مجله علمی-پژوهشی
رایانش نرم و فناوری اطلاعات
جلد۲، شماره۱، سال ۱۳۹۲

رویکرد رقابت استعماری یادگیر برای حل مسئله تخصیص درجه دوم

علی صفری ممقانی[۱] و محمدرضا میبدی[۲]

[۱]دانشگاه آزاد اسلامی، واحد بناب، دانشکده برق و کامپیوتر، گروه مهندسی کامپیوتر، بناب، ایران، ali.safari.m@gmail.com

[۲]دانشگاه صنعتی امیرکبیر، دانشکده کامپیوتر و فن‌آوری اطلاعات، گروه مهندسی کامپیوتر، تهران، ایران، mmeybody@aut.ac.ir

چکیده – مسئله QAP[۱] یک مسئله NP کامل می‌باشد و الگوریتم‌های قطعی تنها قادر به حل نمونه‌های کوچکتر این مسئله (حداکثر به اندازه ۲۰) می‌باشند. بنابراین، توسعه روش‌های متاهیوریستیکی برای حل آن می‌تواند مورد توجه واقع شود، در این مقاله دو الگوریتم مبتنی بر روش ICA و روش ترکیبی ICA[۲] و آتوماتون یادگیر مهاجرت اشیا برای حل مسئله ارائه شده است. الگوریتم رقابت استعماری معمولی دارای این خاصیت است که عموما کشورهای (جواب‌های) ضعیف‌تر پس از مدتی به طرف کشورهای قویتر متمایل خواهند شد و احتمالا منجر به گیر کردن الگوریتم در بهینه های محلی خواهند شد، ولی با آشفته کردن آگاهانه فضای جواب می‌توانیم به جواب‌هایی که بعنوان بهینه سراسری تلقی شوند دست یابیم که در الگوریتم ترکیبی ارائه شده این نقش به عهده آتوماتون یادگیر می‌باشد.

برای بررسی کارایی الگوریتم‌ها، تعدادی مسائل نمونه با اندازه‌های متفاوت از مجموعه استاندارد $QAPLIB$ بکار گرفته شده‌اند که این مسائل نمونه عمدتا از مسائل مهندسی واقعی گرفته شده‌اند. نتایج نشان دهنده این است که الگوریتم ترکیبی جدید قادر می‌باشد که جواب‌های بسیار مناسبی را تولید نماید که بیشتر این جواب‌ها بهینه و یا نزدیک به بهینه خواهند بود. بنابراین، این روش یک الگوریتم کارا در حل مسئله QAP خواهد بود.

کلید واژه‌ها- مسئله تخصیص درجه دوم، مسائل NP، الگوریتم‌های متاهیوریستیک، الگوریتم رقابت استعماری، آتوماتون یادگیر مهاجرت اشیا

۱. مقدمه

مسئله تخصیص درجه دو در طراحی جایابی ساختمان‌ها و همچنین چیدمان تجهیزات واحدهای صنعتی و ... می‌تواند بعنوان یک مسئله بهینه‌سازی ترکیبی فرموله شود. شرکت‌های تولیدی، زمان و هزینه زیادی را برای طراحی و یا طراحی مجدد تجهیزات خود می‌پردازند. این طراحی‌ها تاثیر عمده‌ای در عملکرد سیستم دارد و جایابی تجهیزات به شکل ضعیف موجب افزایش هزینه و کاهش کارایی سیستم نسبت به آنچه که مطلوب مشتری می‌باشد خواهد بود. این مسئله که به مسئله QAP نیز مشهور است یکی از مسائل بهینه سازی ترکیبی کلاسیک می‌باشد و بعنوان یکی از مشکل‌ترین مسائل در این کلاس در نظر گرفته می‌شود. بخاطر NP کامل بودن مسئله، الگوریتم‌های قطعی تنها قادر به حل مسائل کوچکتر (حداکثر به اندازه ۲۰) می‌باشند. بنابراین، توسعه روش‌های تصادفی و متاهیوریستیک برای حل آن می‌تواند مورد توجه واقع شود، زیرا این الگوریتم‌ها قادر هستند که جواب‌های نزدیک به بهینه را مدت زمان معقول بدست آورند.

۱.۱. تعریف مسئله تخصیص درجه دوم

قبل از شروع بحثمان ابتدا باید مسئله تخصیص درجه دوم بطور کاملا واضح تعریف گردد. در این قسمت یک توصیف فرمال از مسئله ارائه می‌شود.

با داشتن یک مجموعه به نام $N = \{1,2,3,...,n\}$ و ماتریس‌های $n \times n$ به نام $F = \{f_{ij}\}$ و $D = \{d_{ij}\}$ و ماتریس $C = \{c_{ij}\}$، هدف در مساله QAP یافتن یک جایگشت از مجموعه N می‌باشد که بتواند مقدار تابع زیر را کمینه کند.

$$z = \sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij} d_{\phi(i)\phi(j)} + \sum_{i=1}^{n} c_{i\phi(i)} \qquad (۱)$$

به عنوان یک کاربرد از مساله QAP، مسئله طراحی پردیس دانشکده‌ها[۳] را در نظر بگیرید. دریک پردیس قرار است که تعدادی تجهیزات (ساختمان) جدید ایجاد شود و هدف مینیمم کردن میزان فاصله حرکتی برای دانشجویان و کارمندان داخل پردیس می‌باشد. فرض کنید که n مکان موجود باشد و قرار است که n ساختمان در آنها قرار گیرد. در نظر بگیرید که d_{kl} فاصله حرکتی بین دو مکان k و l باشد که قرار است ساختمان‌های جدید در آنها بنا شود. همچنین فرض کنید که $f_{i,j}$ برابر با

تعداد دانشجویانی باشد که در هر هفته بین دو ساختمان i و j حرکت می‌کنند. مسئله QAP، تخصیص ساختمان‌ها به مکان‌های موجود می‌باشد بطوریکه میزان مسافت پیموده توسط افراد، کمینه شود[۱].

هر تخصیص بصورت ریاضی به فرم یک جایگشت ϕ از عناصر مجموعه $N=\{1,2,3,...,n\}$ تعریف می‌شود، بطوری‌که مفهوم $\phi(i)=k$ بدین معنی است که ساختمان i به محل k تخصیص داده شده است. حاصلضرب $f_{ij}d_{\phi(i)\phi(j)}$ میزان مسافت پیموده هفتگی توسط افرادی می‌باشد که بین دو ساختمان i و j در حال تردد هستند. بنابراین مسئله کمینه کردن مجموع مسافت پیموده شده توسط افراد پردیس می‌تواند به مسئله شناسایی تخصیص ϕ تبدیل شود که قرار است تابع z تعریف شده را کمینه کند. این نمونه ای از کاربرد QAP می‌باشد که در آن $c_{ik}=0$ می‌باشد. یعنی فرض می‌کنیم که هزینه ایجاد یک ساختمان به محل آن بستگی ندارد. در تابع z، مقدار $c_{i,k}$ دلالت بر هزینه ایجاد ساختمان i در محل k دارد.

کاربردهای دیگر مسئله QAP می‌تواند شامل چیدمان ماشین آلات یک واحد صنعتی[۴]، مسئله سیم بندی مدار[۵]، طراحی پانل‌های کنترلی و صفحه‌کلید دستگاه تایپ[۶]، مرتب‌سازی داده‌های مرتبط به هم در یک نوار مغناطیسی[۷]، تخصیص فرایندها به پردازنده‌ها در یک محیط پردازشی توزیع شده، مسئله چینش در طراحی VLSI، تحلیل واکنش‌های شیمیایی در ترکیبات آلی، رتبه‌بندی داده‌های باستان‌شناسی باشند[۱].

۱.۲. الگوریتم‌های قبلی بکار رفته برای حل مسئله QAP

کاربردهای متنوع مسئله QAP و سختی ذاتی آن، باعث شده است که مسئله توسط محققین مختلفی مورد بررسی قرار گرفته است. اثبات شده است که مسئله QAP یک مسئله NP کامل است. بنابراین الگوریتم‌های قطعی و هیوریستیکی مختلفی برای حل آن ارائه شده است.

