Journal of Soft Computing and Information Technology (JSCIT)
Babol Noshirvani University of Technology, Babol, Iran
Journal Homepage: jscit.nit.ac.ir
DOI: 10.22034/jscit.2026.565981.2202
Volume 15, Number 3, Autumn 2026, pp. 87-98
Received: 12/22/2025, Revised: 07/07/2026, Accepted: 07/18/2026

Combining Bat and Ant Colony Metaheuristic Algorithms for Service Quality Preservation in Adaptive Mixed-Criticality Systems

Farzaneh gholami[1], Vahid Haghighstdoost[2*]
1- Department of Computer Engineering, Shahed University, Tehran, Iran.
2*- Department of Computer Engineering, Shahed University, Tehran, Iran.
[1]farzaneh.gholami@shahed.ac.ir, [2*]haghighatdoost@shahed.ac.ir

Corresponding author's address: Vahid Haghighatdoost, Faculty of Computer Engineering and Technology, Shahed University, Tehran, Iran.

Abstract- Scheduling in uniprocessor real-time systems is a fundamental challenge to guarantee the timing constraints of tasks. Traditional methods such as Rate Monotonic and Earliest DeadLine First lack sufficient efficiency in complex systems. As an effective solution, Adaptive Mixed-Criticality (AMC) systems enable mode switches between different criticality levels, ensuring that high-criticality tasks do not miss their DeadLines, even under heavy workloads. However, determining the optimal priority assignment for tasks in these systems is an NP-Hard problem. In this paper, to overcome this challenge, a hybrid method based on the Bat Algorithm (BA) and Ant Colony Optimization (ACO) is proposed. In this approach, the output of the Bat Algorithm is utilized as the initial pheromone value for the Ant Colony algorithm to increase the convergence speed. Simulation results demonstrate that the proposed method, compared to baseline approaches and previous research in this field, increases the schedulability ratio of Low-criticality tasks by 91% and reduces the response time compared to machine learning algorithms, which indicates the higher efficiency of the hybrid algorithm in managing mixed-criticality systems.

Keywords- Adaptive Mixed-Criticality Systems, Fixed-Priority Scheduling, Bat Algorithm, Ant Colony Optimization, Response Time Analysis.

مجله علمی رایانش نرم و فناوری اطلاعات
دانشگاه صنعتی نوشیروانی بابل
صفحه مجله: jscit.nit.ac.ir
شناسه دیجیتال: 10.22034/jscit.2026.565981.2202
جلد ۱۵، شماره ۳، پاییز ۱۴۰۵، صفحه ۸۷-۹۸
دریافت: ۱۴۰۴/۱۰/۰۱، بازنگری: ۱۴۰۵/۰۴/۱۶، پذیرش: ۱۴۰۵/۰۴/۲۷

ترکیب الگوریتم‌های فراابتکاری خفاش و کلونی مورچگان برای حفظ کیفیت سرویس در سیستم‌های بحرانی مختلط تطبیقی

فرزانه غلامی[۱]، وحید حقیقت دوست*[۲]
۱- دانشکده فنی و مهندسی، دانشگاه شاهد، تهران، ایران.
*۲- دانشکده فنی و مهندسی، دانشگاه شاهد، تهران، ایران.
[1]farzaneh.gholami@shahed.ac.ir, [2*]haghighatdoost@shahed.ac.ir

* نشانی نویسنده مسئول: وحید حقیقت دوست، تهران، دانشگاه شاهد، دانشکده فنی و مهندسی.

چکیده – مسئله زمانبندی در سیستم‌های بلادرنگ تک پردازنده برای تضمین رعایت محدودیت‌های زمانی وظایف، چالشی اساسی است. روش‌های سنتی مانند نرخ یکنواخت و زودترین ضرب العجل در سیستم‌های پیچیده کارایی لازم را ندارند. سیستم‌های بحرانی مختلط تطبیقی (AMC) به عنوان یک راهکار موثر، امکان تغییر حالت سیستم بین سطوح بحرانی مختلف را فراهم می‌کند تا اطمینان حاصل شود وظایف با اهمیت بالا حتی در شرایط بار کاری زیاد، مهلت اجرای خود را از دست نمی‌دهند. با این حال، تعیین اولویت بهینه برای وظایف در این سیستم‌ها یک مسئله NP-Hard است. در این مقاله، برای غلبه بر این چالش، روشی ترکیبی مبتنی بر الگوریتم خفاش (BA)و بهینه‌سازی کلونی مورچگان (ACO) ارائه شده است. در این روش، خروجی الگوریتم خفاش به عنوان مقدار اولیه فرومون برای الگوریتم مورچگان استفاده می‌شود تا سرعت همگرایی افزایش یابد. نتایج شبیه‌سازی نشان می‌دهد که روش پیشنهادی در مقایسه با روش‌های پایه و تحقیقات پیشین در این زمینه، موجب افزایش نسبت وظایف با سطح بحرانی کم اولویت به میزان ۹۱ درصد و کاهش زمان پاسخگویی نسبت به الگوریتم‌های یادگیری ماشین شده است که نشان‌دهنده کارایی بالاتر الگوریتم ترکیبی در مدیریت سیستم‌های بحرانی مختلط است.

واژه‌های کلیدی: سیستم‌های بحرانی مختلط تطبیقی، زمان‌بندی اولویت ثابت، الگوریتم خفاش، الگوریتم کلونی مورچگان، تحلیل زمان پاسخ.

۱- مقدمه

امروزه سیستم‌های تعبیه شده[۱] نقش حیاتی در کنترل فرآیندهای حساس در صنایع مختلف ایفا می‌کنند. در بسیاری از این کاربردها، صحت عملکرد سیستم تنها به درستی منطقی خروجی وابسته نیست، بلکه به زمان تولید آن خروجی نیز بستگی دارد. به این دسته از سیستم‌ها، سیستم‌های بلادرنگ[۲] گفته می‌شود [۱]. چالش اصلی در طراحی این سیستم‌ها، مدیریت منابع محدود پردازشی به گونه‌ای است که تمام وظایف بتوانند پیش از مهلت اجرای خود به پایان برسند.

در سال‌های اخیر، نیاز به ادغام چندین نرم‌افزار با درجات اهمیت متفاوت بر روی یک پردازنده واحد افزایش یافته است. برای مثال، در سیستم‌های کنترلی پیشرفته، ممکن است وظایف بسیار حساسی مانند کنترل ترمز یا پایداری در کنار وظایف کم‌اهمیت‌تری مانند مانیتورینگ دما یا ثبت رویدادها اجرا شوند. این سیستم‌ها که به عنوان سیستم‌های بحرانی مختلط شناخته می‌شوند، [۲] مدل‌های زمان‌بندی سنتی را با چالش مواجه کرده‌اند. در روش‌های کلاسیک، معمولاً بدبینانه‌ترین حالت اجرا برای تمام وظایف در نظر گرفته

می‌شد که منجر به هدررفت شدید ظرفیت پردازنده می‌گردید.

پروتکل زمان‌بندی بحرانی مختلط تطبیقی[۳] (AMC) به‌عنوان یک رویکرد نوین برای حل این مسئله معرفی شده است [۳]. در این پروتکل، سیستم در شرایط عادی تمامی وظایف را اجرا می‌کند، اما در صورت افزایش بار پردازشی و وقوع اضافه‌بار، با تغییر حالت، منابع پردازشی به وظایف با سطح بحرانی بالا اختصاص یافته و اجرای برخی از وظایف با سطح بحرانی پایین محدود یا متوقف می‌شود تا تضمین‌های زمانی وظایف بحرانی حفظ گردد. با وجود این، کارایی پروتکل AMC تا حد زیادی به نحوه تعیین اولویت وظایف وابسته است و انتخاب نامناسب اولویت‌ها می‌تواند موجب افزایش حذف وظایف کم اولویت[۴] (LCT) و کاهش بهره‌وری منابع پردازشی شود.

در سال‌های اخیر، الگوریتم‌های فراابتکاری به دلیل توانایی بالا در جستجوی فضای پاسخ و یافتن جواب‌های نزدیک به بهینه، برای حل مسائل پیچیده زمان‌بندی مورد توجه قرار گرفته‌اند [۵,۴] . این الگوریتم‌ها می‌توانند بدون نیاز به جستجوی کامل فضای حالت، راه‌حل‌های مناسبی برای مسائل NP-Hard ارائه دهند.

۲- تحقیقات پیشین

سیستم‌های AMC با هدف تضمین اجرای وظایف با سطح بحرانی بالا (HI) و در عین حال افزایش کیفیت سرویس وظایف با سطح بحرانی پایین(LO) ، طی سال‌های اخیر به یکی از مهم‌ترین موضوعات پژوهشی در حوزه سیستم‌های بلادرنگ تبدیل شده‌اند. با وجود پیشرفت‌های صورت‌گرفته، چالش اصلی همچنان دستیابی به روشی است که ضمن افزایش قابلیت زمان‌بندی، نرخ موفقیت اجرای وظایف کم اولویت را نیز بدون ایجاد سربار محاسباتی قابل توجه در زمان اجرا افزایش دهد.

