مجله محاسبات نرم، جلد ۱۱، شماره ۲ (پاییز و زمستان ۱۴۰۱)، صفحه ۱۳۴ـ۱۴۸
https://doi.org/10.22052/scj.2023.243360.1029

مجله محاسبات نرم
SOFT COMPUTING JOURNAL
تارنمای مجله: scj.kashanu.ac.ir

University of Kashan

ارائه الگوریتمی جهت تسریع روش تکرار سیاست در راستی‌آزمایی فرآیندهای تصمیم مارکوف با استفاده از یادگیری ماشین ✧

محمدصادق محققی[1*]، استادیار

[1] گروه علوم کامپیوتر، دانشکده علوم ریاضی، دانشگاه ولیعصر رفسنجان، رفسنجان، ایران.

اطلاعات مقاله

تاریخچه مقاله:
دریافت ۲۵ آذر ماه ۱۴۰۱
پذیرش ۰۳ تیر ماه ۱۴۰۲

کلمات کلیدی:
راستی‌آزمایی صوری
وارسی مدل احتمالاتی
فرآیندهای تصمیم مارکوف
تکرار سیاست
دسترس‌پذیری بهینه
یادگیری ماشین

چکیده

فرآیندهای تصمیم مارکوف در هوش مصنوعی و راستی‌آزمایی رسمی برای مدل‌سازی سیستم‌های کامپیوتری که دارای رفتارهای تصادفی و غیرقطعی هستند، استفاده می‌شوند. دو دسته مهم از ویژگی‌هایی که در وارسی مدل احتمالاتی استفاده می‌شوند شامل احتمال بهینه رسیدن به حالت هدف و پاداش انباشته شده مورد انتظار هستند. تکرار مقدار و تکرار سیاست دو روش عددی تکراری شناخته شده برای تقریب مقادیر بهینه هستند. چالش اصلی این روش‌ها زمان اجرایی بالای آنها است. در این مقاله روشی جدید برای تسریع همگرایی به سیاست بهینه ارائه می‌شود که زمان اجرایی روش تکرار سیاست را کاهش می‌دهد. این روش بر پایه استفاده از یادگیری ماشین برای تخمین یک سیاست نزدیک به بهینه است. برای هر کلاس از مدل‌های فرآیند تصمیم مارکوف، تعدادی مدل کوچک را برای مرحله آموزش و ساخت دسته‌بند در نظر می‌گیریم. دسته‌بند ساخته شده در فرآیند یادگیری، برای پیش‌بینی کنش بهینه هر حالت فرآیند تصمیم مارکوف داده شده به کار می‌رود. این دسته‌بند همچنین برای پیشنهاد یک سیاست نزدیک به بهینه برای فرآیندهای تصمیم مارکوف بزرگ از همان دسته مدل‌ها، استفاده می‌شود تا زمان مصرفی کل را کاهش دهد. پیاده‌سازی روش ارائه شده در وارسی‌گر مدل PRISM نشان می‌دهد زمان اجرا به طور میانگین ۵۰ درصد کاهش می‌یابد.

۱. مقدمه

وارسی مدل احتمالاتی یک روش راستی‌آزمایی رسمی[1] برای تحلیل ویژگی‌های کمی و کیفی سیستم‌های کامپیوتری تصادفی است. در این روش، سیستم‌های تغییر وضعیت احتمالاتی[2] برای مدل‌سازی سیستم‌های کامپیوتری مورد بررسی، استفاده می‌شوند [۱]. زنجیره‌های مارکوف در مدل‌سازی سیستم‌هایی که رفتاری کاملا احتمالاتی دارند، بکار می‌روند. از سوی دیگر، فرآیندهای تصمیم مارکوف[3] (MDPs) برای مدل‌سازی سیستم‌هایی که جنبه‌هایی از رفتارهای احتمالاتی و غیرقطعی را توامان دارند، استفاده می‌شوند [۱]، [۲]. به عبارتی در وارسی مدل احتمالاتی از فرآیندهای تصمیم مارکوف به عنوان توسعه‌ای بر سیستم‌های

✧ نوع مقاله: پژوهشی
* نویسنده مسئول
پست(های) الکترونیک: mohagheghi@vru.ac.ir (محققی)

[1] Formal verification
[2] Probabilistic transition systems
[3] Markov decision processes

نحوه ارجاع به مقاله: محققی، محمدصادق، «ارائه الگوریتمی جهت تسریع روش تکرار سیاست در راستی‌آزمایی فرآیندهای تصمیم مارکوف با استفاده از یادگیری ماشین»، مجله محاسبات نرم، جلد ۱۱، شماره ۲، ص۱۳۴ـ۱۴۸، پاییز و زمستان ۱۴۰۱.

تغییر وضعیت که مفاهیم عدم قطعیت و احتمالات را نیز در بر دارند، استفاده می‌شود. عدم قطعیت غالبا زمانی استفاده می‌شود که در مدل‌سازی با مساله‌هایی مانند «در نظر گرفتن تصمیم‌گیری توسط سیستم» و یا «عدم وجود اطلاعات لازم در مورد بخش‌هایی از آن» مواجهیم. منطق درخت محاسبات احتمالاتی[1] (PCTL) یک منطق زمانی شناخته شده برای توصیف ویژگی‌های مورد نیاز در وارسی مدل احتمالاتی است. راستی‌آزمایی ویژگی‌های PCTL در فرآیندهای تصمیم مارکوف به طور معمول به محاسبه احتمالات دسترس‌پذیری بهینه (کمینه یا بیشینه) یا پاداش مورد انتظار بهینه نیاز دارد. در این موارد نیاز به انجام محاسبات عددی برای وارسی ویژگی‌های مورد نظر است. یکی از این روش‌های عددی فرآیند وارسی را به ساخت و حل یک مساله برنامه‌ریزی خطی (LP) تبدیل می‌کند. هرچند پیچیدگی زمان مصرفی این روش برحسب اندازه مدل چندجمله‌ای است اما کارایی آن برای حل بیشتر نمونه‌های مساله‌های واقعی پایین است [۱]- [۳]. تکرار ارزش[2] و تکرار سیاست[3] دو روش شناخته شده دیگر برای تخمین احتمالات دسترس‌پذیری یا پاداش مورد انتظار بهینه هستند که کارایی بیشتری نسبت به روش برنامه‌ریزی خطی دارند [۲]. ابزارهای PRISM و STORM مشهورترین ابزارهای وارسی‌گر مدل احتمالاتی هستند که از روش‌های تکرار ارزش و تکرار سیاست پشتیبانی می‌کنند [۴]، [۵].

نیاز بالا به زمان یا حافظه به طور معمول کارایی و امکان‌سنجی وارسی مدل را در همه انواع آن محدود می‌کند [۲]، [۳]. دلیل اصلی این محدودیت نیاز به ذخیره کل اطلاعات مدل در حافظه اصلی است. علاوه بر این، در مورد وارسی مدل احتمالی، محاسبات عددی بسیار زمانبر است. طیف وسیعی از رویکردها برای مقابله با این مشکلات پیشنهاد شده‌اند (برای مرور تعدادی از آنها به [۳]، [۵] و [۶] مراجعه کنید). در بیشتر موارد، این رویکردها عمومی هستند و می‌توانند در هر روش عددی تکراری اعمال شوند. در این مقاله، بر روی روش تکرار سیاست (PI) تمرکز شده و رویکردی برای بهبود عملکرد آن ارائه می‌شود. این رویکرد متکی بر استفاده از مفهوم سیاست برای حل مساله عدم قطعیت در مدل‌های MDP است. برای هر سیاست (به عنوان یک نگاشت از حالت‌ها به کنش‌ها[4])، یک زنجیره مارکوف زمان گسسته[5] (DTMC) القایی به نام DTMC خارج قسمت تولید می‌شود. سپس از یک روش تکراری برای محاسبه (یا تقریب) احتمال دسترس‌پذیری به حالت‌های هدف استفاده می‌شود [۷]. سیاست‌ها بر اساس رویکرد حریصانه بهبود می‌یابند که در آن بهترین کنش هر حالت انتخاب می‌شود. دلیل اصلی در نظر گرفتن روش تکرار سیاست در این مقاله این است که نیازی به داشتن تمام اطلاعات یک مدل در هر تکرار نیست و در مقابل، با در نظر گرفتن DTMCهای القایی مقادیر حالت‌ها به‌روز می‌شود. از این رو تعداد کل محاسبات و مدت زمان اجرا در هر تکرار کاهش می‌یابد.

یک عیب روش تکرار سیاست در مقایسه با سایر روش‌های عددی تکراری این است که تعداد کل تکرارهای آن به دقت سیاست‌های محاسبه شده بستگی دارد [۲]، یعنی هر چقدر که سیاست‌های انتخابی به سیاست بهینه نزدیک‌تر باشند و تعداد بیشتری از کنش‌های بهینه را انتخاب کنند، تعداد کل تکرارها کمتر می‌شود. به طور کلی، هیچ رویکرد ساده‌ای برای تقریب یک سیاست اولیه مناسب برای مدل MDP داده شده، وجود ندارد [۲]، [۶]. روش تکرار سیاست استاندارد یک سیاست دلخواه (یا تصادفی) را به عنوان سیاست اولیه انتخاب می‌کند و سیاست‌ها را تا رسیدن به یک مورد ثابت بهبود می‌دهد. برای هر DTMC القایی، تکرارها تا زمان برآورده‌سازی یک معیار توقف از پیش تعریف شده، ادامه می‌یابند. در تکرار سیاست اصلاح شده[6] تعداد محدودی از تکرارها برای بروزرسانی مقادیر هر DTMC القایی اعمال می‌شود.

