \Psi)$
  - where $\Psi = (\lceil \phi_{1} \rceil_{p_{1}} \land \ldots \land \lceil \phi_{n} \rceil_{p_{n}}) \land (\lceil \rho_{1} \rceil_{r_{1}} \land \ldots \land \lceil \rho_{m} \rceil_{r_{m}})$

- First, we impose the following assumption:
  - $\sup \{ \text{ExpTot}_{M}^{\sigma}(\rho) \mid M, \sigma \models \land_{i} \lceil \phi_{i} \rceil_{p_{i}} \} < \infty$
    - for all $\rho \in \{\rho_{0}, \rho_{1}, \ldots, \rho_{m}\}$ that are being maximised
  - verifiable during model checking
Model checking qmo–properties

- **Algorithm summary for** $\text{ExpTot}_{M}^{\text{max}}(\rho_0 \mid \Psi)$
  - convert probabilistic predicates to $[\phi_i]_{\triangleright p_i}$ where $\triangleright \in \{\geq, >\}$
  - build product $M'$ of PA $M$ and determ. Rabin automata for $\phi_i$
  - check assumption, remove actions yielding infinite rewards
  - convert predicates $[\phi_i]_{\triangleright p_i}$ to reward predicates $[\lambda_i]_{\triangleright p_i}$
  - build and solve (dual) linear programming problem
  - yielding randomised (memoryless) adversary $\sigma'$ of $M'$
  - convert to randomised (finite–memory) adversary $\sigma$ of $M$
  - similar approach to [Etessami et al., TACAS’07]

- **Complexity:** polynomial in $|M|$ (but 2EXP in property)
  - i.e. same as for standard (single–objective) verification
  - in practice, slightly less scalable since can’t use value iteration
Overview

• Verification of probabilistic systems
  – probabilistic automata (or Markov decision processes)
  – quantitative verification of temporal logic specifications

• Multi-objective quantitative verification
  – formalise trade-offs between several different objectives
  – probabilistic ω-regular properties & expected total reward
  – verification, achievability & numerical queries
  – flexible property specification, efficient model checking

• Controller synthesis
  – synthesis of optimal adversaries/schedulers for PAs/MDPs

• Compositional probabilistic verification
  – assume-guarantee framework for probabilistic automata
Controller synthesis

• Achievability and numerical queries are directly applicable to the problem of controller synthesis

• Running example, using numerical query:
  - \( \text{ExpTot}_{M}^{\min} \left( \rho_{\text{time}} \mid [\rho_{\text{pow}} \leq 30 \land [\Diamond \text{done}] \geq 1] \right) \) on \( M = M_d \parallel M_m \)
  - i.e. “minimise expected completion time, subject to upper bound on expected energy consumption”

• Result: minimum expected time: \( \frac{49}{11} = 4.4545\ldots \)
  - controller: job 1 fast/slow with prob. 5/6 and 1/6; job 2 slow
Case study: Dynamic power management

- **Synthesis of dynamic power management schemes**
  - for an IBM TravelStar VP disk drive
  - 5 different power modes: active, idle, idlelp, stby, sleep
  - power manager controller bases decisions on current power mode, disk request queue, etc.

- **Build controllers that**
  - minimise energy consumption, subject to constraints on e.g.
  - probability that a request waits more than $K$ steps
  - expected number of lost disk requests

- **See:** [http://www.prismmodelchecker.org/files/tacas11/](http://www.prismmodelchecker.org/files/tacas11/)
Overview

• Verification of probabilistic systems
  – probabilistic automata (or Markov decision processes)
  – quantitative verification of temporal logic specifications

• Multi-objective quantitative verification
  – formalise trade-offs between several different objectives
  – probabilistic \( \omega \)-regular properties & expected total reward
  – verification, achievability & numerical queries
  – flexible property specification, efficient model checking

• Controller synthesis
  – synthesis of optimal adversaries/schedulers for PAs/MDPs

• Compositional probabilistic verification
  – assume-guarantee framework for probabilistic automata
Compositional verification

• **Goal: scalability through modular verification**
  – e.g. decide if $M_1 \parallel M_2 \models G$ using separate analysis of $M_1, M_2$