یکی از پژوهش‌های پایه در این حوزه توسط Baruah ، Burns و Davis ارائه شد که در آن تحلیل زمان پاسخ برای سیستم‌های AMC مبتنی بر اولویت ثابت معرفی شد [۳]. در این روش، سیستم در حالت عادی تمامی وظایف را اجرا می‌کند، اما در صورتی که زمان اجرای یک وظیفه اولویت بالا از بودجه زمانی تعیین‌شده آن در سطح پایین، فراتر رود، سیستم وارد حالت بحرانی شده و بخشی از وظایف کم اولویت حذف یا متوقف می‌شوند تا تضمین اجرای وظایف اولویت بالا[۵] (HCT) حفظ شود. نتایج تحلیل‌های ارائه‌شده نشان داد که این روش نسبت به مدل‌های اولیه زمانبندی بحرانی مختلط که تطبیقی نبودند، قابلیت زمان‌بندی بالاتری فراهم می‌کند. با این وجود، تعیین اولویت وظایف همچنان بر اساس قواعد ثابت انجام شده و هیچ سازوکار هوشمندی برای یافتن ترتیب بهینه اولویت‌ها ارائه نشده است.

در ادامه، Burns و Davis تحلیل زمان پاسخ AMC را برای سیستم‌هایی با مهلت‌های دلخواه (Arbitrary Deadlines) توسعه دادند [۶]. در این پژوهش، مدل تحلیلی به گونه‌ای گسترش یافت که بتواند علاوه بر مهلت‌های دلخواه، اطلاعات مربوط به زمان اجرای واقعی وظایف را نیز در تحلیل لحاظ کند. ارزیابی این روش با استفاده از مجموعه وظایف استاندارد و تحلیل ریاضی انجام شد و نتایج نشان داد که نسبت به نسخه کلاسیک AMC ، قابلیت زمان‌بندی سیستم بهبود می‌یابد. با این حال، این پژوهش صرفاً بر توسعه مدل تحلیلی متمرکز بوده و مکانیزمی برای بهینه‌سازی ترتیب اولویت وظایف یا کاهش حذف وظایف کم اولویت ارائه نمی‌کند.

به‌منظور افزایش کیفیت سرویس وظایف LO ، Liu و همکاران روشی مبتنی بر کاهش تدریجی کیفیت سرویس (Degraded Service) ارائه کردند [۷]. ایده اصلی این پژوهش آن است که به‌جای حذف کامل وظایف کم اولویت پس از تغییر حالت سیستم، کیفیت سرویس این وظایف به‌صورت کنترل‌شده کاهش یابد تا تعداد بیشتری از آن‌ها همچنان امکان اجرا داشته باشند. ارزیابی این روش از طریق شبیه‌سازی و مقایسه با سیاست استاندارد AMC انجام شد و نتایج نشان داد که کاهش تدریجی کیفیت سرویس می‌تواند نرخ موفقیت اجرای وظایف کم اولویت را افزایش دهد، بدون آنکه تضمین اجرای وظایف اولویت بالا نقض شود. با وجود این، فرآیند تعیین اولویت وظایف در این روش همچنان مبتنی بر قواعد از پیش تعیین‌شده است و از هیچ الگوریتم جستجوی هوشمند برای یافتن بهترین ترتیب اولویت استفاده نمی‌شود.

در سال‌های اخیر، استفاده از یادگیری ماشین نیز برای بهبود عملکرد سیستم‌های AMC مورد توجه قرار گرفته است.Mendes و همکاران روشی مبتنی بر یادگیری تقویتی عمیق (Deep Reinforcement Learning) ارائه کردند که در آن عامل یادگیرنده پس از آموزش، در زمان اجرا بودجه زمانی وظایف کم اولویت را به‌صورت پویا تنظیم می‌کند تا احتمال وقوع تغییر حالت کاهش یافته و تعداد بیشتری از وظایف کم اولویت بدون نقض تضمین اجرای وظایف اولویت بالا اجرا شوند [۸]. نتایج ارزیابی این پژوهش نشان داد که روش پیشنهادی نسبت به AMC استاندارد کیفیت سرویس بالاتری ایجاد می‌کند، اما تصمیم‌گیری عامل یادگیری در زمان اجرا مستلزم اجرای مدل آموزش‌دیده و انجام محاسبات اضافی است که سربار پردازشی و مصرف حافظه را افزایش می‌دهد و استفاده از آن را در برخی سیستم‌های بلادرنگ با منابع محدود دشوار می‌سازد.

بررسی مطالعات فوق نشان می‌دهد که روش‌های موجود هر یک بخشی از چالش زمان‌بندی سیستم‌های بحرانی مختلط را برطرف کرده‌اند. روش‌های تحلیلی، تضمین‌های زمانی قابل اعتمادی ارائه می‌کنند، اما فاقد سازوکار بهینه‌سازی هوشمند برای تعیین ترتیب اولویت وظایف هستند. از سوی دیگر، روش‌های مبتنی بر کاهش کیفیت سرویس و یادگیری ماشین، اگرچه نرخ موفقیت اجرای وظایف کم اولویت را افزایش می‌دهند، اما یا ترتیب اولویت وظایف را بهینه نمی‌کنند و یا با افزایش سربار محاسباتی در زمان اجرا همراه هستند. ازاین‌رو، همچنان نیاز به روشی وجود دارد که فرآیند تعیین اولویت وظایف را به‌صورت آفلاین انجام داده و بدون ایجاد سربار ناشی از الگوریتم بهینه‌سازی در زمان اجرا، عملکرد سیستم را بهبود دهد.

در پژوهش حاضر، این هدف با استفاده از ترکیب الگوریتم خفاش (BA) و الگوریتم کلونی مورچگان (ACO) دنبال شده است. در روش پیشنهادی، تعیین ترتیب اولویت وظایف به‌طور کامل در مرحله طراحی انجام می‌شود و در زمان اجرای سیستم تنها اولویت‌های بهینه به‌دست‌آمده در کنار تحلیل زمان پاسخ AMC-rtb مورد استفاده قرار می‌گیرند. بنابراین، برخلاف روش‌های مبتنی بر یادگیری ماشین، الگوریتم پیشنهادی سربار ناشی از فرآیند بهینه‌سازی را در زمان اجرا به سیستم تحمیل نمی‌کند.

هدف اصلی و نوآوری این مقاله، تعیین یک چیدمان اولویت بهینه در فاز طراحی با استفاده از الگوریتم ترکیبی هوشمند (BA-ACO) است. این رویکرد مبتنی بر مفروضات مدل استاندارد ارائه‌شده توسط دیویس و برنز [۳] بنا شده است. در شرایط اضافه‌بار، سیستم پیشنهادی به صورت کاملاً ایمن تغییر حالت می‌دهد تا هیچ‌یک از وظایف حیاتی مهلت اجرای خود را از دست ندهند؛ اما نوآوری کار در این است که اولویت‌بندی بهینه‌شده در فاز طراحی باعث می‌شود حتی پس از وقوع شرایط بحرانی، همچنان شانس اجرای وظایف با درجه بحرانی پایین به حداکثر برسد و از قطع کامل آن‌ها جلوگیری شود، بدون اینکه هیچ سربار محاسباتی اضافه‌ای در زمان اجرای سیستم تحمیل گردد.

در ادامه، این مقاله بدین شرح سازمان‌دهی شده است: در بخش ۳، مدل سیستم، نمادگذاری‌ها و تعاریف اولیه مسئله تشریح می‌شود. بخش ۴ به معرفی جزئیات روش پیشنهادی و نحوه ترکیب الگوریتم‌های هوشمند برای تعیین اولویت‌ها اختصاص دارد. در بخش ۵، تنظیمات شبیه‌سازی، سناریوهای آزمایش و تحلیل نتایج ارزیابی کارایی روش پیشنهادی در مقایسه با سایر روش‌ها ارائه می‌گردد و در نهایت، بخش ۶ به نتیجه‌گیری و پیشنهاداتی برای کارهای آتی می‌پردازد.

۳- مدل سیستم و تعاریف

در این پژوهش، مسئله زمان‌بندی بر روی یک سیستم تک‌پردازنده[۶] در نظر گرفته شده است که باید مجموعه‌ای از n وظیفه مستقل را مدیریت کند. این مجموعه با نماد $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$ نشان داده می‌شود. در سیستم‌های بلادرنگ، وظایف از نظر الگوی ورود به سه دسته دوره‌ای، اسپورادیک[۷] و غیردوره‌ای تقسیم می‌شوند. در مدل اسپورادیک، زمان دقیق وقوع رخدادها مشخص نیست (تصادفی است)، اما یک محدودیت فیزیکی وجود دارد که تضمین می‌کند بین دو رخداد متوالی، حداقل زمان T فاصله وجود دارد [۶]. برای مثال، یک سیستم ترمز در خودرو را در نظر بگیرید؛ زمان ترمز گرفتن راننده مشخص نیست، اما راننده نمی‌تواند در یک ثانیه صد بار پدال را فشار دهد. این "حداقل فاصله زمانی" اجازه می‌دهد تا بتوان بدترین حالت بار کاری سیستم را تحلیل و تضمین کرد.

