فصلنامه مدیریت صنعتی دانشکده علوم انسانی، دانشگاه آزاد اسلامی
واحد سنندج - سال دوازدهم/ شماره ۴۲ / زمستان ۱۳۹۶

ارائه راهکاری جدید برای حل مسئله n–وزیر به کمک الگوریتم‌های ژنتیک موازی

منیره طاهری سروتمین (نویسنده مسؤول)
گروه مهندسی کامپیوتر، واحد کرمان، دانشگاه آزاد اسلامی، کرمان، ایران
Email: mtaheri@iauk.ac.ir
عمید خطیبی بردسیری
گروه مهندسی کامپیوتر، واحد کرمان، دانشگاه آزاد اسلامی، کرمان، ایران
Email: a.khatibi@srbiau.ac.ir

تاریخ دریافت: ۹۶/۶/۲۲ * تاریخ پذیرش: ۹۶/۱۱/۸

چکیده

در طول چند دهه گذشته تلاش‌های زیادی برای حل مسائل بهینه‌سازی ترکیبی غیرقطعی انجام شده است. مسئله n–وزیر یکی از همین مسائل است که تاکنون راه‌حل‌های زیادی برای حل این مسئله ارائه شده است. روش‌های سنتی حل این مسئله از نظر زمان اجرا، به صورت نمایی هستند و ازنظر پیچیدگی نمایی و فضایی قابل قبول نیستند. در مطالعه حاضر الگوریتم‌های ژنتیک موازی برای حل مسئله n–وزیر پیشنهاد شده است تا راه‌حل‌های این مسئله را پیدا کند. موازی‌سازی الگوریتم ژنتیک جزیره‌ای و الگوریتم ژنتیک سلولی با استفاده از جعبه‌ابزار محاسبات موازی متلب پیاده‌سازی و روی یک سیستم با پردازنده دو هسته‌ای اجرا شده است. نتایج نشان می‌دهد که این الگوریتم‌ها توانایی پیدا کردن راه‌حل‌های مربوط به این مسئله را دارند. این الگوریتم‌ها حتی بدون استفاده از سخت‌افزار موازی و با اجرا روی یک هستهٔ پردازنده، نه فقط به الگوریتم‌های سریع‌تر بلکه به عملکرد بهتر نیز منجر می‌شوند. مقایسه‌های خوبی بین روش پیشنهادی و نسخه‌های سریال الگوریتم ژنتیک برای سنجش عملکرد روش پیشنهادی انجام شده است. نتایج تجربی نشان می‌دهد این الگوریتم‌ها در مقایسه با الگوریتم ژنتیک سریال برای اندازه‌های بزرگ مسئله کارایی بالایی دارند و در برخی موارد می‌توانند به تسریع فوق‌خطی دست یابند. روش پیشنهادی این مقاله می‌تواند به آسانی برای حل دیگر مسائل بهینه‌سازی توسعه داده شود.

کلمات کلیدی: الگوریتم‌های ژنتیک موازی، الگوریتم ژنتیک جزیره‌ای، الگوریتم ژنتیک سلولی، مسئله n–وزیر.

۱- مقدمه

مسئله n–وزیر[1] یک مسئله بهینه‌سازی ترکیبی معروف است. این مسئله شامل قرار دادن n وزیر روی یک صفحه شطرنج است به طوری که هیچ‌یک از وزیرها قادر نباشند به دیگری حمله کنند. تعداد ترکیب‌های ممکن از n–وزیر در یک صفحه شطرنج بسیار زیاد است. روش‌های سنتی حل این مسئله همگی بر اساس برگشت به عقب[2] بوده است. زمان جستجوی بازگشت به عقب، به صورت نمایی است و قادر نیست مسائل n–وزیر با مقیاس بزرگ را حل کند. اگرچه مسئله n–وزیر کاربرد عملی چندانی ندارد، اما یک کلاس بزرگ از مسائل غیرقطعی را نشان می‌دهد که با استفاده از الگوریتم‌های قطعی نمی‌تواند در زمان قابل قبولی حل شود.

برای حل مسئله n–وزیر الگوریتم‌ها و روش‌های زیادی استفاده شده است که از آن جمله می‌توان به موازد زیر اشاره کرد: (Agarwal, Sinha, & Bindu, 2012; Bashir & Mahdi, 2015; El-Qawasmeh & Al-Noubani, 2004; N. Hu, 2016; X. Hu, Eberhart, & Shi, 2003; Mandziuk, 1995; Mohabbati-Kalejahi, Akbaripour, & Masehian, 2015; Ohta, 2002). در مقاله Ahrabian و همکاران(Ahrabian, Mirzaei, & Nowzari-Dalini, 2008) از الگوریتم DNA Sticker و در مقاله Khan(Khan, Bilal, Sharif, Sajid, & Baig, 2009) به‌طور موفق از الگوریتم کلونی مورچه برای حل مسئله مزبور استفاده شده است. Farhan و همکاران(Farhan, Tareq, & Awad, 2015) از الگوریتم ژنتیک برای حل این مسئله استفاده کردند و تمامی۹۲ راه‌حل ممکن را برای مسئله هشت وزیر پیدا کردند. در سال ۲۰۰۳، Božikovic و همکاران(Božikovic, Golub, & Budin, 2003) از الگوریتم ژنتیک موازی سراسری و روش انتخاب مسابقه[3] سه‌تایی برای حل این مسئله استفاده کردند آن‌ها شبیه‌سازی را روی یک کامپیوتر تک‌پردازنده اجرا کردند و به این نتیجه رسیدند که الگوریتم ژنتیک موازی سراسری برای پردازش‌های موازی حجیم مناسب نیست اما برای تعداد کم واحدهای پردازشی موازی مناسب است.