به عنوان نوآوری اصلی این مقاله برای بهبود عملکرد تکرار سیاست، روشی با استفاده از یادگیری ماشین و درخت تصمیم به منظور تقریب یک سیاست مناسب برای مدل MDP داده شده، ارائه می‌شود. این روش مساله یافتن سیاست بهینه را به یک

[1] Probabilistic computation tree logic
[2] Value iteration
[3] Policy iteration
[4] Actions
[5] Discrete-time Markov chain
[6] Modified policy iteration

مساله دسته‌بندی[1] تبدیل می‌کند که در آن به هر حالت یک کنش از دسته مربوطه نسبت داده می‌شود. برای مرحله آموزش از چندین نمونه کوچکتر از مدل MDP داده شده، استفاده می‌شود. رویکرد مورد نظر، محاسبه سیاست بهینه برای هر یک از این مدل‌های کوچک برای آموزش دسته‌بند انتخاب شده (در این مقاله درخت تصمیم) است. در این رویکرد، حالت‌های MDP داده شده به عنوان ورودی در نظر گرفته می‌شوند. به ازای هر حالت، دسته‌بند (درخت تصمیم) ساخته شده یکی از کنش‌های MDP را به عنوان کنش بهینه پیشنهاد می‌دهد.

روش پیشنهادی این مقاله به عنوان پیش‌پردازش استفاده می‌شود. این روش برای یک مدل MDP بزرگ و ویژگی PCTL داده شده یک سیاست اولیه مناسب را بدون هیچ تکرار اضافی فراهم می‌کند. استفاده از سیاست اولیه مناسب (در مقابل سیاست تصادفی) می‌تواند به کاهش تعداد تکرارهای لازم تا همگرایی به جواب بهینه شود. در ضمن با داشتن یک سیاست اولیه مناسب، این امکان به وجود می‌آید که از برخی تکنیک‌های شناخته شده برای DTMCها جهت کاهش فضای حالات و افزایش کارایی استفاده کرد. به عنوان مثال می‌توان به روش شبیه‌سازی دوگانه[2] [۸] و ترتیب توپولوژیکی برای DTMCها [۹]، [۱۰]، اشاره کرد. در روش پیشنهادی، امیدواریم بیشترین تعداد تکرار را برای اولین DTMC داشته باشیم و آن را به مدل معادل کوچکتر تقلیل دهیم تا تعداد کل محاسبات کاهش یابد. این در حالی است که در روش تکرار سیاست استاندارد یا اصلاح شده، به طور معمول سیاست آغازین از دقت پایینی برخوردار است و استفاده از روش‌های کاهشی برای DTMCهای القایی دارای سربار زمان مصرفی بالایی خواهد بود.

روش پیشنهادی مقاله در وارسی‌گر مدل PRISM[3] پیاده‌سازی شده و بر روی تعدادی از مثال‌های استاندارد مورد محک قرار گرفته است. نتایج تجربی از بهبود تقریبی تا بیش از ۹۰ درصد در کاهش تعداد تکرارها و زمان اجراها حکایت دارد.

[1] Classification
[2] Bisimulation
[3] Probabilistic Symbolic model checker

۲. مروری بر کارهای مرتبط

یادگیری ماشین یکی از روش‌های رایج هوش مصنوعی برای حل کارای بسیاری از مساله‌های حوزه‌های گوناگون علوم است [۱۱]، [۱۲]. استفاده از یادگیری ماشین برای بهبود کارایی وارسی مدل احتمالاتی در سالیان اخیر مورد توجه تعدادی از محققان قرار گرفته است. استفاده از الگوریتم‌های مبتنی بر یادگیری با تمرکز بر الگوریتم‌های برنامه‌ریزی پویای بلادرنگ[4] در مرجع [۹] بررسی شده است. در این روش تمرکز بر حالت‌های مهمتر و بروزرسانی مقادیر آنها، صرف نظر از حالت‌های کم اهمیت است. یکی از ایرادات این کار، وابستگی آن به ساختار گرافی مدل MDP است به شکلی که ممکن است برای دسته‌ای از مدل‌ها سربار محاسبات بیشتر از روش‌های استاندارد شود. در مرجع [۶]، از روش‌های یادگیری عمیق با تاکید بر ایده‌های یادگیری تقویتی استفاده شده تا دسته‌ای از مساله‌های زمانبندی که با MDPها مدل شده‌اند به شکل موثر حل شوند. یادگیری ماشین برای تعریف MDPها در محیط‌های جدید در مرجع [۱۳] استفاده شده است. در این کار احتمالات تغییر وضعیت طی یک فرآیند یادگیری تخمین زده می‌شوند. تکنیک‌های سریع‌تر جبر خطی عددی در مرجع [۱۴] استفاده شده است تا زمان محاسبات تکراری و عددی کاهش یابد. در پژوهش [۱۵] از درونیابی برای تخمین سیاست بهینه استفاده شده است. هرچند ایده اصلی مقاله جاری که در بخش‌های بعد توضیح داده می‌شود نیز به تخمین سیاست بهینه می‌پردازد، اما تفاوت اصلی آن با مرجع [۱۵] در استفاده از دسته‌بندی و درخت تصمیم است. در ضمن روش ارائه شده در مرجع [۱۵] محدود به یک مطالعه موردی[5] است در حالی که روش ما عمومیت دارد. در مرجع [۱۶]، از یادگیری ماشین برای تخمین مجموعه‌های مورد نیاز در وارسی مدل کیفی برای MDP استفاده شده است.علاوه بر این، پژوهش‌هایی در دو دهه گذشته برای کاهش زمان مصرفی روش‌های تکراری عددی مورد استفاده در وارسی مدل احتمالاتی، ارائه شده‌اند. برخی از این روش‌ها بر روی MDPهایی با ساختار خلوت تمرکز دارند

[4] Real-time dynamic programming
[5] Case study

[۴]، [۷]، [۱۰]، [۱۷] و برخی دیگر برای مدل‌هایی با ساختار غیرخلوت توسعه یافته‌اند [۱۴]، [۱۸]، [۱۹]. هر چند این روش‌های بهبود یافته قادرند با کاهش زمان مصرفی محاسبات، کارایی روش‌های وارسی مدل احتمالاتی را بهبود دهند اما کابرد آنها به مدل‌های با اندازه استاندارد که حافظه کامپیوتر قادر به نگهداری اطلاعات آنها است، محدود می‌شوند. از طرف دیگر، تمرکز اصلی پژوهش ما بر روی توسعه روش‌های آماری وارسی مدل برای نمونه‌های بزرگی است که روش‌های یاد شده قادر به اعمال نیستند. همچنین کاربردهای متنوعی از DTMCها و MDPها در مراجع [۲۰]- [۲۳] و [۲۴] ارائه شده است.

۳. تعریف‌های مربوط به مساله

در این بخش به مرور تعریف‌های مرتبط با فرآیندهای تصمیم مارکوف، وارسی مدل احتمالاتی و یادگیری ماشین پرداخته می‌شود. توضیحات نظری بیشتر در مراجع [۱]- [۳] و [۱۳] ارائه شده‌اند. در این مقاله از نماد $Dist(S)$ برای مجموعه همه توزیع احتمالات گسسته روی فضای S داده شده، استفاده می‌شود. هر توزیع احتمال تابعی به صورت $p: S \to [0,1]$ است که در آن $\sum_{s\in S} p(s) = 1$ است.

تعریف ۱: یک فرآیند تصمیم مارکوف به صورت چندتایی $M = (S, s_0, Act, \delta, G)$ تعریف می‌شود که S مجموعه متناهی از حالت‌های مدل، $s_0 \in S$ حالت آغازین، Act مجموعه‌ای متناهی از کنش‌ها است به شکلی که برای هر حالت $s \in S$ یک یا چند کنش از Act به عنوان کنش‌های مجاز شناخته می‌شوند و با $Act(s)$ نشان داده می‌شوند، $\delta: S \times Act \to Dist(S)$ یک تابع تغییر وضعیت است و $G \subset S$ یک مجموعه غیرتهی هدف است. منظور از $\delta(s, \alpha, s')$ احتمال رفتن از حالت s به حالت بعدی s' با استفاده از کنش α است. هر تغییر وضعیت یک سه تایی به صورت (s, α, s') است اگر $\delta(s, \alpha, s') > 0$ برقرار باشد. اندازه M که با $|M|$ نشان داده می‌شود به شکل مجموع تعداد حالت‌ها و تغییر وضعیت‌های مدل تعریف می‌شود.

فرآیندهای تصمیم مارکوف به طور گسترده در ریاضیات، علوم کامپیوتر، اقتصاد و سایر و علوم برای مدل‌سازی سیستم‌هایی که رفتارهای غیرقطعی و تصادفی دارند به کار می‌روند [۱۰]. در وارسی مدل، فرآیندهای تصمیم مارکوف به عنوان سیستم‌های تغییر وضعیت با توضیع‌های احتمالاتی به کار می‌روند. برای هر حالت $s \in S$ و کنش مجاز $\alpha \in Act(s)$ از $Post(s, \alpha)$ برای مجموعه حالت‌های بعد از $s \in S$ با کنش α استفاده می‌شود (رابطه (۱)). همچنین از $Post(s)$ برای مجموعه حالت‌های بلافاصله پس از s استفاده می‌شود (رابطه (۲)).

$$Post(s, \alpha) = \{s' \in S | \delta(s, \alpha, s') > 0\} \quad (۱)$$

$$Post(s) = \{\cup_{\alpha\in Act(s)} | Post(s, \alpha)\} \quad (۲)$$

یک زنجیره مارکوف زمان-گسسته یک فرآیند تصمیم مارکوف است که در آن به ازای هر حالت دقیقا یک کنش مجاز تعریف می‌شود. یک مسیر در M یک اجرای ممکن از سیستم مدل شده است که به شکل دنباله‌ای ناتهی (متناهی یا نامتناهی) از حالت‌ها و کنش‌ها به شکل $\pi = s_0 \xrightarrow{\alpha_0} s_0 \xrightarrow{\alpha_0} \dots$ تعریف می‌شود که برای مقدار $i \geq 0$ داریم $\alpha_i \in Act(s_i)$ و $s_{i+1} \in Post(s_i, \alpha_i)$. از $Path_s$ برای مجموعه همه مسیرهای نامتناهی M که از حالت $s \in S$ شروع می‌شوند، استفاده می‌کنیم. از $FPath_s$ نیز برای زیرمجموعه مسیرهای متناهی آن استفاده می‌کنیم. در ضمن از $\pi(i)$ برای نمایش حالت $i+1$ ام از مسیر π استفاده می‌شود، یعنی $\pi(i) = s_i$ است.

