تاریخ دریافت مقاله: ۹۷/۱۱/۰۷
تاریخ پذیرش مقاله: ۹۸/۰۵/۱۳

الگوریتم موازی ممتیکی جستجوی ممنوعه برای حل مسئله تخصیص درجه دوم

هادی محمدی*
مربی، گروه مهندسی کامپیوتر، دانشگاه پیام نور
پست الکترونیکی: h.mohammadi@pnu.ac.ir

کمال میرزائی
استادیار گروه مهندسی کامپیوتر، واحد میبد، دانشگاه آزاد اسلامی، میبد، ایران
پست الکترونیکی: k.mirzaie@maybodiau.ac.ir

چکیده:

مسئله تخصیص درجه دو یکی از مسائل بهینه‌سازی ترکیباتی متعلق به ردهٔ مسائل سخت بوده که دارای کاربردی وسیع در جایابی تجهیزات، طراحی صفحه کلید، طراحی تخته مدارهای کنترلی و سایر علوم مهندسی است. در این مقاله به بهبود سرعت و کارائی الگوریتم ژنتیک برای حل این مسئله پرداخته‌می‌شود. بدین منظور الگوریتم ممتیکی جستجوی ممنوعه، مطرح‌شده‌است. جستجوی ممنوعه با ایفای نقش به‌عنوان جستجوی محلی باعث افزایش استخراج در فضای جستجو می‌شود. به همین دلیل از همگرائی زودرس الگوریتم ژنتیک جلوگیری می‌کند. از طرفی به‌منظور جبران محاسبات ناشی از استفاده جستجوی ممنوعه، از واحد پردازش گرافیکی در بستر کودا برای موازی‌سازی پردازش‌ها استفاده‌شده‌است. به‌منظور مقایسه نتایج از مسئله تخصیص درجه دوم با اندازه‌های مختلف استفاده‌می‌شود. نتایج حاکی از افزایش سرعت اجرای پردازش‌ها تا ۱۳ برابر نسبت به الگوریتم سریال ممتیکی جستجوی ممنوعه است. همچنین به دلیل ترکیب الگوریتم ژنتیک با جستجوی ممنوعه کارائی برای یافتن برازش مسئله، بهبود داشته است.

واژه‌های کلیدی: الگوریتم ژنتیک، الگوریتم جستجوی ممنوعه، الگوریتم موازی ممتیکی، واحد پردازش گرافیکی، کودا، مسئله تخصیص درجه دوم، پردازش موازی.

۱- مقدمه

مسئله تخصیص درجه دوم[1]، یکی از مسائل بهینه‌سازی است که دارای پیچیدگی بالا بوده و جزو مسائل NP-Hard می‌باشد. این مسئله توسط کوپمنز و بکمن در سال ۱۹۵۷ به‌عنوان یک مدل رسمی برای تخصیص فعالیت‌های اقتصادی معرفی‌شد [۱]. در این مسئله یک تعداد معین از امکانات (تسهیلات) به همان تعداد مکان، اختصاص‌داده می‌شود که در آن فاصله بین مکان‌ها و وابستگی بین امکانات، با یک عدد بیان‌می‌شود. در تعریف ریاضی این مسئله با فرض این‌که $F=\{f_{ij}\}$ ماتریس هزینه باشد که f_{ij}

1- Quadratic assignment problem

* نویسندهٔ مسئول

میزان وابستگی تسهیل i و تسهیل j را نمایش دهد و $D=\{d_{ij}\}$ ماتریس فاصله باشد که d_{ij} فاصله مکان i و مکان j را نمایش دهد و ماتریس $C=\{c_{ij}\}$ ماتریس تخصیص که c_{ij} هزینه تخصیص تسهیل i در مکان j را نشان دهد. مجموعه اعداد صحیح مثبت $1,2,3,\ldots,n$ در نظر گرفته شود که S_n مجموعه‌ای از جایگشت‌های $1,2,3,\ldots,n$ باشد. هدف مسئله تخصیص درجه دوم پیدا کردن همه جایگشت‌های $\phi \epsilon S_n$ می‌باشد که رابطه (۱) را مینیمم نماید. منظور از نماد ϕ یک جایگشتی از اعداد صحیح مثبت $1,2,3,\ldots,n$ است که خود عضوی از مجموعه‌ای از تمام جایگشت‌ها یعنی S_n می‌باشد. منظور از $\phi(i)$ نیز شماره مکانی است که تسهیل i در آنجا قرار گرفته است. n نشان‌دهنده تعداد مکان‌هاست که تسهیلات در آن جایگذاری می‌شوند. البته در این مقاله همانند اکثر مقالات دیگر مقدار C_{ij} ها برابر صفر در نظر گرفته شده است.

$$\min\left(\sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij}\, d_{\varphi(i)\varphi(j)} + \sum_{i=1}^{n} c_{i\varphi(i)}\right) \qquad (1)$$

به‌عنوان مثال اگر n=4 در نظر گرفته شود (یعنی چهار مکان وجود داشته باشد) دو ماتریس ٤*٤ تسهیل و فاصله که داده‌های آن از قبل در مسئله مشخص گردیده است، تشکیل می‌شود. بنابراین j, iها نیز از مقدار یک تا چهار تغییر پیدا می‌کند. به‌طور مثال در حالت i=۱,j=۲ باید $\boldsymbol{f_{12}}$ و $\boldsymbol{d_{\varphi(1)\varphi(2)}}$ طبق ماتریس‌های فاصله و تسهیل مقداردهی شوند. تاکید می گردد منظور از $\boldsymbol{d_{\varphi(1)\varphi(2)}}$ فاصله بین دو مکانی است که تسهیل ۱و ۲ در آن‌ها قرار دارد. f_{12} نیز از سطر اول و ستون دوم ماتریس تسهیل به‌دست می‌آید. به همین ترتیب، برای تمامی مقادیر i و j این محاسبات انجام می‌شود. حاصل جمع این محاسبات برای i,j های مختلف نشان‌دهنده هزینه یک جایگشت می‌باشد. به همین ترتیب، حال باید رابطه (۱) براساس تمام جایگشت‌های عضو مجموعه S_n یعنی ϕ های مختلف محاسبه شود. در نهایت مینیمم هزینه حاصل از این جایگشت‌های مختلف در رابطه (۱) به‌عنوان هزینه نهایی در نظر گرفته می‌شود. با این کار ترتیب چینش تسهیلات در مکان‌های مختلف که منجر به هزینه تسهیل کمتری می‌شود، به‌دست می‌آید.

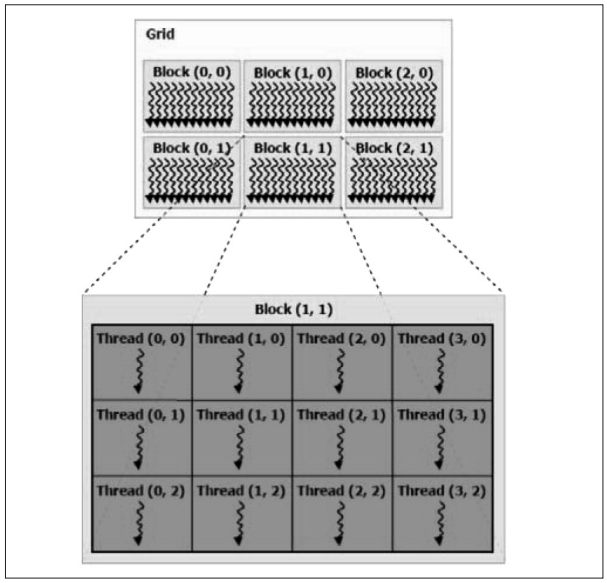
به‌منظور روشن‌تر شدن مسئله مدلی برای ایجاد شهرک دانشگاهی جدید به کمک مسئله تخصیص درجه دوم بیان می‌شود:

مقرر شده یک شهرک دانشگاهی جدید تأسیس گردد که دارای چند ساختمان مختلف می‌باشد. این ساختمان‌ها باید در مکان‌هایی در محوطه دانشگاه ساخته شوند که به‌طور مثال میزان مسافتی که دانشجویان و کارمندان میان این ساختمان‌ها تردد می‌کنند به حداقل برسد. فرض کنید که n مکان و n ساختمان برای تخصیص وجود داشته باشد با فرض این‌که d_{ij} مسافت پیاده‌روی بین دو مکان i و j باشد و f_{ij} تعداد افرادی باشد که هر هفته بین ساختمان‌های i و j رفت و آمد می‌کنند. مسئله، تخصیص ساختمان‌ها به جایگاه‌هاست به‌طوری که مسافت پیاده‌روی افراد مینیمم شود. حاصل ضرب $f_{ij}\, d_{\phi(i)\phi(j)}$ مسافت پیاده‌روی هفتگی افرادی را نشان می‌دهد که بین ساختمان‌های $\phi(j)$ و $\phi(i)$ در تردد بوده‌اند. به‌منظور به‌دست آوردن مینیمم مسافت کل باید رابطه (۲) مینیمم گردد. منظور از مسافت کل، مجموع مسافت پیاده‌روی هفتگی افراد که بین ساختمان‌های $\phi(j)$ و $\phi(i)$ در تردد می‌باشند به‌طوری که iوj از ۱ تا n در حال تغییر می‌باشد.

$$\min\left(\sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij}\, d_{\varphi(i)\varphi(j)}\right) \qquad (2)$$

برای حل این‌گونه مسائل یکی از روش‌ها استفاده از الگوریتم‌های فراابتکاری است. در این مقاله‌این مسئله به کمک الگوریتم ممتیک[2] جستجوی ممنوعه[3] حل‌می‌شود. همچنین به دلیل این‌که بسیاری از مسائل با پیچیدگی زمانی بالا و همچنین ابعاد بزرگ مسئله، با کاهش سرعت مواجه می‌شوند، برای حل این‌گونه مسائل می‌توان با استفاده از پردازش موازی، سرعت آن‌ها را افزایش داد. بسیاری از مسائل قابلیت حل به‌صورت پردازش موازی را دارند از جمله مرتب‌سازی لیست، ضرب ماتریس‌ها، حل

