# Synthesis for Multi-Objective Stochastic Games: An Application to Autonomous Urban Driving

Taolue Chen, Marta Kwiatkowska, Aistis Simaitis, and Clemens Wiltsche

Department of Computer Science, University of Oxford, United Kingdom

**Abstract.** We study strategy synthesis for stochastic two-player games with multiple objectives expressed as a conjunction of LTL and expected total reward goals. For stopping games, the strategies are constructed from the Pareto frontiers that we compute via value iteration. Since, in general, infinite memory is required for deterministic winning strategies in such games, our construction takes advantage of randomised memory updates in order to provide compact strategies. We implement our methods in PRISM-games, a model checker for stochastic multi-player games, and present a case study motivated by the DARPA Urban Challenge, illustrating how our methods can be used to synthesise strategies for high-level control of autonomous vehicles.

## 1 Introduction

The increasing reliance on sensor-enabled smart devices in a variety of applications, for example, autonomous parking and driving, medical devices, and communication technology, has called for software quality assurance technologies for the development of their embedded software controllers. To that end, techniques such as formal verification, validation and synthesis from specifications have been advocated, see e.g. [17, 3]. One advantage of synthesis is that it guarantees correctness of the controllers by construction. This technique has been successfully demonstrated for LTL (linear temporal logic) specifications in robotic control and urban driving [19, 21], where it can be used to synthesise controllers that maximise the probability of the system behaving according to its specification, which is expressed as an LTL formula. As a natural continuation of the aforementioned studies, one may wish to synthesise controllers that satisfy several objectives simultaneously. These objectives might be LTL formulae with certain probabilities, or a broader range of quantitative, reward-based specifications. Such *multi-objective* specifications enable the designers to choose the best controller for the given application by exploiting the Pareto trade-offs between objectives, such as increasing the probability of reaching a goal, while at the same time reducing the probability of entering an unsafe state or keeping the expected value of some reward function above a given value.

The majority of research in (discrete) controller synthesis from LTL specifications has centred on modelling in the setting of Markov chains or Markov decision processes (MDPs). In this paper, we focus on stochastic two-player games, which

generalise MDPs, and which provide a natural view of the system (represented as Player 1) playing a game against an adversarial environment (Player 2). In the setting of multi-objective stochastic games, several challenges for controller synthesis need to be overcome that we address in this paper. Firstly, the games are *not* determined, and thus determinacy cannot be leveraged by the synthesis algorithms. Secondly, the winning strategies for such games require both memory and randomisation; previous work has shown that the number of different probability distributions required by the winning strategy may be exponential or even infinite, and so one needs to adopt a representation of strategies which allows us to encode such distributions using finitely many memory elements.

Modelling via multi-objective stochastic games is well suited to navigation problems, such as urban driving, where the environment makes choices to which the system has to react by selecting appropriate responses that, for example, avoid obstacles or minimise the likelihood of accidents. The choices of the environment can be nondeterministic, but can also be modelled probabilistically, e.g. where statistical observations about certain hazards are available. In addition to probabilities, one can also annotate the model with rewards, to evaluate various quantities by means of expectations. For instance, we can model a trade-off between the probability $p$ of the car reaching a certain position without an accident, and the expected quality $r$ of the roads it drives over to get there. While both $(p, r) = (0.9, 4)$ and $(p', r') = (0.8, 5)$ might be achievable, the combination $(p'', r'') = (0.9, 5)$ could be unattainable by any controller, since $(p, r)$ and $(p', r')$ are already Pareto optimal.

In this paper we extend the results of [9], where verification techniques for multi-objective stochastic games were proposed. We focus here on quantitative multi-objective conjunctions for stochastic two-player games, where each property in the conjunction can be either an LTL formula or a reward function. We formulate a value iteration method to compute an approximation of the Pareto frontiers for exploring the trade-offs between different controllers and show how to construct control strategies. We also develop a prototype implementation as an extension of PRISM-games [7] using the Parma Polyhedra Library [1], and apply it to a case study of urban driving inspired by the 2007 DARPA Urban Challenge [11]. We construct and evaluate strategies for autonomous driving based on OpenStreetMap [16] data for a number of English villages.

**Contributions.** The paper makes the following contributions:

- We show how to solve stopping games with conjunctions of LTL objectives by reduction to reachability reward objectives.
- We formulate and implement algorithms for computing Pareto frontiers for reachability games using value iteration.
- We construct strategies from the results of the value iteration algorithm and evaluate their respective trade-offs.
- We present a case study of urban driving demonstrating all of the above.

**Related work.** Multi-objective optimisation has been applied in the context of formal verification, mainly for MDPs and (non-stochastic) games, where it

is usually referred to as multidimensional optimisation. For MDPs, [5, 2] introduced multiple discounted objectives and multiple long-run objectives, respectively, whereas [13] studied the problem of Boolean combinations of quantitative $\omega$-regular objectives, which were extended in [15] with total rewards. In the setting of non-stochastic games, multidimensional energy games were introduced, including their complexity and the strategy synthesis problem [4, 14, 6, 18]. In [8] stochastic games, where the objective of Player 1 is to achieve the expectation of an objective function "precisely $p$", were introduced. This problem is a special case of the problem that we study.

Our case study is inspired by the DARPA Urban Challenge [11], whose guidelines and technical evaluation criteria we use, and particularly the report by the winning team, which encourages the use of formal verification of urban driving systems [17]. Team Caltech suggested the use of temporal logic synthesis for high level planning [3]. This was later formulated for MDPs and LTL goals for autonomous vehicles in adversarial environments (pedestrians crossing a street) [19]. Receding horizon control [20] and incremental approaches [21] have been suggested to alleviate the computational complexity by leveraging structure specific to the autonomous control problems.

## 2 Preliminaries

Given a vector $\boldsymbol{x} \in \mathbb{R}^n_{\geq 0}$ and a scalar $z \in \mathbb{R}$, we write $x_i$ for the $i$-th component of $\boldsymbol{x}$ where $1 \leq i \leq n$, and $\boldsymbol{x} + z$ for the vector $(x_1 + z, \ldots, x_n + z)$. Moreover, the *dot product* of vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined by $\boldsymbol{x} \cdot \boldsymbol{y} \stackrel{\text{def}}{=} \sum_{i=1}^{n} x_i \cdot y_i$ and the sum of two sets of vectors $X, Y \subseteq \mathbb{R}^n_{\geq 0}$ is defined by $X + Y \stackrel{\text{def}}{=} \{\boldsymbol{x} + \boldsymbol{y} \mid \boldsymbol{x} \in X, \boldsymbol{y} \in Y\}$. Given a set $X \subseteq \mathbb{R}^n_{\geq 0}$, we define the *downward closure* of $X$ as $\mathsf{dwc}(X) \stackrel{\text{def}}{=} \{y \mid \exists x \,.\, x \in X \text{ and } y \leq x\}$, and its convex hull as $\mathsf{conv}(X) \stackrel{\text{def}}{=} \{x \mid \exists x_1, x_2 \in X, \alpha \in [0,1] \,.\, x = \alpha x_1 + (1-\alpha) x_2\}$.