مثال ۱: برای درک بهتر نقش فرآیندهای تصمیم مارکوف در وارسی مدل احتمالات رباتی را در نظر بگیرید که در یک محیط شامل سه اتاق عمل می‌کند. ربات دارای سلول‌های خورشیدی است که مقدار محدودی شارژ را فراهم می‌کنند. وظیفه ربات جستجو و پیدا کردن یک شی مشخص در یکی از اتاق‌ها است. فرض کنید ربات از قبل نمی‌داند شی در کدام اتاق قرار دارد و احتمال وجود آن در هر اتاق یک سوم است. ربات در هر یک از اتاق‌ها می‌تواند در یکی از دو وضعیت آهسته یا سریع کار کند. در وضعیت سریع احتمال یافتن شی بیشتر خواهد بود اما احتمال اتمام شارژ نیز بیشتر می‌شود. مدت زمان حضور در هر اتاق یک ساعت است. در صورتی که قبل از اتمام جستجوی یک اتاق شارژ ربات تمام شود، ماموریت شکست خورده گزارش می‌شود.

از طرفی در صورتی که قبل از اتمام شارژ، یک ساعت جستجو به اتمام برسد، ربات می‌تواند به اتاق بعدی گام بگذارد. همچنین به دلیل وجود برخی شرایط محیطی در صورت عدم یافتن شی مورد نظر احتمال بازگشت به خانه قبل نیز وجود دارد. برای مدل‌سازی رفتار این ربات یک فرآیند تصمیم مراکوف در نظر گرفته شده که اطلاعات هر یک از تغییر وضعیت‌های ممکن آن در جدول (۱) ارائه شده است. این فرآیند شامل ۵ حالت است که حالت‌های ۱ تا ۳ نشان‌دهنده قرار داشتن در اتاق‌های سه‌گانه است. حالت ۴ نشان‌دهنده اتمام شارژ و در نتیجه شکست جستجو و حالت ۵ نشان‌دهنده یافتن شی مورد نظر است. در وارسی مدل این ربات علاقه‌مند به یافتن بیشترین احتمال رسیدن به حالت هدف (یافتن شی مورد جستجو) هستیم. در این مثال حالت اول به عنوان شروع در نظر گرفته می‌شود. مقادیر احتمالات براساس شرایط محیطی تخمین زده شده است.

جدول(۱): نمایش اطلاعات تغییر وضعیت فرآیند تصمیم مارکوف مثال ۱

مبدا	مقصد	کنش	احتمال
۱	۴	s	۰/۰۳
۱	۵	s	۰/۰۷
۱	۲	s	۰/۹
۱	۴	f	۰/۱۵
۱	۵	f	۰/۱۵
۱	۲	f	۰/۷
۲	۴	s	۰/۰۸
۲	۵	s	۰/۰۲
۲	۳	s	۰/۸
۲	۱	s	۰/۱
۲	۴	f	۰/۰۹
۲	۵	f	۰/۰۶
۲	۳	f	۰/۷
۲	۱	f	۰/۱۵
۳	۴	s	۰/۰۶
۳	۵	s	۰/۰۴
۳	۳	s	۰/۷
۳	۲	s	۰/۲
۳	۴	f	۰/۰۷
۳	۵	f	۰/۰۸
۳	۳	f	۰/۶
۳	۲	f	۰/۲۵
۴	۴	d	۱
۵	۵	d	۱

۳,۱. سیاست

مفهوم سیاست یک سازوکار استاندارد برای حل انتخاب‌های غیرقطعی در مدل‌های MDP است که با استفاده از آن می‌توان به تحلیل رفتارهای احتمالاتی این مدل‌ها پرداخت. در این مقاله تنها سیاست‌های قطعی و بی‌حافظه[1] در نظر گرفته می‌شوند که برای محاسبه ویژگی‌های ارائه شده در منطق درخت محاسبات احتمالاتی (PCTL)، که در ادامه توضیح داده می‌شوند، کافی هستند [۲]، [۹].

تعریف ۲: یک سیاست قطعی برای یک فرآیند تصمیم مارکوف مثل M به صورت تابع $\sigma: FPath_s \to Act$ تعریف می‌شود که برای هر مسیر متناهی به شکل $\pi = s_0 \xrightarrow{\alpha_0} s_0 \xrightarrow{\alpha_0} \dots \xrightarrow{\alpha_{i-1}} s_i$ یک کنش مجاز $\alpha_i \in Act(s_i)$ را انتخاب می‌کند. یک سیاست σ بی‌حافظه نامیده می‌شود اگر تنها به آخرین حالت مسیر وابسته باشد. از Adv_M برای مجموعه همه سیاست‌های قطعی M استفاده می‌شود.

برای سادگی در ادامه کار برای کنش‌های هر حالت مثل s تنها از ترتیب آن کنش استفاده می‌شود. برای هر حالت $s \in S$ قرارداد می‌کنیم $\sigma(s) = 0$ اگر سیاست σ اولین کنش M را انتخاب کند، $\sigma(s) = 1$ اگر سیاست σ دومین کنش M را انتخاب کند و به همین صورت برای سایر کنش‌ها نگاشت متناسب تعریف می‌شود. برای فرآیند تصمیم مارکوف مثال ۱، یک سیاست ممکن است به حالت‌های ۱ و ۲ کنش s (معادل با حرکت آهسته) و برای حالت سوم کنش f را انتخاب کند.

برای هر سیاست σ یک زنجیره مارکوف زمان گسسته متناظر (که به آن DTMC خارج قسمت می‌گویند)، قابل تعریف است که می‌توان از آن برای تحلیل رفتار سیستم تحت σ استفاده کرد. محاسبه احتمال دسترس‌پذیری بیهنه برای یک فرآیند تصمیم مارکوف داده شده به مساله محاسبه یک سیاست بهینه که احتمال رسیدن به یکی از حالت‌های هدف با شروع از s_0 را کمینه یا بیشینه می‌کند، کاهش می‌یابد [۱]، [۷].

تعریف ۳: برای فرآیند تصمیم مارکوف داده شده M و سیاست

[1] Memory-less and deterministic

معین و بی‌حافظه σ (نه لزوما بهینه)، DTMC خارج قسمت به صورت $M^{\sigma}=(S,s_0,P)$ تعریف می‌شود که در آن S و s_0 همانند M تعریف می‌شود و $P:S\times S\to[0,1]$ تابع تغییر وضعیت متناظر بوده و به صورت $P(s,s')=\delta(s,\sigma(s),s')$ تعریف می‌شود [۲]. گاه به P ماتریس تغییر وضعیت نیز گفته می‌شود. از $Path_s^{\sigma}$ برای مجموعه همه مسیرهای متناهی M^{σ} با شروع از حالت s استفاده می‌شود.

۳،۲. احتمالات دسترس‌پذیری

در فرآیند وارسی مدل از انواع منطق زمانی[1] برای توصیف ویژگی‌های مورد نظر، استفاده می‌شود. توسعه‌های احتمالاتی منطق زمانی مانند PCTL و LTL احتمالی برای توصیف خصوصیاتی استفاده می‌شوند که لازم است در برابر ساختارهای احتمالی مورد بررسی قرار گیرند [۱]، [۲]، [۱۳]. یک کلاس مهم از خصوصیات PCTL برای فرآیندهای تصمیم مارکوف مجموعه احتمالات دسترس‌پذیری بهینه[2] است. به عنوان مثال، حداقل یا حداکثر احتمال رسیدن به یکی از حالت‌های G در هنگام شروع از حالت $s\in S$ به صورت زیر تعریف می‌شود:

$$P_s^{\min}(G)=\inf_{\sigma\in Adv_M}p_s^{\sigma}(G) \qquad (۳)$$

$$P_s^{\max}(G)=\sup_{\sigma\in Adv_M}p_s^{\sigma}(G) \qquad (۴)$$

که در رابطه‌های فوق $p_s^{\sigma}(G)$ به عنوان فضای احتمال روی $Path_s^{\sigma}$ و به صورت رابطه زیر تعریف می‌شود:

$$p_s^{\sigma}(G)=\mathrm{Prob}_s^{\sigma}(\{\pi\in Path_s^{\sigma}\mid\exists i.\,\pi(i)\in G\}) \qquad (۵)$$

برای سادگی در ادامه این مقاله، احتمالات دسترس‌پذیری بیشینه را در نظر می‌گیریم. احتمالات دسترس‌پذیری کمینه به شکل مشابه محاسبه می‌شوند. محاسبه احتمالات دسترس‌پذیری بیشینه به محاسبه سیاست بهینه σ^* کاهش می‌یابد که معادله موجود در رابطه (۶) (معروف به معادله *بلمن*) را ارضا کند:

$$p_s^{\sigma^*}(G)=\begin{cases}1 & if\ s\in G\\ \max\limits_{\alpha\in Act(s)}\Big(\sum\limits_{s'\in Post(s,\alpha)}P(s,\alpha,s').p_{s'}^{\sigma^*}(G)\Big) & if\ s\notin G\end{cases} \qquad (۶)$$

به شکل مشابه سیاست بهینه، سیاستی است که بهترین کنش‌ها را به هر حالت نسبت دهد (رابطه (۷)).

$$\sigma^*(s)=\mathrm{argmax}_{\alpha\in Act(s)}\Big(\sum_{s'\in Post(s,\alpha)}P(s,\alpha,s').p_{s'}^{\sigma^*}(G)\Big) \qquad (۷)$$

بیشتر وارسی‌گرهای مدل احتمالاتی مانند PRISM و STORM از روش‌های عددی تکراری مانند تکرار ارزش و تکرار سیاست برای حل معادله‌های بلمن استفاده می‌کنند [۴]، [۵]. برای ربات مثال ۱، احتمال بیشینه رسیدن به حالت هدف پس از اعمال روش تکرار ارزش در PRISM برابر با ۰/۵۱۳۱ و احتمال کمینه رسیدن به حالت هدف برابر با ۰/۳۸۹۴ به دست آمده است، یعنی در بهترین عملکرد ربات (با انتخاب بهترین سیاست) با احتمال ۰/۵۱۳۱ قبل از اتمام شارژ قادر خواهد بود شی مورد نظر را پیدا کند. در ادامه این بخش، به مرور روش تکرار سیاست پرداخته می‌شود. جزئیات بیشتر در مورد این روش‌ها و شرایط همگرایی آنها در مراجع [۲] و [۷] ارائه شده است.