مدل ورود وظایف به سیستم از نوع اسپورادیک فرض شده است که مدلی استاندارد و واقع‌گرایانه برای سیستم‌های بلادرنگ تعبیه‌شده محسوب می‌شود.

۳-۱- پارامترهای وظایف

هر وظیفه τ_i در سیستم‌های با بحرانی مختلط تطبیقی، با یک تاپل ۵ عضوی به صورت رابطه (۱) تعریف می‌شود [۶]:

$$\tau_i = (L_i, C_i(LO), C_i(HI), T_i, D_i) \qquad (۱)$$

که در آن پارامترها عبارتند از:

- L_i (سطح اهمیت): بیانگر میزان حیاتی بودن وظیفه است و می‌تواند یکی از دو مقدار $\{LO, HI\}$ را داشته باشد. وظایف HI (حیاتی) آن‌هایی هستند که شکست در اجرای آن‌ها می‌تواند خطرات جانی یا مالی شدید داشته باشد، در حالی که وظایف LO (کم‌اهمیت) وظایف جانبی سیستم هستند.
- پارامترهای $C_i(LO)$ **و** $C_i(HI)$: این دو پارامتر بدترین زمان اجرای (WCET) وظیفه را نشان می‌دهند.
- $C_i(LO)$: برآوردی واقع‌بینانه و استاندارد از زمان اجراست که برای همه وظایف تعریف می‌شود.
- $C_i(HI)$: برآوردی بسیار بدبینانه و محافظه‌کارانه است که تنها برای وظایف حیاتی (HI) مقدار دارد. بر اساس مدل

بنیادی سیستم‌های با بحرانی مختلط [۲]، برای تضمین ایمنی همواره فرض می‌شود $C_i(HI) \geq C_i(LO)$.

- T_i (حداقل فاصله زمانی): حداقل فاصله زمانی بین دو درخواست متوالی برای اجرای وظیفه τ_i است (که معادل دوره تناوب در نظر گرفته می‌شود).
- D_i (مهلت زمانی): حداکثر زمانی که وظیفه پس از ورود فرصت دارد تا اجرای خود را تکمیل کند. در این مدل، "مهلت زمانی" بطور ضمنی برابر با "فاصله زمانی اجرا" فرض شده است، به این معنی که مهلت زمانی برابر با دوره تناوب است ($D_i = T_i$).

۳-۲- حالتهای عملیاتی و تغییر حالت

سیستم پیشنهادی بر اساس پروتکل «بحرانی مختلط تطبیقی تطبیقی» (AMC) عمل می‌کند که دارای دو حالت عملیاتی مجزا است:

۱- حالت عادی (LO-Mode): سیستم در ابتدای کار در این حالت قرار دارد. فرض بر این است که رفتار تمام وظایف مطابق با برآورد $C_i(LO)$ است. در این حالت، تمام وظایف (اعم از LOو HI) باید پیش از سررسید مهلت زمانی خود تکمیل شوند.

۲- حالت بحرانی (HI-Mode): اگر در حین اجرا، حتی یکی از وظایف حیاتی (HI) به اندازه $C_i(LO)$ اجرا شد اما هنوز تکمیل نشده بود، سیستم تشخیص می‌دهد که رفتار سیستم غیرعادی است. در این لحظه، یک تغییر حالت[۸] رخ می‌دهد. برای تضمین انجام این تغییر حالت در شرایط سربار[۹]، الگوریتم پیشنهادی از مکانیزم پایش نرم‌افزاری موردنیاز پروتکل استاندارد AMC برای تشخیص تغییر حالت سیستم استفاده می‌کند. این مکانیزم در زمان اجرا، زمان جاری سیستم و وضعیت اجرای وظایف را پایش می‌کند تا در صورت وقوع شرایط اضافه‌بار، تغییر حالت مطابق پروتکل AMC انجام شود. لازم به ذکر است که این مکانیزم بخشی از ساختار استاندارد پروتکل AMCاست و توسط روش پیشنهادی ایجاد نشده است. از آنجا که فرآیند بهینه‌سازی اولویت وظایف به‌صورت آفلاین و در مرحله طراحی انجام می‌شود، روش پیشنهادی تنها سربار ذاتی مکانیزم پایش AMC را دارد و سربار ناشی از اجرای الگوریتم‌های بهینه‌سازی در زمان اجرا به سیستم تحمیل نمی‌شود. چنانچه زمان اجرای سپری شده از آستانه تعیین شده ($C_i(LO)$) عبور کند و وظیفه تکمیل نشده باشد، سیستم بلافاصله و بدون تأخیر، فرمان تغییر حالت را صادر می‌کند. در حالت بحرانی، برای آزاد کردن ظرفیت پردازنده:

- تمام وظایف با اهمیت پایین (LO) بلافاصله متوقف و حذف می‌شوند و یا کیفیت آن‌ها تنزل می‌یابد.
- به وظایف حیاتی (HI) اجازه داده می‌شود تا سقف زمان $C_i(HI)$ اجرا شوند.

ساده‌ترین سیاست که به عنوان زمان‌بندی ایستا[۱۰] (SMC) شناخته می‌شود، تمام وظایف با اهمیت پایین را به محض تغییر حالت سیستم، متوقف می‌کند [۳]. این امر معادل آن است که زمان اجرای تخصیص یافته به این وظایف در حالت بحرانی صفر در نظر گرفته شود.

۳-۳- تحلیل زمان پاسخ (AMC-rtb)

مسئله اصلی در این پژوهش، یافتن یک ترتیب اولویت ثابت[۱۱] برای وظایف است به طوری که سیستم در هر دو حالت عملیاتی «قابل زمان‌بندی» باشد. برای اثبات صحت زمان‌بندی، از تحلیل زمان پاسخ [۱۲](RTA) و محدوده زمان پاسخ [۱۳] (RTB) استفاده می‌شود. لازم به ذکر است که روابط ریاضی حاکم بر روشAMC ، اگرچه ابتدا در مرجع پایه [۳] معرفی شدند، اما صحت و کارایی آن‌ها در مطالعات مروری جامع ژورنالی نظیر [۹] که به عنوان استاندارد تحلیل سیستم‌های بحرانی مختلط تأیید و تثبیت شده است. زمان پاسخ R_i، طولانی‌ترین فاصله زمانی بین لحظه ورود وظیفه تا لحظه پایان آن است.

الف) **در حالت عادی:** زمان پاسخ برای هر وظیفه τ_i طبق رابطه (۲) محاسبه می‌شود که در آن $hp(i)$ مجموعه وظایف با اولویت بالاتر از τ_i است [۳]:

$$R_i^{LO} = C_i(LO) + \sum_{j \in hp(i)} \left\lceil \frac{R_i^{LO}}{T_j} \right\rceil C_j(LO) \qquad (۲)$$

ب) **در حالت بحرانی (تحلیل AMC-rtb):** این تحلیل پیچیده‌تر است. برای تضمین ایمنی وظایف حیاتی در حالت بحرانی، باید بدترین سناریوی تداخل را در نظر گرفت. طبق روش AMC-rtb، زمان پاسخ بحرانی (R_i^{HI}) برابر است با مجموع زمان اجرای خود وظیفه در حالت بحرانی، به اضافه تداخل وظایف حیاتی دیگر (hpH) و تداخل محدود وظایف کم‌اهمیت (hpL) [۳] که این محاسبات در رابطه‌ی (۳) بیان شده است:

$$R_i^{HI} = C_i(HI) + \sum_{j \in hpH(i)} \left\lceil \frac{R_i^{HI}}{T_j} \right\rceil C_j(HI) + \sum_{k \in hpL(i)} \left\lceil \frac{R_i^{LO}}{T_k} \right\rceil C_k(LO) \quad (۳)$$

نوآوری اصلی رابطه (۳) در قسمت سوم آن است. از آنجا که با ورود سیستم به حالت بحرانی، وظایف LO حذف می‌شوند، تداخل آن‌ها نمی‌تواند تا بی‌نهایت ادامه یابد. بنابراین، تداخل وظایف کم‌اهمیت تنها تا لحظه R_i^{LO} (زمانی که سیستم هنوز در حالت عادی فرض می‌شده) محاسبه می‌شود. اگر زمان پاسخ محاسبه شده در هر دو رابطه کمتر از مهلت زمانی (D_i) باشد، سیستم قابل زمان‌بندی است.