• **Assume–guarantee (A/G) reasoning**
  – use assumptions $A$ about the context of a component $M$
  – $\langle A \rangle M \langle G \rangle$ – “whenever $M$ is part of a system that satisfies $A$, then the system must also guarantee $G$”
  – example (asymmetric) A/G proof rule:

\[
\begin{align*}
M_1 & \models A \\
\langle A \rangle M_2 \langle G \rangle \\
\hline
M_1 \parallel M_2 & \models G
\end{align*}
\]
Probabilistic assume guarantee

• **Assumptions** $\Psi_A$ and guarantees $\Psi_G$ are qmo–properties
  – i.e. combinations of probabilistic $\omega$–regular properties (incl. probabilistic safety, liveness), expected total reward properties

• **Assume–guarantee triples** $\langle \Psi_A \rangle M \langle \Psi_G \rangle$ for a PA $M$
  – checking reduces to qmo verification query

• **Extends our earlier A/G framework for PAs** [TACAS’10]
  – where $A$ and $G$ are probabilistic safety properties
  – much richer class of properties for $G$ and (crucially) $A$

• **Adapt proof rules to incorporate (unconditional) fairness**
  – with probability 1, $M_1$ and $M_2$ make transitions infinitely often
Probabilistic assume guarantee triple

- Assume–guarantee triple $\langle \Psi_A \rangle M \langle \Psi_G \rangle$

- Informally:
  - “when $M$ is a component of a system satisfying $\Psi_A$, then the combined system (under fairness) is guaranteed to satisfy $\Psi_G$”

- Formally:
  - $\langle \Psi_A \rangle M \langle \Psi_G \rangle$
  - $\iff$
    - $\forall \sigma \in \text{Adv}_{M'} \ (M',\sigma \models \Psi_A \rightarrow M',\sigma \models \Psi_G)$
    - $\iff$
      - $M' \models \forall (\neg \Psi_A \lor \Psi_G)$

  - where $M' = M[\alpha_A]$, i.e. $M$ with alphabet extended to that of $\Psi_A$
An assume–guarantee rule

• The following (asymmetric) proof rule holds

\[ \frac{M_1 \models \forall \psi_A}{\langle \psi_A \rangle M_2 \langle \psi_G \rangle} \quad (ASYM) \]

\[ M_1 \parallel M_2 \models \forall_{\text{fair}} \psi_G \]

• So, verifying \( M_1 \parallel M_2 \models \forall_{\text{fair}} \psi_G \) reduces to 2 sub–problems:
  – premise 1: \( M_1 \models \forall \psi_A \) – standard model checking (usually)
  – premise 2: \( \langle \psi_A \rangle M_2 \langle \psi_G \rangle \) – multi–objective model checking

• Compositional verification can be much more efficient
  – for small assumption \( \psi_A \) about large \( M_1 \)
Running example

• **Compositional probabilistic model checking:**
  - verify that: \( M_c \parallel M_m \models \forall \text{fair} \ [\rho_{\text{time}}] \leq 3.2 \)
    (“expected completion time \( \leq 3.2 \)”)
  - using rule (ASYM) with assumption: \( \Psi_A = [\square \neg \text{off}] \geq 1 \land [\rho_{\text{slow}}] \leq 0.5 \)
    (“never switch off and expected num. of slow jobs \( \leq 0.5 \)”)

**Reward structures:**

- \( \rho_{\text{slow}} = \{\text{slow} \rightarrow 1\} \)
- \( \rho_{\text{time}} = \{\text{fast} \rightarrow 1, \text{slow} \rightarrow 3, \text{on} \rightarrow 5\} \)
Running example

- **Premise 1:**
  - verify that: \( M_c \models \forall \left[ \square \neg \text{off} \right] \geq 1 \land \left[ \rho_{\text{slow}} \right] \leq 0.5 \)
  - yes, since \( \Pr_{M}^{min} (\neg \text{off}) = 1 \) and \( \text{ExpTot} (\rho_{\text{slow}}) = 0.5 \)

\[
\begin{align*}
M_c & \models \forall \langle \Psi_A \rangle \\
\langle \Psi_A \rangle M_m \langle [\rho_{\text{time}}] \leq 3.2 \rangle \\
\begin{array}{c}
M_c \parallel M_m \models \forall \text{fair} \left[ \rho_{\text{time}} \right] \leq 3.2 \\
M_c \models \forall \left[ \square \neg \text{off} \right] \geq 1 \land \left[ \rho_{\text{slow}} \right] \leq 0.5
\end{array}
\end{align*}
\]

with assumption:
\( \Psi_A = \left[ \square \neg \text{off} \right] \geq 1 \land \left[ \rho_{\text{slow}} \right] \leq 0.5 \)