۳،۳. تکرار سیاست

ایده روش تکرار سیاست تولید دنباله‌ای از سیاست‌های بی‌حافظه σ_i و ساخت DTMC خارج‌قسمت برای هر یک از این سیاست‌ها است. شمای کلی این روش در الگوریتم (۱) ارائه شده است. در این الگوریتم از یک بردار $\bar{x}$ برای مقادیر استفاده می‌شود به شکلی که برای هر $s\in S$ از x_s برای ذخیره مقادیر تخمین زده شده برای $p_s^{\sigma}(G)$ استفاده می‌شود. پس از مقدارگذاری اولیه $\bar{x}$ در خط ۲ و انتخاب سیاست اولیه σ_1 این الگوریتم از یک روش تکراری استاندارد (مانند گاوس–سایدل [۲]) برای تقریب احتمال دسترس‌پذیری هر یک از حالت‌ها در DTMC متناظر استفاده می‌کند.

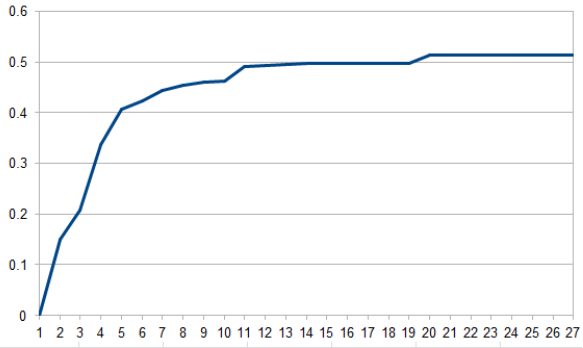
پس از برآورده ساختن شرط توقف محاسبات برای هر یک از DTMCهای متناظر با سیاست‌های σ_i، الگوریتم از یک روش حریصانه برای محاسبه یک سیاست بهبود یافته σ_{i+1} در خط‌های ۸ تا ۱۰ استفاده کرده و محاسبات را در چرخه دیگری از تکرارها ادامه می‌دهد.

[1] Temporal logic
[2] Optimal reachability probabilities

الگوریتم (۱): شبه کد روش تکرار سیاست

Algorithm 1. Policy Iteration for $p_s^{max}(G)$
Input: An MDP $M = (S, s_0, Act, \delta, G)$
Output: $p_s^{max}(G)$ for all $s \in S$
for all $s \in S$ do
  $x_s = \begin{cases} 1 & if\ s \in G \\ 0 & otherwise \end{cases}$
endfor
$i = 1$
Select an initial policy σ_1
do
  Compute $x_s = p_s^{\sigma}(G)$ for all $s \in S$
  foreach $s \in S$ do
    $\sigma_{i+1}(s) = \arg\max_{\alpha \in Act(s)} \sum_{s' \in S} \delta(s, \alpha, s').x_{s'}(G)$
  endfor
  $i = i + 1$
while $\sigma_{i+1} \neq \sigma_i$
return $(x_s)_{s \in S}$

تکرار سیاست زمانی به پایان می‌رسد که دو سیاست متوالی یکسان باشند و شرط $\sigma_i = \sigma_{i+1}$ برقرار باشد. سیاست پایانی را می‌توان به عنوان تقریبی خوب از سیاست بهینه در نظر گرفت. البته ممکن است سیاست پایانی پس از اتمام محاسبات بهینه نباشد، اما به طور معمول چنین وضعیتی برقرار است [۲]. به عنوان یک مثال عددی از اعمال روش تکرار سیاست برای تعیین بیشینه احتمال رسیدن به حالت هدف برای ربات مثال ۱، مراحل اجرا در شکل (۱) نشان داده شده است.



شکل (۱): مقادیر عددی احتمال بیشینه رسیدن به حالت هدف

در این شکل محور افقی، شماره تکرار و محور عمودی، مقدار محاسبه شده در آن تکرار را نمایش می‌دهد. در واقع این نمودار چگونگی همگرا شدن به مقدار ۰/۵۱۳۱ را در تکرارهای مختلف نشان می‌دهد. برای این مثال، روش تکرار سیاست دو مرتبه به اصلاح تکرار نیاز دارد. به ازای سیاست اولیه بعد از ۱۱ تکرار و به ازای سیاست دوم بعد از ۹ تکرار (در تکرار بیستم) اصلاح سیاست انجام می‌شود. درنهایت چون سیاست سوم سیاست بهینه است تکرار سیاست در تکرار بیست و هفتم به مقدار مورد نظر همگرا شده و خاتمه می یابد.

۳،۴. تکرار سیاست اصلاح شده

یک چالش تکرار سیاست، همگرایی دیرهنگام آن به سیاست بهینه برای تعدادی از مدل‌های MDP است. نسخه استاندارد این روش از یک خط سیاست دلخواه (تصادفی) شروع می‌شود و ممکن است برای همگرایی به روش‌های بهینه به چندین اصلاح سیاست نیاز داشته باشد. نسخه اصلاح شده تکرار سیاست (که در ادامه مقاله، MPI نامیده می‌شود) برای بروزرسانی هر سیاست پس از حداکثر تعداد مشخصی تکرار (به‌عنوان مثال ۱۰۰ تکرار) کار می‌کند. درنتیجه تعداد محدودی از تکرارها را برای DTMC خارج قسمت مربوطه اعمال می‌کند تا مقادیر حالت‌ها را بروز کند. یک مزیت MPI این است که از تکرارهای زیاد محاسبه برای DTMCهای اولیه جلوگیری می‌کند. با این حال، MPI ممکن است سعی کند سیاست‌ها را حتی در موارد بهینه نیز بهبود بخشد. در این حالت، در مقایسه با تکرار سیاست استاندارد، ممکن است تعداد کلی اصلاحات سیاست افزایش یابد. در بخش ۴ و به عنوان نوآوری اصلی این مقاله ما روشی را پیشنهاد می‌کنیم که تعداد کلی تکرار روش MPI را کاهش دهد.

۳،۵. یادگیری ماشین

یادگیری ماشین روشی است که در آن از مجموعه‌ای از نمونه‌های اولیه برای آموزش استفاده شده و هدف آن است که با روش‌های مبتنی بر ریاضیات و احتمالات یک تعمیم از آن برای نمونه‌های جدید به دست آید تا امکان پیش‌بینی در مورد هر نمونه جدید ممکن فراهم آید [۶]. در دسته‌بندی با نظارت[1] چندین ویژگی برای نمونه‌ها در نظر گرفته شده و از یک تابع برچسب‌گذاری برای نگاشت هر نمونه به دسته مورد نظر استفاده می‌شود. وظیفه

[1] Supervised

یادگیری ماشین تحت نظارت، تعریف رابطه بین ویژگی‌های مجموعه نمونه با برچسب‌ها می‌باشد.

بسته به نوع مساله مورد استفاده و ماهیت داده‌ها، روش‌های مختلفی در دهه‌های گذشته برای یادگیری ماشین توسعه داده شده‌اند. برای مساله دسته‌بندی، بسته به اینکه داده‌ها خطی تفکیک‌پذیر باشند یا نه، دسته‌بندهای متفاوتی ارجحیت خواهند داشت. به طور ویژه، برای مساله مورد بررسی در این مقاله، به دلیل خطی تفکیک‌ناپذیر بودن داده‌ها، بهتر است از روش‌های مناسب این رده مانند درخت تصمیم استفاده شود.

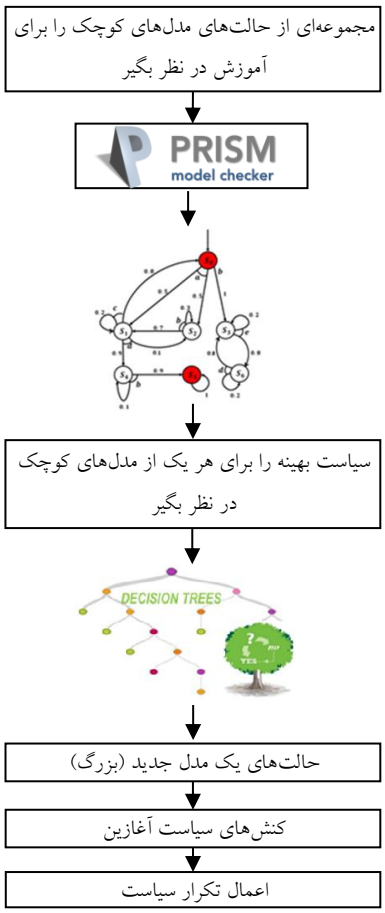
۳،۶. زبان مدل‌سازی PRISM

برای ارائه یک توصیف سطح بالا از یک سیستم کامپیوتری با رفتار تصادفی چندین زبان مدل‌سازی ارائه شده است. در این مقاله زبان مدل‌سازی PRISM به عنوان یک مورد شناخته شده (که در بیشتر ابزارهای وارسی مدل احتمالاتی استفاده می‌شود) در نظر گرفته می‌شود. در PRISM هر مدل به صورت مجموعه‌ای از پیمانه‌ها تعریف می‌شود. هر پیمانه شامل یک یا چندین متغیر با مقادیر صحیح با دامنه مقادیر تعریف شده است. حد بالای مقادیر مجاز هر متغیر را می‌توان به طور صریح و یا بر اساس یک پارامتر مشخص کرد. همچنین ممکن است تعدادی متغیر سراسری تعریف شوند. برای نام‌گذاری متغیرها در حالت کلی از var_i استفاده می‌شود: var_1 برای متغیر اول، var_2 برای متغیر دوم و مانند آن. تعدادی دستور چگونگی تغییر مقادیر متغیرها را مشخص می‌کنند. توصیف سطح بالای یک مدل (که مدل برنامه نام دارد) به یک سیستم تغییر وضعیت سطح پایین (که در مورد مدل‌های غیرقطعی MDP است) ترجمه می‌شود.