2- Memetic
3- Tabu search



شکل ۱: بلوک و ریسه در واحد پردازش گرافیکی [۸]

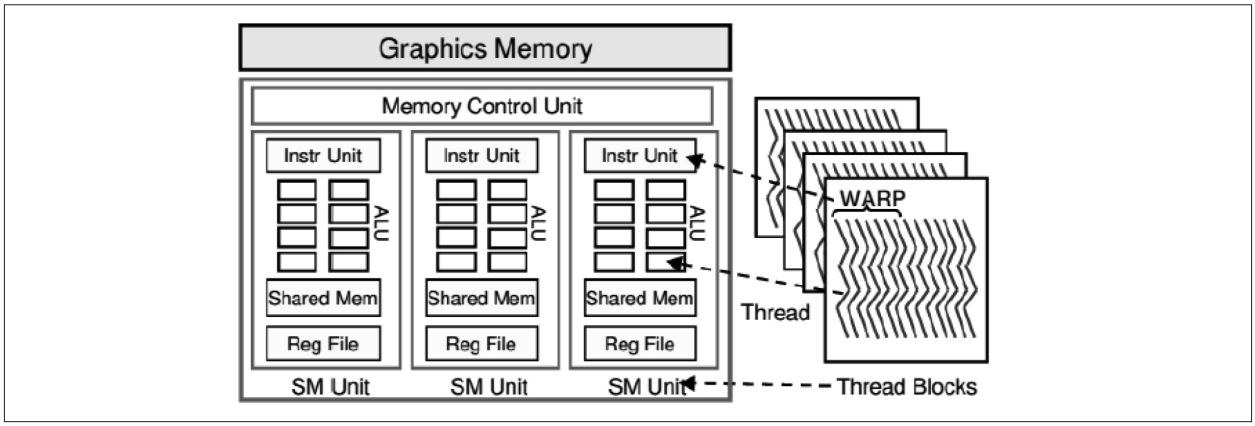
مسئله فروشـــنده دوره گرد و حل مسئله تخصیص درجه دوم. برای حل مســـائل پیچیده می‌تـــوان از الگوریتم‌های بهینه‌ســـازی مبتنی بر جمعیت اســـتفاده نمـــود[۲،۳]. این الگوریتم‌ها قابلیت موازی‌ســـازی خوبی را از خود نشـــان داده‌اند. به‌عنوان نمونه، مسئله تخصیص درجه دوم،که در ابتدای مقدمه این مقاله به آن اشـــاره شـــد، از نوع مسائل پیچیده است. به‌عنوان مثال با تعداد تسهیل N=10 نیازمند بررســـی!۱۰ حالت برای روش دقیق می‌باشـــد چون باید جایگشـــت‌های مختلف با ده تســـهیل محاســـبه شود که حـــدوداً برابر با ۳،۶۰۰،۰۰۰ حالت می‌باشـــد که باید مورد بررسی قرار گیرد. این امر نشان می‌دهد اگر اندازه مسئله بزرگ باشد حجم محاسبات ســـنگینی باید صورت گیرد. به همین علت پیاده‌ســـازی به کمک الگوریتم موازی اهمیت خود را نشـــان می‌دهد. علت این‌که این مســـئله تخصیص درجه دوم را می‌توان موازی پیاده‌ســـازی کرد این اســـت کـــه اگر از الگوریتم ژنتیک به کمک جســـتجوی محلی این مسئله حل شـــود می‌توان در قسمت‌های مختلف الگوریتم ژنتیک مانند عملگر تقاطع، جهـــش و انتخاب جمعیت اولیه از پردازش موازی استفاده نمود. با این شرایط استفاده از الگوریتم موازی به کمـــک پردازنده گرافیکی برای حل این مسئله توجیه دارد[۴،۵]. در ادامه به بررسی ساختار واحد پردازش گرافیکی پرداخته شده است.

واحد پـــردازش گرافیکی سیســـتمی اختصاصی برای نمایش تصاویر گرافیکی در کامپیوترهای شخصی است. با انتشار بسته‌های توسعه نرم‌افزار روی این واحد، از سوی ســـازندگان بزرگی همچون Nvidia و ATI استفاده از واحد پـــردازش گرافیکی به‌عنوان یک واحد محاســـباتی موازی قدرتمند به جای واحد پردازش مرکزی به‌عنوان رویکردی کلیدی در تســـریع پردازش‌های محاسباتی، پذیرفته شده اســـت. دلیل اصلی این تحول بزرگ محاســـباتی آن است که معماری پـــردازش گرافیکی به‌طور ویـــژه برای انجام محاســـبات فشـــرده و عملیات موازی، طراحی‌شده‌است. بدین منظور، شـــرکت Nvidia در ســـال ۲۰۰۶ جهت انجام محاســـبات غیرگرافیکی، بســـتر نرم‌افزاری به‌نام کودا را عرضه نمود. به کمک بســـتر کودا می‌توان سرعت اجرای برنامه‌های غیرگرافیکی را افزایش‌داد[۶،۷].

از دیدگاه برنامه‌نویسی، در بســـتر کودا دو فرایند در محاســـبات وجود دارد: فرایند میزبان[۴] و فرایند دستگاه[۵]. فرایند میزبان، روی واحد پردازش مرکزی اجرامی‌شود و در واقـــع برنامه اصلی را اجرامی‌کنـــد؛ در حالی که فرایند دســـتگاه روی واحد پردازش گرافیکی اجرا می‌شـــود. هر برنامه‌ای که در کودا نوشته‌می‌شود ممکن است از چندین هســـته[۶] تشکیل‌شده‌باشد. هر هسته توســـط یک گرید[۷] که خود از چندین بلوک تشکیل شـــده، اجرامی‌شود. در واقع پردازش‌های ریســـه‌ای[۸] اجرای برنامـــه را بر عهده دارند. شـــکل ۱ ساختار یک هســـته را با شش بلوک که هر بلوک شامل ۱۲ نخ می‌باشد، نشان‌می‌دهد.

در شـــکل ۲ معماری پردازنده گرافیکی و مدل اجرای ریســـه در کودا به تصویر کشـــیده شـــده اســـت. روش مدیریـــت اجرای کودا، تقســـیم گروه‌هایی از ریســـه‌ها در میـــان بلوک‌هاســـت. گریدهـــا از مجموعـــه‌ای از بلوک‌ها

4- Host Process
5- Device Process
6- Core
7- Grid
8- Thread Processing



شکل۲: معماری یک پردازنده گرافیکی و مدل اجرای ریسه در کودا

تشکیل‌شده‌اند. ریسه‌ها به گروه‌های ۳۲ بیتی به‌نام کلاف[9] گروه‌بندی شده‌اند و به هسته‌ها نگاشت‌می‌شوند. همچنین، هر پردازنده گرافیکی دارای یک DRAM می‌باشد که به آن حافظه سراسری گفته‌می‌شود. شش دسته فضای حافظه‌ای ثباتی[10]، حافظه محلی[11]، حافظه مشترک[12]، حافظه داده[13]، حافظه ثابت[14] و حافظه بافت متداول[15] در پردازنده‌های گرافیکی وجود دارد.

همچنین به‌منظور بهبود عملکرد الگوریتم ژنتیک تاکنون از روش‌های مختلف برای حل مشکل همگرائی زودرس الگوریتم ژنتیک، استفاده‌شده‌است که یکی از اساسی‌ترین راه‌حل‌ها برای بهبود این وضعیت استفاده از الگوریتم‌های جستجوی محلی مانند جستجوی ممنوعه، تپه‌نوردی و شبیه‌سازی تبرید[16] است. در این مقاله الگوریتم جستجوی ممنوعه با الگوریتم ژنتیک ترکیب‌شده‌است[۹،۱۰،۱۱].

۲- تحقیقات انجام شده

به‌منظور حل مسائل مختلف پیچیده از جمله تخصیص درجه دوم الگوریتم‌های ممتیکی مختلفی ارائه‌شده‌است که نشان از عملکرد بهتر نسبت به الگوریتم پایه می‌باشد [۱۲]. همچنین در سال‌های اخیر به‌منظور افزایش سرعت اجرای الگوریتم‌های ممتیکی از موازی‌سازی استفاده‌می‌شود. در ادامه به برخی از آن‌ها اشاره‌می‌شود.

در [۱۳]، بر روی الگوریتم پیمایش جستجوی اول سطح به‌صورت پردازنده مرکزی و پردازنده گرافیکی مورد بررسی قرار گرفته است. کشاورزی و همکارانش به تشریح معماری پردازنده گرافیکی پرداخته‌اند.

در سال ۱۳۹۵ میلاد رفیعی و همکارانش، دسته‌بندی بسته درخت سلسله‌مراتبی را بر روی پردازنده گرافیکی به‌صورت موازی، پیاده‌سازی نموده اند[۱٤]. پردازنده‌های ویژه‌ای که برای محیط‌های شبکه مانند مسیریاب استفاده می‌شود را پردازنده شبکه‌ای گویند. همچنین، می‌توان بسته‌های مختلف موجود در تجهیزات شبکه مانند مسیریاب و سوئیچ را به جریان‌های مختلف طبقه‌بندی کرد. این فرآیند را دسته‌بندی بسته‌ها گویند. در پردازنده‌های شبکه‌ای، دسته‌بندی بسته‌ها، پردازشی اساسی محسوب می‌شود. پیاده‌سازی الگوریتم‌های دسته‌بندی به‌صورت نرم‌افزاری با وجود هزینه کم‌تر و توسعه‌پذیری بیشتر نسبت به پیاده‌سازی‌های سخت‌افزاری، سرعت پایین‌تری دارد. به همین منظور رفیعی و همکارانش از واحد پردازش گرافیکی برای موازی‌سازی این دسته‌ها، استفاده کرده‌اند. همچنین در آن سناریوهای مختلفی براساس معماری حافظه‌های سراسری و اشتراکی مورد بررسی قرار گرفته است. نتایج آزمایش‌ها حاکی از عملکرد بهتر سناریویی است که بتواند درخت سلسله مراتبی و مجموعه پالایه‌های متناظر را بدون

9- Warp
10-Registers
11- Local Memory
12- Shared memory
13- Data Memory
14- Constant Memory
15- Texture Memory
16- Simulated Annealing

افراز در حافظه اشتراکی در خود جای دهد.

