Journal of Soft Computing and Information Technology (JSCIT)
Babol Noshirvani University of Technology, Babol, Iran
Journal Homepage: jscit.nit.ac.ir
Volume 13, Number 2, Summer 2024, pp. 69-87
Received: 11/07/2022, Revised: 10/17/2023, Accepted: 11/16/2024

A Multi-objective Data-Driven Adaptation Method with Goal Type Transformation for Software Systems

Mahnaz MohammadkhanpourYamchi[1], Eslam Nazemi[2*] and Mitra Mirzarezaee[3]
1- Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran.
2*- Faculty of Science & Computer Engineering, Shahid Beheshti University, Tehran, Iran.
3- Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran.
[1] mahnaz.mohammadkhanpour@srbiau.ac.ir, [2*] nazemi@sbu.ac.ir, and [3] mirzarezaee@srbiau.ac.ir

Corresponding author's address: Eslam Nazemi, Faculty of Science & Computer Engineering, Shahid Beheshti University, Tehran, Iran.

Abstract- Self-adaptive software systems can operate continuously despite uncertainties. Control-theoretic adaptation approaches, which construct a mathematical model of the system, provide guarantees for the satisfaction of the self-adaptive system requirements, but systems undergoing frequent changes in their behavior can't use these approaches due to continuous re-identification. Therefore, researchers present adaptation approaches that combine control theory with artificial intelligence or machine learning. The adaptation solution in these approaches does not rely on the mathematical model of the system, but it does not consider the priorities of the goals, and how to provide guarantees for the goals of the self-adaptive system is not determined. This paper proposes a multi-objective data-driven adaptation method for software systems and presents a formal analysis of the guarantee of satisfying the self-adaptive system requirements. The adaptation mechanism in the method relies on the input and output data of the system and determines an adaptation decision by solving a multi-objective optimization problem. The self-adaptive software system based on the data-driven adaptation method can deal with uncertainties in the execution environment, requirements, and system parameters, and it also supports goal transformation at runtime. Two cases in different domains evaluate the data-driven adaptation method. It is also compared with one of the model-based control-theoretic adaptation approaches and a data-driven adaptation method without goal transformation. The results confirm that the requirements of the self-adaptive software system based on the proposed adaptation method are satisfied despite uncertainties.

Keywords- Self-adaptive Software System, Adaptation approaches in Software Systems, Model-based Control Theory, Data-Driven Control Theory, Model-Free Adaptive Control Method, Multi-objective Optimization Problem, Goal Type Transformation.

مجله علمی رایانش نرم و فناوری اطلاعات
دانشگاه صنعتی نوشیروانی بابل
صفحه مجله: jscit.nit.ac.ir
جلد ۱۳، شماره ۲، تابستان ۱۴۰۳ ، صفحه ۶۹-۸۷
دریافت: ۱۴۰۱/۰۸/۱۶، بازنگری: ۱۴۰۲/۰۷/۲۵، پذیرش: ۱۴۰۳/۰۸/۲۶

روش تطبیق داده‌رانه چندهدفه با تبدیل نوع هدف برای سیستم‌های نرم‌افزاری

مهناز محمدخان پور یامچی[۱]، اسلام ناظمی*[۲]، میترا میرزارضائی[۳]
۱- دانشکده برق و کامپیوتر، واحد علوم و تحقیقات، دانشگاه آزاد اسلامی، تهران، ایران.
*۲- دانشکده مهندسی و علوم کامپیوتر، دانشگاه شهید بهشتی، تهران، ایران.
۳- دانشکده مکانیک، برق و کامپیوتر، دانشگاه آزاد اسلامی واحد علوم و تحقیقات، تهران، ایران.
[1] mahnaz.mohammadkhanpour@srbiau.ac.ir, [2*]nazemi@sbu.ac.ir, [3] mirzarezaee@srbiau.ac.ir

* نشانی نویسنده مسئول: اسلام ناظمی، تهران، اوین، میدان شهید شهریاری، دانشگاه شهید بهشتی، دانشکده مهندسی و علوم کامپیوتر.

چکیده - سیستم‌های نرم‌افزاری خودتطبیق می‌توانند علی‌رغم عدم‌قطعیت‌های مختلف به فعالیت خود ادامه دهند. رویکردهای تطبیق مبتنی بر تئوری کنترل که یک مدل ریاضی از سیستم ایجاد می‌کنند، تضمین‌هایی را برای تأمین نیازمندی‌های سیستم خودتطبیق ارائه می‌دهند؛ اما سیستم‌هایی با تغییرات رفتاری متناوب نمی‌توانند این رویکردها را به دلیل بازشناسایی مداوم مدل سیستم، بکار برند؛ لذا محققان رویکردهای تطبیقی ارائه دادند که تئوری کنترل را باهوش مصنوعی یا یادگیری ماشین تلفیق می‌کنند. راه‌حل تطبیق در این رویکردها بر مبنا مدل ریاضی از سیستم نیست؛ اما اولویت اهداف در آن در نظر گرفته نمی‌شود و نحوه ارائه تضمین‌ها برای اهداف سیستم خود تطبیق نیز مشخص نمی‌باشد. این مقاله، یک روش تطبیق چندهدفه مبتنی بر اصول تئوری کنترل داده‌رانه برای سیستم‌های نرم‌افزاری پیشنهاد و تحلیل رسمی از تضمین تأمین نیازمندی‌های سیستم خودتطبیق ارائه می‌دهد. سازوکار تطبیق در این روش، وابسته به داده‌های ورودی و خروجی سیستم است و تصمیم تطبیق نیز از طریق حل یک مسئله بهینه‌سازی چندهدفه تعیین می‌شود. سیستم نرم‌افزاری خود تطبیق مبتنی بر روش تطبیق داده‌رانه می‌تواند با عدم‌قطعیت‌ها در محیط اجرایی، نیازمندی‌ها و پارامترهای سیستم مقابله و نیز از تبدیل نوع هدف در زمان اجرا پشتیبانی نماید. روش تطبیق داده‌رانه با دو مطالعه موردی در دامنه‌های مختلف مورد ارزیابی قرار می‌گیرد. همچنین این روش با یکی از رویکردهای تطبیق تئوری کنترل مبتنی بر مدل و نیز روش تطبیق داده‌رانه بدون پشتیبانی از تبدیل نوع هدف، مقایسه می‌شود. نتایج ارزیابی نشان می‌دهند که علی‌رغم عدم‌قطعیت‌ها، نیازمندی‌های سیستم نرم‌افزاری خود تطبیق مبتنی بر روش تطبیق پیشنهادی تأمین می‌شوند.

واژه‌های کلیدی: سیستم نرم‌افزاری خود تطبیق، رویکردهای تطبیق در سیستم‌های نرم‌افزاری، تئوری کنترل مبتنی بر مدل، تئوری کنترل داده‌رانه، روش کنترل تطبیق عاری از مدل، مسئله بهینه‌سازی چندهدفه، تبدیل نوع هدف.

۱- مقدمه

علی‌رغم عدم‌قطعیت‌هایی که پیش‌بینی آنها در زمان طراحی دشوار است، سیستم‌های نرم‌افزاری پیشرفته می‌بایست بتوانند بدون وقفه به فعالیت خود ادامه دهند [۱]. برای این منظور، محققان، رویکردهایی را پیشنهاد دادند که به سیستم نرم‌افزاری امکان می‌دهند که خودش را در پاسخ به تغییرات ناشی از عدم‌قطعیت‌ها، تنظیم نماید [۲] [۳] [۴]و چنین سیستمی را سیستم نرم‌افزاری خودتطبیق نامیدند که شامل دو قسمت اصلی؛ سیستم مدیریتی[۱] و سیستم تحت مدیریت[۲] است [۵]. سیستم تحت‌مدیریت اشاره به سیستم نرم‌افزاری با نیازمندی‌های قابل کمی‌سازی دارد که مورد تطبیق قرار می‌گیرد [۶] و سیستم مدیریتی، منطق تطبیق را از طریق یک حلقه بازخورد پیاده‌سازی می‌کند [۵].

سیستم مدیریتی برای سیستم نرم‌افزاری با نیازمندی‌های سخت[۳]، می‌بایست تضمین‌هایی را در خصوص تأمین نیازمندی‌ها ارائه دهد

[۱] [۷] [۸]. تئوری کنترل، ابزارها و تکنیک‌هایی رابرای طراحی و تحلیلی رسمی، سیستم حلقه بسته در اختیار قرار می‌دهد [۹] [۱۰] [۱۱] [۱۲]، اما طراحی سیستم مدیریتی مبتنی بر اصول تئوری کنترل برای تطبیق سیستم نرم‌افزاری در یک دامنه معین را نمی‌توان برای دامنه دیگر بکار برد [۱۳] [۱۴]. لذا محققان راه‌حل‌های تطبیق تئوری کنترلی را معرفی نمودند که به‌طور خودکار راهکار کنترلی را می‌سازند [۱] [۷] [۱۵] [۱۱] [۱۶] [۱۷] [۱۸] [۱۹]. اکثر آنها مبتنی بر متدولوژی فشار-دکمه بودند و یا از روش کنترلی پیش‌گویانه مدل[۴](MPC) برای ارائه رویکردهای تطبیقی تئوری کنترلی خودکار استفاده کردند. این رویکردها دارای سه مرحله اصلی؛ ساخت مدل، ایجاد کنترلر و عملیات بودند [۵] [۱۶] [۲۰]. مرحله ساخت مدل، یک مدل خطی ساده از سیستم نرم‌افزاری با استفاده از مجموعه مقادیر سیگنال کنترلی اِعمال‌شده به سیستم نرم‌افزاری و خروجی‌های اندازه‌گیری‌شده آن، می‌سازد [۵]. سپس، مرحله ایجاد کنترلر با استفاده از مدل ریاضی سیستم، یک کنترلر ایجاد می‌کند [۵] [۱۸]. سیستم نرم‌افزاری در مرحله عملیات، از طریق خروجی کنترلر تطبیق می‌یابد [۲۱] [۷] [۱۵]. برای مقابله با رفتارهای غیرخطی سیستم نرم‌افزاری، این رویکردها می‌بایست از فیلتر کالمن برای به‌روزرسانی مدل سیستم در طول مرحله عملیات استفاده نمایند [۷] [۱۵] [۱۱] [۱۸]. همچنین، آنها برای شناسایی تغییرات سیستم نرم‌افزاری، به سازوکار شناسایی نقطه تغییر[۵] نیاز دارند تا مرحله ساخت مدل جدید را راه‌اندازی نمایند [۷] [۱۵] [۱۱]. علی‌رغم این‌سازوکارها، این رویکردهای تطبیق تئوری کنترل مبتنی بر مدل را نمی‌توان برای سیستم‌هایی بکاربرد که رفتار آنها متناوباً در زمان اجرا تغییر می‌کنند، چون آنها برای ساخت مدل سیستم، مرحله ساخت مدل را به‌طور پیوسته راه‌اندازی می‌کنند [۱۶]. مرحله ساخت مدل در این رویکرهای تطبیق، از سیگنال کنترلی سیستم نرم‌افزاری نمونه برداری و خروجی حاصل از اعمال آنها به سیستم را اندازه‌گیری می‌نماید؛ سپس یک مدل خطی را با داده‌های ورودی و خروجی می‌سازد [۱۶]. سیگنال کنترلی برداری از محرک‌ها است و با افزایش تعداد محرک‌ها و مجموعه مقادیر آنها، تعداد پیکربندی‌های سیستم‌نرم‌افزاری به‌صورت نمایی رشد می‌کند؛ به‌طوری‌که، کاوش تمام پیکربندی‌های سیستم برای ساخت مدل ریاضی دقیق، بسیار زمانبر و یا غیرممکن است و پیچیدگی مدل ریاضی نیز منجر به پیچیدگی کنترلر و در نتیجه افزایش زمان محاسبه پارامترهای کنترلی برای تطبیق سیستم خواهد شد. از طرف دیگر، سیستم نرم‌افزاری در مرحله ساخت مدل، سیگنال‌های کنترلی را بدون در نظر گرفتن تأثیر عوامل عدم‌قطعیت اجرا می‌کند که در زمان طراحی، پیش‌بینی کامل تأثیرات عوامل عدم‌قطعیت بر روی سیستم نرم‌افزاری خودتطبیق اجرایی در مرحله عملیات دشوار است؛ بنابراین سیستم نرم‌افزاری خودتطبیق در مرحله عملیات، به ساز وکارهایی برای شناسایی نقطه تغییر برای راه‌اندازی مرحله ساخت مدل نیاز دارد.

برای مواجه با محدودیت‌های مرتبط با ساخت مدل ریاضی، رویکردهای جدید تطبیق [۲۲] [۲۳] [۲۴] مدل ریاضی از سیستم نرم‌افزاری ایجاد نمی‌کنند و روش‌های هوش مصنوعی، یادگیری ماشین و حلقه پایش-تحلیل-طراحی-اجرا[۶](MAPE) را با اصول تئوری کنترل ترکیب می‌نمایند. این رویکردها سازوکارهایی را برای نحوه مواجه سیستم نرم‌فزاری با عدم‌قطعیت‌ها در زمان اجرا ندارند و نیز نحوه ارائه تضمین‌ها برای تأمین اهداف سیستم نرم‌افزاری در این رویکردها مشخص نیست.

در این مقاله، برای غلبه بر این محدودیت‌ها، رویکرد تطبیق داده‌رانه مبتنی بر روش کنترل تطبیقی عاری از مدل[۷] [۲۵] برای تطبیق سیستم نرم‌افزاری پیشنهاد می‌شود که نسبت به رویکردهای تطبیق قبلی، ویژگی‌های زیر را دارد.

- یک مدل داده خطی‌سازی پویا[۸] به‌جای مدل ریاضی برای ساخت کنترلر استفاده می‌کند.
- عملیات تطبیق سیستم در هر چرخه تطبیق با حل یک مسئله بهینه‌سازی چندهدفه و با درنظرگرفتن اولویت‌های اهداف، تعیین می‌شوند.
- فعال‌سازی، غیرفعال‌سازی نیازمندی‌ها و نیز تبدیل نیازمندی نقطه‌تنظیم به بهینه‌سازی و بالعکس در طول مرحله عملیات را پشتیبانی می‌کند.
- منطق تطبیق اجرایی را برای مقابله با عدم‌قطعیت‌ها در محیط اجرایی و نیز پارامترهای سیستم، در طول عملیات با استفاده از داده‌های ورودی و خروجی سیستم، به‌روزرسانی می‌کند.
- سیستم نرم‌افزاری خود تطبیق مبتنی بر روش تطبیق داده‌رانه در نقطه تعادل، پایدار است. به‌عبارت‌دیگر سیستم نرم‌افزاری می‌تواند نیازمندی‌های مشخص‌شده در زمان طراحی را برآورده نماید.

عملکرد روش تطبیق ارائه‌شده در دو دامنه کاربردی مورد ارزیابی قرار می‌گیرد. دامنه اول، سازوکار انقیاد پویا چندهدفه[۹] [۲۶] در سیستم‌های نرم‌افزاری سرویس گرا [۱۷] [۱۸] و دامنه دوم، سیستم زیردریایی بدون‌سرنشین[۱۰] است [۲۷] [۱۵].

همچنین روش تطبیق پیشنهادی با یک رویکرد تطبیق تئوری کنترل مبتنی بر مدل و نیز با روش تطبیق داده‌رانه بدون پشتیبانی از تبدیل نوع هدف مقایسه می‌شود.

ابتدا مقاله در بخش دوم، به بیان رویکردهای تطبیق قبلی مبتنی بر

جدول ۱: رویکردهای تطبیق سیستم نرم‌افزاری مبتنی بر اصول تئوری کنترل [۱].

