دو فصلنامه علمی
فناوری اطلاعات و ارتباطات ایران

سال پانزدهم، شماره ۵۵ و۵۶ ، بهار و تابستان ۱۴۰۲
صفحات: ۳۳۰ الی ۳۴۷

Using limited memory to store the most recent action in XCS learning classifier systems in maze problems

Ali Yousefi[*], Kambiz Badie[**], Mohammad Mehdi Ebadzade[***], Arash Sharifi[****]

[*]PhD student, Artificial Intelligence and Robotics, Department of Computer Engineering, Faculty of Mechanics, Electricity and Computers, Science and Research Branch, Islamic Azad University, Tehran, Iran

[**]Professor, Faculty of Information Technology, ICT Research Institute, Tehran, Iran

[***] Professor, Faculty of Computer Engineering, Amirkabir University of Technology, Tehran, Iran

[****] Assistant Professor, Department of Computer Engineering, Faculty of Mechanics, Electricity and Computers, Science and Research Unit, Islamic Azad University, Tehran, Iran

Abstract

Nowadays, learning classifier systems have received attention in various applications in robotics, such as sensory robots, humanoid robots, intelligent rescue and rescue systems, and control of physical robots in discrete and continuous environments. Usually, the combination of an evolutionary algorithm or intuitive methods with a learning process is used to search the space of existing rules in assigning the appropriate action of a category. The important challenge to increase the speed and accuracy in reaching the goal in the maze problems is to use and choose the action that the stimulus is placed on the right path instead of repeatedly hitting the surrounding obstacles. For this purpose, in this article, an intelligent learning classifier algorithm of accuracy-based learning classifier systems (XCS) based on limited memory is used, which according to the input and actions applied to the environment and the reaction of the stimulus, the rules It is optimally identified and added as a new classifier set to the accuracy-based learning classifier systems (XCS) algorithm in the next steps. Among the achievements of this method, it can be based on reducing the number of necessary steps and increasing the speed of reaching the stimulus to the target compared to the accuracy-based learning classifier systems (XCS) algorithm.

Keywords: Learning classifier systems, XCS algorithm, limited memory, maze problems

بکارگیری حافظه ای محدود برای نگهداری برترین کنش اخیردر سیستم های طبقه بندی کننده یادگیر XCS در مسائل هزارتو

علی یوسفی*،کامبیز بدیع**×، محمد مهدی عباد زاده***، آرش شریفی****

*دانشجوی دکتری تخصصی هوش مصنوعی و رباتیک، گروه مهندسی کامپیوتر، دانشکده مکانیک، برق و کامپیوتر، واحد علوم و تحقیقات، دانشگاه آزاد اسلامی، تهران، ایران

** استاد، پژوهشکده فناوری اطلاعات، پژوهشگاه ارتباطات و فناوری اطلاعات، تهران، ایران

*** استاد، دانشکده مهندسی کامپیوتر، دانشگاه صنعتی امیر کبیر، تهران، ایران

**** استادیار، گروه مهندسی کامپیوتر، دانشکده مکانیک، برق و کامپیوتر، واحد علوم و تحقیقات، دانشگاه آزاد اسلامی، تهران، ایران

تاریخ دریافت: ۱۴۰۱/۷/۳۰ تاریخ پذیرش: ۱۴۰۱/۸/۲۴

نوع مقاله: پژوهشی

چکیده

امروزه، سیستم‌های طبقه‌بندی کننده یادگیر درکاربردهای متنوع در رباتیک مانند رباتهای حسی، رباتهای انسان‌نما، سامانه‌های امداد و نجات هوشمند وکنترل ربانهای فیزیکی در محیطهای گسسته و پیوسته، مورد توجه قرار گرفته است. معمولا از ترکیب یک الگوریتم تکاملی یا روش‌های شهودی با یک فرایند یادگیری برای جستجو در فضای قوانین موجود در انتساب کنش مناسب یک دسته‌بند استفاده می شود. چالش مهم برای بالا بردن سرعت و دقت در رسیدن به هدف در مسائل هزار تو، بکارگیری و انتخاب کنشی است که محرک بجای برخورد تکراری به موانع اطراف، در مسیر درست قرار گیرد. بدین منظور در این مقاله یک الگوریتم طبقه بندی کننده یادگیر هوشمند سیستمهای طبقه بند یادگیر مبتنی بر دقت(XCS) مبتنی بر حافظه محدود بکار گرفته شده است که با توجه به ورودی و کنش‌های اعمال شده به محیط و عکس العمل محرک، قوانین بهینه شناسایی شده و در اولویت انتخاب با احتمال بیشتری در مراحل بعدی، به عنوان مجموعه دسته‌بند جدید به الگوریتم سیستمهای طبقه بند یادگیر مبتنی بر دقت (XCS) اضافه گردد. از جمله دستاوردهای این روش می توان به کاهش تعداد مراحل لازم و افزایش سرعت در رسیدن محرک به هدف در مقایسه با الگوریتم سیستمهای طبقه بند یادگیر مبتنی بر دقت (XCS) پایه داشت .

واژگان کلیدی: سیستم های طبقه بند یادگیر، الگوریتم **XCS**، حافظه ی محدود، مسائل هزارتو

× نویسنده مسئول: کامبیز بدیع k_badie@itrc.ac.ir

۱. مقدمه

سیستم‌های طبقه بندی کننده یادگیر (LCS)، سیستم‌هایی مبتنی بر قانون هستند. در این سیستم‌ها قوانین عمدتاً به فرم "IF condition THEN action" هستند. از یک الگوریتم تکاملی یا روشهای شهودی می‌توان برای جستجو در فضای قوانین موجود و در همان زمان از فرایند یادگیری دیگر برای انتساب نحوه استفاده به قوانین موجود می‌توان استفاده نمود که در واقع باعث هدایت جستجو به سمت قوانین بهتر می‌شود [۱] و [۲]. اصطلاح LCS نخست، توسط هلند معرفی شد و در ابتدا توسعه‌ای برای الگوریتم‌های ژنتیکی بود[۲]. سالها بعد، با همکاری رایتمن، سیستمهای شناختی سطح یک (CS-1) ارائه شد[۲]. سپس، هلند کار قبلی را اصلاح کرد تا به شکل استانداردی درآید. با این وجود، سیستم هلند به قدری پیچیده بود که پیاده‌سازی سیستمهای واقعی با آن دشوار بود. سالها بعد، استوارت ویلسون نوعی از LCS به نام XCS را ارائه داد که در آن برازش قوانین تنها بر اساس میزان دقت کنشهای صورت گرفته توسط آنها تعیین می-شود[۳].

کاربردهای سیستمهای LCS در محیطهای واقعی بسیار گسترده است [۱] و [۴] تا [۶]. تقسیم بندی تصاویر زیر دریایی[۷]. استخراج دانش در عامل های مبتنی بر مدل های اقتصادی[۸] و[۹]. فهم الگوها در داده ها[۱۰]. مقایسه استراتژی های اکتشاف واستخراج در مسائل مهندسی [۱۱]. تشخیص ناهنجاری ها بر اساس الگوریتم های فازی رقابتی همکار[۱۲]. تست خودکار در سیستم های محاسباتی ارگانیک [۱۳]. محیط های رقابتی که بر اساس عامل های مبتنی بریادگیری است [۱۴]. اما موارد مهمی چون داده کاوی در زمینه‌های متنوع، کنترل رباتهای فیزیکی در محیط زمان پیوسته با کنشهای کوچک و گسسته نیز از جمله کاربردهای مهم آن است [۱۵] تا [۱۷]. از این منظر، از LCS به عنوان معماری مناسبی برای مطالعه سیستمهای شناختی چون روباتهای حسی نیز می‌توان استفاده کرد[۱۸]. مروری بر پژوهشهای مرتبط نشان می‌دهد که استفاده از فرایند تکاملی سیستمهای یادگیر برای یادگیری و نمایش دانش با استفاده از حافظه به غایت خود نرسیده و فاصله زیادی از طرح بهینه دارد. در دو دهه اخیر این مدل LCS استفاده زیادی در حل مسائل واقعی چون داده‌کاوی داشته است. توجه به این نکته ضروری است که هرچند در دو دهه اخیر، شناخت رسمی از LCS افزایش یافته است اما سه حیطه ی چالش انگیز در توسعه و کاربرد آنها وجود دارد که عبارتند از مقیاس پذیری، کاربرد در محیطهای ایمنی مصنوعی وکاربرد در محیطهای واقعی.

در ادامه به توضیح موارد فوق می پردازیم. حیطه ی سوم، یکی از حیطه های مهم و بحث برانگیز است که در تحقیق پیشنهادی به توضیح ابعاد آن پرداخته و در پی ارائه راه حلی برای آن هستیم. در این حیطه، سیستمهای طبقه بندی کننده یادگیر در کاربردهای واقعی مورد استفاده قرار می گیرند. دقت در عملکرد و همچنین توسیع درست طبقه بندها در این سیستمها با توجه به ورودیهای حس شده از مهمترین موضوعات مورد توجه محققین است. بنابراین، نتایج تحقیق در این حیطه دو حیطه ی دیگر را نیز تحت تاثیر قرار می‌دهد. با توجه به اهمیت این حیطه، پژوهش پیشنهادی در پی ارائه مدلی جدید برای این نوع سیستمهای طبقه بند است که بتواند در محیط‌های واقعی پیچیده، حتی با وجود ناکافی بودن ورودیهای سنسوری که منجر به همپوشانی حالات می شود، پاسخ و کنش نزدیکتری به پاسخ بهینه بیابد. در ادامه به توضیح سه حیطه ی مذکور پرداخته و درنهایت روی رویکرد موجود برای حل مشکلات حوزه سوم و راه حل پیشنهادی تمرکز می کنیم.

یکی از چالشهای اساسی که سیستمهای LCS با آن روبرو هستند، کاربرد آنها برای محیط‌هایی با ابعاد بزرگ است. در چنین محیطهایی سیستمهای مذکور به دلیل افزایش حجم محاسبات، قادر به پیشنهاد کنش مناسب در بازه زمانی مشخص نیستند. در زمینه حل این چالش جدی رویکردهای متنوعی از جمله کنترل عمومی سازی (خصوصی سازی) توسط الگوریتمهای تکاملی[۱۹] تا [۲۴]، استفاده از دانش مسائل در توسعه طبقه بندها [۲۴] و [۲۵] و استفاده از دانش محیطی در توسعه طبقه بندها[۲۶] پیشنهاد شده و برخی پیاده‌سازی و ارزیابی شده است .