در این مقالـــه برای مدیریت بهتر زمـــان اجرا هنگامی که تعداد گره‌های ســـطح جـــاری کمتـــر از ۵۱۲ (کمتر از تعداد ریســـه‌های یک بلوک در پردازنده گرافیکی) باشد از پردازنده مرکزی استفاده‌می‌شـــود. در غیر این صورت از پردازنده گرافیکی برای پردازش الگوریتم سطح اول مورد استفاده قرارمی‌گیرد.

در مقالـــه‌ای دیگر در ســـال ۱۳۹۶ که بـــرای الگوریتم بهینه‌سازی غذایابی باکتری که برای مسائل جستجو مورد استفاده قرار می‌گیرد، به‌طور موازی در پردازنده گرافیکی پیاده‌سازی شـــده است[۱۵]. الگوریتم بهینه‌سازی غذایابی باکتری در ســـال ۲۰۰۱ توسط پاســـینو ابداع شده است. برای کاربردهای متعددی از جمله طراحی کنترل کننده‌های PID، کنترل تطبیقی، تخمین هارمونی و آموزش‌های عصبی به‌کار برده شـــده است. به‌منظور بهینه‌ســـازی مسائل به کمک الگوریتـــم غذایابی باکتری بعد از تولید جمعیت اولیه به ترتیب ســـه عملیات کموتکســـیز، تولید مثل و حذف – پراکندگـــی بر روی باکتری‌های مصنوعی انجام‌می‌شـــود. باکتری‌ها برای رســـاندن خود به محیط با بیشترین تراکم مواد غذائی شنای رو به جلو یا تصادفی انجام می‌دهند که در علم زیست‌شناســـی اصطلاحاً به این حرکت باکتری‌ها کموتکســـیز گویند. منظور از تولید مثل نیز تقسیم باکتری از وسط به دو باکتری می‌باشـــد. هر باکتری پس از مدت معینی می‌میرد و از چرخه حیات خارج می‌شـــود. به جای این باکتریی که از چرخه حیات حذف شده است یک باکتری به‌طور تصادفی انتخاب می‌شود و در نقطه‌ای از دامنه مسئله قرار داده می‌شـــود. این عمل را حذف– پراکندگی می‌نامند. یکی از اساسی‌ترین مشـــکلات الگوریتم غذایابی باکتری، ســـرعت پایین همگرائی آن است. در این مقاله با استفاده از حافظه اشتراکی در GPU نیاز به استفاده از گذرگاه داده سیســـتم از بین می‌رود. همچنین در حین اجرای الگوریتم نیاز به تولید اعداد تصادفی می‌باشد. برای افزایش سرعت از قابلیت CURAND در معماری کودا اســـتفاده شده است.

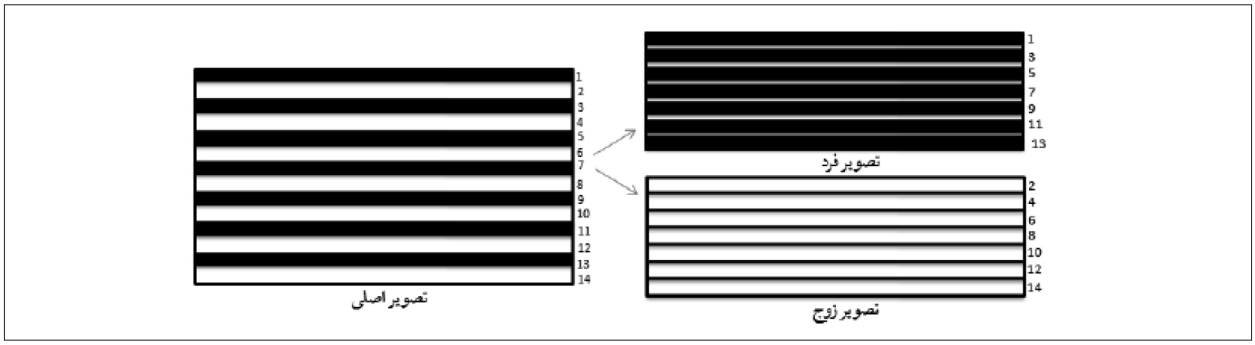
ســـپس، به‌منظور ارزیابی نتایج از توابع محک[۱۷]، استفاده شـــده است. لازم به توضیح اســـت که برای ارزیابی نتایج الگوریتم‌های مبتنی بر جمعیت می‌توان از توابع استاندارد به‌نام محک استفاده نمود.

فاطمه ســـیر و ســـعید مظفـــری در ســـال ۱۳۹۶ برای بالا بردن ســـرعت اجرا در الگوریتـــم حذف درز از تکنیک مرتب‌ســـازی زوج و فـــرد به‌صـــورت موازی اســـتفاده نموده‌اند [۱۶]. گفتنی اســـت، یکی از روش‌های تغییر ابعاد مبتنی بر محتوا الگوریتم حذف درز می‌باشـــد که در سال ۲۰۰۷ معرفی شده اســـت. فاطمه سیر و مظفری به‌منظور موازی‌سازی حذف درز از تجزیه تصاویر به زیر تصاویر استفاده کرده‌اند که آن را با تصاویر زوج و فرد نامگذاری کرده‌اند. همان‌طور که در شـــکل۳ نشـــان داده شده است، تصویر اصلی به دو زیرتصویر زوج و فرد تجزیه شده و عمل جستجو به‌طور جداگانه روی هر یک از آن‌ها صورت می‌گیرد. با این تکنیک ســـرعت تا دو برابر روش معمولی افزایش داده شده اســـت. در روش زوج و فرد، جستجوی درز به دو پردازش موازی تقسیم می‌شود. تصویر زوج از ســـطرهای زوج تصویر اصلی و تصویر فرد از سطرهای فرد تصویر اصلی به دست می‌آیند. به‌منظور موازی‌سازی الگوریتم روش پیشنهادی ارائه شده در این مقاله می‌تواند به کمک تصاویر زوج و فرد در یک گام همزمان دو درز را از تصویر حذف نماید.

در مقاله [۱۷] به بررســـی الگوریتم‌های مختلف تکاملی ماننـــد ژنتیک، شبیه‌ســـازی تبرید و بهینه‌ســـازی ازدحام ذرات[۱۸] پرداخته‌شده‌اســـت. در آن الگوریتـــم SSPCO[۱۹] که در ســـال ۲۰۱۵ بـــا الهام از رفتار جوجه‌هـــای پرنده تیهو ابداع‌شده‌است، معرفی و توسط نظریه آشوب، بهبود داده شده است.

در جـــدول ۱ خلاصه ای از مهمتریـــن تحقیقات انجام شـــده در زمینه مسئله تخصیص درجه دوم، الگوریتم‌های فرابتکاری و روش‌های ممتیکی موازی ارائه شده است.

17- Bench Functions
18- Particle Swarm Optimization
19- See See Partridge Chicks Optimization



شکل ۳: نحوه تشکیل تصویر زوج و فرد از کنار هم قرارگیری سطرهای تصویر اصلی

جدول ۱: مهمترین تحقیقات انجام شده پیشین

موضوع	توضیحات
حل مسئله تخصیص درجه دوم به روش الگوریتم ممتیکی	حل مسئله تخصیص درجه دوم به کمک الگوریتم ممتیکی انجام شده است. که نسبت به روش الگوریتم پایه عملکرد بهتری از خود ارائه داده است
موازی‌سازی الگوریتم بهینه‌سازی غذایابی باکتری	این الگوریتم برای جستجو مورد استفاده قرار می‌گیرد. در این مقاله به‌منظور افزایش سرعت الگوریتم به کمک پردازنده گرافیکی در کودا موازی شده است.