روش تطبیق	روش کنترلی	اهداف سیستم خودتطبیق		عدم قطعیت			تبدیل اهداف	تضمین‌ها	
		تعداد اهداف	اولویت	نیازمندی	پارامتر سیستم	محیطی		پایداری	بهینگی
PBM [۱۱]	مبتنی بر مدل	یک						✓	
AMOCS [۱۷]	PBM	چند	✓					✓	
SimCA [۷]	PBM	چند		✓				✓	✓
AMOCS-MA [۱۸]	MPC	چند	✓					✓	✓
SimCA* [۱]	PBM	چند		✓	✓	✓	✓	✓	✓
رویکرد تطبیق کالداس [۲۳]	PI	یک							
رویکرد تطبیق صبوحی [۲۲]	PID	یک							
رویکرد تطبیق وینز [۲۴]	PI	چند							
روش پیشنهادی	MFAC	چند	✓	✓	✓	✓	✓	✓	✓

تئوری کنترل می‌پردازد. بخش سوم به معرفی مفاهیم کاربردی روش کنترل تطبیقی عاری از مدل و بلوک‌های اصلی یک سیستم نرم‌افزاری خودتطبیق مبتنی بر روش تطبیق پیشنهادی در قالب شِمای کنترلی می‌پردازد. بخش چهارم، نحوه تطبیق سیستم نرم‌افزاری توسط روش تطبیق پیشنهادی را توضیح می‌دهد و بخش پنجم نیز روش تطبیق را ارزیابی و با سایر روش‌ها مقایسه می‌کند. بخش ششم، به نتایج و تحقیقات آتی می‌پردازد.

۲- کارهای مرتبط

اولین رویکرد تطبیق تئوری کنترل خودکار، تحت عنوان متدولوژی فشار - دکمه[۱۱]، توسط آقای فیلیری و همکارانش در سال ۲۰۱۴ معرفی و پایه‌ای برای رویکردهای تطبیق بعدی شد[۱۱] . محدودیت اصلی متدولوژی فشار – دکمه، این است که سیستم نرم‌افزاری را برای تأمین یک نیازمندی غیرعملکردی تطبیق می‌دهد. محققان برای رفع این محدودیت، رویکردهایی را پیشنهاد دادند که چندین هدف تطبیق را برآورده می‌کنند. در این راستا، آقای فیلیری و همکارانش، درسال ۲۰۱۵ متدولوژی فشار - دکمه را توسعه دادند و یک رویکرد تطبیقی تحت عنوان کنترل چندهدفه برای نرم‌افزار خودتطبیق[۱۲](AMOCS) را ارائه کردند[۱۷] . این رویکرد، محرک‌ها را براساس اولویت اهداف تطبیق، به آنها اختصاص می‌دهد، به‌طوری‌که یک هدف با اولویت کمتر فقط می‌توانست از محرک‌هایی استفاده نماید که بر روی اهداف با اولویت بالاتر اثرگذار نباشد [۱۷]. سپس برای هر هدف، یک راه‌حل کنترلی مجزا با استفاده از متدولوژی فشار - دکمه، طراحی می‌شود[۱۷] . این رویکرد فقط درصورتی‌که اهداف با اولویت بالاتر تامین شوند، تضمین‌هایی را برای اهداف با اولویت پایین ارائه می‌دهد[۱] .

رویکرد تطبیق کنترلی سیمپلکس[۱۳] (SimCA) در سال ۲۰۱۶ توسط آقایان شوتسوف و وینز مبتنی بر متدولوژی فشار - دکمه پیشنهاد شد که تأمین چندین هدف نقطه‌تنظیم[۱۴] تضمین و یکی از اهداف با استفاده از روش بهینه‌سازی سیمپلکس، بهینه می‌شود [۷]. این رویکرد برای هر هدف نقطه‌تنظیم، یک کنترلر با استفاده از تمام محرک‌ها می‌سازد و سپس روش سیمپلکس با درنظر گرفتن سیگنال‌های کنترلی تمام کنترلرها، مسئله بهینه‌سازی را حل می‌کند [۷].

رویکرد تطبیق SimCA توسط شوتسوف و همکارانش در سال ۲۰۱۹ توسعه یافت و رویکرد SimCA* معرفی شد که قابلیت پشتیبانی از تبدیل نوع اهداف را به رویکرد SimCA افزود و به سیستم نرم‌افزاری خودتطبیق امکان مقابله با عدم‌قطعیت‌ها در تعاملات مؤلفه‌های نرم‌افزاری، نیازمندی‌های سیستم و نیز پارامترهای سیستم را می‌دهد[۱] .

راه‌حل بهینه ارائه‌شده توسط رویکردهای تطبیق SimCA و SimCA* ، امکان اولویت‌بندی اهداف را ندارند[۲۰] و لذا محققان به بررسی و بکارگیری روش کنترل پیش‌گویانه مدل در راه‌حل‌های تطبیق مبتنی بر تئوری کنترل پرداختند[۲۰] و در این راستا، خانم ماگیو و همکارانش در سال ۲۰۱۷، روش کنترل خودکار برای چندین هدف نرم‌افزاری با چندین محرک[۱۵](AMOCS-MA) را پیشنهاد دادند که یک راه‌حل کنترلی بهینه را با درنظرگرفتن تمام وابستگی‌ها بین محرک‌ها و اهداف، تولید می‌کند[۱۸] . این رویکرد تطبیق به اختلالات تکرارشونده و نیز عدم‌دقت مدل حساس است و از هیچ سازوکاری برای مقابله با عدم‌قطعیت‌ها در زمان اجرا

جدول ۲: قابلیت(ها) و محدودیت(ها)رویکردهای تطبیق سیستم نرم‌افزاری مبتنی بر اصول تئوری کنترل

روش تطبیق	قابلیت(ها)	محدودیت(ها)
PBM [۱۱]	تحلیل رسمی از ضمانت‌های قابل ارائه برای سیستم نرم افزاری خودتطبیق	تأمین یک نیازمندی غیرعملکردی
AMOCS [۱۷]	تأمین چندین نیازمندی غیرعملکردی	عدم ارائه تضمین برای تأمین اهداف با اولویت پایین
SimCA* [۱]	• تأمین چندین هدف و ضمانت بهینه‌بودن یک هدف اضافی • سازوکار برای مواجه با برخی از عدم‌قطعیت‌ها • پشتیبانی از تبدیل نوع اهداف	• عدم امکان اولویت‌بندی اهداف در راه‌حل بهینه • تغییرات مداوم در سیستم نرم‌افزاری منجر به راه‌اندازی پیاپی مرحله ساخت مدل می‌شود. • تبدیل به هدف نقطه تنظیم در زمان اجرا نیاز به راه‌اندازی مرحله ساخت مدل دارد.
AMOCS-MA [۱۸]	اولویت بندی اهداف سیستم خودتطبیق	• حساسیت به اختلالات تکرارشونده و عدم دقت مدل • فاقد سازوکاری برای مقابله با عدم‌قطعیت‌ها در زمان اجرا
رویکرد تطبیق کالداس [۲۳]	بهبود عملکرد سیستم مدیریتی	• تأمین یک نیازمندی غیرعملکردی • عدم ارائه تضمین رسمی برای تأمین اهداف سیستم نرم افزاری خودتطبیق • عدم وجود سازوکارهایی برای مواجه با عدم‌قطعیت‌های زمان اجرا
رویکرد تطبیق وینز [۲۴]	طراحی رویکرد تطبیق با بکارگیری داده‌های سیستم؛ مدل‌سازی پویایی سیستم با روش‌های یادگیری ماشین و اجرای تصمیمات تطبیق توسط کنترلر(های) PI	• عدم ارائه تضمین رسمی برای تأمین اهداف سیستم نرم افزاری خودتطبیق • عدم وجود سازوکارهایی برای مواجه با عدم‌قطعیت‌های زمان اجرا
رویکرد تطبیق صبوحی [۲۲]	طراحی رویکرد تطبیق با بکارگیری داده‌های سیستم نرم‌افزاری ابری و کنترلر تطبیقی PID	• تأمین یک نیازمندی غیرعملکردی • عدم ارائه تضمین رسمی برای تأمین اهداف سیستم نرم افزاری خودتطبیق • عدم وجود سازوکارهایی برای مواجه با عدم‌قطعیت‌های زمان اجرا
روش پیشنهادی	• تطبیق سیستم نرم‌افزاری با بکارگیری داده‌های سیستم • تعیین پیکربندی بهینه با توجه به اولویت اهداف • تحلیل رسمی از پایداری سیستم خودتطبیق • امکان تبدیل نوع اهداف سیستم خودتطبیق در زمان اجرا • سازوکار برای مواجه با برخی از عدم‌قطعیت‌ها	• عدم امکان بکارگیری در محیط‌های دارای نویز شدید به دلیل تأثیر نویز بر روی داده‌های سیستم • عدم ارائه تحلیل رسمی برای سایر ویژگی‌های کنترلر همچون فراجهش و خطای حالت پایدار به دلیل محدودیت روش کنترلی • تعیین وزن اهداف در زمان طراحی با توجه به اولویت آنها

پشتیبانی نمی‌کند[۱۶] .

برای مواجه با محدودیت‌های مرتبط با ساخت مدل ریاضی، رویکردهای جدید تطبیق [۲۲] [۲۳] [۲۴] مدل ریاضی از سیستم نرم‌افزاری ایجاد نمی‌کنند و روش‌های هوش مصنوعی، یادگیری ماشین و حلقه MAPE را با اصول تئوری کنترل ترکیب می‌نمایند.

آقای صبوحی و همکارانش در سال ۲۰۲۱ سیستم مدیریتی برای سیستم‌های نرم‌افزاری ابری کانتینری مبتنی بر کنترلر تطبیقی PID را طراحی کردند که مدل کارایی سیستم نرم‌افزاری را با استفاده از شبکه عصبی و داده‌های سیستم مدل‌سازی می‌کند[۲۲] . این مدل، مقدار زمان پاسخ سیستم‌نرم‌افزاری و پویایی‌های سیستم نرم‌افزاری در هرگام تطبیق را تخمین می‌زند؛ سپس مقادیر پارامترهای کنترلرPID بر اساس پویایی‌های تخمین‌زده‌شده، به‌روز رسانی می‌شوند [۲۲]. این رویکرد تطبیق صرفاً یک هدف سیستم خودتطبیق را می‌تواند تأمین نماید.

آقای کالداس و همکارانش در سال ۲۰۲۰، یک سیستم دوسطحی برای سیستم‌خودتطبیق پیشنهاد دادند [۲۳]. مؤلفه مدیر راهبرد در سطح اول با جستجو در فضای پیکربندی‌های سیستم، مجموعه راهبردها برای دست‌یابی به اهداف سیستم، محاسبه می‌نماید [۲۳]. مؤلفه اجرا کننده راهبرد در سطح دوم مبتنی بر حلقه MAPE، سیستم را پایش، تحلیل و در صورت لزوم مجموعه اعمال مشخص شده در راهبرد را با کمک کنترلر PI، اجرا می‌کند [۲۳]. الگوریتم تکاملی ژنتیکNSGA-II، مقادیر بهینه پارامترهای فرآیند جستجو در سطح اول و نیز پارامترهای کنترلر در سطح دوم را تعیین می‌کند [۲۳]. هدف اصلی این رویکرد، بهبود عملیات تطبیق در سیستم مدیریتی با بهینه‌سازی زمان یافتن راهبرد تطبیق و پارامترهای تئوری کنترل شامل فراجهش[۱۶] و خطای حالت پایدار[۱۷] است [۲۳]. تمرکز این رویکرد بر تأمین یک نیازمندی است و نتایج آن وابسته به تکنیک‌های یادگیری مدل بکارگرفته است [۲۳].

آقای وینز و همکارانش در سال ۲۰۲۱، سیستم مدیریتی دوسطحی را مبتنی بر حلقه MAPE و تئوری کنترل ارائه دادند [۲۴]. سیستم مدیریتی در سطح اول با پایش اهداف سیستم و نیز بکارگیری تکنیک‌های یادگیری ماشین، تصمیمات تطبیق و نیز نقاط تنظیم را

برای سطح دوم مشخص می‌کند [۲۴]. سطح دوم، تصمیمات تطبیق را مبتنی بر اصول تئوری کنترل با مجموعه‌ای از کنترلرهایPI، محقق می‌نماید [۲۴].

سه رویکرد اخیر تطبیق برای یک دامنه خاص پیشنهاد شدند و تعمیم‌پذیری آنها در سایر دامنه‌ها مورد بررسی قرار نگرفت. نحوه ارائه تضمین‌ها برای اهداف سیستم نرم‌افزاری خودتطبیق در این سه رویکرد تطبیق مشخص نمی‌باشد و نیز سازوکارهایی برای نحوه مواجه با انواع مختلف عدم‌قطعیت ارائه ندادند.

رویکردهای تطبیق مبتنی بر اصول تئوری کنترل در جدول ۱ مورد مقایسه قرار گرفتند و قابلیت‌ها و محدودیت‌های این رویکردها در جدول ۲ بیان شدند.

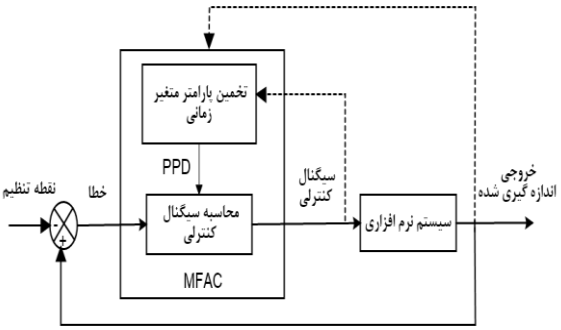
نوآوری روش تطبیق پیشنهادی عبارت است از حصول اطمینان از تأمین نیازمندی‌های سیستم نرم‌افزاری خودتطبیق از طریق تطبیق بهینه سیستم نرم‌افزاری در مواجه با عدم‌قطعیت در پارامترهای سیستم، عدم‌قطعیت در نیازمندی‌ها و عدم‌قطعیت در محیط اجرایی، بدون نیاز به ساخت/بازسازی/به‌روزرسانی مدل ریاضی از سیستم و صرفاً با استفاده از داده‌های ورودی و خروجی سیستم نرم‌افزاری و نیز پشتیبانی از تبدیل نوع هدف‌ها (تبدیل هدف بهینه‌سازی به هدف نقطه‌تنظیم و بالعکس).

۳- روش کنترل تطبیقی عاری از مدل

روش تطبیق داده‌رانه ارائه‌شده در این مقاله، از روش کنترل تطبیقی عاری از مدل برای سیستم‌های زمان گسسته غیرخطی چند ورودی چند خروجی[۱۸]، استفاده می‌کند که اولین‌بار توسط آقای هو و همکارانش معرفی شد[۲۵] [۲۸]. این روش کنترلی مشابه سایر روش‌های کنترلی داده‌رانه از داده‌های ورودی و خروجی سیستم برای طراحی کنترلر استفاده می‌کند [۲۵] [۲۹]و نیز مطابق اصول کنترل تطبیقی، با بکارگیری روش‌های تخمین، پارامترهای سیستم کنترلی را به‌روزرسانی می‌کند [۲۸] [۳۰].

روش کنترل تطبیقی عاری از مدل، بر اساس تکنیک خطی‌سازی پویا، سیستم غیرخطی چند ورودی چند خروجی را با یک مدل داده خطی‌سازی پویا جایگزین می‌کند [۲۵]. این مدل داده، یک پارامتر متغیر زمانی تحت‌عنوان شبه مشتق جزئی[۱۹](PPD) دارد [۲۵].