الگوریتم‌های قطعی برای حل این مسئله شامل روش برنامه‌نویسی پویا [۲]، روش صفحه برش[۸] [۳]، روش انشعاب و تحدید[۴و۵] می‌باشند. از میان این روش‌ها، فقط الگوریتم انشعاب و تحدید است که رسیدن به جواب بهینه را تضمین می‌کند ولی این روش نیز قادر به حل مسائل با اندازه بزرگتر از ۲۰ نمی‌باشد. از آنجایی که اکثر کاربردهای مسئله QAP در دنیای واقعی برای مسائل با اندازه بزرگتر می‌باشد، نیاز به هیوریستیک‌های مناسبی می‌باشد که بتواند مسائل با اندازه‌های بزرگ را حل کند. هیوریستیک‌های مشهور برای حل آن به دسته‌های زیر تقسیم می‌شوند:

روش‌های ساخت[۹] [۶و۷]، روش‌های شمارش محدود شده[۱۰] [۸و۹] ، روش‌های بهبود محلی[۱۱] [۱۰]، روش‌های شبیه‌سازی سرد کردن فلزات [۱۱]، روش‌های جستجوی تابو [۱۲و۱۳] و الگوریتم‌های ژنتیکی [۱۴-۱۲]. از بین این روش‌ها، الگوریتم جستجوی تابوی ارائه شده توسط Kapov-Skorin[۱۲] و الگوریتم بهبود محلی تصادفی ارائه شده توسط Li و همکارانش [۱۰] و همچنین الگوریتم ارائه شده دیگر توسط Pardalos و همکارانش[۱۵] که به آنها روش جستجوی حریصانه تصادفی تطابقی[۱۲] گفته می‌شود، دو نمونه از مناسب‌ترین الگوریتم‌های حریصانه برای حل مسئله QAP می‌باشند.

سایر الگوریتم‌های هیوریستیکی که برای حل مسئله QAP بکار گرفته شده‌اند عبارت از : الگوریتم کلونی مورچه‌ها[۱۶و۱۷]، الگوریتم اتصال دوباره مسیر[۱۳] [۱۸]، ترکیب الگوریتم GRASP با جستجوی تابو[۱۹]، الگوریتم ترکیبی کلونی مورچه و ژنتیک و جستجوی محلی[۲۰] و الگوریتم جستجوی تابوی موازی هستند. ساختار بقیه مقاله به این صورت است که بخش دوم و سوم به ترتیب مقدمه‌ای در مورد ساختار الگوریتم رقابت استعماری و آتوماتون یادگیر می‌باشند. بخش چهارم اختصاص به الگوریتم‌های جدید ارائه شده برای حل مسئلهQAP دارد و نهایتا در بخش پنجم نتیجه گیری مقاله بحث خواهد شد.

۲. مقدمه ای بر الگوریتم رقابت استعماری

الگوریتم رقابت استعماری یک الگوریتم جدید در زمینه محاسبات تکاملی است. همانند دیگر الگوریتم‌های تکاملی، این الگوریتم، نیز با تعدادی جمعیت اولیه تصادفی که هر کدام از آنها یک "کشور" نامیده می‌شوند، شروع می‌شود. تعدادی از بهترین عناصر جمعیت (معادل نخبه‌ها در الگوریتم ژنتیک) به عنوان امپریالیست[۱۴] انتخاب می‌شوند. باقیمانده جمعیت نیز به عنوان مستعمره[۱۵] در نظر گرفته می‌شوند[۲۱]. در یک مسئله‌ی بهینه‌سازی N_{var} بعدی ، یک کشور، یک آرایه‌ی $1\times N_{var}$ است. این آرایه به صورت زیر تعریف می‌شود.

$$country=[p_1,p_2,p_3,...,p_{N_{var}}] \quad (۲)$$

هزینه یک کشور با ارزیابی تابع f به ازای متغیرهای $(p_1,p_2,p_3,...,p_{N_{var}})$ یافته می‌شود. بنابراین

$$C_i=f(country_i)=f(p_{i1},p_{i2},...,p_{iN_{var}}) \quad (۳)$$

برای شروع الگوریتم، تعداد $N_{country}$ کشور اولیه را ایجاد می‌کنیم. N_{imp} تا از بهترین اعضای این جمعیت (کشورهای دارای کمترین مقدار تابع هزینه) را به عنوان امپریالیست انتخاب می‌کنیم. باقیمانده N_{col} تا از کشورها، مستعمراتی را تشکیل می‌دهند که هرکدام به یک امپراطوری تعلق دارند. برای تقسیم مستعمرات اولیه بین امپریالست‌ها، به هر امپریالیست، تعدادی از مستعمرات را که این تعداد، متناسب با قدرت آن است، می‌دهیم. کشورهای استعمارگر با اعمال سیاست جذب[۱۶] (همگون سازی) در راستای محورهای مختلف بهینه سازی، کشورهای مستعمره را به سمت خود می‌کشند. رقابت استعماری[۱۷] در کنار سیاست همگون سازی که در شکل۱ نشان داده شده است هسته اصلی این الگوریتم را تشکیل می‌دهد و باعث می‌شود کشورها به سمت مینیمم تابع حرکت کنند. استعمارگران بسته به قدرتشان، این مستعمرات را با رابطه (۴) به سمت خود می‌کشند. قدرت کل هر امپراطوری، به هر دو بخش تشکیل دهنده آن یعنی کشور امپریالیست (به عنوان هسته مرکزی) و مستعمرات آن، بستگی دارد. در حالت ریاضی، این وابستگی با تعریف قدرت امپراطوری به صورت مجموع قدرت کشور امپریالیست، به اضافه در صدی از میانگین قدرت مستعمرات آن، مدل شده است. بدین ترتیب برای هزینه کل یک امپراطوری داریم.

$$T.C._n = Cost(imperialist_n) + \xi\, mean\{Cost(colonies\ of\ empire_n)\} \qquad (۴)$$

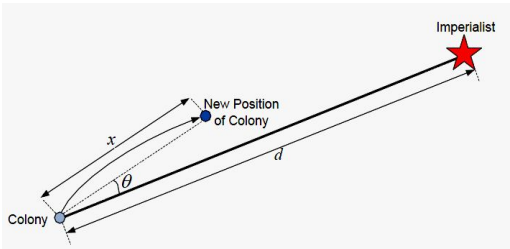
که در آن $T.C._n$ هزینه کل امپراطوری nام و ξ عددی مثبت است که معمولاً بین صفر و یک و نزدیک به صفر در نظر گرفته می‌شود. کوچک در نظر گرفتن ξ باعث می‌شود که هزینه کل یک امپراطوری تقریباً برابر با هزینه حکومت مرکزی آن (کشور امپریالیست) شود و افزایش ξ نیز باعث افزایش تاثیر میزان هزینه مستعمرات یک امپراطوری در تعیین هزینه کل آن می‌شود. در حالت نوعی $\xi = 0.05$ در اکثر پیاده‌سازی به جوابهای مطلوبی منجر شده است. کشور مستعمره به اندازه x واحد در جهت خط واصل مستعمره به استعمارگر حرکت کرده است و سپس به موقعیت جدید خود کشانده می‌شود. در شکل ۱ فاصله میان استعمارگر و مستعمره با d نشان داده شده است. x نیز عددی تصادفی با توزیع یکنواخت (و یا هر توزیع مناسب دیگر) می‌باشد. یعنی برای x داریم.

$$x \sim U(۰, \beta \times d) \qquad (۵)$$

که در آن β عددی بزرگتر از یک و نزدیک به ۲ می‌باشد. یک انتخاب مناسب می‌تواند $\beta = 2$ باشد. وجود ضریب $\beta > 1$ باعث می‌شود تا کشور مستعمره در حین حرکت به سمت کشور استعمارگر از جهت‌های مختلف به آن نزدیک شود. همچنین زاویه حرکت بصورت توزیع یکنواخت زیر در نظر گرفته شده است.

