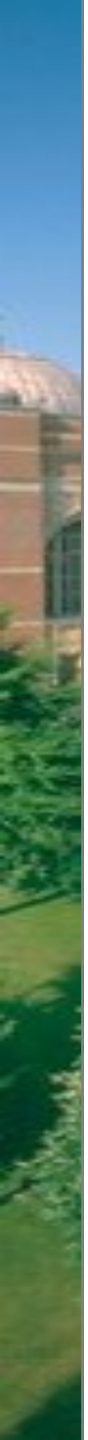


Verification and Strategy Synthesis for Stochastic Games

Dave Parker

University of Birmingham

Radboud University, March 2020

Verification and Strategy Synthesis for Stochastic Games

Dave Parker

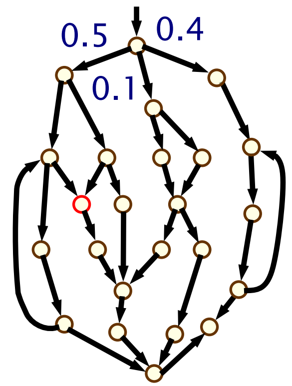
University of Birmingham

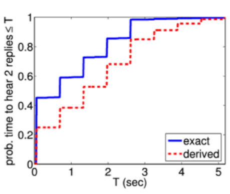
Joint work with:

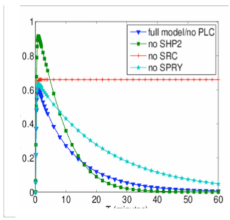
Gabriel Santos, Gethin Norman, Marta Kwiatkowska, …

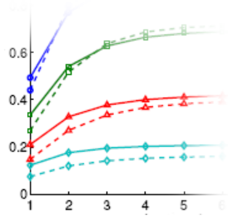
Probabilistic model checking

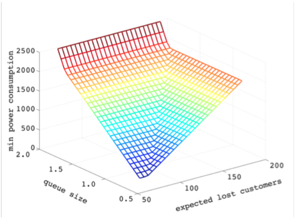
- Probabilistic model checking
  - formal construction/analysis of probabilistic models
  - "correctness" properties expressed in temporal logic
  - e.g. trigger → $P_{\geq 0.999}$ [$F^{\leq 20}$ deploy]
  - mix of exhaustive & numerical/quantitative reasoning







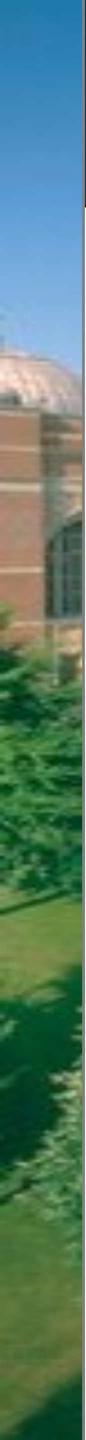




- Trends and advances
  - improvement in scalability to larger models
  - increasingly expressive/powerful model classes
  - from verification problems to control problems
  - ever widening range of application domains

Stochastic games

- Verification of systems with
  - competitive or collaborative behaviour between multiple rational agents, possibly with differing/opposing goals
  - e.g. security protocols, algorithms for distributed consensus, energy management, autonomous robotics, auctions
- Goals
  - synthesise (single or joint) strategies that are robust in adversarial settings and stochastic environments
  - analyse the effectiveness of incentive/reward schemes designed for robustness against selfish behaviour
- Natural to take a game-theoretic approach
  - we use stochastic multi-player games
  - probabilistic model checking using PRISM-games

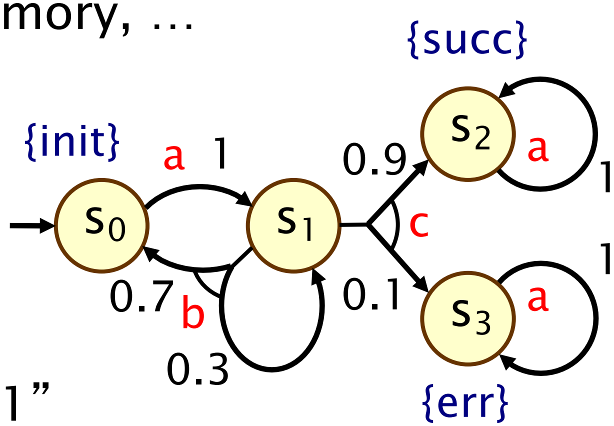


Overview

- Strategy synthesis
  - Markov decision processes (MDPs)
  - example: robot navigation
- Stochastic multi-player games (SMGs)
  - rPATL model checking and strategy synthesis
  - example: energy management
- Concurrent stochastic games (CSGs)
  - example: investor models
- Equilibria-based properties
  - (social welfare) Nash equilibria
  - example: multi-robot coordination

Verification vs. Strategy synthesis

- Markov decision processes (MDPs)
  - models nondeterministic (actions, strategies) and probabilistic behaviour
  - strategies (policies): randomisation, memory, …

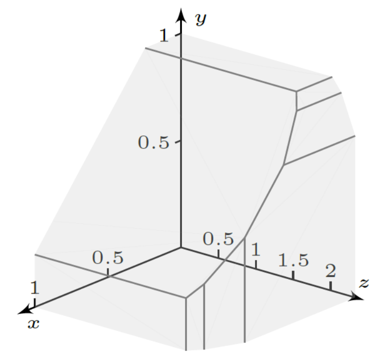


- 1. Verification
  - quantify over all possible strategies (i.e. best/worst-case)
  - $P_{\leq 0.1}$ [F *err*] : "for all strategies, the probability of an error occurring is $\leq$ 0.1"

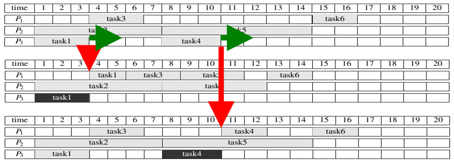
- 2. Strategy synthesis
  - generation of "correct-by-construction" controllers
  - $P_{\leq 0.1}$ [F *err*] : "does there exist a strategy for which the probability of an error occurring is $\leq$ 0.1?"