A *discrete probability distribution* over a (countable) set $S$ is a function $\mu : S \to [0,1]$ such that $\sum_{s \in S} \mu(s) = 1$. We write $\mathcal{D}(S)$ for the set of all discrete distributions over $S$. Let $\mathsf{supp}(\mu) \stackrel{\text{def}}{=} \{s \in S \mid \mu(s) > 0\}$ be the *support* of $\mu \in \mathcal{D}(S)$. A distribution $\mu \in \mathcal{D}(S)$ is a *Dirac distribution* if $\mu(s) = 1$ for some $s \in S$. Sometimes we identify a Dirac distribution $\mu$ with the unique element in $\mathsf{supp}(\mu)$. We represent a distribution $\mu \in \mathcal{D}(S)$ on a set $S = \{s_1, \ldots, s_n\}$ as a map $[s_1 \mapsto \mu(s_1), \ldots, s_n \mapsto \mu(s_n)] \in \mathcal{D}(S)$, and we usually omit the elements of $S$ outside $\mathsf{supp}(\mu)$ to simplify the presentation.

**Definition 1 (Stochastic two-player game).** *A* stochastic two-player game *(called* game *henceforth) is a tuple* $\mathcal{G} = \langle S, (S_\Box, S_\Diamond, S_\bigcirc), \Delta \rangle$*, where*

- $S$ *is a countable set of states partitioned into sets* $S_\Box$*,* $S_\Diamond$*, and* $S_\bigcirc$*; and*
- $\Delta : S \times S \to [0,1]$ *is a transition function such that* $\Delta(\langle s, t \rangle) \in \{0, 1\}$ *if* $s \in S_\Box \cup S_\Diamond$*, and* $\sum_{t \in S} \Delta(\langle s, t \rangle) = 1$ *if* $s \in S_\bigcirc$*.*

$S_\Box$ and $S_\Diamond$ represent the sets of states controlled by players Player 1 and Player 2, respectively, while $S_\bigcirc$ is the set of stochastic states. For a state $s \in S$, the set of successor states is denoted by $\Delta(s) \stackrel{\text{def}}{=} \{t \in S \mid \Delta(\langle s,t\rangle)>0\}$. We assume that $\Delta(s) \neq \emptyset$ for all $s \in S$. Moreover, we denote a set of *terminal states* $\mathsf{Term} \stackrel{\text{def}}{=} \{s \in S \mid \Delta(\langle s,t\rangle)=1 \text{ iff } s=t\}$.

An infinite *path* $\lambda$ of a stochastic game $\mathcal{G}$ is an infinite sequence $s_0 s_1 \ldots$ of states such that $s_{i+1} \in \Delta(s_i)$ for all $i \geq 0$. A finite path is a finite such sequence. For a finite or infinite path $\lambda$ we write $\mathsf{len}(\lambda)$ for the number of states in the path. For $i < \mathsf{len}(\lambda)$ we write $\lambda_i$ to refer to the $i$-th state $s_i$ of $\lambda$, and we denote the suffix of the path $\lambda$ starting at position $i$ by $\lambda^i \stackrel{\text{def}}{=} s_i s_{i+1} \ldots$. For a finite path $\lambda = s_0 s_1 \ldots s_n$ we write $\mathsf{last}(\lambda)$ for the last state of the path, i.e., $\mathsf{last}(\lambda) = s_n$. We write $\Omega_{\mathcal{G},s}$ for the set of infinite paths starting in state $s$.

**Strategy.** In this paper we use an alternative formulation of strategies [2] that generalises the concept of strategy automata [12].

**Definition 2.** *A* strategy *of Player 1 in a game* $\mathcal{G} = \langle S, (S_\Box, S_\Diamond, S_\bigcirc), \Delta\rangle$ *is a tuple* $\pi = \langle \mathcal{M}, \pi_u, \pi_n, \alpha\rangle$*, where:*

- $\mathcal{M}$ *is a countable set of* memory elements*,*
- $\pi_u \colon \mathcal{M} \times S \to \mathcal{D}(\mathcal{M})$ *is a* memory update function*,*
- $\pi_n \colon S_\Box \times \mathcal{M} \to \mathcal{D}(S)$ *is a* next move function *s.t.* $\pi_n(s,m)[s']>0$ *only if* $s' \in \Delta(s)$*,*
- $\alpha \colon S \to \mathcal{D}(\mathcal{M})$ *defines for each state of* $\mathcal{G}$ *an* initial memory distribution

*A strategy* $\sigma$ *for Player 2 is defined in an analogous manner. We denote the set of all strategies for Player 1 and Player 2 by* $\Pi$ *and* $\Sigma$*, respectively.*

A strategy is *memoryless* if $|\mathcal{M}| = 1$. We say that a strategy requires finite memory if $|\mathcal{M}| < \infty$ and infinite memory if $|\mathcal{M}| = \infty$. We also classify the strategies based on the use of randomisation. A strategy $\pi = \langle \mathcal{M}, \pi_u, \pi_n, \alpha\rangle$ is *pure* if $\pi_u$, $\pi_n$, and $\alpha$ map to Dirac distributions; *deterministic update* if $\pi_u$ and $\alpha$ map to Dirac distributions, while $\pi_n$ maps to an arbitrary distributions; and *stochastic update* where $\pi_u$, $\pi_n$, and $\alpha$ can map to arbitrary distributions.

**Markov chain induced by strategy pairs.** Given a game $\mathcal{G}$ with initial state distribution $\varsigma \in \mathcal{D}(S)$, a Player 1 strategy $\pi = \langle \mathcal{M}_1, \pi_u, \pi_n, \alpha_1\rangle$ and a Player 2 strategy $\sigma = \langle \mathcal{M}_2, \sigma_u, \sigma_n, \alpha_2\rangle$ induce a countable Markov chain $\mathcal{G}(\varsigma,\pi,\sigma) = \langle S', (\emptyset,\emptyset,S'), \Delta'\rangle$ with initial state distribution $\varsigma(\pi,\sigma) \in \mathcal{D}(S')$, where

- $S' = S \times \mathcal{M}_1 \times \mathcal{M}_2$,
- $\Delta' \colon S' \times S' \to [0,1]$ is such that for all $(s,m_1,m_2),(s',m_1',m_2') \in S'$ we have $(\langle (s,m_1,m_2),(s',m_1',m_2')\rangle) \mapsto$

$$\begin{cases} \pi_n(s,m_1)[s'] \cdot \pi_u(m_1,s')[m_1'] \cdot \sigma_u(m_2,s')[m_2'] & \text{if } s \in S_\Box \\ \sigma_n(s,m_2)[s'] \cdot \pi_u(m_1,s')[m_1'] \cdot \sigma_u(m_2,s')[m_2'] & \text{if } s \in S_\Diamond \\ \Delta(\langle s,s'\rangle) \cdot \pi_u(m_1,s')[m_1'] \cdot \sigma_u(m_2,s')[m_2'] & \text{if } s \in S_\bigcirc, \end{cases}$$

- $\varsigma(\pi,\sigma) : S' \to [0,1]$ is defined such that for all $(s,m_1,m_2) \in S'$ we have that $\varsigma(\pi,\sigma)[s,m_1,m_2] = \varsigma[s] \cdot \alpha_1(s)[m_1] \cdot \alpha_2(s)[m_2]$.

