New Researches in the Smart City
Vol 1 (Issue 4), 26-44
Nrsc.apadana.ac.ir

Apadana Institute of Higher Education

Improving Memory for Solving Dynamic Distributed Factory Coordination Problem

Roohallah Daneshpayeh*
Assistant Professor, Department of Mathematics, Payame Noor University, p.o.box. 19395-4697, Tehran, Iran.

Majid Mohammadpour
Ph.D. Student in Computer Engineering, Yazd University, Yazd, Iran.

Received: 24/February/2023 Accepted: 15/August/2023 ISSN: 2980-8936

Abstract

Many real-world problems involve the coordination of multiple agents in dynamic environments. Machines in a factory may need to coordinate the scheduling and execution of jobs to ensure smooth operation as customer demands shift. In the domain of factory operations, adaptive, self-organizing agent-based approaches have been shown to provide very robust solutions. However, these adaptive approaches may require non-trivial amounts of time to respond to large environmental shifts. Techniques exist that have been shown to help many different approaches perform better when problems are dynamic. One common technique is the use of information from the past to improve current performance. In many dynamic problems, the current state of the environment is often similar to previously seen states. Using information from the past may help to make the system more adaptive to large changes in the environment and to perform better over time. One way to maintain and exploit information from the past is the use of memory, where solutions are stored periodically and can be retrieved and refined when the environment changes. This paper introduces several density-estimation memory systems that are inspired by estimation of distribution for solving one of the hard dynamic problems (Factory coordination). In this proposed method, instead of storing only single points in memory, we propose to store clusters of points in each memory entry and to create a model of the points in each cluster. In this proposed method we will be able to store many more points, the computation overhead required for the memory will remain low. The experimental results show the efficiency of the proposed algorithm in comparison with other methods.

Keywords: density estimation memory**,** $R-wasps$, factory coordination, dynamic problem

* Corresponding Author: Rdaneshpayeh@pnu.ac.ir

فصلنامه پژوهش‌های نوین در شهر هوشمند
دوره ۱، شماره ٤، تابستان ١٤٠٢، ٢٦-٤٤
Nrsc.apadana.ac.ir

بهبود حافظه برای حل مسئله هماهنگ‌سازی کارخانه‌های پویای توزیع‌شده

روح اله دانش پایه * استادیار گروه ریاضی، دانشگاه پیام نور، تهران، ایران.

مجید محمدپور دانشجوی دکتری مهندسی کامپیوتر، دانشگاه یزد، ایران.

تاریخ دریافت: ۱۴۰۱/۱۲/۰۵ تاریخ پذیرش: ۱۴۰۲/۰۵/۲۴ ISSN: 2980-8936

چکیده

در دنیای واقعی، مشکلات بسیاری روند هماهنگ‌سازی عامل‌های چندگانه را در محیط‌های پویا تهدید می‌کند. دستگاه‌ها و ماشین‌های موجود در کارخانه اصولاً به یک برنامه‌ریزی مشخص احتیاج دارند تا بدین وسیله بتوانند عملیات را با اطمینان بیشتر و مطابق با تقاضای مشتری انجام دهند. در حیطهٔ فعالیت‌ها و عملیات کارخانه، رویکردهای مبتنی بر عامل‌های خودسازمانده و انطباقی، اصولاً قادر هستند راه‌حل‌های قوی و معتبری فراهم کنند. با این حال، رویکردهای انطباقی به میزان مشخصی زمان برای پاسخ‌دهی به تغییرات محیطی احتیاج دارند. به هنگام رویارویی با مسائل پویا، تکنیک‌هایی که در دست دارید می‌تواند به بهتر انجام‌شدن روش‌های مختلف دیگر کمک کند. یکی از این روش‌های معمول، استفاده از اطلاعات گذشته به‌منظور بهبود و ارتقاء عملکردهای فعلی می‌باشد. در بسیاری از مسائل پویا، وضعیت کنونی محیط شباهت قابل‌توجهی با حالت‌هایی که در گذشته مشاهده شده است، دارد. استفاده از اطلاعات پیشین با گذشت زمان ممکن است به منطبق‌سازی هرچه بیشتر سیستم با تغییرات وسیع محیطی و اجرای بهتر کمک کند. یکی از راه‌هایی که به‌موجب آن می‌توان اطلاعات گذشته را حفظ و نگهداری کرد استفاده از حافظه‌ای است که راه‌حل‌ها را به صورت دوره‌ای ذخیره کرده، بازیابی نموده و به هنگام اعمال تغییرات محیطی آن‌ها را تصحیح نماید. مقالهٔ حاضر قصد دارد چندین سیستم حافظهٔ ارزیابی چگالی را که از تخمین الگوریتم‌های توزیع، الهام گرفته‌اند ارائه نموده و از این سیستم برای حل یکی از مشکل‌ترین مسائل پویا (مسئلهٔ هماهنگ‌سازی توزیع‌شدهٔ پویا) استفاده نماید. در سیستم‌های حافظهٔ ارائه‌شده، به‌جای ذخیره‌کردن تنها نقاط تکی در حافظه، خوشه‌هایی از نقاط در هر مدخل حافظه ذخیره می‌شوند و مدلی از هر نقاط در هر خوشه ایجاد می‌شود. این نوع حافظه، قادر به ذخیره‌کردن نقاط بیشتری بوده و سربار محاسباتی برای این نوع حافظه کم است. نتایج آزمایش‌ها، حاکی از برتری روش پیشنهادی نسبت به سایر روش‌ها است.

کلیدواژه‌ها: حافظه تخمین تراکم، $R-wasps$، هماهنگ‌سازی کارخانه، مسئلهٔ پویا

* Corresponding Author: Rdaneshpayeh@pnu.ac.ir

۱- مقدمه

در مسائل جهان واقعی عامل‌های چندگانه با مشکلاتی مواجه هستند که روند هماهنگی عامل‌ها را در یک محیط پویا مورد تهدید قرار می‌دهد. ابزار موجود در کارخانه به یک برنامه‌ریزی مشخص نیاز دارند تا بتوانند عملیات را با قابلیت اعتماد بیشتری و سازگار با تقاضای مشتریان انجام دهند. در یک کارخانه، گروهی از روبات‌ها ممکن است نیاز به هماهنگ‌سازی و تخصیص صحیح وظایف داشته باشند. بدین ترتیب، آن‌ها می‌توانند در محیط‌های جدید و در حال تغییر نیز کار کنند. عامل‌های مبتنی بر وب به هماهنگ‌سازی خدماتی از قبیل جمع‌آوری اطلاعات بسته به نوع ورودی یا تغییر تقاضای مربوطه مستلزم می‌باشند. عموماً، یک رویکرد مطابق با ماهیت مسئله انتخاب می‌شود که قادر است مسائل پویا را به طرق مختلفی حل کند. تغییرات به‌وجودآمده بایستی سرتاسر با رویهٔ حل مسائل در هر برههٔ زمانی که اتفاق می‌افتد، هم‌خوانی داشته باشد. استفاده از شیوهٔ بهینه‌سازی متمرکز به مرور زمان راه‌حل خوبی را کشف کرده، مدل توزیع‌شده ثابتی را با تغییرات وفق داده و مدل جدیدی را جهت هماهنگ‌سازی رویکردها فرا می‌گیرد. در حیطهٔ فعالیت‌ها و عملیات کارخانه، رهکارهای مبتنی بر عامل‌های خودسازمانده و انطباقی، قادر هستند راه‌حل‌های قوی و معتبری فراهم کنند. یک مرکز تولیدی اساساً محیطی پویا و در عین حال پیچیده است که تغییرات در میزان تقاضای محصول و در دسترس‌بودن منابع آن به صورت کاملاً ثابت و معین رخ می‌دهد. این نوع تغییرات اغلب اوقات با تلاش برای ایجاد برنامه‌های پیشرفته تناقض دارد. به‌واسطهٔ استفاده از روش‌های انطباقی مذکور، یک زمان‌بند می‌تواند نسبت به رویدادهای غیرقابل‌پیش‌بینی حساس بوده و از برنامه‌ریزی‌های غیرمجاز جلوگیری کند. با این حال، رویکردهای انطباقی ذکرشده نیاز به زمان زیادی برای پاسخ به تغییرات محیطی دارند. به هنگام مواجهه با مسائل پویا، روش‌های موجود، می‌توانند به انجام‌شدن بهتر روش‌های آتی کمک کنند. یکی از این روش‌های معمول، استفاده از راه‌حل‌های گذشته به‌منظور بهبود عملکردهای فعلی می‌باشد. در بسیاری از مسائل پویا، وضعیت فعلی محیط شباهت قابل‌توجهی با حالت‌هایی که در گذشته مشاهده شده؛ دارد. استفاده از اطلاعات گذشته ممکن است با گذشت زمان به منطبق‌سازی هرچه بیشتر سیستم با تغییرات وسیع محیطی و اجرای بهتر کمک کند. یکی از راه‌هایی که به‌موجب آن می‌توان اطلاعات گذشته را حفظ و نگهداری کرد، استفاده از حافظه‌ای است که راه‌حل‌ها را به صورت دوره‌ای ذخیره کرده، بازیابی نموده و به هنگام اعمال تغییرات محیطی آن‌ها را تصحیح نماید. پیرامون مسائل پویا، آن رویکردی که به‌خوبی تهیه و ارائه شده باشد اصولاً می‌تواند رویهٔ کار حافظه را تسهیل بخشیده و مستقیماً تعداد مشخصی از راه‌حل‌های گذشته را ذخیره کند. به‌محض این‌که تعداد حالات محیطی امکان‌پذیر در یک مسئلهٔ پویا افزایش یابد، حافظه‌ای که حجم مشخص و ثابتی دارد با دشواری‌های بسیاری جهت ذخیره راه‌حل‌ها مواجه خواهد شد. در صورتی که حجم حافظه را بتوان افزایش داد، آن‌گاه میزان سربار مشخص می‌کند که از چه مقدار حافظه می‌توان به صورت مؤثر استفاده کرد. مقالهٔ حاضر قصد دارد چندین سیستم حافظهٔ ارزیابی چگالی را که از تخمین الگوریتم‌های توزیع الهام گرفته‌اند، مورد بحث و بررسی قرار دهد. شایان ذکر است که این الگوریتم‌ها اصولاً موجب بهبود سیستم‌های حافظهٔ استاندارد و در عین حال، جلوگیری از افزایش وسیع سربار می‌شوند. عملکرد چنین حافظه‌هایی را می‌توان بر مبنای مسئلهٔ هماهنگ‌سازی کارخانه و ساخت پویای توزیع‌شده، ارزیابی کرد. این مسئله اصولاً به یک توافق‌نامهٔ پویا با برنامه‌هایی تحت شرایط شبیه‌سازی‌شده احتیاج دارد. در صورتی که محصولات از انواع و اقسام گوناگونی برخوردار باشند، آن‌گاه می‌بایست هر دستگاه را به یک روند پردازش مشخص اختصاص داد. زمانی که دستگاه از یک نوع فرآوری محصول به نوع دیگر تغییر وضعیت می‌دهد، زمان برپایی، اندکی متضرر می‌شود. مقالهٔ حاضر، مسئلهٔ هماهنگ‌سازی توزیع‌شده و بعضی از رویکردهای مبتنی بر عامل را توضیح خواهد داد. هم‌چنین، راجع به گوناگونی جدید این مسائل، رویکرد پایهٔ مبتنی بر وب و نقاط ضعف آن نیز صحبت خواهد کرد. سپس، نحوهٔ استفاده از حافظه را جهت بهبود

عملکرد مسئلهٔ هماهنگ‌سازی توزیع‌شدهٔ پویا معرفی کرده و تعدادی سیستم حافظهٔ ارزیابی چگالی را ارائه خواهد نمود. درنهایت نیز عملکرد سیستم پیشنهادی را با سیستم‌های حافظهٔ معرفی‌شده مقایسه خواهد کرد. نوآوری اساسی این مقاله، ارائهٔ چندین نوع حافظه برای حل مسئلهٔ هماهنگ‌سازی کارخانه می‌باشد. روش حافظهٔ پیشنهادی در این مقاله، نقاط ضعف حافظهٔ استاندارد را برطرف نموده است.