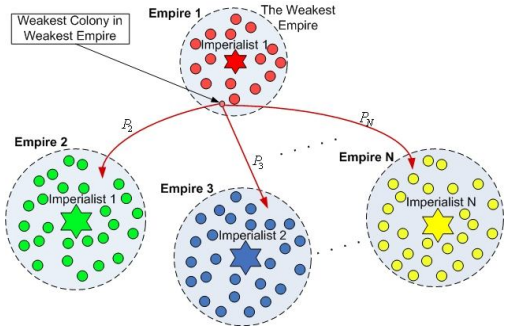
$$\theta \sim U(-\gamma, \gamma) \qquad (۶)$$

در الگوریتم ICA، با یک انحراف احتمالی، مستعمره در مسیر جذب استعمارگر پیش می‌رود. این انحراف با زاویه θ نشان داده شده است.



شکل ۱: حرکت مستعمرات به سمت امپریالیست(سیاست جذب[۲۱]

رقابت استعماری، بخش مهم دیگری از این الگوریتم را تشکیل می دهد. با شکل‌گیری امپراطوری‌های اولیه، رقابت امپریالیستی میان آن‌ها شروع می‌شود. هر امپراطوری‌ای که نتواند در رقابت استعماری، موفق عمل کرده و بر قدرت خود بیفزاید (و یا حداقل از کاهش نفوذش جلوگیری کند)، از صحنه رقابت استعماری، حذف خواهد شد. این حذف شدن، به صورت تدریجی صورت می پذیرد .بدین معنی که به مرور زمان، امپراطوری های ضعیف، مستعمرات خود را از دست داده و امپراطوری‌های قویتر، این مستعمرات را تصاحب کرده و بر قدرت خویش می‌افزایند. رقابت استعماری در شکل ۲ نشان داده شده است.



شکل۲ :رقابت استعماری[۲۱]

الگوریتم مورد نظر تا برآورده شدن یک شرط همگرایی و یا تا اتمام تعداد کل تکرارها، ادامه می یابد. پس از مدتی، همه امپراطوری ها سقوط کرده و تنها یک امپراطوری خواهیم داشت و بقیه کشورها تحت کنترل این امپراطوری واحد قرار می گیرند. شبه کد الگوریتم رقابت استعماری در شکل ۳ نشان داده شده

است.

۱. چند نقطه تصادفی روی تابع انتخاب کرده و امپراطوری‌های اولیه را تشکیل بده.
۲. مستعمرات را به سمت کشور امپریالیست حرکت بده (سیاست همسان‌سازی و انقلاب).
۳. اگر مستعمره‌ای در یک امپراطوری، وجود داشته باشد که هزینه‌ای کمتر از امپریالیست داشته باشد؛ جای مستعمره و امپریالیست را با هم عوض کن.
٤. هزینه‌ی کل یک امپراطوری را حساب کن (با در نظر گرفتن هزینه‌ی امپریالیست و مستعمراتشان).
٥. یک مستعمره از ضعیف‌ترین امپراطوری انتخاب کرده و آن را به امپراطوری‌ای که بیشترین احتمال تصاحب را دارد، بده.
٦. امپراطوری‌های ضعیف را حذف کن.
۷. اگر تنها یک امپراطوری باقی مانده باشد، توقف کن وگرنه به ۲ برو

شکل۳ : شبه کد مربوط به الگوریتم رقابت استعماری

۳. مقدمه ای بر آتوماتون یادگیر

یادگیری به معنی تغییر در رفتار و کارایی سیستم به خاطر تجربیات گذشته می‌باشد. بنابراین، مهمترین ویژگی یک سیستم یادگیرنده این است که باید قادر باشد که رفتارش را با گذشت زمان نسبت به هدف نهایی بهبود ببخشد. از دیدگاه ریاضی هدف یک چنین سیستم یادگیرنده، بهینه کردن تابعی است که ممکن است کاملا شناخته شده نباشد. آتوماتای یادگیر در محیطی احتمالی عمل نموده و قادر است بر اساس ورودی‌های دریافت شده از محیط، احتمال انجام عملیات خود را به هنگام در آورده و کارایی خود را بهبود ببخشد. این ماشین، می تواند تعداد محدودی عمل را انجام دهد و هر عملی که انجام می‌دهد، توسط محیطی احتمالی ارزیابی می‌گردد و پاسخی به آن داده می‌شود. آتاماتا از این پاسخ استفاده نموده و عمل بعدی را انتخاب می-کند. در طی این فرایند، آتوماتا، یاد می‌گیرد که چگونه بهترین عمل را انتخاب کند[۲۲و۲۳]. هدف آتوماتای یادگیر این است که از بین یکسری اعمال مجاز، عمل بهینه را تعیین کند. عمل بهینه عملی است که احتمال دریافت پاداش از محیط را به حداکثر برساند. کارکرد آتوماتای یادگیر را می‌توان به صورت چرخه های بازخورد تکراری دید که در آن، آتوماتا با محیط تعامل می‌کند. این تعامل را می‌توان در شکل۴ مشاهده کرد. در هر چرخه، آتاماتا یک عمل را انجام می‌دهد که باعث دریافت پاسخ ازمحیط می‌شود. این پاسخ می‌تواند پاداش و یا جریمه باشد. آتوماتا ازپاسخ به عنوان دانشی برای عمل بعدی استفاده می‌کند. به این ترتیب، خود را با محیط سازگار می‌کند.

شکل ۴: ارتباط بین آتاماتای یادگیر و محیط

۴. الگوریتم های جدید برای حل مسئله QAP

در این بخش، الگوریتم‌های جدیدی که برای حل مسئله تخصیص درجه دوم پیشنهاد شده است، توصیف می‌گردد. دو الگوریتم جدید برای حل این مسئله بیان گردیده است. اولین روش، الگوریتمی مبتنی بر رقابت استعماری و دومین الگوریتم نیز یک روش ترکیبی مبتنی بر رقابت استعماری و آتوماتون یادگیر می‌باشد.

۴.۱. الگوریتم ارائه شده مبتنی بر رقابت استعماری

اولین الگوریتم ارائه شده یک روش بر اساس الگوریتم رقابت استعماری می‌باشد که در بخش دوم بحث شد. حال لازم است که اجزای این الگوریتم که برای حل مسئله QAP ارائه شده است توضیح داده شود.

کدگذاری کشورها

برای کدگذاری جواب‌های مسئله هر کشور را بصورت یک آرایه $1\times n$ تعریف می‌کنیم که اندیس‌های آرایه نشان دهنده مکان‌ها و مقادیر آرایه‌ها نشان دهنده تسهیلاتی است که قرار است در آن مکان‌ها بنا شود. نمونه‌ای از یک کشور در شکل ۵ نشان داده شده است. در این شکل۱۰ تسهیلات متفاوت وجود دارد که قرار است به ۱۰ مکان مختلف تخصیص یابند. برای مثال دومین

عنصر در آرایه نشان دهنده این است که تسهیلات شماره ۵ در مکان شماره ۲ قرار خواهد گرفت.

Country=	2	5	10	7	8	3	1	4	6	9

یعنی تسهیلات پنجم به مکان دوم اختصاص یافته است

شکل ۵: نمایش کشورها برای حل مسئله QAP

در گام اول الگوریتم ICA بطور تصادفی شروع به تولید کشورها به شکل فوق خواهد نمود.