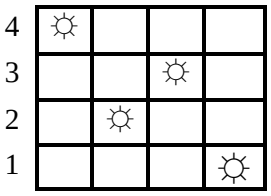
از الگوریتم ژنتیک موازی یا PGA[4] ها به‌طور موفق در محدوده وسیعی از مسائل بهینه‌سازی استفاده شده است(Dash, Dehuri, & Rayaguru, 2013; Mihaylova & Brandisky, 2006; Soufan, Kleftogiannis, Kalnis, & Bajic, 2015; Yu et al., 2011). قدرت خوب این الگوریتم‌ها در مسائل با پیچیدگی بالا منجر به افزایش تعداد کاربرد آنها در رشته‌های هوش مصنوعی، بهینه سازی های عددی و ترکیبی، تجارت، مهندسی و غیره شده است به همین علت در این پژوهش از آن‌ها استفاده می‌شود (Alba & Troya, 1999). این الگوریتم‌ها علاوه بر کاهش زمان محاسبات، منجر به افزایش اکتشافات و تنوع بهتر در مقایسه با الگوریتم‌های ژنتیک ترتیبی می‌شوند(Kacprzyk & Pedrycz, 2015).

مطالعات اخیر تلاش‌هایی در جهت حل مسئله n–وزیر با روش‌های مختلف انجام داده‌اند. هدف اصلی این مقاله حل مسئله n–وزیر با استفاده از الگوریتم‌های ژنتیک موازی و پیدا کردن راه‌حل‌ها می‌باشد همچنین کاهش زمان اجرای این برنامه با الگوریتم‌های ژنتیک موازی نسبت به الگوریتم ژنتیک ترتیبی است که در ادامه این مقاله کارایی این دو الگوریتم نسبت به نسخه سریال نشان داده شده است. موازی‌سازی در این پژوهش با استفاده از جعبه‌ابزار محاسبات موازی[5] متلب روی یک کامپیوتر شخصی انجام شده است. این مقاله به بخش‌های ذیل سازماندهی شده است: بخش ۲ به معرفی نحوهٔ نمایش و کدگذاری مسئله n–وزیر می‌پردازد. در بخش ۳ روش پیشنهادی و تنظیمات مربوط به آن شرح داده شده است. در بخش ۴ طراحی تجربی و بخش ۵ نتایج تجربی آمده است. نتیجه گیری در بخش ۶ ارائه شده است.

---

[1] N-Queen
[2] Back tracking
[3] Tournament
[4] Parallel Genetic Algorithm
[5] Parallel computing toolbox

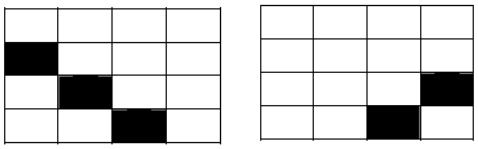
در این بخش از مقاله به تعریف مسئله می‌پردازیم. نمایش مسئله n–وزیر به صورت یک n–تایی است. چون هر وزیر باید روی یک ردیف و ستون متفاوت باشد، می‌توان فرض کرد که وزیر i در ستون i–م قرار داده شده است. راه‌حل‌های n–تایی این مسئله به صورت $(q_1, q_2, \ldots, q_n)$ نمایش داده می‌شوند که جایگشت‌های یک n–تایی $(1,2,3, \ldots, n)$ است. مکان یک عدد در چند–تایی مکان ستون وزیر را بیان می‌کند، درحالی‌که مقدارش، مکان سطر وزیر را نشان می‌دهد (شمارش از سمت پایین). شکل ۱ نمایش ۴–تایی مسئله را نشان می‌دهد.



شکل شماره (۱): مثال ثبت ۴–تایی

تابع برازندگی برای این مسئله باید تعداد برخوردهای وزیرها را بشمارد و بایستی تعداد برخوردهای سطری، ستونی و قطری، محاسبه شوند. در راه‌حل درست این مسئله، این تعداد باید صفر باشد. برای این مسئله دو نمونه تابع برازندگی می‌توان نوشت یکی با پیچیدگی زمانی O(n2) و دیگری با پیچیدگی زمانی O(n). نوشتن تابع برازندگی با پیچیدگی زمانی O(n2) بسیار ساده و مرسوم است؛ اما روش محاسبه برای تابع برازندگی با پیچیدگی زمانی O(n) به صورت زیر است:

۲– برای محاسبه تعداد برخوردهای وزیرها در این مدل، تعداد برخوردها را به صورت قطری بدست می‌آوریم. توضیح این روش این است که چون نمایش n–تایی، برخوردهای سطر و ستون را حذف می‌کند بنابراین فقط حمله‌های قطری بین وزیرها وجود دارد. بر این اساس تابع برازندگی فقط حمله‌های قطری را محاسبه می‌کند. همان‌گونه که در شکل ۲ نشان داده شده است تعداد 2n-1 چپ (بالا–پایین، چپ به راست) و 2n-1 قطر راست (پایین–بالا، راست به چپ) وجود دارد.



شکل شماره (۲): سومین قطر چپ و دومین قطر راست برای ۴–وزیر

روش محاسبه برخوردهای قطری به این صورت است که وزیر i–م و j–م بخشی از یک قطرند اگر:

$$|q_i - q_j| = |i - j| \qquad (۱)$$

رابطه (۱) پیچیدگی O(n) را برای تابع برازندگی نتیجه می‌دهند.