۴- روش پیشنهادی

روش پیشنهادی از یک ساختار دو مرحله‌ای تشکیل شده است. هدف، یافتن بردار جایگشت $P = [p_1, p_2, \ldots, p_n]$ است که ترتیب اولویت وظایف را مشخص می‌کند به طوری که تابع هزینه (تعداد شکست ضرب‌الاجل‌ها) کمینه شود. در سیستم‌های AMC به دلیل وجود حالت‌های کاری متفاوت (حالت نرمال و حالت بحرانی) و پویایی تغییر وضعیت، یافتن بهترین چیدمان اولویت که همزمان ایمنی وظایف حیاتی را تضمین کند و هم نرخ موفقیت وظایف کم‌اهمیت را بیشینه سازد، یک مسئله NP-Hard محسوب می‌شود [۳]. در چنین فضای جستجوی وسیعی (که شامل $n!$ حالت مختلف برای n وظیفه است)، روش‌های جستجوی تصادفی ساده ممکن است بی‌ثبات باشند؛ بدین معنا که در یک اجرا ممکن است به یک جواب بسیار خوب برسند و در اجرای دیگر در دام بهینه‌های محلی با کیفیت پایین گرفتار شوند.

برای غلبه بر این عدم قطعیت و تضمین کیفیت نهایی زمان‌بندی، یک رویکرد ترکیبی نوین پیشنهاد شده است. منطق حاکم بر این روش بر پایه تقسیم وظایف بین دو الگوریتم قدرتمند خفاش (Bat Algorithm - BA) و کلونی مورچگان (Ant Colony Optimization - ACO) استوار است:

۴-۱- فاز پویش با الگوریتم خفاش

در فاز پویش[۱۴] ، ابتدا الگوریتم خفاش با قابلیت جستجوی سریع و سراسری خود، فضای جایگشت‌های اولویت را کاوش می‌کند. این الگوریتم به دلیل ماهیت تصادفی خود، ممکن است در اجراهای مختلف کیفیت‌های متفاوتی ارائه دهد (به عنوان مثال در یک سناریو نرخ موفقیت ۳۵٪ و در سناریوی دیگر ۹۱٪ را کشف کند).

الگوریتم خفاش یک الگوریتم فرابتکاری مبتنی بر هوش ازدحامی است که برای اولین بار توسط یانگ در سال ۲۰۱۰ معرفی شد [۵]. این الگوریتم از رفتار شگفت‌انگیز خفاش‌های کوچک در ردیابی طعمه و اجتناب از موانع در تاریکی مطلق، با استفاده از سیستم پژواک‌یابی[۱۵] الهام گرفته شده است.

الف) اصول بیولوژیک و مدل‌سازی

خفاش‌ها پالس‌های صوتی با فرکانس متغیر منتشر می‌کنند و با گوش دادن به بازتاب آن، فاصله و موقعیت طعمه را تشخیص می‌دهند. در مدل ریاضی، هر خفاش i دارای موقعیت مکانی x_i، که نماینده یک راه‌حل است و سرعت v_i در یک فضای d- بعدی است. رفتار حرکتی خفاش‌ها با سه مؤلفه اصلی کنترل می‌شود: فرکانس (f_i)، سرعت (v_i) و موقعیت (x_i).

معادلات حرکت در هر تکرار t در رابطه‌ی (۴) بیان شده است:

$$\begin{aligned} f_i &= f_{min} + (f_{max} - f_{min})\beta \\ v_i^t &= v_i^{t-1} + (x_i^{t-1} - x_*)f_i \\ x_i^t &= x_i^{t-1} + v_i^t \end{aligned} \quad (۴)$$

که در آن $\beta \in [0,1]$ یک بردار تصادفی با توزیع یکنواخت است، x_* بهترین موقعیت جهانی یافت شده تا کنون (Best Global Solution) است و بازه $[f_{min}, f_{max}]$ محدوده فرکانسی پالس‌ها را تعیین می‌کند. این معادلات نشان می‌دهند که حرکت خفاش‌ها ترکیبی از حرکت پیشین و جذب شدن به سمت بهترین خفاش گروه (که بر اساس پالس ارسالی و کمترین فاصله تا طعمه شناسایی میشود) است.

ب) مدیریت پویش و بهره‌برداری

یکی از نقاط قوت الگوریتم خفاش، مکانیزم کنترل پویای تعادل بین پویش (جستجوی نواحی جدید) و بهره‌برداری (جستجوی دقیق اطراف جواب فعلی) است. این کار از طریق تغییرات «بلندی صدا» (A_i) و «نرخ انتشار پالس» (r_i) انجام می‌شود.

زمانی که خفاش به طعمه (جواب بهینه) نزدیک می‌شود، به طور طبیعی بلندی صدای خود را کاهش داده و نرخ ارسال پالس را افزایش می‌دهد تا دقت ردیابی بالا رود. مدل ریاضی این رفتار به صورت رابطه‌ی (۵) تعریف می‌شود:

$$A_i^{t+1} = \alpha A_i^t, r_i^{t+1} = r_i^0[1 - \exp(-\gamma t)] \quad (۵)$$

که در آن $0 < \alpha < 1$ ضریب کاهش صدا و $\gamma > 0$ ضریب افزایش نرخ پالس است. این مکانیزم باعث می‌شود الگوریتم در مراحل ابتدایی بر جستجوی سراسری تمرکز کند و با گذشت زمان به صورت

خودکار به سمت جستجوی محلی دقیق همگرا شود.

تابع برازش ($Fitness$) در این مرحله بر اساس فرمول‌های AMC-rtb (روابط (۲) و (۳)) محاسبه می‌شود. بهترین راه‌حل یافت شده در این فاز (Sol_{Bat}) ذخیره می‌گردد تا به عنوان ورودی اولیه فاز بعدی استفاده شود.

ج) گسسته‌سازی برای مسئله جایگشت

از آنجا که معادلات فوق در فضای پیوسته تعریف شده‌اند و مسئله تخصیص اولویت یک مسئله گسسته (جایگشت) است، در این پژوهش از تکنیک‌های نگاشت نظیر **بردار** موقعیت کوچکترین مقدار (SPV)[۱۶] یا عملگرهای جابجایی استفاده می‌شود تا موقعیت‌های پیوسته x_i به یک ترتیب اولویت معتبر تبدیل شوند.

در این مقاله برای ایجاد ارتباط سازگار میان دو الگوریتم خفاش و کلونی مورچگان، ابتدا لازم است فضای جستجوی هر دو الگوریتم به یک نمایش (Representation) مشترک نگاشت شود. در این راستا، هر راه‌حل ممکن در مسأله‌ی بهینه‌سازی به‌صورت یک بردار ویژگی $X = [x_1, x_2, \dots, x_n]$ تعریف می‌شود که هر مؤلفه‌ی آن نشان‌دهنده‌ی یک متغیر تصمیم یا پارامتر قابل تنظیم در مسأله است.

در مرحله‌ی اجرای الگوریتم خفاش (Bat Algorithm)، هر خفاش با یک بردار X_i نمایش داده می‌شود که موقعیت آن در فضای جستجو بیانگر یک راه‌حل محتمل است. در طول فرایند بهینه‌سازی، این موقعیت با استفاده از مکانیزم‌های پرواز، فرکانس، سرعت و نرخ ارسال پالس خفاش‌ها به‌روزرسانی می‌شود تا در نهایت به بهترین موقعیت، یعنی X^* برسد. این موقعیت، در واقع «کد عددی راه‌حل بهینه در فضای الگوریتم خفاش» است (فرایند Coding).

در گام بعد، یک فرایند بازکدگذاری (Decoding) انجام می‌شود تا بردار X^* مجدداً به ساختار فیزیکی یا منطقی مسأله اصلی (مانند تخصیص منابع، مسیریابی، یا انتخاب ویژگی‌ها) ترجمه شود. این بازکدگذاری موجب می‌شود که محتوای هر مؤلفه از X^*، معنای مشخص و ملموسی در فضای واقعی مسأله پیدا کند.

برای انتقال دانشِ حاصل از الگوریتم خفاش به الگوریتم کلونی مورچگان (ACO)، همین بردارهای نرمال‌شده (Normalized Vectors) به عنوان ماتریس اولیه‌ی فرومون یا مجموعه‌ی مسیرهای آغازین مورچه‌ها مورد استفاده قرار می‌گیرند. بدین ترتیب، مؤلفه‌های بردار X^* معادل شدت اولیه‌ی فرومون بر روی یال‌های متناظر در گراف مورچگان در نظر گرفته می‌شوند. به بیان دیگر، مسیرهایی که در الگوریتم خفاش منجر به مقادیر بهینه شده‌اند، در الگوریتم مورچگان با احتمال انتخاب بالاتری برای شروع جستجو همراه خواهند بود (فرایند تبدیل مجدد).

این رویکرد سبب می‌شود جستجوی مورچه‌ها در فضای راه‌حل، با آگاهی از موقعیت‌های بهینه‌ی اکتشافی خفاش‌ها آغاز گردد؛ در نتیجه نرخ همگرایی افزایش یافته و احتمال گرفتار شدن در کمینه‌های محلی به شدت کاهش می‌یابد. در نهایت، خروجی الگوریتم کلونی مورچگان نیز به‌صورت یک بردار بهینه‌ی جدید X^*_{ACO} بازنمایی شده و مجدداً به راه‌حل واقعی مسأله (Decoding نهایی) تبدیل می‌شود.