شکل ۱، شِمای کنترلی یک سیستم نرم‌افزاری خود تطبیق را نشان می‌دهد که بر اساس روش کنترل تطبیقی عاری از مدل طراحی‌شده است. مطابق شکل ۱، از سمت چپ به راست، «نقطه‌تنظیم»، مقدار هدف برای یک نیازمندی غیرعملکردی است که سیستم خود تطبیق می‌بایست به آن برسد. مثلاً، سیستم نرم‌افزاری سرویس‌گرا می‌بایست مقدار معینی را برای نیازمندی قابلیت اطمینان فراهم نماید. همچنین مقدار «خطا»، بر اساس مقدار نقطه‌تنظیم و مقدار



شکل ۱: نمودار بلوکی شِمای کنترل تطبیقی عاری از مدل.

«خروجی اندازه‌گیری شده»، محاسبه می‌شود.

روش کنترل تطبیق عاری از مدل، مشابه روش‌های کنترل تطبیقی [۳۰]، دو حلقه بازخورد دارد. مؤلفه «محاسبه سیگنال کنترلی» در حلقه بازخورد اول (خط توپر) «سیگنال کنترلی» را با استفاده از الگوریتم تصویر[۲۰] و داده‌های ورودی و خروجی در هر گام کنترلی محاسبه می‌کند و این سیگنال کنترلی، سیستم نرم‌افزاری را به نحوی تطبیق می‌دهد که خروجی‌های سیستم به مقادیر نقطه‌تنظیم برسند یا به آنها نزدیک شوند. مؤلفه «تخمین پارامتر متغیر زمانی» در حلقه بازخورد دوم (خط‌چین) با استفاده از الگوریتم تصویر و داده‌های سیستم، مقدار پارامتر متغیر زمانی مدل داده سیستم را به‌روزرسانی می‌کند.

الگوریتم تصویر در حلقه بازخورد اول، تابع هزینه تعریف‌شده بر اساس خطا را کمینه می‌کند و این الگوریتم در حلقه بازخورد دوم، تابع هزینه‌ای را کمینه می‌کند که مبتنی بر خطای تخمین پارامتر متغیر زمانی تعریف‌شده است.

روش تطبیق داده‌رانه پیشنهادی در حلقه بازخورد اول، از مسئله بهینه‌سازی چندهدفه به‌جای الگوریتم تصویر برای تعیین تصمیمات بهینه تطبیق استفاده می‌کند. این مسئله بهینه‌سازی چندهدفه، تابع هزینه‌ای را کمینه می‌نماید که بر اساس مقادیر خطا و اولویت‌های نیازمندی‌ها تعریف‌شده است.

روش کنترل تطبیق عاری از مدل، پایداری سیستم کنترلی را تضمین می‌کند [۲۹]، به‌عبارت‌دیگر سیستم کنترلی می‌تواند به اهداف خود در نقطه تعادل برسد.

۴- روش تطبیق داده‌رانه چندهدفه با قابلیت تبدیل نوع اهداف

روش تطبیق داده‌رانه ارائه‌شده در این بخش، مبتنی بر اصول تئوری کنترل تطبیقی عاری از مدل است و از داده‌های ورودی و خروجی سیستم نرم‌افزاری برای تطبیق سیستم استفاده می‌کند؛ بنابراین مشکلات مرتبط با عدم دقت مدل‌سازی سیستم در رویکردهای تئوری کنترل مبتنی بر مدل، در این روش مطرح نیست و سیستم

نرم‌افزاری خود تطبیق مبتنی بر این روش در برابر عدم‌قطعیت در مدل سیستم، مقاوم است.

کنترلر در روش تطبیق داده‌رانه با استفاده از مدل داده خطی‌سازی پویا به‌جای مدل ریاضی سیستم، ساخته می‌شود. مدل خطی‌سازی پویا، دقیقاً معادل با سیستم نرم‌افزاری است[۳۱] که بر اساس تکنیک خطی‌سازی پویا و داده‌های ورودی و خروجی سیستم، ایجاد می‌شود.

این روش تطبیق از قابلیت تبدیل نوع هدف نقطه‌تنظیم به نوع هدف بهینه‌سازی و بالعکس پشتیبانی می‌کند؛ به‌طوری‌که با فرض تأمین اهداف نقطه‌تنظیم، منظور از تأمین یک هدف بهینه‌سازی، محاسبه مقدار بهینه آن است.

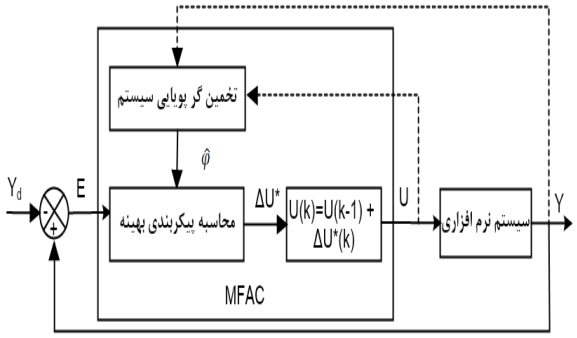
مفروضات روش تطبیق داده‌رانه عبارت‌اند از:

- اهداف سیستم نرم‌افزاری خودتطبیق، ویژگی‌های کیفی با قابلیت کمی‌سازی هستند؛ به‌طوری‌که، مقدار آنها در هر چرخه تطبیق را می‌توان اندازه‌گیری نمود.
- ذی‌نفعان سیستم خود تطبیق، مقادیر نقطه‌تنظیم و ترجیحات اهداف را مشخص می‌کنند.
- تزاحم بین اهداف بر اساس وزن اختصاص‌یافته به آنها، حل می‌شود.
- سیستم خود تطبیق، حس‌گرهایی برای اندازه‌گیری خروجی‌های سیستم در هر چرخه تطبیق دارد و این حس‌گرها به‌صورت نرم‌افزاری پیاده‌سازی شده‌اند.
- سیستم خودتطبیق، محرک‌هایی دارد که حداقل بر روی یکی از اهداف اثرگذار هستند.
- محرک‌ها در هر چرخه تطبیق به سیستم نرم‌افزاری شبیه‌سازی‌شده اِعمال و خروجی‌ها اندازه‌گیری می‌شوند.

روش تطبیق داده‌رانه پیشنهادی، دارای سه مرحله اصلی؛ ساخت مدل داده خطی‌سازی پویا، ساخت کنترلر و عملیات است. مرحله ساخت کنترلر نیز شامل دو زیر مرحله تخمین‌گر پویایی سیستم و محاسبه پیکربندی بهینه است. هریک از این مراحل در ادامه بیان خواهند شد.

مطابق اصل داخلی سیستم خود تطبیق[۲۱]، یک سیستم خود تطبیق دارای دو بخش متمایز است. [۳۲] بخش اول، سیستم تحت مدیریت و مسئول پیاده‌سازی عملکردهای سیستم نرم‌افزاری است و با مسائل دامنه سروکار دارد [۳۲] [۵]. بخش دوم، سیستم مدیریتی است که با تطبیق سیستم تحت مدیریت، اهداف تطبیق را محقق می‌سازد و به مسائل سیستم تحت مدیریت می‌پردازد [۳۲] [۵].

رویکردهای قبلی تطبیق، منطق تطبیق را از طریق یک حلقه بازخورد پیاده‌سازی می‌کنند که محیط و سیستم تحت مدیریت را پایش می‌کند و درصورتی‌که اهداف تطبیق نقض شوند، سیستم را تطبیق می‌دهد. روش تطبیق ارائه‌شده در مقایسه با این رویکردها، سیستم مدیریتی را از طریق دو حلقه بازخورد پیاده‌سازی می‌کند. مطابق شکل ۲، حلقه بازخورد اول که با خط توپر نمایش یافته، مرتبط با پیاده‌سازی منطق تطبیق و حلقه بازخورد دوم که با خط‌چین نمایش یافته، منطق تطبیق اجرایی را با به‌روزرسانی پارامتر متغیر زمانی مدل داده، تنظیم می‌کند.



شکل ۲: شِمای کنترلی سیستم نرم‌افزاری خودتطبیق داده‌رانه.

۴-۱- ساخت مدل داده خطی‌سازی پویا

سیستم نرم‌افزاری دارای چند ورودی قابل‌تنظیم تحت عنوان محرک‌ها و چند خروجی قابل‌اندازه‌گیری تحت عنوان اهداف است. مطابق روش کنترلی عاری از مدل، این سیستم، به‌عنوان یک سیستم چند ورودی چند خروجی شناخته شده و با معادله (۱) نمایش می‌یابد [۲۵].

$$y(k+1) = f(y(k), \dots, y(k-n_y), u(k), \dots, u(k-n_u)) \quad (۱)$$

بردارهای $y(k) \in R^n$ و $u(k) \in R^m$ به ترتیب مقادیر محرک‌ها و خروجی‌های اندازه‌گیری شده سیستم نرم‌افزاری در چرخه تطبیق k را نشان می‌دهند و تابع f یک تابع غیرخطی است [۲۵].

روش تطبیق داده‌رانه، پویایی‌های سیستم نرم‌افزاری را با استفاده از تکنیک خطی‌سازی پویا[۲۲] [۲۸] نمایش می‌دهد اما سایر روش‌های تطبیق تئوری کنترل مبتنی بر مدل، پویایی‌های سیستم را با معادلات دیفرانسیل نشان می‌دهند[۱۶] .

روش تطبیق ارائه‌شده فرض می‌کند که سیستم نرم‌افزاری شرایط زیر را برآورده می‌نماید[۲۵] .

- مشتقات جزئی تابع غیرخطی f نسبت به بردار u(k) همواره پیوسته است.
- سیستم نرم‌افزاری شرط تعمیم‌یافته لیپشیتز[۲۳] (۲) را برآورده می‌کند.

$$\|\Delta y(k+1)\| \leq b \|\Delta u(k)\| \quad (۲)$$

پارامتر b یک مقدار ثابت است و شرط (۲) میزان تغییرات در خروجی‌های سیستم ناشی از تغییرات ورودی را محدود می‌کند. سپس این سیستم نرم‌افزاری با به‌کارگیری تکنیک خطی‌سازی پویا شکل فشرده[۲۴]، به مدل داده خطی‌سازی پویا (۳) تبدیل می‌شود.

$$\begin{aligned}&\Delta y(k+1) = \varphi(k).\Delta u(k)\\&\Delta y(k+1) = y(k+1) - y(k)\\&\Delta u(k) = u(k) - u(k-1)\\&y(k+1) = [y_1 \quad y_2 \quad \cdots \quad y_n]^T\\&\qquad = [f_1(\dots) \quad f_2(\dots) \quad \cdots \quad f_n(\dots)]\end{aligned} \tag{۳}$$

$$\varphi = \begin{bmatrix}\frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} & \cdots & \frac{\partial f_1}{\partial u_m}\\ \frac{\partial f_2}{\partial u_1} & \frac{\partial f_2}{\partial u_2} & & \frac{\partial f_2}{\partial u_m}\\ \vdots & & \ddots & \vdots\\ \frac{\partial f_n}{\partial u_1} & \frac{\partial f_n}{\partial u_2} & \cdots & \frac{\partial f_n}{\partial u_m}\end{bmatrix}_{n\times m}$$

پارامتر متغیر زمانی φ، مشتقات جزئی را نشان می‌دهد. ابعاد ماتریس مشتقات جزئی، n در m است چون بردار Δy(k+1)، n عنصر و بردار Δu(k)، m عنصر دارند. اما ابعاد این ماتریس در روش MFAC، n در n است.

مرحله ساخت کنترلر از این مدل داده خطی‌سازی پویا (۳) به‌جای مدل ریاضی استفاده می‌کند. باتوجه‌به آنکه، این مدل داده، هیچ اطلاعاتی از سیستم را حذف نمی‌کند و دقیقاً معادل با سیستم غیرخطی است [۳۱]؛ بنابراین سازوکار تطبیق در این روش می‌تواند سیستم نرم‌افزاری را به طور دقیق تطبیق دهد؛ اما رویکردهای قبلی، برای مقابله با غیرخطی‌های سیستم و تغییرات اساسی در سیستم، به سازوکارهایی نیاز داشتند.

۴-۲- ساخت کنترلر

کنترلر دارای دو مرحله اصلی است: تخمین‌گر پویایی سیستم و محاسبه پیکربندی بهینه. بخش اول، با به‌روزرسانی پویایی سیستم نرم‌افزاری با استفاده از داده‌های ورودی و خروجی، منطق تطبیق را برای مقابله با عدم‌قطعیت‌ها، تنظیم می‌کند. بخش دوم، تصمیم بهینه تطبیق را باتوجه‌به پویایی سیستم و اولویت اهداف محاسبه می‌کند.

۴-۲-۱- تخمین‌گر پویایی سیستم

این مؤلفه، مقدار پارامتر متغیر زمانی مدل داده سیستم نرم‌افزاری را مبتنی بر الگوریتم کازمارز[۲۵] یا الگوریتم تصویر، تخمین می‌زند [۳۰] [۳۳]. این الگوریتم، پارامتر متغیر زمانی(φ) یا پویایی سیستم را به فضای برداری تصویر می‌کند که توسط عناصر بردار محرک تعیین می‌شوند. شکل ۳، الگوریتم تصویر را برای یک هدف سیستم(y_i) با دو محرک نشان می‌دهد که پویایی سیستم(۴) مرتبط با هدف را روی بردارها نگاشت می‌کند.

$$\varphi_i = [\varphi_{i1} \quad \varphi_{i2}]^T = \left[\frac{\partial f_i}{\partial u_1} \quad \frac{\partial f_i}{\partial u_2}\right]^T \tag{۴}$$

الگوریتم تصویر، یک الگوریتم تکرارشونده برای تعیین مقدار پارامتر متغیر زمانی است [۳۰] که تابع هزینه (۵) مبتنی بر خطای تخمین هر گام را با درنظرگرفتن محدودیت (۶)کمینه می‌کند.

$$J(\hat{\varphi}(k)) = \frac{1}{2}\|\hat{\varphi}(k) - \hat{\varphi}(k-1)\|^2 \tag{۵}$$

$$y(k) = \hat{\varphi}(k).\Delta u(k-1) \tag{۶}$$

تابع هزینه(۸) با تعریف ضریب لاگرانژ(α) برای محدودیت (۶) و نیز جایگزینی سمت راست رابطه(۷)، به‌دست‌می‌آید.

$$\begin{aligned}&\|\hat{\varphi}(k) - \hat{\varphi}(k-1)\|^2\\&\qquad = (\hat{\varphi}(k) - \hat{\varphi}(k-1))^T(\hat{\varphi}(k)\\&\qquad - \hat{\varphi}(k-1))\end{aligned} \tag{۷}$$

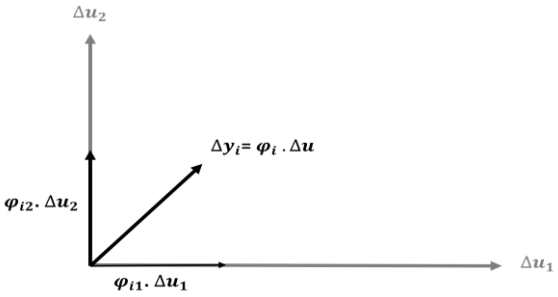
$$\begin{aligned}J_c(\hat{\varphi}(k)) = \frac{1}{2}&(\hat{\varphi}(k) - \hat{\varphi}(k-1))^T(\hat{\varphi}(k)\\&- \hat{\varphi}(k-1))\\&+ \alpha\,(\Delta y(k) - \hat{\varphi}(k).\Delta u(k-1))\end{aligned} \tag{۸}$$

برای کمینه نمودن تابع هزینه (۸)، مشتقات جزئی آن نسبت به $\hat{\varphi}(k)$ و α را مساوی صفر قرار داده و رابطه(۹) حاصل می‌شود که برای به‌روز رسانی پارامتر متغیر زمانی مورد استفاده قرار می‌گیرد.

$$\begin{aligned}\hat{\varphi}(k) = \hat{\varphi}(k-1) + &\frac{\Delta u(k-1)}{\|\Delta u(k-1)\|^2}(\Delta y(k)\\&- \hat{\varphi}(k-1).\Delta u(k-1))\end{aligned} \tag{۹}$$

۴-۲-۲- محاسبه پیکربندی بهینه

روش تطبیق داده‌رانه مشابه رویکرد تطبیق AMOCS-MA، یک مسئله بهینه‌سازی چندهدفه را با درنظرگرفتن وابستگی‌های بین محرک‌ها و اهداف حل می‌کند. روش تطبیق پیشنهادی، می‌بایست دو نوع هدف؛ نقطه‌تنظیم و بهینه‌سازی را تأمین نماید، به‌طوری‌که پیکربندی بهینه محاسبه‌شده باید مقادیر خطا اهداف نقطه‌تنظیم را باتوجه‌به اولویت آنها کمینه و مقدار اندازه‌گیری شده اهداف بهینه‌سازی را کمینه/بیشینه نماید. روش تطبیق داده‌رانه برای اهداف نقطه‌تنظیم، مقدار خطای یک گام بعدی (E(k+1)) را کمینه



شکل ۳: پویایی سیستم برای هدف Y_i در سیستم خودتطبیق با دو محرک.