چالش اساسی دیگر که سیستمهای LCS با آن روبرو می باشد، کاربرد آنها برای محیطهای ایمنی مصنوعی است[۲۷] تا [۳۲]. همانطور که در کار بول نشان داده شده است، معماری XCSC را می‌توان به عنوان یک سیستم ایمنی مصنوعی [1](AIS) دانست[۳۳]. در طی ۳۰ سال گذشته شباهت‌های میان AIS و XCS پیوسته گزارش شده است، ولی هردو حوزه مستقل از هم توسعه یافته است. روش انتخاب از میان موجودیتهای فعال شده در الگوریتم ژنتیکی XCS، شبیه انتخاب کلونال در AIS است[۲۸]. فرایند تکاملی تحریک شده ی زمانی در XCS، شبیه فرایند تکامل دندریتی در AIS است. بنابراین، یک حوزه ی بالقوه برای پژوهش آتی، موضوع امتزاج متقابل مکانیسمهای این دو حوزه بلوغ یافته با هم است. کارهای انجام شده در این زمینه معدود و انگشت شمار بوده و به تحقیقات [۲۷]و [۲۸] و[۳۰] و [۳۱] محدود می‌شود. چالش اساسی دیگر که سیستمهای LCS با آن روبرو می باشد، کاربرد آنها برای محیط‌هایی واقعی است. از مهمترین کاربردهای واقعی سیستمهای طبقه بند یادگیر، رباتهای انسان نما، سامانه های

[1] Artificial Immune System

یافتن مسیر درست در مسیرهای پر پیچ و خم و رباتهای یادگیر امداد و نجات را می‌توان نام برد[۱۷] و [۱۸] و [۳۴] .

با توجه به اهمیت LCS ها در سیستمهای طبیعی و با عنایت به موضوعات فوق، در این پژوهش مدل یک سیستم طبقه بند یادگیر برای یک سیستم واقعی ارائه می‌شود. سیستم مذکور به کمک مکانیزم حافظه ی پیشنهادی قادر خواهد بود در کنار حالت فعلی سیستم و ورودی، از حالات قبلی سیستم برای ارائه بهترین پاسخ به ورودی استفاده کند. این سیستم از حالات قبلی محرک جهت پیدا کردن کنشی که باعث حرکت محرک می شود استفاده می‌کند. در این موقعیت، محرک به جای درجا زدن و برخورد به موانع اطراف خود، حرکت کرده و احتمال رسیدن به غذا (هدف) در زمان محدود افزایش می‌یابد. بنابراین روش پیشنهادی ارائه شده در این پژوهش باعث افزایش کارایی، دقت و سرعت سیستم دسته‌بند XCS می‌شود که می تواند برای استفاده در محیطهای واقعی بسیار مناسب باشد.

ساختار مقاله به‌صورت زیر سازماندهی شده است. ابتدا در بخش دو، کارهای پیشین در زمینه سیستم‌های طبقه‌بند یادگیر بررسی می‌شود. در بخش سوم، مدل پیشنهادی جهت بهبود عملکرد سیستم طبقه‌بند XCS بررسی می‌شود. در این بخش روش پیشنهادی جهت بهینه‌سازی تعداد مراحل محرک با استفاده از شناسایی شرط-کنش های موثر و اضافه کردن آن به مجموعه دسته‌بندها در سیستم طبقه بند بیان شده است. پیاده‌سازی و ارزیابی کارایی روش پیشنهادی پس از معرفی شاخص‌های کارایی، در بخش چهارم توضیح داده می‌شود. بخش پایانی مقاله به نتیجه-گیری و ارائه راهکارهایی جهت پیشرفت تحقیق موجود اختصاص می‌یابد.

۲. کارهای مرتبط

هلند مفهوم LCS را در راستای توسعه یادگیری تقویتی، یا همان روش تجربه و خطا، ارائه داد. روش‌های یادگیری تقویتی به دنبال آن هستند که ارزش واقعی اجرای هر کنش ممکن را برای هر حالت از مساله مورد نظر به دست آورد. سابقه مطالعه روش یادگیری تجربه و خطا در روانشناسی به ادوارد تورندیک و "قانون تاثیر" او و در کامپیوتر به آلن تورینگ و " ماشین ساخت نیافته نوع P" او بر می‌گردد. کار هلند تحت تاثیر کار آرتور ساموئل برای سیستم پیشنویس/بازرس آغاز شد که خود بر پایه کار کلود شانون برای بازی شطرنج بود

آنچه مورد توجه هلند بود این بود که چگونه یک سیستم هوش مصنوعی خود را بصورت پیوسته با تجربه‌های نو و توسعه یافته مهم قبلی سازگار می‌کند و به این منظور او به دنبال پاسخ این سوال بود که چگونه می‌توان سیستمی مناسبی برای نمایش دانش با استفاده از یادگیری پیوسته و قابل انعطاف مبتنی بر تجربه و خطا با افزایش پاداش ارائه داد. نوعی از الگوریتم‌های ژنتیک برای این منظور مورد استفاده قرار گرفت. این پیشنهاد که یک شبیه-سازی فرایند تکاملی در هوش مصنوعی موضوعی سودمند است توسط آلن تورینگ مطرح شد و منجربه ظهور CS-1 گردید.

در هر چرخه زمانی گسسته، CS-1 یک توصیف رمز شده دودویی از وضعیت فعلی محیط خود به دست می‌آورد. سپس، بر اساس ورودی، کنش قبلی و محتوای فعلی، یک فضای حافظه پاسخ مناسبی را تعیین می‌کند که لیست پیام نام دارد. این موضوع در شکل ۳ نمایش داده شده است. پایگاه قوانین در این سیستم، از N قانون به فرم شرط- ادعا[1]است که به هریک از آنها یک طبقه بند گویند. شرایط قوانین رشته هایی ازنمادهای رمزگذاری سه مقداری{# ,0,1} است. نماد # به عنوان "همه-منظوره" عمل نموده و اجازه تعمیم شرط به "0" یا "1" را می‌دهد. به عنوان مثال شرط 1#1 با 101 و ۱۱۱ تطبیق دارد. بخش ادعا در هر قانون مشتمل بر یک کنش[2] و یک پیام داخلی[3] است که هردو به صورت رشته های دودویی نمایش داده می‌شوند. همه اجزا قانون، ابتدا به صورت تصادفی مقداردهی می‌شوند. همچنین، در کنار هر قانون یک سری پارامتر شامل، سن قانون، دفعات استفاده شدن و پیش بینی از مقدار پاداش حاصل از استفاده شدن (که معمولا به عنوان معیار برازش نیز استفاده می‌گردد) دیده می‌شود.

چند سال بعد، هلند CS-1 را اصلاح نمود و توضیح داد که چه چیزهایی برای رسیدن به یک معماری استاندارد باید تغییر کند. او سیستم جدید را "سیستم طبقه بندی کننده یادگیر" نامید[۲]. به نظر می رسد که هلند از اصطلاح یادگیری در آن زمان استفاده نکرده است و گولدبرگ اولین کسی است که چنین اصطلاحی را استفاده نموده است[۳۵].

مهمترین تغییر ایجاد شده از CS-1، این بود که سیستم یادگیری تقویتی بر مبنای یک استعاره اقتصادی به نام "زنجیره سطلها"[4] معرفی شد که در آن کاربرد قانون با میزان تعهدات اعتباری داوری می‌شد و امروزه در مباحث تشخیص طبقه بندی های پزشکی مورد استفاده می باشد[۳۶]. دراین روش، قوانینی که در زنجیره‌های زمانی عمل می‌نموده و منجر به یک پاداش خارجی می‌شدند به عنوان افراد میانی زنجیره های عرضه و تقاضا در نظر گرفته می‌شوند. قوانین، در بردارنده پارامتری به نام "قدرت"[5] هستند که نشان دهنده اعتبار آنها در این زنجیره است. این پارامتر

[1] Condition-Assertion

[2] Action

[3] Internal message

[4] Bucket brigade

[5] Strength

هم برای انتخاب کنش و هم برای کشف قوانین جدید توسط الگوریتم ژنتیک مورد استفاده قرار می‌گیرد. در این مدل لیست پیام‌ها توسعه یافته تا چندین قانون بتوانند ادعاهای خود را ارسال کنند. شرایط قوانین دیگر ساختار ثابتی برای توجه به وضعیت محیطی فعلی، محتوای لیست پیام‌ها و کنش نهایی ندارند. به جای آن، همه شرایط و ادعاها طول یکسانی دارند و شرایط می‌توانند یک NOT منطقی را هم داشته باشند. ادعاها از همان الفبای مورد استفاده در شرایط ساخته می‌شوند به طوری که اطلاعات ممکن است از شرط یا رشته‌ای (ورودی خارجی یا پیام داخلی) که قانون در صورت حضور # با آن انطباق پیدا می‌کند عبور نماید.

بوکر نوعی از سیستم استاندارد هلند را ارائه داد که در آن ایده استفاده از الگوریتم ژنتیک برای کشف تنظیمات موجود در فضای مساله در راستای جداسازی وظیفه یادگیری چنین ساختاری از پشتیبانی از بخش کنشهای مناسب برای دریافت پاداش بیرونی، توسعه داده شده بود[۳۷]. اینجا برای هریک از این دو بعد مذکور، یک LCS متفاوت وجود دارد. اولین LCS، توصیف رمزگذاری شده‌ای را به صورت دودویی از محیط با هدف کشف و یادگیری قواعد مناسب درون موارد عمومی قابل مشاهده از محیط دریافت می‌کند. این موضوع معادل با یادگیری نمایش طبقه بندی های موضوعات در محیط است. قوانین تطبیق یافته، نه تنها پیامهای خود را به لیست پیامها ارسال می‌کنند که برخی از آنها خود به عنوان ورودی به LCS دوم پاس داده می‌شوند. بنابراین LCS دوم فقط زمانی که چنین طبقه‌بندیهایی را با توجه به وظیفه فعلی خود به درستی استفاده کند پاداش می‌گیرد.

بوکر نوآوریهایی چون تطبیق جزیی و سطوح تحریک قوانین را دارد؛ اما ایده آن در واقع محدود نمودن فرایند کشف قوانین جدید به قوانین فعالی (و نه همه قوانین پایگاه دانش) است که نشان داده‌اند تاثیر بیشتری دارند. در این سیستم والدین از داخل لیست پیامها [M] انتخاب می‌شوند. بنابراین از ترکیب قوانین با کلیاتی که به طور قابل توجهی ابعاد مختلف مساله را نشان می‌دهند ممانعت می‌شود. بوکر ایده ی خود را بر مبنای تحریک الگوریتم ژنتیک در خلال یادگیری توسعه داد و اجازه داد که این فرایند با نرخ ثابتی، مشابه یادگیری تقویتی هلند در حال اجرا بماند. به ویژه که قوانین در این مدل یادگیری در بردارنده تقریبی از ثبات خود هستند که در واقع نشان دهنده میزات تغییرات آنها در صورت دریافت پاداش است. اگر درصد مشخصی از قوانین در [M] بی‌ثباتی بالاتر از یک حد آستانه‌ای داشته باشند میزان برازندگی قوانین باثبات افزایش یافته و الگوریتم ژنتیک اجرا می‌شود. بنابراین طبقه بندهای باثبات برای الگوریتم ژنتیک جذاب تر خواهند بود. سیستم XCS از هر دو نوع ژنتیک بنیادی و ژنتیک تحریک شده با آستانه ثبات استفاده می‌کند[۳۸] تا [۴۰].