بخش‌بندی این مقاله بدین صورت آرایش یافته است:

در بخش اول مقدمه و کلیات موضوع بیان شده است. بخش دوم پیش‌زمینهٔ تحقیق و مروری بر کارهای گذشته را بیان می‌کند. بخش سوم روش پیشنهادی را بیان می‌کند. بخش چهارم آزمایش‌ها و نتایج تجربی را توصیف می‌کند. در بخش پنجم نتیجه‌گیری و کارهای آتی تشریح شده است.

۲- پیش‌زمینهٔ تحقیق و مروری بر کارهای گذشته

فرآیندهای ساخت و تولید، اغلب مثال‌های جالب‌توجهی پیرامون مسائل پویا ارائه می‌کنند. به عنوان مثال، مسئلهٔ هماهنگ‌سازی مرکز تولید مرسوم به مسئلهٔ تخصیص وظائف پویا اساساً شامل کارهای مخصوصی برای دستگاه‌ها می‌شود که درنهایت عملکرد پردازش را به دنبال خواهد داشت. با گذشت زمان، کارها واگذار شده و زمان‌بند اطلاعات اندکی راجع به کار به دست خواهد آورد. در صورت عدم اطلاعات کافی، به‌جای استفاده از یک زمان‌بند متمرکز جهت محاسبهٔ برنامه‌های مطلوب، سیستم‌های زمان‌بندی انطباقی، به ازای انواع و اقسام مسائل این‌چنینی ارائه می‌گردد. دانشمندی به نام مورلی نمونه‌ای از مسئلهٔ هماهنگ‌سازی کارخانه را از کارخانهٔ الکتریکی عمومی ارائه داد (Morrison & De Jong, 1999). در این مسئله، هر کامیون را متناسب با رنگ اتاقک آن مشخص کرده و بدین وسیله، عامل‌ها را تخصیص داد. کامیون‌هایی که به خط مونتاژ می‌رسیدند می‌بایست رنگ شده و سپس برای رنگ اتاقک در صف گذاشته می‌شدند. رنگ هر کامیون از روی احتمال بر اساس توزیع رنگ‌ها مشخص می‌گردد. هر اتاقک دارای صفی از کامیون می‌باشد که منتظر است رنگ شوند. تمامی اتاقک‌ها می‌توانند با هر رنگی که در همان نقطه در دسترس می‌باشد رنگ شوند؛ اما زمانی که یک اتاقک بین رنگ‌ها تعویض می‌گردد، می‌بایست با جریان سریع رنگ قبلی شسته شود که این امر در زمان برپایی، یک وقفه ایجاد کرده و هزینهٔ رنگ را بالا می‌برد. اتاقک‌هایی که تنها با یک رنگ مشخص نقاشی می‌شوند عملاً برپایی‌های کمتری را متضرر می‌گردند. مورلی ثابت کرد که یک رویکرد مبتنی بر بازار می‌بایست قبل از این که اتاقک‌ها برای سوارشدن روی کامیون به خط مونتاژ برسند، یک زمان‌بند متمرکز داشته باشد تا به‌واسطهٔ فروشگاه مورد استفاده قرار گرفته شود (Morrison & De Jong, 1999). این سیستم تقریباً یک میلیون دلار طی ۹ ماه اول استفاده صرفه‌جویی می‌کند. در این رویکرد، بر اساس طول فعلی صف اتاقک و همچنین تشخیص این که آیا وقفهٔ زمانی جهت پردازش کامیون لازم است یا خیر، اتاقک رنگ‌آمیزی‌شده بر روی یکی از کامیون‌ها سوار می‌شود. کمپوس[1] و همکاران (۲۰۰۰) و سیسیرلو[2] و همکاران (۲۰۰۴) هرکدام به صورت جداگانه رویکردهای مبتنی بر عامل را به ازای مسئلهٔ هماهنگ‌سازی کارخانه که از تخصیص وظیفهٔ خودسرانهٔ جنبنده‌های اجتماعی نظیر مورچه‌ها و زنبورها الهام گرفته شده است، مورد بحث و بررسی قرار دادند (Campos et al., 2000; Cicirello & Smith, 2004). همانند شیوهٔ مورلی، اتاقک‌ها همچنان در مقابل دیگری صف کشیده‌اند، ولیکن به‌جای یک سیاست مشخص، عامل‌هایی که هر اتاقک را نشان می‌دهند می‌بایست از روش یادگیری تقویتی به منظور توسعهٔ سیاست‌ها استفاده کنند. عامل‌ها اغلب از مفهوم آستانهٔ پاسخ برای تشخیص پیشنهاد برای هر کامیون بهره می‌گیرند. به‌واسطهٔ چنین الهامی، تفاوت‌های قابل‌توجهی در بین این رویکردها به وجود می‌آید. سیسیرلو و همکاران

1. Campos
2. Cicirello

(۲۰۰۴) سیستم‌های خود را با سیستم پیرامون شش مسئله مقایسه نمودند (Campos et al., 2000): مسئلهٔ اصلی، نسخه‌ای با زمان‌های برپایی مشخص‌تر، نسخه‌هایی با دو احتمال مجزا به ازای از کارافتادگی دستگاه، نسخه‌ای با توزیع رنگ کامیون یدکی و نسخه‌ای که در آن توزیع رنگ کامیون در وسط کار تغییر پیدا می‌کند. $R-Wasp$ قبل از سایر رویکردها علی‌الخصوص به‌منظور به حداقل رساندن تعداد تنظیمات (برپایی) موردنیاز به کار گرفته شد. محمدپور و پروین (۱۳۹۵) نیز انواع مسائل مربوط به فروشگاه رنگ و نقاشی را به‌منظور تجزیه‌وتحلیل هرچه بهتر عملکرد الگوریتم نشان دادند. در این رابطه، انواع کارهای N، به‌واسطهٔ دستگاه‌های چندمنظورهٔ M که به صورت موازی فعالیت می‌کنند، پردازش گردید. کارها از روی احتمال و بر اساس انواع و اقسام آن مشخص می‌شوند. هر کار دارای $15+$ $N(0,1)$ واحد زمانی می‌باشد. به هر دستگاه اجازه داده می‌شود تا یک صف نامتناهی داشته باشد. تنظیمات یک ماشین طی واحدهای زمانی ۳۰ ممکن است تغییر پیدا کند. سیسیرلو و همکاران (۲۰۰۴) مسائل را در حضور ۲ نوع کار در ۲ یا ۴ دستگاه مورد بحث و بررسی قرار دادند (Cicirello & Smith, 2004). آن‌ها وضعیت را هم به‌وسیلهٔ یک نوع توزیع کار و هم بین دو نوع توزیع کار آزمایش کردند. یکی از یافته‌ها ثابت کرد که این رویکرد ممکن است جهت انطباق با تغییرات موجود در حیطهٔ توزیع انواع و اقسام کار، اندکی آهسته عمل کند.

برای حل مسائل پویا راهکارهای متعددی اخیراً توسط محققین ارائه شده است. در ادامه برخی از این راهکارها تشریح شده است.

محمدپور و پروین (۲۰۱۶) یک حافظهٔ صریح را پیشنهاد دادند که از این حافظه برای حداکثرسازی تنوع در جمعیت استفاده شده است (Mohammadpour & Parvin, 2016). در این حافظه از یک راهکار مناسب برای به‌روزرسانی حافظه استفاده شده است.

اوسویدان و بایکاسوگلو[1] (۲۰۱۵) از راهکار چندجمعیتی همراه با روش تکاملی کرم شب‌تاب برای حل مسائل پویا استفاده کرده‌اند (Ozsoydan & Baykasoglu, 2015). در این تحقیق نشان داده شده است که روش‌های چندجمعیتی به بهبود کارآیی الگوریتم‌ها در حل مسائل پویا کمک می‌کند.

براوو[2] و همکاران (۲۰۱۶) یک روش مبتنی بر حافظه را برای حل مسائل بهینه‌سازی پویا ارائه نمودند (Bravo et al., 2016). در این مقاله از یک حافظه با ترکیب سه الگوریتم ژنتیک، ازدحام ذرات و جستجوی محلی تپه‌نوردی استفاده شده است. در این روش از حافظهٔ ارائه‌شده برای حفظ سطح مناسبی از تنوع استفاده شده است. این حافظه در این مقاله تحت عنوان حافظهٔ سراسری[3] نام‌گذاری شده است.

صادقی و همکارانش (۲۰۱۵) یک الگوریتم مبتنی بر ازدحام ذرات برای حل مسائل بهینه‌سازی پویا ارائه نموده‌اند (Sadeghi et al., 2015). در این روش برای ردیابی سریع بهینهٔ تغییریافته از یک حافظهٔ صریح استفاده شده است. در این الگوریتم بهترین راه‌حل‌های گذشته که خیلی قدیمی نباشند در حافظه ذخیره شده است.

یزدانی و میبدی روشی برای حل مسائل بهینه‌سازی پویا ارائه نموده‌اند (Nasiri & Meybodi, 2012). این روش بر مبنای مفهوم «اجزا[4]» طراحی شده است که مرکز یک جزء، دانهٔ جزء نامیده می‌شود. دانهٔ جزء ذره‌ای است که همواره بهترین برازش را در آن جزء دارد. همهٔ ذراتی که در شعاع از پیش تعریف‌شده دانهٔ جزء واقع شوند در همان جزء قرار می‌گیرند.

1. Ozsoydan & Baykasoglu
2. Bravo
3. Global Memory
4. species

ریچر و یانگ[1] (۲۰۰۸ و ۲۰۰۹) حافظه‌ای ارائه دادند که به جای ذخیرهٔ مستقیم راه‌حل‌ها، چکیده‌ای از راه‌حل‌ها را ذخیره می‌نمود و با نگه‌داشتن یک ماتریس، فضای جستجو را درون سلول‌ها تقسیم می‌کرد (Richter & Yang, 2008a, 2008b, 2009). هنگامی که یک راه‌حل به حافظه اضافه می‌شد، شمارندهٔ مربوط به سلول ماتریس افزایش می‌یافت. این امر به ماتریس اجازه می‌داد تا به عنوان یک مدل چکیده از راه‌حل‌های خوب عمل کند. بعد از یک تغییر، راه‌حل‌ها به‌وسیلهٔ نمونه‌برداری از ماتریس و مقداردهی مجدد به بخشی از جمعیت، بازیابی می‌شدند.