۲- روش شناسی

الف) پیشینه نظری

الگوریتم ژنتیک موازی به‌طور اساسی شامل الگوریتم‌های ژنتیک مختلفی است که هرکدام از آن‌ها یک بخش از جمعیت یا جمعیت‌های مستقلی را با یا بدون ارتباط بین آن‌ها پردازش می‌کنند؛ بنابراین الگوریتم‌های ژنتیک موازی می‌توانند تنوع جمعیت را افزایش داده و زمان محاسباتی را کاهش دهند. این الگوریتم‌ها یک نوع جدید از فراکتشافی‌ها را ارائه می‌دهند که کارایی و تأثیر بالاتری بر روی جمعیت ساختاری و اجرای موازی دارند(Alba & Troya, 1999). همانطور که گفته شد، این الگوریتم‌ها می‌توانند به دو نوع اصلی تقسیم شوند، که نام آن‌ها ، نوع دانه‌درشت[۶] یا الگوریتم‌های ژنتیک جزیره‌ای و نوع ریزدانه[۷] یا

[6] Coarse-grain
[7] Fine-grain

الگوریتم‌های ژنتیک سلولی است. الگوریتم ژنتیک جزیره‌ای، چندین زیرجمعیت را در حالت موازی اجرا می‌کند. زیرجمعیت‌ها باهم جمعیت کل مدل جزیره‌ای را می‌سازند. اکثر مواقع، جزایر به صورت مستقل در مدل اجرا کار می‌کنند. اما به صورت دوره‌ای راه‌حل‌ها بین جزایر مبادله می‌شوند که به این فرایند مهاجرت گفته می‌شود. با استفاده از مهاجرت، مدل جزیره‌ای قادر است تا تفاوت‌ها را در زیر جمعیت‌های مختلف استخراج کند، این تغییر یک منبع تنوع ژنتیکی را نشان می‌دهد.

مدل دیگر، الگوریتم‌های ژنتیک سلولی هستند که برجسته‌ترین خصوصیت آن‌ها این است که هر جزیره فقط یک شخص را در بر دارد. در این مفهوم جزایر، اغلب سلول‌ها خوانده می‌شوند و به هر شخص فقط اجازه داده می‌شود که با اشخاص همسایه ترکیب شود. تفاوت این مدل با مدل جزیره‌ای این است که هیچ تکاملی در خود این سلول‌ها اتفاق نمی‌افتد؛ در واقع هیچ تکامل درون جزیره‌ای وجود ندارد. بهبودها فقط می‌توانند بوسیله سلول‌هایی بدست آیند که با یکدیگر تعامل دارند(Kacprzyk & Pedrycz, 2015).

ب) مدل مفهومی

۱. دستورالعمل انتخاب

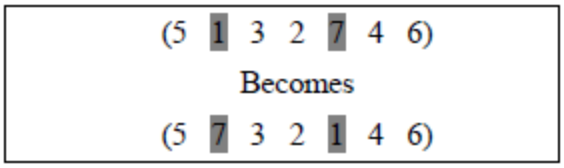
بخش اصلی فرایند انتخاب، انتخاب یک نسل برای ایجاد بنیان نسل بعدی به صورت تصادفی است. مناسب‌ترین افراد شانس بالاتری نسبت به افراد ضعیف‌تر دارند. در اینجا از روش انتخاب چرخ رولت[8] برای انتخاب کروموزوم استفاده شده است.

۲. عملگر تقاطع

عملگر تقاطع با یک نرخ احتمال Pc مشخص می‌شود. در اینجا از روشی استفاده شده است که تلفیقی از روش‌های تقاطع تک‌نقطه‌ای، تقاطع دونقطه‌ای و تقاطع یکنواخت است. روش‌های تقاطع با احتمالات مختلف قرار داده شده‌اند. مقدار احتمال برای تقاطع تک‌نقطه‌ای PsinglePoint=0.1، احتمال تقاطع دونقطه‌ای PdoublePoint=0.2 و احتمال تقاطع یکنواخت Puniform=1-PsinglePoint-PdoublePoint در نظر گرفته شده است. هر بار که تابع تقاطع فراخوانی می‌شود با استفاده از روش چرخ رولت یکی از سه روش بالا برای انجام عمل تقاطع انتخاب می‌شود.

۳. عملگر جهش

همانند عملگر تقاطع، یک عملگر جهش با نرخ P_m برای تعیین اینکه آیا عملگر جهش به کروموزوم اعمال شود یا نه در نظر گرفته شده است. در اینجا از مکانیسم جابجایی[9] یا به عبارتی تغییر جای درایه‌ها استفاده شده است که شکل ۳ نحوه عملکرد آن را نشان می‌دهد.



شکل شماره (۳): عملگر جهش

۴. شرط توقف

شرط توقف رسیدن به تعداد معین از تکرار نسل‌ها در نظر گرفته شده است. برای تعداد مختلف از وزیرها این عدد تغییر می‌کند اما به‌گونه‌ای انتخاب شده است که این اطمینان وجود داشته باشد که الگوریتم همگرا شود و الگوریتم بتواند پاسخ‌های این مسئله را پیدا کند.

ج) طراحی تجربی

۱. استراتژی‌های مربوط به الگوریتم ژنتیک جزیره‌ای

الگوریتم ژنتیک جزیره‌ای چندین زیرجمعیت (جزیره) دارد که در حالت موازی اجرا می‌شوند. تبادل اطلاعات بین این زیرجمعیت‌ها در فواصل زمانی مشخص (دوره‌ای[10]) در تکرار حلقه‌ها انجام می‌شود. با تبادل کروموزوم‌های برجسته[11] بین

[8] Roulette wheel
[9] Swap

زیرمجموعه‌ها، فضای جستجو زیرمجموعه‌ها دارای تنوع می‌شود تا به‌طور مؤثر از همگرایی زودرس جلوگیری کند. n و P_n، به ترتیب، مقدار زیرمجموعه‌ها و مقیاس آن‌ها را بیان می‌کنند، بنابراین جمعیت کل برابر است با $P_{size} = n \times P_n$.

در اینجا از توپولوژی حلقوی برای مهاجرت استفاده شده و جهت آن یک‌طرفه است و با یک فرکانس مهاجرت، تعداد مشخص از بهترین افراد در زیرجمعیت‌ها برای مهاجرت انتخاب می‌شوند و یک کپی از این افراد به همسایه‌ها فرستاده می‌شوند. مهاجران رسیده در جزایر مقصد جایگزین بدترین افراد جامعه می‌شوند.