استوارت ویلسون با استفاده از شبیه سازی سیستم LCS هلند به فکر شبیه سازی هوش انسان و حیوان افتاد. اولین سیستم ایجاد شده در این راستا ANIMAT نام داشت که در ساختار LCS هلند، ساده سازیهایی را درنظر گرفته بود و بعنوان مسائل مرتبط با هوش نرونی امروزه نیز بکار گرفته شده است[۴۱]. به ویژه، در این سیستم لیست پیام‌ها حذف شده و قوانین تطبیق یافته بر اساس کنش در فرایند زنجیره انسانی گروه‌بندی می‌شدند و مجموعه‌کنش‌ها[1]، [A] را ایجاد می‌کردند. الگوریتم ژنتیک در چنین سیستمی حساس به کنش قوانین است و تا حدی شبیه CS-1 است: اولین والد بر اساس قدرت از پایگاه دانش انتخاب می-شود، اما دومی از زیرمجموعه‌ای از جمعیت که کنش یکسانی داشته باشند انتخاب می‌شود. سیستم ANIMAT یک عامل ساده را در محیط دوبعدی کنترل می کند که قادر است محتوای هشت موقعیت اطراف خود را حس کند و در صورت باز بودن مسیر در هریک از این هشت مسیر حرکت کند. ویلسون نشان داد که یادگیری ممکن است حتی تاحدی که در آن پروژه، مسیرهایی که منتج به پاداش غذا بودند کشف شد. با این‌حال، او متوجه شد که سیستم برای یادگیری تقویتی طبقه بندهای عجول چیزی نداشت! برای تحریک رسیدن به پاداش از یک موقعیت آغازین با کوتاه‌ترین مسیر، ساختار قوانین را تغییر داد تا تخمینی از تعداد گامهای بعدی که برای گرفتن پاداش لازم است نگهداری شود و بر اساس تخمین فرزندان هر قانون به روز شود. این ایده در بخش انتخاب کنش، از طریق تقسیم قدرت بر فاصله اعمال شد. علاوه براین، ANIMAT یک عملگر بازترکیب کننده دارد که بیت‌های نامشابه را در شرایط والد با یک # جایگزین می‌کند تا عمومی‌سازی مفید حاصل شود.

ویلسون سپس با ارائه" سیستم سطح صفر" خود،ZCS ، سیستم ANIMAT را ساده‌تر کرد[۴۲] و [۴۳]. بخش مهم و قابل توجه در این کار آن بود که الگوریتم زنجیره انسانی در کار جدید اصلاح شده بود تا مکانیسمی از یادگیری تفاضلی زمانی را نیز در کار وارد سازد.

نتایج به دست آمده از ZCS نشان می‌دهد که این سیستم قادر است کارایی خوب اما نه ایده آلی داشته باشد[۳۶] و [۳۵] و [۴۲]. ویلسون نسخه‌ای از الگوریتم بدون سیاست در یادگیری تفاضلی زمانی به نام سیستم Q-learning را در کنار الگوریتم اصلی استفاده نمود [۳۵] و [۴۴]. او پیشنهاد داد که از الگوریتم بنیادین ژنتیکی در فرم توضیح داده شده استفاده شود. بول این موضوع را تحقیق نمود. علاوه براین نشان داده شده است که ZCS در برخی مسائل نمونه کارایی خوبی دارد ولی به نظر می‌رسد به

[1] Action sets

برخی از پارامترهای خود، حساس است. همچنین کارایی ZCS درمحیط‌های نویزی بیشتر ازXCS است[۱۵]. هرچند XCS برخی ویژگیهای اساسیZCS را دارد ولی تفاوتهای پایه‌ای با ZCS را داراست[۲۰].

پس از ایجاد تغییراتی در معماری ANIMAT و قبل از ارائه ZCS، ویلسون معماری خاصی را ارائه داد که برای یادگیری تقویتی طراحی شده بود و در آن پاداش فوری هم در نظر گرفته شده بود.

سیستم BOOLE برای کاربردهای تصمیم سازی دودویی ایجاد شده بود که امروزه در سیستم ACS نیز کاربرد دارد [۴۵]. سیستم BOOLE مکانیسم $[M] \rightarrow [A]$ را از ANIMAT دارد ولی لیست پیام [M] در آن حذف شده است. همچنین، محدودیت انتخاب والد دوم در الگوریتم ژنتیک (شرط داشتن کنش یکسان با والد اول) در عمل آمیزش، حذف شده است که باعث می‌شود طبق مکانیسمی که در ذات ساختارZCS است، قدرت والدین از طریق انتقال به فرزندان کاهش یابد. نشان داده شده است که کاهش قدرت باعث فشار بر قوانین عمومی تر می‌شود تا سریعتر به روز شده و بنابراین سریعتر پاداش دریافت کنند[۴۶]. همچنین در ZCS، قوانینی که در[M] باشند و در [A] نباشند، قدرت خود را با نرخ مالیات مشخصی از دست می‌دهند.

ساختار سیستم طبقه بند یادگیری که توسط هلند ارائه شد به صورت قوانین شرط –کنش –پرداخت[1] بود. تفاوت اساسیXCS ویلسون با معماری هلند آن است که معیار برازندگی در این سیستم، به جای ارزش پیش‌بینی، دقت پیش‌بینی است. هرچند مشکلاتی چون عمومی‌سازی توسط این معماری حل شد اما از آنجا که XCS مانند CS-1 دارای لیست داخلی پیام و یا ساختار حافظه‌ای دیگر نبود، تنها برای یادگیری سیاست بهینه در سیستمهای مارکوفی قابل استفاده می باشد. در چنین سیستمهایی سیاست بهینه تنها به وضعیت ورودیهای سنسوری سیستم بستگی دارد و عامل می‌تواند حالت محیط را به صورت کامل براساس این ورودیها تعیین کند. اما در بسیاری از کاربردها، ورودیهای سنسوری اطلاعات جزیی در مورد حالت فعلی محیط در اختیار عامل قرار می‌دهند. از این رو عامل در تشخیص حالت دقیق سیستم ناتوان بوده و حالات همپوشان توسط عامل طبقه بند یکسان در نظر گرفته می‌شوند. در چنین حالتی عامل طبقه بند به مشکلی به نام حالت مخفی یا مشاهدات همپوشان دچار شده و کنش صادر شده از سوی عامل در پاسخ به ورودی ها دقت پایینی خواهد داشت. در چنین وضعیتی گفته می‌شود که سیستم غیر مارکوفی است یا گفته می‌شود محیط به صورت جزیی قابل مشاهده[2] است. در برخورد با محیطی که به صورت جزیی قابل مشاهده است عامل طبقه بند نیازمند حافظه‌ای است که با کمک آن بتواند نقصان داده‌های سنسوری ورودی را جبران نماید زیرا در این وضعیت سیاست بهینه تنها بر اساس ورودی‌های سنسوری قابل تشخیص نیست[۴۷].

معماری اولیه ارائه شده توسط هلند دارای یک لیست داخلی پیام است. این لیست برای ذخیره اطلاعات مورد استفاده قرار می‌گیرد و به عبارت دیگر می‌توان از آن به عنوان یک حافظه ی موقت استفاده نمود. با این وجود، تحقیقات نشان می‌دهد که معماری هلند در برخورد با محیط‌های غیرمارکوفی موفقیت ناچیزی دارد. تحقیقات دیگری در این زمینه مطرح شده و رویکردهای متنوعی ارائه شده است.

اولین تلاش در این زمینه توسط ویلسون در سال ۱۹۹۵ انجام شد[۳] و توسط لنزی در سال ۱۹۹۸ پیاده سازی شد[۲۳]. به پیشنهاد آنها حافظه داخلی به فرم ثبات به سیستم افزوده شد که دارای یک یا تعداد بیت محدودی بود. در ادامه، آنها معماری پیشنهادی را با افزودن یک شرط و یک کنش توسعه دادند که برای حس کردن و اعمال سیاست روی ثبات داخلی استفاده می‌شد. سیستم مذکور XCSM نام داشت. این سیستم بر سیستم‌های واقعی غیر مارکوفی دارای ورودیهای سنسوری همپوشان اعمال شد. نتایج کار ویلسون و لنزی نشان داد که XCSM تنها در مسائل ساده ای که ورودیهای در دو یا حداکثر چهار وضعیت همپوشان هستند خوب و کارا است اما در بقیه موارد چون مسائل مسیرهای پرپیچ وخم، XCSM به پاسخ بهینه همگرا نمی‌شود. این انگیزه باعث شد تا لنزی نسخه توسعه یافته XCSM را به نام XCSMH ارائه دهد. در این طبقه بند یادگیر، سیاست جدیدی برای به - روزرسانی حافظه داخلی استفاده شد و از یک استراتژی سلسله مراتبی برای انتخاب کنش مناسب در مقابل ورودی استفاده گردید. آزمایش سیستم XCSMH روی مسائل مسیر پرپیچ با پیچیدگیهای متفاوت نشان داد که این سیستم قادر است پاسخی نزدیک به پاسخ بهینه در مسائل ساده به دست آورد.

سپس، حمزه و همکاران دو سیستم طبقه بند با معماری توزیع شده ارائه دادند که PSXS و RPXCS نام داشت [۴۸] و [۴۹]. در این سیستم‌ها همپوشانی شرایط تشخیص داده شده و ورودی‌ها به سمت XCS های با رتبه کمتر هدایت می‌شود. XCS های رتبه پایین، مجهز به پنجره‌های سابقه بوده و هریک مسئول یکی از حالات همپوشان هستند. در این سیستم، ساختار طبقه بند بسیار پیچیده است. نتایج موید موفقیت روش حمزه و همکاران در برخی تستهای استاندارد است.

[1] condition-action-payoff

[2] Partial obsevable

درتحقیق دیگر، پرین و بول از روش گراف مبتنی بر برنامه نویسی پویای ژنتیکی[1] برای نمایش قوانین در XCS و XCSF (که معادل یک نوع شبکه اتفاقی بولین است) استفاده کردند. سیستم DGP-XCS و DGP-XCSF آنها، برخی مسائل مسیر پرپیچ را حل می نماید اما پاسخ به دست آمده در کار آنها بهینه نیست[۵۰].

اخیراً، ژانگ و همکاران برای بهبود کارایی XCS روشی جدید برای استفاده از حافظه در سیستم‌های طبقه بند یادگیر ارائه دادند[۳۴]. در روش آنها یک لیست داخلی پیام به عنوان لیست حافظه به سیستم اضافه می‌شود که طول آن برابر بارشته‌های ایجاد شده توسط تشخیص دهنده‌ها است. در این روش برای برخورد با حالات غیرمارکوفی محیط تعداد کمی از طبقه بندها با شرایط حافظه توسعه می‌یابند. همچنین، یک شرط توسعه یافته به صورت (شرط - حافظه) در کنار مکانیزمی برای تشخیص همپوشانی حالت‌ها، کارایی و دقت طبقه بند را در حالتهای مخفی افزایش می‌دهد. طبقه بند XCSMD کارایی نزدیکتری به حالت بهینه دارد و بهتر از طبقه بندهای قبلی در محیط‌های واقعی عمل می کند. اما مشکل جدی این روش هدررفت حافظه و سربار اعمال شرط – حافظه (برای رد شدن از حالات همپوشان که ایجاد ابهام می‌کنند). در چنین حالتی توسعه پذیری سیستم با افزایش اندازه ی محیط دچار مشکل سربار محاسبات و حافظه خواهد بود. لذا در اعمال طبقه بند یادگیر به سیستم‌های پیچیده، پاسخ‌ها از پاسخ بهینه فاصله زیادی خواهد داشت.

۳. الگوریتم XCS

در این بخش مختصراً الگوریتم XCS توضیح داده می شود. در این پژوهش از این الگوریتم برای بهبود حرکت ANIMAT وتصمیم گیری آن برای رسیدن به هدف استفاده شده است.