در یک کار تخمین توزیع جمعیت، یانگ[2] (۲۰۰۶ و ۲۰۰۷) حافظهٔ انجمنی را معرفی کرد که یک راه‌حل و یک تخمین توزیع را با هم در یک مدخل حافظه ذخیره می‌کند (Yang, 2006, 2007a, 2007b). بعد از یک تغییر، راه‌حل‌های تمام مدخل‌های حافظه محاسبه می‌شوند. توزیعی از مدخل حافظه با بهترین راه‌حل برای مقداردهی مجدد به بخشی از جمعیت انتخاب می‌گردد. حافظهٔ انجمنی با حافظهٔ مستقیم و یک حالت ترکیبی از حافظهٔ مستقیم و انجمنی مقایسه شده است. استفاده از حافظهٔ انجمنی به طور قابل‌توجهی بهتر از حافظهٔ مستقیم تنها بوده و حالت ترکیبی بهترین عملکرد را در بین آن‌ها داشته است.

برنک و متفلد[3] (۲۰۱۲) یک مدل عمومی‌تر از حافظه ارائه کردند که نیازی به ذخیرهٔ اطلاعات محیط نداشت (Branke, 2012; Branke & Mattfeld, 2002). حافظه تلاش می‌کرد تا به صورت تناوبی بهترین اعضای جمعیت را ذخیره کند. حافظه یک اندازهٔ محدود و مشخص داشت و در زمان پرشدن حافظه از یک راه‌حل جایگزینی استفاده می‌شد که تصمیم می‌گرفت آیا نیازی به جایگزینی بهترین عضو جمعیت با یکی از مدخل‌های حافظه هست یا خیر.

محققین سیستم‌های حافظهٔ مختلفی را ارائه نموده‌اند و هر کدام از سیستم‌های ارائه‌شده به نوعی برای حل مسائل پویا استفاده شده‌اند (Bendtsen & Krink, 2002; Chen et al., 2015; Chryssolouris & Subramaniam, 2001; Eggermont & Lenaerts, 2002; Eggermont et al., 2001; Ramsey & Grefenstette, 1993).

۱-۲- مسئلهٔ هماهنگی کارخانه‌های توزیع‌شدهٔ پویا

در بخش حاضر، نسخهٔ توسعه‌داده‌شدهٔ مسئلهٔ هماهنگ‌سازی کارخانهٔ توزیع‌شدهٔ پویا، شبیه به گفته‌های سیسیرلو و همکاران مورد بحث و بررسی قرار خواهد گرفت. در نسخه‌های پیشین از این مسئله، الگوریتم‌ها طی بازهٔ زمانی کوتاه، به سبب بروز چندین تغییر در روند توزیع کارها ارزیابی شدند. در حیطهٔ مسئلهٔ هماهنگ‌سازی کارخانهٔ پویایی که هم‌اکنون توصیف شد، عملکرد طی بازهٔ زمانی بلندمدت و با در نظر گرفتن توزیع تغییرات انواع و اقسام کار، ارزیابی می‌شود. در این مسئله، کارخانه‌ها N فرآورده تولید می‌کنند (N نوع کار) که به‌وسیلهٔ M دستگاه چندمنظوره به صورت موازی پردازش می‌شوند. طول صف کار یک دستگاه، اصولاً نامحدود می‌باشد. زمان تنظیمات (برپایی) هر ماشین، بسته به نوع کاری که قرار است انجام دهد برابر با ۳۰ واحد زمانی است. زمان پردازش هر کار $15 + N(0,1)$ واحد است. مدت‌زمان‌های بیشتر از ۱۵ واحد زمانی به صورت عدد صحیح گرد می‌شود، در حالی که مقادیر کمتر از آن زیر ۱۵ گرد می‌شوند. کارها به کف کارخانه بسته به توزیع انواع و اقسام کار واگذار می‌گردد؛ این توزیع به مرور زمان تغییر پیدا می‌کند. به عنوان مثال، مطابق با دو نوع کار به نسبت ۶۰ به ۳۰ و فرصت ۱۰ درصدی به‌وجودآمدن یک کار جدید در هر بازهٔ زمانی، توزیع برابر $D = [0.06, 0.04]$ خواهد بود. تنها یک کار ممکن است در هر بازهٔ زمانی وجود داشته باشد. بدین ترتیب، نرخ میانگین ۰/۰۵ به ازای هر دستگاه جهت نمایش کارخانه‌ای با قابلیت بارگذاری سیستم تعریف می‌شود که $L = 1.00$ میزان معمول بار را نشان می‌دهد. مقادیر بزرگ‌تر اصولاً سیستم را درون

1. Richter & Yang
2. Yang
3. Branke & Mattfeld

وضعیت ازدیاد بار قرار می‌دهند. متوسط زمان ورود $\lambda = 0.5ME$ می‌باشد. این وضعیت طی بازهٔ زمانی ۱۵۰۰۰۰ سپری شده و به ۵۰ دورهٔ ۳۰۰۰تایی تقسیم می‌شود؛ در هنگام آغاز هر دوره، نحوهٔ توزیع نوع کار تغییر پیدا می‌کند.

۲-۲- الگوریتم یادگیری مبتنی بر عامل $R-Wasps$

ما از الگوریتم $R-Wasps$ به عنوان رویکردی پایه در مسئلهٔ فعلی استفاده می‌کنیم (Cicirello & Smith, 2004). در $R-Wasps$، هر دستگاه متناسب با عامل مسیریابی مشخص می‌شود. هر عامل دارای آستانهٔ پاسخ می‌باشد:

$$\Theta_w = \{\theta_{w,0},\ldots,\theta_{w,N-1}\} \quad (۱)$$

در این معادله $\theta_{w,j}$ آستانهٔ w نسبت به کارهای نوع j می‌باشد. کارهای تخصیص‌نشده باعث پخش محرک s_j متناسب با طول مدت‌زمانی که کار منتظر توافق می‌باشد، می‌شود. این عامل قادر است محرک s_j را با احتمال ذیل نشان بدهد:

$$P\left(bid \mid \theta_{w,j}, s_j\right) = \frac{S_j^2}{S_j^2 + \theta_{w,j}^2} \quad (۲)$$

در غیر این صورت، عامل به کار مربوطه پیشنهاد نمی‌شود. هرچه آستانهٔ مزبور به ازای کار مشخصی کمتر باشد، احتمال پیشنهاد عامل به کار مربوطه نیز بیشتر می‌شود. مقادیر آستانه در بازه $[\theta_{min}, \theta_{max}]$ متفاوت است. هر عامل مسیریابی w می‌بایست کاملاً نسبت به وضعیت دستگاه آگاه باشد، ولی نباید هیچ اطلاعاتی پیرامون سایر دستگاه‌های موجود در کارخانه داشته باشد. از دانش مذکور اغلب جهت تنظیم آستانه‌ها در هر بازهٔ زمانی استفاده می‌شود. اگر ماشین یک نوع کار j را پردازش کند، آن‌گاه:

$$\theta_{w,j} = \theta_{w,j} - \delta_1 \quad (۳)$$

اگر دستگاه، پیرامون یک نوع کار j را پردازش کند، آن‌گاه:

$$\theta_{w,j} = \theta_{w,j} + \delta_2 \quad (۴)$$

در صورتی که دستگاه برای مدت‌زمان t بی‌کار بوده و دارای صف خالی نیز باشد، آن‌گاه به ازای همه نوع کار j خواهیم داشت:

$$\theta_{w,j} = \theta_{w,j} - \delta_3^t \quad (۵)$$

مقادیر پارامترهای سیستم که به کار گرفته شده‌اند، عبارت است از:

$$\theta_{min} = 1, \theta_{max} = 1000, \delta_1 = 2, \delta_2 = 1, \delta_3 = 1.001 \quad (۶)$$

زمانی که بیش از یک عامل پیرامون شغلی پیشنهاد داده می‌شود، یک تناقض قابل‌توجه به وجود می‌آید. نیروی F_W یک عامل عبارت است از: $Fw = 1.0 + Tp + Ts$.

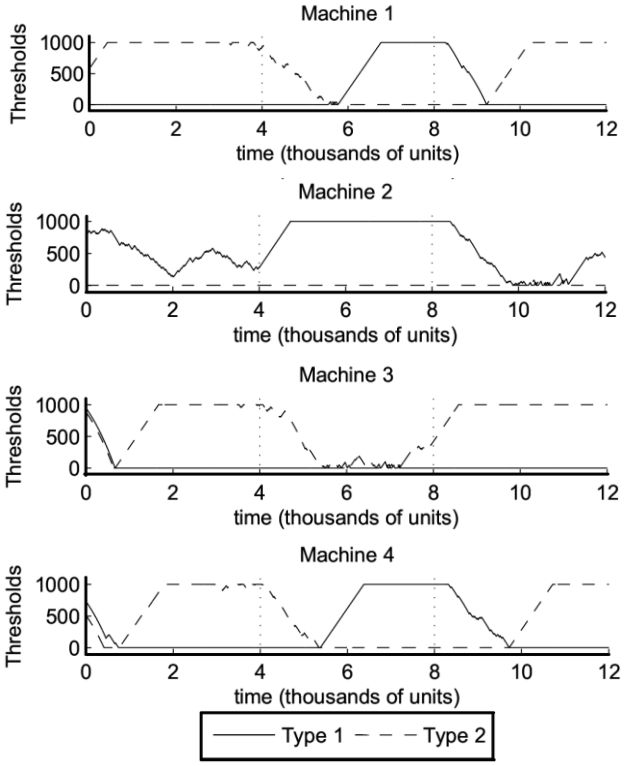
در این معادله T_P و T_S به‌ترتیب مجموع زمان‌های پردازش و زمان‌های تنظیمات را به ازای تمامی کارهای موجود در صف نشان می‌دهد. فرض کنید F_1 و F_2 به‌ترتیب نیروهای عامل ۱ و ۲ هستند. در آن صورت، عامل ۱ تناقض احتمال مربوطه را خواهد برد.

$$P(Agent\ 1 wins \mid F1, F2) = \frac{F_2^2}{F_1^2 + F_2^2} \quad (۷)$$

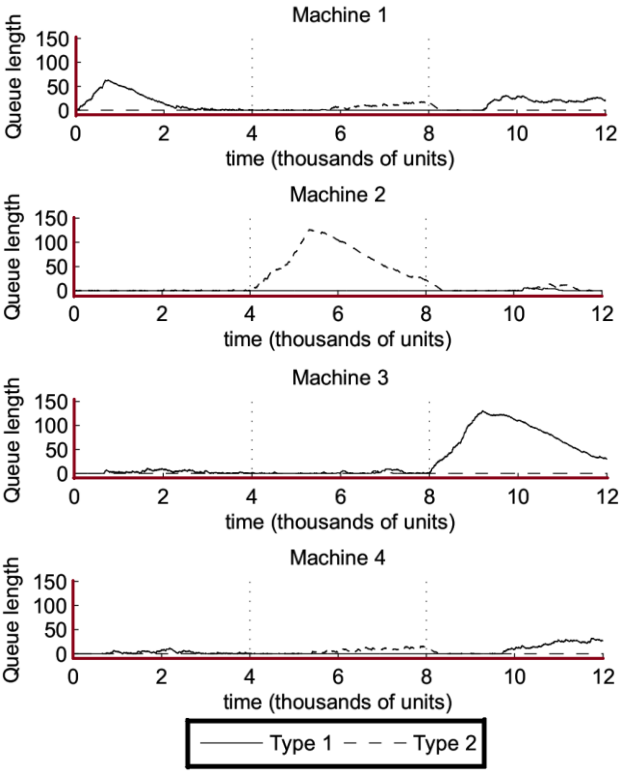
اگر بیش از دو عامل پیرامون کار معرفی شود، از یک تناقض مشخص جهت تعیین پیشنهاد برنده استفاده می‌شود (Cicirello & Smith, 2004).