۳- الگوریتم پیشنهادی

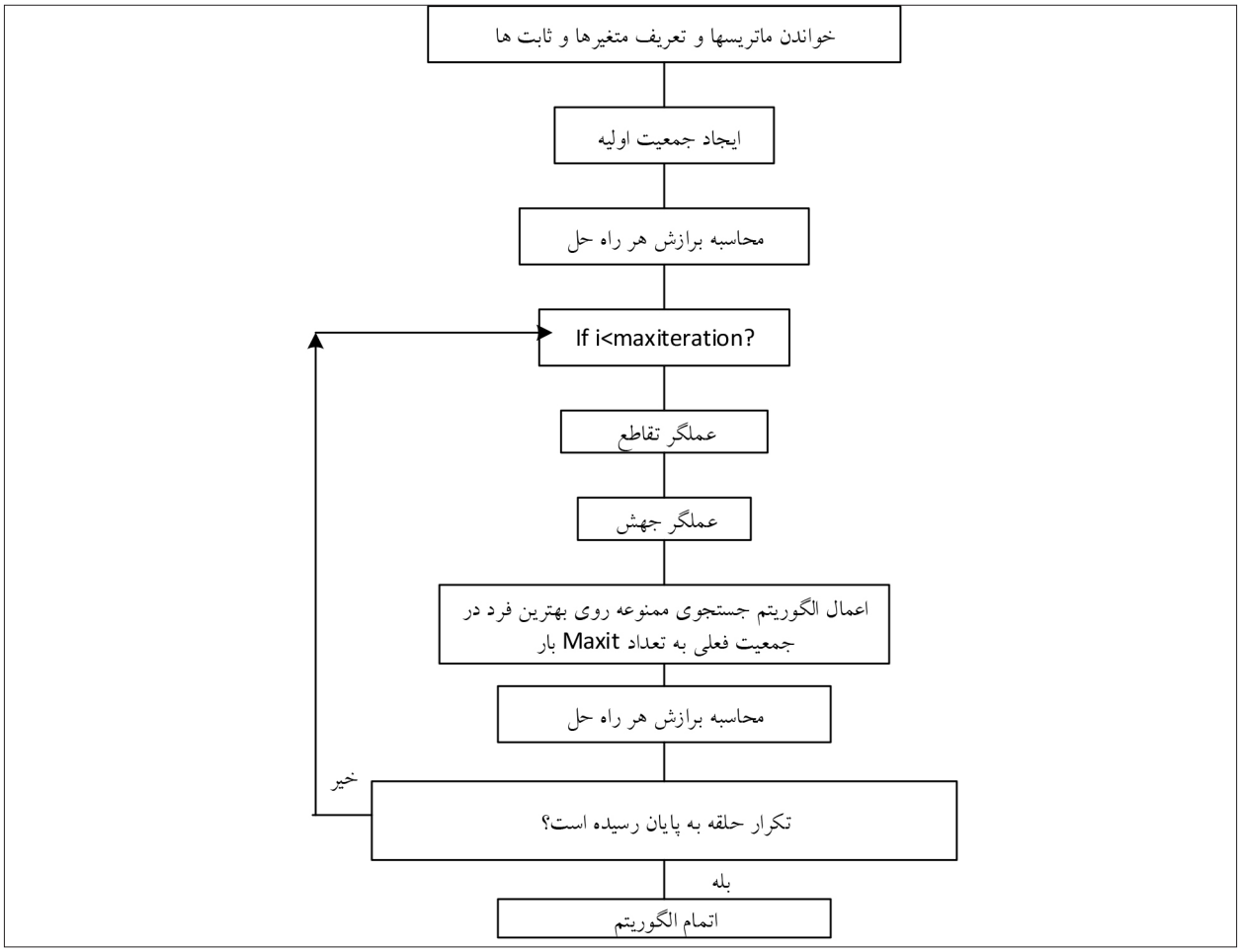
برای حل مسئله تخصیص درجه دوم الگوریتم ژنتیک با الگوریتم جستجوی محلی جستجوی ممنوعه ترکیب شده است. سپس به‌منظور تسریع در عملکرد الگوریتم ترکیبی به کمک بستر کودا و Microsoft Visual studio C++ و واحد پردازش گرافیکی Nvidia Geforce 920M مسئله تخصیص درجه دوم به‌صورت موازی حل شده است. در واقع یک الگوریتم موازی ممتیکی جستجوی ممنوعه ارائه می‌شود. در این بخش از مقاله، به روندنمای الگوریتم پیشنهادی سریال (شکل ٤) مربوط به ترکیب الگوریتم ژنتیک و جستجوی ممنوعه، به همراه تشریح هر کدام از الگوریتم‌ها، پرداخته‌شده‌است.

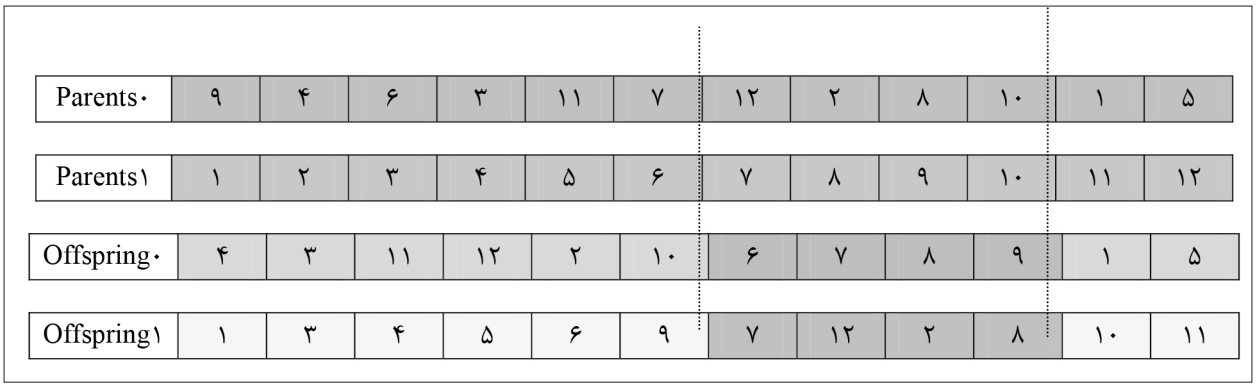
۳-۱ الگوریتم ژنتیک:

در مسئله تخصیص درجه دوم هر راه‌حل جایگشتی از تسهیل‌های مختلف در کل مکان‌های موجود می‌باشد. برای حل این مسئله به کمک الگوریتم ژنتیک، کروموزوم‌ها همان جایگشتی از تسهیل‌ها می‌باشند که به‌عنوان راه‌حل مسئله اطلاق می‌شوند و ژن‌ها نیز همان تسهیل‌ها می‌باشند. در هر مرحله از الگوریتم ژنتیک هدف به حداقل رساندن هزینه این راه‌حل‌ها می‌باشد. برای حل این مسئله به کمک الگوریتم ژنتیک مراحل زیر انجام می‌شود: در ابتدا یک جمعیت اولیه تولید می‌شود سپس برازش این جمعیت اولیه محاسبه می‌شود. در ادامه به‌صورت تکراری به تعداد MaxIteration مراحل عملگرهای تقاطع و جهش روی جمعیت اولیه اعمال‌می‌شود[۱۸]. همان‌طور که در شکل۵ نشان داده شده است در الگوریتم پیشنهادی از عملگر تقاطع دو نقطه‌ای، استفاده‌شده‌است. در ابتدا راه‌حل موجود در دو والد انتخابی عیناً به دو فرزند کپی می‌شود. در این نمونه نقطه اول و دوم به ترتیب برابر ٦ و ١٠ در نظر گرفته شده‌اند. پس از آن، در فرزند دوم تمام عناصری که برابر با مقادیر موجود در محدوده ٦ تا ١٠ والد اول می‌باشد با صفر جایگزین می‌شوند تا از تکراری بودن عناصر جلوگیری شود. سپس همه عناصر صفر قبل و بعد از دو نقطه ۱و ۲ به کمک عملیات شیفت در محدوده دو نقطه ۱و۲ قرارمی‌گیرند. پس از آن، عناصر بین دو نقطه والد یک عیناً در فرزند دو کپی‌می‌شود. همین عملیات برای والد ۲ و فرزند یک اعمال‌می‌گردد.

در ادامه عملگر جهش بر روی خروجی عملگر تقاطع

شکل۴ : روندنمای الگوریتم پیشنهادی ممتیکی جستجوی ممنوعه سریال



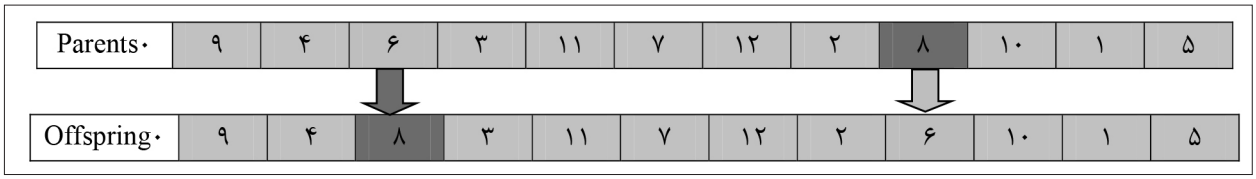


شکل ۵: عملگر تقاطع دو نقطه‌ای

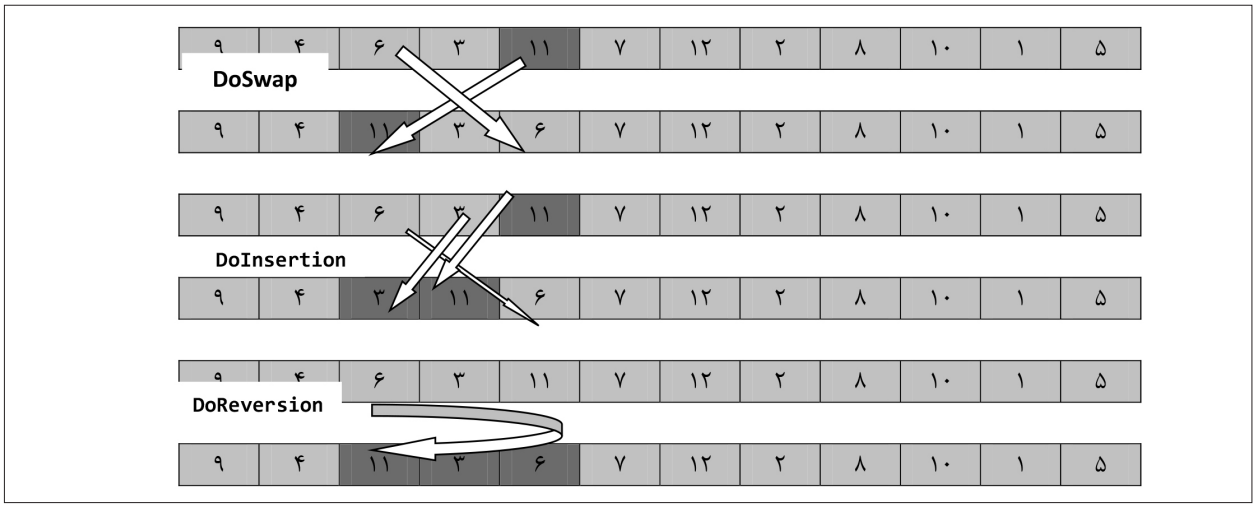
اعمال می‌شـــود تا یک تغییرات ناگهانـــی روی راه‌حل ایجاد نماید. همان‌طور که در شکل ٦ نشان داده شده است در عملگر جهش دو یا چند خانه از راه‌حل، با یکدیگر تعویض‌می‌شود.