محاسبه تابع هزینه

هزینه هر کشور مطابق با تعریف مسئله محاسبه خواهد شد برای مساله QAP تابع هزینه بوسیله C_m نشان داده می‌شود که C_m نشان دهنده هزینه کشور m ام خواهد بود.این تابع بصورت زیر تعریف می‌شود.

$$C_m = Cost(country_m) = \sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij} d_{\phi(i)\phi(j)} \qquad (۷)$$

عملگر جذب

نسخه ارائه شده اولیه الگوریتم ICA بر روی مقادیر پیوسته بکار می‌رود در حالی‌که مسئله QAP یک مساله گسسته است. اما می‌توانیم عملگرهای جذب(همسان سازی) و انقلاب را که مناسب مساله باشند را تعریف کنیم. برای اعمال سیاست جذب، کشور مستعمره و استعمارگر انتخاب می‌شوند و سپس عملگر جذب بر روی آنها انجام می‌شود. مثالی از عملگر جذب بکار رفته در این الگوریتم در شکل ۶ نشان داده شده است. این عملگر بصورت زیر انجام می‌گیرد.

- بعضی از عناصر آرایه *Imperialist* (حدود نصف عناصر آرایه) را بطور تصادفی انتخاب کن(در مثال خانه های با محتوای ۱ ، ۶ ، ۲، و ۸و۷ انتخاب شده‌اند).
- عناصر انتخاب شده دقیقا به آرایه *new_colony* کپی می‌شوند.
- آرایه *colony* از ابتدا تا انتها پیمایش می‌شود و اگر یکی از عناصر آن در داخل آرایه *new_colony* نباشد به اولین مکان خالی آرایه *new_colony* کپی می‌شوند(خانه‌های با محتوای ۱۰،۵، ۳، ۴ و ۹)

Imperialist	1	5	6	2	9	10	3	8	4	7
Colony	2	5	10	7	8	3	1	4	6	9
New_Colony	1	5	6	2	10	3	4	8	9	7

شکل ۶: مثالی از عملگر جذب بکار رفته برای حل مسئله QAP

عملگر انقلاب

برای اعمال عملگر انقلاب دو عنصر از آرایه مستعمره بطور تصادفی انتخاب می‌شود و سپس محتوای آنها با همدیگر عوض می‌شود. مثالی از این عملگر در شکل ۷ نشان داده شده است.

Colony	2	5	10	7	8	3	1	4	6	9
New_Colony	2	5	6	7	8	3	1	4	10	9

شکل ۷: مثالی از عملگر انقلاب بکار رفته برای حل مسئله QAP

۴.۲. الگوریتم ارائه شده مبتنی بر ترکیب رقابت استعماری و آتوماتون یادگیر

الگوریتم دوم یک الگوریتم ترکیبی است که با افزودن مفهوم آتوماتون یادگیر به ICA بدست آمده است.

کدگذاری کشورها

در الگوریتم پیشنهادی برخلاف الگوریتم هایICA کلاسیک، از کدگذاری معمولی برای نمایش کشورها استفاده نمی شود. هر کشور توسط یک آتوماتون یادگیر از نوع مهاجرت اشیا[۱۸] نشان داده می شود به‌طوریکه هر کدام از ویژگی‌های کشور به یکی از اقدام های آتوماتون نسبت داده می شود و در یک عمق مشخصی از آن اقدام قرار می گیرند[۲۴]. برای حل این مسئله، کشورها را می‌توان به صورت شش‌تایی $<V,\alpha,\varphi,\beta,F,G>$ نشان داد که در آن:

$V=\{V_1,V_2,...,V_n\}$ مجموعه اشیای بکار رفته می‌باشند که این اشیا، همان شماره تسهیلات موجود می‌باشد که قرار است به مکان‌های متفاوت تخصیص یابند. این مجموعه دارای مقادیر ۱و۲و .. .و n می‌باشند. این اشیا بین وضعیت‌های مختلف آتوماتون حرکت می‌کنند و تخصیص‌های مختلفی از تسهیلات به مکان‌های موجود را ایجاد می‌کنند.

$\alpha=\{\alpha_1,\alpha_2,\alpha_3,...\alpha_n\}$ مجموعه اقدام‌های مجاز برای آتوماتون یادگیر است. هر یک از این اقدام‌ها نمایانگر یک مکان موجود در مجموعه مکان‌ها خواهد بود که قرار است تسهیلات به آنها تخصیص یابد. مثلا a_k نشان‌دهنده مکان k ام است. این آتوماتون n اقدام دارد (تعداد اقدام های این آتوماتون برابر با مکان‌ها می‌باشد).

$\phi=\{\phi_1,\phi_2,...,\phi_{nN}\}$ مجموعه وضعیت‌ها و N عمق حافظه

برای آتوماتون می‌باشد. مجموعه وضعیت‌های این آتوماتون به n زیر مجموعه $\{\phi_1,\phi_2,...,\phi_N\}$ و $\{\phi_{N+1},\phi_{N+2},...,\phi_{2N}\}$ و ... و $\{\phi_{(k-1)N+1},\phi_{(k-1)N+2},...,\phi_{kN}\}$ تقسیم می‌شوند و اشیا بر اساس این که در کدام وضعیت قرار داشته باشند دسته بندی می‌گردند. اگر شی u در مجموعه وضعیت‌های $\{\phi_{(j-1)N+1},\phi_{(j-1)N+2},...,\phi_{jN}\}$ قرار داشته باشد در اینصورت تسهیلات u در مکان j قرارخواهد گرفت. در مجموعه وضعیت-های اقدام j، به وضعیت $\phi_{(j-1)N+1}$ وضعیت داخلی و به وضعیت ϕ_{jN}، وضعیت مرزی گفته می شود. شئی که در وضعیت $\phi_{(j-1)N+1}$ قرار دارد شی با اهمیت بیشتر و شئی که در وضعیت ϕ_{jN} قرار دارد، شی با اهمیت کمتر نامیده می شود.

$\beta=\{0,1\}$ مجموعه ورودی‌های آتوماتون می‌باشد . در این مجموعه ۱ شکست و مقدار صفر موفقیت را نشان می‌دهد.

$F:\phi\times\beta\rightarrow\phi$ تابع نگاشت وضعیت‌ها می‌باشد. این تابع از روی وضعیت فعلی و ورودی آتوماتون وضعیت بعدی آن را تولید می‌نماید. این تابع چگونگی گردش اشیا را در وضعیت‌های آتوماتون مشخص می‌نماید.

$G:\phi\rightarrow\alpha$ تابع نگاشت خروجی می‌باشد. این تابع تصمیم می‌گیرد که به ازای هر وضعیت آتوماتون چه اقدامی را انجام - دهد. پس اگر شی u در مجموعه وضعیت‌های $\{\phi_{(j-1)N+1},\phi_{(j-1)N+2},...,\phi_{jN}\}$ قرار داشته باشد، اقدام j انتخاب می‌گردد (در اینصورت تسهیلات u در مکان j قرار خواهد گرفت). بعنوان مثال در نظربگیرید که قرار است ۶ تسهیلات به ۶ مکان متفاوت تخصیص یابد و تخصیصی به صورت [۴، ۲، ۳، ۶ ، ۱،۵] داریم. این تخصیص را می‌توانیم بصورت آتوماتون مهاجرت اشیا با اتصالات ستلین، کرینسکی و کرایلو در شکل ۸ نشان دهیم.

آتوماتون مهاجرت اشیاء مبتنی بر اتصالات آتوماتون ستلین

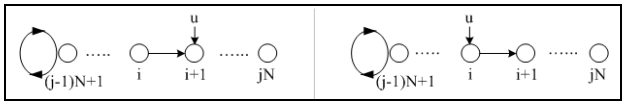
این آتوماتون مهاجرت اشیا از آتوماتون ستلین برای پاداش و یا جریمه‌کردن استفاده می کند. برای سهولت در ارائه مطالب، آتوماتونی با K اقدام و عمق حافظه N که برای حل مساله تخصیص درجه دوم بکار گرفته شده است، توسط $Tsetline-QAP(K,N)$ نمایش داده می‌شود. نحوه جریمه و پاداش دادن در شکل‌های ۹ و ۱۰ و ۱۱ نشان داده شده است. برای تشریح تابع نگاشت وضعیت ها دو حالت زیر را در نظر می گیریم:

۱) اگر شی u در مجموعه وضعیت‌های $\{\phi_{(j-1)N+1},\phi_{(j-1)N+2},...,\phi_{jN}\}$ قرار داشته باشد در اینصورت تسهیلات u در مکان j قرارخواهد گرفت. اگر مقدار هزینه شارش به (از) تسهیلات u (میزان مسافت پیموده توسط افراد به مکان متناظرش) کمتر از مقدار آستانه باشد (مقدار آستانه بصورت تطبیقی محاسبه می‌شود و در هر لحظه مقدار آن برابر با میانگین مقدار هزینه شارش به تمامی تسهیلات می‌باشد)، به این شی پاداش داده می شود و به سمت وضعیت های داخلی تر این اقدام حرکت می کند. اگر شی u در وضعیت قرار داشته باشد و پاداش بگیرد در همان وضعیت یاقی می ماند. نحوه حرکت چنین شئی در شکل ۹ نشان داده شده است.