می‌کند؛ اما رویکرد AMOCS-MA مقادیر خطاها روی افق زمانی را کمینه می‌کند. مقادیر بردار خطای گام بعدی ((E(k+1)) از طریق مقایسه مقادیر بردار هدف در گام k+1 با مقادیر بردار نقطه‌تنظیم (Y_d)، محاسبه می‌شوند، اما مقادیر بردار Y(k+1) در گام جاری غیرقابل‌دسترس هستند. مطابق مدل داده خطی‌سازی پویا (۳)، مقادیر این بردار را می‌توان با معادله (۱۰) با کمک مقادیر گام جاری به دست آورد.

$$y(k+1) = y(k) + \varphi(k).\Delta u(k) \qquad (۱۰)$$

تابع هزینه (۱۱) با استفاده از بردارهای وزن $W=[w_1,w_2,\ldots,w_{n1}]$ برای تعیین اهمیت اهداف نقطه‌تنظیم و $G=[g_1,g_2,\ldots,g_{n2}]$ برای تعیین اهمیت اهداف بهینه‌سازی و نیز بردار خروجی گام بعدی (Y(k+1)) تعریف می‌شود. اهداف نقطه‌تنظیم و اهداف بهینه‌سازی در تابع هزینه از یکدیگر جدا شده‌اند.

$$Minimize \sum_{i=1}^{n1} w_i \left(y_{d,i} - y_{m,i}(k+1)\right)^2 + \sum_{j=1}^{n2} \pm g_j \left(y_{m,j}(k+1)\right)^s \qquad (۱۱)$$

در تابع هزینه(۱۱)، w_i، g_j، $y_{d,i}$، $y_{m,i}$ و $y_{m,j}$ به ترتیب وزن هدف نقطه‌تنظیم i، وزن هدف بهینه‌سازی j، مقدار نقطه‌تنظیم برای هدف نقطه‌تنظیم i، مقدار خروجی هدف نقطه‌تنظیم i و مقدار خروجی هدف بهینه‌سازی j در گام بعدی هستند. n1 تعداد اهداف نقطه‌تنظیم و n2 تعداد اهداف بهینه‌سازی را نشان می‌دهند. مقدار متغیر s برای اهداف بهینه از نوع بیشینه‌سازی، ۱/۲ و علامت بخش دوم تابع هزینه برای امکان تجمیع با یک تابع کمینه، منفی است و مقدار این متغیر برای اهداف بهینه از نوع کمینه‌سازی، ۲ و علامت بخش دوم تابع هزینه، مثبت است.

بر اساس مدل داده خطی‌سازی، مقادیر خروجی در گام بعدی با سمت راست رابطه (۱۰) جایگزین می‌شوند و تابع هزینه (۱۲) به دست می‌آید.

$$Minimize \sum_{i=1}^{n1} w_i \left(y_{d,i} - y_{m,i}(k) - \hat{\varphi}_i(k).\Delta U(k)\right)^2 + \sum_{j=1}^{n2} \pm g_j \left(y_{m,j}(k) + \hat{\varphi}_j(k).\Delta U(k)\right)^s \qquad (۱۲)$$

همان‌طور که معادله (۱۳) نشان می‌دهد، عناصر بردار پویایی سیستم در مدل داده، بردارهایی هستند که پویایی مرتبط با هر هدف را نشان می‌دهد. مثلاً عناصر بردار φi در تابع هزینه (۱۲)، پویایی هدف y_i را نشان می‌دهد.

$$\varphi = [\varphi_{y_1} \quad \varphi_{y_2} \ldots . \varphi_{y_i} \ldots \cdot \quad \varphi_{y_n}]^T \qquad (۱۳)$$

با فرض ثابت‌بودن مقادیر نقطه‌تنظیم اهداف در تمام گام‌های زمانی، عبارت $(e_i(k))$ k، خطای هدف i در گام زمانی $y_{d,i} - y_{m,i}(k)$ را نشان می‌دهد و تابع هزینه را می‌توان به‌صورت تابع هزینه (۱۴) بازنویسی نمود.

$$Minimize \sum_{i=1}^{n1} w_i \left(e_i(k) - \hat{\varphi}_i(k).\Delta U(k)\right)^2 + \sum_{j=1}^{n2} \pm g_j \left(y_{m,j}(k) + \hat{\varphi}_j(k).\Delta U(k)\right)^s \qquad (۱۴)$$

$$U_{min} \leq U(K-1) + \Delta U(K) \leq U_{max} \qquad (۱۵)$$

$$\Delta U_{min} \leq \Delta U(K) \leq \Delta U_{max} \qquad (۱۶)$$

محدودیت‌های مسئله بهینه‌سازی (۱۵) و (۱۶)، مقادیر محرک‌ها و تغییرات آنها را محدود می‌نماید. همچنین محدودیت‌های مخصوص هر دامنه نیز به این مجموعه محدودیت‌ها اضافه می‌شوند.

بنابراین، روش تطبیق داده‌رانه با حل مسئله بهینه‌سازی چندهدفه از طریق کمینه نمودن تابع هزینه (۱۴)، پیکربندی بهینه را تعیین می‌کند. اگر مسئله بهینه‌سازی نتواند راه‌حلی را بیابد که تمام نیازمندی‌ها را برآورده کند، راه‌حل بهینه‌ای را ارائه می‌دهد که اهداف مهم تأمین و سایر اهداف به نقطه‌تنظیم‌هایشان نزدیک شوند. رویکرد تطبیق AMOCS-MA مشابه روش تطبیق ارائه‌شده، یک مسئله بهینه‌سازی چندهدفه را حل می‌کند؛ اما رویکرد AMOCS-MA برای حل مسئله بهینه‌سازی، می‌بایست متغیر حالت مدل ریاضی سیستم نرم‌افزاری را با استفاده از فیلتر کالمن، کمی‌سازی نماید.

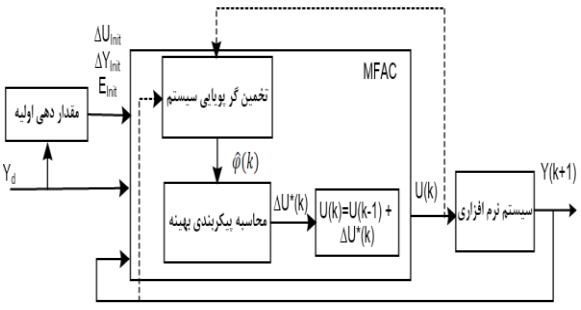
۴-۳- عملیات

شکل ۴، گام‌های مختلف مرحله عملیات را نشان می‌دهد. ابتدا گام مقداردهی اولیه، دو بردار محرک دلخواه ((U(0), U(1)) را انتخاب و به سیستم نرم‌افزاری اِعمال می‌کند. سپس بردار خطا با استفاده از بردار نقطه‌تنظیم ($E(1)=Y_d - Y(1)$) محاسبه و ماتریس پویایی‌های سیستم یا پارامتر متغیر زمانی مدل داده سیستم نرم‌افزاری به ماتریس واحد مقداردهی می‌شود.

در اولین چرخه تطبیق، مؤلفه تخمین‌گر پویایی سیستم، از بردارهای مقداردهی‌شده در فاز قبل، برای تخمین پویایی‌های سیستم ($\hat{\varphi}(0)$) بر اساس الگوریتم تصویر استفاده می‌کند. سپس، مؤلفه محاسبه پیکربندی بهینه، بردار خطا($E(1)$) و پویایی سیستم ($\hat{\varphi}(0)$) را دریافت و بردار $\Delta U^*(1)$ ، شامل مقادیر تفاضلی بهینه محرک‌ها، را تعیین می‌نماید. بردار محرک‌ها($U(1)$) چرخه تطبیق جاری از طریق افزودن مقادیر بردار محرک‌ها چرخه قبل به بردار تفاضلی به‌دست می‌آید. بردار محرک($U(1)$) شامل طرح تطبیق است که سیستم نرم‌افزاری را تطبیق می‌دهد.

مطابق شِمای کنترلی در شکل ۴، سیستم نرم‌افزاری خود تطبیق دارای دو حلقه بازخورد است که در طول اجرای سیستم در مرحله عملیات، حلقه اول برای تطبیق سیستم و حلقه دوم برای به‌روزرسانی سازوکار تطبیق بکار می‌روند؛ به‌طوری‌که، در هر چرخه تطبیق، در راستای حلقه بازخورد اول، مقادیر بردار خطا با مقایسه مقادیر بردار خروجی‌های اندازه‌گیری شده با مقادیر بردار نقطه‌تنظیم محاسبه‌شده و سپس سازوکار تطبیق با دریافت بردار خطا، طرح تطبیق را تولید می‌کند.

همچنین در راستای حلقه بازخورد دوم، مؤلفه تخمین‌گر پویایی سیستم با دریافت داده‌های ورودی و خروجی سیستم، پارامتر متغیر زمانی مدل داده سیستم را تخمین می‌زند. باتوجه‌به آنکه، تغییرات سیستم و نیز عدم‌قطعیت‌ها بر روی داده‌های ورودی و خروجی سیستم اثرگذار هستند، لذا این مؤلفه می‌تواند سازوکار تطبیق را در



شکل ۴: مرحله عملیات در روش تطبیق داده‌رانه.

شرایط عملیاتی مختلف به‌روزرسانی نماید؛ بنابراین، روش تطبیق داده‌رانه به سازوکاری برای به‌روزرسانی مدل سیستمی در طول عملیات و یا شناسایی تغییرات اساسی برای رهانا نمودن مرحله بازشناسایی، مشابه رویکردهای تطبیق قبلی نیاز ندارد.

۴-۴- تحلیل پیچیدگی زمانی:

پیچیدگی زمانی روش تطبیق داده‌رانه ارائه‌شده، شامل پیچیدگی‌های زمانی مؤلفه‌های تخمین‌گر پویایی سیستم و محاسبه پیکربندی بهینه از کنترلر است. مؤلفه تخمین‌گر پویایی سیستم، مقدار ماتریس پویایی سیستم را با رابطه (۹) تخمین می‌زند و پیچیدگی زمانی آن، خطی است. پیچیدگی زمانی محاسبه پیکربندی بهینه در مؤلفه محاسبه پیکربندی بهینه به پیچیدگی زمانی بهینه‌ساز MOSEK بستگی دارد.

رویکردهای قبلی مبتنی بر متدولوژی فشار - دکمه در مرحله ساخت مدل ریاضی، نمونه‌هایی را از سیگنال کنترلی انتخاب و به سیستم اِعمال می‌کنند. سیگنال کنترلی؛ برداری از محرک‌ها است و دامنه مقادیر محرک‌ها، فضای پیکربندی‌های سیستم نرم‌افزاری را تشکیل می‌دهند. اگر تعداد محرک‌ها، n و مجموعه مقادیر هریک از آنها دارای m مقدار متمایز باشد، آنگاه تعداد پیکربندی‌های سیستم m^n و محاسبه تمام این پیکربندی‌ها داری پیچیدگی نمایی است؛ بنابراین پیچیدگی زمانی مرحله ساخت مدل رویکردهای تطبیق تئوری کنترل مبتنی بر مدل، نمایی است.

پیچیدگی زمانی رویکردهای SimCA و SimCA* [۱] [۱۵] [۷] دارای سازوکارهایی در مرحله عملیات برای شناسایی تغییرات سیستم جهت راه‌اندازی مرحله ساخت مدل و نیز به‌روز رسانی مدل می‌باشند. پیچیدگی زمانی آنها در مرحله عملیات شامل پیچیدگی‌های زمانی سازوکارهای شناسایی نقطه تغییر، فیلتر کالمن جهت به‌روز رسانی مدل و نیز پیچیدگی الگوریتم سیمپلکس است.

پیچیدگی زمانی رویکرد AMOCS-MA [۱۸]در مرحله عملیات شامل پیچیدگی‌های زمانی فیلتر کالمن و بهینه‌ساز CVXOPT است.

پیچیدگی زمانی رویکرد تطبیق آقای صبوحی [۲۲] شامل پیچیدگی زمانی ساخت مدل شبکه عصبی برای مدل‌سازی کارایی سیستم و پیچیدگی زمانی خطی برای به‌روزرسانی پارامترهای کنترلر PID در طول مرحله عملیات سیستم خودتطبیق است. پیچیدگی زمانی رویکرد تطبیق آقای کالداس [۲۳] شامل پیچیدگی‌های زمانی مؤلفه مدیر راهبرد و مؤلفه اجراکننده راهبرد است که مرتبط با جمع‌آوری داده، مدل‌سازی، حل مسئله بهینه‌سازی با الگوریتم ژنتیک و نیز فرایند جستجو برای یافتن راهبرد مناسب می‌باشد.

پیچیدگی زمانی رویکرد تطبیق آقای وینز [۲۴] شامل پیچیدگی زمانی حلقه MAPE در سطح اول جهت پایش مقادیر اندازه‌گیری شده اهداف در هر بخش سیستم تحت مدیریت و تعیین راهبردها و

نقاط تنظیم برای کنترلرهای PI در لایه کنترلی، پیچیدگی زمانی پیش‌بینی تأثیرات عوامل عدم‌قطعیت مبتنی بر تکنیک‌های یادگیری ماشین، پیچیدگی زمانی باز پیکربندی سیستم جهت افزایش یا کاهش تعداد کانتینرهای بخش‌های سیستم تحت مدیریت است.