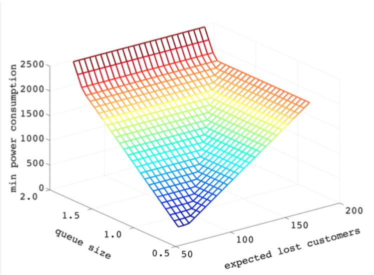
Strategy synthesis for MDPs

- Core property: probabilistic reachability
  - solvable with value iteration, policy iteration, linear programming, interval iteration, …



- Wide range of useful extensions
  - expected costs/rewards
  - linear temporal logic (LTL)
  - multi-objective model checking
  - real-time (PTAs)
  - partial observability (POMDPs)

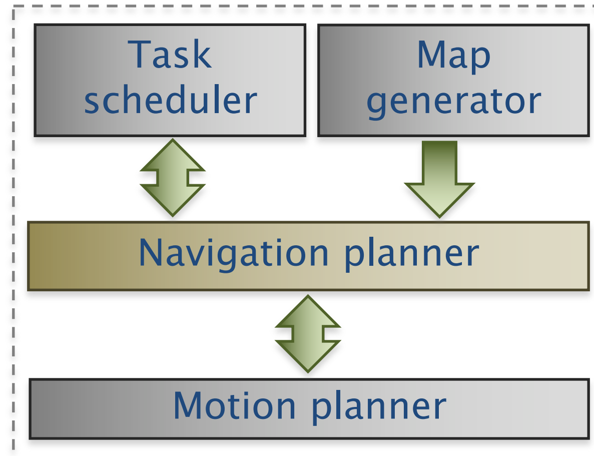


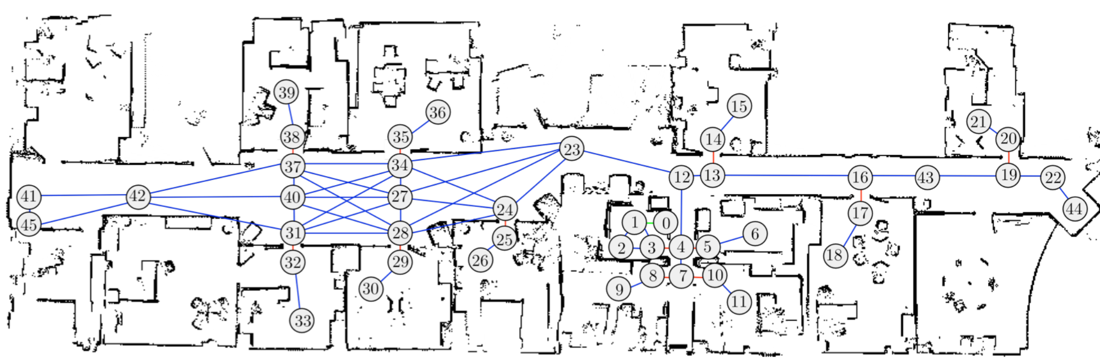


- Applications
  - dynamic power management, robot navigation, UUV mission planning, task scheduling

Application: Robot navigation

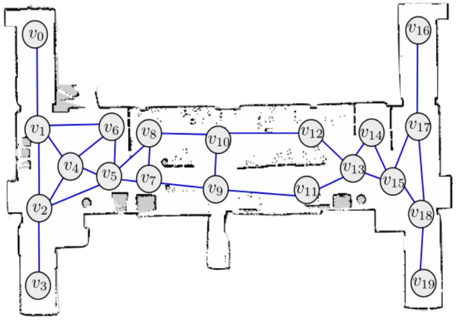
- Robot navigation planning: [IROS'14,IJCAI'15,ICAPS'17,IJRR'18]
  - learnt MDP models navigation through uncertain environment
  - co-safe LTL used to formally specify tasks to be executed by robot
  - finite-memory strategy synthesis to construct plans/controllers
  - ROS module based on PRISM
  - 100s of hrs of autonomous deployment





G4S Technology, Tewkesbury (STRANDS)

Application: Robot navigation



- Navigation planning MDPs
  - expected timed on edges + probabilities
  - learnt using data from previous explorations
- LTL-based task specification
  - expected time to satisfy (one or more) co-safe LTL formulas
  - e.g. $R_{min=?}$ [$\neg$zone$_3$ U (room$_1$ $\wedge$ (F room$_4$ $\wedge$ F room$_5$)]
- Benefits of the approach
  - LTL: flexible, unambiguous property specification
  - efficient, fully-automated techniques
  - generates meaningful guarantees on performance
    - c.f. ad-hoc reward structures, e.g. with discounting
    - QoS guarantees fed into task planning

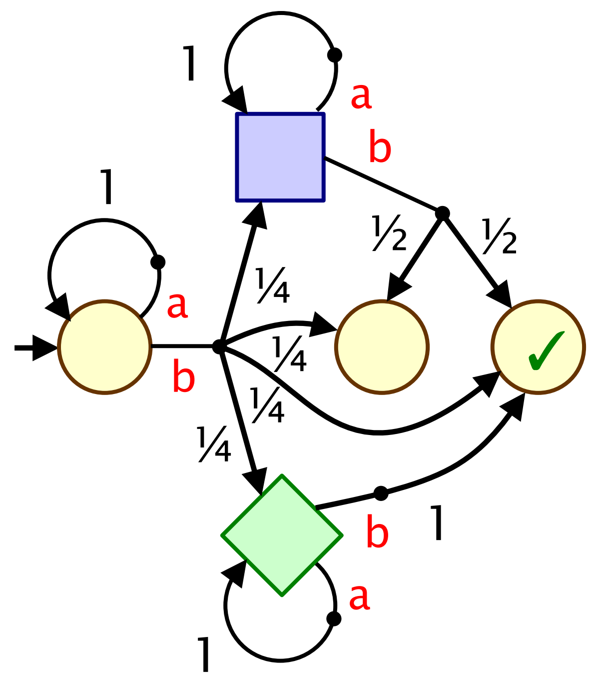
Overview

- Strategy synthesis
  - Markov decision processes (MDPs)
  - example: robot navigation
- Stochastic multi-player games (SMGs)
  - rPATL model checking and strategy synthesis
  - example: energy management
- Concurrent stochastic games (CSGs)
  - example: investor models
- Equilibria-based properties
  - (social welfare) Nash equilibria
  - example: multi-robot coordination

Stochastic multi-player games

- Stochastic multi-player game (SMGs)
  - nondeterminism + probability + multiple players
  - for now: turn-based (players control states)
  - applications: e.g. security (system vs. attacker), controller synthesis (controller vs. environment)

- A (turn-based) SMG is a tuple $(N, S, \langle S_i \rangle_{i \in N}, A, \delta, L)$ where:
  - N is a set of n players
  - S is a (finite) set of states
  - $\langle S_i \rangle_{i \in N}$ is a partition of S
  - A is a set of action labels
  - $\delta : S \times A \to Dist(S)$ is a (partial) transition probability function
  - $L : S \to 2^{AP}$ is a labelling function