شکل ۹: وضعیت شی u قبل از دریافت پاداش (سمت راست)، وضعیت شی u پس از دریافت پاداش (سمت چپ) [۲۴]

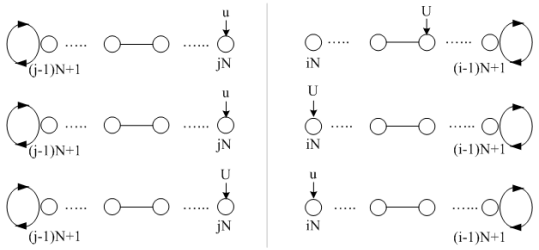
۲) اگر مقدار هزینه شارش به (از) تسهیلات u بیشتر از مقدار آستانه باشد در اینصورت، تخصیص ایجاد شده مناسب نبوده و این شی جریمه می‌شود. نحوه حرکت چنین شئی برای دو حالت مختلف در زیر آمده است:

الف) اگرشئ u در مجموعه وضعیت های $\{\phi_{(j-1)N+1},...,\phi_{jN}\}$ قرار دارد. جریمه نمودن آن باعث کاهش عمق این شی می گردد. نحوه حرکت چنین شئی در شکل ۱۰ نشان داده شده است.



شکل ۱۰: وضعیت شی u قبل از جریمه شدن (سمت راست)، وضعیت شی u پس از جریمه شدن (سمت چپ) [۲۴]

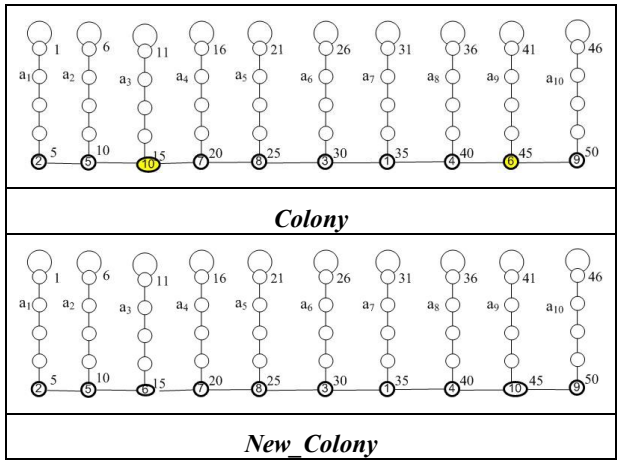
ب) شی u در وضعیت ϕ_{jN} قرار دارد. در این حالت شیU از تخصیص موجود را پیدا کرده بطوریکه اگر در تخصیص موجود جای u و U عوض شوند، بیشترین کاهش در مقدار هزینه شارش کل تسهیلات حاصل گردد. در اینصورت اگر U در وضعیت مرزی قرار داشته باشد جای u و U عوض می شود و در غیر اینصورت ابتدا U به وضعیت مرزی اقدام خود منتقل و سپس جابجایی صورت می پذیرد. نحوه حرکت چنین شئی در شکل ۱۱ نشان داده شده است.



شکل۱۱: وضعیت شی u قبل از جریمه شدن(شکل بالا)، انتقال شیU به وضعیت مرزی(شکل وسط)، وضعیت شی پس از جریمه شدن (شکل پایین) [۲۴]

عملگر جذب و انقلاب

عملگر جذب و انقلاب دقیقا همانند عملگرهای بکار رفته در الگوریتم قبلی ICA خواهد بود با این تفاوت که فقط بجای عناصر بکار رفته در آرایه کشور از اشیا آتوماتون مهاجرت اشیا استفاده می‌کنیم. بعنوان مثال نمونه‌ای از عملگر انقلاب در آتوماتون ستلین در شکل ۱۲ نشان داده شده است.



شکل ۱۲:نحوه انجام عملگر انقلاب

اکنون، با توجه به توصیفات قبلی می‌توانیم الگوریتم بکار رفته برای حل مسئله را نشان دهیم. شبه کد الگوریتم ترکیبی در شکل ۱۳ نشان داده شده است.

۱- تعدادی کشور بطور تصادفی ایجاد کن و امپراطوری-های اولیه را تشکیل بده.

۲- مستعمرات را به سمت کشورهای امپریالیست حرکت بده(سیاست همگون‌سازی و انقلاب)

۳- اگر مستعمره‌ای در یک امپراطوری، وجود داشته باشد که هزینه‌ای کمتر از امپریالیست داشته باشد؛ جای مستعمره و امپریالیست را با هم عوض کن.

۴- هزینه‌ی کل یک امپراطوری را حساب کن (با در نظر گرفتن هزینه‌ی امپریالیست و مستعمراتشان).

۵- یک مستعمره از ضعیف‌ترین امپراطوری انتخاب کرده و آن را به امپراطوری‌ای که بیشترین احتمال تصاحب را دارد، بده.

۶- امپراطوری‌های ضعیف را حذف کن.

۷- در هر امپراطوری، کشور امپریالیست و مستعمره‌های آن را انتخاب کن و گام های زیر را بطور جداگانه بر روی همه آنها اعمال کن.

۷-۱- مراحل زیر را n بار انجام بده.

۷-۱-۱- یک عدد تصادفی u بین $1..n$ انتخاب کن.

۷-۱-۲- اگر هزینه شارش تسهیلات u کمتر از حد آستانه باشد آنگاه به گرهu پاداش بده و در غیر اینصورت گره u را جریمه کن.

۸- اگر تنها یک امپراطوری باقی مانده باشد، توقف کن وگرنه به ۲ برو.

شکل ۱۳: شبه کد الگوریتم ترکیبی

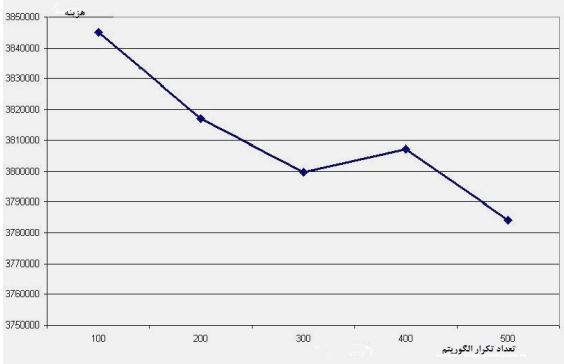
۵. ارزیابی نتایج عملی

در این بخش نتایج پیاده‌سازی الگوریتم‌های جدید توصیف شده است. تحلیل نتایج به دو بخش تقسیم شده است. در بخش اول تاثیر پارامترهای الگوریتم ICA و الگوریتم ترکیبی ICA-LA نشان داده شده است. در بخش دوم، نتایج اعمال این الگوریتم‌ها بر روی نمونه‌های استاندارد از مجموعه QAPLIB که توسط Burkard[۲۵] توسعه داده شده است را نشان می‌دهیم. این نمونه‌ها از سایت http://www.seas.upenn.edu/qaplib قابل دریافت می‌باشد.

