Synthesis of Pareto-optimal Policies for Continuous-Time Markov Decision Processes

Naif Alasmari
Department of Computer Science
University of York, York, UK
email: nnma500@york.ac.uk

Radu Calinescu
Department of Computer Science
University of York, York, UK
email: radu.calinescu@york.ac.uk

***Abstract*—We present a work-in-progress method for the synthesis of continuous-time Markov decision process (CTMDP) policies—an important problem not handled by current probabilistic model checkers. The policies synthesised by this method correspond to configurations of software systems or software controllers of cyber-physical systems (CPS) that satisfy predefined nonfunctional constraints and are Pareto-optimal with respect to a set of optimisation objectives. We illustrate the effectiveness of our method by using it to synthesise optimal configurations for a client-server system, and optimal controllers for a driver-attention management CPS.**

***Index Terms*—continuous-time Markov decision process, probabilistic model checking, Pareto-optimal policy**

I. Introduction

Markov decision processes (MDPs) have long been used to support the synthesis of software and cyber-physical system configurations and discrete-event controllers that satisfy predefined dependability, performance and other nonfunctional requirements [1]–[3]. However, for many systems of practical importance, time appears explicitly in their nonfunctional requirements, and therefore using MDPs for their modelling and synthesis is not possible. For these systems, the modelling of the timing aspects is essential, and this can only be achieved through using continuous-time models.

Continuous-time Markov decision processes (CTMDPs) [4], [5] provide a powerful paradigm for modelling and synthesising stochastic systems in different field ranging from queuing systems [6], autonomous applications [7], power management [8] and inventory systems [9]. CTMDPs represent an extension of continuous-time Markov chains (CTMCs) that also permits the modelling of nondeterminism, where this nondeterminism can be managed by a policy [10]. Nevertheless, existing probabilistic model checkers such as PRISM [11] and Storm [12] do not support CTMDPs.

Our paper addresses this limitation of current tools by introducing a work-in-progress method for the synthesis of CTMDP policies. This method supports the synthesis of optimal configurations for software systems, and of software controllers for cyber-physical systems (CPS) whose timing properties and nondeterminism need to be taken into account. Our new method operates by encoding the CTMDP used in the synthesis as a *parametric* CTMC, so that the synthesis of policies for the original CTMDP is equivalent to synthesising the values for the parameters of this CTMC. As such, the parameteric CTMC can be employed to synthesise the required CTMDP policies either (i) manually, by using a probabilistic model checker; or (ii) in a fully automated way, by using the EvoChecker probabilistic model synthesis tool [13].

We carried out a preliminary evaluation of our new method by applying it to the synthesis of: (i) Pareto-optimal configurations for a queue whose requests are handled by a server with dual operating mode, and (ii) a software controller for a realistic human-in-the-loop self-adaptive system. The latter system comes from the autonomous driving domain, and was adopted from the existing work within the 'Safety of Shared Control in Autonomous Driving' (SafeSCAD) project at www.york.ac.uk/assuring-autonomy/projects/safe-scad.

II. Background

Similar to MDPs, each state of a continuous-time Markov decision processes has one or several associated *actions* such that the outgoing transitions from that state and their rates depend on the action *selected* in that state. We use the following formal definition adapted from [4], [5], [14].

Definition 1. *A CTMDP with countable number of states and actions is a four-tuple $M = (S, s_0, A, R)$, where:*

- *S is a countable set of states;*
- *$s_0 \in S$ represents the initial state;*
- *A is a finite set of actions;*
- *$R : S \times A \times S \to \mathbb{R}_{\geqslant 0}$ is a transition rate function such that, for any states $s_i, s_j \in S$ and any action $a \in A$, $R(s_i, a, s_j)$ specifies the rate of transition from state s_i to state s_j when action a is selected in state s_i.*

We have $R(s_i, a, s_j) = 0$ when $s_j = s_i$, and $\sum_{s_j \in S} R(s_i, a, s_j) = 0$ if action a is not available in state s_i. Finally, for any action $a \in A$ available in state s_i, the probability that the CTMDP leaves state s_i within $t > 0$ time units when action a is selected is given by $1 - e^{-t \cdot \sum_{s_k \in S} R(s_i, a, s_k)}$, and the probability that this transition is to state s_j is given by $R(s_i, a, s_j) / \sum_{s_k \in S} R(s_i, a, s_k)$.