Strategies, probabilities & rewards

- Strategy for player i: resolves choices in S_i states
  - based on execution history, i.e. $\sigma_i : (SA)^*S_i \rightarrow Dist(A)$
  - can be: deterministic (pure), randomised, memoryless, finite-memory, …
  - Σ_i denotes the set of all strategies for player i

- Strategy profile: strategies for all players: $\sigma=(\sigma_1,\ldots,\sigma_n)$
  - probability measure Pr_s^σ over (infinite) paths from state s
  - expectation $E_s^\sigma(X)$ of random variable X over Pr_s^σ

- Rewards (or costs)
  - non-negative integers on states/transitions
  - e.g. elapsed time, energy consumption, number of packets lost, net profit, …

Property specification: rPATL

- rPATL (reward probabilistic alternating temporal logic)
  - branching-time temporal logic for SMGs
- CTL, extended with:
  - coalition operator $\langle\langle C \rangle\rangle$ of ATL
  - probabilistic operator P of PCTL
  - generalised (expected) reward operator R from PRISM
- In short:
  - zero-sum, probabilistic reachability + expected total reward
- Example:
  - $\langle\langle \{1,3\} \rangle\rangle \, P_{<0.01} [\, F^{\leq 10} \, \mathrm{error} \,]$
  - "players 1 and 3 have a strategy to ensure that the probability of an error occurring within 10 steps is less than 0.01, regardless of the strategies of other players"

rPATL syntax/semantics

- Syntax:

$$\phi ::= \mathrm{true} \mid a \mid \neg\phi \mid \phi \wedge \phi \mid \langle\langle C\rangle\rangle P_{\bowtie q}[\psi] \mid \langle\langle C\rangle\rangle R^{r}_{\bowtie x}\,[\rho]$$

$$\psi ::= X\,\phi \mid \phi\, U^{\leq k}\, \phi \mid \phi\, U\, \phi$$

$$\rho ::= I^{=k} \mid C^{\leq k} \mid F\,\phi$$

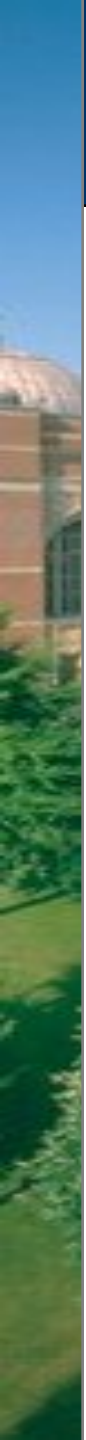
- where:
  - $a \in AP$ is an atomic proposition, $C \subseteq N$ is a coalition of players, $\bowtie \in \{\leq,<,>,\geq\}$, $q \in [0,1] \cap \mathbb{Q}$, $x \in \mathbb{Q}_{\geq 0}$, $k \in \mathbb{N}$

    r is a reward structure

- Semantics:
- e.g. P operator: $s \vDash \langle\langle C\rangle\rangle P_{\bowtie q}[\psi]$ iff:
  - "there exist strategies for players in coalition C such that, for all strategies of the other players, the probability of path formula ψ being true from state s satisfies ⋈ q"

rPATL and beyond

- Quantitative (numerical) properties:
  - $\langle\langle\{1\}\rangle\rangle\ P_{max=?}[\ F\ error\ ]$, i.e. $\sup_{\sigma_1\in\Sigma_1} \inf_{\sigma_2\in\Sigma_2} Pr_s^{\sigma_1,\sigma_2}(F\ error)$
  - "what is the maximum probability of reaching an error state that player 1 can guarantee?" (against player 2)
- Nesting (and n>2 players)
  - players: $sensor_1$, $sensor_2$, repairer
  - $\langle\langle sensor_1\rangle\rangle\ P_{<0.01}[\ F\ (\neg\langle\langle repairer\rangle\rangle\ P_{\geq 0.95}[\ F\ \text{"operational"}\ ]\ )\ ]$
- Generalised reward operators [TACAS'12, FMSD'13]
  - $\langle\langle C\rangle\rangle R^r_{\bowtie x}\ [F^\star \phi]$ where $\star \in \{\infty,c,0\}$
  - F^0 is tricky: needs finite-memory strategies
- And more…
  - rPATL*, reward-bounded [FMSD], exact bounds [CONCUR'12]
  - multi-objective model checking [QEST'13,TACAS15,I&C'17]



Model checking rPATL

- Main task: checking individual P and R operators
  - reduction to solution of zero-sum stochastic 2-player game
  - (probabilistic reachability + expected total reward)
  - e.g. $\langle\langle C\rangle\rangle P_{\geq q}[\psi] \;\Leftrightarrow\; \sup_{\sigma_1 \in \Sigma_1} \inf_{\sigma_2 \in \Sigma_2} \mathrm{Pr}_s^{\sigma_1,\sigma_2}(\psi) \geq q$
  - complexity: NP ∩ coNP (without any $R[F^0]$ operators)
  - complexity for full logic: NEXP ∩ coNEXP (due to $R[F^0]$ op.)

- In practice though:
  - (usual approach taken in probabilistic model checking tools)
  - value iteration (evaluation of numerical fixed points)
  - and more: graph-algorithms, sequences of fixed points, …

Example: Probabilistic reachability

- E.g. $\langle\langle C\rangle\rangle P_{\geq q}[\ F\ \phi\ ]$: max/min reachability probabilities
  - compute $\sup_{\sigma_1 \in \Sigma_1} \inf_{\sigma_2 \in \Sigma_2} Pr_s^{\sigma_1,\sigma_2}(F\ \phi)$ for all states s
  - deterministic memoryless strategies suffice

- Value $p(s)$ for state s is least fixed point of:

$$p(s) = \begin{cases} 1 & \text{if } s \in Sat(\phi) \\ \max_{a \in A(s)} \Sigma_{s' \in S}\ \delta(s,a)(s') \cdot p(s') & \text{if } s \in S_1 \backslash Sat(\phi) \\ \min_{a \in A(s)} \Sigma_{s' \in S}\ \delta(s,a)(s') \cdot p(s') & \text{if } s \in S_2 \backslash Sat(\phi) \end{cases}$$

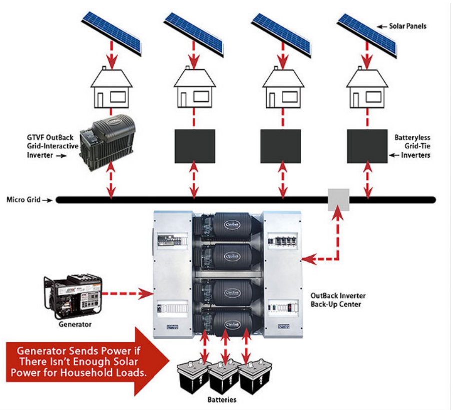
- Computation (value iteration):
  - start from zero, propagate probabilities backwards
  - guaranteed convergence; apply “usual” termination criteria