۴-۲- فاز انتفاع با الگوریتم کلونی مورچگان

در این مرحله برای انتفاع[۱۷] ، الگوریتم ACO برای بهبود دقیق جواب خفاش وارد عمل می‌شود.

الگوریتم ACO که توسط دوریگو معرفی شد، [۴] مدلی ریاضی از رفتار مورچه‌های واقعی در یافتن کوتاه‌ترین مسیر میان لانه و منبع غذا است. این الگوریتم بر پایه تعامل غیرمستقیم عامل‌ها بنا شده است. مورچه‌ها در حین حرکت، ماده‌ای شیمیایی به نام فرومون از خود بر جای می‌گذارند که شدت آن با کیفیت مسیر طی شده رابطه مستقیم دارد.

الف) ساخت گراف و قانون انتقال

در مسئله زمان‌بندی، فرآیند یافتن جواب به حرکت در یک گراف ساختاری تشبیه می‌شود که در آن گره‌ها وظایف و یال‌ها (یا موقعیت‌ها) نشان‌دهنده اولویت هستند. هر مورچه k برای انتخاب وظیفه بعدی جهت قرار دادن در اولویت j، از یک قانون انتقال احتمالی[۱۸] استفاده می‌کند. احتمال انتخاب وظیفه iبرای موقعیت jطبق رابطه زیر محاسبه می‌شود:

$$P_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha}[\eta_j]^{\beta}}{\sum_{l \in Allowed}[\tau_{il}]^{\alpha}[\eta_l]^{\beta}} \qquad (۶)$$

در این رابطه:

- $\tau_{ij}(t)$: غلظت فرومون روی یال مربوطه است (حافظه بلندمدت کلونی).
- η_i: اطلاعات ابتکاری[۱۹] یا دید حریصانه است (معمولاً معکوس دوره تناوب $\frac{1}{T_i}$ یا مهلت زمانی در نظر گرفته می‌شود).
- α و β: پارامترهایی هستند که وزن نسبی اثر فرومون و اطلاعات ابتکاری را تعیین می‌کنند.

- $\mathcal{N}_k$: مجموعه وظایفی است که هنوز توسط مورچه k انتخاب نشده‌اند.

ب) به‌روزرسانی فرومون و تبخیر

پس از اینکه تمام مورچه‌ها راه‌حل‌های خود را ساختند، مقدار فرومون در مسیرها به‌روزرسانی می‌شود. این فرآیند شامل دو بخش است: تقویت مسیرهای خوب و تبخیر فرومون[۲۰] از مسیرهای ضعیف.

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \qquad (۷)$$

پارامتر $\rho \in (0,1]$ نرخ تبخیر است که از انباشت بیش از حد فرومون و همگرایی زودرس به بهینه‌های محلی جلوگیری می‌کند. عبارت $\Delta\tau_{ij}^{k}$ مقدار فرومونی است که مورچه kبر اساس کیفیت جواب خود (مقدار تابع برازش) روی مسیر می‌ریزد. در روش پیشنهادی این پژوهش، این مکانیزم به‌روزرسانی با اطلاعات حاصل از الگوریتم خفاش ترکیب شده تا سرعت همگرایی افزایش یابد.

ج) تابع برازش و فرمول‌بندی مسئله

در الگوریتم‌های تکاملی و هوش ازدحامی، «تابع برازش[۲۱]» معیاری است که میزان شایستگی هر راه‌حل پیشنهادی را می‌سنجد. در این پژوهش، هدف نهایی یافتن یک چیدمان اولویت[۲۲] است که بتواند سیستم را در شرایط بحرانی ایمن نگه دارد و در شرایط عادی، کیفیت سرویس را حفظ کند [۱۰]. از آنجا که این مسئله دارای اهداف چندگانه با درجات اهمیت متفاوت است، تابع برازش F به صورت یک «تابع جریمه وزنی»[۲۳] تعریف شده است.

در این رویکرد، مسئله بهینه‌سازی به صورت کمینه‌سازی مقدار تابع F تعریف می‌شود، به این معنی که هرچه مقدار F برای یک بردار اولویت کمتر باشد، آن راه‌حل «برازش» یا شایستگی بیشتری دارد و به جواب بهینه نزدیک‌تر است.

- فرمول‌بندی ریاضی تابع برازش

برای هر بردار اولویت پیشنهادی $\mathbf{P}$، مقدار برازش $F(\mathbf{P})$ طبق رابطه (۸) محاسبه می‌گردد:

ساختار این رابطه بر اساس روش استاندارد مدیریت قیود به کمک توابع جریمه[۲۴] [۱۱] فرمول‌بندی شده و مؤلفه‌های آن با استناد به تعاریف ایمنی و کیفیت سرویس در سیستم‌های مخلوط بحرانی [۱۰] استخراج شده‌اند:

$$Minimize\ F(\mathbf{P}) = (w_1 \times N_{miss}^{HI}) + (w_2 \times N_{miss}^{LO}) + \Omega_{excess} \qquad (۸)$$

که در آن، اجزای فرمول به شرح زیر تعریف می‌شوند:

۱- جریمه نقض ایمنی (N_{miss}^{HI}): این پارامتر تعداد وظایف حیاتی (HI) را می‌شمارد که در تحلیل بدترین حالت (حالت بحرانی)، زمان پاسخ آن‌ها از مهلت زمانی تعیین شده (D_i) تجاوز کرده است.

$$N_{miss}^{HI} = \sum_{\tau_i \in \mathbb{H}} \mathbb{I}(R_i^{HI} > D_i) \qquad (۹)$$

۲- جریمه کاهش کیفیت سرویس (N_{miss}^{LO}): این پارامتر بیانگر تعداد وظایف غیرحیاتی (LO) است که در حالت کاری عادی، موفق به رعایت ضرب‌الاجل خود نشده‌اند.

$$N_{miss}^{LO} = \sum_{\tau_j \in \mathbb{L}} \mathbb{I}(R_j^{LO} > D_j) \qquad (۱۰)$$

۳- جریمه سربار زمانی (Ω_{excess}): این عبارت مجموع مقادیر زمانی است که وظایف از مهلت خود تجاوز کرده‌اند. نقش این عبارت، ایجاد تمایز بین راه‌حل‌هایی است که تعداد خطای یکسانی دارند اما میزان انحراف زمانی آن‌ها متفاوت است.

$$\Omega_{excess} = \sum_{\forall \tau_k} \max(0, R_k - D_k) \qquad (۱۱)$$

- تحلیل ضرایب وزنی ($\boldsymbol{w_1}$ و $\boldsymbol{w_2}$)

نکته اساسی در طراحی این تابع، انتخاب مقادیر ضرایب وزن‌دهی w_1و w_2 است تا سلسله‌مراتب اهداف سیستم بلادرنگ رعایت شود. منطق حاکم بر این وزن‌دهی به صورت زیر است:

۱- اولویت مطلق ایمنی ($w_1 \gg w_2$): ضریب w_1 با یک مقدار عددی بسیار بزرگ تنظیم می‌شود. این کار باعث می‌شود که الگوریتم به محض مواجهه با راه‌حلی که حتی یک وظیفه حیاتی را به خطر می‌اندازد، مقدار برازش F را به شدت افزایش داده و آن راه‌حل را به عنوان یک جواب نامطلوب جریمه کند.

۲- اولویت کیفیت سرویس ($w_2 > 1$): ضریب w_2 مقدار متوسطی دارد تا پس از تضمین ایمنی، الگوریتم به سمت راه‌حل‌هایی همگرا شود که کمترین تعداد حذف وظایف غیرحیاتی را داشته باشند.

۳- بهینه‌سازی نهایی: عبارت آخر (Ω_{excess}) بدون ضریب (یا با ضریب یک) عمل می‌کند تا به عنوان یک معیار دقیق‌سازی[۲۵] نهایی، زمان‌های پاسخ را کاهش دهد.

بنابراین، الگوریتم‌های خفاش و مورچگان در طول فرآیند جستجو، همواره به دنبال بردار اولویتی هستند که مقدار F را به سمت صفر

میل دهد. شبه کد ۱، مربوط به ترکیب این دو الگوریتم می‌باشد که ادامه آورده شده است.

در نهایت مجموعه وظایف مورد نظر شامل n وظیفه است که پارامترهای زمان اجرای مورد نیاز (C_i) و دوره تناوب (T_i) برای هر وظیفه به صورت دستی و از پیش تعیین شده تنظیم گردیده‌اند تا بار محاسباتی سیستم شبیه‌سازی شود. بررسی نتایج الگوریتم ها جهت نمایش زمان‌بندی پذیر بودن یا نبودن اولویت ها توسط فرمول های تحلیل زمان پاسخ انجام می‌شود، اگر قابل زمان‌بندی باشد پروتکل اجرا می‌شود و اگر قابل زمان‌بندی نباشد، خطایی تحت عنوان "با پارامتر های موجود این سیستم زمان‌بندی پذیر نیست" نمایش داده می‌شود.