**Probability measure.** A stochastic game $\mathcal{G}$ together with a strategy pair $(\pi, \sigma) \in \Pi \times \Sigma$ and a starting state $s$ induces a (possibly infinite) Markov chain on the game. We define the probability measure over the set of paths $\Omega_{\mathcal{G},s}$ and a strategy pair $(\pi, \sigma)$ in the following way. The basic open sets of $\Omega_{\mathcal{G},s}$ are the *cylinder sets* $\mathrm{Cyl}(\lambda) \stackrel{\text{def}}{=} \lambda \cdot S^\omega$ for every finite path $\lambda = s_0 s_1 \dots s_k$ of $\mathcal{G}$, and the probability assigned to $\mathrm{Cyl}(\lambda)$ equals $\prod_{i=0}^{k} \Delta(\langle s_i, s_{i+1}\rangle) \cdot p_i(s_0, \dots, s_i)$, where $p_i(\lambda) = \pi(\lambda)$ if $\mathsf{last}(\lambda) \in S_\Box$, $p_i = \sigma(\lambda)$ if $\mathsf{last}(\lambda) \in S_\Diamond$ and 1 otherwise. This definition induces a probability measure on the algebra of cylinder sets, which can be extended to a unique probability measure $\Pr^{\pi,\sigma}_{\mathcal{G},s}$ on the $\sigma$-algebra generated by these sets. The expected value of a measurable function $f\colon S^\omega \to \mathbb{R} \cup \{\infty\}$ under a strategy pair $(\pi, \sigma) \in \Pi \times \Sigma$ is defined as $\mathbb{E}^{\pi,\sigma}_{\mathcal{G},s}[f] \stackrel{\text{def}}{=} \int f \, d\Pr^{\pi,\sigma}_{\mathcal{G},s}$. We say that a game $\mathcal{G}$ is a *stopping game* if for every pair of strategies $\pi$ and $\sigma$ a terminal state is reached with probability 1.

**Winning objectives.** In this paper we study objectives which are conjunctions of LTL and expected total reward goals. This section provides definitions of these concepts, as well as that of Pareto frontiers representing the possible trade-offs.

*LTL.* To specify the LTL goals, we use the following standard notation:

$$\Xi ::= T \mid \neg\Xi \mid \Xi_1 \wedge \Xi_2 \mid \mathsf{X}\,\Xi \mid \Xi_1 \,\mathsf{U}\, \Xi_2,$$

where $T \subseteq S$. Given a path $\lambda$ and a LTL formula $\Xi$, we define $\lambda \models \Xi$ as:

$$\begin{aligned}
\lambda &\models T && \Leftrightarrow \lambda_0 \in T \\
\lambda &\models \neg\Xi && \Leftrightarrow \lambda \not\models \Xi \\
\lambda &\models \Xi_1 \wedge \Xi_2 && \Leftrightarrow \lambda \models \Xi_1 \text{ and } \lambda \models \Xi_2 \\
\lambda &\models \mathsf{X}\,\Xi && \Leftrightarrow \lambda^1 \models \Xi \\
\lambda &\models \Xi_1 \,\mathsf{U}\, \Xi_2 && \Leftrightarrow \lambda^i \models \Xi_2 \text{ for some } i \in \mathbb{N}_0 \\
& && \qquad \text{and } \lambda^j \models \Xi_1 \text{ for } 0 \le j < i\,.
\end{aligned}$$

Operators $\mathsf{F}\,\Xi \stackrel{\text{def}}{=} S\,\mathsf{U}\,\Xi$ and $\mathsf{G}\,\Xi \stackrel{\text{def}}{=} \neg\mathsf{F}\neg\Xi$ have their usual meaning. We use a formula and the set of paths satisfying the formula interchangeably, e.g. the set of paths reaching a state in $T \subseteq S$ is denoted by $\mathsf{F}\,T = \{\omega \in \Omega_{\mathcal{G}} \mid \exists i\,.\,\omega_i \in T\}$.

*Expected total reward.* To specify the reward goals, we define a ($k$-dimensional) reward function $\boldsymbol{r} : S \to \mathbb{R}^k_{\ge 0}$, which for each state $s$ of the game $\mathcal{G}$ assigns a reward vector $\boldsymbol{r}(s) \in \mathbb{R}^k_{\ge 0}$. We define a vector of *total reward* random variables $rew(\boldsymbol{r})$ as $rew(\boldsymbol{r})(\lambda) \stackrel{\text{def}}{=} \sum_{j \ge 0} \boldsymbol{r}(\lambda_j)$ for any path $\lambda$. Under a fixed strategy pair $(\pi, \sigma)$ for both players, the expected total reward is the expected value of the total reward random variable, i.e.,

$$\mathbb{E}^{\pi,\sigma}_{\mathcal{G},s}[rew(\boldsymbol{r})] = \int_{\Omega_{\mathcal{G},s}} rew(\boldsymbol{r}) \, d\Pr^{\pi,\sigma}_{\mathcal{G},s}.$$

We require the expected total rewards to be bounded. Hence, due to the self-loops in terminal states, in particular we require $\boldsymbol{r}(s) = 0$ for all $s \in \mathsf{Term}$.

*Conjunctive queries.* A *conjunctive query* (CQ) is a tuple of LTL formulae, reward functions, and their respective *lower* bounds,

$$\varphi = (\boldsymbol{\Xi}, \boldsymbol{r}, \boldsymbol{v}) = ((\Xi_1, \dots, \Xi_m), (r_1, \dots, r_{n-m}), (v_1, \dots, v_m, v_{m+1}, \dots, v_n)),$$

where $m$ is the number of LTL objectives, $n-m$ is the number of expected total reward objectives, $v_i \in [0,1]$ for $1 \leq i \leq m$ and $v_i \in \mathbb{R}$ for $m+1 \leq i \leq n$, $\Xi_i$ is an LTL formula, and $r_i$ is a reward function. We call $\boldsymbol{v}$ the *target vector* of the CQ. Additionally, we define a *reward conjunctive query* (rCQ) to be a CQ with $m = 0$. Player 1 *achieves* a CQ $\varphi$ at state $s$ if there exists a strategy $\pi \in \Pi$ such that, for any strategy $\sigma \in \Sigma$ of Player 2, it holds that

$$\bigwedge_{i=1}^{m} (\Pr_{\mathcal{G},s}^{\pi,\sigma}(\Xi_i) \geq v_i) \wedge \bigwedge_{i=m+1}^{n} (\mathbb{E}_{\mathcal{G},s}^{\pi,\sigma}[rew(\boldsymbol{r})_{i-m}] \geq v_i).$$

**Pareto frontiers.** A *Pareto frontier* for objectives $\boldsymbol{\Xi}$ and $\boldsymbol{r}$ is a set of points $P \subseteq \mathbb{R}^n$ such that for any $\boldsymbol{p} \in P$ the following hold:

(a) for all $\varepsilon > 0$ the query $(\boldsymbol{\Xi}, \boldsymbol{r}, \boldsymbol{p} - \varepsilon)$ is achievable, and
(b) for all $\varepsilon > 0$ the query $(\boldsymbol{\Xi}, \boldsymbol{r}, \boldsymbol{p} + \varepsilon)$ is not achievable.

Given $\varepsilon > 0$, an *$\varepsilon$-approximation* of the Pareto frontier $P$ is a set of points $Q$ satisfying that for any $\boldsymbol{q} \in Q$ there is a point $\boldsymbol{p} \in P$ such that $\|\boldsymbol{p} - \boldsymbol{q}\| \leq \varepsilon$, and for every $\boldsymbol{p} \in P$ there is a vector $\boldsymbol{q} \in Q$ such that $\|\boldsymbol{p} - \boldsymbol{q}\| \leq \varepsilon$, where $\|\cdot\|$ is the Manhattan distance of two points in $\mathbb{R}$.