PRISM-games

- PRISM-games: www.prismmodelchecker.org/games
  - extension of PRISM modelling language (see later)
  - implementation in explicit engine
  - prototype symbolic (MTBDD) version also available
- Example application domains
  - security: attack-defence trees; DNS bandwidth amplification
  - self-adaptive software architectures
  - autonomous urban driving
  - human-in-the-loop UAV mission planning
  - collective decision making and team formation protocols
  - energy management protocols

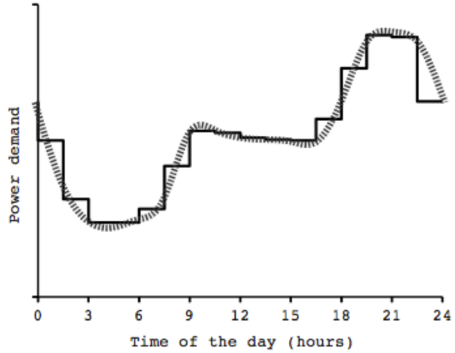
Application: Energy management

- Energy management protocol for Microgrid
  - randomised demand management protocol
  - random back-off when demand is high



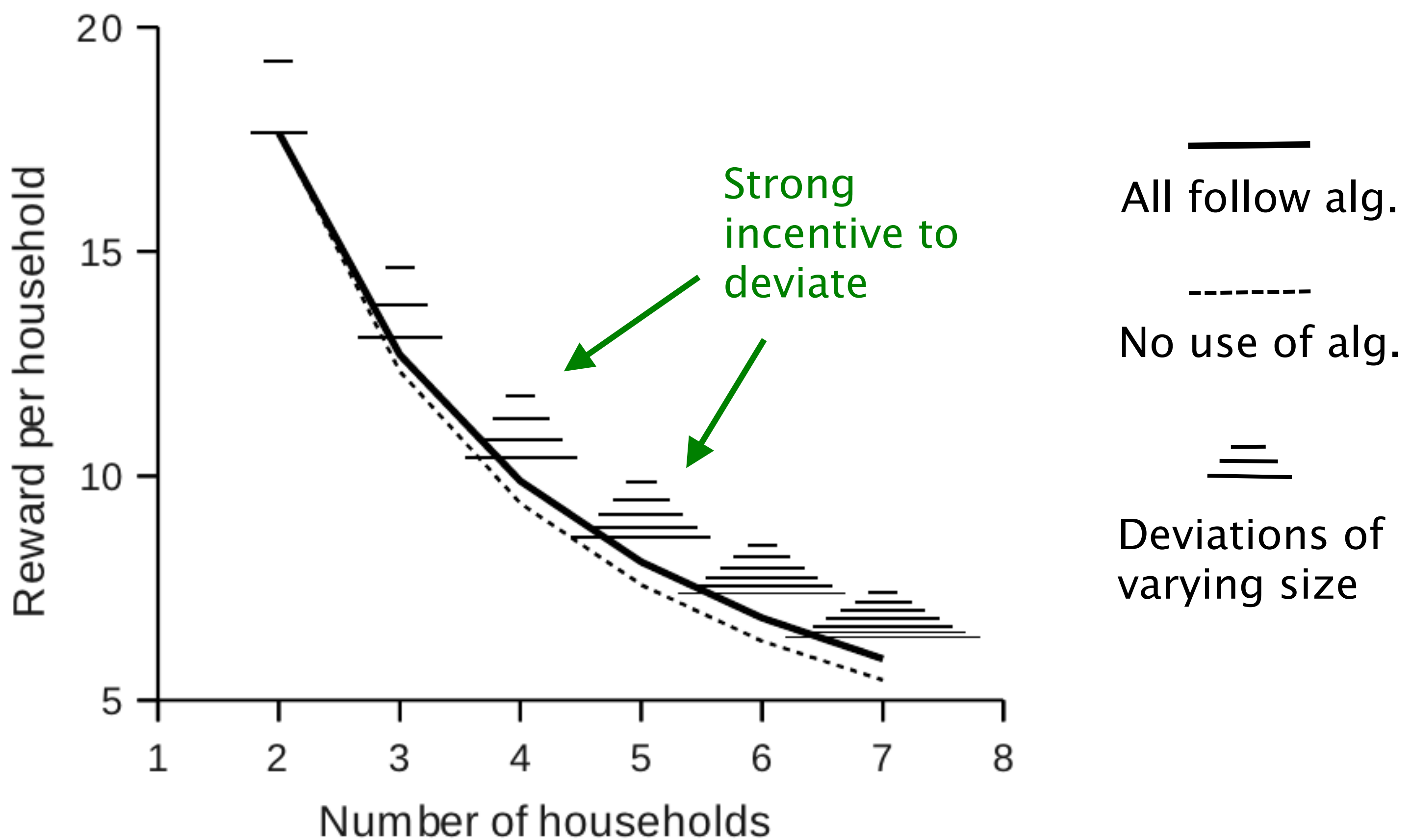
- Original analysis [Hildmann/Saffre'11]
  - protocol increases "value" for clients
  - simulation-based, clients are honest

- Our analysis
  - stochastic multi-player game model
  - clients can cheat (and cooperate)
  - model checking: PRISM-games
  - exposes protocol weakness (incentive for clients to act selfishly
  - propose/verify simple fix using penalties