۳-۲- نقاط ضعف الگوریتم $R-Wasps$

$R-Wasps$ بر مبنای مسئلهٔ هماهنگ‌سازی کارخانهٔ توزیع‌شده عمل می‌کند؛ اما از آن جایی که یادگیری آستانه‌ها زمان بسیار زیادی می‌برد، پس ممکن است، رویهٔ انطباق با تغییرات توزیع این نوع کار، اندکی آهسته صورت گرفته شود. اگر توزیع کار مربوطه هم‌چنان با گذشت مدت‌زمان به همان صورت باقی بماند، آن‌وقت سیستم می‌تواند هر گونه مشکلی که انطباق پدید آورده است را برطرف سازد. با این حال، طی بازهٔ زمانی کوتاه، این امر ممکن است تأثیر منفی بر عملکرد داشته باشد. به عنوان مثال، یک مسئله با چهار ماشین را در نظر بگیرید. دو نوع کار طی سه دورهٔ زمانی با طول ۴۰۰۰ واحد زمانی حضور دارد. در اولین دورهٔ زمانی، ۸۵ درصد کارها در نوع ۱ و ۱۵ درصد در نوع ۲ وارد می‌شوند. در ابتدا، هر ماشین قادر است آستانهٔ خود را جهت پذیرفتن مؤثر کارها تطبیق بدهد. در دومین دوره، زمانی که توزیع کارها به ۱۵ درصد نوع ۱ و ۸۵ درصد نوع ۲ تغییر پیدا می‌کند، هر ماشین می‌تواند خود را با توزیع جدید وفق دهد. در سومین دوره، توزیع به ۸۵ درصد از نوع ۱ و ۱۵ درصد از نوع ۲ بازمی‌گردد. همان‌گونه که در شکل ۱ نشان داده شده است، اعمال تغییر در توزیع احتمالاً بیشتر از نخستین انطباق در طی اولین مرحله، زمان می‌برد. این عملکرد بنا به چند دلیل رخ می‌دهد. اولاً، توزیع آشکارا بسیار آهسته‌تر از توزیع مربوطه رخ می‌دهد؛ بدین ترتیب کارهایی که طی اولین دورهٔ زمانی توزیع ارائه می‌شوند هم‌چنان در آغاز دومین دوره به صورت پردازش‌نشده باقی می‌مانند. دوماً، تغییرات توزیع ممکن است منجر به درهم‌ریختگی صف شود. این امر را به‌وضوح می‌توان در شکل ۲ مشاهده کرد؛ در این حالت صف به ازای ماشین ۳ پس از اعمال تغییر نوع دوم بسیار وسیع رشد پیدا می‌کند. این امر اغلب زمانی رخ می‌دهد که ماشین در پردازش یک نوع کار نسبت به سایرین از تخصص بیشتری برخوردار باشد. در صورتی که کار مربوطه بسیار متداول باشد، آن‌گاه ماشین پیشنهاد مربوطه را تا زمانی که سایر ماشین‌ها فرصتی برای تخلیهٔ صف دارند، خواهد برد. در آن هنگام، سایر ماشین‌ها حرفه‌ای‌تر عمل کرده و ماشینی که دچار ازهم‌پاشیدگی صف شده است روند پذیرش کارها را متوقف می‌سازد. این واقعه ممکن است منجر به بیکاری سیستم شود، ولیکن مسئلهٔ واقعی این است که دورهٔ زمانی بزرگ و بزرگ‌تر شده و سپس، سیستم دچار انطباق‌پذیری کمتری خواهد شد. یکی از دلایل اصلی این امر بدین صورت است که به هنگام تغییر توزیع یک نوع کار، تمامی ماشین‌ها در صف خود دارای کار می‌باشند. در مثال نشان‌داده‌شده، سه ماشین در ازای نخستین بازه در حیطهٔ کار نوع ۱ تخصصی عمل می‌کنند. زمانی که رویهٔ توزیع تغییر پیدا می‌کند، کار نوع ۲ برجسته‌تر می‌شود. ماشین ۲ نیز که قبلاً در این نوع کار حرفه‌ای بوده است جهت پیشنهاد کار، مزیت‌های قابل‌توجهی پیدا می‌کند. تحت چنین شرایطی، آستانهٔ پیشنهاد کمتر از قبل با سایر دستگاه‌ها مقایسه می‌شود. در زمانی که دستگاه‌های دیگر صف‌های خود را تخلیه می‌کنند، صف دستگاه ۲ درهم‌ریخته شده و چرخه‌های زمانی بزرگ‌تری در صف حضور پیدا می‌کنند. همین موضوع به ازای دستگاه ۳ نیز در طی دورهٔ سوم رخ می‌دهد. این عملکرد از چندین مکانیسم $R-Wasp$ نشأت گرفته می‌شود و بدین سبب، به هنگام تغییر توزیع بسیار موفق عمل خواهد کرد. بهبود عملکرد در طی نسخهٔ پایهٔ $R-Wasp$ احتمالاً موجب کاهش این مدت‌زمان انطباق متناسب با تغییرات توزیع کار مربوطه می‌شود. از آن جایی که اعمال تغییر مکانیسم $R-Wasp$ به صورت مستقیم امکان‌پذیر است؛ پس می‌توان گفت که به‌جای اعمال تغییر، بهتر است در اکثر موارد، عملکردها به همان صورت حفظ شوند.



شکل ۱: نمونهٔ اجرای چهار دستگاه که نشان می‌دهد چه مدت‌زمان آستانه‌ها به ازای هر تغییر منطبق باقی می‌مانند.



شکل ۲: نمونهٔ اجرای چهار دستگاه که رشد صف را برای هر دستگاه در طول زمان نشان می‌دهد.

۱-۳-۲- حافظهٔ تخمین تراکم (روش پیشنهادی)

به‌جای ذخیره‌کردن تنها نقاط تکی در حافظه، پیشنهاد می‌شود خوشه‌هایی از نقاط در هر مدخل حافظه ذخیره شوند و مدلی از هر نقاط در هر خوشه ایجاد شود. اگر چه ما قادر به ذخیره‌کردن نقاط بیشتری خواهیم بود، سربار محاسباتی برای حافظه کم می‌شود.

اولین سیستم حافظهٔ تخمین تراکم، DEM_c، از خوشه‌بندی اقلیدسی افزایشی جهت ذخیره‌کردن نقاط در حافظه استفاده می‌کند. هر حافظهٔ ماشین شامل تعداد محدودی از مدخل‌های حافظه است. هر مدخل حافظه، یک خوشه از نقاط ذخیره‌شده و وضعیت‌های ماشین در یک زمان به‌خصوص است. آن نقاط معادل نقاط ذخیره‌شده برای حافظهٔ استاندارد است: آستانه‌های پاسخ θ_w و توزیع نوع کار D_t است.

هر مدخل حافظه، میانگین مقادیر تمام نقاط در خوشهٔ خودش را گرفته و یک مدل خوشه‌ای سادهٔ تشکیل شده از مرکز توزیع c_d و یک مرکز آستانهٔ c_θ را ایجاد می‌کند. این مقادیر زمان فعل و انفعالاتی با آن مدخل حافظه استفاده می‌شود. حافظه با فرکانس یکسان، همانند حافظهٔ استاندارد به‌روزرسانی می‌شود. وقتی نقطهٔ جدیدی ذخیره می‌شود، یک مدخل حافظهٔ جدید ایجاد می‌شود که شامل تنها آن نقطه است و به حافظه اضافه می‌شود. آن‌گاه دو مدخل در حافظه که مراکز توزیعشان نزدیک به یکدیگر هست را پیدا نموده و خوشه‌های آن‌ها درون یک مدخل حافظهٔ تکی ادغام می‌شود و دوباره c_d و c_θ محاسبه می‌شوند. یک مدخل مانند روش‌های مدل حافظهٔ استاندارد بازیابی می‌شود.

دومین حافظهٔ تخمین تراکم، DEM_g، از خوشه‌بندی گاوسی افزایشی جهت ذخیره‌کردن نقاط در حافظه استفاده می‌کند. استفاده از مدل گاوسی بازیابی مدخل‌های حافظه را بعد از تغییر توزیع، بهبود می‌بخشد. علاوه بر c_d و c_θ، یک مدل گاوسی از توزیع‌های نوع کار در آن خوشه، m_d، را ایجاد می‌کند. برای خوشه‌هایی با تعداد کمتر از ۱۰ نقطه، مدل انباشته‌شده به‌وسیلهٔ نقاط تصادفی اضافی اطراف c_d است (به طور یکنواخت توزیع شده در هر بعد در فاصله $[-0.125,0.125]$). به‌جای محاسبهٔ فاصلهٔ بین توزیع نوع کار جاری و توزیع در هر مدخل حافظه، مدل گاوسی در هر مدخل به محاسبهٔ احتمال آن مدخل حافظه که متعلق به مدل گاوسی است استفاده می‌کند. آن مدخل با بالاترین احتمال انتخاب می‌شود و آستانه‌های ماشین به c_θ تغییر می‌کنند. همهٔ دیگر اجزاء سیستم حافظه شبیه DEM_c هستند.

سومین حافظهٔ تخمین تراکم، DEM_{gw} می‌باشد. DEM_{gw} DEM_g را با استفاده از یک مدل گاوسی در سرتاسر سیستم حافظه توسعه می‌دهد. علاوه بر استفاده از این مدل محیطی، زمانی که مدخلی از حافظه بازیابی می‌شود، این مدل جهت ارزیابی احتمال، وقتی‌که نقاط جدیدی به حافظه اضافه می‌شوند، استفاده می‌شود. اندازه‌گیری مسافت میان مدخل‌ها زمانی تصمیم به ادغام می‌شوند که یک میانگین احتمالی محاسبه شود. میانگین احتمال از هر نقطه در مدخل ۱ در آن مدل، برای مدخل ۲ می‌باشد و برعکس. دو مدخل با بالاترین احتمال، ادغام می‌شوند. مانند DEM_g، مدل گاوسی در DEM_{gw} از بازیابی یک مدخل بعد از تغییر توزیع اساسی که کشف شد، استفاده می‌شود. به‌جای تغییردادن آستانهٔ ماشین به C_θ، یک وزن مرکزی جدید محاسبه شده است. برای هر نقطه در مدخل حافظه، یک وزن w_j به‌وسیلهٔ یافتن احتمالی که کار نوع توزیع برای نقطهٔ j قسمتی از m_d است محاسبه شده است، سپس وزن‌ها به‌وسیلهٔ تقسیم آن‌ها بر جمع کل وزن‌ها نرمال می‌شوند. یک مجموعهٔ آستانه‌های وزن‌دهی‌شده به‌وسیلهٔ ضرب وزن برای هر نقطه، از طریق آستانهٔ θ_W تشکیل داده می‌شوند. وزن مرکزی wc_θ میانگین تمام آستانه‌های وزن‌دهی‌شده است.