XCS از یک جمعیت [P] که بیانگر طبقه بند می باشد تشکسل شده است. هر طبقه بند، شامل یک قانون و یک مجموعه است. هر طبقه بند شامل سه پارامتر اصلی است که در ادامه توضیح داده می شود:

۱- پیش بینی بازده : میزان بازده ی یک طبقه بند در صورت انتخاب عمل خود، دریافت می کند.
۲- خطای پیش بینی: تخمین میزان خطای پیش بینی طبقه بند و بازده ی دریافت شده را مشخص می کند.
۳- تابع ارزیاب : معمولا به صورت یک تابع معکوس از خطای پیش بینی تعریف می شود.

۳.۱. مولفه ی کارایی

در هر مرحله، یک نمونه ورودی از محیط اخذ می شود. مطابقت این ورودی با مجموعه ای از دسته بندهای موجود [P] توسط مجموعه [M] صورت می گیرد. اگر ورودی با شرط موجود در دسته بند مطابقت داشته باشد آن دسته بند انتخاب خواهد شد، در غیر اینصورت (اگر مجموعه [M] خالی باشد) و یا شرایط ورودی با شرایط موجود در دسته بند برابر نباشد، عملگر پوشش با شرایط جدید، یک دسته بند جدید را را ایجاد می کند. ممکن است در این حالت کنش جدیدی در مجموعه [M] وجود نداشته باشد. در صورتیکه یک یا چند دسته بند انتخاب شود، برای هر کنش از این دسته بند ها پارامتر پیش بینی بازده محاسبه می گردد و برای هر کدام از دسته بندها، تابع ارزیاب بطور میانگین بدست می آید. هر دسته بند که تابع ارزیاب مناسبتری داشت کنش مربوط به آن دسته بند با اکتشاف یا استخراج با بالاترین پیش بینی انتخاب می شود.

۳.۲. فاز به روزرسانی

با انتخاب بهترین عمل و اعمال آن به محیط پاداش r از محیط دریافت می شود. از این پاداش در تنظیم پارامتر های طبقه بند در مجموعه [A] استفاده می شود

ابتدا طبق رابطه (۱) زیر مقدار p تنظیم می شود.

$$p \leftarrow p + \beta(1-p) \qquad (۱)$$
$$0 \le \beta \le 1$$

در ابن رابطه βنرخ یادگیری است. خطای پیش بینی طبق رابطه (۲) به هنگام می شود.

$$\varepsilon \leftarrow \varepsilon + (|r-p| - \varepsilon) \qquad (۲)$$

سپس دقت طبقه بند با استفاده از تابع معکوس رابطه (۳) بدست می آید.

$$f(v) = \begin{cases} k = \alpha\left(\varepsilon / \varepsilon_0\right)^{-v} & \text{for } \varepsilon \ge \varepsilon_0 \\ 1 & \text{otherwise} \end{cases} \qquad (۳)$$

پارامتر ε_0 $(\varepsilon_0 \ge 0)$ مشخص می کند که دقت خطای حد آستانه یک دسته بند چقدر است. همچنین پارامترهای α ، $(0<\alpha<1)$ و v ، $(v>0)$ کاهش درجه دقت را کنترل می کنند. در پایان تابع ارزیاب طبق رابطه (۴) با استفاده از دقت نسبی k' بهنگام می گردد. در این رابطه مقدار k' یک دقت نسبی است که از تقسیم دقت k به کل دقت کنش ها بدست آمده است.

[1] Graph based Dynamic Genetic Programming

$$F \leftarrow F + \beta(K' - F) \quad (۴)$$

۳.۳. مولفه ی کشف

معمولا از الگوریتم ژنتیکی (GA) در XCS بطور خاص در بخش مجموعه کنش ها استفاده می شود. ابتدا دو والد از مجموعه واقعی کنش ها [A] با احتمال مناسب از تابع ارزیاب انتخاب می شود. سپس با استفاده از عملگر های جهش و ترکیب، فرزندان والد بدست می آید. .معمولا در این راستا برای انجام XCS پارامتر های مختلفی نیز می بایستی تنظیم شود این پارمترها بصورت کامل در جدول(۱) آورده شده است.

جدول (۱): توصیف پارامترهای مورد نیاز در XCS

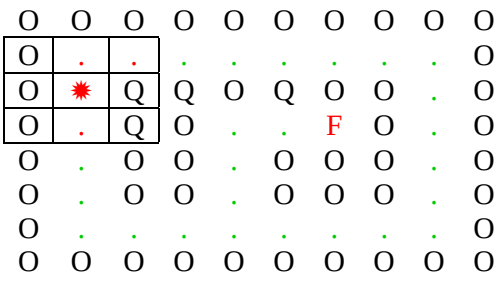
نام پارامتر	توصیف پارامتر	مقدار پارامتر
condition size	number of bits in condition	24
dontcare probability	probability of having a don't care in a random condition	0.5
mutate with dontcare	true if #s are used in mutation	on
crossover method	default crossover type (0=uniform, 1=one-point, 2=two-points)	1
mutation method	default mutation type (1=niche mutation, 2=two-values, 3=pure or three values)	1
number of bits	number of bits in action	3
learning rate	beta parameter	0.2
discount factor	gamma parameter, the discount factor	0.7
covering strategy	strategy for create covering classifiers	action_based 0
discovery component	true if the GA is on	on
theta GA	threshold for GA activation	25
crossover probability	probability to apply crossover	0.8
mutation probability	probability to apply mutation	0.01
epsilon zero	epsilon zero parameter, determines the threshold	10
vi	vi parameter, determines the decay rate	5
alpha	alpha parameter, determines the start of the decay	0.1
prediction init	initial prediction for newly created classifiers	10.0
error init	initial predition error for newly created	0.0
fitness init	initial fitness for newly created classifiers	0.01
set size init	initial set size for newly created classifiers	1
population init	init strategy for [P]	empty
exploration strategy	string to set exploration strategy	SEMIUNIFORM:1.0
deletion strategy	deletion strategy	ACCURACY-BASED
theta delete	theta_del parameter for deletion	20
theta GA sub	threshold for subsumption activation	20
theta AS sub	threshold for subsumption activation	100
GA subsumption	true if GA subsumption is on	on
AS subsumption	true if AS subsumption is on	on
update error first	if true, prediction error is updated first	on
use MAM	reading MAM settings	on
GA tournament selection	reading GA tournament selection	off
tournament size	Tournament size	0.4

۴. روش پیشنهادی

در این بخش به بررسی، انتخاب و بکارگیری آن دسته از شرط کنش های موجود در دسته بند سیستم طبقه بند یادگیر مبتنی بر دقت که بهترین عملکرد را در جلوگیری از برخورد به موانع تکراری و قرار گرفتن در مسیر درست در مسائل هزار تو مورد نیاز است پرداخته شده است. در تمامی این موارد که در بخش های بعد به صورت جزیی و کامل به آن پرداخته شده است، با انتخاب یک حافظه مناسب که در نگهداری شرط کنش موثردر الگوریتم پیشنهادی آورده شده است می تواند به حل مسائل هزارتو که در آن کاهش مراحل رسدن به هدف و افزایش سرعت و دقت در آن بسیار مهم است، منجر شود.

۴.۱. انتخاب شرط-کنش مناسب در هر مرحله

محرک در برخی مواقع در نقطه‌ای قرار می گیرد که اطراف آن موانعی وجود دارد. به‌عنوان‌مثال در شکل ۱ محرک (*) در موقعیتی قرار گرفته است که تنها سه نقطه خالی در اطراف خود جهت حرکت دارد. در این شکل این نقاط با رنگ قرمز نشان داده شده است. مابقی نقاط اطرافش (۵ نقطه دیگر) موانع وجود دارد. محرک با انتخاب کنش هایی همچون 000، 001 و 100 محرک می‌تواند حرکت کند در غیر این صورت با انتخاب کنشهای دیگر محرک به مانع برخورد کرده و حرکت نمی کند. شکل ۲ تاثیر کنش‌های انتخاب شده جهت اعمال به محیط و نحوه حرکت محرک نشان داده شده است.



شکل ۱. مثالی از محرک در محیط

7	0	1
6	✸	2
5	4	3

111	000	001
110	✸	010
101	100	011

شکل ۲. کنش اعمال شده به محیط و نحوه حرکت محرک

Step	Condition	Action
1	000000011011000010010010	011
2	000000011011000010010010	010
3	000000011011000010010010	010
4	000000011011000010010010	100
5	000011011010000010010010	010
6	000011011010000010010010	011
7	000011011010000010010010	010

8	000011011010000010010010	001
9	000011011010000010010010	110
10	000011011010000010010010	000
11	000000011011000010010010	101
12	000000011011000010010010	101
13	000000011011000010010010	101
14	000000011011000010010010	011

شکل ۳. مثالی از Detector وَ Action محرک در ۱۴ مرحله

شکل ۳، detector و کنش چندین مرحله از حرکت محرک شکل ۱ را نشان می دهد. در مرحله اول کنش ۰۱۱ انتخاب و به محیط اعمال شده است. اما با توجه به اینکه این نقطه از موقعیت فعلی محرک، مانعی وجود دارد، محرک حرکت نکرده و در محل خود می ماند. در مرحله دوم کنش ۰۱۰ به محیط اعمال شده است که این نقطه نیز مانع وجود دارد. در مرحله سوم دوباره کنش ۰۱۰ که باعث عدم حرکت محرک می شود انتخاب شده است. اما در مرحله چهارم کنش ۱۰۰ که باعث حرکت محرک به سمت پایین می شود انتخاب شده است. بنابراین یکی از کنش هایی که باعث حرکت محرک با این ورودی از detector می شود، ۱۰۰ است.

در مرحله یازدهم دوباره محرک ورودی مشابه به ورودی‌های مرحله یکم تا چهارم دارد. در این وضعیت محرک پس از پنج کنش مختلف که به محیط اعمال کرده با کنش ۰۰۰ باعث حرکت محرک به سمت بالا شده است. محرک در مرحله چهاردهم کنش مربوط به مرحله اول را که باعث برخورد محرک با مانع می شود را تکرار کرده است. همین‌طور در مرحله هجدهم از کنش ۰۱۰ که قبلاً در مرحله دوم استفاده شده است به محیط اعمال می کند. بنابراین با انتخاب چنین کنش هایی محرک از نتایج مراحل قبل جهت بهبود رفتارش هیچ استفاده ای نمی کند.

هدف از ارائه شکل فوق این است که محرک در صورت دریافت ورودی تکراری از محیط، از کنش های تکراری که قبلاً از آن‌ها استفاده کرده و باعث حرکتش نشده است، استفاده نکند. در عوض از کنشی که باعث حرکت محرک در آن موقعیت شده است استفاده کند. به‌عنوان‌مثال محرک در مرحله یازدهم از کنش مرحله چهارم استفاده کند. نحوه شناسایی این شرط – کنش مؤثر در بخش بعد ارائه شده است.

۲.۴. شناسایی شرط-کنش های مؤثر در حرکت محرک

در این بخش شرط - کنشی که باعث حرکت محرک پس از امتحان کردن بیش از یک کنش می شود را شناسایی خواهیم کرد. در هر مرحله شرط-کنش را ذخیره می کنیم. از طرفی در هر مرحله در صورت تکرار شدن شرط قبلی تعداد مراحلی که این شرط به‌صورت متوالی تکرار می شود را ذخیره می کنیم. سپس در مرحله ای که شرط فعلی نسبت به قبلی تغییر کرد، شرط-کنش قبلی به‌عنوان شرط-کنش مؤثر در حرکت محرک شناسایی می شود.