۲. اندازه‌گیری کارایی

همانطور که Alba (Alba, 2002) اشاره کرده است، مقایسه کارایی الگوریتم‌های تکاملی موازی و سریال فقط در موردی که آن‌ها به دقت یکسانی می‌رسند مفهوم دارد. در یک تقسیم بندی انجام شده توسط Alba (Alba, 2002) برای مقایسه الگوریتم ژنتیک موازی با الگوریتم سریال آن به دو نوع کلی تسریع قوی[۱۲] و تسریع ضعیف[۱۳] وجود دارد. تسریع قوی معمولاً استفاده نمی‌شود. در تسریع ضعیف زمان اجرای موازی یک الگوریتم با زمان اجرای سریال آن مقایسه می‌شود و شامل دو مورد می‌باشد:

تک ماشین/panmixia: الگوریتم موازی با یک نسخه استاندارد، panmictic آن مقایسه می‌شود که روی یک ماشین تک اجرا می‌شود. برای نمونه، ممکن است یک مدل جزیره‌ای را با m جزیره را با یک الگوریتم ژنتیک که یک جزیره را اجرا می‌کند مقایسه کنیم. به موجب آن، الگوریتمی که روی همه جزایر اجرا می‌شود در هر دو مورد یکی است.

رسمی[۱۴]: الگوریتم موازی که روی m ماشین اجرا می‌شود با همان الگوریتم که روی یک ماشین اجرا می‌شود مقایسه می‌شود.

۳. موازی‌سازی با جعبه‌ابزار محاسبات موازی متلب

برای هدف موازی‌سازی، جعبه‌ابزار محاسبات موازی (PCT[۱۵]) متلب انتخاب شده است. جعبه‌ابزار محاسبات موازی این امکان را فراهم می‌کند تا مسائل حجیم و محاسباتی را با استفاده از پردازنده‌های چندهسته‌ای، GPU ها و کلاسترهای کامپیوتری اجرا شود. جعبه‌ابزار اجازه استفاده از همه قدرت پردازشی یک کامپیوتر رومیزی چندهسته‌ای را می‌دهد. استفاده کامل از پردازنده چندهسته‌ای یک کامپیوتر رومیزی، از طریق کارگرهایی امکان پذیر است که به طور محلی اجرا می‌شوند(Sharma & Martin, 2009).

این جعبه‌ابزار یک کلاستر محلی از کارگرها را برای ماشین محلی فراهم می‌کند و برنامه‌های کاربردی را روی کارگرهای محلی (موتورهای محاسباتی متلب) اجرا می‌کند. می‌توان همان برنامه کاربردی را بدون تغییر کد، روی یک کلاستر از کامپیوترها یا یک سرویس محاسباتی گرید، با استفاده از سرویس‌دهنده محاسبه توزیع‌شده متلب اجرا کرد. تعیین تعداد کارگرها در این جعبه‌ابزار به سخت‌افزارها و نحوه اتصال آن‌ها بستگی دارد. در هر صورت برای استفاده از پردازشگر یک سیستم چندهسته‌ای، معمولاً تعداد کارگرها برابر با تعداد هسته‌ها در نظر گرفته می‌شود. در نسخه ۲۰۱۲ متلب، جعبه‌ابزار به کاربران اجازه می‌دهد تا حداکثر ۱۲ کارگر متلب روی یک تک ماشین کار کنند.

۳- نتایج و بحث

الگوریتم‌های ژنتیک موازی با موفقیت روی مسئله n–وزیر اعمال شدند. هر دو الگوریتم قادر بودند راه‌حل‌های مختلفی را برای تعداد وزیرهای داده شده پیدا کنند. نتایج روی یک کامپیوتر با پردازنده Pentium(R) Dual-Core CPU E5700 @ 3.00GHz و سیستم‌عامل ویندوز ۷ و حافظه اصلی 2.00 GB اجرا شد و اندازه مسئله بین ۱۰ وزیر و ۵۰۰ وزیر تنظیم شد. اجرای الگوریتم‌های ژنتیک موازی بر روی پردازنده دو هسته‌ای توانست زمان اجرا را کاهش دهد. برای مقایسه نتایج از هر دو روش رسمی و تک ماشین برای مقایسه کارایی الگوریتم‌ها استفاده شده است.

---

[10] Epoch
[11] Outstanding chromosome
[12] Strong speedup
[13] Weak speedup
[14] Orthodox
[15] Parallel Computing Toolbox

روش مقایسه رسمی:

در این روش مقایسه الگوریتم‌های ژنتیک موازی برای مسئله n–وزیر یک بار به صورت موازی روی دو هسته پردازشگر و بار دیگر به صورت سری و روی یک هسته اجرا شدند و زمان اجرای آن‌ها اندازه‌گیری شد. مقادیر تسریع و کارایی این الگوریتم‌ها نسبت به نسخه سریال خودشان محاسبه شد. جدول‌های ۱ و ۲ زمان اجرا، تسریع و کارایی الگوریتم‌های ژنتیک و جزیره‌ای و سلولی را نسبت به نسخه سریال خودشان برای تعداد مختلف وزیرها نشان می‌دهد.