پس به طور کلی سه حافظه پیشنهادی به صورت زیر تعریف می‌شوند:

حافظهٔ تخمین تراکم خوشه‌بندی اقلیدسی افزایشی: این الگوریتم با خوشه‌بندی افزایشی، راه‌حل‌ها را درون مدخل‌های حافظه مجزا می‌کند. مراکز خوشه به عنوان مدل‌ها در حافظه استفاده می‌شوند. این ساده‌ترین نوع حافظهٔ تخمین تراکم است.

حافظهٔ تخمین تراکم خوشه‌بندی گاوسی افزایشی: این الگوریتم از خوشه‌بندی افزایشی به شکل مدخل‌های حافظه استفاده می‌کند، سپس مدل‌های گاوسی را برای هر مدخل از حافظه ایجاد می‌کند. این اجرا تخمین تراکم خوبی از راه‌حل‌های قبلی فراهم می‌کند.

حافظهٔ تخمین تراکم مدل ترکیبی گاوسی: برای مسائلی که زمان بیشتری برای ساخت مدل دارند، یک مدل ترکیبی گاوسی از حافظهٔ تخمین تراکم ارائه شده است. این پیاده‌سازی به ایجاد یک مدل روی همهٔ راه‌حل‌ها در حافظه نیاز دارد که هزینهٔ پیاده‌سازی بیشتری نسبت به خوشه‌بندی افزایشی دارد.

۱-۱-۳-۲- ذخیره‌سازی راه‌حل‌ها در حافظهٔ تخمین تراکم

در حافظهٔ استاندارد، یک نقطه اصولاً با استفاده از راهکار جایگزینی در حافظه ذخیره می‌شود. شایان ذکر می‌باشد که این رویکرد وظیفهٔ تصمیم‌گیری راجع به نحوهٔ جایگزینی یک نقطه درون یکی از مدخل‌های حافظهٔ موجود بر عهده دارد. اکثر راهکارهای جایگزینی، تنوع در حافظه را حفظ می‌کنند. به عنوان مثال، راهکار جایگزینی ابتدا یک نقطهٔ جدید به حافظه اضافه می‌کند، سپس دو مدخل حافظهٔ نزدیک به یکدیگر پیدا کرده و آن که از شایستگی کم‌تری برخوردار می‌باشد را حذف می‌کند.

به غیر از مدخل‌های از بین رفته، حافظهٔ تخمین تراکم مدخل‌ها را با هم ادغام می‌نماید. در وهلهٔ نخست، نقطهٔ جدید اضافه‌شده به حافظه به دلیل ایجاد مدخل حافظهٔ جدید، مورد استفاده قرار می‌گیرد. پس از این‌که مدل‌ها برای این مدخل جدید ساخته شد، دو مدخل که در حافظه بیشتر از همه به یکدیگر شبیه می‌باشند پیدا شده و ادغام می‌گردند. زمانی که فضای جستجوی جدیدی برای نخستین بار ساخته می‌شود، نقطهٔ جدید نیز جهت آغاز مدل‌سازی این ناحیه از حافظه، مورد استفاده قرار خواهد گرفت.

به‌واسطهٔ افزایش نقاط خوشه‌بندی، ناحیه‌های جدید فضای جستجو را می‌توان بسته به تغییرات محیط پویا مدل‌سازی کرد. به محضی که خوشه‌ها ادغام شدند، خوشه‌ها متشکل از چندین نقطه خواهند شد. هم‌چنین، چنان‌چه تراکم آن نقاط افزایش یابد، یک مدل مناسبی نیز جهت برآورد مناسب از مکان راه‌حل‌های خوب در آن ناحیه ارائه خواهد شد.

شایان ذکر می‌باشد که حافظهٔ تخمین تراکم همانند حافظهٔ استاندارد شدیداً به الگوریتم بهینه‌سازی یا یادگیری وابسته است. حافظهٔ تخمین تراکم اصولاً مدل‌ها را بر مبنای نقاط ذخیره‌شده در حافظه می‌سازد، ولیکن اگر الگوریتم مزبور راه‌حل‌های ضعیفی را ذخیره کند، آن‌گاه مدل‌های موجود در حافظه راه‌حل خوبی را نمایش نخواهند داد. در صورتی که راه‌حل‌های خوبی از سوی الگوریتم معرفی گردد، آن‌گاه حافظهٔ تخمین تراکم به الگوریتم کمک خواهد کرد تا ناحیهٔ امیدبخش موردنظر را کشف کند، ولیکن قبل از آن الگوریتم باید بتواند به‌خوبی عملیات جستجو را در آن ناحیه انجام دهد.

زمانی که نقاط جدید در حافظه ذخیره می‌شوند تا حد زیادی به الگوریتم‌های بهینه‌سازی یا یادگیری و نوع مسئله بستگی دارد. زمانی که از الگوریتم تکاملی استفاده می‌شود، نقطهٔ جدید ممکن است در هر چند نسل ذخیره گردد. هم‌چنین، موقعی که از الگوریتم یادگیری تقویتی بهره گرفته می‌شود، بهترین راه این است که نقاط جدید در زمان یادگرفتن راه‌حل‌های مطلوب ذخیره شوند (در ذخیرهٔ نقاط جدید یک بار راه‌حل‌های خوب را آموزش ببینند). در بعضی از مسائل هنگامی‌که تغییرات به‌وضوح قابل‌رؤیت باشد، بهترین حالت این است که راه‌حل‌ها قبل از اعمال تغییر ثبت شوند. در رابطه با مسائل دیگری که تغییرات در آن به صورت تدریجی رخ می‌دهد، بهتر است که نقاط به طور

دوره‌ای ذخیره شوند. ذخیره‌سازی یک راه‌حل، درست بعد از اعمال تغییر، اصلاً کار مطلوبی نیست زیرا که هیچ فرصتی برای تطبیق آن وجود ندارد.

سربار محاسباتی استفاده از حافظهٔ تخمین تراکم به مسئله و مدل‌های احتمالاتی به‌کارگرفته‌شده در حافظه وابسته می‌باشد. زمانی که نقاط خوشه‌بندی افزایش می‌یابد، می‌توان انتخاب دو مدخل برای ادغام در $O(m^2)$ انجام داد. در این حالت، m تعداد مدخل‌های حافظه تلقی می‌شود. وقتی از مدل‌های گاوسی درون حافظهٔ تخمین تراکم استفاده می‌شود، کوواریانس در $O(nd^2)$ محاسبه می‌گردد؛ که n تعداد نقاط موجود در مدخل حافظه و d تعداد ابعاد موجود در داده‌های کنترلی یا محیطی می‌باشد. از آن جایی که حافظه به چندین خوشه تقسیم می‌شود، n عمدتاً کم‌تر از تعداد کل نقاط موجود در حافظه انتخاب می‌گردد. اگرچه استفاده از حافظه، مقداری سربار محاسباتی دارد، زمان لازم برای ارزیابی راه‌حل‌ها به طور معمول بر زمان موردنیاز جهت نگهداری حافظه غالب است. از آن جایی که حافظهٔ تخمین تراکم اساساً به ارزیابی راه‌حل‌های اضافی احتیاجی ندارد، درنتیجه سربار حافظهٔ تخمین تراکم تنها به‌واسطهٔ ساخت و نگهداری مدل‌های احتمالاتی موجود در حافظه حاصل می‌شود.

در بعضی از مسائلی که داده‌های کنترلی یا محیطی، ابعاد بسیار زیادی دارند، ممکن است زمان زیادی جهت ساخت حافظه صرف شود و قبل از بازیابی از حافظه می‌تواند در جهت بهبود کارایی مؤثر باشد. هر چه ابعاد داده بیشتر باشد، نقاط بیشتری جهت ارائهٔ تخمین تراکم خوب موردنیاز است. در رابطه با این مسائل، بهتر است که ابتدا حافظه ساخته شود تا این که به صورت خالی باشد.

۲-۱-۳-۲- بازیابی راه‌حل‌ها از حافظهٔ تخمین تراکم

راه‌حل‌ها در حافظهٔ تخمین تراکم شبیه به حافظهٔ استاندارد بازیابی می‌شوند. پیرامون الگوریتم‌های جستجوی مبتنی بر جمعیت، راه‌حل‌ها ممکن است از حافظه بازیابی شوند یا در کل اجرا و یا بلافاصله بعد از تغییر شناسایی شوند. رایج است که راه‌حل‌ها در هر نسل از الگوریتم تکاملی از حافظه بازیابی می‌شوند.

زمانی که مدخل‌های حافظه تنها به هنگام ذخیره‌سازی نقاط جدید به حافظه تغییر پیدا می‌کنند، راه‌حل‌های ارائه شده به‌واسطهٔ بازیابی از حافظه می‌تواند در هر نسل به‌وسیلهٔ الگوریتم جستجو مورد استفاده قرار گیرد. برای الگوریتم‌های یادگیری که تنها یک نقطه را جستجو می‌کنند، راه‌حل‌ها می‌بایست از حافظه با تکرار کم‌تر و انتخاب بیشتر بازیابی شوند. از آن جایی که بازیابی از حافظه در هر مرحله باعث می‌شود که الگوریتم راه‌حل مربوطه را اصلاح کند، راه‌حل‌ها یا به صورت دوره‌ای یا در هر نسل بازیابی می‌شوند. در مقالهٔ حاضر، راه‌حل حاصل از حافظه، زمانی بازیابی می‌گردد که راه‌حل جاری ضعیف عمل کرده و محیط کنونی نیز کاملاً با یکی از مدخل‌های حافظه منطبق باشد.

برخلاف حافظهٔ استاندارد، راه‌حل‌های بازیابی‌شده از حافظه چندان شباهتی به تک‌تک راه‌حل‌های ذخیره‌شده ندارند. به‌واسطهٔ تجمع بسیاری از نقاط موجود در مدخل حافظه، راه‌حل بازیابی‌شده از حافظه تحت تأثیر راه‌حل‌های قبلی که در محیط مشابه به‌خوبی عمل می‌کرده‌اند، قرار خواهد گرفت.

۳-۱-۳-۲- ساختار حافظهٔ تخمین تراکم

یک حافظهٔ تخمین تراکم نقاط را در تعداد محدودی از مدخل‌های حافظه ذخیره‌سازی می‌کند.