۳-۲ الگوریتم جستجوی ممنوعه:

به‌منظور بهبود الگوریتم ژنتیک و حل مشکل همگرائی زودرس و عملکرد نسبتاً پایین الگوریتم ژنتیک برای استخراج راه‌حل‌ها، از جستجوی محلی استفاده‌می‌شود[۱۹,۲۰,۲۱]. در این مقاله بدین منظور از جستجوی ممنوعه استفاده شده اســـت. روش عملکرد آن به‌صورت زیر می‌باشـــد: بهترین جواب از آخرین مرحله الگوریتم ژنتیک به‌عنوان یک راه‌حل



شکل۶: عملگر جهش برای یک راه‌حل



شکل ۷: عملیات جابه‌جائی، درج و معکوس روی راه‌حل مسئله

به الگوریتم داده می‌شـــود. در الگوریتم جستجوی ممنوعه سه نوع عملیات جابه‌جائی، درج و معکوس برای راه‌حل‌ها اعمال‌می‌شـــود که با این عملیات بتواند یک تغییراتی را در راه‌حل مورد نظر ایجاد نمایند. این سه نوع عمل، در شکل۷ نشان داده شده اند. منظور از عمل جابه جائی این است که دو ژن از کروموزوم به‌طور تصادفی انتخاب شده و جای آن‌ها با یکدیگر تعویض می‌گردد. منظور از عملیات درج این است که دو ژن از کروموزوم به‌طور تصادفی انتخاب شده و ژن انتخابـــی اول در جلـــوی ژن انتخابی دوم جایگذاری می‌شود. همچنین ژن‌ها مابین این دو ژن نیز یکی به سمت چپ انتقال پیدا می‌کند. اگر شماره آرایه ژن اول بزرگ‌تر از شماره آرایه ژن دوم باشد به ژن‌های مابین دو ژن انتخابی به جای انتقال به چپ، یک خانه به سمت راست انتقال پیدا می‌کننـــد. همچنیـــن، منظور از عمل معکوس این اســـت که دو ژن از کروموزوم مورد نظـــر به‌طور تصادفی انتخاب می‌شـــود. این دو ژن به همراه تمام ژن‌های مابین آن‌ها به ترتیب معکوس در کروموزوم مرتب می‌شوند. نکته‌ای که برای اعمال این عملیات در این مقاله درنظرگرفته شده است از تکرار عملیات مشـــابه که خروجی یکســـان دارند برای جلوگیری از ســـربار و عملیات اضافی، گرفته شده است. همان‌طور که در شـــکل ٤ بیان شده است، به‌منظور یافتن جـــواب بهتر حول یک راه‌حل عملیات جســـتجوی ممنوعه به تعداد Maxit بار تکرار شـــده است. با این کار جواب با برازش مناسب‌تری انتخاب می‌شود.

تعداد عملیات حاصل از سه روش مختلف از فرمول (۳) محاسبه شده است:

$$nAction = nSwap + nReversion + nInsertion \quad (۳)$$

در الگوریتـــم جســـتجوی ممنوعـــه از دو پارامتر مهم به‌نام‌های TL و TC اســـتفاده شده است. TL یک مقدار ثابت در طـــول اجرای الگوریتم دارد و TC یک لیســـتی به طول nAction می‌باشـــد. TL نشـــان‌دهنده تعداد عملیاتی که در لیست ممنوعه قرارمی‌گیرند. در واقع، این مقدار نشان‌دهنده تعـــداد راه‌حل‌هایی که در حین اجرای برنامه اســـتفاده از آن‌ها ممنوع می‌باشـــد است. با این کار بخشی از راه‌حل‌ها (به تعداد TL) در لیســـت ممنوعه قـــرار می‌گیرند. به همین دلیل است که این روش به‌نام جستجوی ممنوعه نامگذاری

شده اســت. در این مقاله TL برابر یک دهم مقدار nAction در نظر گرفته شــده اســت. TC هم می‌تواند مقادیر ۰ تا TL را در خود جای دهد. TC نشان‌دهنده ممنوع بودن یا نبودن یک عمل می‌باشــد. اگر برابر صفر باشد ممنوع نمی‌باشد در غیر این صورت ممنوع می‌باشــند و تا زمان خروج از لیســت ممنوعه، نمی‌تواند به‌عنوان یک عمــل برای اعمال تغییرات در راه‌حل استفاده‌شود. هر عملی که انجام‌می‌شود در انتها مقدار پارامتر TC آن برابر TL (بیشترین مقدار TC) قرارداده‌می‌شــود. بقیه عناصر هم یک مقــدار از آن‌ها کم می‌شود تا به صفر برسند.

در این مقاله عملیات جستجوی ممنوعه برای یک راه‌حل maxIt بار تکرارمی‌شود. در پایان بهترین راه‌حل به‌عنوان خروجی به انتهای خروجی به انتهای حلقه الگوریتم ژنتیک، داده می‌شــود. خود عملیات ژنتیک این مراحل را به اندازه maxIteration بــار تکرار می‌کنــد. در نهایت در خروجی بهترین راه‌حل، به همراه برازش آن چاپ می‌شود. لازم به توضیح است که تابع برازش راه‌حل‌ها براساس فرمول (۱) یعنی هزینه جایگشــت براساس شماره تسهیل‌هایی که در مکان‌های موجود جایگذاری می‌شوند، محاسبه می‌شوند.

۳-۳ الگوریتم موازی ممتیکی جستجوی ممنوعه

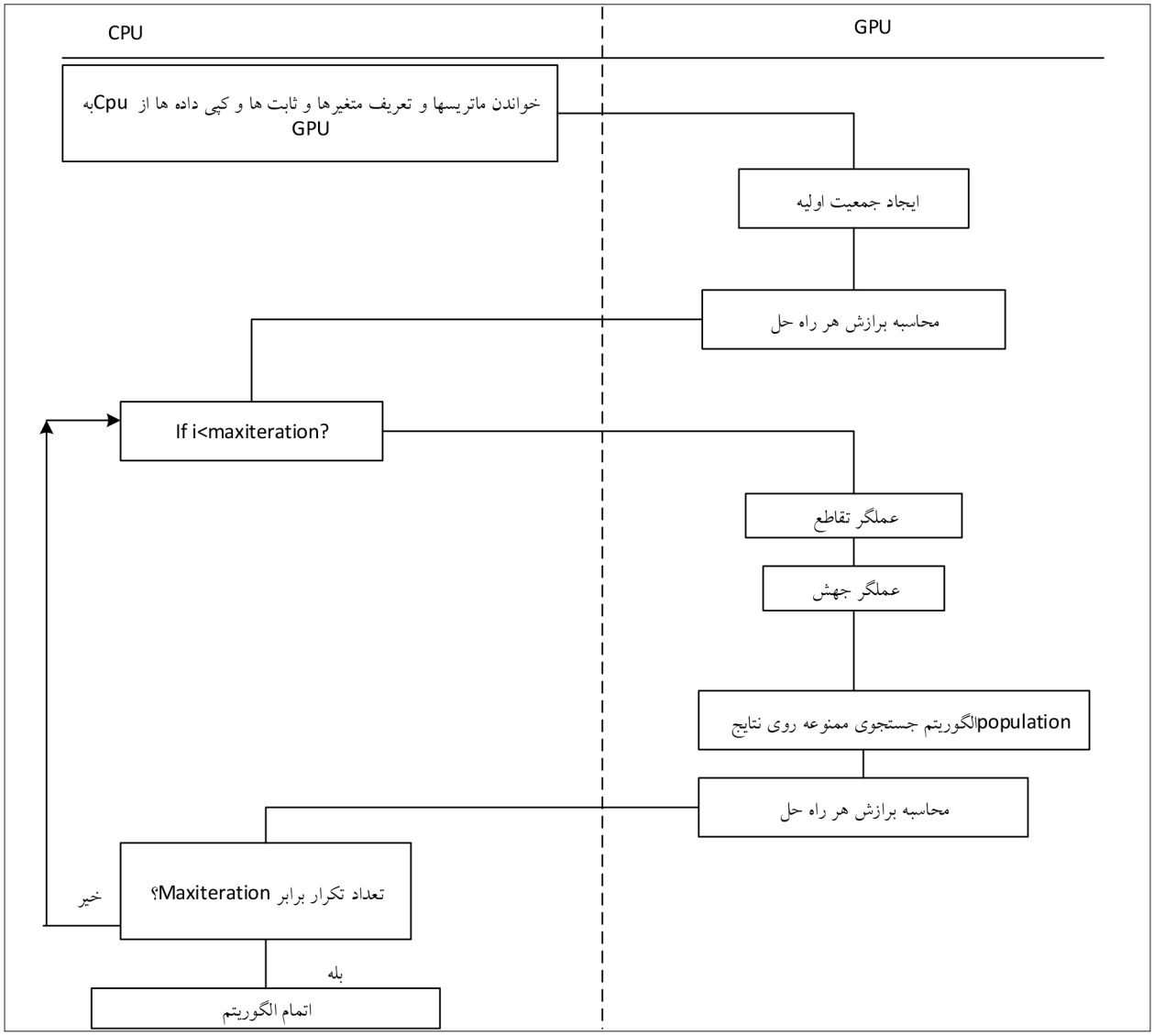
به‌منظور تســریع در عملیــات و کاهش زمان اجرای الگوریتم ممتیکی جســتجوی ممنوعــه کلیه عملیات مطرح شده در قسمت قبل به‌صورت موازی در بستر کودا مورد بررسی قرار گرفته است. روندنمای این عملیات در شکل ۸ آورده شــده است. در الگوریتم ژنتیک مهم‌ترین بخش‌های آن که لازم به موازی‌ســازی است عملیات تقاطع و جهش می‌باشد. به‌منظور موازی‌سازی عمل تقاطع نیاز به تشکیل نخ از طریق رابطه (٤) می‌باشد:

i = blockIdx.x*blockDim.x + threadIdx.x. (٤)

blockIdx.x، blockDim.x و threadIdx.x بــه ترتیــب نشان‌دهنده شماره گرید، شماره بلوک و شماره ریسه در یک هســته در محیط کودا می‌باشد. در واقع i نشان‌دهنده شماره ریسه‌های مختلف موجود در یک هسته می‌باشد که تعداد آن بستگی به مقادیر گرید، بلوک و ریسه موجود در یک هسته دارد. به‌منظور کنترل تعداد ریسه‌های تخصیصی برای این عمــل، باید مقدار i با Ncross کــه تعداد عملیات تقاطع در هر تکرار را نشان می‌دهد، مقایسه شود که بیشتر از Ncross نباشــد. در این مرحله i ریسه برای انجام تمام عملیات تقاطع که به تعداد Ncorss می‌باشد آماده است. در واقع، برای هر عمل تقاطع یک پردازنده در نظر گرفته شده اســت. با این کار تمام عملیات تقاطع به‌طور موازی انجام می‌شــود. بر خلاف روش سریال دیگر نیاز به وجود حلقه تکرار که هر بار یک عمل تقاطع را انجام دهد نمی‌باشد.

به‌منظــور موازی‌ســازی عملیات جهش بــه اندازه i تا ریســه همانند روش تقاطع به‌کارگیری می‌شــود که به کمک آن بتوان عملیات موازی‌ســازی انجام شود. در ابتدا بررسی می‌شــود که این i ها بیشتر از مقدار Nmut یعنی تعداد عملیات جهش در هر تکرار، نباشــد. در این صورت هر پردازنده عهده‌دار یک عمل جهش که در قسمت سریال توضیح داده شــده است، می‌باشــد. با این کار کل عملیات جهــش مورد نیــاز در این تکــرار یعنی به تعــداد Nmut، به‌صورت موازی انجام می‌گیرد.

به‌منظور موازی‌سازی الگوریتم جستجوی ممنوعه نیز همان مفاهیم موجود در الگوریتم جستجوی ممنوعه سریال همچنان پا برجا می‌باشد. با این تفاوت که همانند رابطه (۳) i تا ریسه برای nAction تا عمل که از فرمول (۲) محاسبه می‌شــود در نظر گرفته می‌شــود. همچنین j تا ریسخ نیز که برابر با مقدار Maxit یعنی تعداد تکرار عمل جستجوی ممنوعه می‌باشــد در کودا برای موازی‌سازی اختصاص داده می‌شــود. بقیه مراحل الگوریتم جســتجوی ممنوعه همانند سریال انجام می‌شــود. در واقع، تعداد کل عملیات جســتجوی ممنوعه در هر تکرار برابر با nAction*Maxit می‌باشــد که با ایــن کار برای تک تک این تعــداد عملیات جســتجوی ممنوعه یک پردازنده، تخصیص داده می‌شود. با این کار برای هر عمل یک پردازنده در نظر گرفته شــده است.



شکل ۸: روندنمای پیشنهادی ممتیکی جستجوی ممنوعه موازی

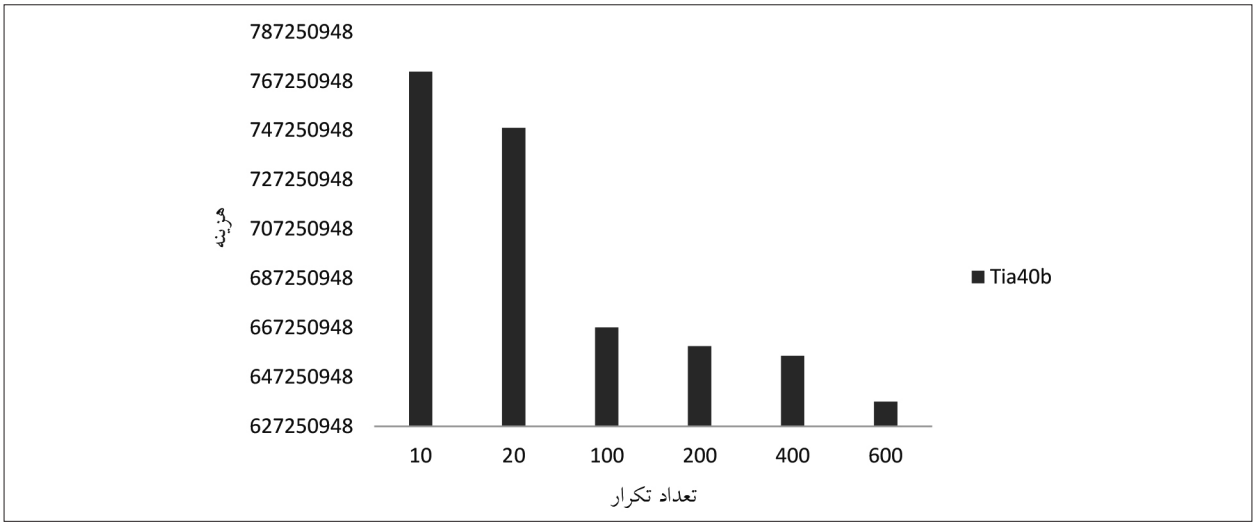
همان‌طور که شکل۸ نشان می‌دهد، بسیاری از عملیات اصلی در داخل GPU صورت‌می‌گیرد. به همین دلیل سرعت پردازش اطلاعات بسیار بالاتر از حالت سریال است.

۴- ارزیابی و نتایج

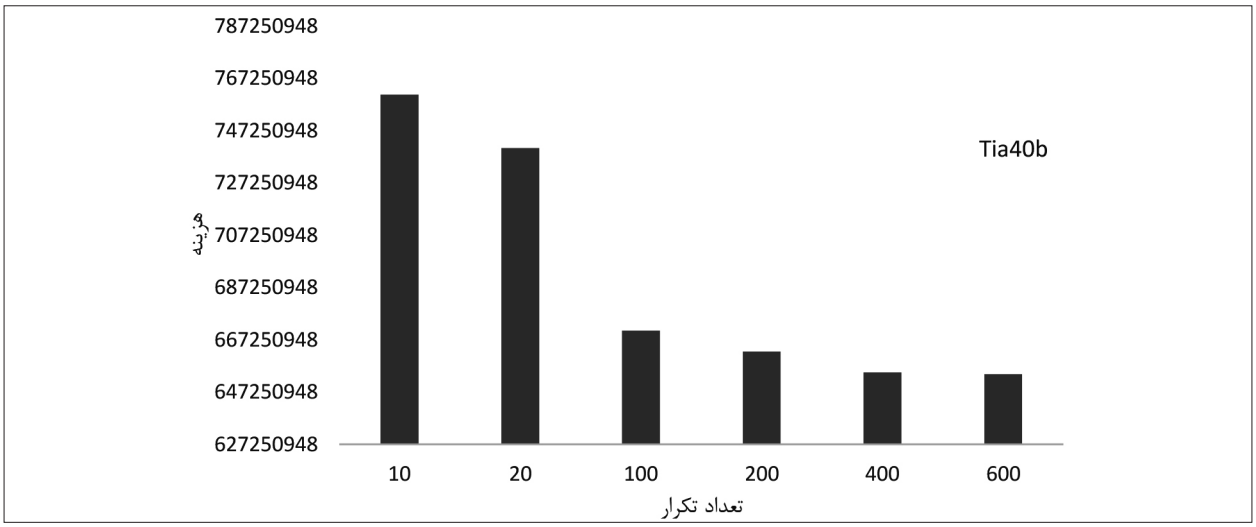
در این بخش به‌منظور مقایسه و ارزیابی دو الگوریتم سریال و موازی برای حل مسئله تخصیص درجه دوم، پنج سناریوی مختلف استفاده شده است:

در سناریوی یکم به‌منظور مقایسه عملکرد از تعداد تکرار مختلف هر الگوریتم با یکدیگر مورد مقایسه قرار گرفته شده است. (شکل ۹) در این مقایسه اندازه جمعیت اولیه به‌طور ثابت ۲۰۴۸ و اندازه مسئله یعنی تعداد مکان‌های موجود که در آن‌ها تسهیل‌ها جای می‌گیرند، ۴۰ در نظر گرفته شده است. نتایج حاکی از این است که با افزایش تعداد تکرار، برازش حل مسئله تخصیص درجه دوم کاهش می یابد. به‌منظور تاکید بیشتر مناسب‌ترین هزینه به‌دست آمده در پایان هر الگوریتم هزینه نهایی نامگذاری شده است.