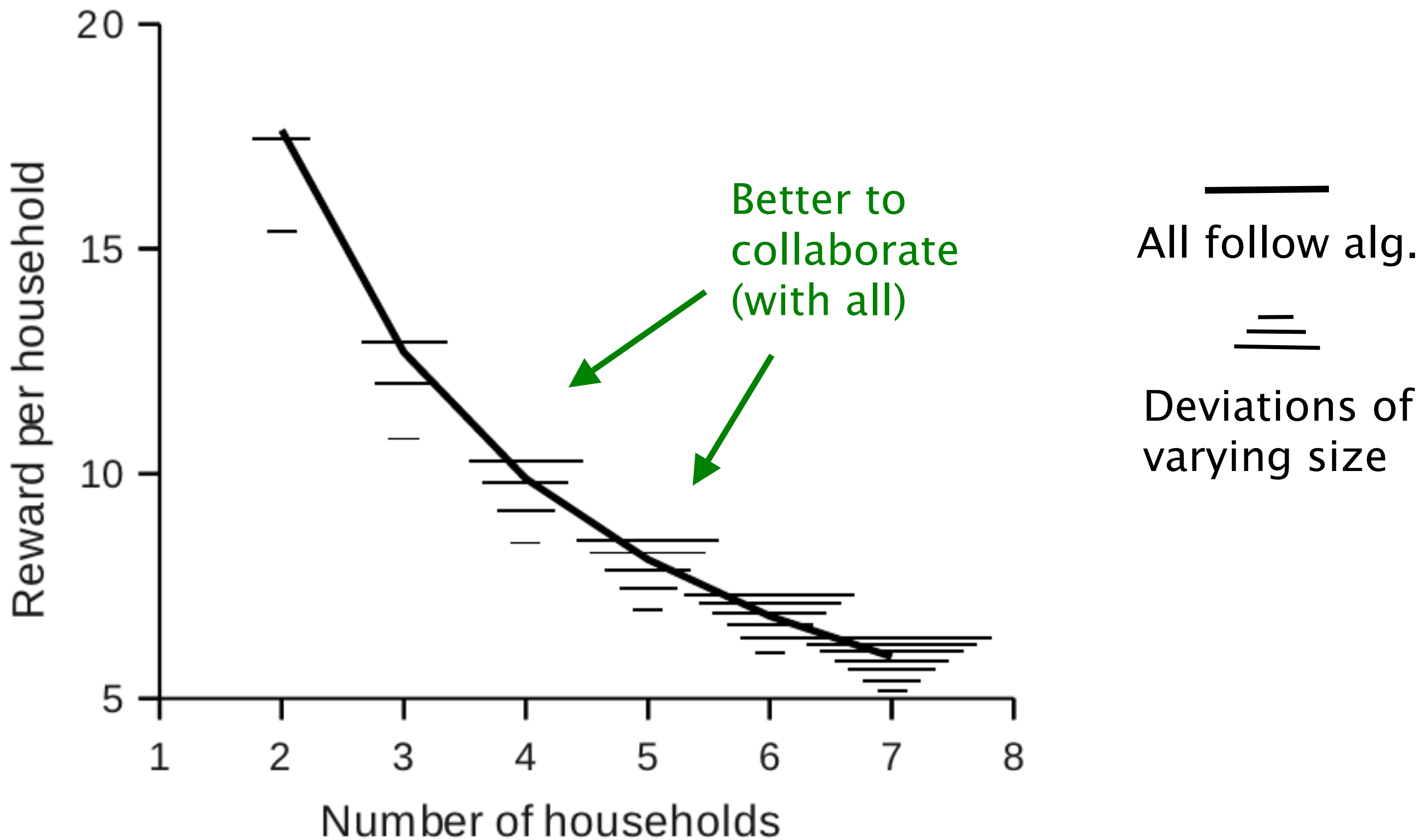
Results: Competitive behaviour

- Expected total value V per household
  - in rPATL: $\langle\langle C\rangle\rangle R^{r_C}_{max=?}\,[F^0\ time{=}max\ time]\ /\ |C|$
  - where r_C is combined rewards for coalition C



Results: Competitive behaviour

- Algorithm fix: simple punishment mechanism
  - distribution manager can cancel some loads exceeding c_{lim}



Overview

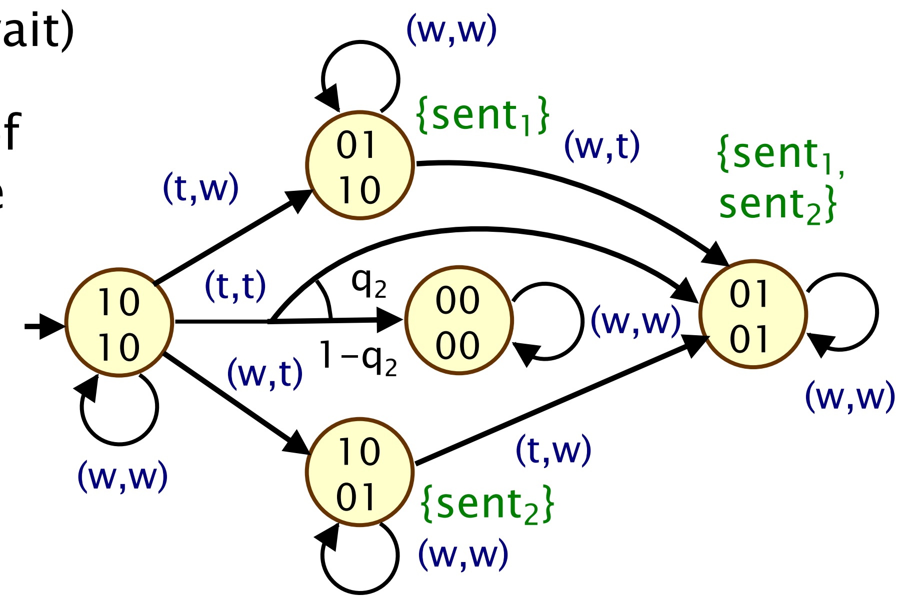
- Strategy synthesis
  - Markov decision processes (MDPs)
  - example: robot navigation
- Stochastic multi-player games (SMGs)
  - rPATL model checking and strategy synthesis
  - example: energy management
- Concurrent stochastic games (CSGs)
  - example: investor models
- Equilibria-based properties
  - (social welfare) Nash equilibria
  - example: multi-robot coordination

Concurrent stochastic games

- Concurrent stochastic games (CSGs)
  - players choose actions concurrently
  - jointly determines (probabilistic) successor state
  - generalises turn-based stochastic games
- Key motivation:
  - more realistic model of components operating concurrently, making action choices without knowledge of others
- Formally
  - set of n players N, state space S, actions A_i for player i
  - transition probability function $\delta : S \times A \rightarrow Dist(S)$
  - where $A = (A_1 \cup \{\perp\}) \times \ldots \times (A_n \cup \{\perp\})$
  - strategies $\sigma_i : FPath \rightarrow Dist(A_i)$, strategy profiles $\sigma=(\sigma_1,\ldots,\sigma_n)$
  - probability measure Pr_s^{σ}, expectations $E_s^{\sigma}(X)$

Example CSG: medium access control

- Example CSG: medium access control
  - 2 players (senders on a shared channel)
  - CSG states: $e_1 s_1$ $e_2 s_2$ ($energy_1/sent_1$, $energy_2/sent_2$)
  - actions = t (transmit), w (wait)
  - transmission costs 1 unit of energy and is only possible if energy is positive
  - q_2 = probability of transmission success if 2 messages sent simultaneously