To enable the analysis of additional properties, the states and transitions of a CTMDP can be annotated with *rewards*.

Definition 2. *A* reward structure *over a CTMDP with state set S is a pair of real-valued functions $r^X = (r_1, r_2)$, where $r_1 : S \to \mathbb{R}_{\geqslant 0}$ is a* state reward function *that determines the*

rate $r_1(s)$ at which the reward is acquired while the Markov model remains in state s; and $r_2 : S \times A \times S \to \mathbb{R}_{\geqslant 0}$ is a transition reward function *that describes the reward $r_2(s_i, s_j)$ obtained each time a transition occurs from state s_i to state s_j following the selection of action $a \in A$ in state s_i.*

The choice of which action from A to take in every state $s \in S$ of the CTMDP is assumed to be nondeterministic, and reasoning about the behaviour of CTMDPs involves the use of *policies*. A policy resolves the nondeterministic choices of an CTMDP by choosing the action taken in every state. In this paper, we consider *deterministic memoryless policies*, i.e., policies for which the same action is chosen each time when a CTMDP state is reached. We use the term "policy" to refer to this class of CTMDP policies in the rest of the paper.

Definition 3. *A (deterministic memoryless) policy of an CTMDP is a function $\sigma : S \to A$ that maps each CTMPD state $s \in S$ to an action from A that is available in state s.*

Note that each such policy σ maps the CTMDP over which it is defined to a standard continuous-time Markov chain $M_\sigma = (S, s_0, \mathbf{R})$ with same state set S and initial state s_0 as the original CTMDP, and with transition rate matrix $\mathbf{R}$ defined by $\mathbf{R}(s_i, s_j) = R(s_i, \sigma(s_i), s_j)$ for any states $s_i, s_j \in S$.

Finally, we use *continuous stochastic logic* (CSL) augmented with rewards [15] to express the requirements (including constraints and optimisation objectives) for the CTMDP policies to synthesise. Particularly relevant to our paper, these include reward-based requirements such as '$R^{dropped}_{=?}[C^{\leqslant T}] \leqslant MaxDropped$' (to express the requirement that not more than $MaxDropped$ requests are dropped by a server within T time units) and 'minimise $R^{length}_{=?}[I^{=T}]$' (to express the requirement that the instantaneous length of a server queue at time T should be minimised).

III. CTMDP Policy Synthesis Method

Our method for (deterministic memoryless) CTMDP policy synthesis comprises two steps. The input for the first step is a K-action CTMDP $M = (S, s_0, A, R)$ with $A = \{a_0, a_1, \ldots, a_{K-1}\}$. Using the notation $I_{s_i} = \{k \in \{0, 1, \ldots, K-1\} \mid \sum_{s_j \in S} R(s_i, a_k, s_j \neq 0\}$ to denote the set of indices of the actions available in state $s_i \in S$, this step builds a parametric CTMC (pCTMC) $M' = (S, s_0, \mathbf{R})$ whose transition rate between states $s_i \in S$ and $s_j \in S$ is given by

$$\mathbf{R}(s_i, s_j) = \sum_{k \in I_{s_i}} equal(x_{s_i}, k) R(s_i, a, s_j), \qquad (1)$$

where $x_{s_i} \in I_{s_i}$ is a parameter, and $equal(a, b) = 1$ if $a = b$ and zero otherwise. Fixing the value of each parameter x_{s_i} for the pCTMC M' reduces it to a (non-parametric) CTMC identical to the CTMC obtained for the policy of the original CTMDP that selects action $a_{x_{s_i}}$ for each state $s_i \in S$.