با توجه به مقادیر شرط و کنش مرحله چهارم شکل۳ :

000000011011000010010010	100

سبب حرکت موثر محرک پس از سه بار تلاش شده است.

همچنین در مرحله دهم شرط-کنش:

000011011010000010010010	000

یک دسته بند دقیق است. اگر تنها معیار ما حرکت محرک در برابر عدم حرکت محرک باشد، این می تواند به‌عنوان یک دسته بند دقیق در XCS باشد.

جهت استفاده از این شرط-کنش ها روش های مختلفی می تواند بکار گرفته شود .یکی از این روشها ، اضافه کردن آن، به مجموعه[p] است که در بخش بعدبا استفاده از الگوریتم پیشنهادی که روی چندین نمونه مساله ANIMATپیاده سازی شده است، بیان شده است .

۳.۴. اضافه کردن شرط-کنش مؤثر به مجموعه [p]

یکی از روش های استفاده از شرط-کنشهایی که باعث حرکت محرک می شود، اضافه کردن آن‌ها به‌عنوان یک دسته بند جدید به مجموعه [p] است. پس از شناسایی شرط- کنش یک دسته بند جدید ایجاد می شود. متغیرهای fitness، prediction، error و ... مربوط به دسته بند جدید با مقادیر پیش فرض اولیه مقداردهی می شود. متغیر action آن با کنش به‌دست‌آمده مقداردهی می شود. شرط به‌دست‌آمده ابتدا با استفاده از شیء cover برخی از بیت‌های آن با احتمال مشخصی برابر با (#) قرار می گیرد و سپس در متغیر شرط آن دسته بند ذخیره می شود. پس از مقداردهی تمام متغیرهای دسته بند جدید، این دسته بند به مجموعه [p] اضافه می شود. در مرحله بعد تعداد دسته بندهای موجود در [p] بررسی می شوند که از تعداد مجاز بیشتر نشده باشند. در این صورت با توجه به استراتژی مورد استفاده در XCS (ACCURACY-BASED) دسته بند اضافی انتخاب و حذف می شود.

۴.۴. الگوریتم روش پیشنهادی

الگوریتم ۱ شبه کد روش پیشنهادی را نشان می‌دهد. ورودی های این الگوریتم flag_repeat_condition، previous_condition، previous_action، current_input و action است. متغیر flag_repeat_condition جهت مشخص کردن condition های تکراری است. مقدار اولیه این متغیر برابر با ۱- است. متغیرهای previous_condition و previous_action به ترتیب condition و کنش مرحله قبل محرک را نگهداری می‌کنند.

متغیرهای current_input و action به ترتیب برای نگهداری condition و کنش مرحله جاری استفاده می‌شود.

در اولین مرحله از حرکت محرک شرط خط ۱ چک می‌شود و چون برقرار نیست به خط ۱۵ برنامه می‌رود. در اینجا مقدار متغیر flag_repeat_condition برابر با 0 شده و دو متغیر previous_condition و متغیر previous_action با مقادیر condition ورودی جاری(خط ۱۷) و کنش آن (خط ۱۸) مقداردهی می‌شوند. در صورتی که falg_repeat_condition بزرگتر مساوی 0 باشد، بدین معنی است که این مرجله از حرکت محرک اولین مرحله آن نیست. بنابراین باید شرط جاری محرک با قبلی آن بررسی شود(خط ۲). در صورتی که یکسان باشد یعنی محرک ثابت بوده است. حال یک واحد به متغیر flag_repeat_condition یک واحد اضافه می‌شود(خط ۳). ولی در صورتی که یکسان نباشند، باید بررسی شود که آیا تعداد مرحله-هایی که شرط محرک یکسان بوده بیش از 0 است یا نه(خط ۴). در صورتی که این تعداد حداقل ۱ باشد بدین معنی که کنش جاری باعث حرکت محرک شده است. بنابراین باید این شرط و کنش به عنوان یک دسته‌بند جدید به مجموعه [p] اضافه شود. (خطوط ۵ تا ۹) سپس با فراخوانی تابع delete_classifier() حداکثر تعداد دسته‌بند در مجموعه [p] چک می‌شود و در صورت نیاز جهت حذف دسته‌بند از الگوریتم‌های مرتبط با XCS استفاده می‌شود. در آخر متغیر flag_repeat_condition برابر با 0 جهت مرحله بعد می‌شود.

Input:	flag_repeat_condition = -1 previous_condition previous_action current_input action
Output:	new condition, new action ;
1:	if (flag_repeat_condition >= 0){
2:	if (previous_condition ==
3:	current_input){
4:	flag_repeat_condition ++;
5:	}else if (flag_repeat_condition > 0)
6:	{
7:	t_classifier classifier;
8:	
9:	classifier.cover(previous_condition);
10:	classifier.action= previous_action;
11:	init_classifier(classifier, false);
12:	insert_classifier(classifier);
13:	delete_classifier();
14:	flag_repeat_condition = 0;
15:	}
16:	previous_condition = current_input;
17:	previous_action = action;
18:	}else{ // first step
19:	flag_repeat_condition = 0;
	previous_condition = current_input;
	previous_action = action;}
	new condition=condition
	new action=action

الگوریتم ۱. شبه کد روش پیشنهادی

۵. پیاده‌سازی و ارزیابی روش پیشنهادی

برای انجام شبیه سازی های مناسب در این مقاله، از نرم افزار C++ در محیط لینوکس استفاده شده است. در آزمایشهای انجام گرفته شده، از هشت نقشه استاندارد که معمولا به عنوان محیط های استاندارد مساله هزار تو مطرح می شود، استفاده شده است. درتمام این هشت مساله (Woods1،Littman57، Maze7،MazeF4،Woods101،Maze10، Woods101_0.5 و Woods102) بصورت مساله، رسیدن به غذا بعنوان هدف، با خانه (F) ، مانع با خانه (T) و محل ها ی خالی برای حرکت محرک (ANIMAT) به عنوان راهرو با محل های خالی یا شماره گذاری شده نمایش داده شده است.

در ادامه در این بخش به بررسی نتایج حاصل از پیاده‌سازی روش ارائه‌شده با XCS می پردازیم. روش پیشنهادی روی چندین نقشه متفاوت از لحاظ پیچیدگی محیطی بررسی شده اند. این آزمایشات روی ۱۰۰۰۰ مسئله اجرا شده است. همچنین نتایج بدست آمده میانگین ۱۰ بار اجرا می‌باشد. معیارهای ارزیابی، تعداد مسائل رسیده شده به غذا در بین ۱۰۰۰۰ مسئله است و میانگین تعداد

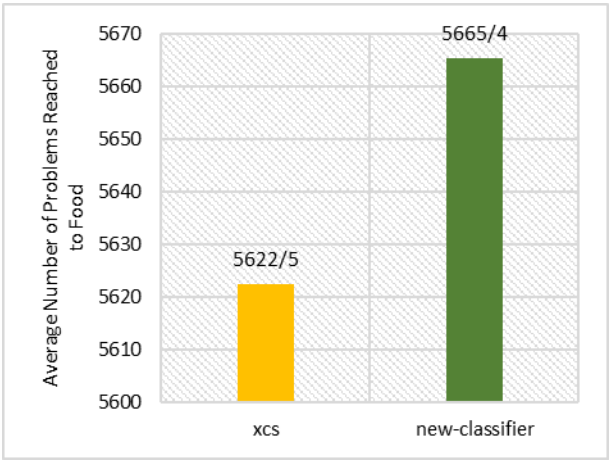
مراحل مسائل رسیده به غذا است، به عبارتیANIMAT (از مسئله‌هایی که منجر به رسیدن به غذا شده است)، بصورت میانگین از زمان شروع حرکت محرک در چند مرحله پیموده شده است.

۵.۱. [p]=800 – Woods1

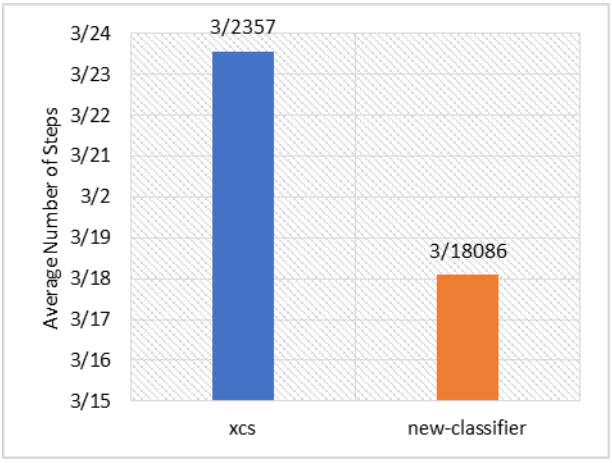
در نقشه woods1 حداکثر تعداد دسته‌بند ها در مجموعه [p] برابر با ۸۰۰ است. همچنین حداکثر تعداد مراحل تا رسیدن به غذا ۵ مرحله است. شکل ۴ نقشه Woods1 را نشان می‌دهد. نمودارهای شکل ۵ نتایج پیاده‌سازی شده روی این نقشه را نشان می‌دهد.

	T	T	F	
	T	T	T	
	T	T	T	

شکل ۴. نقشه Woods1



شکل ۵.الف. میانگین تعداد مسائل رسیده به غذا



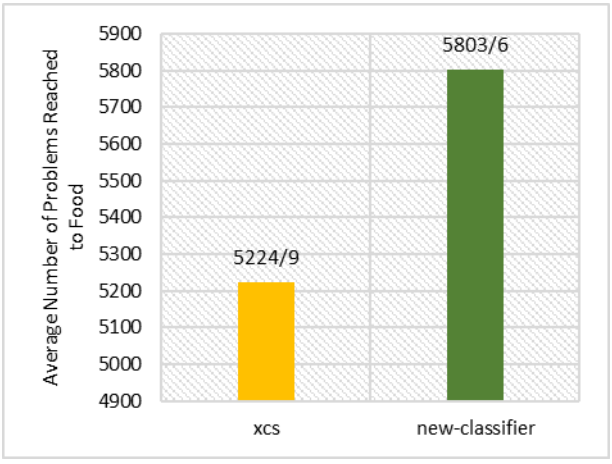
شکل ۵.ب. میانگین تعداد مراحل محرک در ۱۰۰۰۰ مسئله

۵.۲. [p]=1600 -Littman57

در نقشه Littman57 حداکثر تعداد دسته‌بند ها در مجموعه [p] برابر با ۱۶۰۰ است. همچنین حداکثر تعداد مراحل تا رسیدن به غذا ۲۰ مرحله است. شکل ۶ نقشه Littman57 و شکل ۷ نتایج حاصل از پیاده سازی روش پیشنهادی و XCS روی این نقشه را نشان می‌دهند.