یک متغیر را پارامتری گوییم اگر دامنه آن با یک پارامتر برنامه مشخص شود. در نتیجه تعدادی مدل MDP با اندازه متفاوت بر اساس مقادیر گوناگون پارامتر ساخته خواهند شد.

۴. روش ارائه شده برای بهبود تکرار سیاست

تکرار سیاست (PI) یکی از روش‌های استاندارد برای محاسبه احتمال دسترس‌پذیری بهینه برای یک فرآیند تصمیم مارکوف داده شده است. دقت سیاست اولیه σ_1 تاثیر قابل توجهی در کارایی این روش می‌تواند داشته باشد. انتخاب کنش‌های غیربهینه برای حالت‌ها در بیشتر موارد می‌تواند تاثیر منفی در همگرایی این روش به مقادیر واقعی داشته باشد [۲۰]. در حالت کلی، سیاست‌ها به تدریج به سیاست بهینه σ^* همگرا می‌شوند. در نتیجه دقت σ_1 می‌تواند در تعداد کلی تکرارها و زمان مصرفی کلی روش تکرار سیاست موثر باشد. به عنوان نوآوری اصلی این مقاله و برای تخمین یک سیاست مناسب و استفاده از این تخمین به عنوان سیاست آغازین روشی مبتنی بر یادگیری ماشین معرفی و توسعه داده می‌شود. در ادامه از عنوان «تکرار سیاست مبتنی بر یادگیری» که به اختصار LBPI (مخفف learning-based policy iteration) نامیده می‌شود به عنوان نوآوری اصلی این مقاله که ایده اصلی آن در شکل (۲) نشان داده شده است، استفاده می‌کنیم. برای ارزیابی دقت سیاست داده شده σ، زمان مصرفی محاسبات و دقت پاسخ نهایی در نظر گرفته می‌شود.



شکل (۲): شمای کلی روش پیشنهادی

۴،۱. استفاده از یادگیری ماشین برای تخمین سیاست بهینه

یک روش مناسب برای کاهش تعداد تکرارهای روش PI و MPI استفاده از یک سیاست نزدیک به بهینه σ^+ به عنوان سیاست اولیه σ_1 است. به عبارت دیگر، علاقه‌مند به یافتن سیاست نزدیک به بهینه σ^+ هستیم که برای اکثر حالت‌های $s \in S$ کنش‌هایی که σ^+ انتخاب می‌کند با کنش بهینه یکسان باشد، یعنی شرط $\sigma^+(s) = \sigma^*(s)$ برای بخش عمده حالت‌ها برقرار باشد. برای یک فرآیند تصمیم مارکوف داده شده، روش ارائه شده برای محاسبه σ^+ به عنوان تخمینی از سیاست بهینه استفاده از یک دسته‌بند برای نگاشت هر یک از حالت‌ها به کنش بهینه مطرح است. از این حقیقت استفاده می‌شود که اکثر مدل‌های برنامه از یک یا چند پارامتر استفاده می‌کنند که مقدار این پارامتر(ها) در اندازه مدل تاثیرگذار است. مجموعه‌ای از n پارامتر $p_1, p_2, \dots, p_n$ از یک مدل برنامه PRISM و بردار $\overline{Vals}$ از مقدارگذاری برای این پارامترها را در نظر بگیرید. قرارداد می‌کنیم که از $P(p_1 = val_1, \dots, p_n = val_n)$ برای فرآیند تصمیم مارکوف متناظری استفاده کنیم که در آن هر یک از پارامترهای p_i دارای مقدار val_i باشد. به عنوان مثال، از $P(K = 6)$ استفاده می‌شود تا فرآیند تصمیم مارکوف متناظر با مدلی را که در آن مقدار پارامتر برابر ۶ است، نشان دهد.

همان‌طور که در ایده کلی روش LBPI در شکل (۲) نشان داده شده است، ورودی الگوریتم یک مدل برنامه به زبان PRISM است. با در نظر گرفتن مقادیر کوچک مختلف برای پارامترهای این برنامه مجموعه‌ای از حالت‌ها به دست می‌آیند. سپس با توجه به حالت‌های هدف، با اجرای PRISM مقادیر احتمال دسترس‌پذیری بهینه و سیاست‌های بهینه به دست می‌آیند. با در نظر گرفتن مجموعه حالت‌های مدل‌های مورد نظر و کنش‌های بهینه متناظر هر حالت به عنوان ورودی مرحله آموزش، با استفاده از یادگیری ماشین یک درخت تصمیم حاصل می‌شود. استفاده از این درخت تصمیم برای حالت‌های یک مدل جدید می‌تواند یک سیاست خوب را به عنوان سیاست شروع در روش تکرار سیاست پیشنهاد دهد.

روش کلی یادگیری ماشین استفاده از مجموعه‌ای از داده‌های نمونه در مرحله آموزش است. نمونه‌های مساله دارای مجموعه‌ای از ویژگی‌ها هستند که وضعیت هر کدام از این نمونه‌ها را نشان می‌دهند. در مساله مورد نظر ما چندین مدل MDP کوچک به عنوان نمونه در نظر گرفته می‌شوند. این MDPهای کوچک با در نظر گرفتن مقادیر کوچکتر $\overline{Val_i}$ (نسبت به $\overline{Vals}$) ساخته می‌شوند. همچنین هر متغیر برنامه به عنوان یکی از ویژگی‌های مساله در نظر گرفته می‌شود. در نتیجه فضای حالات مدل‌های MDP نمونه به عنوان ورودی برای مرحله آموزش در نظر گرفته می‌شوند. سیاست بهینه هر یک از MDPهای نمونه تعیین کننده برچسب هر یک از حالت‌ها و دسته متناظر آن است. به عنوان مثال، اگر سیاست بهینه برای یک حالت داده شده، اولین کنش را به عنوان کنش بهینه انتخاب کند آنگاه آن حالت در دسته نخست قرار می‌گیرد و به همین ترتیب سایر دسته‌ها برای حالت‌های مختلف در نظر گرفته می‌شوند. به طور دقیق‌تر، با در نظر گرفتن k مدل MDP کوچک $M_1, M_2, \dots, M_k$ در اولین گام شکل (۲) و سیاست‌های بهینه معادل آنها $\sigma_1^*, \sigma_2^*, \dots, \sigma_k^*$ که در گام بعد محاسبه شده‌اند و مدل MDP بزرگ M_{k+1} به عنوان ورودی، باید یک دسته‌بند مناسب اعمال شود. این دسته‌بند به شکلی است که سیاست $\sigma^+ \cong \sigma_{k+1}^*$ به عنوان تقریب مناسبی ارائه شود که در آن برای بیشتر حالت‌های s از مدل M_{k+1} داشته باشیم $\sigma^+(s) \cong \sigma_{k+1}^*(s)$. برای مدل برنامه داده شده P و مجموعه‌ای از n پارامتر آن، k بردار $\overline{vals_1}, \overline{vals_2}, \dots, \overline{vals_k}$ از مقادیر کوچک برای این پارامترها در نظر گرفته می‌شوند (بخش نخست شکل (۲))، تا k مدل MDP به شکل $M_1, M_2, \dots, M_k$ به عنوان نمونه‌های آموزش به دست آیند. هر مدل M_i به صورت $M_i = P(\overline{p} = \overline{vals_i})$ ساخته می‌شود که در آن $\overline{p} = \overline{vals_i}$ به معنای انتساب مقادیر $\overline{vals_i}$ به بردار پارامترهای $\overline{p}$ است. برای ایجاد تمایز میان مجموعه حالت‌های مدل‌های مختلف و ارائه تعریف غیرمبهم از اشیا ورودی، تعدادی ویژگی اضافه در نظر گرفته می‌شوند. با هدف تعمیم بهتر دسته‌بند آموزش یافته، برای هر متغیر پارامتری مثل var_j با در نظر گرفتن $bound_j$ به عنوان حد بالای مقادیر آن (که تابعی از پارامتر p_j است)، یک ویژگی

اضافه به صورت $bound_j - var_j$ (تفاوت مقدار متغیر با حد بالای آن) تعریف می‌شود. در ضمن سیاست دلخواه σ هر یک از حالت‌های $s \in S/G$ را به دسته متناظر آن نگاشت می‌کند.

از یک روش تکراری استاندارد می‌توان برای محاسبه سیاست بهینه هر یک از مدل‌های M_i استفاده کرد. هر چند یک روش عددی تکراری نمی‌تواند تضمین کند که سیاست دقیقا بهینه را محاسبه کند، اما می‌توان با در نظر گرفتن یک مقدار ϵ کوچک از روش تکرار بازه [۷]، برای محاسبه سیاست بهینه با حداکثر خطای ϵ ستفاده نمود. با داشتن مدل‌های MDP کوچک برای مرحله آموزش انتظار داریم زمان اجرای کمی برای مرحله آموزش در مقایسه با زمان لازم برای یافتن سیاست بهینه برای یک مدل بزرگ داشته باشیم.