In the second step of our method, we search the parameter space $\times_{s_i \in S} I_{s_i}$ of the pCTMC from step 1 for combinations of parameter values $\{x_{s_i}\}_{s_i \in S}$ that reduce the parameteric CTMC to a non-parametric CTMC which satisfies a set of CSL-encoded requirements of interest. Two options are available for performing this search. First, the pCTMC can be encoded in the modelling language of the probabilistic model checking PRISM [11] for a manual analysis of the different combinations of parameter values using this model checker. This option is suitable when only a small number of such combinations are possible. Alternatively, the parametric CTMC can be encoded in the extended PRISM modelling language used by the probabilistic model synthesis tool EvoChecker [13], and the multiobjective genetic algorithm search engine provided by this tool can be used to perform an automated search for parameter value combinations that satisfy the requirements. This option can handle very large search spaces. We illustrate the use of both options in the next section.

IV. Preliminary Evaluation

To evaluate the effectiveness of the CTMDP policy synthesis approach in producing Pareto-optimal solutions for complex combinations of requirements,[1] we applied it within two case studies. The first case study is based on the simple queueing system from our running example. The second case study involves the synthesis of a controller for managing the attentiveness of drivers of vehicles with Level 3 automated driving systems. We present these cases studies next.

A. CTMDP policy synthesis for a queueing system

For this case study, we used the new method for the synthesis of CTMDP policies corresponding to the synthesis of optimal configurations for a simple queue. As shown in Figure 1, this queue consists of a single server that offers services incoming requests. This system has a state space $q = \{0, 1, \ldots, N\}$ where q denotes the number of requests that are waiting for service or being served, and $N > 0$ is the maximum number of requests that the queue can process. Each request arrives at a $\mu > 0$ rate. Assume the server has two states: ready ($s = 0$) when the server is free and ready to process a request from the queue, or busy ($s = 1$) when the server has just serviced a request and needs to perform a clean-up operation before servicing to another request. Each request can be processed by the server with one of two service rates: $\lambda_F > 0$ for fast mode or $\lambda > 0$ for standard mode. When the server is busy, the arriving requests join the queue and wait until the server becomes available. If the queue is full, the arriving request is dropped from the queue. Finally, we assume that the cost of serving a request using the standard and fast mode of operation is $c_1 > 0$ and $c_2 > c_1$, respectively.

Given this queue, we suppose that the mode of operation (i.e., standard or fast) used by its server for each queue size $q \in \{1, 2, \ldots, N\}$ needs to be determined such that the following constraints and optimisation objectives are satisfied:

1) the number of requests dropped within a period of time of length T should not exceed a given bound $MaxDropped$;

[1] i.e., sets of requirements comprising multiple constraints and multiple optimisation objectives

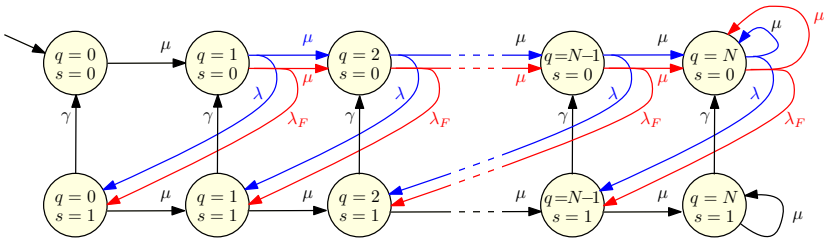

Fig 1. CTMDP model of a queue of size $N > 0$ with request arrival rate μ, and a server that can process each request using one of two modes of operation: a "standard" mode with service rate λ and a "fast" mode with service rate $\lambda_F > \lambda$; the server needs to perform a clean-up operation (with rate γ) after each serviced request. Note how two actions (corresponding to the blue and red state transitions) are available in each state when the queue contains $q > 0$ requests, and the server is ready to process a request (i.e., $s = 0$).