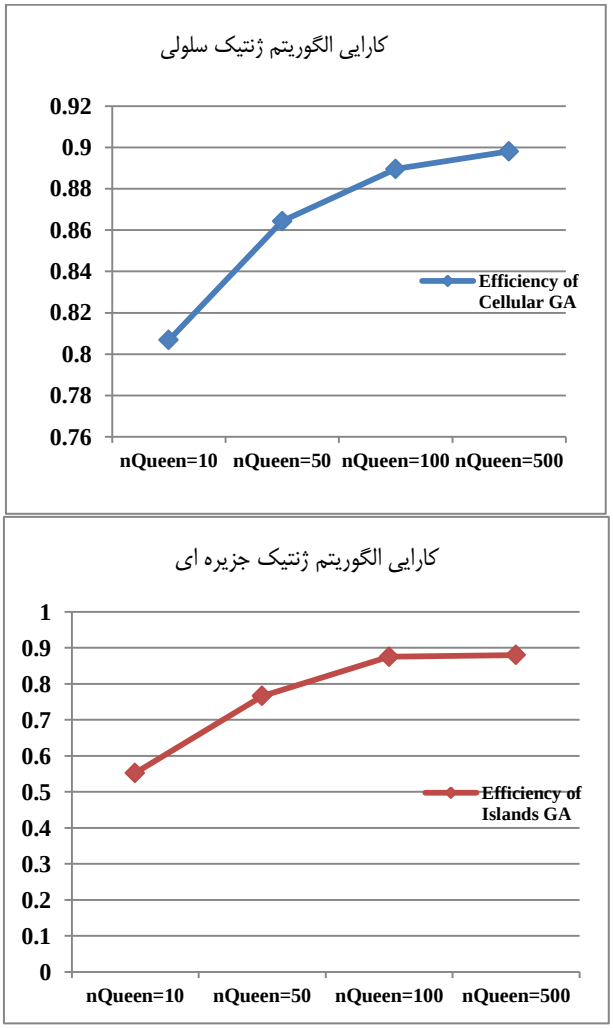
تعداد وزیرها	T_S	T_P	تسریع	کارایی
۱۰	۵/۰۸	۳/۱۵	۱/۶۱	۰/۸۰۶۸
۵۰	۱۰/۸۳	۶/۲	۱/۷۲	۰/۸۶۴۳
۱۰۰	۲۹/۱۷	۱۶/۳۹	۱/۷۷	۰/۸۸۹۵
۵۰۰	۱۸۸/۴۱	۱۰۴/۸۸	۱/۷۹	۰/۸۹۸۱
T_S= زمان اجرای الگوریتم ژنتیک سلولی به صورت سریال				
T_P=زمان اجرای الگوریتم ژنتیک سلولی به صورت موازی				

جدول شماره (۱): زمان اجرا، کارایی و تسریع الگوریتم ژنتیک سلولی در مقایسه با نسخه سریال

تعداد وزیرها	T_S	T_P	تسریع	کارایی
۱۰	۱/۴۸	۱/۳۴	۱/۱	۰/۵۵
۵۰	۳/۲۴	۲/۱۱	۱/۵۳	۰/۷۶
۱۰۰	۱۴/۰	۸/۰	۱/۷۵	۰/۸۷۵
۵۰۰	۱۲۶/۳۳	۷۲/۰۸	۱/۷۵۲	۰/۸۷۹
T_S= زمان اجرای الگوریتم ژنتیک جزیره‌ای به صورت سریال				
T_P=زمان اجرای الگوریتم ژنتیک جزیره‌ای به صورت موازی				

جدول شماره (۲): زمان اجرا، تسریع و کارایی الگوریتم ژنتیک جزیره‌ای در مقایسه با نسخه سریال

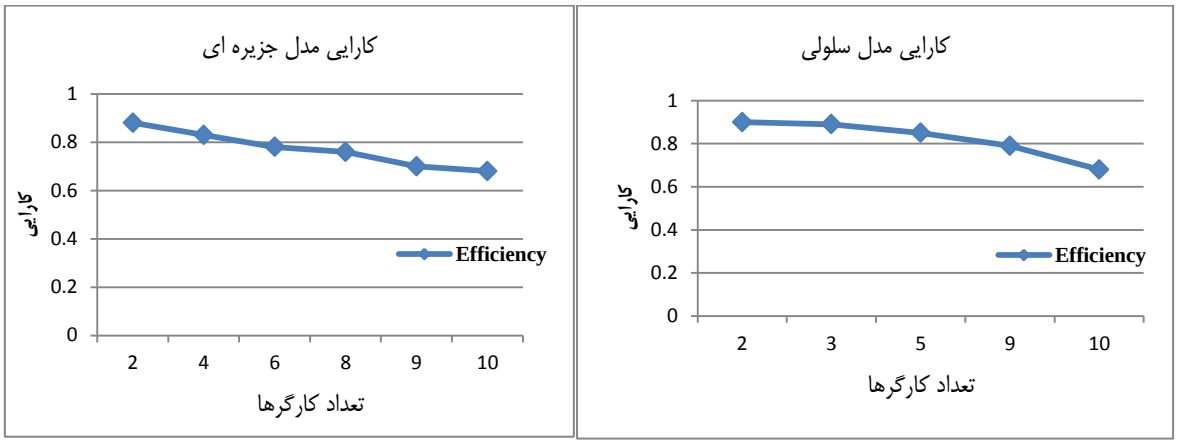
همانطور که در جدول‌های ۱ و ۲ مشخص است هر دو الگوریتم ژنتیک جزیره‌ای و ژنتیک سلولی قادر هستند مسئله n–وزیر را در زمان بسیار کمتری نسبت به نسخه سریال اجرا کنند. ضمناً مقادیر جدول‌ها نشان می‌دهد که کارایی این الگوریتم‌ها با بالا رفتن تعداد وزیرها افزایش پیدا می‌کند و می‌توان به این نتیجه رسید که الگوریتم‌های ژنتیک موازی برای مسائل حجیم و زمان‌بر، عملکرد بهتری از خود نشان می‌دهند. نمودار ۱ کارایی الگوریتم‌های ژنتیک موازی را برای تعداد مختلف وزیرها نمایش می‌دهد.



نمودار شماره (۱): کارایی الگوریتم‌های ژنتیک جزیره‌ای (ب) کارایی الگوریتم ژنتیک سلولی

چندین فاکتور در ناقص بودن تسریع (مقدار تسریع حدوداً ۱٫۷ برای پردازشگر دو هسته‌ای) و درنتیجه مناسب نبودن کارایی دخیل هستند ازجمله انتقال کد و داده بین کلاینت و کارگرها و رقابت منابع بین فرآیندهای کارگر و فرآیندهای سیستم‌عامل. همچنین همانطور که از نمودار ۱ مشخص است با افزایش تعداد وزیرها کارایی این الگوریتم بالاتر رفته است. البته علت کارایی پایین برای تعداد وزیرهای کم را با دلالت به این مسئله که باز و بسته کردن ابزار موازی‌سازی خود عملی زمان‌بر است، می‌توان توجیه کرد؛ بنابراین استفاده از این الگوریتم برای تعداد وزیرهای کم پیشنهاد نمی‌شود اما همانطور که در نمودار ۱ مشاهده می‌شود با افزایش تعداد وزیرها و پیچیده‌تر شدن این مسئله کارایی این الگوریتم بالاتر می‌رود و استفاده از آن مقرون‌به‌صرفه خواهد بود.