## 3 Computing the Pareto Frontiers

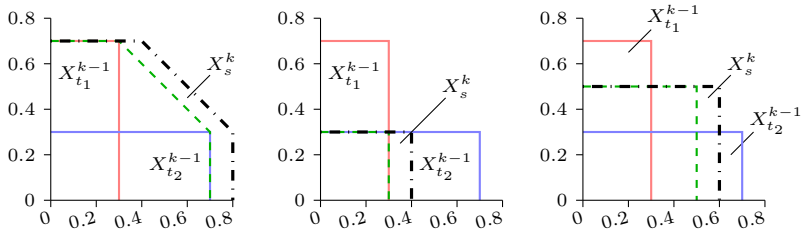
We now turn our attention to the computation of Pareto frontiers for the states of the game, which will be required for the strategy construction. First, we recall from our work in [9] how to compute them for rCQs, and then extend this by showing how a stopping game $\mathcal{G}$ with a general CQ $\varphi$ can be reduced to a stopping game $\mathcal{G}'$ with a rCQ $\varphi'$, such that there exists a strategy for Player 1 in $\mathcal{G}$ to satisfy $\varphi$ if and only if there is a strategy for Player 1 in $\mathcal{G}'$ to satisfy $\varphi'$.

**Value iteration.** To compute the successive approximations of the Pareto frontiers we iteratively apply the functional from the theorem below. For stopping games this is guaranteed to compute an $\varepsilon$-approximation of the Pareto frontiers.

**Theorem 1 (Pareto frontier approximation).** *For a stopping game $\mathcal{G}$ and a rCQ $\varphi = (\boldsymbol{r}, \boldsymbol{v})$, an $\varepsilon-$approximation of the Pareto frontiers for all states can be computed in $k = |S| + \lceil |S| \cdot \frac{\ln(\varepsilon \cdot (n \cdot M)^{-1})}{\ln(1-\delta)} \rceil$ iterations of the operator $F : (S \to \mathcal{P}(\mathbb{R}^n_{\geq 0})) \to (S \to \mathcal{P}(\mathbb{R}^n_{\geq 0}))$ defined by*

$$F(X)(s) \stackrel{\text{def}}{=} \begin{cases} \mathit{dwc}(\mathit{conv}(\bigcup_{t \in \Delta(s)} X_t) + \boldsymbol{r}(s)) & \text{if } s \in S_\Box \\ \mathit{dwc}(\bigcap_{t \in \Delta(s)} X_t + \boldsymbol{r}(s)) & \text{if } s \in S_\Diamond \\ \mathit{dwc}(\sum_{t \in \Delta(s)} \Delta(\langle s,t \rangle) \times X_t + \boldsymbol{r}(s)) & \text{if } s \in S_\bigcirc, \end{cases}$$

*where initially $X_s^0 \stackrel{\text{def}}{=} \{\boldsymbol{x} \in \mathbb{R}^n_{\geq 0} \mid \boldsymbol{x} \leq \boldsymbol{r}(s)\}$ for all $s \in S$, $M = |S| \cdot \frac{\max_{s \in S, i} r_i(s)}{\delta}$ for $\delta = p_{\min}^{|S|}$, and $p_{\min}$ is the smallest positive probability in $\mathcal{G}$.*




(a) Player 1 state $s \in S_\Box$. (b) Player 2 state $s \in S_\Diamond$. (c) Stochastic state $s \in S_\bigcirc$. $\Delta(\langle s, t_1 \rangle) = \Delta(\langle s, t_2 \rangle) = \frac{1}{2}$.

**Fig. 1.** Illustration of value iteration operations for rCQs. We plot the polytopes of $s$ and its successors $t_1$ and $t_2$ on the same graph. The reward at $s$ is $\boldsymbol{r}(s) = (0, 0.1)$ and $\Delta(s) = \{t_1, t_2\}$. The dashed (green) polytope is the result before adding the reward, which is shown in the dash-dotted (red) polytope.

The value iteration computes a sequence of polytopes $X^k_s$ for each state $s$ that converges to the Pareto frontiers. It starts with the bottom element $\bot : S \to \mathcal{P}(\mathbb{R}^n_{\geq 0})$ defined by $\bot(s) \stackrel{\text{def}}{=} X^0_s$. The operations in the operator $F$ are illustrated in Figure 1. We assume that the expected reward for the game is bounded under any pair of strategies, which in particular implies that, for any terminal state $t \in \mathsf{Term}$, $\boldsymbol{r}(t) = \mathbf{0}$.

**Strategy iteration.** We use the sets $X^k$ computed using the above iteration to obtain the memory elements of the stochastic update strategy (see Section 4). Thus, the iteration performed above can be regarded as a strategy iteration algorithm: it generates strategies that are optimal for achieving the goal in $k$ steps (initially $k = 1$). After performing another iteration, we obtain the optimal strategy for $k+1$ steps, etc. Since all rewards are non-negative, $F$ is monotone, i.e. $X^k_s \subseteq X^{k+1}_s$ for all states $s$, and thus in every step the strategy either improves or it has converged to the optimal one. The strategy iteration can be performed either until the objective vector $\boldsymbol{v}$ is in $X^k_{s_0}$, or until it is within the required accuracy determined by the stopping criterion.

**LTL to expected reward.** We present a reduction from LTL objectives to expected total reward objectives for stopping games. The reduction is based on a similar reduction for Markov decision processes ([10, 13]) which relies on constructing a deterministic Rabin automaton for each LTL formula and then building their product with the game $\mathcal{G}$. As the game $\mathcal{G}$ is stopping, almost all runs reach some terminal state with a positive probability. Hence, it is sufficient to analyse the outcomes of the formulae in the *terminal* states of the product game in order to establish whether the runs ending in them satisfy the formula.

**Definition 3 (Rabin automaton).** *A* deterministic Rabin automaton *is a tuple* $\langle Q, \Sigma, \tau, q^0, ((L^1, R^1), \ldots, (L^j, R^j)) \rangle$, *where* $Q$ *are the states of the automaton with initial state* $q^0 \in Q$, $\Sigma = S$ *is the alphabet,* $\tau : Q \times \Sigma \to Q$ *is a transition function and* $L^l, R^l \subseteq Q$ *are Rabin pairs.*

**Theorem 2 (LTL to expected reward).** *Given a stopping game* $\mathcal{G}$, *a state* $s$ *and a CQ* $\varphi = ((\Xi_1, \ldots, \Xi_m), (r_1 \ldots r_{n-m}), \boldsymbol{v})$, *there exists a game* $\mathcal{G}'$, *a state*

*$s'$, and a rCQ $\varphi' = ((r'_1 \dots r'_n), \boldsymbol{v})$ such that there is a Player 1 strategy to satisfy $\varphi$ in $s$ of $\mathcal{G}$ if and only if there is a Player 1 strategy to satisfy $\varphi'$ in $s'$ of $\mathcal{G}'$.*

For each LTL formula $\Xi_i$, we construct a *deterministic* Rabin automaton $\mathcal{A}_i = \langle Q_i, \Sigma_i, \tau_i, q_i^0, ((L_i^1, R_i^1), \dots, (L_i^j, R_i^j))\rangle$ with $\Sigma_i = S$, such that any path $\lambda$ satisfies $\Xi_i$ iff $\lambda$ is accepted by $\mathcal{A}_i$. W.l.o.g. we assume each DRA is *complete*.