در این پژوهش، به‌منظور ارزیابی عملکرد الگوریتم ترکیبی پیشنهادی (خفاش–مورچگان) و بررسی رفتار سیستم در هنگام وقوع تغییر حالت بحرانی ، یک مجموعه داده شامل ۱۰ وظیفه به‌عنوان مطالعه موردی انتخاب شده است. انتخاب این ابعاد با هدف ارائه تحلیل دقیق رفتار روش پیشنهادی و امکان نمایش شفاف نتایج شبیه‌سازی انجام شده است.

از یک سو، استفاده از تعداد محدود وظایف، نمایش گرافیکی نتایج از طریق نمودار گانت را خواناتر می‌کند و امکان بررسی دقیق رخدادهایی نظیر پیش‌دستی (Preemption)، تغییر حالت سیستم و نحوه تخصیص منابع را فراهم می‌سازد. در مقابل، افزایش تعداد وظایف موجب تراکم بیش از حد اطلاعات در نمودارها شده و تحلیل بصری رفتار سیستم را دشوار می‌کند.

از سوی دیگر، هدف این پژوهش ارزیابی اثر ترکیب الگوریتم‌های BA و ACO بر کیفیت زمان‌بندی در یک سناریوی کنترل‌شده بوده است. ازاین‌رو، تمامی الگوریتم‌های مورد مقایسه بر روی یک مجموعه وظایف یکسان و با پارامترهای ثابت اجرا شدند تا تأثیر روش پیشنهادی بر عملکرد سیستم، بدون دخالت عوامل ناشی از تغییر مشخصات مجموعه وظایف، به‌صورت منصفانه ارزیابی شود.

به‌منظور تبیین دقیق مراحل اجرای روش پیشنهادی و نمایش روند تعامل الگوریتم‌های خفاش و کلونی مورچگان در فرآیند زمان‌بندی سیستم‌های AMC، شبه‌کد الگوریتم ترکیبی پیشنهادی در ادامه ارائه شده است.

شبه کد ۱: ترکیب الگوریتم خفاش و مورچگان

Algorithm Hybrid_BA_ACO_Scheduler
Input: Task Set T, Parameters $(N_{\{bat\}}, MaxIter_{\{BA\}}, N_{\{ant\}}, MaxIter_{\{ACO\}}, \alpha, \beta, \rho)$
Output: Optimal Permutation P^*
// --- PHASE 1: BAT ALGORITHM (Exploration)
1: Initialize $N_{\{bat\}}$ bats randomly across the valid permutation space.
2: Calculate initial Fitness(Sol_i) for all bats using $EvaluateByRTA(Sol_i)$.
4: FOR $t = 1$ TO $MaxIter_{\{BA\}}$:
5: Update parameters: $A \leftarrow 0.98$, $r \leftarrow r\left(1 - e^{(-0.1t)}\right)$.
6: FOR $i = 1$ TO $N_{\{bat\}}$:
7: Generate New Position (Sol_i') based on random walk and $Sol_{\{Best\}}$ influence (using Swap operators for permutation validity).
8: $Fit' \leftarrow EvaluateByRTA(Sol_i')$.
9: IF $Fit' < Fitness(Sol_i)$:
10: Update Sol_i and $Fitness(Sol_i)$.
11: IF $Fit' < Fitness\left(Sol_{\{Best\}}\right)$:
12: Update $Sol_{\{Best\}}$.
13: END FOR
14: END FOR
15: $P_{\{BA\}} \leftarrow Sol_{\{Best\}}$
// --- PHASE 2: ACO (Exploitation & Seeding) ---
16: Initialize Pheromone Matrix $\tau(p,t)$ of size $N \times N$ (where N is number of tasks).
17: Initialize Heuristic Information $\eta(t) = \frac{1}{Period_t}$ for all tasks t.
18: **// KNOWLEDGE INJECTION **
19: FOR $p = 1$ TO N:
20: $TaskID \leftarrow P_{\{BA\}}[p]$ (Task at priority level p from BA result)
21: $\tau(p, TaskID) \leftarrow \tau(p, TaskID) + \Delta\tau_{\{BA\}}$ // $\Delta tau_{\{BA\}}$ is the bonus factor (e.g., 5.0)
22: END FOR
23: $P^* \leftarrow P_{\{BA\}}$ (Initialize global best with BA result).
24: FOR $it = 1$ TO $MaxIter_{\{ACO\}}$:
25: FOR $k = 1$ TO $N_{\{ant\}}$ (For each ant):
26: Construct Solution P_k iteratively from $p = 1$ to N:
27: Calculate Selection Probability: $$Prob(p,t) \propto \left(\tau(p,t)\right)^{\alpha} . \left(\eta(t)\right)^{\beta}$$ for all available tasks t.
28: Select Task t using Roulette Wheel selection based on $Prob$.
29: $P_k[p] \leftarrow t$. Remove t from available tasks.
30: $Fit_k \leftarrow EvaluateByRTA(P_k)$.
31: IF $Fit_k < Fit(P^*)$: $UpdateP^*$.
32: END FOR
33: // **Pheromone Update**
34: Evaporate: $\tau \leftarrow (1-\rho).\tau$.
35: Deposit: Reinforce τ based on $1/\left(1 + Fit_{k_{best}}\right)$ for the best ant of the iteration and $1/\left(1 + Fit(P^*)\right)$ for the global best ant.
36: END FOR

۵- نتایج شبیه‌سازی و ارزیابی

پیاده‌سازی و شبیه‌سازی در محیط نرم‌افزار MATLAB انجام شده

است. مجموعه وظایف (Task Set) شامل ۱۰ وظیفه با پارامترهای متفاوت (زمان اجرا، دوره تناوب و سطح اهمیت) است. تعداد ۱۰ وظیفه برای این پژوهش انتخاب شد تا بتوان تمرکز آزمایش‌ها را بر اثبات برتری مکانیزم هیبریدی در برابر الگوریتم‌های پایه BA و AMCحفظ کرد و امکان انجام تنظیم دقیق (Tuning) پارامترهای فرابتکاری را فراهم آورد. این تعداد برای نمایش توانایی الگوریتم در یافتن ترکیب بهینه اولویت‌بندی در حضور تضادهای زمان‌بندی کافی است. پارامترهای تنظیم‌شده برای الگوریتم‌های فرابتکاری به شرح زیر و مطابق با مقادیر استاندارد در مقالات الگوریتم خفاش و کلونی مورچگان انتخاب شده‌اند:

- **فاز اول** (الگوریتم خفاش - BAT): تعداد ۴۰ خفاش، ۵۰ تکرار، ضریب بلندی صدا $A = 0.95$ و نرخ پالس $r = 0.1$ برای جستجوی اکتشافی در فضای مسئله.
- **فاز دوم** (الگوریتم مورچگان - ACO): تعداد ۲۰ مورچه، ۴۰ تکرار، ضریب تبخیر فرومون $\rho = 0.1$ و ضرایب اهمیت فرومون ($\alpha = 1$) و اطلاعات ابتکاری ($\beta = 2$).

۵-۱- تحلیل همگرایی و بهبود پاسخ

در فاز نخست، الگوریتم خفاش با جستجوی سراسری، یک راه‌حل اولیه با کیفیت مناسب تولید می‌کند. سپس این راه‌حل به عنوان بذر اولیه (Seed) برای ماتریس فرومون در الگوریتم ACO استفاده می‌شود تا جستجوی محلی دقیق‌تری صورت گیرد.

همان‌طور که در خروجی‌های محاسباتی شکل ۱ مشاهده می‌شود، الگوریتم ترکیبی توانسته است مقدار تابع هزینه (Fitness) را که ترکیبی از جریمه‌های از دست دادن مهلت زمانی و تداخلات سیستم است، به شکل چشمگیری کاهش دهد.

```
Command Window
>>> Phase 1: Running Bat Algorithm (Exploration)...
   -> Initial BAT Best Fitness: 5583
   -> BAT Phase Finished. Best Fitness: 5255
   -> BAT Solution: [9 3 1 2 7 6 5 10 8 4]

>>> Phase 2: Running ACO (Improvement & Exploitation)...
    Using BAT solution to seed Pheromones.
    ACO Iter 10 Best Fit: 5255
    ACO Iter 20 Best Fit: 5255
    ACO Iter 30 Best Fit: 5255
    ACO Iter 40 Best Fit: 5255
✅ FINAL Optimized Order (Hybrid BAT+ACO): [9 3 1 2 7 6 5 10 8 4]
✅ Final Fitness: 5255
```

شکل ۱: روند همگرایی و مقادیر تابع برازش نهایی در خروجی متلب

الگوریتم خفاش به‌طور تصادفی همانند خروجی شکل ۲ ترتیب اولویت‌ها و درصد موفقیت اجرای وظایف کم اولویت را متفاوت نشان می‌دهد در بعضی از اجراها الگوریتم خفاش در بهینه های محلی گیر می‌کند و اولویتی را تولید می‌کند که لزوما اولویت بهینه نیست، بنابراین با ترکیب شدن این الگوریتم با الگوریتم کلونی مورچگان همیشه بهترین جواب از الگوریتم خفاش به عنوان ورودی الگوریتم کلونی مورچگان قرار می‌گیرد و اگر اولویت بهتری توسط الگوریتم کلونی مورچگان یافت شود جایگزین این اولویت خواهد شد.

```
Command Window
✅ Initial best fitness: 1020
✅ Final optimized order (BA+AMC): [7 9 2 1 3 6 8 10 4 5]
Final Fitness: 1020

=== Final Report ===
HI: 22/22 (100.0%)
LO: 7/23 (30.4%)
fx >>
```

شکل ۲: روند مقادیر تصادفی الگوریتم خفاش

۵-۲- تحلیل و ارزیابی نتایج شبیه‌سازی

در این بخش، کارایی روش پیشنهادی در مقایسه با دو روش دیگر، یعنی الگوریتم خفاش پایه و روش کلاسیک بحرانی ترکیبی تطبیقی (AMC)، مورد ارزیابی قرار می‌گیرد. نتایج حاصل از شبیه‌سازی در محیط متلب، در سه سطح مختلف تحلیل شده‌اند: تحلیل آماری کلان (Pie Charts)، تحلیل رفتار زمانی پردازنده (Gantt Charts) و تحلیل ریزدانه وضعیت نمونه‌ها (Task Instances).