جعبه‌ابزار محاسبات موازی متلب زمانی که یک برنامه را به صورت محلی روی یک کامپیوتر اجرا می‌کند به صورت پیش‌فرض تعداد کارگرها را برابر تعداد هسته‌های پردازنده تنظیم می‌کند؛ اما این جعبه‌ابزار این امکان را می‌دهد که تعداد کارگرها بیشتر از تعداد هسته‌های پردازنده تنظیم شود. در آزمایش دیگری تعداد کارگرها افزایش داده شد و زمان اجرا و میزان تسریع الگوریتم‌های موازی برای تعداد ۱۰۰ وزیر اندازه‌گیری شد. نمودار ۲ مقدار کارایی الگوریتم‌های ژنتیک موازی نسبت به نسخه سریال را برای تعداد مختلف کارگران نمایش می‌دهد.



نمودار شماره (۲): کارایی الگوریتم ژنتیک جزیره‌ای با افزایش تعداد کارگرها (ب) کارایی الگوریتم ژنتیک سلولی با افزایش تعداد کارگرها

بر اساس نمودار ۲ افزایش تعداد کارگرهای متلب بدون افزایش تعداد پردازنده‌ها یا تعداد هسته‌های پردازنده، تأثیری در زمان اجرا نخواهد داشت و بهبودی در سرعت ایجاد نخواهد کرد و علت آن را می‌توان سربار ناشی از هم‌روندی دانست. چون تا زمانی که تعداد کارگرها برابر یا کمتر از هسته‌های پردازنده در نظر گرفته شود عملاً برنامه به صورت موازی اجرا می‌شود اما پس از آن زمانی که تعداد کارگرها بیشتر از هسته‌های پردازنده در نظر گرفته می‌شود مفهوم هم‌روندی پیدا می‌کند. نکته قابل توجه در هنگام کار کردن با جعبه‌ابزار موازی متلب این است که مثلاً هنگامی که برنامه برای اجرا روی چهار هسته تنظیم می‌شود، سرعتی که بدست می‌آید نزدیک به ۱–۳ هسته است، زیرا آخرین هسته باید فرآیندهای سیستمی را جرا کند.

بنظر می‌رسد اجرا برنامه روی یک کلاستر از کامپیوترها نتیجه بهتری داشته باشد و به تسریع فوق خطی دست یابد و منابع فیزیکی به عنوان یک دلیل ممکن برای تحقق این امر در نظر گرفته می‌شود. هنگام اجرا روی یک کلاستر ممکن است منابع بیشتری برحسب حافظه یا کش در اختیار برنامه قرار بگیرد و هنگام انتقال کد از یک ماشین تک به یک کلاستر از کامپیوترها، الگوریتم – احتمالاً– از این منابع اضافی استفاده می‌کند. همچنین هر ماشین ممکن است فقط با یک بسته کوچک‌تر از داده سروکار داشته باشد و ممکن است داده‌های کوچک‌تر در کش جا شوند در حالی که در یک ماشین تک این‌گونه نیست و این می‌تواند تفاوت کارایی چشمگیری را ایجاد کند.

روش مقایسه تک ماشین:

در این روش مقایسه برای مسئله n–وزیر، الگوریتم‌های ژنتیک جزیره‌ای و سلولی با جعبه‌ابزار محاسبات موازی روی دو هسته پردازشگر به صورت موازی اجرا شدند و نتایج آن‌ها با الگوریتم ژنتیک استاندارد– که تک جمعیت دارد و تا رسیدن به شرط مشخص اعمال تکاملی را انجام می‌دهد– مقایسه شدند تا عملکرد آن نسبت به الگوریتم ژنتیک استاندارد سنجیده شود. به‌طورکلی الگوریتم ژنتیک استاندارد نسبت به الگوریتم‌های ژنتیک موازی به تعداد تکرار بیشتری برای رسیدن به جواب نیاز دارد و همین عامل باعث افزایش زمان اجرای آن می‌شود. جدول ۳ زمان اجرا، تسریع و کارایی الگوریتم‌های ژنتیک موازی نسبت به الگوریتم ژنتیک استاندارد را نشان می‌دهد.

تعداد وزیرها	T_{GA}	T_{IGA}	T_{CGA}	S_{IGA}	S_{CGA}	E_{IGA}	E_{CGA}
۱۰	۱/۵۵	۱/۴۸	۱/۳۴	۱/۱۵	۰/۴۹	۰/۵۷	۰/۲۴
۵۰	۵/۰	۳/۲۴	۲/۱۱	۲/۳۶	۰/۷۹	۱/۱۸	۰/۳۹
۱۰۰	۲۷/۴۳	۱۴/۰	۸/۰	۳/۴۲	۱/۶۸	۱/۷۱	۰/۸۴
۵۰۰	۲۶۴/۱۳	۱۲۶/۳۳	۷۲/۰۸	۳/۶۶	۲/۵۱	۱/۸۳	۱/۲۵

T_{GA}: زمان اجرای الگوریتم ژنتیک استاندارد (تک جمعیت)