برای درک بهتر روش ارائه شده، شکل (۳) را در نظر بگیرید که قسمتی از یک سیاست بهینه را به عنوان نگاشتی از ویژگی‌های (متغیرهای) مدل برای سه مدل از یک MDP نمونه را نشان می‌دهد که برنامه آن شامل ۵ متغیر است. همچنین ویژگی اضافه استفاده شده مقدار $bound - counter$ است که در ششمین ستون نشان داده شده است. به ازای هر مقدارگذاری برای ویژگی‌ها شماره دسته متناظر در شکل (۳) نمایش داده شده است. چنین نگاشتی از ویژگی‌ها به برچسب کنش‌ها برای آموزش یک دسته‌بند استفاده می‌شود. در این مثال، هر حالت (مقدارگذاری ویژگی‌ها) به یکی از سه دسته نسبت داده می‌شود و انتظار داریم دسته‌بند آموزش دیده بتواند دسته متناظر با هر حالت (کنش معادل) را برای یک مدل بزرگ داده شده، پیش‌بینی کند. در این مقاله، از درخت تصمیم برای دسته‌بندی استفاده می‌شود. دلیل اصلی انتخاب این دسته‌بند این است که داده‌های مورد استفاده در این مقاله به راحتی تفکیک‌پذیر خطی نیستند و از طرفی نتایج تجربی (بخش ۵) نشان می‌دهد این دسته‌بند با دقت خوبی می‌تواند کنش‌های بهینه را برای مدل‌های داده شده، پیش‌بینی کند. هر چند تضمینی برای یافتن سیاست بهینه در روش ارائه شده در این قسمت وجود ندارد، اما این روش می‌تواند یک تخمین مناسب برای اولین سیاست برای روش‌های PI و MPI ارائه دهد و از تعداد کلی تکرارها (در مقایسه با روش‌های استاندارد) بکاهد. از آنجایی که تمرکز اصلی این مقاله روی ارائه یک تخمین مناسب برای اولین تکرار است، برای ادامه کار از روش PI استاندارد استفاده می‌شود.

$$
\begin{aligned}
&3,1,0,3,1,12 \rightarrow 2\\
&3,1,1,0,0,12 \rightarrow 1\\
&3,1,1,1,0,12 \rightarrow 1\\
&3,1,1,1,1,12 \rightarrow 0\\
&3,1,1,2,0,12 \rightarrow 1\\
&\vdots\\
&3,1,0,3,1,16 \rightarrow 2\\
&3,1,1,0,0,16 \rightarrow 1\\
&3,1,1,1,0,16 \rightarrow 1\\
&3,1,1,1,1,16 \rightarrow 0\\
&3,1,1,2,0,16 \rightarrow 1\\
&\vdots\\
&3,1,0,3,1,20 \rightarrow 2\\
&3,1,1,0,0,20 \rightarrow 1\\
&3,1,1,1,0,20 \rightarrow 1\\
&3,1,1,1,1,20 \rightarrow 0\\
&3,1,1,2,0,20 \rightarrow 1\\
&\vdots
\end{aligned}
$$

شکل (۳): نگاشتی از ویژگی‌های متناظر با هر حالت به کنش‌های بهینه. برای سادگی شماره حالت نشان داده نشده است.

برای درک بهتر چگونگی استفاده از روش ارائه شده یک بار دیگر فرآیند تصمیم مارکوف مثال ۱ را در نظر بگیرید. یادآوری می‌شود هدف از استفاده از فرآیندهای تصمیم مارکوف در این مقاله، مدل‌سازی مساله‌هایی است که جنبه‌هایی از عدم‌قطعیت و احتمالات را با یکدیگر دارند. در صورتی که مدل حاصل به اندازه‌ای باشد که حافظه جاری سیستم قادر به تحلیل آن باشد می‌توان از روش‌های استاندارد وارسی مدل احتمالاتی استفاده کرد. در مورد ربات مثال ۱ در صورتی که اندازه محیط به میزانی باشد که فرآیند تصمیم مارکوف حاصل دچار مشکل انفجار حالات نشود، می‌تواند از روش‌های مثل PI برای تحلیل آن استفاده کرد. اما در صورتی که این فرآیند بیش از اندازه بزرگ باشد (که موضوع اصلی مقاله جاری است)، ابتدا با افزایش سطح انتزاع و کوچک کردن محیط عملیاتی می‌توان چندین MDP استاندارد (که حافظه جاری سیستم قادر به نگهداری اطلاعات آنها باشد) به دست آورد.

۴،۲. تحلیل دقت روش LBPI

معیارهای زیادی را می‌توان برای دقت یک سیاست در روش پیشنهادی این مقاله ارائه داد. روش‌های مورد استفاده استاندارد مانند روش‌های استفاده شده در مراجع [۷]، [۹]، [۱۰] و [۲۰]، مقایسه زمان مصرفی و یا تعداد تکرارهای روش استاندارد و یک روش بهبود یافته است. از طرف دیگر، یک روش کلی در یادگیری ماشین برای تحلیل دقت یک دسته‌بند، مقایسه تعداد پیش‌بینی‌های نادرست روی داده‌های آزمون است. اما در مورد مساله وارسی مدل احتمالاتی تعداد پیش‌بینی‌های نادرست الزاما نشان‌دهنده مناسب بودن معیار برای دقت یک سیاست پیشنهادی نیست. به عنوان مثال، ممکن است احتمال دسترس‌پذیری حالتی مانند s_i به طور مستقیم یا غیرمستقیم روی مقادیر تعداد زیادی از حالت‌های دیگر تاثیرگذار باشد. در چنین حالتی انتخاب یک کنش نادرست برای s_i می‌تواند بر روی مقادیر سایر حالت‌های وابسته (حتی در صورت انتخاب حالت بهینه برای آنها) تاثیر منفی بگذارد. برای تعریف یک معیار مناسب برای دقت محاسبات سیاست پیشنهادی σ^+، احتمالات دسترس‌پذیری محاسبه شده برای حالت‌های DTMC خارج قسمت M^{σ^+} را با مقادیر محاسبه شده با استفاده از روش تکرار بازه مقایسه می‌کنیم. برای هر حالت $s \in S$ و سیاست مورد نظر σ از $precision(s,\sigma)$ به عنوان دقت مقدار محاسبه شده برای s تحت σ استفاده شده و به صورت زیر تعریف می‌شود:

$$precision(s,\sigma) = \frac{\min(p_s^{\sigma}(G), p_s^{II}(G))}{\max(p_s^{\sigma}(G), p_s^{II}(G))} \qquad (۸)$$

توجیه این تعریف این است که برای دسترس‌پذیری کمینه و برای یک سیاست غیربهینه مقدار تخمین زده شده ممکن است بیشتر از مقدار واقعی باشد. در نتیجه مقدار کمتر در صورت کسر و مقدار بیشتر در مخرج کسر قرار می‌گیرد. بر اساس این تعریف و برای سیاست ارائه شده σ^+ توسط روش مبتنی بر یادگیری ماشین، سه کلاس برای دقت مقادیر محاسبه شده در نظر گرفته می‌شود که در رابطه (۹) تا (۱۱) نشان داده شده‌اند.

$$C_1 = \{s \in S | precision(s,\sigma) \geq 0.99\} \qquad (۹)$$

$$C_2 = \{s \in S | 0.9 \leq precision(s,\sigma) < 0.99\} \qquad (۱۰)$$

$$C_3 = \{s \in S | precision(s,\sigma) < 0.9\} \qquad (۱۱)$$

طبق این تعریف‌ها C_1 کلاس همه حالت‌هایی است که مقادیر محاسبه شده برای احتمالات دسترس‌پذیری آنها تحت سیاست σ^+ با مقدار دقیق آن در دو رقم نخست بعد از اعشار یکسان است. C_2 شامل همه حالت‌هایی است که مقدار محاسبه شده برای آنها در اولین رقم معنادار بعد از اعشار با مقدار دقیق یکسان است. هر اندازه که درصد بیشتری از حالت‌ها به کلاس C_1 تعلق داشته باشند، دقت سیاست σ^+ بیشتر بوده و امیدواریم با تعداد تکرار کمتری به مقادیر مورد نظر همگرا شود.

۵. نتایج تجربی

برای اعتبارسنجی و بررسی کارایی روش پیشنهادی LBPI و مقایسه آن با روش‌های PI و MPI پنج کلاس از فرآیندهای تصمیم مارکوف استاندارد به عنوان بررسی موردی در نظر گرفته می‌شوند. این پنج کلاس شامل Consensus، Zeroconf، Mer، Wlan و Firewire از مجموعه بررسی‌های موردی وارسی‌گر PRISM هستند که به طور گسترده در کارهایی از این دست استفاده شده‌اند [۲]، [۳]، [۷]، [۹]، [۲۰]- [۲۲]. برای هر دسته یک پارامتر برای ساخت MDPهای گوناگون در نظر گرفته شده است. در جدول (۲)، برخی اطلاعات مربوط به این دسته از مدل‌ها نمایش داده شده‌اند. در این جدول نام مدل‌ها و مقادیر بکار رفته برای پارامترها در مرحله آموزش ارائه شده است. تعداد متغیرهای برنامه نشان می‌دهد در برنامه PRISM هر دسته از مدل‌ها شامل چه تعداد متغیر است. همچنین تعداد کل حالت‌های بکار رفته در مرحله آموزش برای هر دسته از مدل‌ها نیز ارائه شده است.