الف) ارزیابی پارامترهای الگوریتم های ارائه شده

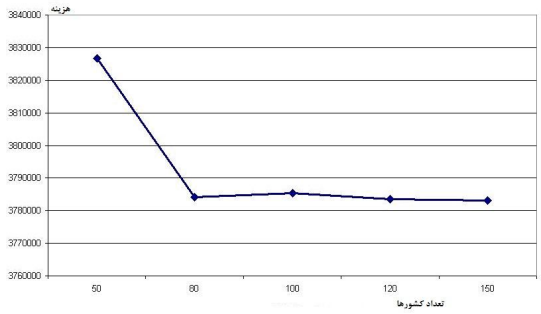
از آنجائی‌که کارایی الگوریتم‌های تصادفی تحت تاثیر تنظیم پارامترها قرار دارد، در ابتدا بهتر است که فرایند تنظیم پارامترها برای الگوریتم‌های ارائه شده جهت بدست آوردن نتایج مناسب‌تر صورت گیرد. برای انجام آزمایش‌ها مسئله bur۲۶f از کتابخانه QAPLIB مورد بررسی قرار گرفته است. اندازه این مسئله ۲۶ می‌باشد یا به‌عبارت دیگر مسئله دارای ۲۶ تجهیزات می‌باشد.

در آزمایش اول تعداد تکرارهای الگوریتم ICA را از ۱۰۰ تا ۵۰۰ افزایش می‌دهیم و داده‌های بدست آمده را ثبت می‌کنیم. نتایج بصورت شکل ۱۴ خواهد بود. اولین نکته‌ای که از شکل قابل دریافت است این است که با افزایش تعداد تکرارهای الگوریتم، جواب‌ها بهبود می‌یابد یا به‌عبارت دیگر جواب‌هایی با هزینه کمتر تولید می‌شوند. نکته دوم این است که هزینه جواب‌ها تا ۳۰۰ تکرار کاهش می‌یابد. بعد از تکرار ۳۰۰ کاهش کمتری در هزینه جواب‌ها دیده می‌شود. بنابراین جهت ایجاد تعادل بین کیفیت جواب‌ها و مدت زمان اجرایی در آزمایش‌های بعدی حداکثر تکرار الگوریتم را ۳۰۰ در نظر می‌گیریم.



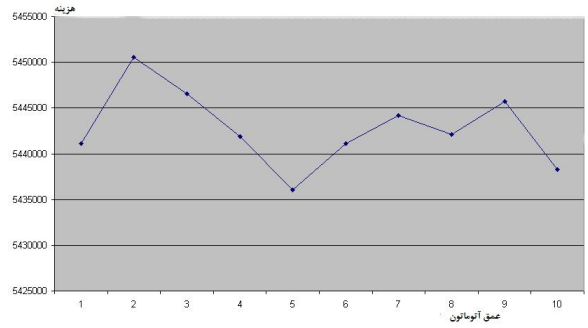
شکل ۱۴: تاثیر افزایش تعداد تکرارهای الگوریتم ICA بر روی کیفیت جوابهای تولیدی

در آزمایش بعدی به بررسی تاثیر افزایش اندازه جمعیت (تعداد کشورهای الگوریتم ICA) می‌پردازیم. بنابراین تعداد کشورها را از مجموعه {50,80,100,120,150} انتخاب می‌کنیم. نتایج بصورت شکل ۱۵ خواهد بود. همچنانکه در این شکل دیده می‌شود افزایش تعداد کشورهای الگوریتم ICA موجب بهبود جواب‌های تولیدی خواهدشد. هزینه جوابهای تولیدی به طور ناگهانی تا ۱۲۰ کشور کاهش می‌یابد و پس از آن شاهد کاهش غیرمحسوس در نتایج خواهیم بود. بنابراین در آزمایش‌های بعدی حداکثر تعداد کشورهای الگوریتم را ۱۲۰ در نظر خواهیم گرفت.



شکل ۱۵: تاثیر افزایش تعداد کشورهای الگوریتم ICA بر روی کیفیت جواب‌های تولیدی

آزمایش سوم درباره تاثیر اندازه عمق آتوماتون بر روی کیفیت جواب‌های الگوریتم ترکیبی ICA-LA خواهد بود. نتایج در شکل ۱۶ نشان داده شده است. در این آزمایش عمق آتوماتون از ۱ تا ۱۰ افزایش یافته است. با توجه به نتایج تولیدی عمق مناسب آتوماتون برابر با ۵ خواهد بود. بنابراین در آزمایش‌های بعدی عمق آتوماتون را برابر با ۵ در نظر گرفته‌ایم.



شکل ۱۶: تاثیر عمق آتوماتون یادگیر در الگوریتم ترکیبی

ب) بررسی کارایی الگوریتم‌ها

در این بخش نتایج مقایسه الگوریتم های ارائه شده و الگوریتم شبیه‌سازی سرد کردن فلزات (*SA*) بررسی شده است. نتایج در جدول ۱ و ۲ نشان داده شده است. در این جدول الگوریتم *ICA* و *ICA-LA* به ترتیب نشان دهنده الگوریتم رقابت استعماری و الگوریتم ترکیبی رقابت استعماری و آتوماتون یادگیر است که با انواع اتصالات متفاوت ستلین، کرینسکی و کرایلو آمده است. در جدول ۱ تعدادی از نمونه مسائل مجموعه *QAPLIB* با اندازه‌های متفاوت از ۱۰ تا ۱۵۰ مورد استفاده واقع شده است که ستون اول و دوم به ترتیب نشان‌دهنده نام مسئله و اندازه مسئله می‌باشد. این نمونه‌ها عمدتا از مسائل مهندسی واقعی گرفته شده

اند. ستون سوم شامل بهترین جواب شناخته شده تاکنون (*BKS*) می‌باشد. ستون *Solution*و *Gap* بترتیب نشاندهنده جواب بدست آمده توسط هر الگوریتم و درصد انحراف از جواب *BKS*[19] می‌باشد که از رابطه $\frac{(Solution - BKS)}{BKS} \times 100$ قابل محاسبه است.

با دقت در نتایج جدول ۴-۱ متوجه می‌شویم که الگوریتم ICA-$LA_{Tsetline}$ در اکثر موارد بهتر از سایر روش‌های موجود عمل می‌کند . این روش قادر می‌باشد که جوابهایی با هزینه کمتر تولید کند که درصد انحراف این جوابها از BKS کمتر از سایر روش‌ها می‌باشد و حتی در برخی موارد جوابهای تولید شده توسط ICA-$LA_{Tsetline}$ همان جوابهای BKS خواهد بود.

نکته قابل تامل دیگری که با دقت در نتایج جدول ۱ دریافت می‌شود نقش ترکیب ICA با آتوماتون یادگیر می‌باشد. همچنانکه مشاهده می‌شود نتایج الگوریتم‌های ترکیبی ICA-LA با تمامی اتصالات موجود یعنی ستلین، کرینسکی و حتی کرایلو از نتایج الگوریتم ICA ساده بهتر می‌باشد. در حقیقت این نکته نشان‌دهنده نقش آتوماتون یادگیر در بهبود نتایج الگوریتم ICA می‌باشد که با جریمه و پاداش‌های مناسب برای عناصر تخصیص داده شده در یک جایگشت، موجب ایجاد جواب‌هایی با هزینه کمتر می‌گردد و از گیر کردن الگوریتم در بهینه‌های محلی ممانعت خواهد نمود.

اما در این راستا لازم است که عوارض ترکیب دو روش نیز تحلیل شود. برای این کار مدت زمان اجرای الگوریتم ها(سرعت همگرایی جواب‌ها) مورد بررسی واقع شده است که نتایج در جدول ۲ آمده است. در این جدول میانگین جواب‌های بدست آمده برای حل تمامی نمونه‌ها آمده است.