الف) بررسی نرخ موفقیت و حالت نمونه‌ها

دقیق‌ترین ابزار برای سنجش "کیفیت سرویس" (QoS)، نمودار وضعیت نمونه‌ها در شکل ۳ است که سرنوشت تک‌تک کارهای تولید شده همراه با دوره‌ی اجرای آن‌ها را نمایش می‌دهد. خطوط سبز نشان دهنده موفقیت اجرای هر دوره در حالت عادی و خطوط قرمز نشان دهنده ی حذف آن دوره هستند.

بر اساس شکل ۳، تحلیل عمیق زیر ارائه می‌شود:

۱. پایداری وظایف حیاتی: در هر سه نمودار، تمام دوره‌های وظایف حیاتی کاملا سبز شده به این معنا که اجرای وظایف بحرانی به طور ۱۰۰ درصد تضمین شده است،دوره‌هایی که با رنگ قرمز نمایش داده شده‌اند نشان‌دهنده عملکرد صحیح مکانیزم تشخیص تغییر حالت است. یعنی سیستم متوجه بحران شده و زمان اجرای بدبینانه

(WCET-HI) را اعمال کرده، اما همچنان مهلت زمانی را پاس کرده است.

۲. تفاوت آشکار در بقای وظایف کم اولویت: در روش AMC (نمودار پایین) مشاهده می‌شود که برخی از وظایف کم اولویت در اولین دوره اجرای خود قرمز و در ادامه وظایف کم اولویت کاملاً خالی شده‌اند. این یعنی به محض ورود به حالت بحرانی، این وظایف قربانی شده‌اند، در نمودار بالا مشاهده میشود که در سیستم AMC وظیفه ی شماره ۷ در اولین دوره از اجرای خود مهلت اجرا خود را رد کرده و باعث ورود به حالت بحرانی و حذف شدن وظایف کم اولویت شده است.

در روش پیشنهادی (شکل ۳) نکته برجسته پژوهش در اینجا نمایان می‌شود، الگوریتم ترکیبی توانسته است با چینش هوشمندانه وظایف اولویت بالا در زمان‌هایی که تداخل کمتری دارند، "پنجره‌های زمانی امنی" برای عبور وظایف کم اولویت ایجاد کند.

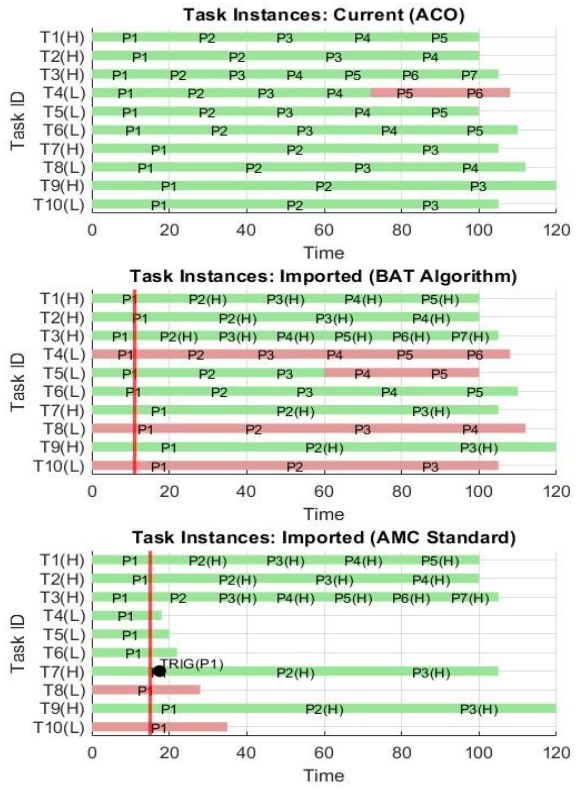
۵-۳- تحلیل زمان‌بندی و نمودار گانت

شکل ۴ نمودار گانت (Gantt Chart) حاصل از زمان‌بندی بهینه را در طول دوره شبیه‌سازی نشان می‌دهد. در این نمودار:

۱- محور افقی نشان‌دهنده زمان و محور عمودی شناسه وظایف است.

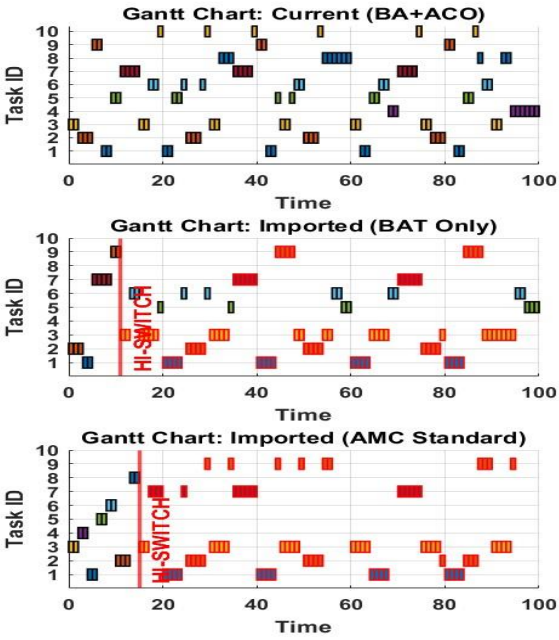
۲- بلوک‌هایی که دارای حاشیه قرمز هستند، نشان‌دهنده اجرای وظایف در حالت بحرانی می‌باشند.

۳- مطابق با منطق AMC-rtb، در لحظاتی که سیستم دچار بار کاری بیش از حد (Overload) می‌شود و احتمال از دست رفتن مهلت زمانی وظایف حیاتی وجود دارد، سیستم به حالت بحرانی سوئیچ کرده و اجرای وظایف کم اولویت را تعلیق می‌کند.



شکل ۳: نمودار مقایسه Task Instances

این نمودار رفتار دینامیک تخصیص پردازنده به وظایف را در طول ۱۰۰ واحد زمانی شبیه‌سازی نشان می‌دهد. خط قرمز عمودی در نمودارها نشان‌دهنده لحظه تغییر حالت (Mode Switch) از حالت LO به HI است.



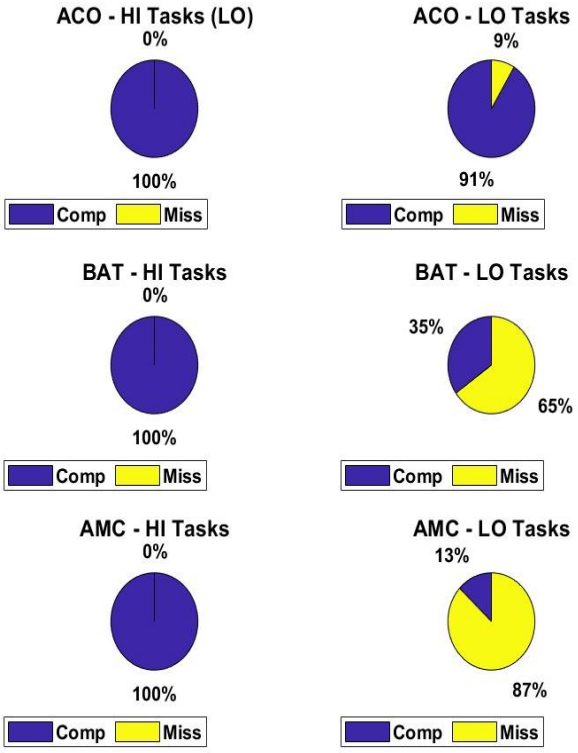
شکل ۴: نمودار مقایسه Gantt Chart

تحلیل نمودار گانت در شکل ۴، نکات فنی زیر را آشکار می‌سازد:

۱. تراکم اجرا (Packing Density): در نمودار روش پیشنهادی (Current)، بلوک‌های رنگی (نشان‌دهنده اجرای وظایف) با فشردگی و نظم بیشتری کنار هم قرار گرفته‌اند. این نشان‌دهنده کاهش زمان‌های بیکاری (Idle Time) پردازنده است. الگوریتم ACO با استفاده از ماتریس فرومون، ترکیبی از وظایف را پیدا کرده که مانند قطعات پازل، کمترین فاصله زمانی را ایجاد کنند.