T	T	T	T	T	T	T	T	T	T	T	T	T
T	4	5	2a	3a	2b	3b	2c	6	7	8	9	T
T	T	T	1a	T	1b	T	1c	T	F	T	T	T
T	T	T	T	T	T	T	T	T	T	T	T	T

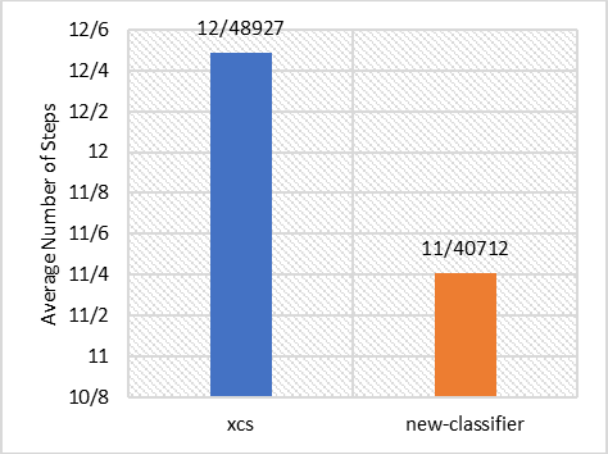
شکل ۶. نقشه Littman57



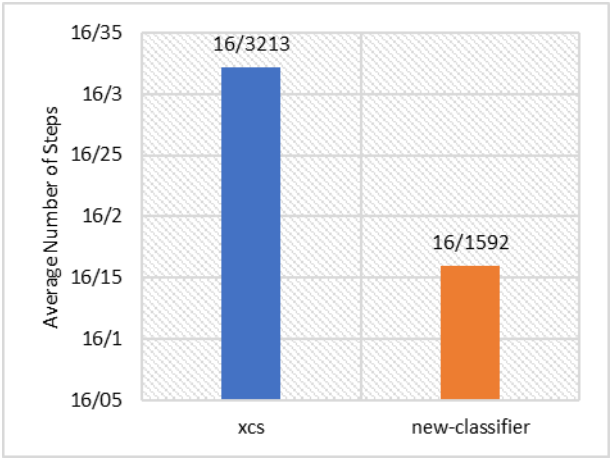
شکل ۷.الف. میانگین تعداد مسائل رسیده به غذا

۵.۳. [p]=1600 -Maze7

در نقشه Maze7 حداکثر تعداد دسته بند ها در مجموعه [p] برابر با ۱۶۰۰ است. همچنین حداکثر تعداد مراحل تا رسیدن به غذا ۲۰ مرحله است. نقشه Maze7 در شکل ۸ نشان داده شده است. نمودارهای شکل ۹ نتایج پیاده‌سازی روی این نقشه را نشان می‌دهند.



شکل ۷. ب. میانگین تعداد مراحل محرک



شکل ۹. ب. میانگین تعداد مراحل محرک

۵. ۴. MazeF4 - [p]=1600

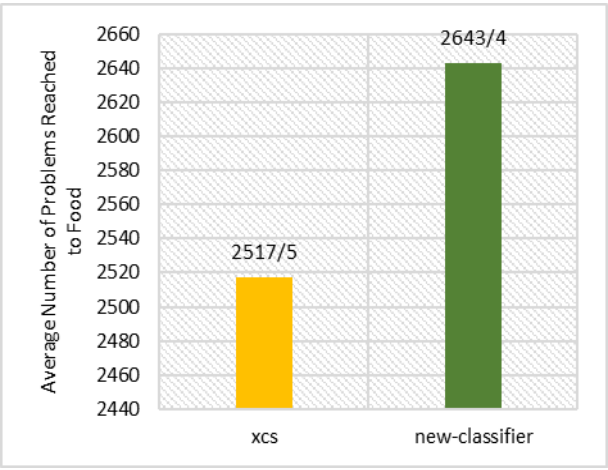
در نقشه MazeF4 حداکثر تعداد دسته‌بند ها در مجموعه [p] برابر با ۱۶۰۰ است. همچنین حداکثر تعداد مراحل تا رسیدن به غذا ۲۰ مرحله است. شکل ۱۰ نقشه MazeF4 و نمودارهای شکل ۱۱ نتایج پیاده‌سازی را نشان می‌دهند.

T	T	T	T	T	T	T
T	2	3	4			T
T	5	T	6			T
T	1a	T	1b			T
T	7	T	8			T
T	F	T	T	T		T
T	T	T	T	T	T	T

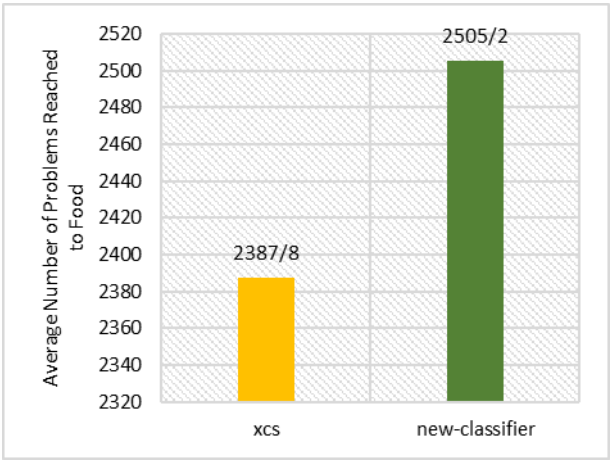
شکل ۸. نقشه Maze7

T	T	T	T	T	T	T
T			1a		F	T
T		T	T	T	T	T
T			1b		T	T
T		T	T	T	T	T
T	T	T	T	T	T	T

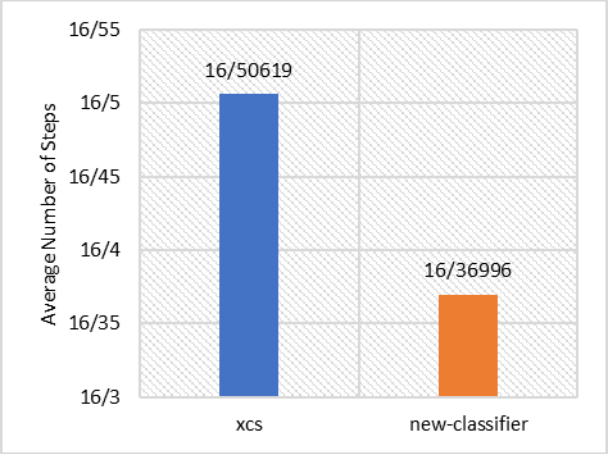
شکل ۱۰. نقشه MazeF4



شکل ۹. الف. میانگین تعداد مسائل رسیده به غذا



شکل ۱۱.الف. میانگین تعداد مسائل رسیده به غذا



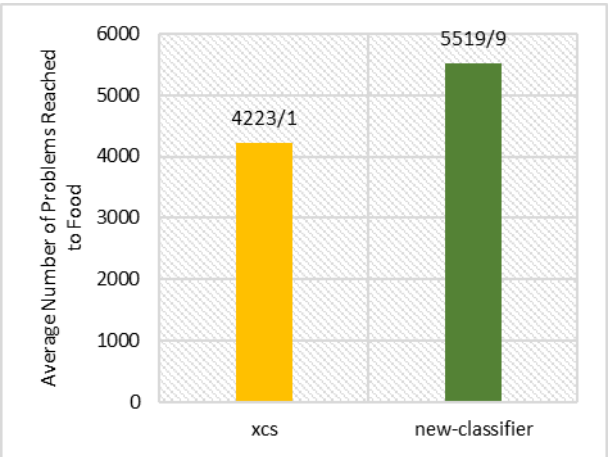
شکل ۱۱- ب. میانگین تعداد مراحل محرک

۵. ۵ Woods101 [p]=8004

در نقشه Woods101 حداکثر تعداد دسته‌بند ها در مجموعه [p] برابر با ۸۰۰ است. همچنین حداکثر تعداد مراحل تا رسیدن به غذا ۲۰ مرحله است. این نقشه در شکل ۱۲ به تصویر کشیده شده است. همچنین نتایج پیاده‌سازی روی این نقشه در شکل ۱۳ نشان داده شده است.

T	T	T	T	T	T	T
T	3	1a	4	1b	5	T
T	6	T	7	T	8	T
T	2a	T	F	T	2b	T
T	T	T	T	T	T	T

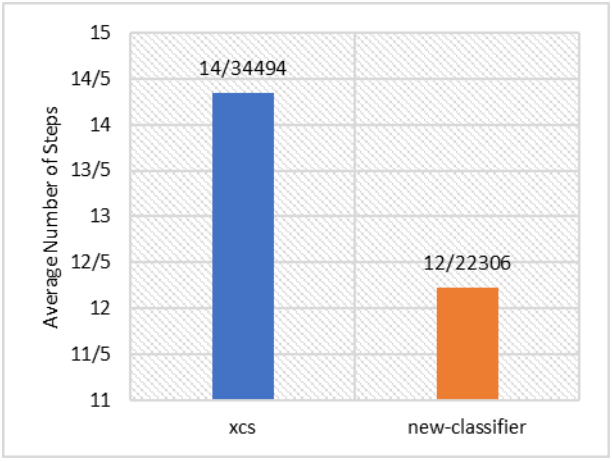
شکل ۱۲. نقشه Woods101



شکل ۱۳- الف. میانگین تعداد مسائل رسیده به غذا

۶. ۵ Maze10 -[p]=2800

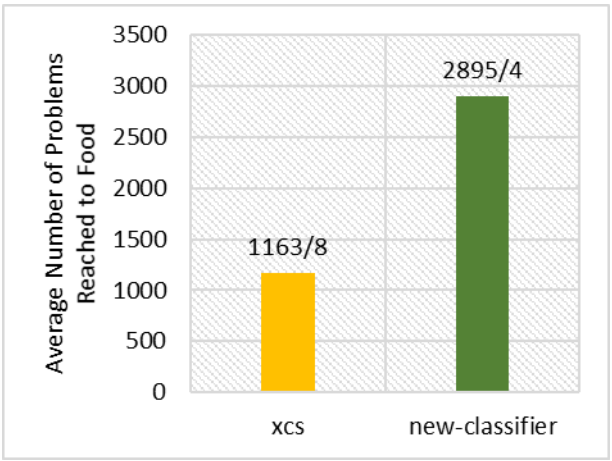
در نقشه Maze10 حداکثر تعداد دسته‌بند ها در مجموعه [p] برابر با ۲۸۰۰ است. همچنین حداکثر تعداد مراحل تا رسیدن به غذا ۲۰ مرحله است. شکل ۱۴ نقشه Maze10 و شکل ۱۵ نتایج پیاده سازی روی این نقشه را نشان می‌دهند.