2) the total cost for serving requests over a period of time of length T should not exceed a given bound $MaxCost$;
3) the expected queue length at time T should be minimised;
4) the total costs for serving requests over a period of time of length T should be minimised,

where $T > 0$ is a predefined period of time. These constraints and optimisation criteria can be formalised using rewards-extended CSL as follows:

1) $R^{dropped}_{=?}[C^{\leqslant T}] \leqslant MaxDropped$;
2) $R^{cost}_{=?}[C^{\leqslant T}] \leqslant MaxCost$;
3) minimise $R^{length}_{=?}[I^{=T}]$;
4) minimise $R^{cost}_{=?}[C^{\leqslant T}]$.

Our evaluation considered a version of the queueing system with $N = 6$, $\mu = 1.6$, $\lambda = 1.8$, $\lambda_F = 4$, $\gamma = 20$, $c_1 = 1$ and $c_2 = 5$. The pCTMC model induced by the CTMDP queueing system for these parameter values is shown in Figure 2.

To find the Pareto-optimal policies that satisfy the objectives and constraints mentioned earlier, we manually ran PRISM experiments covering all possible policies for the CTMDP, i.e., all combinations of $(x_1, x_2, \ldots, x_6) \in \{0, 1\}^6$. The size of this search space is 2^6 since we have six parameters with two types of service rates (fast or standard). The Pareto front associated with the set of Pareto-optimal policies for the queuing system is shown in Figure 3.

B. CTMDP policy synthesis for driver attentiveness management controller

For the second case study, we applied our method to a CPS for managing the attentiveness of a driver in Level 3 autonomous driving vehicles. This CPS is adopted from our previous work [16] and manages the driver's attentiveness by using a Monitoring-Analysis-Planning-Execution (MAPE) control loop, that involves four steps:

- The monitoring step employs a number of sensors that are mounted on the car and the driver (as a wearable device) to gather data about them.
- The analysis step receives the data from the previous step, analyses them, forecasts the driver's reactions, and measures the quality of the driver's response.
- The planning step selects the speed of the car and the alerts to be activated by means of a discrete-event controller

```
ctmc

const int N = 6;
const double mu = 1.6;
const double lambda = 1.8;
const double lambda_fast = 4;
const double gamma = 20;

module queue
 q : [0..N] init 0;

 [request]  q<N -> mu:(q'=q+1);
 [dropped]  q=N -> mu:(q'=q);
 [serve]    q>0 -> 1:(q'=q-1);
endmodule

const int x1;
const int x2;
const int x3;
const int x4;
const int x5;
const int x6;

module server
 s : [0..1] init 0;

 [serve]    s=0 & q=1 & x1=0 -> lambda:(s'=1);
 [serve]    s=0 & q=1 & x1=1 -> lambda_fast:(s'=1);
 [serve]    s=0 & q=2 & x2=0 -> lambda:(s'=1);
 [serve]    s=0 & q=2 & x2=1 -> lambda_fast:(s'=1);
 [serve]    s=0 & q=3 & x3=0 -> lambda:(s'=1);
 [serve]    s=0 & q=3 & x3=1 -> lambda_fast:(s'=1);
 [serve]    s=0 & q=4 & x4=0 -> lambda:(s'=1);
 [serve]    s=0 & q=4 & x4=1 -> lambda_fast:(s'=1);
 [serve]    s=0 & q=5 & x5=0 -> lambda:(s'=1);
 [serve]    s=0 & q=5 & x5=1 -> lambda_fast:(s'=1);
 [serve]    s=0 & q=6 & x6=0 -> lambda:(s'=1);
 [serve]    s=0 & q=6 & x6=1 -> lambda_fast:(s'=1);
 [prepare]  s=1 -> gamma:(s'=0);
endmodule

rewards "dropped"
 [dropped] true : 1;
endrewards

rewards "length"
 true : q;
endrewards

rewards "cost"
 [serve] q=1 : (x1=0)?1:5;
 [serve] q=2 : (x2=0)?1:5;
 [serve] q=3 : (x3=0)?1:5;
 [serve] q=4 : (x4=0)?1:5;
 [serve] q=5 : (x5=0)?1:5;
 [serve] q=6 : (x6=0)?1:5;
endrewards
```

N: queue size
mu: request arrival rate
lambda: standard mode service rate
lambda_fast: fast mode service rate
gamma: the clean-up operation rate.