جدول ۲: نتایج حاصل از مقایسه الگوریتم‌ها از نظر کیفیت و سرعت

مجموعه نمونه‌ها ها : شامل ۹ نمونه با اندازه های ۱۰ تا ۱۵۰					
الگوریتم	*ICA*	*ICA-LA*$_{Tsetline}$	*ICA-LA*$_{Kinsky}$	*ICA-LA*$_{Krylov}$	*SA*
میانگین مقادیر جواب ها	2241663	2187240	2189771	2211184	4054385
میانگین درصد انحراف از بهترین جواب	13.51	5.35	6.71	7.29	6.31
میانگین مدت زمان اجرایی(ثانیه)	1.65	1.81	1.85	1.76	0.85
نسبت مدت زمان اجرایی به مدت زمان SA	1.94	2.13	2.18	2.07	1

نتایج نشان‌دهنده این مورد است که هرچند کیفیت نتایج حاصل از الگوریتم ترکیبی ICA-$LA_{Tsetline}$ مناسب می‌باشند و به بهترین جواب‌های موجود نزدیک هستند ولی مدت زمان اجرای آن در مقایسه با الگوریتم ICA و SA نسبتا بیشتر است. بنظر می‌رسد که وجود چنین تفاوتی طبیعی باشد بخاطر اینکه الگوریتم ترکیبی نسبت به الگوریتم های دیگر روش حجیم تری می‌باشد. البته با بررسی نتایج جدول مشاهده می‌شود که این تفاوت چندان فاحش نمی‌باشد که کارایی الگوریتم را تحت تاثیر قرار دهد. پس می‌توان الگوریتم ترکیبی را عنوان روشی تلقی کرد که قادر است جوابهای با کیفیت بسیار مناسب را در مدت زمان معقول بدست آورد. بنابراین با در نظر گرفتن نتایج بدست آمده الگوریتم ICA-LA بعنوان یک روش کارا و قابل اتکا در حل مسئله تخصیص درجه دوم خواهد بود.

۶. نتیجه‌گیری

در این مقاله دو الگوریتم تصادفی با بکارگیری الگوریتم رقابت استعماری و همچنین ترکیب این الگوریتم و آتوماتون یادگیر برای حل مسئله QAP ارائه شده است. بکارگیری آتوماتون یادگیر در کنار ICA موجب خواهد شد که الگوریتم از گیر کردن در بهینه‌های محلی نجات یابد. با توجه با این خاصیت الگوریتم ICA که در آن کشورهای مستعمره(جواب‌های ضعیف) جذب کشورهای استعمارگر(جواب‌های قوی‌تر) خواهند شد و این عمل باعث همگرا شدن سریع جوابها خواهد شد بکارگیری آتوماتون یادگیر موجب آشفته کردن فضای جواب‌ها بصورت آگاهانه خواهد شد که با تکرار اجرا، الگوریتم قادر خواهد بود که پس از مدتی به بهینه سراسری دست یابد. در حقیقت در این روش ترکیبی، استفاده از الگوریتم رقابت استعماری موجب exploration و استفاده از آتوماتون یادگیر موجب exploitation در فضای جواب خواهد شد. نتایج مقایسه‌ها نشان‌دهنده این نکته می‌باشد که الگوریتم ترکیبی در اکثر موارد قادر به بدست آوردن جواب‌های بهینه و یا نزدیک به بهینه در حل مسئله QAP خواهد بود. برای بررسی کارایی الگوریتم نمونه هایی از مسائل مجموعه استاندارد QAPLIB مورد استفاده واقع شده‌اند. در نهایت با در نظر گرفتن نتایج بدست آمده می‌توان این الگوریتم را بعنوان یک روش ترکیبی قابل اتکا و کارا در حل مسئله QAP تلقی کرد.

سپاسگزاری

این مقاله تحت حمایت مالی دانشگاه آزاد اسلامی واحد بناب انجام گرفته است. لذا از همکاری مسئولین دانشگاه که در انجام این پژوهش ما را یاری کرده‌اند سپاسگزاری می‌شود.

مراجع

[۱] R.K. Ahuja, J.B. Orlin, and A. Tiwari," A greedy genetic algorithm for the quadratic assignment problem ", Computers and Operations Research ۲۷, ۹۱۷-۹۳٤. ۲۰۰۰.

[۲] Christo"des N, Benavent E. An exact algorithm for the quadratic assignment problem. Oper. Res. ۱۹۸۹;۳۷:۷٦۰-۸.

[۳] Bazara MS, Sherali MD. Benders' partitioning scheme applied to a new formulation of the quadratic assignment problem. Naval Res. Logist. Quart. ۱۹۸۰;۲۷:۲۹-٤۱.

[٤] Lawler EL. The quadratic assignment problem. Manag. Sci. ۱۹٦۳;۹:۵۸٦-۹۹.

[۵] Pardalos PM, Crouse J. A parallel algorithm for the quadratic assignment problem. Proceedings of the Supercomputing ۱۹۸۹ Conference, ACM Press. New York, ۱۹۸۹, pp. ۳۵۱-٦۰.

[٦] Armour GC, Bu!a ES. Heuristic algorithm and simulation approach to relative location of facilities. Manag. Sci.۱۹٦۳;۹:۲۹٤ -۳۰۹.

[۷] Bula ES, Armour GC, Vollmann TE. Allocating facilities with CRAFT. Harvard Business Rev. ۱۹٦۲;٤۲:۱۳٦-۵۸.

[۸] West DH. Algorithm ٦۰۸: approximate solution of the quadratic assignment problem.ACMTrans. Math. Software۱۹۸۳;۹:٤٦۱-٦.

[۹] Burkard RE, Bonniger T. A heuristic for quadratic boolean programs with applications to quadratic assignment problems. European J. Oper. Res. ۱۹۸۳;۱۳:۳۷٤-۸٦.

[۱۰] Li T, Pardalos PM, Resende MGC. A greedy randomized adaptive search procedure for the quadratic assignment problem. In: Pardalos PM, Wolkowicz H (Eds.), Quadratic Assignment and Related Problems. DIMACS Series in Discrete Mathematics and Theoretical Computer Science. Providence. RI: Americal Mathematical Society, ۱۹۹٤, pp. ۲۳۷-۲٦۱.

[۱۱] Wilhelm MR, Ward TL. Solving quadratic assignment problems by simulated annealing. IEEE Trans.۱۹۸۷;۱۹:۱۰۷-۱۹.

[۱۲] Skorin-Kapov J. Tabu search applied to the quadratic assignment problem. ORSA J. Comput. ۱۹۹۰;۲:۳۳-٤۵. R.K. Ahuja et al. Computers & Operations Research ۲۷ (۲۰۰۰) ۹۱۷-۹۳٤.

[۱۳] Taillard E. Robust tabu search for the quadratic assignment problem. Parallel Comput. ۱۹۹۱;۱۷:٤٤۳-۵۵.

[۱٤] Fleurent C, Ferland JA. Genetic hybrids for the quadratic assignment problem. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. ۱٦, Providence. RI: American Mathematical Society, ۱۹۹٤, pp. ۱۷۳-۸۷.

[۱۵] Resende MGC, Pardalos PM, Li Y. Fortran subroutines for approximate solution of dense quadratic assignment problems using GRASP. ACM Trans. Math. Software ۱۹۹٤; ۲۲:۱۰٤-۱۸.

[۱٦] Stutzle, T., Dorigo, M., ۱۹۹۹. ACO algorithms for the quadratic assignment problem. In: Corne, D., Dorigo, M., Glover, F. (Eds.), New Ideas for Optimization. McGraw-Hill, pp. ۳۳_۵۰.

[۱۷] Gambardella, L., Taillard, E., Dorigo, M., ۱۹۹۷. Ant Colonies for the QAP, Tech. Report IDSIA-٤-۹۷, IDSIA, Lugano, Switzerland.

[۱۸] James, T., Rego, C., Glover, F., ۲۰۰۵. Sequential and parallel pathrelinking algorithms for the quadratic assignment problem. IEEE Intelligent Systems ۲۰ (٤), ۵۸_٦۵.

[۱۹] Oliveira, C.A., Pardalos, P.M., Resende, M.G.C., ۲۰۰٤. GRASP with pathrelinking for the quadratic assignment problem, Efficient and Experimental Algorithms. In: Ribeiro, C.C., Martins, S.L. (Eds.), . In: Lecture Notes in Computer Science, vol. ۳۰۵۹. Springer-Verlag, pp. ۳۵٦_۳٦۸.