در شکل۱۰ نسبت تعداد تکرار الگوریتم به هزینه نهایی برای الگوریتم ممتیکی موازی برای مسئله تخصیص درجه



شکل۹: نسبت تعداد تکرار الگوریتم به هزینه نهایی در الگوریتم ممتیک سریال



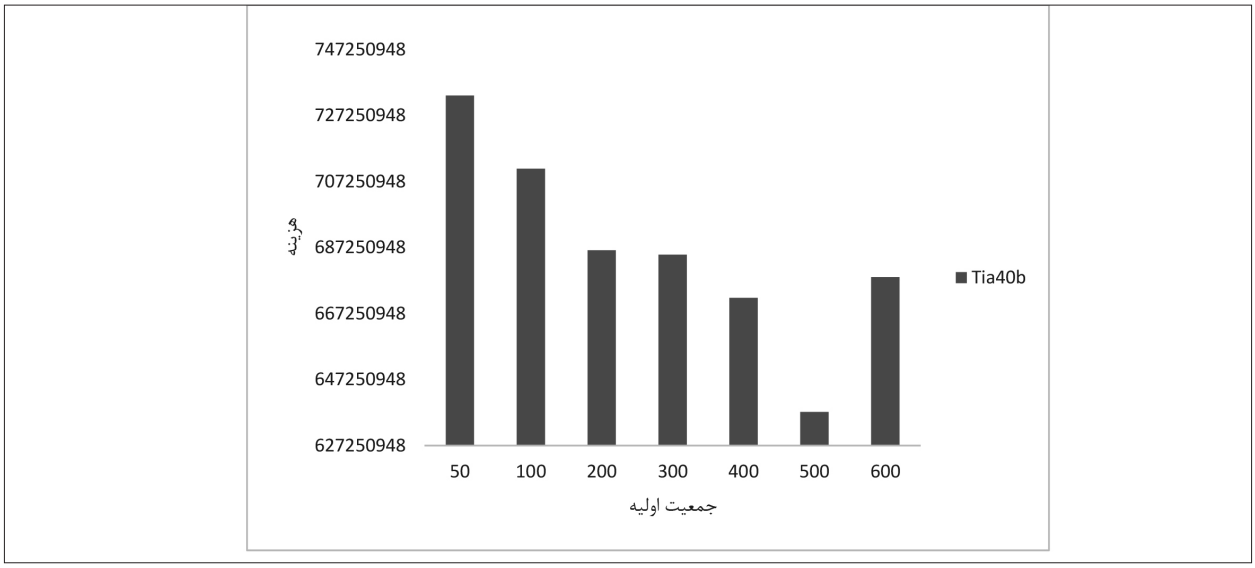
شکل۱۰: نسبت تعداد تکرار الگوریتم به هزینه نهایی در الگوریتم ممتیک موازی

دوم با اندازه ٤٠ نشـــان داده شـــده است. لازم به توضیح اســـت مانند روش قبل منظور از اندازه مسئله همان تعداد مکان‌هایی است که تسهیل‌ها در آن جای می‌گیرند. در این ارزیابی همـــواره میزان جمعیت اولیه ۲۰٤۸ در نظر گرفته شده اســـت. همان‌طور که نمودار نشان می‌دهد با افزایش تعداد تکرار الگوریتم پیشـــنهادی مـــوازی برازش کمتری برای راه‌حل‌ها از خود نشان می‌دهد.

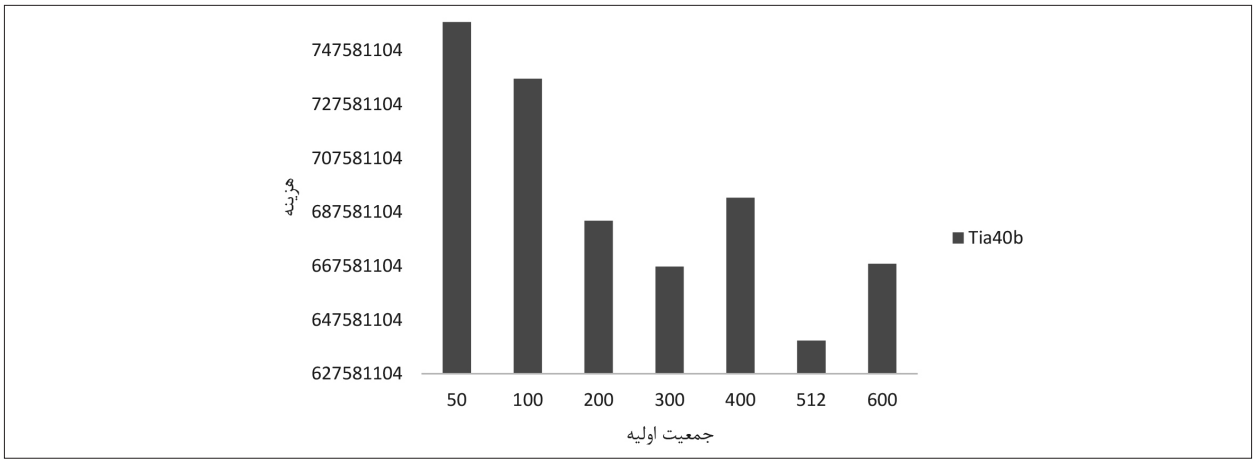
در ســـناریوی دوم مقایســـه، از تعـــداد جمعیت اولیه متفاوت اســـتفاده شده است.(شـــکل ۱۱) در حالت سریال تعداد تکرار به‌طور ثابت ۱۰۰۰ در نظر گرفته شده است.

در شـــکل۱۲ میزان تاثیر جمعیت اولیه بر هزینه نهایی الگوریتم ممتیک موازی برای مســـئله Tia40b نشـــان داده شـــده است. در آن میزان تکرار به‌طور ثابت ۱۰۰۰ در نظر گرفته شده است.

در سناریوی ســـوم مقدار تسریع الگوریتم سریال و موازی به تصویر کشـــیده شده اســـت. (شکل۱۳) در این نمودار میزان سرعت سریال بر موازی تقسیم شده است (speed up) و در محـــور y قرار گرفته اســـت و جمعیت اولیه هم در هر اجرا افزایش داده شده است. همان‌طور که از نمودار مشخص می‌باشد با افزایش جمعیت اولیه زمان سریال نسبت به موازی افزایش داشته است. این عملکرد بهتر الگوریتم ممتیک موازی را نسبت به سریال را نشان



شکل۱۱: مقایسه میزان تاثیر جمعیت اولیه بر نتیجه نهایی الگوریتم ممتیک سریال



شکل۱۲: میزان تاثیر جمعیت اولیه بر نتیجه نهایی الگوریتم ممتیک موازی

می‌دهد. در این مسئله نیز تعداد تکرار به‌طور ثابت ۲۰۴۸ در نظر گرفته شده است.

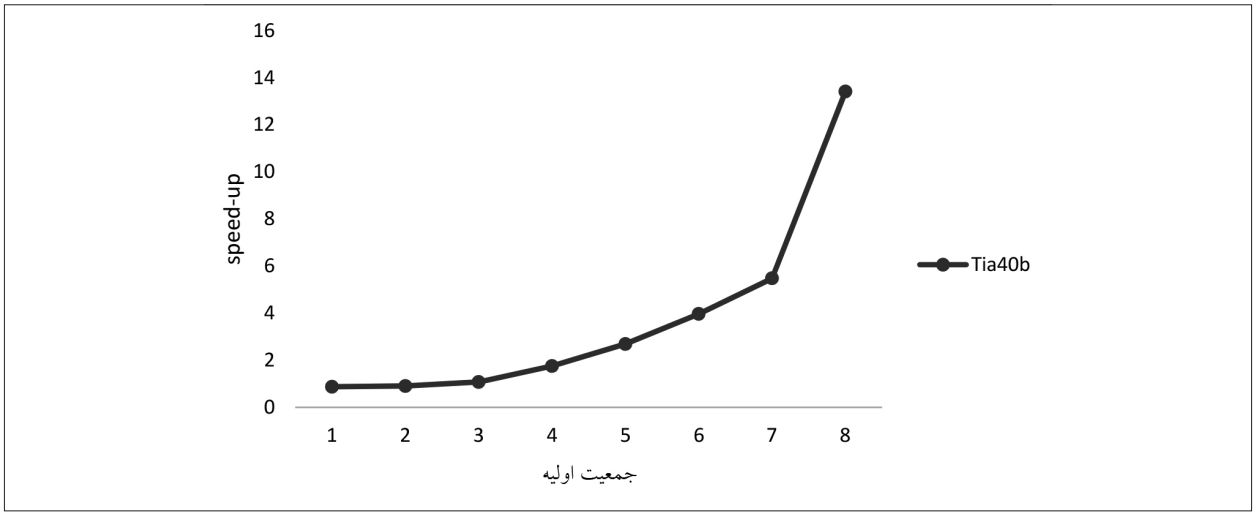
در سناریوی چهارم مقایسه، کارائی الگوریتم ممتیک سریال و موازی برای ابعاد مختلف مسئله مورد مقایسه قرار گرفته شده است. در ادامه به بررسی این مورد پرداخته شده است.