We index elements of tuples by subscripts, e.g., for a tuple $s' = (s, q^0)$, $s'_0 = s$ and $s'_1 = q^0$. Given a stopping game $\mathcal{G} = \langle S, (S_\Box, S_\Diamond, S_\bigcirc), \Delta\rangle$ and DRAs $\mathcal{A}_i$ for $1 \leq i \leq m$, we construct a stopping game $\mathcal{G}' = \langle S', (S'_\Box, S'_\Diamond, S'_\bigcirc), \Delta'\rangle$, where:

- $S' = S \times Q_1 \times \dots \times Q_m \cup \{term\}$, $S'_\# = \{s' \in S' \mid s'_0 \in S_\#\}$ for $\# \in \{\Box, \Diamond, \circ\}$;
- for states $s', t' \in S'$, using $T' \stackrel{\text{def}}{=} \{t' \in S' \,|\, t'_0 \in \mathsf{Term}\}$,

$$\Delta'(s', t') = \begin{cases} \Delta(s'_0, t'_0) & \text{if } s' \notin T' \text{ and } \forall\,.\, 1 \leq i \leq m,\ \tau_i(s'_i, t'_0) = t'_i \\ 1 & \text{if } s' \in T' \cup \{term\} \text{ and } t' = term \\ 0 & \text{otherwise;} \end{cases}$$

- the initial state is $s' \in S'$ s.t. $s'_0 = s$ and, for all $1 \leq i \leq m$, $\tau_i(q_i^0, s) = s'_i$.

The new multi-objective query is $\varphi' = ((r'_1, \dots, r'_n), (v_1, \dots, v_n))$, where each $r'_i$ for $1 \leq i \leq m$ is defined as

$$r'_i(s') = \begin{cases} 1 & \text{if } s' \in T' \text{ and } {s'_0}^\omega \text{ is accepted by } \mathcal{A}_i \text{ with initial state } s'_i \\ 0 & \text{otherwise,} \end{cases}$$

and for $m+1 \leq i \leq n$, $r'_i(s') = r_{i-m}(s'_0)$ for $s' \neq term$ and $r'_i(term) = 0$, i.e., the reward functions of $\varphi$ for $m+1 \leq i \leq n$ stay unchanged with respect to $S$. Correctness follows by the standard argument of one-to-one correspondence between strategies in $\mathcal{G}'$ and $\mathcal{G}$ for both players, see e.g. [15].

## 4 Strategy Synthesis

In this section we present a construction of the strategy for Player 1, which, given a stopping game $\mathcal{G}$, the sets $X^k$ computed using the iteration from Section 3, and a state $s$, achieves the rCQ $\varphi = (\boldsymbol{r}, \boldsymbol{v})$, where $\boldsymbol{v} \in X_s^k$. The resulting strategy uses random updates on its memory elements to ensure that at every step the expected total reward is kept above the target vector $\boldsymbol{v}$. The assumption that the game $\mathcal{G}$ is stopping implies that $\mathcal{G}$ terminates with probability 1, and thus the target vector is achieved. Note that the stochastic *memory update* is crucial to provide a compact representation of the strategy, as our previous work has demonstrated the need for exponential (Proposition 3 in [8]), or even infinite (Theorem 2 in [9]) memory if the strategy is only allowed to use randomisation in Player 1 states.

We denote the set of vertices (corner points) of a polytope $X$ as $Cnr(X)$. The strategy $\pi = \langle \mathcal{M}, \pi_u, \pi_n, \alpha\rangle$ is defined as follows.

- $\mathcal{M} = \bigcup_{t' \in S'} \{(t', \boldsymbol{p}) \mid \boldsymbol{p} \in Cnr(X^k_{t'})\}$.
- $\pi_u((t', \boldsymbol{p}), u) = [(u', \boldsymbol{q}_0^{u'}) \mapsto \beta_0^u, \ldots, (u', \boldsymbol{q}_l^{u'}) \mapsto \beta_l^u]$, where $t', u' \in S'$ such that $t'_0 = t, u'_0 = u$ for $t, u \in S$, and such that $\Delta'(t', u') > 0$ (note that because DRAs are deterministic, such $u'$ is unique), and where for all $0 \leq i \leq l$, $\boldsymbol{q}_i^{u'} \in Cnr(X^k_{u'})$, $\beta_i^u \in [0, 1]$, and $\sum_i \beta_i^u = 1$, such that
  - for $t' \in S'_\Box \cup S'_\Diamond$ we have $\sum_i \beta_i^u \cdot \boldsymbol{q}_i^{u'} \geq \boldsymbol{p} - \boldsymbol{r}'(t)$,
  - for $t' \in S'_\bigcirc$, $\boldsymbol{q}_i^{u'}$ and $\beta_i^u$ have to be chosen together with the respective values $\boldsymbol{q}_i^{v'}$, and $\beta_i^v$ assigned by $\pi_u((t', \boldsymbol{p}), v)$ for the remaining successors $v \in S \setminus \{u\}$ of $t$, so that they satisfy

$$\Delta(t, u) \cdot \sum_i \beta_i^u \cdot \boldsymbol{q}_i^{u'} + \sum_{v \in S \setminus \{u\}} \Delta(t, v) \cdot \sum_i \beta_i^v \cdot \boldsymbol{q}_i^{v'} \geq \boldsymbol{p} - \boldsymbol{r}'(t'),$$

    which ensures that the expected total reward is kept larger than the current memory element.
- $\pi_n(t, (t', \boldsymbol{p})) = [u \mapsto 1]$ for some $u \in S$ such that $\Delta'(t', u') > 0$ (where $u' \in S'$, $u'_0 = u$), and for all $0 \leq i \leq l$ there exist $\boldsymbol{q}_i^{u'} \in Cnr(X^k_{u'})$, $\beta_i^u \in [0, 1]$, such that $\sum_i \beta_i^u = 1$ and $\sum_i \beta_i^u \cdot \boldsymbol{q}_i^{u'} \geq \boldsymbol{p} - \boldsymbol{r}'(t')$.
- $\alpha(s) = [(s', \boldsymbol{q}_0^{s'}) \mapsto \beta_0^s, \ldots, (s', \boldsymbol{q}_l^{s'}) \mapsto \beta_l^s]$, where $s'$ is the respective initial state of $\mathcal{G}'$, and $\boldsymbol{q}_i^{s'} \in Cnr(X^k_{s'})$, $\beta_i^s \in [0, 1]$ (for all $0 \leq i \leq l$), and $\sum_i \beta_i^s = 1$ such that $\sum_i \beta_i^s \cdot \boldsymbol{q}_i^{s'} \geq \boldsymbol{v}'$.

Note that, for all $u' \in S'$, it is always possible to choose $l \leq n$, i.e. the number of points $\boldsymbol{q}_i^{u'}$ and respective coefficients $\beta_i^u$ may be less than the number of objectives. Also, the points $\boldsymbol{q}_i^{t'}$ can indeed be picked from $X^k_{t'}$ because they exist both in $X^{k-1}_{t'}$, and $X^k_{t'} \supseteq X^{k-1}_{t'}$ due to the monotonicity of $F$.

**Theorem 3.** *The strategy constructed above achieves the expectation for the total reward functions which is greater than or equal to $\boldsymbol{v}$ for the rCQ $\varphi$.*

The proof follows the structure from [8], establishing, by induction on the length of finite prefixes, that the expectation of the vector held in memory is always between the target vector $\boldsymbol{v}$ and the expected total reward.

## 5 Case Study

We implement a prototype multi-objective strategy synthesis engine in PRISM-games, a model checking and synthesis tool for stochastic games [7], and present a case study applying our value iteration and strategy synthesis methods to perform control tasks. Our case study is motivated by the DARPA Urban Challenge 2007 (henceforth referred to as the Challenge), a competition for autonomous cars to navigate safely and effectively in an urban setting [11]. In the Challenge, cars were exposed to a set of traffic situations, and had to plan routes, avoid hazards, and cope with the presence of other vehicles.

| Hazard | Abbreviation | $\lambda$ | Reaction | Accident Probability |
|---|---|---|---|---|
| Pedestrian | p | 0.05 | Brake | 0.01 |
| | | | Honk | 0.04 |
| | | | Change Lane | 0.03 |
| Jam | j | 0.1 | Honk | 0.01 |
| | | | U-Turn | 0.02 |
| Obstacle | o | 0.02 | Change Lane | 0.02 |
| | | | U-Turn | 0.02 |

**Table 1.** Model parameters for our prototype.

### 5.1 Problem Setting

We identify desired functions of vehicle controllers from the Challenge, and model a scenario of a car driving through a map imported from OpenStreetMap [16].