[۲۰] Tseng, L., Rego, C., Glover, F.,۲۰۰۹. A cooperative parallel tabu search algorithm for the quadratic assignment problem. European Journal of Operational Research ۱۹۵, ۸۱۰_۸۲٦.James, T., Liang, S., ۲۰۰۵. A hybrid metaheuristic for the quadratic assignment problem. Computational Optimization and Applications ۳٤, ۸۵_۱۱۳.

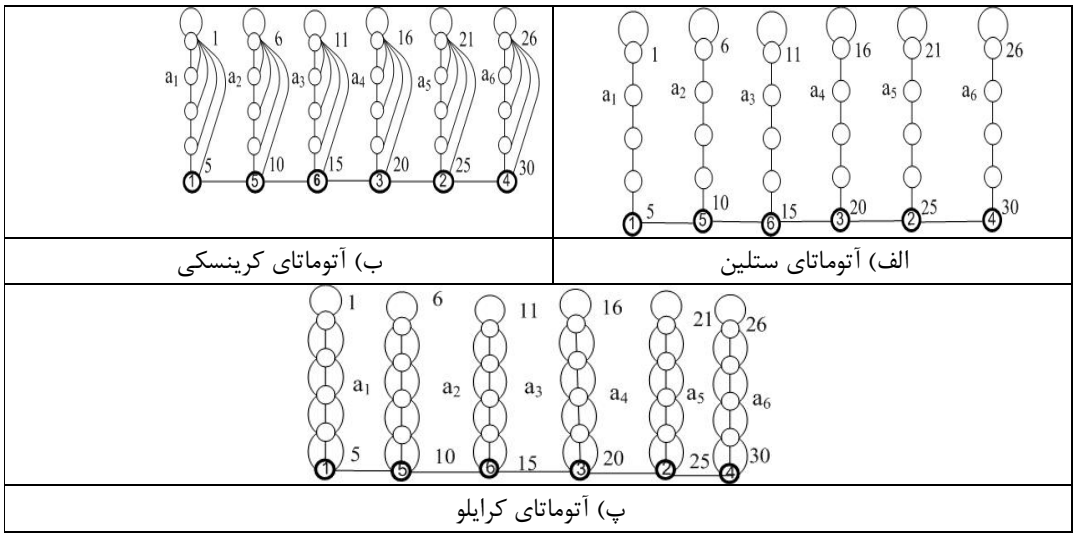
[۲۱] Atashpaz-Gargari, E., Lucas, C., ۲۰۰۷. "Imperialist Competitive Algorithm: An algorithm for optimization inspired by imperialistic competition". IEEE Congress on Evolutionary Computation, pp.٤٦٦۱_٤٦٦۷.

[۲۲] K. Narendra and M. A. L. Thathachar, Learning Automata: An Introduction, Prentice Hall, ۱۹۸۹.

[۲۳] K. Najim and A. S. Poznyak, Learning Automata: Theory and Application, Elsevier Science Ltd., ۱۹۹٤.

[۲٤] B. J. Oommen and D. C. Y. Ma, "Deterministic Learning Automata Solutions to the equipartitioning problem", IEEE Traactions on Computers, Vol. ۳۷ , pp. ۲-۱۳, ۱۹۹۸.

[۲۵] Burkard RE, Karisch SE, Rendl F. QAPLIB - A quadratic assignment program library. J. Global Optim.۱۹۹۷; ۱۰:۳۹۱-٤۰۳.



شکل ۸: نمایش تخصیص [۴، ۲، ۳، ۶ ، ۵،۱] بوسیله آتوماتون‌های یادگیر با اتصالهای مشابه ستلین، کرینسکی و کرایلو

جدول ۱: مقایسه کارایی الگوریتم‌ها مطابق با اندازه مسائل

			ICA		*ICA-LA$_{Tsetline}$*		*ICA-LA$_{Krinsky}$*		*ICA-LA$_{Krylov}$*		*SA*	
Problem	***n***	***BKS***	***Solution***	***Gap(%)***	***Solution***	***Gap(%)***	***Solution***	***Gap(%)***	***Solution***	***Gap(%)***	***Solution***	***Gap(%)***
TAI۱۰A	۱۰	۱۳۵۰۲۸	۱۳۵۶۴۰	۰.۴۵۳	۱۳۵۰۲۸	۰	۱۳۵۰۲۸	۰	۱۳۵۰۲۸	۰	۱۳۵۳۲۸	۰.۲۲۲
CHR۱۲A	۱۲	۹۵۵۲	۱۰۴۷۶	۹.۶۷۳	۹۵۵۲	۰	۹۵۵۲	۰	۹۵۵۲	۰	۹۵۵۲	۰
CHR۱۸A	۱۸	۱۱۰۹۸	۱۵۰۲۴	۳۵.۳۷۵	۱۲۲۸۰	۱۰.۶۵۰	۱۳۲۲۴	۱۹.۱۵۶	۱۲۹۳۸	۱۶.۵۷۹	۱۲۴۵۱	۱۲.۱۹۱
CHR۲۵A	۲۵	۳۷۹۶	۵۶۷۸	۴۹.۵۷۸	۴۶۷۴	۲۳.۱۲۹	۴۷۹۰	۲۶.۱۸۵	۴۹۱۴	۲۹.۴۵۲	۴۶۹۰	۲۳.۵۵۱
BUR۲۶A	۲۶	۵۴۲۶۶۷۰	۵۴۴۲۲۵۹	۰.۲۸۷	۵۴۳۳۰۰۶	۰.۱۱۶	۵۴۳۴۷۳۰	۰.۱۴۸	۵۴۳۳۰۱۸	۰.۱۱۶	۵۴۳۹۹۴۰	۰.۲۴۴
TAI۳۰A	۳۰	۱۸۱۸۱۴۶	۱۹۲۸۰۱۸	۶.۰۴۳	۱۸۵۵۷۱۰	۲.۰۶۶	۱۸۶۶۰۶۴	۲.۶۳۵	۱۸۸۴۸۳۶	۳.۶۶۸	۱۸۷۰۷۱۰	۲.۸۹۱
THO۴۰	۴۰	۳۱۳۹۳۷۰	۳۱۵۹۶۶۸	۰.۶۴۶	۳۱۴۰۷۵۵	۰.۰۴۴	۳۱۴۳۰۶۶	۰.۱۱۷	۳۱۴۵۱۴۰	۰.۱۸۳	۳۱۶۲۵۴۱	۰.۷۳۸
WIL۱۰۰	۱۰۰	۲۷۳۰۳۸	۲۹۰۹۷۶	۶.۵۶۹	۲۸۳۳۶۲	۳.۷۸۱	۲۸۴۴۳۰	۴.۱۷۲	۲۸۶۹۹۶	۵.۱۱۲	۲۹۵۵۴۳	۸.۲۴۲
THO۱۵۰	۱۵۰	۸۱۳۳۳۹۸	۹۱۸۷۲۲۶	۱۲.۹۵۶	۸۸۱۰۷۹۴	۸.۳۲۸	۸۷۸۱۰۵۲	۷.۹۶۲	۸۹۸۸۲۳۴	۱۰.۵۱۰	۸۹۳۲۴۵۶	۹.۸۲۴

[۱] Quadratic Assignment Problem
[۲] Imperialist Competitve Algorithm
[۳] Campus Planning Problem
۴ Planet layout Problem
۵ Backboard wiring problem
۶ Design of control panels and typewriter
۷ Ordering of interrelated data
۸ Cutting Plane
۹ Construction methods
۱۰ Limited enumeration
۱۱ local improvement
۱۲Greedy Randomized Adaptive Search Procedure(GRASP)
۱۳ Path Relinking
۱۴ Imperialist
۱۵ Colony
۱۶ Assimilation
۱۷ Imperialist Competitive
[۱۸] Object Migrating Automata (OMA)
[۱۹] Best Known Solution