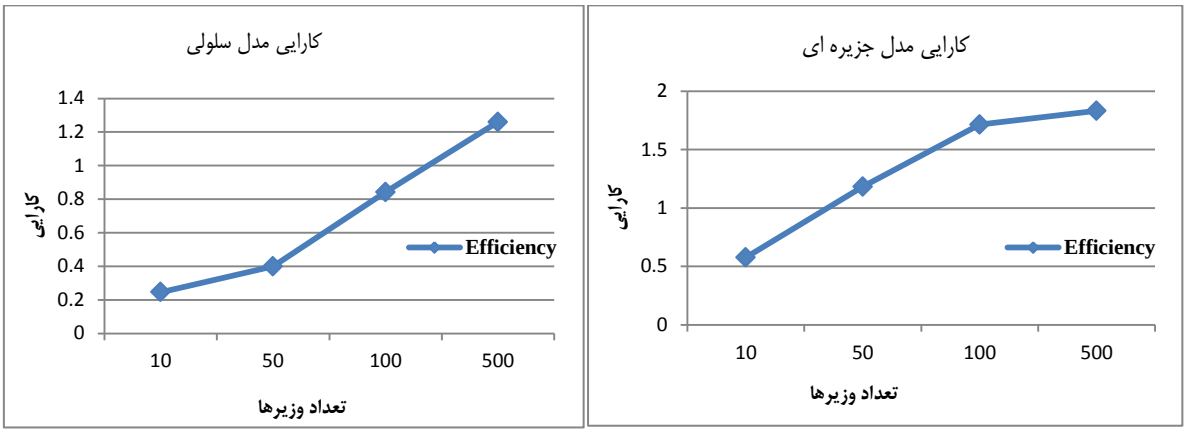
T_{IGA}: زمان اجرای الگوریتم ژنتیک جزیره‌ای — S_{CGA}: تسریع الگوریتم ژنتیک سلولی نسبت به الگوریتم ژنتیک استاندارد

T_{CGA}: زمان اجرای الگوریتم ژنتیک سلولی — E_{IGA}: کارایی الگوریتم ژنتیک جزیره‌ای نسبت به الگوریتم ژنتیک استاندارد

E_{CGA}: کارایی الگوریتم ژنتیک سلولی نسبت به الگوریتم ژنتیک استاندارد S_{IGA}: تسریع الگوریتم ژنتیک جزیره‌ای نسبت به الگوریتم ژنتیک استاندارد

جدول شماره (۳): زمان اجرا، تسریع و کارایی الگوریتم‌های ژنتیک جزیره‌ای و سلولی نسبت به الگوریتم ژنتیک استاندارد

همانطور که از جدول ۳ مشخص است با افزایش تعداد وزیرها تسریع و کارایی الگوریتم‌ها بالاتر می‌رود و بر اساس نتایج تجربی برای بالاترین تعداد وزیر آزمایش شده یعنی ۵۰۰، هر دو الگوریتم به تسریع فوق خطی دست یافتند. نمودار ۳ کارایی الگوریتم‌های ژنتیک جزیره‌ای و سلولی را نسبت به الگوریتم ژنتیک استاندارد نشان می‌دهد.



نمودار شماره (۳): کارایی الگوریتم ژنتیک جزیره‌ای و سلولی نسبت به الگوریتم ژنتیک استاندارد

آزمایش‌های دیگری نیز انجام شد. برای مسئله مذکور الگوریتم ژنتیک جزیره‌ای و سلولی بدون سخت‌افزار موازی و به صورت سری روی یک هسته پردازنده اجرا شد و با الگوریتم ژنتیک استاندارد مقایسه شدند تا عملکرد آن‌ها نسبت به الگوریتم ژنتیک استاندارد سنجیده شود. نتایج تجربی نشان داد که الگوریتم ژنتیک استاندارد معمولاً به تعداد تکرار بیشتری (معمولاً دو یا سه برابر) نسبت به الگوریتم ژنتیک موازی برای رسیدن به جواب نیاز دارد و همین افزایش تعداد تکرارها، زمان اجرای آن را افزایش می‌دهد.

هر دو الگوریتم ژنتیک موازی که بدون موازی‌سازی سخت‌افزاری و به صورت سریال اجرا شدند در تعداد تکرار کمتر و درنتیجه زمان کمتری به جواب رسیدند؛ اما مخصوصاً الگوریتم ژنتیک جزیره‌ای بدلیل عملکرد بهتر درنتیجه زمان بسیار کمتری به جواب می‌رسد. این عملکرد بهتر را می‌توان بعلت انجام عمل مهاجرت بین زیر جمعیت‌ها دانست زیرا مهاجرت به زیرجمعیت‌ها اجازه می‌دهد که مواد ژنتیکی را به اشتراک بگذارند و تنوع را در زیر جمعیت‌ها افزایش می‌دهد.

طبق آزمایش‌های تجربی انجام شده می‌توان گفت استفاده از الگوریتم‌های ژنتیک موازی حتی زمانی که الگوریتم روی یک پردازنده با یک هسته اجرا می‌شود، نه فقط به الگوریتم‌های سریع‌تر، بلکه اغلب به عملکرد عددی بهتر نیز منجر می‌شود. نکته جالب این است استفاده از جمعیت ساختاریافته در قالب جزایر یا در قالب در یک شبکه مسئول چنین منافع عددی است.