This module models the queue, which is full when q=N.

pCTMC parameters. Each parameter xi could have one of two values: 0 or 1.

This module models the server. It has two rates of transition from s=0 to s=1:
– fast (when x_i=1)
– standard (when x_i=0).

the 'dropped' reward assigns a reward of 1 to transitions that model a request being dropped (i.e., when q=N).

'length' reward measuring the queue size

"cost" reward assigning a reward of 1 for requests server in standard mode and a reward of 5 for requests served in fast mode

Fig 2. The PRISM-encoded parametric CTMC model for the queuing system

engaged when the driver's attentiveness changes or the driver takes to long to respond to previous alerts.
- The execution step performs the controller's actions from the previous step, e.g., it reduces the car speed when the driver is inattentive, or it operates the planned alerts via visual, acoustic and/or haptic actuators.

The self-adaptive system is modeled as a CTMDP and converted to a parametric CTMC by using our method. The

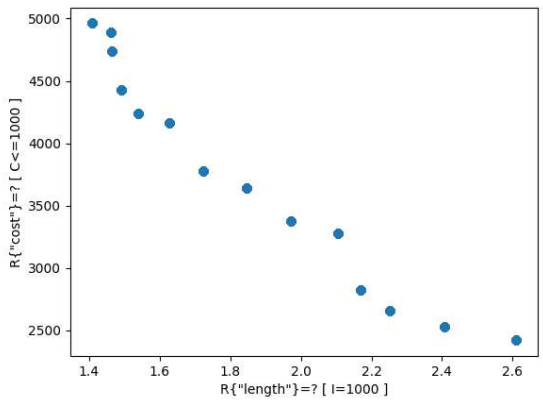

Fig 3. Pareto front the queuing system (results obtained running PRISM for 9.53 seconds on a MacBook Pro computer with 2.5 GHz Dual-Core Intel Core i5 processor and 8 GB of memory)

parametric CTMC is then employed to synthesise the controller component in the planning MAPE step. Due to space constraints, we cannot provide these models in the paper; the interested reader can find them in our previous work [16]. The purpose of the controller synthesis is to achieve optimal trade-offs among three measures: nuisance, progress and risk. We want to produce Pareto-optimal policies for the cumulative reward properties described below:

1) minimising the driver nuisance (due to alerts) over a four-hour journey: minimise $R^{nuisance}_{=?}[C^{\leqslant 4}]$;
2) maximising the progress with the trip over a four-hour journey: maximise $R^{progress}_{=?}[C^{\leqslant 4}]$;
3) minimising the overall risk incurred over a four-hour journey: minimise $R^{risk}_{=?}[C^{\leqslant 4}]$.

The controller decision space comprises 8^{16} combinations of actions due to the 16 options of controller configurations with 8 values in which represents the alerts and the car's speed level when the driver is not fully attentive. To synthesise a Pareto-optimal set of controllers for this system, we utilised the search-based software engineering tool EvoChecker [13]. The resulting Pareto front is shown in Figure 4.

V. Related Work

Many studies have been carried out to synthesise multi-objective optimisation policies for MDPs [1]–[3], and CT-MDP policies for simple optimisation requirements [17], [18]. However, none of these projects consider obtaining policies for combinations of multiple constraints and multi-objectives requirements for CTMDPs. Finally, our previous work in [16] uses a parametric CTMC to synthesise parameter values for the controller from the second case study in Section IV, but does not provide a generally applicable method for CTMDP policy synthesis like we do in this paper.