برای پیاده‌سازی روش LBPI از وارسی‌گر مدل احتمالاتی PRISM که یک نرم‌افزار متن باز است، استفاده شده است. این پیاده‌سازی با استفاده از ساختمان داده خلوت مورد استفاده در PRISM و با استفاده از زبان C انجام شده است. هرچند مولفه‌های سطح بالاتر PRISM با استفاده از JAVA توسعه داده شده‌اند اما

برای بخش‌های مربوط به محاسبات عددی آن به طور عمده از زبان C استفاده شده است. علاوه بر این، برای ساخت درخت تصمیم از زبان پایتون و کتابخانه DecisionTreeClassifier از مجموعه sckitlearn استفاده شده است.

جدول (۲): اطلاعات مربوط به مدل‌های مورد مقایسه

نام مدل (پارامتر)	پارامترها برای آموزش	تعداد متغیرهای برنامه	تعداد کل حالت‌ها برای آموزش
Consensus (K)	3, 4, 5	9	129408
Zeroconf (K)	2, 3, 4	22	577128
Mer (n)	30, 40, 50	21	1450110
Wlan (TTM)	40, 50, 06	13	713532
Firewire (ddl)	500, 600, 700	3	510732

پیاده‌سازی LBPI انجام شده، اطلاعات مربوط به هر سیاست را (که توسط درخت تصمیم تولید شده) از فایل می‌خواند و از آن به عنوان سیاست شروع استفاده کرده و محاسبات عددی را طبق الگوریتم (۱) ادامه می‌دهد. برای هماهنگی بین هر مدل و اطلاعات سیاست متناظر، از یک برنامه shell استفاده شده است. محاسبات مربوط به ساخت درخت تصمیم در ویندوز و سایر محاسبات در اوبونتو ۲۰ و بر روی کامپیوتری با ۸ گیگابایت حافظه اصلی و پردازنده اینتل corei-7 انجام شده است.

با استفاده از مجموعه نمونه‌های آموزشی و با بکارگیری درخت تصمیم، برای هر دسته از مدل‌های پنجگانه یک درخت تصمیم متناظر ایجاد می‌شود. برای بررسی تاثیر روش LBPI بر کاهش زمان اجرا، برای هر دسته از مساله‌ها، تعدادی مدل بزرگتر از مدل‌های آموزشی در نظر گرفته شده و سیاست بهینه با استفاده از درخت تصمیم ساخته شده است.

به منظور اعتبارسنجی و تحلیل تاثیرگذاری روش پیشنهادی این مقاله بر روی کاهش زمان اجرایی روش PI، تعداد گام‌های لازم برای همگرایی پاسخ‌ها و زمان اجرا برای هر یک از نمونه‌های موردی گزارش و بررسی می‌شود. همچنین از آنجایی که تعداد تکرارها در روش‌های تکراری استاندارد بر روی دقت محاسبات تاثیر مستقیم دارد، میزان خطای ناشی از عدم بهینگی سیاست آغازین را نیز به عنوان معیاری دیگر در نظر می‌گیریم.

در جدول (۳)، نتایج حاصل از اجرای روش PI با استفاده از سیاست‌های ساخته شده پیشنهادی توسط درخت تصمیم با روش‌های PI استاندارد و MPI مقایسه شده است. در تمامی اجراها از $\epsilon = 10^{-6}$ به عنوان مقدار آستانه همگرایی جواب‌ها استفاده شده است. همچنین در تمام موارد زمان اجرایی برحسب ثانیه گزارش شده است. در این جدول نخست نام کلاس‌های مدل و سپس مقادیر پارامترهای مورد استفاده برای ساخت مدل‌ها و نیز تعداد حالت‌های هر یک از مدل‌های ساخته شده، ذکر شده است. برای مقایسه کارایی روش LBPI با دو روش دیگر، تعداد تکرارهای انجام شده و نیز زمان مصرفی تا همگرایی جواب‌ها ارائه شده و همچنین دقت روش LBPI با استفاده از شاخص بخش ۴,۲ در جدول (۳) آمده است. بر اساس این شاخص مشخص شده است که چه درصدی از کل فضای حالات هر مدل به کلاس C_1 (با دقت بالای ۹۹٪)، چه درصدی به کلاس C_2 و چه درصدی به کلاس C_3 تعلق دارد. علاوه بر این، میزان خطای سیاست پیشنهادی در ستون خطای LBPI گزارش شده است. برای محاسبه این خطا ابتدا روش تکرار بازه بر روی هر مدل اعمال شده است که مقدار دقیق احتمال بهینه رسیدن به حالت هدف با شروع از s_0 را تا ۶ رقم اعشار می‌دهد، سپس روش تکرار بازه بر روی DTMC خارج‌قسمت حاصل از اعمال سیاست پیشنهادی σ^+ اجرا می‌شود. مقدار گزارش شده از این روش دقت σ^+ در تعیین احتمال دسترس‌پذیری بیشینه تا ۶ رقم اعشار را نشان می‌دهد. با مقایسه این مقدار با مقدار محاسبه شده نخست، دقت نسبی روش LBPI به دست می‌آید. از آنجایی که در روش PI و MPI به طور معمول از یک سیاست تصادفی به‌عنوان سیاست آغازین استفاده می‌شود و با یک نمونه نمی‌توان نتیجه‌گیری کرد، برای تحلیل بهتر روش LBPI، روش PI را با در نظر گرفتن ۲۰ سیاست تصادفی تحلیل کرده‌ایم. به ازای هر سیاست تصادفی مقدار عددی احتمال دسترس‌پذیری در DTMC خارج قسمت محاسبه شده و میانگین مقادیر و بهترین جواب در نظر گرفته شده است و مقدار خطای نسبی هر یک در جدول (۳) (تحت عنوان RPI) گزارش گردیده است.

با توجه به نتایج تجربی، در بیشتر موارد روش LBPI قادر به کاهش تعداد کل تکرارها و زمان محاسبات است. در مورد کلاس Consensus روش MPI تقریبا دو برابر سریع‌تر از روش

PI بوده و روش پیشنهادی LBPI بین ۲۵ تا ۳۰ درصد تعداد تکرارها و زمان مصرفی را در مقایسه با روش MPI کاهش می‌دهد. برای همه مدل‌های این دسته سیاست پیشنهادی درخت تصمیم قادر است مقادیر مورد نظر را با خطای کمتر از ۱ درصد محاسبه کند و تمام حالت‌ها در کلاس C_1 قرار می‌گیرند.

برای مدل‌های کلاس Zeroconf در حالی که تعداد تکرارهای روش‌های PI و MPI به یکدیگر نزدیکند، روش LBPI قادر به کاهش تعداد تکرارها و زمان مصرفی تا حدود ۹۰ درصد است. هر چند در این روش دقت محاسبات انجام شده مبتنی بر سیاست پیشنهادی، چندان زیاد نیست و بخش قابل توجهی از فضای حالات هر مدل به کلاس C_3 تعلق دارد، اما تعداد کل محاسبات کاهش چشم‌گیری یافته است. در این مثال، خطای LBPI با افزایش K کمی بیشتر می‌شود. در حالی که سیاست‌های تصادفی به طور میانگین دقت پایینی برای نمونه‌های این دسته دارند، امکان رسیدن به یک سیاست خوب با در نظر گرفتن تعدادی سیاست تصادفی افزایش می‌یابد و در نتیجه خطای بهترین جواب، کم است.

باید دقت داشت روش LBPI از سیاست پیش‌بینی شده توسط درخت تصمیم تنها برای سیاست آغازین استفاده می‌کند و پس از انجام محاسبات تکراری بر روی DTMC ساخته شده، برای بهبود سیاست از روشی مشابه الگوریتم (۱) استفاده می‌کند. در نتیجه دقت محاسبات پس از پایان اجرای کامل LBPI (و نه تنها سیاست آغازین) مشابه دقت محاسبات روش‌های PI و MPI است.

جدول (۳): نتایج حاصل از اجرای روش پیشنهادی و مقایسه آن با روش‌های تکرار سیاست استاندارد و تکرار سیاست بهبود یافته

نام مدل	پارامتر	تعداد حالت	تعداد تکرار			زمان اجرا			درصد حالت‌ها			خطای LBPI	میانگین خطای RPI	کمترین خطای RPI
			PI	MPI	LBPI	PI	MPI	LBPI	C1	C2	C3			
Consensus	16	166016	83775	44193	29693	372	219	152	100	0	0	<0.0001	0.6452	0.5813
	24	247936	145698	85243	57659	653	419	308	100	0	0	0.0013	0.718	0.6219
	32	329856	268844	138449	90996	1342	720	519	100	0	0	0.0045	0.7649	0.6548
	40	411776	395012	221453	128254	1961	1252	811	100	0	0	0.0179	0.7826	0.6693
Zeroconf	6	798471	1254	1045	80	19	17.2	4.1	97.6	1.2	1.2	0.0004	0.2317	0.0327
	10	3001911	1881	1742	147	93	87	7.8	78.9	7.5	13.6	0.0098	0.2519	0.0428
	14	4427159	1889	1846	160	125	121	10.5	51.5	0	48.5	0.018	0.2484	0.0315
	18	5477150	1912	1989	144	141	144	15.8	41.4	7.5	51.1	0.0211	0.261	0.03258
Mer	300	1773664	83	83	43	17	17	9.4	46.2	21.9	31.9	0.0178	0.0872	0.0785
	500	2955064	83	83	43	31	31	17	46.2	21.9	31.9	0.0181	0.0921	0.0763
	750	4431814	84	84	43	47	47	28	46.1	22	31.9	0.0183	0.1049	0.0592
	1000	5908564	86	86	43	63	63	36	46.1	21.9	32	0.0184	0.112	0.0674
Wlan	100	1920725	279	279	161	16	16	9.2	99.4	0.4	0.2	0.0003	0.0573	0.0069
	110	2192135	480	480	179	31	31	14	98.5	0.9	0.6	0.0004	0.0829	0.0094
	120	2463145	585	585	210	40	40	17	98	1.2	0.8	0.0006	0.0921	0.0121
	130	2733747	795	795	235	56	56	19	97.9	1.3	0.8	0.001	0.0895	0.0158
Firewire	1000	412062	985	985	987	9.2	9.2	9.3	100	0	0	<0.0001	0.3755	0.3416
	1500	717562	1473	1473	1475	25	25	25	100	0	0	<0.0001	0.4119	0.3284
	2000	1023062	2005	2005	2006	33	33	33	99.7	0.02	0.01	0.0007	0.4058	0.3402
	2500	1328562	2493	2493	2494	42	42	42	99.5	0.3	0.2	0.0008	0.4319	0.3593