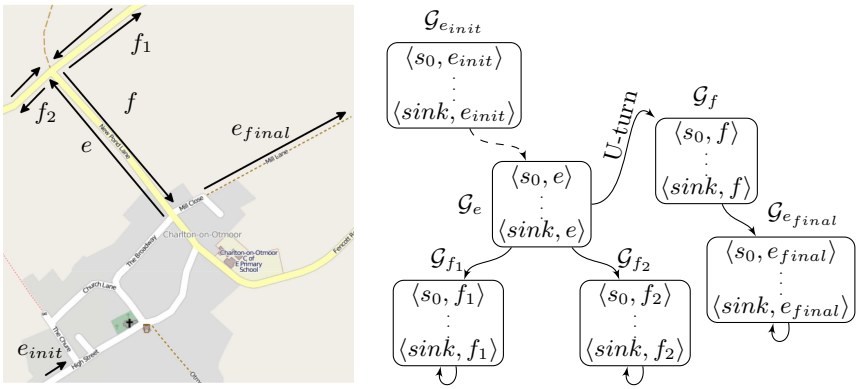
We model the problem as a two-player stochastic game, where Player 1 represents the car navigating through the map and Player 2 represents the environment. The game setting allows us to model the nondeterministic, adversarial nature of hazards selected by the environment, and the car's reaction to each hazard is represented by Player 1 strategy. Probabilities are used to model the relative likelihood of events in a given road segment and between different road segments, and are understood to be the parameters obtained from statistical observations; for example, certain road types are more prone to accidents. Finally, considering multiple objectives enables the exploration of trade-offs when constructing controllers.

Each road segment is modelled as part of a stochastic game between the environment (Player 2) that selects from a set of available hazards, and the car (Player 1) that selects reactions to the hazards, as well as making route selection (steering) decisions. Hazards occur with certain probabilities depending on the properties of the road segment the car is on, and each reaction the car takes is only successful with a given probability. We also model other parameters of the road, e.g. we use rewards for road quality. In our autonomous driving scenario, we consider three types of hazards with the corresponding reactions the car can take (see Table 1). These hazards, as well as the reactions, are chosen according to the Challenge event guidelines and technical evaluation criteria [11].

### 5.2 Model

We model the road network as a directed graph $G = (V, E)$, where each edge $e \in E$ represents a road segment, see Figure 2(a). To each edge $e \in E$, we associate a subgame $\mathcal{G}_e$, parameterised by the properties of the corresponding road segment (e.g., length, quality, number of lanes).

An example illustrating a subgame $\mathcal{G}_e$ is shown in Figure 3. The states have labels of the form $\langle s, e\rangle$; when the context is clear, we simply use $s$. In state $s_0$, a set of at most two hazards is selected probabilistically, from which Player 2 can then choose. To each hazard $h \in \{\mathrm{p}, \mathrm{j}, \mathrm{o}\}$ we associate a tuning parameter $\lambda$, and let the probability of a set of hazards $\{h_1, h_2\}$ be $p_{h_1 h_2} \stackrel{\text{def}}{=} \tanh(\lambda_1 \lambda_2 \mathsf{len}(e))/k$,




(a) Map of Charlton-on-Otmoor, UK, with overlaid graph (only partially shown).

(b) Connection of subgames. All roads that are not one ways have U-turn connections, but only one is shown for the sake of clarity. The dashed arrow abstracts several intermediate subgames.

**Fig. 2.** Illustrating the graph $G = (V, E)$ and the corresponding subgame connections.

where $\mathsf{len}(e)$ is the length of the road corresponding to $e$ in meters, and $k = 6$ is the number of sets of hazards.[1] For a single hazard $h$, $p_h$ is defined similarly, and the empty set of hazards is chosen with the residual probability $p_{\text{none}}$. The parameters for our prototype model are given in Table 1.

Once a set of possible hazards is chosen in $s_0$, and Player 2 has selected a specific one in $s_1$, $s_2$ and $s_3$, Player 1 must select an appropriate reaction in $s_4$, $s_5$ and $s_6$. Then the game enters either the terminal accident state "acc", or a Player 1 state "sink", where the next edge can be chosen. If the reaction is not appropriate in the given road segment (e.g., changing lane in a single lane road), a "violation" terminal is entered with probability 1 (not shown in Figure 3).

From the subgames $\mathcal{G}_e$ and the graph $G$ a game $\mathcal{G}$ is constructed that connects the games $\mathcal{G}_e$ as shown in Figure 2(b). In a local sink, e.g. $\langle sink, e\rangle$, Player 1 makes the decision as to which edge to go to next and enters the corresponding local initial state, e.g. $\langle s_0, f_1\rangle$. Also, in a U-turn, the subgame $\mathcal{G}_f$ of the reverse edge $f$ of $e$ is entered at its initial state, e.g. $\langle s_0, f\rangle$. If a road segment does not have any successors, or is the goal, the local sink for the corresponding game is made into a terminal state. Note that the above construction results in a stopping game.

### 5.3 Objectives

We study three objectives for the autonomous driving scenario. Formally, we consider the CQ $\varphi = ((\mathsf{F}\, T_1, \mathsf{G}\, \neg T_2), (r), (v_1, v_2, v_3))$, where $T_1 = \{\langle s_0, e_{final}\rangle\}$, $T_2 = \{\langle acc, e\rangle \mid e \in E\}$, and $r$ is a reward function explained below.

[1] The use of the sigmoid tanh achieves a valid probability distribution independently of the road length, while the weights $\lambda$ tune the relative frequency of hazard occurrence.

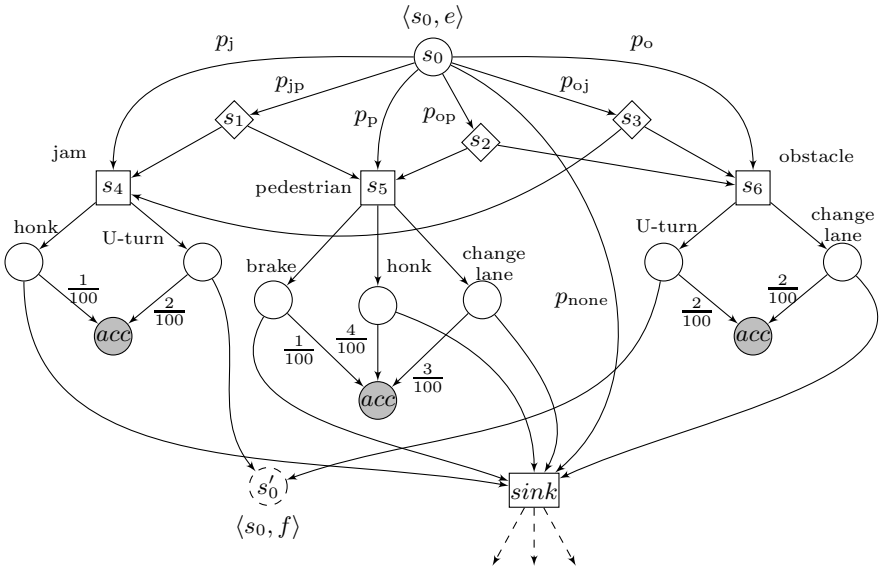



**Fig. 3.** Subgame $\mathcal{G}_e$ with reverse edge $f$ of $e$. The reward in *sink* is $\mathrm{rval}(e)\mathsf{len}(e)$. Hazards and reactions are indicated as annotations.