\( \rho_{\text{slow}} = \{ \text{slow} \mapsto 1 \} \)
Running example

• Premise 2:

  - verify that: \( M_m \models \forall ( [\square \neg \text{off}] \geq 1 \land [\rho_{\text{slow}}] \leq 0.5 ) \rightarrow [\rho_{\text{time}}] \leq 3.2 \)
  - yes, since \( M_m \models \exists ( [\square \neg \text{off}] \geq 1 \land [\rho_{\text{slow}}] \leq 0.5 \land [\rho_{\text{time}}] > 3.2 ) \) is false

\[ M_c \models \forall \langle \Psi_A \rangle \]
\[ \langle \Psi_A \rangle M_m \langle [\rho_{\text{time}}] \leq 3.2 \rangle \]
\[ M_c \parallel M_m \models \forall \text{fair} [\rho_{\text{time}}] \leq 3.2 \]

with assumption
\[ \Psi_A = [\square \neg \text{off}] \geq 1 \land [\rho_{\text{slow}}] \leq 0.5 \]

\[ \rho_{\text{slow}} = \{ \text{slow} \rightarrow 1 \} \]
\[ \rho_{\text{time}} = \{ \text{fast} \rightarrow 1, \text{slow} \rightarrow 3, \text{on} \rightarrow 5 \} \]
“Quantitative” assume-guarantee

- A more “quantitative” approach
  - use numerical queries to obtain best/worst-case bounds

- For example:
  - if $\Psi_G$ is of the form $[\rho]_{\leq r}$ (upper bound on expected reward)
  - then, instead of checking premise 2: $\neg (M_2 \models_3 (\Psi_A \land \neg \Psi_G))$
  - compute $\text{ExpTot}_{M_2}^{\max} (\rho \mid \Psi_A)$ (worst-case under assumption)

- In similar style, we can “optimise” our assumptions
  - e.g. if $\Psi_A$ is of the form $[\phi]_{\geq p}$
  - we can compute $p^* = \Pr_{M_1}^{\min} (\phi)$
  - and use assumption $[\phi]_{\geq p^*}$ instead

- We can also study multi-objective LP, Pareto curves, …
Implementation + Case studies

• **Extension of PRISM model checker**
  – already supports LTL and reward properties for PAs/MDPs
  – added support for multi-objective model checking
    · using LP solvers (ECLiPSe/COIN-OR CBC/lpsolve)
  – fully-automated support for A/G reasoning in progress

• **Two large case studies**
  – randomised consensus algorithm [Aspnes & Herlihy]
    · maximum expected steps required in the first R rounds
  – Zeroconf network protocol
    · termination with (minimum) probability 1
    · minimum/maximum expected time for termination

### Experimental results

<table>
<thead>
<tr>
<th>Case study [parameters] &amp; Property</th>
<th>Non-compositional</th>
<th>Compositional</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>States</td>
<td>Time (s)</td>
</tr>
<tr>
<td><strong>Randomised consensus (3 proc.s) [R,K]</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3, 2</td>
<td>114,559</td>
<td>20.5</td>
</tr>
<tr>
<td>3, 12</td>
<td>507,919</td>
<td>1,361.6</td>
</tr>
<tr>
<td>3, 20</td>
<td>822,607</td>
<td>time-out</td>
</tr>
<tr>
<td>4, 2</td>
<td>3,669,649</td>
<td>728.1</td>
</tr>
<tr>
<td>4, 12</td>
<td>29,797,249</td>
<td>mem-out</td>
</tr>
<tr>
<td>4, 20</td>
<td>65,629,249</td>
<td>mem-out</td>
</tr>
<tr>
<td><strong>ZeroConf [K]</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>57,960</td>
<td>8.7</td>
</tr>
<tr>
<td>6</td>
<td>125,697</td>
<td>16.6</td>
</tr>
<tr>
<td>8</td>
<td>163,229</td>
<td>19.4</td>
</tr>
<tr>
<td><strong>ZeroConf [K]</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>57,960</td>
<td>5.8</td>
</tr>
<tr>
<td>6</td>
<td>125,697</td>
<td>13.3</td>
</tr>
<tr>
<td>8</td>
<td>163,229</td>
<td>18.9</td>
</tr>
</tbody>
</table>
### Experimental results