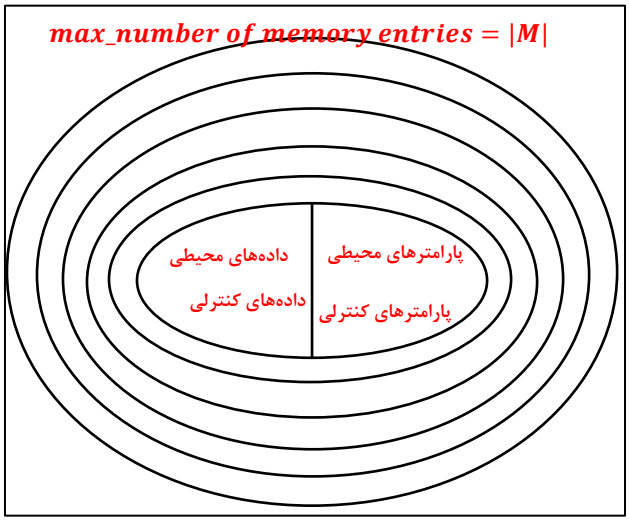
اطلاعات ذخیره‌شده در یک مدخل حافظه را می‌توان به دو دسته تقسیم کرد: اطلاعات محیطی و اطلاعات کنترلی.

اطلاعات محیطی کمک می‌کند تا تصویری از وضعیت فعلی مسئله داشته باشیم. در صورت تغییر مسئله، اطلاعات محیطی ممکن است برای تعیین شباهت بین محیط فعلی و محیطی که در مدخل حافظه ذخیره شده؛ استفاده شود.

اطلاعات محیطی برای حفظ حافظه استفاده می‌شود و زمانی که مدخل‌های جدید موجود هستند تصمیم‌گیری می‌کنند که مدخل‌ها باید در حافظه ذخیره شوند. اطلاعات کنترلی شامل اطلاعاتی است که در فرآیند جستجو استفاده می‌شود. برای مثال اطلاعات کنترلی ممکن است فقط یک راه‌حل ذخیره‌شده در قبل باشد که می‌تواند دوباره در جمعیت جاگذاری شود یا ممکن است حاوی یک مدل احتمالی از جمعیت بر اساس الگوریتم تکاملی باشد که می‌تواند برای دوباره مقداردهی اولیهٔ جمعیت بعد از یک تغییر استفاده شود. در حافظهٔ استاندارد، هر مدخل حافظه می‌تواند یک نقطه تکی را ذخیره کند؛ ولی در حافظهٔ تخمین تراکم، هر مدخل ممکن است تعدادی نقاط را ذخیره کند. یک حافظه ممکن است تا $|M|$ مدخل داشته باشد. هر مدخل اصولاً از مجموعه‌ای نقاط، یک مدل از اطلاعات محیطی در آن نقاط و همچنین یک مدل از اطلاعات کنترلی در آن نقاط تشکیل شده است.

شکل ۳ ساختار یک حافظهٔ تخمین تراکم را نشان می‌دهد. مدل‌ها در حافظهٔ تخمین تراکم ممکن است ساده‌شده‌ای از میانگین تمام نقاط باشند یا از مدل‌های احتمالی پیچیده‌ای نیز استفاده کرده باشند. در این مقاله آن مدل‌های عامی که به کار برده می‌شوند تا یک مدخل حافظه را نشان دهند مدل خوشه‌بندی گاوسی چندمتغیره هستند. در نسخهٔ گاوسی، متوسط و کوواریانس همهٔ نقاط موجود در مدخل حافظه به جهت توصیف مدل مورد استفاده قرار می‌گیرند. زمانی که مدخل‌ها برای نخستین بار ساخته می‌شوند، ممکن است برای محاسبهٔ کوواریانس معتبر، نقاط کافی نباشد. در این حالت، نقاط تصادفی اطراف میانگین به صورت موقت برای تکمیل مدل اضافه می‌شوند. در ادامه، راجع به حافظهٔ تخمین تراکم که از مدل‌های گاوسی بهره می‌گیرند صحبت خواهد شد.

از آن جایی که مدخل حافظه شامل نقاط متعددی می‌باشد، فقط مدل محیطی و مدل کنترلی جهت برقراری تعامل با آن مدخل لازم است. هم‌چنین به سبب این‌که مدخل‌ها از هزاران نقطه تشکیل شده‌اند، این مسئله میزان سربار حافظهٔ تخمین تراکم را پایین نگه می‌دارد.

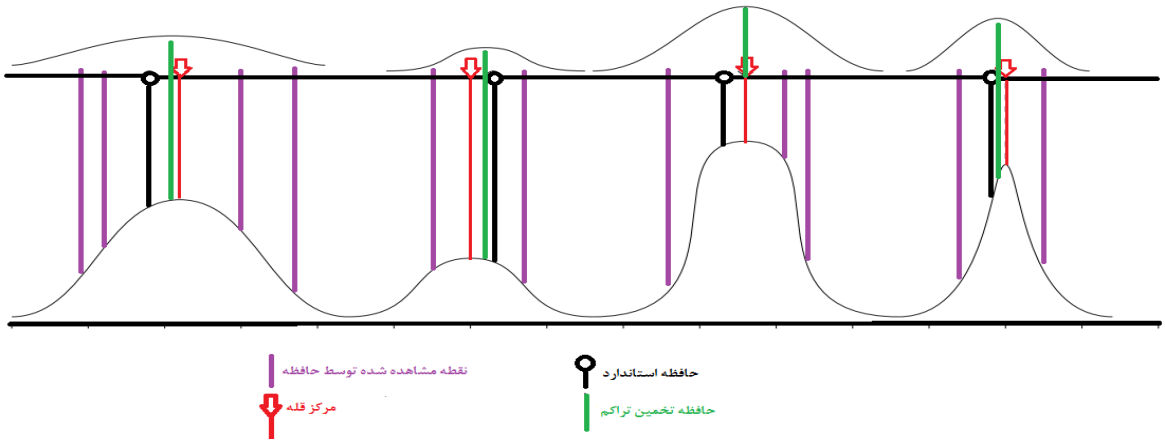


شکل ۳: حافظه تخمین تراکم از تعداد محدودی از مدخل‌ها ایجاد شده است. هر مدخل شامل مجموعه‌ای از نقاط است که هر نقطه شامل دادهٔ محیطی و کنترلی از زمانی که آن نقطه ذخیره شده بوده است.

شکل ۴ نحوهٔ عملکرد متفاوت حافظهٔ تخمین تراکم را از حافظهٔ استاندارد نشان می‌دهد. در این مثال، یک فضای جستجوی یک‌بعدی وجود دارد که شامل چهار قله با شکل‌ها و شایستگی مختلف است. فرض کنید که الگوریتم جستجو سعی دارد تا نقاط علامت‌گذاری‌شده با خطوط ممتد را طی مدت‌زمانی مشخص درون حافظه ثبت کند. حافظهٔ استاندارد نقطهٔ با بهترین شایستگی را انتخاب می‌کند؛ از این رو همیشه نقطهٔ مزبور نزدیک‌ترین نقطه به مرکز حافظه

است. حافظهٔ تخمین تراکم نقاط را بر اساس قله خوشه‌بندی می‌کند و آن‌گاه به ازای هر خوشه یک مدل می‌سازد. در بالای شکل یک مدل گاوسی به ازای هر قله نشان داده شده است. پیرامون این مثال، مقدار میانگین محاسبه‌شده به ازای هر خوشه شایستگی بیش‌تری دارد و نزدیک‌ترین نقطه به قلهٔ بیشینه می‌باشد. فراتر از استفاده از هندسهٔ اقلیدسی، مدل‌های گاوسی اطلاعات بیش‌تری را به هنگام افزودن یک نقطهٔ جدید به حافظه ارائه می‌کنند. قلهٔ در قسمت چپ به حد کافی بزرگ است و هر چه نقاط از حد متوسط دورتر باشند، احتمال پذیرفته‌شدن آن‌ها نیز بیش‌تر است. پیرامون خوشه‌هایی که تا به حال مدل دقیقی برای آن‌ها ارائه نشده است، مدل می‌بایست ضمن افزودن نقاط به حافظه اصلاح گردد. چنان‌چه مدل‌ها دقیق‌تر از قبل شوند، راه‌حل‌های بازیابی شده از حافظه نیز بهتر خواهند شد. الگوریتم جستجو در راستای بهبود راه‌حل‌های بازیابی‌شده از حافظه عمل می‌کند. این حلقهٔ بازخورد نیز در جهت بهبود مدل‌های ذخیره‌شده در حافظه گام برمی‌دارد.

یک نوع از خوشه‌بندی برای حافظه‌های تخمین تراکم، خوشه‌بندی گاوسی[1] است که مدل‌های تخمین تراکم غنی‌تری برای استفادهٔ حافظه فراهم می‌کند. در این روش، هر مدخل حافظه یک مدل گاوسی از نقاط درون آن مدخل را محاسبه می‌کند. ماتریس میانگین و کوواریانس برای محاسبهٔ این‌که نقطهٔ جدید به خوشهٔ موجود متعلق است و یا احتمال این‌که دو خوشه می‌بایست ادغام شوند، استفاده می‌شود.



شکل ٤: حافظهٔ تخمین تراکم مثالی از یک فضای جستجو با ٤ قلهٔ مختلف است. هر دو حافظهٔ استاندارد و تخمین تراکم نقاط یکسانی را می‌بینند. مدخل‌های حافظه برای حافظهٔ استاندارد با نقطه نشان داده شده است، در صورتی که مدل‌های گاوسی محاسبه‌شده برای هر خوشه در حافظهٔ تخمین تراکم در بالای آن فضای جستجو نشان داده شده است.

۳- آزمایش‌ها

در این بخش روش‌های حافظهٔ پیشنهادی با روش‌های زیر در سناریوهای مختلف مورد مقایسه قرار گرفته‌اند:

$R-wasps$ استاندارد، نسخهٔ حافظهٔ بهبودیافته از $R-wasps$، حافظهٔ استاندارد و حافظهٔ $\infty-$.

حافظهٔ $\infty-$ دقیقاً شبیه به سیستم حافظهٔ استاندارد است ولی محدودیت در تعداد مدخل‌های حافظه ندارد. هر یک از حافظه‌های دیگر، حداکثر ذخیرهٔ ۵ مدخل را اجازه می‌دهند (۵ خوشه برای حافظه‌های تخمین تراکم).

ما مسائل را با چهار ماشین و چهار نوع کار بررسی کرده‌ایم. هر سناریو ۱۵۰۰۰۰ واحد زمان طول می‌کشد و به ۵۰ دورهٔ ۳۰۰۰ واحد زمان تقسیم می‌شود. در شروع هر دوره، نوع کار توزیع باعث ایجاد کار جدید وارده شده و تغییر آن به توزیع جدید می‌شود. این توزیع به طور تصادفی از ۱۰ توزیع ساخته‌شده در ابتدای این سناریو انتخاب شده است. توزیع برای این دوره به طور تصادفی‌ست؛ بنابراین توزیع‌ها دقیقاً تکرار نمی‌شوند.