**Target location reachability.** From the initial location, reach a target at a particular orientation with probability $v_1$, i.e. achieve $\Pr^{\pi,\sigma}_{\mathcal{G},\langle s_0,e_{init}\rangle}(\mathsf{F}\, T_1) \geq v_1$. Note that the orientation of the car is implicit, as two-way streets are modelled as separate edges. On a high level, reachability at a correct orientation is a primary goal also in the Challenge.

**Accident avoidance.** Achieve a probability $v_2$ to never make an accident, that is, achieve $\Pr^{\pi,\sigma}_{\mathcal{G},\langle s_0,e_{init}\rangle}(\mathsf{G}\,\neg T_2) \geq v_2$. Note that a traffic rule violation, represented by the "violation" state is not considered an accident. This safety goal represents the other primary goal of the Challenge.

**Road quality.** Achieve a certain road quality $v_3$ over the duration of driving, i.e. achieve $\mathbb{E}^{\pi,\sigma}_{\mathcal{G},\langle s_0,e_{init}\rangle}[rew((r))] \geq v_3$. The road quality is determined according to the road type and length extracted from the map data. Hence, each edge $e$ is assigned a value $\mathrm{rval}(e)$, and the reward function $r$ is defined by $r(\langle e, sink\rangle) \stackrel{\mathrm{def}}{=} \mathrm{rval}(e) \cdot \mathsf{len}(e)$. In the Challenge, cars must be able to navigate over different road types, and select adequate roads.

### 5.4 Implementation

We now give the details of our prototype implementation, focusing on how to make the computation of strategies achieving CQs more efficient. Since the sets $Cnr(X^k_s)$ can be represented as convex polytopes, we use the Parma Polyhedra Library [1] to perform the value iteration operations.

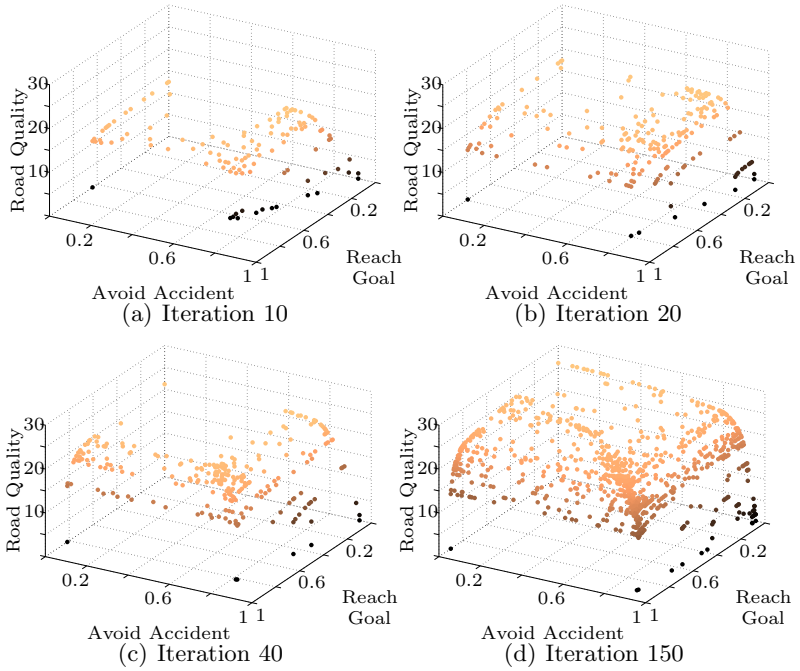



**Fig. 4.** Successive (under-)approximations of the Pareto frontier in $\langle s_0, e_{init} \rangle$ of $\mathcal{G}$ for Charlton-on-Otmoor, UK.

*Gauss-Seidel update.* Optionally, in-place updates can be used when computing $X^{k+1}$ from $X^k$. That is, when $X_s^{k+1}$ is computed from $X^k$, the result is stored in the same memory location as $X_s^k$. Subsequently, if $X_t^{k+1}$ is computed for $t \neq t$, and $s \in \Delta(t)$, then $X_s^{k+1}$ is used instead of $X_s^k$. Correctness of this method, also called Gauss-Seidel update, follows from the monotonicity of the functional $F$ from Theorem 1.

*Dynamic Accuracy Adaptation.* During value iteration, $|Cnr(X_s^k)|$ may increase exponentially with $k$. To mitigate this, for each step $k$, we fix a baseline accuracy $a_k \gg 1$, and round down each coordinate $i$ of each corner of $X_s^k$ to a multiple of $\frac{M_i}{a_k}$, where $M_i$ is of the same order of magnitude as the maximum reward in dimension $i$. The resulting polytopes are denoted by $\tilde{X}_s^k$. Note that we must round *down* in order to maintain safe under-approximations of the Pareto frontiers.[2]

Starting from $a_0$, we dynamically increase the accuracy $a_k$ by some factor $\lambda > 1$ after $N_k$ steps, while at the same time increasing the number $N_k$ of steps until the next increase by the same factor $\lambda$. In the long run, this yields an additive increase in the accuracy of $\frac{a_0(\lambda-1)}{N_0}$ per step. With this approach, we obtain under-approximations of the Pareto frontiers with a small number of points that are gradually refined by allowing more points in the polytopes.

[2] C.f. the induction hypothesis in the proof of Theorem 1, see [9].

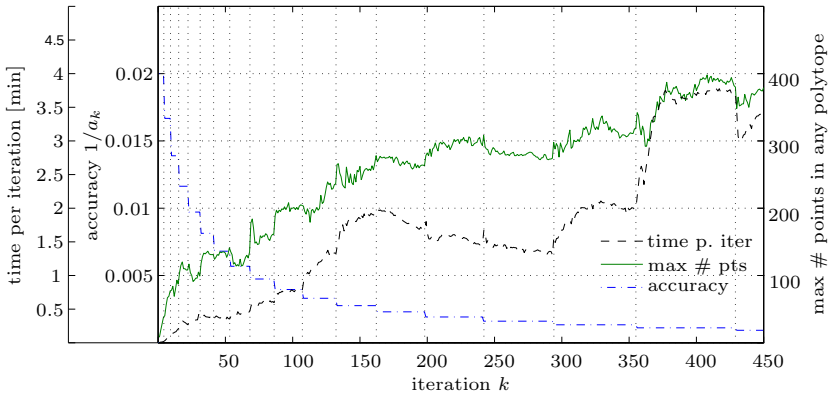



**Fig. 5.** Performance indicators for Charlton-on-Otmoor, cf. Figure 4. Note the changes in the number of points and time to complete an iteration as the accuracy changes.

Note that, while $X_s^k \subseteq F(X^k)(s)$, it is no longer true that $X_s^k \subseteq \tilde{X}'$, where $X' = F(X^k)(s)$. Therefore we use $\tilde{X}_s^{k+1} \cup \tilde{X}_s^k$ to preserve monotonicity.