شکل ۱۳.ب. میانگین تعداد مراحل محرک

T	T	T	T	T	T	T	T	T
T	6	1a	2a	1b	2b	1c	7	T
T	8	T	3a	T	3b	T	9	T
T	4a	T	10	T	4b	T	4c	T
T	5a	T	F	T	5b	T	5c	T
T	T	T	T	T	T	T	T	T

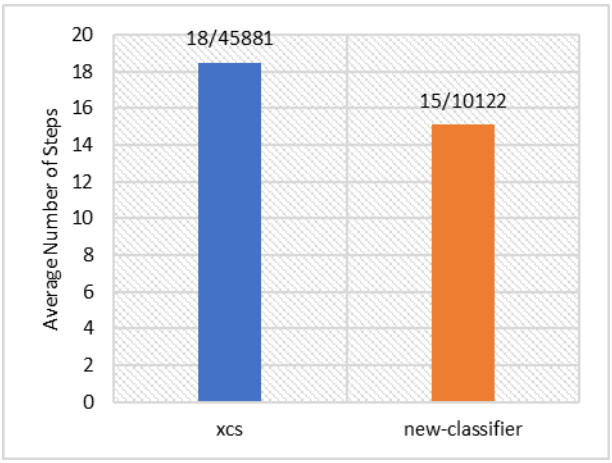
شکل ۱۴. نقشه Maze10



شکل ۱۵. الف. میانگین تعداد مسائل رسیده به غذا

۷. ۵ Woods101_0.5 - [p]=2400

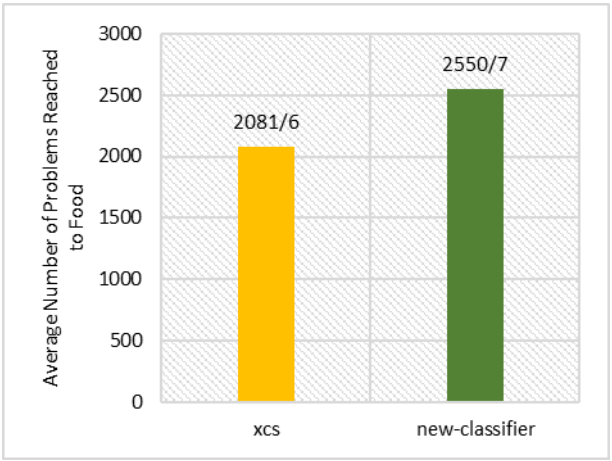
در نقشه Woods101_0.5 حداکثر تعداد دسته‌بند ها در مجموعه [p] برابر با ۲۴۰۰ است. همچنین حداکثر تعداد مراحل تا رسیدن به غذا ۲۰ مرحله است. شکل۱۶ نقشه Woods101_0.5 را نشان می‌دهد. شکل ۱۷ نتایج پیاده سازی روی این نقشه را بصورت نمودار الف و ب نشان می‌دهد.



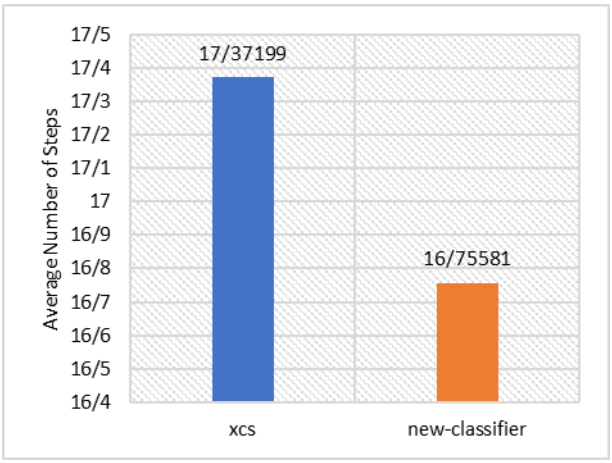
شکل ۱۵.ب. میانگین تعداد مراحل محرک

T	T	T	T	T	T	T
T		T	F	T		T
T		T		T		T
T	T	1a	T	1b	T	T
T		T		T		T
T	T	T	T	T	T	T
T		T		T		T
T	T	1c	T	1d	T	T
T		T		T		T
T		T	F	T		T
T	T	T	T	T	T	T

شکل ۱۶. نقشه Woods101_0.5



شکل ۱۷.الف. میانگین تعداد مسائل رسیده به غذا



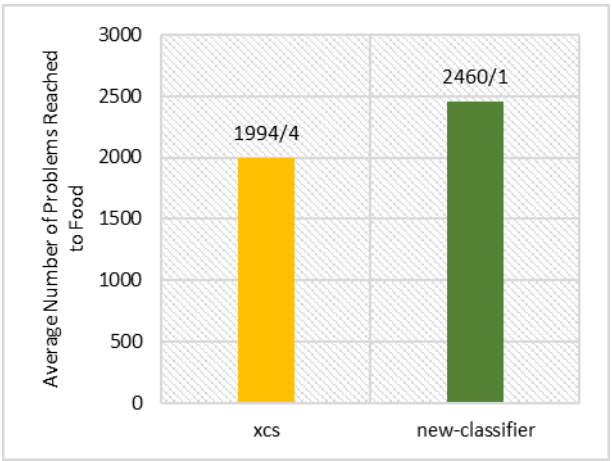
شکل ۱۷. ب. میانگین تعداد مراحل محرک

۵.۸. Woods102 -[p]=2800

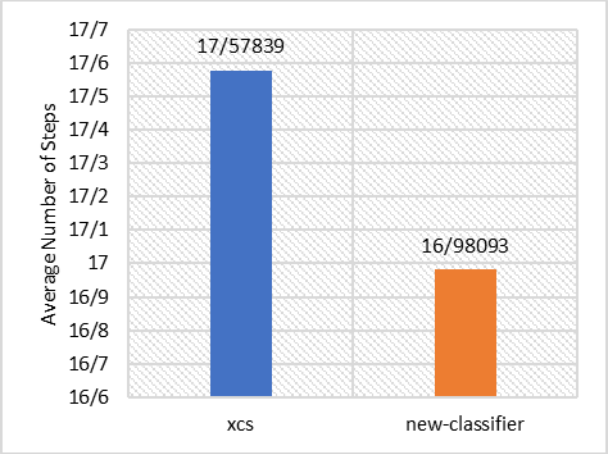
در نقشه Woods102 حداکثر تعداد دسته‌بند ها در مجموعه [p] برابر با ۲۸۰۰ است. همچنین حداکثر تعداد مراحل تا رسیدن به غذا ۲۰ مرحله است. نقشه Woods102 در شکل ۱۸ نشان داده شده است. همچنین نتایج پیاده‌سازی بصورت نمودار در شکل ۱۹ به تصویر کشیده شده است.

T	T	T	T	T	T	T
T		T	F	T		T
T		T		T		T
T		1a	2a	1b		T
T		T		T		T
T	T	T	T	T	T	T
T		T		T		T
T		1c	2b	1d		T
T		T		T		T
T		T	F	T		T
T	T	T	T	T	T	T

شکل ۱۸. نقشه Woods102



شکل ۱۹. الف. میانگین تعداد مسائل رسیده به غذا



شکل ۱۹. ب. میانگین تعداد مراحل محرک

با توجه به نتایج بدست آمده از پیاده‌سازی روش پیشنهادی بهینه-سازی حرکت محرک روی ۸ نقشه مختلف از نظر پیچیدگی محیط و در مقایسه آن با XCS، روش پیشنهادی باعث بهبود عملکرد محرک شده است. یکی از معیارهای ارزیابی بررسی شده میانگین تعداد مسائل رسیده به غذا در بین ۱۰۰۰۰ مسئله است. با بررسی این معیار ارزیابی، مشاهده م یشود که در تمامی نقشه ها تعداد مسائلی که محرک با روش پیشنهادی موفق به رسیدن غذا شده است بیش از مسائل مربوط به XCS است. در نقشه Maze7 که پیچیدگی چندانی ندارد روش پیشنهادی ۲۶۴۳ مسئله به غذا رسیده‌اند اما در روش XCS ۲۵۱۷ مسئله به غذا رسیده است. بنابراین روش پیشنهادی در این نقشه ساده ۱۲۶ مسئله را بیشتر به غذا رسانده است. در شکل ۱۳ نقشه Woods101 که نسبت به نقشه Maze7 پیچیده تر است، در XCS تعداد ۴۲۲۳ مسئله به غذا رسیده است در حالی که در روش پیشنهادی ۵۵۱۹ مسئله به غذا رسیده است. به عبارتی ۱۲۹۶ مسئله در روش پیشنهادی بیشتر از روش XCS به غذا رسیده‌اند. در نقشه woods101_0.5 در روش XCS ۲۰۸۱ مسئله و در روش پیشنهادی ۲۵۵۰ مسئله از بین ۱۰۰۰۰ مسئله به غذا رسیده‌اند. در این نقشه روش XCS ۴۶۹ مسئله کمتری نسبت به روش پیشنهادی به غذا رسانده است. با توجه به این نتایج می‌توان گفت که روش پیشنهادی اضافه کردن دسته‌بند جدید به مجموعه [p] باعث افزایش مسایل رسیده به غذا در نقشه‌های مختلف از نظر پیچیدگی شده است. بنابراین با توجه به این معیار ارزیابی عملکرد محرک بهبود یافته است.

معیار دوم ارزیابی که در این پژوهش بررسی شده است میانگین تعداد مراحل محرک تا رسیدن به غذا می‌باشد. در دنیای واقعی با توجه به اینکه محرک دارای محدودیت انرژی است، باید هر چه سریعتر در تعداد مراحل کم به هدف خود که در اینجا غذا است برسد. با توجه به نتایج بدست آمده از نمودارهای این معیار ارزیابی، روش پیشنهادی در تمام نقشه‌های پیاده‌سازی شده نسبت به روش XCS سریعتر به غذا رسیده است. به عنوان مثال در نقشه Woods101 میانگین تعداد مراحلی که محرک به غذا رسیده است در روش پیشنهادی ۱۲,۲۲ و در روش XCS، ۱۴,۳۴ است. تعداد مراحل محرک در حدود دو واحد کاهش داشته است. در نقشه Maze10 میانگین تعداد مراحل محرک در روش XCS ، ۱۸,۴۶ و در روش پیشنهادی ۱۵,۱ است. این تفاوت در تمام نقشه ها قابل مشاهده است. این نتایج دلیلی بر دقیقتر شدن حرکت محرک از زمان شروع به سمت هدف (غذا)است.

۶ . نتیجه‌گیری

کاربرد سیستم‌های طبقه‌بندی کننده یادگیر در محیطهای واقعی نیازمند به استفاده از دسته‌بندهای دقیق است. معماریهای موجود برای سیستم‌های طبقه‌بند یادگیر موفقیت ناچیزی در محیط‌های واقعی همچون هزارتوها دارند. در چنین محیط‌هایی که نمودهایی از محیط‌های واقعی مانند محیطهای امداد و نجات هستند، به دلیل ناقص بودن قوانین موجود در دسته‌بند در برخی حالات، محرک با برخورد به موانع منجر به اتمام انرژی آن می‌شود. آنچه که بعنوان دو چالش در مسائل هزارتو مطرح شد عبارتند از پیدا کردن راه حلی مناسب که به افزایش تعداد مسائلی که منجر به هدف می شود منجرشود. همچنین یافتن راهکاری برای کاهش تعداد مراحل رسیدن به هدف می باشد. رسیدن به هر دوی این چالش ها به افزایش کارایی در سیستمهای مبتنی یر یادگیری منجر می گردد. بدین جهت نیاز به یک راه حل بهینه، برای انتخاب دسته‌بندهای مناسب برای محیط‌های واقعی که به رسیدن سریعتر محرک به هدف کمک می کند نیاز می باشد. در این مقاله یک روش طبقه بندی کننده یادگیر که بر اساس XCS و شناسایی کنش‌های مناسب برای هر موقعیت است ارائه شد. در این روش در هر مرحله موقعیت فعلی محرک با موقعیت قبلی محرک بررسی می‌شود. در صورتی که محرک نسبت به موقعیت قبلی خود جابه جا شده بود، ورودی دریافت شده از سنسور و کنش اعمال شده به محیط به عنوان یک دسته‌بند جدید در مجموعه دسته‌بندهای سیستم طبقه-بند اضافه می‌شود. بنابراین این الگوریتم بدنبال حرکت محرک و رسیدن سریعتر به هدف است. با توجه به روش پیشنهادی، محرک در هر مسئله نسبت به مسئله قبلی خود هوشمندانه تر رفتار می کند؛ زیرا دسته‌بندهای موجود در سیستم طبقه بند بهینه تر شده و کمتر به موانع برخورد می‌کند. پیاده‌سازی روش پیشنهادی و اعمال آن بر روی تعدادی مساله هزارتوی معروف، نتایج قابل توجهی را در مقایسه با با نتایج معماری XCS نشان داد که موید کارایی روش پیشنهادی از منظر افزایش تعداد موفقیت‌ها در آزمایشهای متفاوت و همچنین کاهش قابل توجه در تعداد مراحل لازم در رسیدن به اهداف است.