۴-۵- تحلیل پایداری سیستم نرم‌افزاری خود تطبیق

پایداری[۲۶] در نقطه تعادل[۲۷]، به توانایی سیستم خود تطبیق برای همگرا شدن به مقادیر نقطه‌تنظیم اهداف، اشاره دارد [۵]. روش کنترلی به‌کاررفته در روش تطبیق پیشنهادی، دو قسمت اصلی؛ تخمین پویایی سیستم و محاسبه پیکربندی بهینه دارد. با فرض آنکه در هر چرخه تطبیق، یک طرح بهینه تطبیق تعیین می‌شود، این بخش به شرایطی می‌پردازد که مطابق معادله (۱۷) پویایی سیستم می‌تواند به φ^* همگرا شود.

$$\Delta y^*(k) = \varphi^* . \Delta u(k-1) \qquad (۱۷)$$

بردار Δy^* ، تغییرات مقادیر بردار خروجی را در نقطه‌تعادل نشان می‌دهد. معادله(۱۸)، تخمین پویایی‌های سیستم است که در آن بردار Δy^* جایگزین بردار Δy، شده‌است.

$$\hat{\varphi}(k) = \hat{\varphi}(k-1) + \frac{\gamma . \Delta u(k-1)}{\xi + \|\Delta u(k-1)\|^2}\left(\varphi^* . \Delta u(k-1) - \hat{\varphi}(k-1).\Delta u(k-1)\right) \qquad (۱۸)$$

$$\hat{\varphi}(k) = \left(I - \frac{\gamma . \Delta u(k-1).\Delta u^T(k-1)}{\xi + \|\Delta u(k-1)\|^2}\right)\hat{\varphi}(k-1) + \frac{\gamma . \Delta u(k-1)}{\xi + \|\Delta u(k-1)\|^2}\varphi^* . \Delta u(k-1) \qquad (۱۹)$$

برای محاسبه خطای تخمین پویایی سیستم ($e_\varphi = \hat{\varphi} - \varphi^*$)، بردار φ^* از هر دو طرف معادله(۱۹) کسر می‌شود.

$$\hat{\varphi}(k) - \varphi^* = \left(I - \frac{\gamma . \Delta u(k-1).\Delta u^T(k-1)}{\xi + \|\Delta u(k-1)\|^2}\right)\hat{\varphi}(k-1) + \frac{\gamma . \Delta u(k-1).\Delta u^T(k-1)}{\xi + \|\Delta u(k-1)\|^2}\varphi^* - \varphi^*$$

$$\hat{\varphi}(k) - \varphi^* = \left(I - \frac{\gamma . \Delta u(k-1).\Delta u^T(k-1)}{\xi + \|\Delta u(k-1)\|^2}\right)\hat{\varphi}(k-1) - \left(I - \frac{\gamma . \Delta u(k-1).\Delta u^T(k-1)}{\xi + \|\Delta u(k-1)\|^2}\right)\varphi^*$$

$$\hat{\varphi}(k) - \varphi^* = \left(I - \frac{\gamma . \Delta u(k-1).\Delta u^T(k-1)}{\xi + \|\Delta u(k-1)\|^2}\right)(\hat{\varphi}(k-1) - \varphi^*)$$

$$e_\varphi(k) = \left(I - \frac{\gamma . \Delta u(k-1).\Delta u^T(k-1)}{\xi + \|\Delta u(k-1)\|^2}\right)e_\varphi(k-1)$$

$$A(k-1) = I - \frac{\gamma . \Delta u(k-1).\Delta u^T(k-1)}{\xi + \|\Delta u(k-1)\|^2} \qquad (۲۰)$$

معادله (۲۱)، سمت راست شکل اقلیدسی را در معادله (۲۰)، بکار می‌برد.

$$A(k-1) = I - \frac{\gamma . \Delta u(k-1).\Delta u^T(k-1)}{\xi + \Delta u(k-1).\Delta u^T(k-1)} \qquad (۲۱)$$

مقادیر ویژه ماتریس A(k-1) به‌جز مقدار ویژه(۲۲) مساوی یک است؛ بنابراین مقدار ویژه (۲۲) برای همگرایی پویایی سیستم، می‌بایست کوچک‌تر از یک باشد.

$$\lambda = \frac{\xi + (1-\gamma)\Delta u(k-1).\Delta u^T(k-1)}{\xi + \Delta u(k-1).\Delta u^T(k-1)} \qquad (۲۲)$$

$$0 < \gamma < 2 \qquad (۲۳)$$

با فرض برقراری شرط (۲۳)، پویایی سیستم به φ^* همگرا می‌شود. زمانی‌که پویایی سیستم به φ^* همگرا شد، پیکربندی نهایی محاسبه شده توسط مسئله بهینه‌سازی(۱۴)، مقادیر خروجی‌های سیستم را به مقادیر بردار نقطه‌تنظیم اهداف با توجه به ترجیحات اهداف نزدیک می‌کند؛ بنابراین، روش تطبیق پیشنهادی می‌تواند پایداری سیستم نرم‌افزاری خودتطبیق را در نقطه تعادل، تضمین نماید.

۴-۶- نحوه برخورد با عدم‌قطعیت‌ها

مطابق موج ششم تکامل خود تطبیقی، یک روش تطبیق مبتنی بر تئوری کنترل می‌بایست در خصوص رفتار سیستم در معرض انواع عدم‌قطعیت، تضمین‌هایی را ارائه نماید [۵] [۲۰].

«عدم قطعیت» در یک سیستم نرم‌افزاری خود تطبیق، به شرایطی اطلاق می‌شود که رفتار سیستم نرم‌افزاری به دلیل پویایی‌ها و رخدادهای غیرقابل‌پیش‌بینی در زمان طراحی، با رفتار مورد انتظارش مطابقت ندارد [۱] [۳۴].

مقاله [۱] و [۵]، عدم‌قطعیت‌هایی که سیستم نرم‌افزاری خودتطبیق در زمان اجرا با آنها مواجه می‌شود را به چهار گروه عمده تقسیم‌بندی می‌کند: محیط، سیستم نرم‌افزاری، نیازمندی‌ها و انسان.

سیستم نرم‌افزاری خود تطبیق مبتنی بر روش تطبیق پیشنهادی می‌تواند با عدم‌قطعیت‌های زیر در زمان اجرا مقابله نماید.

- عدم‌قطعیت در مدل: این عدم‌قطعیت به گروه عدم‌قطعیت سیستم نرم‌افزاری تعلق دارد و اشاره به عدم‌قطعیت ناشی از انتزاع مدل‌سازی و نیز مدل سیستمی غیردقیق دارد. باتوجه‌به آنکه روش تطبیق ارائه‌شده، مبتنی بر روش

کنترل تطبیقی عاری از مدل است، هیچ مدلی از سیستم نرم‌افزاری از طریق یادگیری خودکار از داده‌ها ایجاد نمی‌شود و بنابراین سیستم خود تطبیق با این نوع از عدم‌قطعیت در مرحله عملیات مواجه نمی‌شود.

- عدم‌قطعیت در محیط اجرایی: این عدم‌قطعیت به گروه عدم‌قطعیت محیط تعلق و به مسائل غیرقابل‌پیش‌بینی در محیط اجرایی اشاره دارد که روی سیستم تأثیرگذار هستند. مثلاً، فرض کنید سیستم مدیریتی در برنامه سرویس‌گرا خود تطبیق، یک پیکربندی بهینه‌ای را برای تطبیق سیستم تعیین نماید که به سرویسی نیاز دارد که به دلیل مشکلات زیرساخت غیرقابل‌دسترس است. در این حالت، به دلیل آنکه این پیکربندی بر اساس پویایی سیستم مشخص شد که در آن سرویس مورد اشاره فعال است، لذا اجرای این پیکربندی بر روی سیستم نرم‌افزاری منجر به افزایش مقدار خطا بین خروجی‌های اندازه‌گیری شده و مقادیر نقطه‌تنظیم می‌شود. روش پیشنهادی برای مقابله با این عدم‌قطعیت، منطق تطبیق اجرایی را در راستای حلقه بازخورد دوم به‌روزرسانی می‌کند. به‌عبارت‌دیگر ابتدا پویایی سیستم با استفاده از داده‌های ورودی و خروجی گام جاری تخمین زده و سپس پیکربندی جدید با استفاده از آن و خطای محاسبه‌شده، تعیین می‌شود به‌نحوی‌که سیستم بتواند به اهدافش دست یابد. در مثال فوق، پیکربندی جدید، سرویس غیرقابل‌دسترس را بکار نمی‌برد.
- عدم‌قطعیت در پارامترهای سیستم: این عدم‌قطعیت متعلق به گروه عدم‌قطعیت سیستم نرم‌افزاری است. مقادیر برخی از پارامترهای سیستم نرم‌افزاری در طول عملیات ممکن است تغییر نمایند، به‌عنوان‌مثال، در برنامه سرویس‌گرا، مقدار قابلیت اطمینان یک سرویس ممکن است به دلیل برخی از شرایط غیرقابل‌پیش‌بینی تغییر یابد. در این شرایط، فرض کنید که این تغییر پس از محاسبه پیکربندی تطبیق صورت گیرد؛ بنابراین اجرای این پیکربندی سبب افزایش مقدار خطا محاسبه‌شده در چرخه بعدی می‌شود. روش تطبیق داده‌رانه در چرخه بعدی، پویایی سیستم را با داده‌های ورودی و خروجی جدید به‌روزرسانی می‌کند و سپس با استفاده از پویایی سیستم تخمین زده و مقدار خطا محاسبه‌شده، پیکربندی جدید را تعیین می‌کند که در این پیکربندی، مقدار جدید قابلیت اطمینان سرویس مورد اشاره در نظر گرفته‌شده است.

رویکرد تطبیق SimCA*، در رویکردهای قبلی به این عدم‌قطعیت از طریق شناسایی تغییرات اساسی در پارامترهای سیستم، با کمک یک الگوریتم شناسایی نقطه تغییر، پرداخته است. انتخاب الگوریتم شناسایی نقطه تغییر مناسب، وابسته به دامنه سیستم و تکنیک‌های داده‌کاوی است.

- عدم‌قطعیت در نیازمندی‌ها: روش تطبیق پیشنهادی، از تغییرات در مجموعه نیازمندی‌ها شامل افزودن و حذف نیازمندی، تغییر مقدار نقطه‌تنظیم و نیز تبدیل هدف نقطه‌تنظیم به بهینه‌سازی و بالعکس پشتیبانی می‌نماید. برای این منظور، مطابق شکل ۵، مرحله «به‌روزرسانی سازوکار تطبیق» به مرحله عملیات، اضافه می‌شود که منطق تطبیق اجرایی را باتوجه‌به نوع تغییر به‌روزرسانی می‌کند. هر تغییر در نیازمندی‌ها، مؤلفه «مدیر تغییرات نیازمندی» را فعال نموده و باتوجه‌به نوع تغییر، یکی از مؤلفه‌های مرحله به‌روزرسانی سازوکار تطبیق، فعال می‌شود.

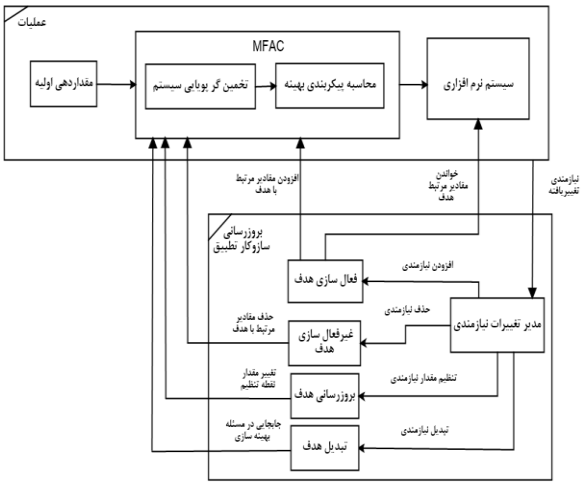
زمانی که یک نیازمندی به مجموعه نیازمندی‌ها، اضافه می‌شود، مؤلفه «فعال‌سازی هدف» مقدار نقطه‌تنظیم و وزن آن هدف را دریافت و کنترلر را با انجام مراحلی تنظیم می‌نماید. ابتدا، پارامترهای مرتبط باهدف جدید را مقداردهی اولیه می‌نماید، به‌طوری‌که مقادیر بردار پویایی سیستم مربوط به هدف جدید را به مقدار یک و مقدار خروجی اولیه آن را به صفر مقداردهی می‌کند. سپس بردارهای نقطه‌تنظیم، وزن، خروجی و نیز ماتریس شبه مشتق جزئی را با افزودن مقادیر نقطه‌تنظیم، وزن و خروجی و بردار پویایی سیستم مرتبط باهدف جدید، به‌روزرسانی می‌نماید.

مؤلفه «غیرفعال‌سازی هدف» با حذف یک نیازمندی از مجموعه نیازمندی‌ها، راه‌اندازی شده و کنترلر را با حذف مقادیر متناظر با آن هدف از بردارها و ماتریس پویایی سیستم، تنظیم می‌کند.

درصورتی‌که، مقدار نقطه‌تنظیم یکی از نیازمندی‌ها، تغییر نماید، مؤلفه «به‌روزرسانی کننده هدف»، مقدار آن را در بردار نقطه‌تنظیم به‌روزرسانی می‌کند.

اگر یک هدف نقطه‌تنظیم به یک هدف بهینه‌سازی تبدیل شود و بالعکس، مؤلفه «تبدیل هدف»، تابع هزینه (۱۴) مسئله بهینه‌سازی را به نحوی تغییر می‌دهد که هدف نقطه‌تنظیم از بخش اول تابع حذف و به بخش دوم اضافه شود و بالعکس.

روش تطبیق داده‌رانه می‌تواند با عدم‌قطعیت در نیازمندی‌ها در مرحله عملیات مواجه شود؛ اما رویکردهای تطبیق قبلی برای هدف



شکل ۵: برخورد با تغییرات نیازمندی در سیستم نرم‌افزاری خودتطبیق.

جدید اضافه‌شده، می‌بایست مرحله ساخت مدل را رهانا نمایند.

۵- ارزیابی روش تطبیق پیشنهادی

ابتدا روش تطبیق پیشنهادی با استفاده از دو موردمطالعه در دو دامنه کاربردی مختلف، ارزیابی و سپس با سایر روش‌های تطبیق مبتنی بر تئوری کنترل مقایسه می‌شود.

بخش اول، به معرفی سازوکار انقیاد پویا چندهدفه در سیستم نرم‌افزاری سرویس‌گرا به‌عنوان اولین موردمطالعه می‌پردازد.

روش تطبیق در این موردمطالعه، هر درخواست دریافتی را به یک سرویس مناسب انقیاد می‌کند، تا اهداف سیستم نرم‌افزاری خودتطبیق تحت شرایط عملیاتی مختلف تأمین شوند.

بخش دوم، سیستم زیردریایی بدون سرنشین[۲۸] خود تطبیق را به‌عنوان دومین موردمطالعه شرح می‌دهد. روش تطبیق داده‌رانه پیشنهادی از طریق فعال و غیرفعال نمودن حس‌گرها، سیستم زیردریایی را تطبیق می‌دهد. آزمایش این موردمطالعه، به بررسی عملکرد روش پیشنهادی در شرایطی می‌پردازد که یکی از نیازمندی‌ها در زمان اجرا حذف می‌شود.

بخش سوم، روش پیشنهادی را با یکی از رویکردهای تطبیق مبتنی بر مدل، در دامنه کاربردی انقیاد پویا چندهدفه مقایسه می‌کند که تصمیمات تطبیق را با حل یک مسئله بهینه‌ساز چندهدفه اتخاذ می‌نماید. همچنین روش تطبیق داده‌رانه پیشنهادی، با روش تطبیق داده‌رانه‌ای در دامنه کاربردی سیستم زیردریایی، مقایسه می‌شود که از تبدیل اهداف نقطه‌تنظیم و بهینه‌سازی در زمان اجرا پشتیبانی نمی‌کند.

۵-۱- سازوکار انقیاد پویا چندهدفه

سازوکار انقیاد پویا، قابلیت خود تطبیقی را در سیستم نرم‌افزاری سرویس‌گرا پیاده‌سازی می‌کند [۲۶]. این سازوکار ، واسط‌های انتزاعی سرویس را به سرویس‌های واقعی ارائه‌شده توسط فراهم‌کنندگان تخصیص می‌دهد [۱۷] [۱۸].