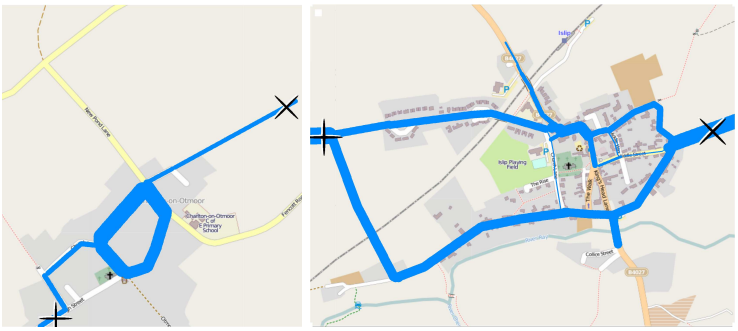
*Stopping criterion.* Given a rCQ $\varphi = (\boldsymbol{r}, \boldsymbol{v})$, a strategy which achieves $\boldsymbol{v}$ is sufficient, even if the Pareto frontier contains a point $\boldsymbol{w} > \boldsymbol{v}$. It is therefore possible to terminate value iteration prematurely after iteration $k$, and yet apply the strategy construction in Section 4 to achieve any points in the polytopes $X_s^k$.

### 5.5 Results

In this section we present the experimental results of the case study, which are computed using the CQ $\varphi$ from Section 5.3.

*Value Iteration.* We visualise the results of the value iteration, providing the intuition behind the trade-offs involved. In Figure 4 we show the polytopes computed for the initial state of the game for Charlton-on-Otmoor for several values of $k$. Rounding decreases the number of corner points, and Gauss-Seidel updates increase the convergence speed, albeit not the time per iteration. In Figure 5 we show performance indicators of the value iteration of Figure 4.

*Strategy Evaluation.* For $v = (0.7, 0.7, 6.0)$, we evaluate the constructed strategy $\pi$ for an adversary $\sigma$ that picks hazards uniformly at random, build the induced Markov chain $\mathcal{G}(\pi, \sigma, [\langle s_0, e_{init}\rangle \mapsto 1])$, and illustrate the resulting strategies for two villages in the UK in Figure 6. In Figure 6(b), one can observe that roads are picked that do not lead towards the goal. This is due to the strategy achieving a point on (or below) the Pareto frontier representing a trade-off between the three objectives, as opposed to maintaining a hard constraint of having to reach the



(a) Charlton-on-Otmoor: 43 edges in $G$, 501 states in $\mathcal{G}$. (b) Islip: 125 edges in $G$, 1527 states in $\mathcal{G}$.

**Fig. 6.** Resulting strategies for the target vector $(0.7, 0.7, 6.0)$. The start and the goal are shown by a plus $(+)$ and a cross $(\times)$ respectively. The thickness of the lines represents the expected proportion of trip time spent on the respective road by the car.

goal. Moreover, since maximising road quality is traded off against other goals, it may be suboptimal to take roads several times to improve the expectation while possibly incurring accidents and violations.

## 6 Conclusion

In this paper we have provided the first application of multi-objective stochastic two-player games. We have proposed algorithms for strategy synthesis and extended the approach to support important classes of LTL objectives. To evaluate the applicability of our techniques, we have developed a prototype implementation of the algorithms in the PRISM-games model checker and conducted a case study, synthesising and evaluating strategies for autonomous urban driving using real map data. There are many directions for future work, including extending the approach to support minimisation of the reward functions, application to assume-guarantee synthesis, and handling more complex multi-objective queries (e.g., combinations of conjunctions and disjunctions of objectives).

*Acknowledgments.* The authors thank Vojtěch Forejt, Mateusz Ujma and Klaus Dräger for helpful discussions and comments. The authors are partially supported by ERC Advanced Grant VERIWARE, the Institute for the Future of Computing at the Oxford Martin School, EPSRC grant EP/F001096, and the German Academic Exchange Service (DAAD).

## References

1. R. Bagnara, P. M. Hill, and E. Zaffanella. The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. *Sci. Comput. Program.*, 72(1-2):3–21, 2008.

2. T. Brázdil, V. Brozek, K. Chatterjee, V. Forejt, and A. Kucera. Two views on multiple mean-payoff objectives in Markov decision processes. In *LICS*, pages 33–42, 2011.
3. M. Campbell, M. Egerstedt, J. P. How, and R. M. Murray. Autonomous driving in urban environments: approaches, lessons and challenges. *Phil. Trans. R. Soc. A*, 368(1928):4649–4672, 2010.
4. K. Chatterjee, L. Doyen, T. A. Henzinger, and J. F. Raskin. Generalized mean-payoff and energy games. In *FSTTCS*, volume 8 of *LIPIcs*, pages 505–516, 2010.
5. K. Chatterjee, R. Majumdar, and T. A. Henzinger. Markov decision processes with multiple objectives. In *STACS*, volume 3884 of *LNCS*, pages 325–336, 2006.
6. K. Chatterjee, M. Randour, and J. F. Raskin. Strategy synthesis for multi-dimensional quantitative objectives. In *CONCUR*, volume 7454 of *LNCS*, pages 115–131, 2012.
7. T. Chen, V. Forejt, M. Kwiatkowska, D. Parker, and A. Simaitis. PRISM-games: A model checker for stochastic multi-player games. In *TACAS*, volume 7795 of *LNCS*, pages 185–191, 2013.
8. T. Chen, V. Forejt, M. Kwiatkowska, A. Simaitis, A. Trivedi, and M. Ummels. Playing stochastic games precisely. In *CONCUR*, volume 7454 of *LNCS*, pages 348–363, 2012.
9. T. Chen, V. Forejt, M. Kwiatkowska, A. Simaitis, and C. Wiltsche. On stochastic games with multiple objectives. In *MFCS*, 2013. (accepted).
10. C. Courcoubetis and M. Yannakakis. Markov decision processes and regular events. *IEEE Trans. Autom. Control*, 43(10):1399–1418, 1998.
11. DARPA. Urban Challenge, 2007. [Online; accessed 8-March-2013].
12. S. Dziembowski, M. Jurdzinski, and I. Walukiewicz. How much memory is needed to win infinite games? In *LICS*, pages 99–110, 1997.
13. K. Etessami, M. Z. Kwiatkowska, M. Y. Vardi, and M. Yannakakis. Multi-objective model checking of Markov decision processes. *LMCS*, 4(4), 2008.
14. U. Fahrenberg, L. Juhl, K. G. Larsen, and J. Srba. Energy games in multiweighted automata. In *ICTAC*, volume 6916 of *LNCS*, pages 95–115, 2011.
15. V. Forejt, M. Z. Kwiatkowska, G. Norman, D. Parker, and H. Qu. Quantitative multi-objective verification for probabilistic systems. In *TACAS*, volume 6605 of *LNCS*, pages 112–127, 2011.
16. OpenStreetMap, 2013. [Online; accessed 8-March-2013].
17. C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *J. Field Robot.*, 25(8):425–466, 2008.
18. Y. Velner, K. Chatterjee, L. Doyen, T. A. Henzinger, A. Rabinovich, and J. F. Raskin. The complexity of multi-mean-payoff and multi-energy games. *CoRR*, abs/1209.3234, 2012.
19. T. Wongpiromsarn and E. Frazzoli. Control of probabilistic systems under dynamic, partially known environments with temporal logic specifications. In *CDC*, pages 7644–7651, 2012.
20. T. Wongpiromsarn, U. Topcu, and Murray R. M. Receding horizon temporal logic planning. *IEEE Trans. Automat. Contr.*, 57(11):2817–2830, 2012.
21. T. Wongpiromsarn, A. Ulusoy, C. Belta, E. Frazzoli, and D. Rus. Incremental synthesis of control policies for heterogeneous multi-agent systems with linear temporal logic specification. In *ICRA*, 2013. (accepted).