مراجع

[1] Lanzi, Pier L. "Learning classifier systems: from foundations to applications.", No. 1813. Springer Science & Business Media, 2000.

[2] J. Holland, L. Booker, M. Colombetti, M. Dorigo, D. Goldberg, S. Forrest, et al., "What Is a Learning Classifier System?," in Learning Classifier Systems. vol. 1813, P. Lanzi ,W. Stolzmann, and S. Wilson, Eds., ed: Springer Berlin Heidelberg, 2000, pp. 3-32.

[3] Risser-Maroix, Olivier, and Benjamin Chamand. "What can we Learn by Predicting Accuracy?." *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2023.

[4] E. Bernad, Mansilla, and J. M. Garrell-Guiu, "Accuracy-based learning classifier systems: models, analysis and applications to classification tasks," Evol. Comput., vol. 11, pp. 209-238, 2003.

[5] J. H. Holmes, P. L. Lanzi, W. Stolzmann, and S.W. Wilson, "Learning classifier systems: New models, successful applications," Information Processing Letters, vol. 82, pp. 23-30, 2002.

[6] M. Shariat Panahi, A. Karkhaneh Yousefi, and M. Khorshidi, "Combining accuracy and success-rate to improve the performance of eXtended Classifier System (XCS) for data-mining and control applications," Engineering Applications of Artificial Intelligence, vol. 26, pp. 1930-1935, 2013.

[7] Irfan, Muhammad, et al. "Enhancing learning classifier systems through convolutional autoencoder to classify underwater images." *Soft Computing* 25.15 (2021): 10423-10440.

[8] Irfan, Muhammad, et al. "Knowledge extraction and retention based continual learning by using convolutional autoencoder-based learning classifier system. " *Information Sciences* 591 (2022): 287-305.

[9] Kato, Jefferson Satoshi, and Adriana Sbicca. "Bounded Rationality, Group Formation and the Emergence of Trust: An Agent-Based Economic Model." *Computational Economics* (2021): 1-29.

[10] Liu, Yi. *Learning Classifier Systems for Understanding Patterns in Data*. Diss. Open Access Te Herenga Waka-Victoria University of Wellington, 2021.

[11] Hansmeier, Tim, and Marco Platzner. "An experimental comparison of explore/exploit strategies for the learning classifier system XCS." *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 2021.

[12] Guendouzi, Wassila, and Abdelmadjid Boukra. "A new differential evolution algorithm for cooperative fuzzy rule mining: application to anomaly detection." *Evolutionary Intelligence* (2021): 1-12.

[13] Hochberger, Christian, Lars Bauer, and Thilo Pionteck. *Architecture of Computing Systems*. Springer International Publishing, 2021.

[14] Büttner, Johannes, and Sebastian Von Mammen. "Training a Reinforcement Learning Agent based on XCS in a Competitive Snake Environment." *2021 IEEE Conference on Games (CoG)*. IEEE, 2021.

[15] D. Mellor, "A Learning Classifier System Approach to Relational Reinforcement Learning," in Learning Classifier Systems. vol. 4998, J. Bacardit, E. Bernadó-Mansilla, M. Butz, T. Kovacs, X. Llorà, and K. Takadama, Eds., ed: Springer Berlin Heidelberg, 2008, pp. 169-188.

[16] P. Wawrzynski and A. K. Tanwani, "Autonomous reinforcement learning with experience replay," Neural Netw, vol. 41, pp. 156-67, 2013

[17] Z. Zang, D. Li, J. Wang, and D. Xia, "Learning classifier system with average reward reinforcement learning," Knowledge-Based Systems, vol. 40, pp. 58-71, 2013.

[18] M. Studley and L. Bull, "X-TCS: accuracy-based learning classifier system robotics ",in Evolutionary Computation, 2005. The 2005 IEEE Congress on, 2005, pp. 2099-2106 Vol. 3.

[19] M. V. Butz and David E. Goldberg, "Generalized state values in an anticipatory learning classifier system." Anticipatory behavior in adaptive learning systems. Springer, Berlin, Heidelberg, 2003. 282-301

[20] M. V. Butz, T. Kovacs, P. L. Lanzi, and S. W. Wilson, "Toward a theory of generalization and learning in XCS," Evolutionary Computation, IEEE Transactions on, vol. 8, pp. 28-46, 2004.

[21] P. Gérard and O. Sigaud, "YACS: Combining Dynamic Programming with Generalization in Classifier Systems," in Advances in Learning Classifier Systems. vol. 1996

[22] P. Luca Lanzi, W. Stolzmann, and S. Wilson, Eds., ed: Springer Berlin Heidelberg, 2001, pp. 52-69.

ed: Springer Berlin Heidelberg, 2011, pp. 371-382.

[34] Z. Zang, D. Li, and J. Wang, "Learning classifier systems with memory condition to solve non-Markov problems," Soft Computing, vol .19,pp. 1679-1699, 2015/06/01 2015.

[35] S. W. Wilson and D. E. Goldberg, "A Critical Review of Classifier Systems," presented at the Proceedings of the 3rd International Conference on Genetic Algorithms, 1989.

[36] Deshpande, Himani S., and Leena Ragha. "A hybrid random forest-based feature selection model using mutual information and F-score for preterm birth classification." *International Journal of Medical Engineering and Informatics* 15.1 (2023): 84-96.

[37] L. B. Booker, "Intelligent behavior as an adaptation to the task environment," University of Michigan, 1982.

[38] L. B. Booker, "Improving the Performance of Genetic Algorithms in Classifier Systems," presented at the Proceedings of the 1st International Conference on Genetic Algorithms, 1985

[39] L. B. Booker, "Classifier systems that learn internal world models," Mach. Lang., vol. 3, pp. 161-192, 1988.

[40] L. B. Booker, "Triggered Rule Discovery in Classifier Systems," presented at the Proceedings of the 3rd International Conference on Genetic Algorithms, 1989.

[41] S. W. Wilson, "Zcs: A zeroth level classifier system," Evol. Comput., vol. 2, pp. 1-18, 1994.

[42] Zador, Anthony, et al. "Catalyzing next-generation artificial intelligence through neuroai." *Nature communications* 14.1 (2023): 1597.

[43] Kaloop, Mosbeh R., et al. "International Roughness Index prediction for flexible pavements using novel machine learning techniques." *Engineering Applications of Artificial Intelligence* 122 (2023): 106007.

[44] S. W. Wilson, "Classifiers that approximate functions," vol. 1, pp. 211-234, 2002.

[45] Śmierzchała, Łukasz, Norbert Kozłowski, and Olgierd Unold. "Anticipatory Classifier System with Episode-based Experience Replay." *IEEE Access* (2023).

[46] L. Bull, "Two Simple Learning Classifier Systems," in Foundations of Learning Classifier Systems. vol. 183, L. Bull and T. Kovacs, Eds., ed :Springer Berlin Heidelberg, 2005, pp. 63-89.

[23] P. L. Lanzi, "An analysis of generalization in the xcs classifier system," Evol. Comput., vol. 7, pp. 125-149, 1999.

[24] P. L. Lanzi, D. Loiacono, S. W. Wilson, and D.E. Goldberg, "Generalization in the XCSF Classifier System: Analysis, Improvement, and Extension," Evol. Comput., vol. 15, pp. 133-168, 2007.

[25] M. Iqbal, W. Browne, and M. Zhang, "XCSR with Computed Continuous Action," in AI 2012: Advances in Artificial Intelligence. vol. 769 ,۱M. Thielscher and D. Zhang, Eds., ed: Springer Berlin Heidelberg, 2012, pp. 350-361.

[26] M. Iqbal, W. N. Browne, and Z. Mengjie, "Reusing Building Blocks of Extracted Knowledge to Solve Complex, Large-Scale Boolean Problems," Evolutionary Computation ,IEEE Transactions on, vol. 18, pp. 465-480, 2014.

[27] G. Bezerra, T. Barra, L. de Castro, and F. Von Zuben, "Adaptive Radius Immune Algorithm for Data Clustering," in Artificial Immune Systems. vol. 3627, C. Jacob, M. Pilat, P. Bentley, and J. Timmis, Eds., ed: Springer Berlin Heidelberg, 202905, pp. 290-303.

[28] H.-P. Cheng, Z.-S. Lin, H.-F. Hsiao, and M.-L. Tseng, "Designing an Artificial Immune System-Based Machine Learning Classifier for Medical Diagnosis," in Information Computing and Applications .vol. 6377, R. Zhu, Y. Zhang, B. Liu, and C. Liu, Eds., ed: Springer Berlin Heidelberg, 2010, pp. 333-341.

[29] J. D. Farmer, N. H. Packard, and A. S. Perelson, "The immune system, adaptation, and machine learning," Physica D: Nonlinear Phenomena, vol. 22 ,pp. 187-204, 1986.

[30] F. Freschi and M. Repetto, "Multiobjective optimization by a modified artificial immune system algorithm," presented at the Proceedings of the 4th international conference on Artificial Immune Systems, Banff, Alberta, Canada, 2005.

[31] J. Timmis, P. Andrews, N. Owens, and E. Clark, "An interdisciplinary perspective on artificial immune systems," Evolutionary Intelligence, vol. 1, pp. 5-26, 2008/03/01 2008.

[32] P. Vargas, L. de Castro, and F. Von Zuben, "Mapping Artificial Immune Systems into Learning Classifier Systems," in Learning Classifier Systems. vol. 2661, P. Lanzi, W. Stolzmann, and S. Wilson, Eds., ed: Springer Berlin Heidelberg, 2003, pp. 163-186.

[33] L. Bull, "Towards a Mapping of Modern AIS and LCS," in Artificial Immune Systems. vol. 6825, P. Liò, G. Nicosia, and T. Stibor, Eds.,

[47] L. Bull, "A brief history of learning classifier systems: from CS-1 to XCS and its variants," Evolutionary Intelligence, pp. 1-16, 2015/01/29 2015.

[48] A. Hamzeh, S. Hashemi, A. Sami, and A. Rahmani, "A Recursive Classifier System for Partially Observable Environments," Fundam. Inform., vol. 97, pp. 15-40, 2009.

[49] A. Hamzeh and A. Rahmani, "A New Architecture for Learning Classifier Systems to Solve POMDP Problems," Fundam. Inform., vol. 84, pp. 329-351,2008

[50] R. Preen and L. Bull, "Discrete and fuzzy dynamical genetic programming in the XCSF learning classifier system," Soft Computing, vol. 18, pp. 153-167, 2014/01/01 2014.