1. Gaussian clustering

برای شناسایی تغییرات توزیع، از $\phi = 2.5$، $\varepsilon = 0.25$ و $\omega_1 = \omega_2 = 100N$ استفاده شده است؛ در حالی که N تعداد انواع کار است. مقادیر پارامترها برای $R - wasps$ در بخش‌های قبل توضیح داده شده است. سناریوها با سه مقدار $\ell \in \{1.00, 1.25, 1.50\}$ اجرا شده و روش پیشنهادی ۳۰ بار مورد اجرا قرار داده شده است.

برای ارزیابی سناریوها، از چهار آمار استفاده شده است: توان عملیاتی ($throughput$)، راه‌اندازی ($setups$)، زمان چرخه ($cycle\ time$)، و طول صف ($queuelengh$). توان عملیاتی بر اساس درصد تمام کارها در سناریو که توسط ماشین پردازش می‌شوند اندازه‌گیری می‌شود. برپای شامل تعداد کل برپایی به‌وسیلهٔ همه ماشین‌ها در سیستم است. زمان چرخه، میانگین زمانی است که یک کار در سیستم وارد می‌شود، پردازش می‌گردد و پایان می‌یابد. طول صف، متوسط تعداد کارها در یک صف ماشین است.

نتایج

جدول‌های ۱، ۳ و ۵ میانگین نتایج از ۲۰ سناریو با مقادیر $\ell \in \{1.00, 1.25, 1.50\}$ را نشان می‌دهند. در جدول‌های ۲، ۴ و ۶ روش‌ها با هم مقایسه می‌شوند و درصد بهبود روش‌های حافظه نشان داده شده است. وقتی نتایج از لحاظ آماری معنادار هستند، نتایج بر این اساس مشخص است. برای تمام آزمایش‌ها در این قسمت، آمار معنادار نتایج با استفاده از آزمایش *kruskwl-walis* با در نظر گرفتن اطمینان ۹۵٪ آزمایش شده است. آزمایش *kruskwl-walis*، یک راه آنالیز واریانس به‌وسیلهٔ رتبه‌بندی، یک معادلهٔ بدون پارامتریک تجزیه‌وتحلیل یک‌طرفه کلاسیک واریانس ($ANOVA$) است که فرض نمی‌کند داده‌ها در یک توزیع نرمال آمده‌اند.

وقتی $\ell = 1$، سیستم یک بار، متوسط به بالا دارد؛ تمام ۶ روش توان عملیاتی بیشتر از ۹۸٪ دارند. کارهای ناتمام باقی‌مانده که در انتهای سناریو موجود است بیشتر به این دلیل است که کارهای واردشده خیلی دیر پردازش شده‌اند، درحالی‌که کارها به طور بالقوه یک‌بار می‌توانند وارد شوند. از آن جا که نسخهٔ استاندارد $R - wasps$ بیش از ۹۹٪ کارها را تکمیل می‌کند، تعداد زیادی جای خالی برای بهبود در توان عملیاتی وجود ندارد. حافظه‌های تخمین تراکم به طور قابل‌توجهی، تعداد راه‌اندازی، زمان چرخه و طول صف را برای این سناریوها کاهش می‌دهند. حافظهٔ استاندارد درواقع باعث آسیب در همهٔ ناحیه‌ها می‌شود.

وقتی‌که $\ell = 1{,}25$ است سیستم، بار بالا دارد. نتایج برای ۶ روش متنوع است. روش استاندارد $R - wasps$ فقط ۸۴ تا ۸۹ درصد وظایف را میانگین انجام می‌دهد، در حالی که وقتی حافظه استفاده شود این میانگین بالای ۹۳٪ برای هر نوع حافظه است. یک‌بار دیگر، حافظه‌های تخمین تراکم عملکرد بهتری، با کاهش در تعداد راه‌اندازی، میانگین زمان چرخه و میانگین طول صف دارند.

وقتی‌که $\ell = 1{,}5$ است، سیستم بیش‌ازحد بار دارد. در مقایسه با سناریوهای با بارهای بالاتر، توان عملیاتی برای همهٔ روش‌ها با میانگین کمتر از ۸۰٪ برای $R - wasps$ استاندارد کاهش می‌یابد. وجود حافظهٔ تخمین تراکم باعث بهبود قابل‌توجهی در $R - wasps$ استاندارد می‌شود.

DEM_g بهترین توان عملیاتی و زمان چرخه را دارد. بیشترین بهبود در $R - wasps$ استاندارد، کاهش تعداد راه‌اندازی می‌باشد.

جدول ۱: متوسط نتایج برای سناریوهایی با $\ell = 1.00$

روش‌ها	توان عملیاتی	راه‌اندازی	زمان چرخه	طول صف
$R - wasps$	۹۹/۰۵	۲۰۲۹/۲۵	۲۰/۰۲	۵۹/۰۴
$Memory$	۹۸/۶۴	۲۴۸۷/۲۰	۲۲/۹۶	۶۸/۹۴
$Memory - \infty$	۹۹/۲۷	۱۹۱۳/۴۵	۱۶/۲۰	۴۸/۰۳
DEM_c	۹۹/۴۳	۱۳۷۵/۴۵	۱۱/۳۱	۳۳/۷۶
DEM_g	۹۹/۶۵	۱۲۱۳/۱۰	۱۰/۱۰	۲۹/۸۰
DEM_{gw}	۹۹/۳۶	۱۵۵۰/۳۰	۱۳/۹۳	۴۱/۷۱

جدول ۲: درصد بهبود روش ۱ روی روش ۲ برای هر معیار با $\ell = 1.00$ (نتایج از نظر آماری به اطمینان ۹۵٪ معنی‌دار هستند که با + یا − نوشته می‌شوند)

روش ۱	روش ۲	توان عملیاتی	راه‌اندازی	زمان چرخه	طول صف
$Memory$	$R - wasps$	−۰/۴۱	−۲۲/۱۲ (−)	−۱۴/۶۶	−۱۶/۷۶
$Memory - \infty$	$R - wasps$	۰/۲۲	۵/۷۱	۱۹/۱۰	۱۸/۶۶
$Memory - \infty$	$Memory$	۰/۶۴ (+)	۲۲/۷۹ (+)	۲۹/۴۴ (+)	۳۰/۳۳ (+)
DEM_c	$R - wasps$	۰/۳۸	۳۲/۴۲ (+)	۴۳/۵۳ (+)	۴۲/۸۳ (+)
DEM_c	$Memory$	۰/۸۰ (+)	۴۴/۶۶ (+)	۵۰/۷۵ (+)	۵۱/۰۳ (+)
DEM_c	$Memory - \infty$	۰/۱۶	۲۸/۳۳ (+)	۳۰/۱۹ (+)	۲۹/۷۱
DEM_g	$R - wasps$	۰/۶۰ (+)	۴۰/۲۲ (+)	۴۹/۵۸ (+)	۴۹/۵۲ (+)
DEM_g	$Memory$	۱/۰۲ (+)	۵۱/۰۵ (+)	۵۶/۰۲ (+)	۵۶/۷۷ (+)
DEM_g	$Memory - \infty$	۰/۳۸ (+)	۳۶/۶۰ (+)	۳۷/۶۷ (+)	۳۷/۹۵ (+)
DEM_g	DEM_c	۰/۲۲	۱۱/۵۵	۱۰/۷۱	۱۱/۷۲
DEM_{gw}	$R - wasps$	۰/۳۲	۲۳/۶۰ (+)	۳۰/۴۵ (+)	۲۹/۳۶
DEM_{gw}	$Memory$	۰/۷۴ (+)	۳۷/۴۴ (+)	۳۹/۳۴ (+)	۳۹/۵۰ (+)
DEM_{gw}	$Memory - \infty$	۰/۰۹	۱۸/۹۸	۱۴/۰۳	۱۳/۱۶
DEM_{gw}	DEM_c	−۰/۰۷	−۱۳/۰۴	−۲۳/۱۶	−۲۳/۵۵
DEM_{gw}	DEM_g	−۰/۲۸	−۲۷/۸۰	−۳۷/۹۳	−۳۹/۹۵

جدول ۳: متوسط نتایج برای سناریوهایی با $\ell = 1.25$

روش‌ها	توان عملیاتی	راه‌اندازی	زمان چرخه	طول صف
$R - wasps$	۸۹/۸۴	۲۵۱۴/۰۵	۱۳۷/۷۰	۵۳۸/۰۶
$Memory$	۹۳/۱۶	۱۸۲۱/۰۰	۱۰۶/۷۹	۴۰۳/۸۷
$Memory - \infty$	۹۳/۱۰	۱۸۴۱/۴۵	۱۱۰/۳۷	۴۱۹/۲۸
DEM_c	۹۵/۲۹	۱۲۰۸/۱۰	۸۵/۸۴	۳۲۳/۹۱
DEM_g	۹۴/۸۶	۱۲۹۹/۰۵	۹۰/۷۹	۳۴۴/۴۱
DEM_{gw}	۹۶/۵۰	۸۲۰/۹۵	۶۱/۱۷	۲۳۴/۵۱

جدول ٤: درصد بهبود روش ۱ روی روش ۲ برای هر معیار با $\ell = 1.25$ (نتایج از نظر آماری به اطمینان ۹۵٪ معنی‌دار هستند که با + یا − نوشته می‌شوند)

روش ۱	روش ۲	توان عملیاتی	راه‌اندازی	زمان چرخه	طول صف
$Memory$	$R-wasps$	۳/۷۰ (+)	۲۷/۵۷ (+)	۲۲/٤٤ (+)	۲٤/۳٤ (+)
$Memory\ -\infty$	$R-wasps$	۳/٦۳ (+)	۲٦/۷۵	۱۹/۸۵	۲۲/۹۸
$Memory$	$Memory\ -\infty$	-۰/۰۷	-۱/۱۲	-۳/۳۵	-۱/۷۹
DEM_c	$R-wasps$	٦/۷۰ (+)	۵۱/۹۵ (+)	۳۷/٦٦ (+)	۳۸/٦۳ (+)
DEM_c	$Memory$	۲/۲۹ (+)	۳۳/٦٦ (+)	۱۹/٦۲	۱۸/۸۹
DEM_c	$Memory\ -\infty$	۲/۳٦ (+)	۳٤/۳۹ (+)	۲۲/۲۲	۲۰/۳۲ (+)
DEM_g	$R-wasps$	۵/۵۹ (+)	٤۸/۳۳ (+)	۳٤/۰٦ (+)	۳۵/٦۰ (+)
DEM_g	$Memory$	۱/۸۲ (+)	۲۸/٦٦ (+)	۱٤/۹۸	۱٤/۸۸
DEM_g	$Memory\ -\infty$	۱/۸۹ (+)	۲۹/٤٦ (+)	۱۷/۷٤	۱٦/۳۸
DEM_g	DEM_c	-۰/٤۵	-۷/۵۳	-۵/۷۷	-٤/۹٤
DEM_{gw}	$R-wasps$	۷/٤۱ (+)	٦۵/۳۷ (+)	۵۵/۵۸ (+)	٤۹/٦٤ (+)
DEM_{gw}	$Memory$	۳/۵۸ (+)	۵٤/۹۲ (+)	٤۲/۷۲ (+)	۳۳/٤۵ (+)
DEM_{gw}	$Memory\ -\infty$	۳/٦۵ (+)	۵۵/٤۲ (+)	٤٤/۵۸ (+)	۳٤/٦۲ (+)
DEM_{gw}	DEM_c	۱/۲٦	۳۲/٦۵ (+)	۲۸/۷۵ (+)	۱۷/۹۵ (+)
DEM_{gw}	DEM_g	۱/۷۳	۳٦/۸۰ (+)	۳۲/٦۳ (+)	۲۱/۸۱ (+)