آزمایش‌های این موردمطالعه با به‌کارگیری شبیه‌سازی سیستم نرم‌افزاری سرویس‌گرا به‌عنوان سیستم تحت مدیریت انجام می‌شوند که ماگیو و همکارانش در پایتون پیاده‌سازی نمودند [۱۸].

سیستم نرم‌افزاری، هر درخواست ورودی را با یکی از سرویس‌های واقعی پردازش می‌کند که عملکرد یکسان اما ویژگی‌های کیفی متفاوتی دارند [۱۸]. جدول ۳، مشخصات سرویس‌ها را نشان می‌دهد.

سیستم نرم‌افزاری دارای محرک‌هایی برای تعیین احتمال انتخاب سرویس‌ها و سطح سرویس آنها است. عناصر اول و دوم بردار محرک در این مطالعه موردی، احتمال انتخاب سرویس‌های اول و دوم و بقیه عناصر، سطوح سرویس را نشان می‌دهند.

جدول ۳: مشخصات سرویس‌ها [۱۸].

سرویس‌های واقعی	قابلیت اطمینان (درصد)	کارایی (میلی‌ثانیه)	هزینه (دلار)
سرویس اول	۹۰	۲/۰	۱۵/۰
سرویس دوم	۶۵	۱۰/۰	۱۰/۰
سرویس سوم	۴۵	۲۰/۰	۵/۰

انتخاب سرویس برای پردازش درخواست ورودی، با انتخاب تصادفی عددی بین صفر ویک صورت می‌گیرد. اگر مقدار عدد تصادفی کمتر از احتمال انتخاب یک سرویس باشد، آن سرویس درخواست را پردازش می‌کند. هزینه پردازش هر درخواست، حاصل‌ضرب سطح سرویس و هزینه سرویس است. زمان پردازش هر درخواست از توزیع t/sl^2 نمونه‌برداری می‌شود و متغیرهای t و sl ، مقادیر کارایی و سطح سرویس را نشان می‌دهند. زمان پردازش سرویس با افزایش سطح سرویس، کاهش ولی هزینه پردازش آن افزایش می‌یابد. وضعیت پردازش یک سرویس (موفق/ناموفق)، بر اساس قابلیت اطمینان سرویس و با انتخاب تصادفی عددی بین صفر ویک تعیین می‌شود. اگر مقدار عدد تصادفی کمتر از میزان قابلیت اطمینان سرویس باشد، پردازش سرویس موفق و در غیر این صورت ناموفق است. میزان پردازش موفق سرویس با افزایش قابلیت اطمینان افزایش می‌یابد؛ اما باتوجه‌به جدول مشخصات سرویس‌ها، هزینه سرویس نیز افزایش می‌یابد.

سیستم نرم‌افزاری خود تطبیق مجهز به حس‌گرهایی برای اندازه‌گیری مقادیر خروجی‌های سیستم پس از پردازش درخواست‌ها است.

روش تطبیق پیشنهادی، سیستم نرم‌افزاری را به یک کنترلر داده‌رانه

تطبیقی عاری از مدل، مجهز نموده که مقادیر احتمال انتخاب سرویس‌ها و سطح سرویس‌ها را تعیین می‌نماید.

سیستم نرم‌افزار خود تطبیق می‌بایست در مرحله عملیات، نیازمندی‌های زیر را تأمین نماید.

- نیازمندی اول: قابلیت اطمینان سیستم نرم‌افزاری عبارت است از نسبت تعداد درخواست‌های موفق به تعداد کل درخواست‌ها و مقدار آن می‌بایست ۶۵ درصد باشد.
- نیازمندی دوم: کارایی یا زمان پاسخ انتها به انتها می‌بایست ۵ میلی‌ثانیه باشد.
- نیازمندی سوم: با فرض برآورده‌شدن نیازمندی‌های اول و دوم، هزینه پردازش درخواست‌ها می‌بایست کمینه شود.

سیستم نرم‌افزاری خود تطبیق در این موردمطالعه باید در زمان اجرا با عدم‌قطعیت‌ها در محیط اجرا (خرابی سرویس)، پارامترهای سیستم (تغییر در مشخصات سرویس‌ها) و عدم‌قطعیت در نیازمندی‌ها (تغییرات نقطه‌تنظیم‌ها) مقابله و از تبدیل اهداف نقطه‌تنظیم به بهینه‌سازی و بالعکس پشتیبانی نماید.

در گام مقداردهی اولیه مطابق شکل ۴، دو بردار محرک دلخواه انتخاب و به سیستم نرم‌افزاری اِعمال شده و مقادیر قابلیت اطمینان، کارایی و هزینه اندازه‌گیری می‌شوند.

باتوجه‌به آنکه مقادیر عناصر بردار محرک مرتبط با سطوح سرویس، عدد صحیح و مقادیر عناصر بردار محرک مرتبط با احتمال انتخاب سرویس، عدد حقیقی هستند، لذا سیستم نرم‌افزاری خود تطبیق می‌بایست یک مسئله برنامه‌ریزی عدد صحیح مختلط (MIP)[۲۹] را حل نماید. برنامه‌ریزی عدد صحیح مختلط، یک مسئله بهینه‌سازی است که برخی از متغیرهای تصمیم در آن، مقادیر صحیح می‌گیرند [۳۵]. سیستم نرم‌افزاری خودتطبیق از بهینه‌ساز MOSEK [۳۶] برای حل مسئله بهینه‌سازی MIP استفاده می‌کند. عناصر صحیح و حقیقی در تابع هزینه(۲۴) مسئله بهینه‌سازی از یکدیگر جدا شده‌اند.

$$\underset{\substack{\Delta U_{1\dots p}(k)\\ \Delta U_{p+1\dots sl}(k)}}{Minimize} \sum_{i=1}^{n1} w_i * \Big(e_i(k) - (\hat{\varphi}_{i,1\dots p}(k).\Delta U_{1\dots p}(k) + \hat{\varphi}_{i,p+1\dots sl}(k).\ \Delta U_{p+1\dots sl}(k)\Big)^2 + \sum_{j=1}^{n2} \pm g_j * \Big(Y_j(k) + \Big(\hat{\varphi}_{j,1\dots p}(k).\Delta U_{1\dots p}(k) + \hat{\varphi}_{j,p+1\dots sl}(k).\ \Delta U_{p+1\dots sl}(k)\Big)\Big)^s \quad (۲۴)$$

با درنظرگرفتن محدودیت‌های زیر:

$$\Delta U_{1\dots p}(K) \in \mathbb{R}$$

$$0 < U_{1\dots p}(K-1) + \Delta U_{1\dots p}(K) < 1$$

$$\Delta U_{p+1\dots sl}(K) \in \mathbb{Z}$$

$$1 \le U_{p+1\dots sl}(K-1) + \Delta U_{p+1\dots sl}(K) \le 5$$

$$n1 + n2 = n$$

عناصر بردار U[1..p]، احتمال انتخاب سرویس‌ها و مقادیر آنها بین صفر و یک هستند. مقادیر عناصر بردار U[p+1…sl] بین یک و پنج و نشانگر سطوح سرویس‌ها هستند. پویایی‌های سیستم نرم‌افزاری مرتبط باهدف i ام برای محرک‌های عدد حقیقی و محرک‌ها عدد صحیح به ترتیب توسط $\hat{\varphi}_{i,1\dots p}$ و $\hat{\varphi}_{i,p+1\dots sl}$ نمایش می‌یابند.

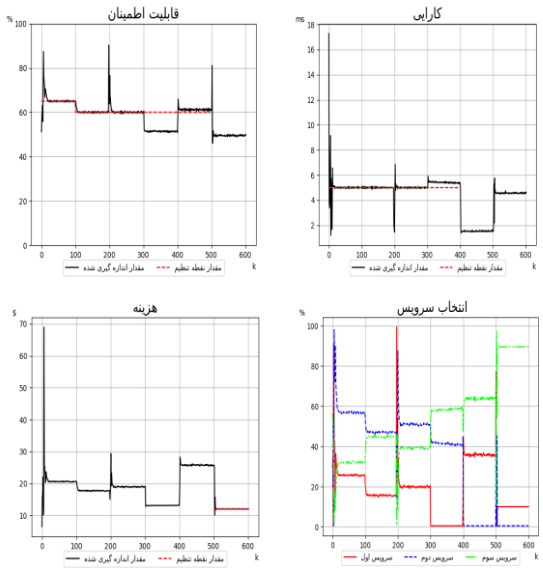
شکل ۶، نتایج آزمایش در مرحله عملیات را پس از اِعمال شِمای کنترلی نشان می‌دهد. سیستم نرم‌افزاری خود تطبیق در حدود گام ۱۷، پایدار می‌شود و مقدار قابلیت اطمینان، ۶۵ درصد، کارایی،۵ میلی‌ثانیه و هزینه ۲۰ واحد هستند. سرویس اول با احتمال ۲۶ درصد، سرویس دوم با احتمال ۵۸ درصد و سرویس سوم با احتمال ۳۱ درصد فراخوانی می‌شوند.

برای ارزیابی روش پیشنهادی در مواجهه با عدم‌قطعیت در نیازمندی‌ها، در گام ۱۰۰، مقدار نقطه‌تنظیم نیازمندی اول یا قابلیت اطمینان به ۶۰ درصد کاهش می‌یابد. سیستم نرم‌افزاری خود تطبیق در این شرایط نیز می‌تواند به اهداف خود دست یابد. برای بررسی عملکرد سیستم نرم‌افزاری خود تطبیق در شرایط عدم‌قطعیت در پارامترهای سیستم، در گام ۲۰۰، مقدار قابلیت اطمینان سرویس دوم به ۶۰ درصد کاهش می‌یابد؛ بنابراین میزان استفاده از سرویس اول از ۱۵ به ۲۰ درصد افزایش می‌یابد، زیرا این سرویس میزان قابلیت اطمینان بیشتری را برای حفظ مقدار قابلیت اطمینان سیستم نرم‌افزاری، فراهم می‌کند.

سرویس اول در گام ۳۰۰ تا ۴۰۰، به علت عدم‌قطعیت در محیط اجرا غیرقابل‌دسترس است. در این شرایط، سرویس سوم با احتمال ۶۰ درصد فراخوانی می‌شود. باتوجه‌به آنکه میزان قابلیت اطمینان این سرویس کمتر از مقدار نقطه‌تنظیم است، بنابراین مقدار قابلیت اطمینان اندازه‌گیری شده (۵۰ درصد) برای سیستم خود تطبیق در نقطه تعادل کمتر از مقدار نقطه‌تنظیم است.

در گام ۴۰۰، برای بررسی نحوه پشتیبانی روش تطبیق پیشنهادی از تبدیل نوع اهداف به یکدیگر، نوع هدف کارایی از نقطه‌تنظیم به بهینه‌سازی تبدیل می‌شود؛ بنابراین مجموعه اهداف سیستم نرم‌افزاری خود تطبیق به‌صورت زیر تعریف می‌شود.

- نیازمندی اول: مقدار قابلیت اطمینان سیستم نرم‌افزاری می‌بایست ۶۰ درصد باشد.
- نیازمندی دوم: با فرض برآورده‌شدن نیازمندی اول، مقدار کارایی یا زمان پاسخ انتها به انتها می‌بایست کمینه شود.
- نیازمندی سوم: با فرض برآورده‌شدن نیازمندی اول، هزینه



شکل ۶: آزمایش‌های مورد مطالعه اول، ساز وکار اقیاد پویا چند هدفه در سیستم نرم‌افزاری سرویس‌گرا.

پردازش درخواست‌ها می‌بایست کمینه شود.

در این حالت، مقادیر بهینه کارایی و هزینه به ترتیب ۱٫۶ میلی‌ثانیه و ۲۶ واحد هستند.

سیستم نرم‌افزاری در گام ۵۰۰، می‌بایست نیازمندی‌های اول و دوم را با هزینه محدود ۱۲ واحد به نحوی تأمین نماید که مقدار قابلیت اطمینان بیشینه و مقدار کارایی یا زمان پاسخ انتها به انتها کمینه شود؛ بنابراین نیازمندی سوم به نقطه‌تنظیم تبدیل می‌شود. در این حالت، اهداف سیستم نرم‌افزاری خود تطبیق به‌صورت زیر تغییر می‌یابند.

مقادیر بهینه قابلیت اطمینان و کارایی ۵۰ درصد و ۴٫۵ میلی‌ثانیه اندازه‌گیری می‌شوند.

- نیازمندی سوم: مقدار هزینه پردازش درخواست‌ها می‌بایست ۱۲ واحد باشد.
- نیازمندی اول: با فرض برآورده‌شدن نیازمندی سوم، مقدار کارایی یا زمان پاسخ انتها به انتها کمینه شود.
- نیازمندی دوم: با فرض برآورده‌شدن نیازمندی سوم، مقدار قابلیت اطمینان بیشینه شود.

۵-۲- سیستم زیردریایی بدون سرنشین

سیستم زیردریایی برای انجام مأموریت در محیط اقیانوس بکار می‌رود و در معرض انواع عدم‌قطعیت‌ها است[۲۷] . شبیه‌سازی سیستم زیردریایی به‌عنوان مورد مطالعه برای ارزیابی رویکرد تطبیق SimCA توسط آقایان شوتسوف و وینز در جاوا پیاده‌سازی و مورد استفاده قرار گرفت [۷] اما در این مورد مطالعه برای ارزیابی روش تطبیق پیشنهادی، با پایتون پیاده‌سازی شد.

سیستم زیردریایی مجهز به پنج حس‌گر است که می‌توانند ویژگی معینی از اقیانوس مثلاً جریان آب را اندازه‌گیری نماید. هر حس‌گر، با سرعت و دقت مشخصی، سطح را پیمایش و میزان مشخصی از انرژی را مصرف می‌کند[۷] . جدول ۴ مشخصات حس‌گرها را نشان می‌دهد. دقت اندازه‌گیری حس‌گرها با مصرف انرژی بیشتر، افزایش می‌یابد؛ اما با افزایش سرعت پایش حس‌گر، میزان دقت اندازه‌گیری با کاهش مصرف انرژی، کاهش می‌یابد.

سیستم زیردریایی برای مقابله با عدم‌قطعیت‌های مختلف می‌بایست خود تطبیق باشد. روش تطبیق پیشنهادی، سیستم زیردریایی را به یک کنترلر تطبیقی عاری از مدل مجهز می‌کند که در طول عملیات، حس‌گرها را باتوجه‌به شرایط اجرایی، روشن یا خاموش می‌کند. با فرض آنکه، فقط یک حس‌گر در هر زمان فعال است، روش تطبیق می‌تواند ترکیبی از حس‌گرها را در هر چرخه تطبیق استفاده نماید. سیستم زیردریایی خود تطبیق مجهز به نظاره‌گرهایی[۳۰] است که مقادیر سرعت پایش، مصرف انرژی و دقت اندازه‌گیری‌های سیستم زیردریایی را اندازه‌گیری می‌کند. این سیستم باید در مرحله عملیات، نیازمندی‌های زیر را برآورده نماید.

- نیازمندی اول: سطحی معادل با ۱۱۲ کیلومتر در زمان

جدول ۴: پارامترهای حس‌گرها در سیستم زیردریایی[۷] .