<table>
<thead>
<tr>
<th>Case study [parameters] &amp; Property</th>
<th>Non-compositional</th>
<th>Compositional</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>States</td>
<td>Time (s)</td>
</tr>
<tr>
<td>Randomised consensus (3 proc.s) [R,K]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3, 2</td>
<td>114,559</td>
<td>20.5</td>
</tr>
<tr>
<td>3, 12</td>
<td>507,919</td>
<td>1,361.6</td>
</tr>
<tr>
<td>3, 20</td>
<td>822,607</td>
<td>time-out</td>
</tr>
<tr>
<td>4, 2</td>
<td>3,669,649</td>
<td>728.1</td>
</tr>
<tr>
<td>“max. steps”</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4, 12</td>
<td>29,797,249</td>
<td>mem-out</td>
</tr>
<tr>
<td>4, 20</td>
<td>65,629,249</td>
<td>mem-out</td>
</tr>
<tr>
<td>ZeroConf [K]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“termination”</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>57,960</td>
<td>8.7</td>
</tr>
<tr>
<td>6</td>
<td>125,697</td>
<td>16.6</td>
</tr>
<tr>
<td>8</td>
<td>163,229</td>
<td>19.4</td>
</tr>
<tr>
<td>ZeroConf [K]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“max. time”</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>57,960</td>
<td>5.8</td>
</tr>
<tr>
<td>6</td>
<td>125,697</td>
<td>13.3</td>
</tr>
<tr>
<td>8</td>
<td>163,229</td>
<td>18.9</td>
</tr>
</tbody>
</table>

- Compositional verification: faster, larger models, scales better
## Experimental results

<table>
<thead>
<tr>
<th>Case study [parameters] &amp; Property</th>
<th>Non-compositional</th>
<th></th>
<th>Compositional</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>States</td>
<td>Result</td>
<td>LP size</td>
<td>Result</td>
</tr>
<tr>
<td>Randomised consensus (3 proc.s) [R,K]</td>
<td>3, 2</td>
<td>114,559</td>
<td>212.0</td>
<td>43,712</td>
</tr>
<tr>
<td></td>
<td>3, 12</td>
<td>507,919</td>
<td>4,352</td>
<td>92,672</td>
</tr>
<tr>
<td></td>
<td>3, 20</td>
<td>822,607</td>
<td>-</td>
<td>131,840</td>
</tr>
<tr>
<td></td>
<td>4, 2</td>
<td>3,669,649</td>
<td>212.0</td>
<td>260,254</td>
</tr>
<tr>
<td>“max. steps”</td>
<td>4, 12</td>
<td>29,797,249</td>
<td>-</td>
<td>351,694</td>
</tr>
<tr>
<td></td>
<td>4, 20</td>
<td>65,629,249</td>
<td>-</td>
<td>424,846</td>
</tr>
<tr>
<td>ZeroConf [K]</td>
<td>4</td>
<td>57,960</td>
<td>1.0</td>
<td>155,458</td>
</tr>
<tr>
<td>“termination”</td>
<td>6</td>
<td>125,697</td>
<td>1.0</td>
<td>156,690</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>163,229</td>
<td>1.0</td>
<td>157,922</td>
</tr>
<tr>
<td>ZeroConf [K]</td>
<td>4</td>
<td>57,960</td>
<td>14.28</td>
<td>154,632</td>
</tr>
<tr>
<td>“max. time”</td>
<td>6</td>
<td>125,697</td>
<td>18.28</td>
<td>155,600</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>163,229</td>
<td>22.28</td>
<td>156,568</td>
</tr>
</tbody>
</table>

- Compositional verification: yields good bounds on actual results
Conclusions

- **Multi–objective model checking techniques for PAs/MDPs**
  - simple, temporal–logic–based language with:
    - probabilistic ω–regular and expected total reward properties
    - verification, achievability and numerical queries

- **Applications to controller synthesis**
  - large case study: dynamic power management for disk–drive

- **Compositional probabilistic verification**
  - assume–guarantee framework for probabilistic automata
  - richer assumptions/guarantees; quantitative results
  - good experimental results: faster verification, larger models

- **Current/future work**
  - assumption generation via learning (done for safety prop.s)
  - continuous– or real–time models