برای مدل‌های Mer و Wlan تعداد تکرارها و نیز زمان مصرفی در روش LBPI تقریبا نصف تعداد تکرارها و زمان مصرفی روش‌های PI و MPI است. در مورد مدل‌های کلاس Mer تعداد قابل توجهی از حالت‌ها به کلاس C_3 تعلق دارند. برای این دسته

از مساله‌ها و همچنین دسته Zeroconf می‌توان از روش‌های قدرتمندتر یادگیری مثل روش یادگیری عمیق برای بهبود دقت محاسبات استفاده کرد. برای مدل‌های firewire تعداد تکرارهای روش‌های PI و MPI با تعداد تکرارهای روش LBPI تقریبا برابر است. در این دسته از مساله‌ها سیاست نخست که توسط PRISM برای روش‌های PI و MPI انتخاب می‌شود تا حد زیادی به سیاست بهینه نزدیک است و در نتیجه تعداد تکرارها در روش مبتنی بر یادگیری چندان کاهش نمی‌یابد.

۶. نتیجه‌گیری

در این مقاله، با استفاده از یادگیری ماشین و درخت تصمیم روشی جدید برای ارائه یک تخمین مناسب از سیاست اولیه برای یک فرآیند تصمیم مارکوف ارائه شد. هدف اصلی این روش کاهش زمان مصرفی روش PI در وارسی مدل برای سیستم‌های مدل شده با فرآیندهای تصمیم مارکوف است. نتایج تجربی نشان می‌دهد تعداد تکرارها و در نتیجه زمان مصرفی روش PI با استفاده از ایده ارائه شده تا ۹۰ درصد کاهش می‌یابد، در حالی که دقت محاسبات این روش کاهش چندانی نمی‌یابد. برای پژوهش‌های آینده می‌توان از سایر روش‌ها مانند شبکه‌های عصبی عمیق برای ارائه یک سیاست نزدیک به بهینه استفاده کرد. همچنین با در نظر گرفتن مدل‌های بزرگ می‌توان امکان بسط روش ارائه شده در راستای وارسی مدل آماری را مورد بررسی قرار داد.

تعارض منافع: نویسندگان اعلام می‌کنند که هیچ تعارض منافعی ندارند.

مراجع

[1] C. Baier, J.P. Katoen, Principles of model checking, MIT Press, ISBN 978-0-262-02649-9, pp. 1-975, 2008.

[2] V. Forejt, M. Kwiatkowska, G. Norman, and D. Parker, "Automated Verification Techniques for Probabilistic Systems," in Formal Methods for Eternal Networked Software Systems, vol. 6659, Springer, Berlin, Heidelberg, 2011, doi: 10.1007/978-3-642-21455-4_3.

[3] J.-P. Katoen, "The Probabilistic Model Checking Landscape," in 31st Ann. ACM/IEEE Symp. Logic Comput. Sci. (LICS), New York, NY, USA, 2016, pp. 1-15.

[4] C. Hensel, S. Junges, J.P. Katoen, T. Quatmann, and M. Volk, "The probabilistic model checker Storm," Int. J. Softw. Tools Technol. Transf, vol. 24, no. 4, pp. 589-610, 2022, doi: 10.1007/s10009-021-00633-z.

[5] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of Probabilistic Real-Time Systems," in Computer Aided Verification, vol. 6806, Springer, Berlin, Heidelberg, 2011, doi: 10.1007/978-3-642-22110-1_47.

[6] T.P. Gros, H. Hermanns, J. Hoffmann, M. Klauck, and M. Steinmetz, "Deep Statistical Model Checking," in Formal Tech. Distributed Obj. Comp. Syst. (FORTE), vol. 12136, Springer, Cham., 2020, doi: 10.1007/978-3-030-50086-3_6.

[7] C. Baier, J. Klein, L. Leuschner, D. Parker, and S. Wunderlich, "Ensuring the Reliability of Your Model Checker: Interval Iteration for Markov Decision Processes," in Comput. Aided Verif. (CAV), vol. 10426, Springer, Cham., 2017, doi: 10.1007/978-3-319-63387-9_8.

[8] C. Baier, P.R. D'Argenio, and H. Hermanns, "On the probabilistic bisimulation spectrum with silent moves," Acta Informatica, vol. 57, no. 3-5, pp. 465-512, 2020, doi: 10.1007/s00236-020-00379-2.

[9] T. Brazdil, C. Krishnendu, C. Martin, F. Vojtech, J. Kretínsky, M. Kwiatkowska, D. Parker, and M. Ujma, "Verification of Markov decision processes using learning algorithms," in 12th Int. Symp. Autom. Technol. Verif. Anal. (ATVA), Sydney, NSW, Australia, 2014, pp. 98-114, doi: 10.1007/978-3-319-11936-6_8.

[10] L. Gui, J. Sun, S. Song, Y. Liu, and J.S. Dong, "SCC-Based Improved Reachability Analysis for Markov Decision Processes," in Formal Methods Softw. Eng. (ICFEM), vol. 8829, Springer, Cham., 2014, doi: 10.1007/978-3-319-11737-9_12.

[11] S. Najafi and S. Noferesti, "Determining dynamic time quantum in round-robin scheduling algorithm using machine learning," Soft Comput. J., vol. 10, no.

2, pp. 32-43, 2022, doi: 10.22052/scj.2022.243181.1002 [In Persian].

[12] H. Veisi, H.R. Ghaedsharaf, and M. Ebrahimi, "Improving the Performance of Machine Learning Algorithms for Heart Disease Diagnosis by Optimizing Data and Features," Soft Comput. J., vol. 8, no. 1, pp. 70-85, 2019, doi: 10.22052/8.1.70 [In Persian].

[13] R. Bouchekir and M.C. Boukala, "Toward Implicit Learning for the Compositional Verification of Markov Decision Processes," in Verif. Eval. Comput. Commun. Syst. (VECoS), vol. 11181, Springer, Cham., 2018, doi: 10.1007/978-3-030-00359-3_13.

[14] S. Oleksandr and C.-G. Lee, "Accelerated modified policy iteration algorithms for Markov decision processes," Math. Methods Oper. Res., vol. 78, no. 1, pp. 61-76, 2013, doi: 10.1007/s00186-013-0432-y.

[15] A. Rataj and B. Wona-Szczeniak, "Extrapolation of an Optimal Policy using Statistical Probabilistic Model Checking," Fundam. Informaticae, vol. 157, no. 4, pp. 443-461, 2018, doi: 10.3233/FI-2018-1637.

[16] M. Mohagheghi and K. Salehi, "Machine Learning and Disk-based Methods for Qualitative Verification of Markov Decision Processes," in Proc. 16th Int. Conf. Edu. Res. Ind. Appl. Integr. Harmon. Knowl. Transfer, vol. II, Kharkiv, Ukraine, 2020, pp. 74-88.

[17] M. Mohagheghi, J. Karimpour, and A. Isazadeh, "Improving Modified Policy Iteration for Probabilistic Model Checking," Comput. Sci., vol. 23, no. 1, 2022, doi: 10.7494/csci.2022.23.1.4139.

[18] M. Mohagheghi, J. Karimpour, and A. Isazadeh, "Prioritizing methods to accelerate probabilistic model checking of discrete-time Markov models," Comput. J., vol. 63, no. 1, pp. 105-122, 2020, doi: 10.1093/comjnl/bxz001.

[19] J. Karimpour, A. Isazadeh, M. Mohagheghi, and K. Salehi, "Improved Iterative Methods for Verifying Markov Decision Processes," in Fundam. Softw. Eng. (FSEN), vol. 9392, Springer, Cham., 2015, doi: 10.1007/978-3-319-24644-4_14.

[20] M. Kwiatkowska, D. Parker, and H. Qu, "Incremental quantitative verification for Markov decision processes," in IEEE/IFIP 41st Int. Conf. Dependable Syst. Networks (DSN), Hong Kong, China, 2011, pp. 359-370, doi: 10.1109/DSN.2011.5958249.

[21] A. Hartmanns, M. Klauck, D. Parker, T. Quatmann, and E. Ruijters, "The Quantitative Verification Benchmark Set," in Tools Algorithms Constr. Anal. Syst. (TACAS), vol. 11427, Springer, Cham., 2019, doi: 10.1007/978-3-030-17462-0_20.

[22] F. Ciesinski, C. Baier, M. Grober, and J. Klein, "Reduction Techniques for Model Checking Markov Decision Processes," in 5th Int. Conf. Quant. Eval. Syst., Saint-Malo, France, 2008, pp. 45-54, doi: 10.1109/QEST.2008.45.

[23] T. Salehi, M. Maadani, and M, Mahdavi, "Delay reduction based on Markov decision process in inter-vehicle wireless networks," in 4th Int. Conf. Modern Stud. Comput. Sci. Inf. Technol., Mashhad, Iran, 2017 [In Persian].

[24] R. Sadeghian, "Determining the equilibrium solution in two-player dynamic discrete markovian games with transition probabilities influenced by competitor strategies," Soft Comput. J., vol. 11, no. 1, pp. 48-59, 2022, doi: 10.22052/scj.2022.242848.0 [In Persian].