حس‌گرها	مصرف انرژی (ژول بر ثانیه)	سرعت پایش (متر بر ثانیه)	دقت (درصد)
حس‌گر اول	۱۷۰	۲٫۶	۹۷
حس‌گر دوم	۱۳۵	۳٫۶	۸۹
حس‌گر سوم	۱۱۸	۲٫۶	۸۳
حس‌گر چهارم	۱۰۰	۳٫۰	۷۴
حس‌گر پنجم	۷۸	۳٫۶	۴۹

مفروض ۱۰ ساعت، باید مورد بررسی قرار گیرد. این نیازمندی مرتبط با سرعت پایش برای بررسی سطح تعیین‌شده است و مقدار نقطه‌تنظیم آن ۳٫۱ متر بر ثانیه است.

- نیازمندی دوم: میزان انرژی ۴٫۳۲ مگاژول برای انجام مأموریت، در دسترس است. این نیازمندی مرتبط با مصرف انرژی است و مقدار نقطه‌تنظیم آن ۱۲۰ ژول بر ثانیه است.
- نیازمندی سوم: با فرض تأمین نیازمندی‌های اول و دوم، دقت اندازه‌گیری‌ها می‌بایست بیشینه شود.

سیستم زیردریایی خود تطبیق در این آزمایش دارای دو حالت عملیاتی است؛ سیستم در حالت عملیاتی عادی، می‌بایست مجموعه

نیازمندی‌های اول، دوم و سوم را تأمین نماید. سیستم زیردریایی ممکن است در طول عملیات با نشت ناگهانی انرژی مواجه شود و در این شرایط، می‌بایست به حالت عملیاتی دیگری انتقال یابد که در آن حالت، می‌بایست به‌جای بیشینه نمودن دقت اندازه‌گیری‌ها، مقدار مصرف انرژی کمینه شود؛ بنابراین نیازمندی سوم از مجموعه نیازمندی‌های سیستم خود تطبیق حذف شده و نیازمندی دوم جدید به‌صورت زیر تعریف می‌شود.

- نیازمندی دوم جدید: با فرض تأمین نیازمندی اول، مصرف انرژی می‌بایست کمینه شود.

سیستم زیردریایی خود تطبیق در طول مأموریت خود با عدم‌قطعیت در محیط (خرابی یک حس‌گر) و عدم‌قطعیت در نیازمندی‌ها (تغییر نقطه‌تنظیم و حذف یک نیازمندی) مواجه می‌شود و از تبدیل هدف نقطه‌تنظیم به هدف بهینه‌سازی نیز پشتیبانی می‌کند.

سیستم زیردریایی خود تطبیق، تابع هزینه (۲۵) را برای محاسبه تنظیمات بهینه، کمینه می‌کند. محدودیت (۲۶) در این تابع هزینه، فعال‌بودن یک حس‌گر در هر لحظه زمانی را تضمین می‌کند.

$$\underset{\Delta U(k)}{Minimize} \sum_{i=1}^{n1} w_i \left(e_i(k) - \hat{\varphi}_i(k).\Delta U(k) \right)^2 + \sum_{j=1}^{n2} \pm g_j \left(Y_j(k) + \hat{\varphi}_j(k).\Delta U(k) \right)^s \quad (۲۵)$$

با درنظرگرفتن محدودیت‌های زیر:

$$\begin{aligned} & U_{min} \le U(K-1) + \Delta U(K) \le U_{max} \\ & \Delta U_{min} \le \Delta U(K) \le \Delta U_{max} \\ & \sum_{j=1}^{m} u_j(k-1) + \Delta u_j(k) = 1 \\ & n1 + n2 = n \end{aligned} \quad (۲۶)$$

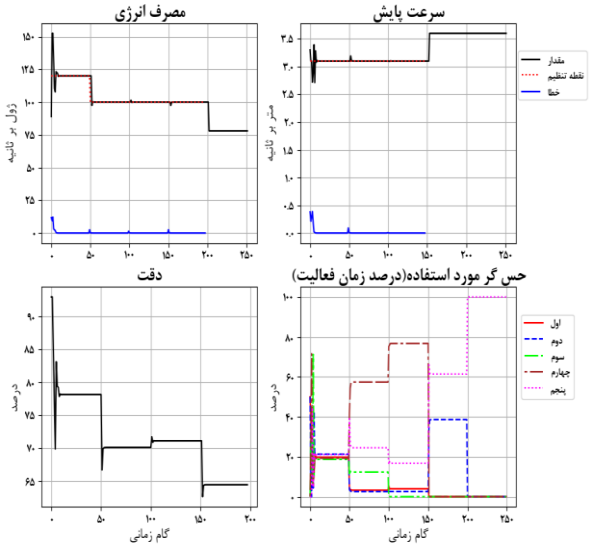
بردار U، درصد زمانی فعال‌بودن هر حس‌گر را در طول عملیات سیستم نشان می‌دهد؛ به‌طوری‌که مقدار u_j، درصد زمان فعال‌بودن حس‌گر j است.

مطابق نتایج آزمایش در شکل ۷، سیستم خود تطبیق به پایداری می‌رسد و مقدار سرعت پایش اندازه‌گیری شده، ۳٫۱ متر بر ثانیه است که پیمایش مقدار سطح تعیین‌شده در نیازمندی اول را تضمین می‌کند. میزان مصرف انرژی، ۱۲۰ ژول بر ثانیه و دقت اندازه‌گیری‌ها ۷۸ درصد است و حس‌گرها حدود ۲۰ درصد زمان فعال هستند.

برای ارزیابی روش پیشنهادی در تطبیق سیستم زیردریایی در شرایط عدم‌قطعیت در مجموعه نیازمندی‌ها، مقدار نقطه‌تنظیم نیازمندی دوم یا مصرف انرژی به ۱۰۰ ژول بر ثانیه در گام ۵۰ کاهش می‌یابد. کاهش مصرف انرژی، دقت اندازه‌گیری‌ها را به ۷۰ درصد کاهش می‌دهد، زیرا سیستم خود تطبیق از حس‌گرهایی استفاده می‌کند که انرژی کمتری را مصرف می‌کنند و این حس‌گرها، دقت اندازه‌گیری کمتری نیز دارند.

حس‌گر اول به دلیل عدم‌قطعیت در محیط، در گام ۱۰۰، خراب می‌شود و روش تطبیق برای مواجهه با این عدم‌قطعیت، میزان استفاده از حس‌گر چهارم را به ۷۷ درصد افزایش می‌دهد.

سیستم زیردریایی خود تطبیق برای پیمایش سطح بیشتری بادقت



شکل ۷: آزمایش‌های مورد مطالعه دوم، تطبیق سیستم زیردریایی.

بالا در گام ۱۵۰، هدف مرتبط با سرعت‌پیمایش را به یک هدف بهینه‌سازی تبدیل می‌کند. در این حالت، روش تطبیق از حس‌گرهای دوم و پنجم استفاده نموده و سرعت‌پیمایش به ۳٫۶ متر بر ثانیه افزایش می‌یابد و دقت اندازه‌گیری‌ها نیز ۶۴ درصد است.

سیستم زیردریایی در گام ۲۰۰، یک نشت شدید انرژی را شناسایی می‌کند، بنابراین به حالت عملیاتی دوم خود انتقال می‌یابد که در آن نیازمندی سوم یا دقت اندازه‌گیری‌ها حذف شده و نیازمندی دوم یا مصرف انرژی به هدف بهینه‌سازی تبدیل می‌شود. سیستم زیردریایی خود تطبیق در این شرایط فقط از حس‌گر پنجم استفاده می‌کند؛ زیرا کمترین میزان مصرف انرژی را دارد و بنابراین میزان مصرف انرژی به ۷۸ ژول بر ثانیه کاهش می‌یابد.

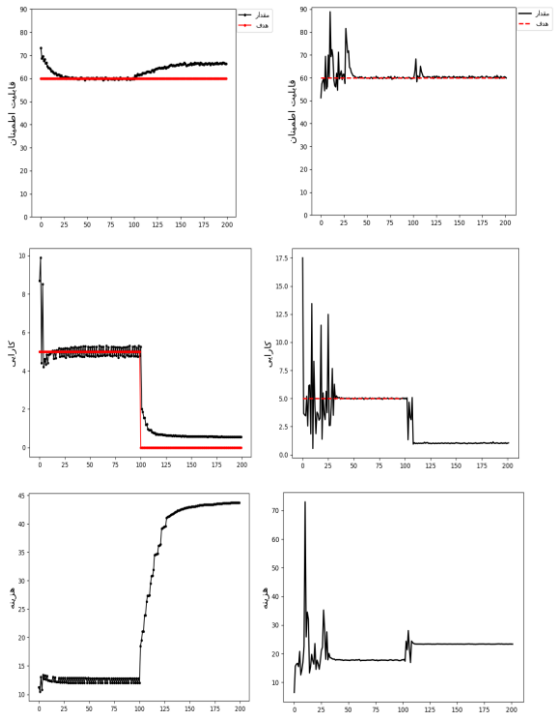
۵-۳- مقایسه روش تطبیق پیشنهادی با سایر روش‌های تطبیق

ابتدا روش تطبیق پیشنهادی با رویکرد تطبیق AMOCS-MA مقایسه می‌شود که مبتنی بر اصول کنترل پیشگویانه مدل گرا است [۱۸]. بخش اصلی کنترلر پیشگویانه مدل‌گرا، یک بهینه‌ساز چندهدفه است که یک راه‌حل بهینه را با درنظر گرفتن وابستگی‌ها بین تمام محرک‌ها و اهداف، محاسبه می‌کند [۱۸].

شکل ۸، مقایسه روش تطبیق پیشنهادی و رویکرد AMOCS-MA را نشان می‌دهد. سمت چپ شکل، نتایج تطبیق را برای رویکرد تطبیق مبتنی بر مدل و سمت راست، نتایج به‌کارگیری روش تطبیق داده‌رانه را نشان می‌دهد.

سیستم‌های خود تطبیق نرم‌افزاری در هر دو روش تطبیق، می‌بایست نیازمندی‌های شرح داده شده در اولین موردمطالعه را برآورده نمایند. نقطه‌تنظیم نیازمندی‌های اول (قابلیت اطمینان) و دوم (کارایی) به ترتیب ۶۰ درصد و ۵ میلی‌ثانیه هستند.

برای بررسی عملکرد هر دو روش تطبیق در تأمین چندین نیازمندی



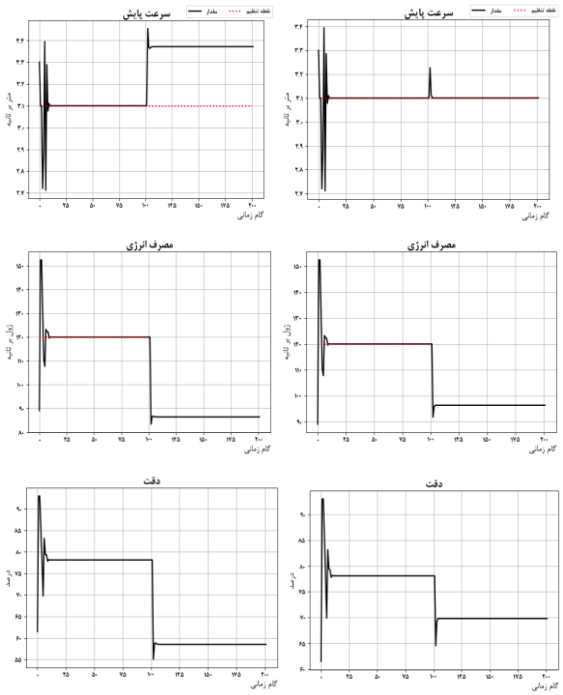
شکل ۸: مقایسه روش تطبیق داده‌رانه پیشنهادی(راست) و رویکرد تطبیق مبتنی بر مدل(چپ).

بهینه‌سازی، نیازمندی دوم (کارایی) در گام ۱۰۰ به هدف بهینه‌سازی در روش تطبیق پیشنهادی تبدیل می‌شود. باتوجه‌به آنکه رویکرد تطبیق مبتنی بر مدل از تبدیل اهداف نقطه‌تنظیم به بهینه‌سازی و بالعکس پشتیبانی نمی‌کند، برای کمینه نمودن مقدار نیازمندی کارایی، مقدار نقطه‌تنظیم این هدف در گام ۱۰۰، به صفر کاهش می‌یابد.

باتوجه‌به نتایج نشان‌داده‌شده در شکل ۸، رویکرد تطبیق مبتنی بر مدل، مقدار کارایی را پس از گام زمانی ۱۰۰ کاهش می‌دهد؛ اما مقدار هزینه آن بسیار زیاد است و مقدار قابلیت اطمینان نیز فراتر از مقدار هدف تعیین‌شده است. نتایج روش تطبیق پیشنهادی نشان می‌دهد که این روش، نیازمندی کارایی را با صرف هزینه کمتر تقریباً نصف هزینه موردنیاز در رویکرد AMOCS-MA کمینه می‌نماید و نیازمندی مرتبط با قابلیت اطمینان نیز به هدف تعیین‌شده می‌رسد.

همچنین در دامنه سیستم زیردریایی، روش تطبیق پیشنهادی با روش تطبیقی مقایسه می‌شود که مشابه روش ارائه‌شده مبتنی بر روش کنترلی عاری از مدل است؛ اما از تبدیل اهداف نقطه‌تنظیم به بهینه‌سازی و بالعکس پشتیبانی نمی‌کند که تحت عنوان روش تطبیق مبنا در این مقایسه مورد ارجاع قرار می‌گیرد.

سیستم زیردریایی خود تطبیق مبتنی بر هر دو روش تطبیق، در ابتدا آزمایش می‌بایست نیازمندی‌های بیان‌شده در دومین مطالعه موردی را برآورده نمایند.



شکل ۹: مقایسه روش تطبیق داده‌رانه پیشنهادی(راست) و رویکرد تطبیق داده‌رانه مبنا(چپ).

نیازمندی مصرف انرژی در گام ۱۰۰ در سیستم زیردریایی خود تطبیق مبتنی بر روش پیشنهادی، به بهینه‌سازی تبدیل می‌شود و روش تطبیق مبنا برای کمینه نمودن مصرف انرژی، نقطه‌تنظیم این نیازمندی را به صفر کاهش می‌دهد.

شکل ۹ مقایسه روش تطبیق پیشنهادی و روش تطبیق مبنا را نشان می‌دهد. سمت چپ شکل، نتایج تطبیق را برای روش تطبیق مبنا و سمت راست شکل، نتایج به‌کارگیری روش تطبیق پیشنهادی را نشان می‌دهد.

روش تطبیق مبنا، بعد از گام ۱۰۰، مقدار مصرف انرژی (۸۶ ژول بر ثانیه) کمتری نسبت به روش تطبیق ارائه‌شده دارد، اما دقت اندازه‌گیری‌های آن (۵۸٪) کمتر است و نیازمندی سرعت پایش نیز نمی‌تواند به هدف خود دست یابد.

سیستم زیردریایی خود تطبیق مبتنی بر روش تطبیق پیشنهادی بعد از گام ۱۰۰ می‌تواند به نقطه‌تنظیم تعیین‌شده برای نیازمندی سرعت پایش برسد و همچنین مقدار دقت اندازه‌گیری‌ها (۷۰٪) را افزایش و میزان مصرف انرژی (۹۶ ژول بر ثانیه) را کاهش می‌دهد.

۶- نتیجه‌گیری

یک روش تطبیق چندهدفه با قابلیت تبدیل نوع هدف و مبتنی بر روش کنترل تطبیق عاری از مدل برای تطبیق سیستم نرم‌افزاری در این مقاله ارائه شد به‌طوری‌که سیستم نرم‌افزاری خود تطبیق مبتنی بر این روش تطبیق می‌تواند چندین نیازمندی غیرعملکردی را علی‌رغم انواع مختلف عدم‌قطعیت برآورده نماید.