(probabilistic extension of [Brenguier'13])

rPATL for CSGs

- We can use the same logic rPATL as for SMGs
- Examples for medium access control game:
  - $\langle\langle 1 \rangle\rangle \, P_{\geq 1} [\, F \; sent_1 \,]$ – can player 1 ensure that it eventually transmits with probability 1?
  - $\langle\langle 1 \rangle\rangle \, P_{max=?} [\, \neg sent_2 \; U \; sent_1 \,]$ – what is the maximum probability user 1 can ensure of being the first to transmit, regardless of the behaviour of user 2?

rPATL model checking for CSGs

- Same overall model checking algorithm [QEST'18]
  - key ingredients are solution of (zero-sum) 2-player CSGs
- E.g. $\langle\langle C\rangle\rangle P_{\geq q}[\ F\ \phi\ ]$: max/min reachability probabilities
  - compute $\sup_{\sigma_1 \in \Sigma_1} \inf_{\sigma_2 \in \Sigma_2} Pr_s^{\sigma_1,\sigma_2}(F\ \phi)$ for all states s
  - note that optimal strategies are now randomised
  - solution of the 2-player CSG is in PSPACE
  - we again use a value iteration based approach
- Value $p(s)$ for state s is least fixed point of:

$$p(s) = \begin{cases} 1 & \text{if } s \in Sat(\phi) \\ val(Z) & \text{if } s \in S \backslash Sat(\phi) \end{cases} \quad \text{where:}$$

  - Z is the matrix game with $z_{ij} = \Sigma_{s' \in S}\ \delta(s,(a_i,b_j))(s') \cdot p(s')$
  - so each iteration requires solution of a matrix game for each state (LP problem of size $|A|$, where A = action set)

Matrix games

- Matrix games
  - finite, one-shot, 2-player, zero-sum games
  - utility function $u_i : A_1 \times A_2 \to \mathbb{R}$ for each player i
  - represented by matrix Z where $z_{ij} = u_1(a_i,b_j) = -u_2(a_i,b_j)$
- Example: rock-paper-scissors
  - rock > scissors, paper > rock, scissors > paper, otherwise draw

$$Z = \begin{array}{c} \\ r \\ p \\ s \end{array} \begin{array}{c} \begin{array}{ccc} r & p & s \end{array} \\ \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix} \end{array}$$

- Optimal (player 1) strategy via LP solution (minimax):
  - compute value $val(Z)$: maximise value v subject to:
  - $v \leq x_p - x_s$
    $v \leq x_s - x_r,$
    $v \leq x_s - x_p$
    $x_r + x_p + x_s = 1$
    $x_r \geq 0, x_p \geq 0, x_s \geq 0$

Optimal strategy (randomised):
$(x_r, x_p, x_s) = (⅓, ⅓, ⅓)$

CSGs in PRISM-games

- CSG model checking implemented in PRISM-games 3.0
- Extension of PRISM modelling language
  - (see next slide)
- Explicit engine implementation
  - plus LPsolve library for matrix games LP solution
  - this is the main bottleneck
  - experiments with CSGs up to ~3 million states
- Case studies:
  - future markets investor, trust models for user-centric networks, intrusion detection policies, jamming radio systems

CSGs in PRISM-games 3.0

Extended version of medium access control example

```
csg
player p1 user1 endplayer
player p2 user2 endplayer
// Users (senders)
module user1
    s1 : [0..1] init 0; // has player 1 sent?
    e1 : [0..emax] init emax; // energy level of player 1
    [w1] true -> (s1'=0); // wait
    [t1]  e1>0 -> (s1'=c' ? 0 : 1) & (e1'=e1-1); // transmit
endmodule
module user2 = user1 [ s1=s2, e1=e2, w1=w2, t1=t2 ] endmodule
// Channel: used to compute joint probability distribution for transmission failure
module channel
    c : bool init false; // is there a collision?
    [t1,w2] true -> q1 : (c'=false) + (1-q1) : (c'=true); // only user 1 transmits
    [w1,t2] true -> q1 : (c'=false) + (1-q1) : (c'=true); // only user 2 transmits
    [t1,t2]  true -> q2 : (c'=false) + (1-q2) : (c'=true); // both users transmit
endmodule
```

CSGs in PRISM-games 3.0

```
csg
player p1 user1 endplayer
player p2 user2 endplayer
// Users (senders)
module user1
    s1 : [0..1] init 0; // has player 1 sent?
    e1 : [0..emax] init emax; // energy level of player 1
    [w1] true -> (s1'=0); // wait
    [t1] e1>0 -> (s1'=c' ? 0 : 1) & (e1'=e1-1); // transmit
endmodule
module user2 = user1 [ s1=s2, e1=e2, w1=w2, t1=t2 ] endmodule
// Channel: used to compute joint probability distribution for transmission failure
module channel
    c : bool init false; // is there a collision?
    [t1,w2] true -> q1 : (c'=false) + (1-q1) : (c'=true); // only user 1 transmits
    [w1,t2] true -> q1 : (c'=false) + (1-q1) : (c'=true); // only user 2 transmits
    [t1,t2] true -> q2 : (c'=false) + (1-q2) : (c'=true); // both users transmit
endmodule
```

Each player comprises one or more modules

Players have distinct actions, executed simultaneously

CSGs in PRISM-games 3.0

```
csg
player p1 user1 endplayer
player p2 user2 endplayer
// Users (senders)
module user1
        s1 : [0..1] init 0; // has player 1 sent?
        e1 : [0..emax] init emax; // energy level of player 1
        [w1] true -> (s1'=0); // wait
        [t1]  e1>0 -> (s1'=c' ? 0 : 1) & (e1'=e1-1); // transmit
endmodule
module user2 = user1 [ s1=s2, e1=e2, w1=w2, t1=t2 ] endmodule
// Channel: used to compute joint probability distribution for transmission failure
module channel
        c : bool init false; // is there a collision?
        [t1,w2] true -> q1 : (c'=false) + (1-q1) : (c'=true); // only user 1 transmits
        [w1,t2] true -> q1 : (c'=false) + (1-q1) : (c'=true); // only user 2 transmits
        [t1,t2]  true -> q2 : (c'=false) + (1-q2) : (c'=true); // both users transmit
endmodule
```