الگوریتم	تعداد تکرارها	تعداد وزیرها	جمعیت	زمان	تسریع
الگوریتم ژنتیک استاندارد	۸۰۰۰	۵۰۰	۱۰۰	۱۳/۲۶۴ ثانیه	۱
الگوریتم ژنتیک جزیره‌ای بدون موازی‌سازی سخت‌افزاری (دو جزیره)	۴۰۰۰	۵۰۰	۱۰۰	۱۲۶/۰۶۴ ثانیه	۲.۰۹۵
الگوریتم ژنتیک سلولی بدون موازی‌سازی سخت‌افزاری	۲۰۰۰	۵۰۰	۱۰۰	۱۸۸/۵۵۱۴ ثانیه	۱.۴۰

جدول شماره (۴): تسریع و کارایی روش‌های مختلف نسبت به الگوریتم ژنتیک استاندارد برای مسئله ۵۰۰ وزیر

یکی از مسائل کلاسیک ترکیبی قدیمی، مسئله n–وزیر است. اگرچه مسئله n وزیر کاربرد عملی چندانی ندارد، اما یک کلاس بزرگ از مسائل غیرقطعی را نشان می‌دهد که با استفاده از الگوریتم‌های قطعی نمی‌تواند در زمان قابل قبولی حل شود. این مقاله نشان داد که مسئله n–وزیر می‌تواند به‌طور موفقیت‌آمیز با الگوریتم‌های ژنتیک موازی اجرا شود. نتایج تجربی نشان داد که این الگوریتم‌ها قادر هستند راه‌حل‌های مختلفی برای تعداد مشخص از وزیرها پیدا کند. الگوریتم‌های ژنتیک موازی حتی زمانی که روی یک سیستم تک پردازنده با یک هسته اجرا شوند عملکرد بهتری نسبت به نسخه سریال الگوریتم ژنتیک دارند و در زمان کمتری به جواب می‌رسند. این الگوریتم‌ها باعث تسریع محاسبات و پیدا کردن راه حل‌های بهتر در مقایسه با نسخه سریال آن‌ها می‌شود. این الگوریتم‌ها علاوه بر کاهش زمان محاسبات، منجر به افزایش اکتشافات و تنوع بهتر در مقایسه با الگوریتم‌های ژنتیک ترتیبی می‌شوند همچنین نتایج نشان داد این الگوریتم‌ها برای تعداد بالای وزیرها نسبت به الگوریتم ژنتیک موازی کارایی بهتری دارند و در یک نوع مقایسه می‌توانند به تسریع فوق‌خطی دست یابند. در نتیجه استفاده از این الگوریتم‌ها برای پردازش‌های موازی حجیم مناسب است. اجرای الگوریتم‌های ژنتیک موازی روی یک کلاستر از کامپیوترها به عنوان کار آینده پیشنهاد می‌شود.

٤- منابع

1- Agarwal, K., Sinha, A., & Bindu, M. H. (2012). A novel hybrid approach to N-queen problem, *Advances in Computer Science, Engineering & Applications*, pp. 519-527. Springer

2- Ahrabian, H., Mirzaei, A., & Nowzari-Dalini, A. (2008). A DNA Sticker Algorithm for Solving N-Queen Problem. *IJCSA, 5*(2): 12-22.

3- Alba, E. (2002). Parallel evolutionary algorithms can achieve super-linear performance. *Information Processing Letters, 82*(1): 7-13.

4- Alba, E., & Troya, J. M. (1999). A survey of parallel distributed genetic algorithms. *Complexity*, 4(4): 31-52.

5- Bashir, L. Z., & Mahdi, N. (2015). Use Genetic Algorithm in Optimization Function For Solving Queens Problem. *World Scientific News*, 11, 138-150.

6- Božikovic, M., Golub, M., & Budin, L. (2003). *Solving n-Queen problem using global parallel genetic algorithm.* Paper presented at the International Conference on Computer as a tool EUROCON 2003.

7- Dash, S. R., Dehuri, S., & Rayaguru, S. (2013). *Discovering interesting rules from biological data using parallel genetic algorithm.* Paper presented at the Advance Computing Conference (IACC), 2013 IEEE 3rd International.

8- El-Qawasmeh, E., & Al-Noubani, K. (2004). *A Polynomial Time Algorithm for the N-Queens Problem.* Paper presented at the IASTED International Conference on Neural Networks and Computational Intelligence.

9- Farhan, A. S., Tareq, W. Z., & Awad, F. H. (2015). Solving N Queen Problem using Genetic Algorithm. *International Journal of Computer Applications*, 122(12).

10- Hu, N. (2016). An Integer Coding Based Optimization Model for Queen Problems. *American Journal of Computational Mathematics*, 6(1): 32.

11- Hu, X., Eberhart, R. C., & Shi, Y. (2003). *Swarm intelligence for permutation optimization: a case study of n-queens problem.* Paper presented at the Swarm intelligence symposium, 2003. SIS'03. Proceedings of the 2003 IEEE.

12- Kacprzyk, J., & Pedrycz, W. (2015). *Springer handbook of computational intelligence*: Springer.

13- Khan, S., Bilal, M., Sharif, M., Sajid, M., & Baig, R. (2009). *Solution of n-queen problem using aco.* Paper presented at the Multitopic Conference, 2009. INMIC 2009. IEEE 13th International.

14- Mandziuk, J. (1995). Solving the n-queens problem with a binary Hopfield-type network. Synchronous and asynchronous model. *Biological Cybernetics*, 72(5): 439-446.

15- Mihaylova, P., & Brandisky, K. (2006). *Parallel genetic algorithm optimization of die press.* Paper presented at the Proc. of 3rd International PhD Seminar "Computational Electromagnetics And Technical Applications.
16- Mohabbati-Kalejahi, N., Akbaripour, H., & Masehian, E. (2015). Basic and Hybrid Imperialist Competitive Algorithms for Solving the Non-attacking and Non-dominating n-Queens Problems *Computational Intelligence*, pp. 79-96. Springer.
17- Ohta, M. (2002). Chaotic neural networks with reinforced self-feedbacks and its application to N-Queen problem. *Mathematics and computers in simulation, 59*(4): 305-317.
18- Sharma, G., & Martin, J. (2009). MATLAB®: a language for parallel computing. *International Journal of Parallel Programming, 37*(1): 3-36.
19- Soufan, O., Kleftogiannis, D., Kalnis, P., & Bajic, V. B. (2015). DWFS: a wrapper feature selection tool based on a parallel genetic algorithm. *PloS one,* 10(2), e0117988.
20- Yu, B., Yang, Z., Sun, X., Yao, B., Zeng, Q., & Jeppesen, E. (2011). Parallel genetic algorithm in bus route headway optimization. *Applied Soft Computing, 11*(8): 5081-5091.