VI. Conclusion

We presented a work-in-progress CTMDP policy synthesis method that allows the generation of Pareto-optimal configurations and controllers for software systems and CPS,

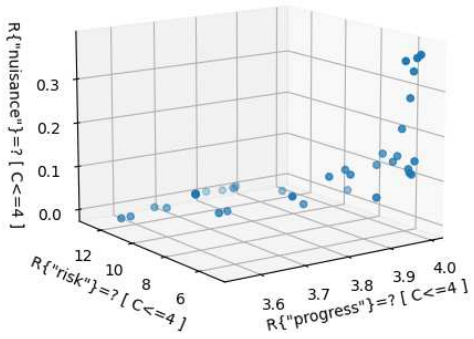

Fig 4. Pareto front for the Safe-SCAD controllers (results obtained running EvoChecker for 18.43 seconds on a Mac mini computer with 3.2 GHz 6-Core Intel Core i7 processor and 32 GB of memory)

respectively. Our preliminary evaluation in two case studies show that the methods can handle synthesis problems from different application domains. In future work, we plan to assess the quality of the Pareto-optimal solutions, fully automate the application of the new method, and evaluate our method in a broader range of case studies and scenarios.

Acknowledgements This project has received funding from the UKRI project EP/V026747/1 'Trustworthy Autonomous Systems Node in Resilience', and the Assuring Autonomy International Programme.

References

[1] F. Delgrange *et al.*, "Simple strategies in multi-objective MDPs," in *TACAS*, 2020, pp. 346–364.
[2] T. Brázdil *et al.*, "Two views on multiple mean-payoff objectives in Markov decision processes," in *LICS*, 2011, pp. 33–42.
[3] S. Gerasimou *et al.*, "Evolutionary-guided synthesis of verified Pareto-optimal MDP policies," in *ASE*, 2021, pp. 842–853.
[4] P. Buchholz, *et al.*, "Model checking algorithms for CTMDPs," in *CAV 2011*, 2011, pp. 225–242.
[5] Q. Qiu, *et al.*, "Dynamic power management based on continuous-time Markov decision processes," in *DAC'99*, 1999, pp. 555–561.
[6] L. I. Sennott, *Stochastic dynamic programming and the control of queueing systems*. John Wiley & Sons, 2009.
[7] Z. Mahboubi *et al.*, "Continous time autonomous air traffic control for non-towered airports," in *CDC*, 2015, pp. 3433–3438.
[8] Q. Qiu *et al.*, "Stochastic modeling of a power-managed system-construction and optimization," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 20, no. 10, pp. 1200–1217, 2001.
[9] Y. Huang *et al.*, "Continuous-time Markov decision processes with controlled observations," in *57th Annu. Allert. Conf. Commun. Control Comput. Allert.*, 2019, pp. 32–39.
[10] Y. Butkova *et al.*, "Optimal continuous time Markov decisions," in *ATVA'15*, 2015, pp. 166–182.
[11] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *CAV*, 2011, pp. 585–591.
[12] C. Dehnert *et al.*, "A Storm is coming: A modern probabilistic model checker," in *CAV*, 2017, pp. 592–600.
[13] S. Gerasimou *et al.*, "Synthesis of probabilistic models for quality-of-service software engineering," *ASE*, vol. 25, no. 4, pp. 785–831, 2018.
[14] F. Arnold *et al.*, "A tutorial on interactive Markov chains," in *ROCKS*, 2012, pp. 26–66.
[15] M. Kwiatkowska *et al.*, "Stochastic model checking," in *SFM-07:PE*, 2007, pp. 220–270.
[16] R. Calinescu *et al.*, "Maintaining driver attentiveness in shared-control autonomous driving," in *SEAMS*, 2021, pp. 90–96.
[17] M. N. Rabe and S. Schewe, "Finite optimal control for time-bounded reachability in CTMDPs and continuous-time Markov games," *Acta Informatica*, vol. 48, no. 5, pp. 291–315, 2011.
[18] H. Fu, "Maximal cost-bounded reachability probability on continuous-time Markov decision processes," in *FoSSaCS-14*, 2014, pp. 73–87.