Additional (deterministic) modules not attached to any player

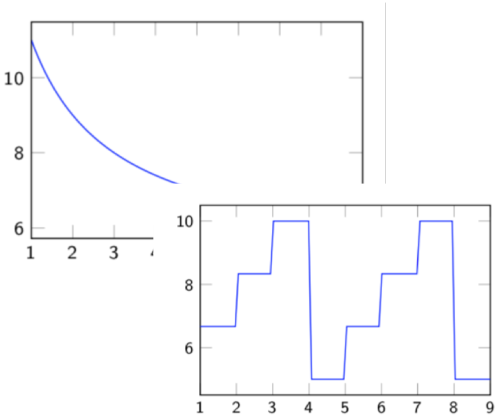
CSGs in PRISM-games 3.0

```
csg
player p1 user1 endplayer
player p2 user2 endplayer
// Users (senders)
module user1
	s1 : [0..1] init 0; // has player 1 sent?
	e1 : [0..emax] init emax; // energy level of player 1
	[w1] true -> (s1'=0); // wait
	[t1]  e1>0 -> (s1'=c' ? 0 : 1) & (e1'=e1-1); // transmit
endmodule
module user2 = user1 [ s1=s2, e1=e2, w1=w2, t1=t2 ] endmodule
// Channel: used to compute joint probability distribution for transmission failure
module channel
	c : bool init false; // is there a collision?
	[t1,w2] true -> q1 : (c'=false) + (1-q1) : (c'=true); // only user 1 transmits
	[w1,t2] true -> q1 : (c'=false) + (1-q1) : (c'=true); // only user 2 transmits
	[t1,t2]  true -> q2 : (c'=false) + (1-q2) : (c'=true); // both users transmit
endmodule
```

Variable updates can refer to other variables updated simultaneously

Action lists used to specify synchronisation

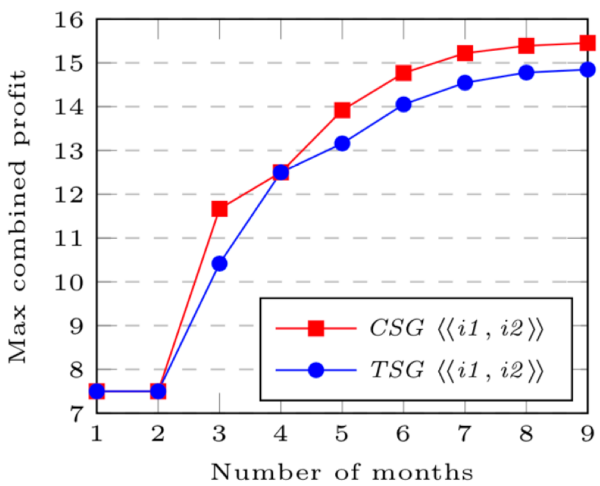
Application: Future markets investor

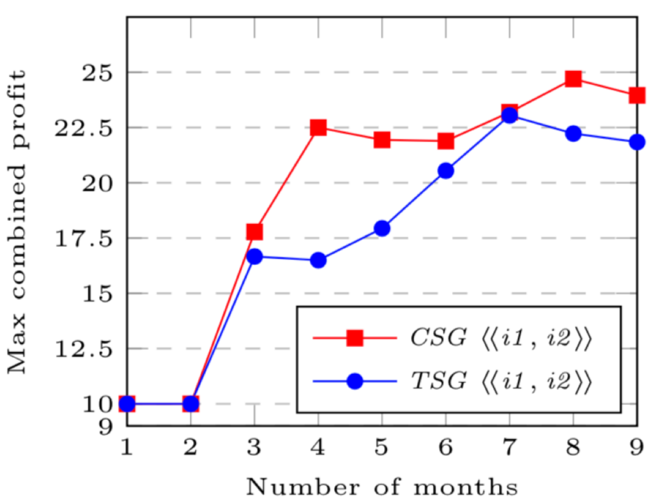


- Model of interactions between:
  - stock market, evolves stochastically
  - two investors i_1, i_2 decide when to invest
  - market decides whether to bar investors
- Modelled as a 3-player CSG
  - extends simpler model originally from [McIver/Morgan'07]
  - investing/barring decisions are simultaneous
  - profit reduced for simultaneous investments
  - market cannot observe investors' decisions
- Analysed with rPATL model checking & strategy synthesis
  - distinct profit models considered: 'normal market', 'later cash-ins' and 'later cash-ins with fluctuation'
  - comparison between TSG and CSG models

Application: Future markets investor

- Example rPATL query:
  - $\langle\langle \text{investor}_1,\text{investor}_2 \rangle\rangle\ R^{\text{profit}_{1,2}}_{\max=?}\ [\ F\ \text{finished}_{1,2}\ ]$
  - i.e. maximising joint profit
- Results: with (left) and without (right) fluctuations
  - optimal (randomised) investment strategies synthesised
  - CSG yields more realistic results (market has less power due to limited observation of investor strategies)





Overview

- Strategy synthesis
  - Markov decision processes (MDPs)
  - example: robot navigation
- Stochastic multi-player games (SMGs)
  - rPATL model checking and strategy synthesis
  - example: energy management
- Concurrent stochastic games (CSGs)
  - example: investor models
- Equilibria-based properties
  - (social welfare) Nash equilibria
  - example: multi-robot coordination

Nash equilibria

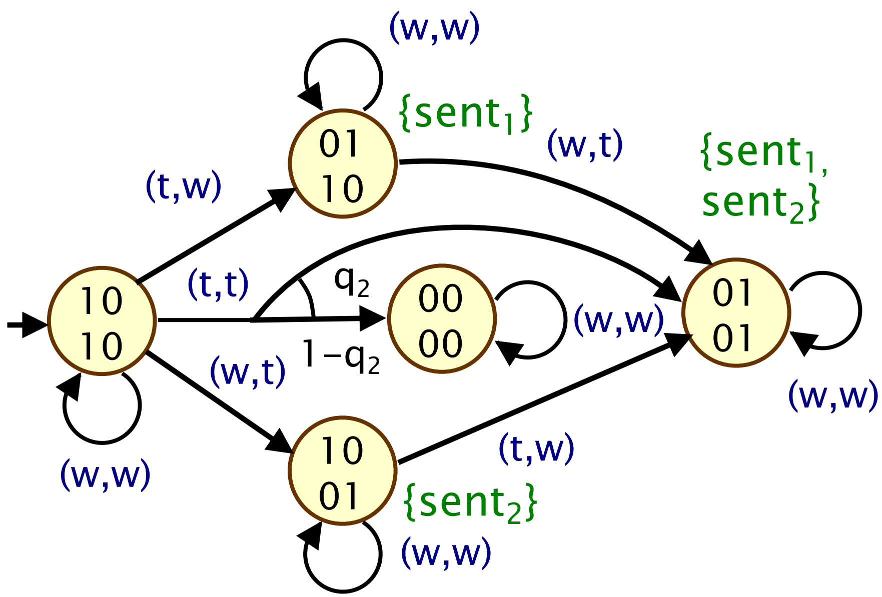
- Now consider distinct objectives X_i for each player i
  - no longer restricted to zero sum goals