۲. مدیریت لحظه سوئیچ: در نمودار پایین (AMC)، پس از خط قرمز عمودی، مشاهده می‌شود که اجرای وظایفLO پس از ورود به حالت بحرانی کاملا حذف شده اند ؛ به این معنی که سیستم فقط روی وظایف خاص (HI) متمرکز شده و سایرین را قطع کرده است. اما در نمودار بالا (روش پیشنهادی و الگوریتم خفاش)، حتی پس از خط قرمز، همچنان شاهد حضور بلوک‌های مربوط به وظایف متنوع هستیم. این یعنی سیستم توانسته است "بودجه زمانی" لازم برای بقای وظایف غیرحیاتی را حتی در شرایط بحرانی حفظ کند.

همچنین، نتایج آماری کلی در شکل ۵ ارائه شده است. همان‌طور که مشاهده می‌شود، روش پیشنهادی موفق شده است 100% وظایف با اهمیت بالا (HI) را با موفقیت اجرا کند که شرط اصلی ایمنی سیستم است. درصد موفقیت وظایف کم اولویت نیز با توجه به محدودیت‌های پردازشی و اولویت‌دهی به وظایف اولویت بالا، در سطح مطلوبی حفظ شده است.



شکل ۵: نمودار مقایسه Pie Chart

با استناد به نمودار فوق، نتایج زیر قابل استخراج است:

۱. تضمین ایمنی مطلق (Safety Guarantee): در هر سه روش (ستون سمت چپ)، نمودارهای مربوط به HI Tasks تماماً آبی رنگ هستند که نشان‌دهنده نرخ تکمیل ۱۰۰٪ است. این نتیجه تأیید می‌کند که روش پیشنهادی مشابه استاندارد سخت‌گیرانه AMC، ایمنی وظایف حیاتی را تحت هر شرایطی (حتی در زمان وقوع Mode Switch) تضمین می‌کند.

۲. بهبود چشمگیر کارایی وظایف کم اولویت: تفاوت اصلی در ستون سمت راست (LO Tasks) مشهود است:

روش AMC (ردیف پایین): بخش بزرگی از نمودار به رنگ زرد (Missed) درآمده است. این امر ناشی از ماهیت بدبینانه AMC است که به محض افزایش بار سیستم یا تغییر حالت، وظایف کم اولویت را به نفع تضمین HI حذف می‌کند.

روش پیشنهادی (ردیف بالا): همان‌طور که مشاهده می‌شود، سهم ناحیه زرد (Missed) به حداقل رسیده (حدود ۹٪) و ناحیه آبی (Completed) به ۹۱٪ افزایش یافته است. این نشان می‌دهد که الگوریتم ترکیبی با تنظیم دقیق اولویت‌ها، توانسته است فضای خالی (Slack Time) موجود در بین وظایف اولویت بالا را به گونه‌ای استخراج کند که حداکثر تعداد وظایف کم اولویت بدون آسیب به ایمنی سیستم اجرا شوند.

۶- نتیجه‌گیری

در این مقاله، مسئله تخصیص اولویت در سیستم‌های بلادرنگ با بحرانی مختلط تطبیقی بر روی تک‌پردازنده بررسی شد. با توجه به پیچیدگی مسئله و محدودیت روش‌های قطعی در یافتن ترتیب مناسب اولویت وظایف، یک روش ترکیبی مبتنی بر الگوریتم‌های خفاش و کلونی مورچگان پیشنهاد گردید. به‌منظور ارزیابی عملکرد روش پیشنهادی، یک مطالعه موردی شامل ۱۰ وظیفه طراحی و رفتار سیستم در شرایط تغییر حالت بحرانی مورد بررسی قرار گرفت.

نتایج شبیه‌سازی نشان داد که روش پیشنهادی در تمامی آزمایش‌ها موفق به حفظ ۱۰۰ درصد وظایف با سطح بحرانی بالا (HI) شد و در نتیجه، تضمین‌های زمانی موردنیاز این دسته از وظایف به‌طور کامل حفظ گردید. همچنین، در بخش وظایف با سطح بحرانی پایین، روش پیشنهادی توانست ۹۱ درصد از وظایف را با موفقیت اجرا کند؛ در حالی که الگوریتم خفاش به‌تنهایی، بسته به نقطه همگرایی، عملکردی بین ۳۵ تا ۹۱ درصد داشت و در برخی اجراها به دلیل گرفتار شدن در بهینه‌های محلی، بخش قابل توجهی از وظایف کم اولویت را از دست داد. این نتایج نشان می‌دهد که استفاده از الگوریتم کلونی مورچگان پس از الگوریتم خفاش، موجب هدایت مؤثرتر فرآیند جستجو، کاهش احتمال همگرایی در بهینه‌های محلی و دستیابی پایدار به بهترین جواب حاصل از الگوریتم خفاش شده است.

همچنین، در مقایسه با اجرای مستقیم پروتکلAMC ، که تنها ۱۳ درصد از وظایف کم اولویت را در شرایط بحرانی حفظ می‌کند، روش پیشنهادی این مقدار را به ۹۱ درصد افزایش داده است که بیانگر بهبودی معادل ۷۸ واحد درصد در نرخ موفقیت اجرای وظایف کم اولویت است. این نتایج نشان می‌دهد که ترکیب دو الگوریتم فراابتکاری می‌تواند ضمن حفظ کامل تضمین اجرای وظایف اولویت بالا، کیفیت سرویس وظایف کم اولویت را نیز به‌طور قابل توجهی افزایش دهد.

در نهایت، باید اشاره کرد که ارزیابی انجام‌شده در این پژوهش با هدف بررسی امکان‌سنجی روش پیشنهادی و تحلیل رفتار آن در یک مطالعه موردی انجام شده است. بررسی عملکرد این روش در مجموعه‌وظایف بزرگ‌تر و مقایسه آن با سایر روش‌های نوین زمان‌بندی مبتنی بر مجموعه‌داده‌های استاندارد، می‌تواند به‌عنوان یکی از مسیرهای ادامه این پژوهش مورد توجه قرار گیرد.

مراجع

[1] G. C. Buttazzo, Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications, New York: Springer, 2011 .

[2] S. Vestal, " Preemptive Scheduling of Multi-Criticality Systems with Varying Degrees of Execution Time Assurance," in *28th IEEE International Real-Time Systems Symposium*, 2007 .

[3] S.K. Baruah , A.Burns ,R.I. Davis, "Response-Time Analisis For Mixed-Criticality Systems," in *Proc. IEEE Real-Time Systems Symposium (RTSS)*, 2011 .

[4] Marco Dorigo, Tomas Stutzle, Ant Colony Optimization, MIT Press, 2004 .

[5] X.-S. Yang, "A New Metaheuristic Bat-Inspired Algorithm," in *Springer Berlin Heidelberg*, 2010 .

[6] Burns, Alan; Davis, Robert I, "Schedulability Analysis for Adaptive Mixed-Criticality Systems with Arbitrary Deadlines and Semi-Clairvoyance," in *IEEE Real-Time Systems Symposium (RTSS)*, 2020 .

[7] J. Liu, W. Zheng, X. Li, and J. Chen, "Scheduling Mixed-Criticality Systems with Degraded Service," *EEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, pp. 2022-2034, 2016 .

[8] e. a. T. Mendes, "Deep Reinforcement Learning for Dynamic Scheduling of Mixed-Criticality Systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 43, pp. 450-463, 2024 .

[9] A. Burns and R. I. Davis, "A Survey of Research on Mixed Criticality Systems," *ACM Computing Surveys (CSUR)*, vol. 50, 2018 .

[10] Alan burns, Robert I.Davis, "Mixed Criticality System - A Rewiew," Department of Computer Science, University of York, 2017.

[11] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer methods in applied mechanics and engineering*, vol. 186, no. 311-338, 2000.

پاورقی‌ها:

Embedded Systems [1]
[2] Real time
[3] Adaptive Mixed-Criticality
[4] Low-criticality tasks
[5] High-criticality tasks
[6] Uniprocessor
[7] Sporadic
[8] Mode Switch
[9] Overload
[10] Static Mixed Criticality
[11] Fixed-Priority
[12] Responses Time Analysis
[13] Responses Time Bound
[14] Exploration
[15] Echo Location
[16] Smallest Position Value
[17] Exploitation
[18] State Transition Rule
[19] Heuristic Information
[20] Pheromone Evaporation
[21] Fitness Function
[22] Permutation
[23] Weighted Penalty Function
[24] Penalty Functions
[25] Fine-tuning