٤-١ مقایسه عملکرد روش سریال و موازی برای حل مسئله تخصیص درجه دوم

در جدول۲ نتایج به‌دست آمده از روش سریال و موازی برای حل مسئله تخصیص درجه دوم مورد مقایسه قرار گرفته است. به‌منظور مقایسه این روش‌ها از مسائل مختلف موجود در کتابخانه QAPlib استفاده شده است[۲۲,۲۳]. همچنین به‌منظور مقایسه نتایج از دو جهت ارزیابی شده اند: یکی میزان Gap که از رابطه (۵) به‌دست می‌آید که در آن منظور از BKS بهترین جوابی است که تاکنون در QAPlib ثبت شده است. Solution نیز راه‌حل فعلی به‌دست آمده می‌باشد.

$$Gap = \frac{Solution-BKS}{BKS} * ۱۰۰ \qquad (۵)$$

دومی استفاده از فرمول Speed-Up برای بررسی وضعیت بهبود عملکرد زمان رسیدن به پاسخ می‌باشد دررابطه (٦) این محاسبه نشان داده شده است. Ts: زمان



شکل ۱۳: میزان تسریع الگوریتم ممتیک موازی نسبت به الگوریتم پیشنهادی سریال

جدول ۲: مقایسه نتایج پردازش روش‌های سریال و موازی برای حل مسئله QAP

QAP instance			Memetic(GA+TS) Cuda			Memetic(GA+TS)			Speed-up
Problems	N	BKS	Solution	Gap	(Time(s	Solution	Gap	Time(s)	
Tai12b	۱۲	۳۹۴۶۴۹۲۵	۳۹۴۶۴۹۲۵	۰	۴,۶۹۲۱۱۷۶۹	۳۹۴۶۴۹۲۵	۰	۳۷,۹۴۴	۸,۰۹
Tai20b	۲۰	۱۲۲۴۵۵۳۱۹	۱۲۲۴۵۵۳۱۹	۰	۸,۸۵۳۲۴۱۹۲	۱۲۳۰۰۹۵۱۳	۰,۴۵	۹۱,۲۲۸	۱۰,۳۰
Tai25b	۲۵	۳۴۴۳۵۵۶۴۶	۳۴۴۵۹۴۷۷۸	۰,۰۷	۱۲,۹۹۳۲۶۰۳۸	۳۴۴۸۵۵۱۶۰	۰,۱۵	۱۵۳,۹۴۷	۱۱,۸۵
Tai30b	۳۰	۶۳۷۱۱۷۱۱۳	۶۳۷۱۳۷۹۵۱	۰	۱۷,۶۴۰۳۹۲۳۰	۶۳۸۶۳۷۰۶۹	۰,۲۴	۲۲۶,۳۴۲	۱۲,۸۳
Tai35b	۳۵	۲۸۳۳۱۵۴۴۵	۲۸۳۹۴۲۰۰۷	۰,۲۲	۴۸,۹۳۹۸۳۰۷۸	۲۸۷۵۲۶۶۸۶	۱,۴۹	۵۷۲,۸۱۸	۱۱,۷۰
Tai40b	۴۰	۶۳۷۲۵۰۹۴۸	۶۳۷۲۵۰۹۴۸	۰	۳۳,۶۱۵۶۸۸۳۲	۶۵۵۳۹۸۸۸۴	۲,۸۵	۳۳۴,۱۰۷	۹,۹۴
Tai50b	۵۰	۴۵۸۸۲۱۵۱۷	۴۳۰۱۴۰۲۶۹	-۶,۲۵	۱۵,۰۴۶۵۱۷۳۷	۴۳۳۵۳۰۷۸۸	۵,۵۱-	۱۹۴,۸۴۹	۱۲,۹۵

محاسبه روش سریال و Tp : زمان محاسبه روش موازی می‌باشد.

$$Speed - up = \frac{\mathrm{Ts}}{\mathrm{Tp}}. \qquad (۶)$$

نتایج حاکی از عملکرد بهتر روش موازی تا حدود ۱۳برابر موازی نسبت به روش سریال می‌باشد. همچنین با سرعت مناسب به Gap حدود صفر در اکثر مسئله با اندازه مختلف رسیده است.

۵- نتیجه گیری و پیشنهادات

در این مقاله برای حل مسئله تخصیص درجه دوم از الگوریتم فراابتکاری استفاده شده است. بدین منظور در ابتدا الگوریتم ژنتیک با الگوریتم جستجوی ممنوعه به‌منظور بهبود عملکرد استخراج راه‌حل‌های الگوریتم ژنتیک استفاده شده است. به‌منظور کاهش سربار خروجی الگوریتم ژنتیک در هر تکرار به الگوریتم جستجوی ممنوعه داده شده است. همچنین به‌منظور افزایش سرعت اجرای الگوریتم ممتیکی در بستر کودا به‌صورت موازی هم اجرا شده است. به‌منظور بررسی نتایج مسئله تخصیص درجه دوم با اندازه‌های مختلف با حالت استاندارد QAPlib مورد ارزیابی قرار گرفته شده است. نتایج و ارزیابی الگوریتم پیشنهادی موازی نسبت به حالت سریال از نظر سرعت اجرا تا حدود ۱۳ برابر افزایش داشته است. همچنین در تعداد تکرار یکسان برازش به‌دست آمده در حالت موازی با فاصله حدود برابر صفر نسبت به حالت استاندارد کتابخانه QAPlib می‌باشد. پیشنهاد می‌شود الگوریتم‌های دیگر با

الگوریتم ژنتیک ترکیب شـــود و با نتایـــج این مقاله مورد مقایسه قرار گیرد.

مراجع

1. Koopmans T, Beckmann M ,” Assignment problems and the location of economic activities”, Econometrica, Vol. 25, pp. 53–76,1957.

۲. محمدی هادی، میرزائی کمال، «حل مسئله فروشنده دوره گرد با الگوریتم ترکیبی بهینه‌سازی جغرافیای زیستی و شبیه‌سازی تبرید»، سومین کنفرانس محاسبات تکاملی و هوش جمعی، بم،زمستان ۱۳۹۶.

3. Mokhtar E., Lhassane I., Julien L. and Mathieu B.,” Gpuparallelization strategies formetaheuristics: a survey”, International Journal Of Parallel, Emergnt and Distributed Systems, 2018.

4. Roberto P., Jonatan G., ”Solving the Quadratic Assignment Problem (QAP) Through a Fine-Grained Parallel Genetic Algorithm Implemented on GPUs”, International Conference on Computational Collective Intelligence, Computational Collective Intelligence pp 145-154, ICCCI 2018.

5. Dawkins Richard, “The Selfish Gene”, Oxford University Pres, 2006.

۶. الـه دانه علـی اکبر، کیافر پیام، برنامه نویسـی موازی با کودا، انتشـارات ناقوس،۱۳۹۷.

۷. شـاه بهرامی اسدالله،جم صدیقه، پردازش موازی و برنامه نویسی با GPU، دانشگاه گیلان و انتشارات نص، ۱۳۹۶.

8. NVIDIA CUDA C Programming Guide, Version 4.2,2012, www.nvidia.com.

9. Abdel M. Doaa B.G., Mirjalili S.A., “Integrating the whale algorithm with Tabu search for quadratic assignment problem: A new approach for locating hospital departments”, Elsevier, Applied Soft Computing Volume 73, Pages 530-546, December 2018.

۱۰. محمـدی جواد، میرزائی کمال، الگوریتم ممتیک موازی برای حل مسـئله تخصیص درجه دو مبتنی بر GPU، پایان نامه کارشناسـی ارشد، دانشگاه علم و هنر، یزد، ۸۰ صفحه، زمستان ۹۴.

۱۱. کامران پور مرضیه، یعقوبی مهدی، کشاورزیان پیمان، ”ایجاد یک الگوریتم ممتیک مبتنی بر کرم شـب تاب و تئوری آشـوب“، یازدهمین کی‌فرانس سیستم های هوشمند ایران، اسفندماه ۱۳۹۱.

12. Dongli Jia , Guoxin Z. , Khurram Khan M. ,“An effective memeticdifferential evolution algorithm based on chaotic local search”, Information Sciences 181 (2011) 3175–3187,2011.

۱۳. کشـاورزی پریسـا، دلداری حسـین، ابریشـمی سـعید، «یک الگوریتم جستجوی اول سطح کارآمد گراف بر روی CPUوGPU»، نشریه مهندسی برق مهندسی کامپیوتر ایران، سال ۱۳، شماره ۲، زمستان ۱۳۹۴.

۱۴. رفیعی، عباسـی، نصیری، «روشی کارا برای پیاده‌سـازی موازی الگوریتم دسته بندی بسته درخت سلسله مراتبی بر روی واحد پردازش گرافیکی»، نشریه مهندسی برق دانشگاه تبریز، جلد ۴۶، شماره ۳، پائیز ۹۵.

۱۵. رفیعـی علی، موسـوی مرتضی، «ارائـه یک الگوریتم موازی بهینه سـازی غذایابی باکتری پیاده سـازی شـده در واحد پردازش گرافیکی»، نشـریه مهندسـی برق و مهندسـی کامپیوتر ایران، سـال ۱۵، شـماره ۲، تابستان ۱۳۹۶.

۱۶. سیر فاطمه، مظفری سعید، «افزایش سرعت الگوریتم حذف درز با تجزیه بـه زیر تصاویر زوج و فرد»، نشـریه مهندسـی برق و مهندسـی کامپیوتر ایران، سال ۱۵، شماره ۴، زمستان ۱۳۹۶.

۱۷. امیدوار روح‌الله، نجاتیان صمد، پروین حمید، «بهبود الگوریتم بهینه‌سـازی SSPCO بـا اسـتفاده از دو نظریـه آشـوب»، مجلـه علوم رایانشـی، بهار ۱۳۹۷.

۱۸. جولا امین، خاتون ناصری نرجس، رحمانی امیرمسـعود، «اسـتفاده ار مدل محاسـباتی مخچه برای محاسبه نرخ جهش در الگوریتم ممتیک برای حل مسـئله برنامه‌ریزی دروس دانشگاهی»، سیزدهمین کنفرانس ملی انجمن کامپیوتر ایران، جزیره کیش، اسفندماه ۸۶.

19. Petalas Y.G., Parsopoulos K.E., Vrahatis M.N., “Memetic particle swarm optimization”, springer, DOI 10.1007/s10479-007-0224-y, August 2007.

20. Puglierin F., ,” A Bandit-Inspired Memetic Algorithm for Quadratic Assignment Problems”, utrecht university, August 2012.

۲۱. اعیان زاده رامین، تشنه لب محمد، «تکامل تدریجی رفتار در الگوریتم‌های ممتیک با اسـتفاده از سازگاری در تقلید افراد جامعه»، هشتمین کنفرانس سیستم های هوشمند،دانشگاه فردوسی مشهد،شهریورماه ۸۶.

22. QAPLIB. 1997. [cited 2017 Mar 31]. Available from: http://anjos.mgi.polymtl.ca/qaplib/.

23. Durkota K., “ Implementaion Of a Discrete Firefly Algorithm For The Qap Problem Within The Seage Framework”, Czech Technical University in Prague Faculty of Electrical Engineerin gy, May 2011.