- Nash equilibria (NE)
  - no incentive for any player to unilaterally change strategy
  - a strategy profile $\sigma=(\sigma_{1,\ldots,}\sigma_n)$ for a CSG is an ϵ–Nash equilibrium for state s and objectives $X_1,\ldots,X_n$ iff:
  - $E_s^{\sigma}(X_i) \geq \sup \{ E_s^{\sigma'}(X_i) \mid \sigma'=\sigma_{-i}[\sigma_i'] \text{ and } \sigma_i' \in \Sigma_i \} - \epsilon$ for all i
  - ϵ–NE (but not 0–NE) guaranteed to exist for CSGs

- Social welfare Nash equilibria (SWNE)
  - NE which maximise sum $E_s^{\sigma}(X_1) + \ldots E_s^{\sigma}(X_n)$
  - i.e., optimise combined goal

Example

- Example CSG: medium access control



- If objective X_i = probability for user i to send successfully:
  - 2 SWNEs when one user waits for the other to transmit and then transmits
- If objective X_i =probability of user i being *first* to transmit:
  - only 1 SWNE: both immediately try to transmit

rPATL + Nash operator

- Extension of rPATL for Nash equilibria [FM'19]

$$\phi ::= \text{true} \mid a \mid \neg\phi \mid \phi \wedge \phi \mid$$
$$\langle\langle C\rangle\rangle P_{\bowtie q}[\psi] \mid \langle\langle C\rangle\rangle R^r_{\bowtie x}\,[\rho] \mid \langle\langle C{:}C'\rangle\rangle_{\max\bowtie x}\,[\theta]$$

$$\theta ::= P[\psi]+P[\psi] \mid R^r[\rho]+R^r[\rho]$$

$$\psi ::= X\,\phi \mid \phi\ U^{\leq k}\ \phi \mid \phi\ U\ \phi$$

$$\rho ::= I^{=k} \mid C^{\leq k} \mid F\,\phi$$

- where:
  - $a\in AP$ is an atomic proposition, $C\subseteq N$ is a coalition of players and $C'=N\backslash C$, $\bowtie\ \in \{\leq,<,>,\geq\}$, $q \in [0,1]\cap\mathbb{Q}$, $x \in \mathbb{Q}_{\geq 0}$, $k \in \mathbb{N}$

    r is a reward structure

- Semantics:
  - $\langle\langle C{:}C'\rangle\rangle_{\max\bowtie x}\,[\theta]$ is satisfied if there exist strategies for all players that form a SWNE between coalitions C and $C'(=N\backslash C)$, and under which the *sum* of the two objectives in θ is $\bowtie x$

Model checking for extended rPATL

- Key ingredient is now:
  - solution of SWNEs for bimatrix games
  - (basic problem is EXPTIME)
  - we adapt known approach using labelled polytopes, and implement using an encoding to SMT

- Two types of model checking operator
  - bounded: backwards induction
  - unbounded: value iteration, e.g.:

$$\mathtt{V}_{\mathsf{G}^C}(s,\theta,n) = \begin{cases} (1,1) & \text{if } s \in Sat(\phi^1) \cap Sat(\phi^2) \\ (1, \mathtt{P}^{\max}_{\mathsf{G},s}(\mathtt{F}\ \phi^2)) & \text{else if } s \in Sat(\phi^1) \\ (\mathtt{P}^{\max}_{\mathsf{G},s}(\mathtt{F}\ \phi^1), 1) & \text{else if } s \in Sat(\phi^2) \\ (0,0) & \text{else if } n{=}0 \\ val(\mathsf{Z}_1, \mathsf{Z}_2) & \text{otherwise} \end{cases}$$

  - where Z_1 and Z_2 encode matrix games similar to before

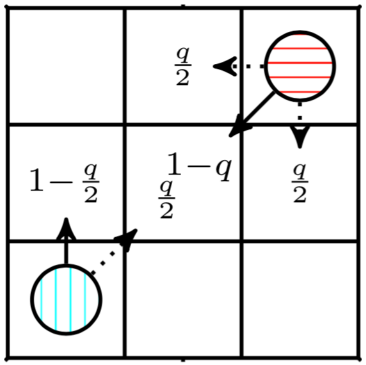
PRISM-games support

- Implementation in PRISM-games
  - extends CSG rPATL model checking implementation
  - bimatrix games solved using Z3/Yices encoding
  - optimised filtering of dominated strategies
  - scales up to CSGs with ~2 million states

- Applications
  - robot navigation in a grid, medium access control, Aloha communication protocol, power control
  - SWNE strategies outperform those found with rPATL
  - ϵ-Nash equilibria found typically have ϵ=0

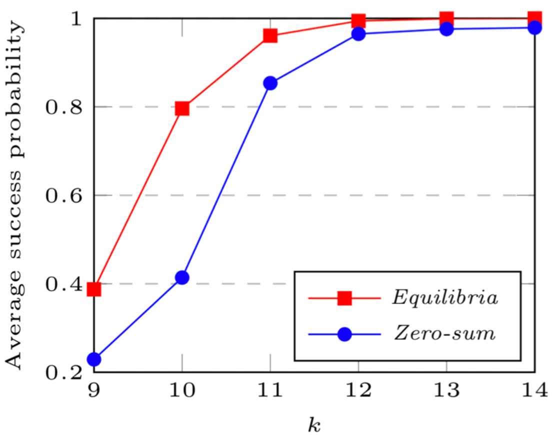
Example: multi-robot coordination

- 2 robots navigating an l x l grid
  - start at opposite corners, goals are to navigate to opposite corners
  - obstacles modelled stochastically: navigation in chosen direction fails with probability q



- We synthesise SWNEs to maximise the average probability of robots reaching their goals within time k
  - $\langle\langle robot1{:}robot2 \rangle\rangle_{max=?}$ (P [$F^{\leq k}$ $goal_1$]+P [$F^{\leq k}$ $goal_2$])

- Results (10 x 10 grid)
  - better performance obtained than using zero-sum methods, i.e., optimising for robot 1, then robot 2



Conclusions

- Probabilistic model checking & PRISM
  - verification & strategy synthesis
- Stochastic multi-player games
  - competitive/collaborative behaviour + stochasticity
  - rPATL model checking & strategy synthesis
  - concurrent stochastic games: more realistic models of competing stochastic components
  - Nash equilibria: beyond zero sum properties
- Challenges & directions
  - partial information/observability & greater efficiency
  - scalability, e.g. symbolic methods, abstraction
  - managing model uncertainty + integration with learning