جدول ۵: متوسط نتایج برای سناریوهایی با $l = 1.50$

روش‌ها	توان عملیاتی	راه‌اندازی	زمان چرخه	طول صف
$R-wasps$	۷۹/۳۳	۱۹۳۵/۱۵	۲٦۲/۸۳	۱۲۲۹/٦۵
$Memory$	۸۲/۷۱	۱۱٦۹/۸۵	۲۳۳/۷۸	۱۰۷۳/۸۵
$Memory\ -\infty$	۸۱/۱۰	۱٤۸۹/۷۰	۲٤٤/۲٦	۱۱٤۸/۹۷
DEM_c	۸۱/۷۱	۱۳۷٤/۰۰	۲٤۱/۲۷	۱۱۱٤/۷٦
DEM_g	۸۳/۰۸	۱۰۵۷/۱۵	۲۲۵/۵۷	۱۰۵۱/۸۰
DEM_{gw}	۸۳/۰٦	۱۰٤٦/۰۰	۲۲٦/۸۷	۱۰٤۱/٦٦

جدول ٦: درصد بهبود روش ۱ روی روش ۲ برای هر معیار با $\ell = 1.50$ (نتایج از نظر آماری به اطمینان ۹۵٪ معنی‌دار هستند که با + یا − نوشته می‌شوند)

روش ۱	روش ۲	توان عملیاتی	راه‌اندازی	زمان چرخه	طول صف
$Memory$	$R-wasps$	٤/۲۷ (+)	۳۹/۵۵ (+)	۱۱/۰۵	۱۲/٦۷
$Memory\ -\infty$	$R-wasps$	۲/۲۳	۲۳/۰۲	۷/۰٦	٦/۵٦
Memory $-\infty$	$Memory$	-۱/۹۵	-۲۷/۳٤	-٤/۹٤	-۷/۰۰
DEM_c	$R-wasps$	۳/۰۰	۲۹/۰۰	۸/۲۰	۹/۳٤
DEM_c	$Memory$	-۱/۲۱	-۱۷/٤۵	-۳/۲۰	-۳/۸۲
DEM_c	$Memory\ -\infty$	۰/۷۵	۳٤/۳۹ (+)	۲۲/۲۲	۲۰/۳۲ (+)
DEM_g	$R-wasps$	٤/۷۳ (+)	٤۵/۳۷ (+)	۱٤/۱۸ (+)	۱٤/٤٦ (+)
DEM_g	$Memory$	۰/٤٤	۹/٦۳	۳/۵۱	۲/۰۵
DEM_g	$Memory\ -\infty$	۲/٤٤	۲۹/۰٤	۷/٦۵	۸/٤۵
DEM_g	DEM_c	۱/٦۷	۲۳/۰٦	٦/۵۱	۵/٦۵

روش ۱	روش ۲	توان عملیاتی	راه‌اندازی	زمان چرخه	طول صف
DEM_{gw}	$R-wasps$	٤/٧١ (+)	٤٥/٩٥ (+)	١٣/٦٨ (+)	١٥/٢٩ (+)
DEM_{gw}	$Memory$	٠/٤٢	١٠/٥٩	٢/٩٥	٣/٠٠
DEM_{gw}	$Memory-\infty$	٢/٤٢	٢٩/٧٦	٧/١٢	٩/٣٤
DEM_{gw}	DEM_c	١/٦٦	٢٣/٨٧	٥/٩٧	٦/٥٧
DEM_{gw}	DEM_g	−٠/٠٢	١/٠٥	−٠/٥٨	٠/٩٧

نتیجه‌گیری

در این مقاله یک حافظهٔ مبتنی بر توزیع تخمین تراکم برای حل مسئلهٔ هماهنگ‌سازی کارخانهٔ توزیع‌شدهٔ پویا ارائه شده است. نتایج حاصله نشان می‌دهد که حافظهٔ پیشنهادی نسبت به $R-wasps$ استاندارد و سایر روش‌ها دارای عملکرد مناسب‌تری در حل مسئلهٔ هماهنگ‌سازی کارخانهٔ توزیع‌شدهٔ پویا می‌باشد. هم‌چنین در حافظهٔ ارائه‌شده نقاط ضعف حافظهٔ استاندارد برطرف گردیده است. در یک حافظهٔ استاندارد فضای ذخیره‌سازی محدود می‌باشد و اگر تعداد راه‌حل‌های یک مسئله زیاد باشند، حافظه به‌سرعت پر می‌شود. حافظهٔ ارائه‌شده در این مقاله این ضعف را برطرف نموده است. در سیستم‌های حافظهٔ ارائه‌شده در این مقاله، به‌جای ذخیره‌کردن تنها نقاط تکی در حافظه، خوشه‌هایی از نقاط در هر مدخل حافظه ذخیره شوند و مدلی از هر نقاط در هر خوشه ایجاد می‌شود. این نوع حافظه، قادر به ذخیره‌کردن نقاط بیشتری بوده و سربار محاسباتی برای این نوع حافظه کم است. روش‌های حافظهٔ ارائه‌شده در این مقاله در سناریوهای مختلفی با دیگر روش‌ها مورد مقایسه قرار گرفته است. نتایج آماری ارائه‌شده نشان می‌دهند که روش‌های حافظهٔ ارائه‌شده از لحاظ عملکرد نسبت به سایر روش‌های مشابه بهتر می‌باشند. برای کارهای آتی پیشنهاد می‌شود که روش‌های حافظهٔ ارائه‌شده را با یکی از الگوریتم‌های تکاملی ترکیب نموده و برای حل مسئله‌های تولیدات کارگاهی پویا و محک قله‌های متحرک مورد بررسی قرار داد تا میزان عملکرد این روش‌ها در حل این مسئله‌ها نیز مورد بررسی قرار گیرد.

منابع

محمدپور، مجید و پروین، حمید. (۱۳۹۵). الگوریتم ژنتیک آشوب‌گونه مبتنی بر حافظه و خوشه‌بندی برای حل مسائل بهینه‌سازی پویا. *مجلهٔ مهندسی برق دانشگاه تبریز*، ۴۶(۳).

Bendtsen, C. N., & Krink, T. (2002). Dynamic memory model for non-stationary optimization. Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600),

Branke, J. (2012). *Evolutionary optimization in dynamic environments* (Vol. 3). Springer Science & Business Media.

Branke, J., & Mattfeld, D. C. (2002). Anticipatory scheduling for dynamic job shop problems. AIPS Workshop on On-line Planning and Scheduling,

Bravo, Y., Luque, G., & Alba, E. (2016). Global memory schemes for dynamic optimization. *Natural computing*, *15*, 319-333.

Campos, M., Bonabeau, E., Theraulaz, G., & Deneubourg, J.-L. (2000). Dynamic scheduling and division of labor in social insects. *Adaptive Behavior*, *8*(2), 83-95.

Chen, X., Zhang, D., & Zeng, X. (2015). A stable matching-based selection and memory enhanced MOEA/D for evolutionary dynamic multiobjective optimization. 2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI),

Chryssolouris, G., & Subramaniam, V. (2001). Dynamic scheduling of manufacturing job shops using genetic algorithms. *Journal of Intelligent Manufacturing*, *12*, 281-293.

Cicirello, V. A., & Smith, S. F. (2004). Wasp-like agents for distributed factory coordination. *Autonomous Agents and Multi-agent systems*, *8*, 237-266.

Eggermont, J., & Lenaerts, T. (2002). Dynamic optimization using evolutionary algorithms with a case-based memory. Proceedings of the Fourteenth Belgium-Netherlands Conference on Artificial Intelligence (BNAIC),

Eggermont, J., Lenaerts, T., Poyhonen, S., & Termier, A. (2001). Raising the dead: Extending evolutionary algorithms with a case-based memory. Genetic Programming: 4th European Conference, EuroGP 2001 Lake Como, Italy, April 18–20, 2001 Proceedings 4,

Mohammadpour, M., & Parvin, H. (2016). Genetic algorithm based on explicit memory for solving dynamic problems. *Journal of Advances in Computer Research*, *7*(2), 53-68.

Morrison, R. W., & De Jong, K. A. (1999). A test problem generator for non-stationary environments. Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406),

Nasiri, B., & Meybodi, M. R. (2012). Speciation based firefly algorithm for optimization in dynamic environments. *International Journal of Artificial Intelligence*, *8*(S12), 118-132.

Ozsoydan, F. B., & Baykasoglu, A. (2015). A multi-population firefly algorithm for dynamic optimization problems. 2015 IEEE International Conference on Evolving and Adaptive Intelligent Systems (EAIS),

Ramsey, C. L., & Grefenstette, J. J. (1993). Case-Based Initialization of Genetic Algorithms. ICGA,

Richter, H., & Yang, S. (2008a). Learning in abstract memory schemes for dynamic optimization. 2008 Fourth International Conference on Natural Computation,

Richter, H., & Yang, S. (2008b). Memory based on abstraction for dynamic fitness functions. Applications of Evolutionary Computing: EvoWorkshops 2008: EvoCOMNET, EvoFIN, EvoHOT, EvoIASP, EvoMUSART, EvoNUM, EvoSTOC, and EvoTransLog, Naples, Italy, March 26-28, 2008. Proceedings,

Richter, H., & Yang, S. (2009). Learning behavior in abstract memory schemes for dynamic optimization problems. *Soft Computing*, *13*, 1163-1173.

Sadeghi, S., Parvin, H., & Rad, F. (2015). Particle swarm optimization for dynamic environments. 14th Mexican Int. Conf. Artificial Intelligence,(MICAI 2015),

Yang, S. (2006). Associative memory scheme for genetic algorithms in dynamic environments. Applications of Evolutionary Computing: EvoWorkshops 2006: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoINTERACTION, EvoMUSART, and EvoSTOC, Budapest, Hungary, April 10-12, 2006. Proceedings,

Yang, S. (2007a). Explicit memory schemes for evolutionary algorithms in dynamic environments. *Evolutionary computation in dynamic and uncertain environments*, 3-28.

Yang, S. (2007b). Genetic algorithms with elitism-based immigrants for changing optimization problems. Applications of Evolutionary Computing: EvoWorkshops 2007: EvoCoMnet, EvoFIN, EvoIASP, EvoINTERACTION, EvoMUSART, EvoSTOC and EvoTransLog. Proceedings,

استناد به این مقاله: دانش پایه، روح الله و محمدپور، مجید. (۱۴۰۲). بهبود حافظه برای حل مسئله هماهنگ‌سازی کارخانه‌های پویای توزیع‌شده. *فصلنامه پژوهش‌های نوین در شهر هوشمند*، ۱(۴)، ۲۶-۴۴.

New Researches in The Smart City is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.