روش تطبیق ارائه‌شده برخلاف رویکردهای قبلی مبتنی بر متدولوژی فشار - دکمه، نیازی به ساخت مدل ریاضی طی مرحله ساخت مدل، نداشت و سیستم نرم‌افزاری در این روش تطبیق به یک مدل داده خطی‌سازی پویا با استفاده از تکنیک خطی‌سازی پویا تبدیل می‌شد. مدل داده خطی‌سازی پویا، نمایش دقیقی از سیستم نرم‌افزاری را ارائه می‌دهد؛ زیرا سیستم‌های خطی و غیرخطی در سطح داده یکسان هستند.

روش تطبیق داده‌رانه با به‌روزرسانی پارامتر متغیر زمانی مدل داده سیستم نرم‌افزاری در هر چرخه تطبیق، می‌توانست با تغییرات ناشی از عدم‌قطعیت‌ها برخورد نماید و در نتیجه این روش نسبت به رویکردهای قبلی در برابر عدم‌قطعیت‌ها، مقاومت بیشتری دارد و نیز سازوکارهایی را برای مواجه با عدم‌قطعیت در نیازمندی‌های سیستم نرم‌افزاری خودتطبیق دارد.

عملیات تطبیق در هر چرخه تطبیق از طریق حل یک مسئله بهینه‌سازی تعیین می‌شد. تابع هزینه این مسئله بهینه‌سازی شامل دو بخش است که بخش اول به اهداف نقطه‌تنظیم و بخش دوم به اهداف بهینه‌سازی اختصاص دارد و این اهداف بر اساس اولویت‌هایشان در هر بخش به طور جداگانه وزن‌دهی شدند.

مطابق اثبات پایداری روش تطبیق ارائه‌شده، سیستم نرم‌افزاری خود تطبیق می‌تواند نیازمندی‌های غیرعملکردی را بر اساس اولویت‌های آنها در نقطه تعادل برآورده کند.

نتایج ارزیابی روش تطبیق پیشنهادی نشان می‌دهد که سیستم نرم‌افزاری سرویس‌گرا خودتطبیق در موردمطالعه اول می‌تواند علی‌رغم عدم‌قطعیت در نیازمندی‌ها (تغییر مقدار نقطه‌تنظیم قابلیت اطمینان)، عدم‌قطعیت در پارامترهای سیستم (کاهش مقدار قابلیت اطمینان سرویس دوم)، عدم‌قطعیت در محیط اجرا (عدم دسترسی به سرویس اول)، به اهداف تعیین‌شده دست یابد و از تبدیل نوع اهداف در زمان اجرا (تبدیل نوع هدف کارایی از نقطه‌تنظیم به بهینه‌سازی و تبدیل نوع هدف هزینه از بهینه‌سازی به نقطه‌تنظیم) نیز پشتیبانی نماید. سیستم زیردریایی در طول عملیات خود با عدم‌قطعیت در محیط (خرابی حس‌گر اول) و عدم‌قطعیت در نیازمندی‌ها (کاهش نقطه‌تنظیم مصرف انرژی و حذف نیازمندی دقت) و تبدیل نوع هدف (تبدیل نوع هدف سرعت‌پیمایش از نقطه‌تنظیم به بهینه‌سازی) مواجه شد و روش تطبیق پیشنهادی با تطبیق سیستم توانست نیازمندی‌های سیستم زیردریایی خودتطبیق را تأمین نماید.

همچنین، روش تطبیق داده‌رانه با یک رویکرد تطبیق مبتنی بر مدل و نیز با روش تطبیق داده‌رانه‌ای بدون تبدیل نوع هدف، مقایسه شد. نتایج مقایسه، کارایی روش تطبیق در تأمین نیازمندی‌ها نسبت به سایر روش‌ها را نشان داد.

روش تطبیق داده‌رانه پیشنهاد شده را می‌توان در تحقیقات آتی به شرح زیر، توسعه داد.

- پشتیبانی از اهداف آستانه‌ای: روش تطبیق ارائه‌شده فقط از اهداف نقطه‌تنظیم و بهینه‌سازی پشتیبانی می‌کند؛ اما در شرایطی سیستم نرم‌افزاری خود تطبیق می‌بایست مقدار یک نیازمندی را کمتر یا بیشتر از یک آستانه معین حفظ نماید [۱]. مثلاً، در دامنه سیستم‌های نرم‌افزاری سرویس‌گرا، حفظ مقدار کارایی سیستم نرم‌افزاری کمتر از ۵ میلی‌ثانیه می‌تواند به‌عنوان یک هدف آستانه‌ای برای سیستم نرم‌افزاری خودتطبیق تعریف شود.
- کاهش اهداف: روش تطبیق پیشنهادی برای تعیین عملیات تطبیق در هر چرخه تطبیق، یک مسئله بهینه‌سازی را حل می‌کند. افزایش تعداد اهداف سیستم نرم‌افزاری خود تطبیق ممکن است پیچیدگی مسئله بهینه‌سازی را افزایش دهد [۳۷] و درنتیجه زمان موردنیاز برای حل مسئله بیشتر از زمان تعیین‌شده برای چرخه تطبیق باشد. رویکردهای کاهش هدف می‌توانند برای به‌دست‌آوردن اهداف اساسی مورداستفاده قرار گیرند.
- طراحی سازوکارهایی برای مواجه با سایر انواع عدم‌قطعیت در سیستم نرم‌افزاری: سیستم نرم‌افزاری خودتطبیق مبتنی بر روش تطبیق پیشنهادی، می‌تواند با تعدادی از عدم‌قطعیت‌ها در زمان اجرا مقابله نماید. روش تطبیق پیشنهادی را می‌توان با طراحی سازوکارهایی برای مواجه با سایر انواع عدم‌قطعیت در زمان اجرا توسعه داد.
- الگوریتم‌های تکاملی: روش تطبیق در این مقاله، پیکربندی بهینه را با روش مجموع وزن دار محاسبه

می‌کند که وزن‌های اهداف در این روش می‌بایست در زمان طراحی سیستم مشخص شوند. با توجه به آنکه الگوریتم‌های تکاملی نیاز به وزن‌دهی اهداف ندارند، تحقیقات آتی می‌توانند برای محاسبه پیکربندی بهینه از آنها استفاده نمایند.

مراجع

[1] S. Shevtsov, D. Weyns and M. Maggio, "SimCA*: A Control-Theoretic Approach to Handle Uncertainty in Self-Adaptive Systems with Guarantees," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 13, pp. 1-34, 2019.

[2] M. U. Iftikhar and D. Weyns, "Activforms: Active formal models for self-adaptation," in *Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, 2014, pp. 125-134.

[3] R. Calinescu, L. Grunske, M. Kwiatkowska, R. Mirandola and G. Tamburrelli, "Dynamic QoS management and optimization in service-based systems," *IEEE Transactions on Software Engineering*, vol. 37, pp. 387-409, 2010.

[4] D. Garlan, B. Schmerl and S.-W. Cheng, "Software Architecture-Based Self-Adaptation," *Autonomic Computing and Networking*, pp. 31-55, 2009.

[5] D. Weyns, "Software engineering of self-adaptive systems: an organised tour and future challenges," *Chapter in Handbook of Software Engineering*, 2017.

[6] S. Mahdavi-Hezavehi, D. Weyns, P. Avgeriou, R. Calinescu, R. Mirandola and D. Perez-Palacin, "Uncertainty in Self-Adaptive Systems: A Research Community Perspective," *arXiv*, 2021.

[7] S. Shevtsov and D. Weyns, "Keep it SIMPLEX: Satisfying multiple goals with guarantees in control-based self-adaptive systems," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2016, pp. 229-241.

[8] D. Weyns, N. Bencomo, R. Calinescu, J. Camara, C. Ghezzi, V. Grassi, L. Grunske, P. Inverardi, J.-M. Jezequel, S. Malek and others, "Perpetual assurances for self-adaptive systems," in *Software Engineering for Self-Adaptive Systems III. Assurances*, Springer, 2017, pp. 31-63.

[9] A. Filieri, M. Maggio, K. Angelopoulos, N. D'ippolito, I. Gerostathopoulos, A. B. Hempel, H. Hoffmann, P. Jamshidi, E. Kalyvianaki, C. Klein, F. Krikava, S. Misailovic, A. V. Papadopoulos, S. Ray, A. M. Sharifloo, S. Shevtsov, M. Ujma and T. Vogel, "Control Strategies for Self-Adaptive Software Systems," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 11, no. 4, pp. 1-31, 2017.

[10] J. L. Hellerstein, "Self-managing systems: a control theory foundation," in *29th Annual IEEE International Conference on Local Computer Networks*, IEEE, 2004, p. 708.

[11] A. Filieri, H. Hoffmann and M. Maggio, "Automated design of self-adaptive software with control-theoretical formal guarantees," in *Proceedings of the 36th International Conference on Software Engineering*, 2014, pp. 299-310.

[12] M. Litoiu, M. a. T. G. Shaw, N. M. Villegas, H. A. Muller, H. Giese, R. Rouvoy and E. Rutten, "What can control theory teach us about assurances in self-adaptive software systems?," in *{Software Engineering for Self-Adaptive Systems III. Assurances*, Springer, 2017, pp. 90-134.

[13] A. Filieri, C. Ghezzi, A. Leva and M. Maggio, "Self-adaptive software meets control theory: A preliminary approach supporting reliability requirements," in *2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*, IEEE, 2011, pp. 283-292.

[14] A. Filieri, C. Ghezzi, A. Leva and M. Maggio, "Reliability-driven dynamic binding via feedback control," in *2012 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, IEEE, 2012, pp. 43-52.

[15] S. Shevtsov, D. Weyns and M. Maggio, "Handling new and changing requirements with guarantees in self-adaptive systems using SimCA," in *2017 IEEE/ACM 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, IEEE, 2017, pp. 12-23.

[16] S. Shevtsov, D. Weyns and M. Maggio, "Self-Adaptation of Software Using Automatically Generated Control-Theoretical Solutions," in *Engineering Adaptive Software Systems: Communications of NII Shonan Meetings*, vol. 1, Singapore, Springer Singapore, 2019, pp. 35-55.

[17] A. Filieri, H. Hoffmann and M. Maggio, "Automated multi-objective control for self-adaptive software design," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, 2015, pp. 13-24.

[18] M. Maggio, A. V. Papadopoulos, A. Filieri and H. Hoffmann, "Automated control of multiple software goals using multiple actuators," in *Joint Meeting on Foundations of Software Engineering,ESEC/FSE'17*, Paderborn, Germany, 2017.

[19] A. V. Papadopoulos, "Designing Self-Adaptive Software Systems with Control Theory: An Overview," in *2022 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C)*, 2022.

[20] D. Weyns, "Control-based Software Adaptation," in *An Introduction to Self-adaptive Systems: A Contemporary Software Engineering Perspective*, Wiley, 2021, pp. 171-199.

[21] S. Shevtsov, D. Weyns and M. Maggio, "SimCA* A Control-theoretic Approach to Handle Uncertainty in Self-adaptive Systems with Guarantees," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 13, pp. 1-34, 2019.

[22] M. Sabuhi, N. Mahmoudi and H. Khazaei, "Optimizing the performance of containerized cloud software systems using adaptive PID controllers," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 15, no. 3, pp. 1556-4665, 2021.

[23] R. D. Caldas, A. Rodrigues, E. B. Gil, G. N. Rodrigues, T. Vogel and P. Pelliccione, "A hybrid approach combining control theory and AI for engineering self-adaptive systems," in *Proceedings of the IEEE/ACM 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, 2020.

[24] D. Weyns, B. Schmerl, M. Kishida, A. Leva, M. Litoiu, N. Ozay, C. Paterson, K. Tei and m. nn, "Towards Better Adaptive Systems by Combining MAPE, Control Theory, and Machine Learning," in *IEEE*, 2021.

[25] Z. a. J. S. Hou, "Data-driven model-free adaptive control for a class of MIMO nonlinear discrete-time systems," *IEEE Transactions on Neural Networks*, vol. 22, no. 12, pp. 2173--2188, 2011.

[26] E. Di Nitto, C. Ghezzi, , A. Metzger, M. Papazoglou and K. Pohl, "A journey to highly dynamic, self-adaptive service-based applications," *Automated Software Engineering*, pp. 313-341, 2008.

[27] S. Gerasimou, R. Calinescu, S. Shevtsov and D. Weyns, "Undersea: an exemplar for engineering self-adaptive unmanned underwater vehicles," in *IEEE/ACM 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, IEEE, 2017, pp. 83-89.

[28] Z. Hou and S. Jin, Model free adaptive control: theory and applications, CRC press, 2013.

[29] Z.-S. Hou and Z. Wang, "From model-based control to data-driven control: Survey, classification and perspective," *Information Sciences*, vol. 235, pp. 3-35, 2013.

[30] K. J. Astrom and B. Wittenmark, Adaptive Control Second Edition, New York: Addison-Wesley, 2008.

[31] Z. Hou, R. Chi and H. Gao, "An overview of dynamic-linearization-based data-driven control and applications," *IEEE Transactions on Industrial Electronics*, vol. 64, pp. 4076-4090, 2016.

[32] D. Weyns, "Basic Principles of Self-Adaptation and Conceptual Model," in *An Introduction to Self-adaptive Systems: A*

Contemporary Software Engineering Perspective, WILEY, 2021, pp. 1-14.

[33] S. Petra and C. Popa, "Single projection Kaczmarz extended algorithms," *Numerical Algorithms*, vol. 73, no. 3, pp. 791-806, 2016.

[34] S. Mahdavi-Hezavehi, P. Avgeriou and D. Weyns, "A Classification Framework of Uncertainty in Architecture-Based Self-Adaptive Systems With Multiple Quality Requirements," in *Managing Trade-Offs in Adaptable Software Architectures*, Morgan Kaufmann, 2017, pp. 45-77.

[35] S. Burer and A. N. Letchford, "Non-convex mixed-integer nonlinear programming: A survey," *Surveys in Operations Research and Management Science*, vol. 17, pp. 97-106, 2012.

[36] M. ApS, MOSEK modeling cookbook, 2020.

[37] D. K. Saxena, J. A. Duro, A. Tiwari, K. Deb and Q. Zhang, "Objective reduction in many-objective optimization: Linear and nonlinear algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 17, pp. 77-99, 2012.

پاورقی‌ها:

[1] Managing system
[2] Managed system
[3] Strict requirements
[4] Model Predictive Control(MPC)
[5] Change-point detection mechanism
[6] Monitor-Analyze-Plan-Execute(MAPE)
[7] Model-Free Adaptive Control(MFAC)
[8] Dynamic Linearization Data Model
[9] Multi-Objective Dynamic Binding mechanism
[10] Unmanned Underwater Vehicle (UUV) system
[11] Push-Button Methodology (PBM)
[12] Automated Multi-objective Control for Self-Adaptive Software
[13] Simplex Control Adaptation
[14] Setpoint
[15] Automated Control of Multiple Software Goals using Multiple Actuators (AMOCS-MA)
[16] Overshoot
[17] Steady-state error
[18] Multiple-Input Multiple-Output (MIMO)
[19] Pseudo Partial Derivative(PPD)
[20] Projection Algorithm
[21] Internal Principle of Self-Adaptive System
[22] Dynamic Linearization Technique
[23] Lipschitz
[24] Compact Form Dynamic Linearization (CFDL) technique
[25] Kaczmarz’s algorithm
[26] Stability
[27] Equilibrium point
[28] Unmanned Underwater Vehicle(UUV)
[29] Mixed Integer Programming(MIP)
[30] Monitors