کاربرد یک الگوریتم اصلاحی رقابت استعماری برای حل مسأله-ی فروشنده دوره‌گرد

مجید یوسفی خوشبخت*[1]، فرزاد دیده‌ور**، فرهاد رحمتی**

* باشگاه پژوهشگران جوان، دانشگاه آزاد اسلامی، واحد همدان،

**دانشکده ریاضی و علوم کامپیوتر، دانشگاه صنعتی امیرکبیر تهران

تاریخ دریافت: ۱۳۹۰/۸/۲۳ تاریخ پذیرش: ۱۳۹۱/۵/۴

چکیده: این مقاله یک روش رقابت استعماری اصلاح‌شده را برای حل مسأله فروشنده دوره‌گرد ارائه می‌کند که در تابع جذب بین کشورهای استعمارگر و استعمارشده و هم‌چنین انقلاب کشورهای مستعمره، با حالت معمولی خود تفاوت دارد. به علاوه برای افزایش کارایی الگوریتم از روش بهبود-دهنده‌ی سه‌گانه استفاده می‌شود. الگوریتم جدید روی ۱۹ مثال استاندارد مسأله فروشنده دوره‌گرد از کتابخانه TSPLIB مورد آزمایش و با الگوریتم‌های رقابت استعماری، ژنتیک، پرندگان، تکاملی و کلونی زنبور مورد مقایسه قرار گرفت. نتایج محاسباتی نشان می‌دهد که الگوریتم پیشنهادی دارای کارایی مناسبی می‌باشد.

واژه‌های کلیدی: مسأله فروشنده دوره‌گرد، الگوریتم رقابت استعماری، مسائل NP -سخت.

رده‌بندی ریاضی: ۶۸J۲۰ و ۶۵Y۲۰

۱- مقدمه

مسأله فروشنده دوره‌گرد[2] (TSP) یکی از مهم‌ترین مسائل در بهینه‌سازی ترکیباتی است که در بسیاری از علوم مهندسی مورد استفاده قرار می‌گیرد و توجه بسیاری از دانشمندان و محققین را به خود جلب کرده است. در این مسأله فروشنده‌ای از گره‌ای دلخواه به‌نام انبار شروع به حرکت می‌کند و بعد از ملاقات کردن n مشتری به محل شروع باز می‌گردد به شرط آن‌که هر مشتری فقط یک‌بار مورد ملاقات قرار گیرد. هدف در این مسأله تعیین مسیری با هزینه کمینه

۱- آدرس الکترونیکی نویسنده مسئول مقاله: مجید یوسفی خوشبخت khoshbakht@iauh.ac.ir

2- Traveling Salesmen Problem (TSP)

برای فروشنده است. در این مسأله تعداد جواب‌های شدنی برای n گره بدون در نظر گرفتن انبار، برابر تعداد جای‌گشت‌هایی است که از اعداد ۱ تا n تشکیل شده است. هدف مسأله یافتن جای‌گشتی با کمترین هزینه جابجایی است. چون تعداد جواب‌های شدنی این مسأله برابر با $n!$ است، با افزایش گره‌های مسأله تعداد جواب‌های شدنی به شدت رشد می‌کند و دیگر نمی‌توان به راحتی و در یک زمان قابل قبول تمامی جواب‌ها را مقایسه کرد و به جواب بهینه دست‌یافت. به‌علاوه، TSP در بسیاری از مسائل کاربردی استفاده می‌شود که به‌طور مثال می‌توان به موارد زمان‌بندی وسیله نقلیه[1] [۱]، زمان‌بندی خدمه‌های هواپیما[2] [۲]، مسأله پستچی چینی ترکیبی[3] [۳]، ترتیب‌دهی کارگاه[4] [۴]، طراحی مأموریت برای ربات‌های متحرک مستقل[5] [۵] و اتصال کامپیوترها[6] [۶] اشاره کرد. البته کاربردهای این مسأله به موارد ذکر شده محدود نمی‌شود و می‌توان برای یافتن اطلاعات کامل به مرجع [۷] مراجعه کرد.

در بخش ۲ این مقاله، مدل مسأله مورد بررسی قرار می‌گیرد. در بخش ۳، تاریخچه مسأله به‌طور کامل شرح داده می‌شود. در بخش ۴، ابتدا به توصیف روش رقابت استعماری[7] (ICA) پرداخته و سپس روش رقابت استعماری اصلاحی[8] (MICA) برای حل مسأله TSP مورد تجزیه تجزیه و تحلیل قرار می‌گیرد. نتایج محاسباتی که بر روی مثال‌های استاندارد اجرا شده است، در بخش ۵ نشان داده می‌شوند و در انتها نتیجه‌گیری و جهت‌گیری‌های آینده در بخش ۶ ارائه می‌گردد.

۲- مدل مسأله

مسأله TSP را می‌توان به‌صورت یک گراف بدون جهت کامل $G = (V , A)$ در نظر گرفت که در آن $V = \{0,1,...,n\}$ نشان‌دهنده مجموعه گره‌ها و $A = \{(i , j) : i , j \in V , i \neq j\}$ مجموعه کمان‌ها است. اگر گراف کامل نبود آن‌گاه هر یال وجودنداشته، به‌وسیله یالی با هزینه مثبت بی‌نهایت جایگزین می‌شود. هم‌چنین اگر در جواب نهایی فروشنده به‌طور مستقیم از گره i به گره j حرکت کند $x_{ij} = 1$ و در غیر این‌صورت $x_{ij} = 0$ در نظر گرفته می‌شود. بنابراین اگر

1-Vehicle Scheduling Problem (VSP)
2- Crew Sequencing
3- Mixed Chinese Postman Problems
4- Workshop Scheduling
5- Mission Planning for Autonomous mobile Robots
6- Computer Wiring
7- Imperialist Competitive algorithm (ICA)
8- Modified Imperialist Competitive algorithm (MICA)

$C = [c_{ij}]$ نشان‌دهنده ماتریس متقارن هزینه روی گراف G باشد که در نامساوی مثلثی صدق می‌کند، آن‌گاه مدل برنامه‌ریزی خطی-صحیح مسأله TSP عبارت است از:

$$min \sum_{i=0}^{n}\sum_{j=0}^{n} c_{ij}x_{ij}$$

$$\sum_{i=0}^{n} x_{ij} = 1 \qquad j = 0,1,...,n; \tag{1}$$

$$\sum_{j=0}^{n} x_{ij} = 1 \qquad i = 0,1,...,n; \tag{2}$$

$$\sum_{i,j\in S} x_{ij} \le |S| - 1, (S \subset \{0,1,...,n\},\ |S| \ge 2); \tag{3}$$

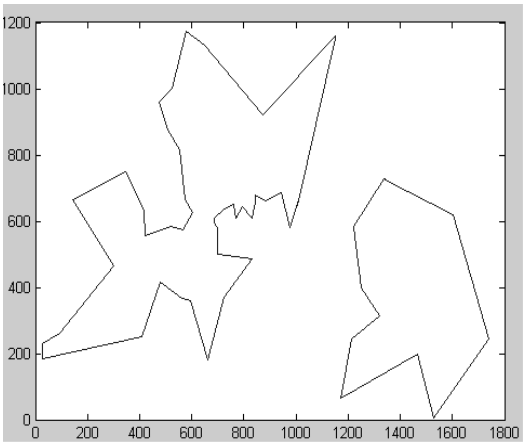
$$x_{ij} \in \{0,1\} \qquad i,j = 0,1,...,n. \tag{4}$$

در این مدل، دسته محدودیت اول نشان می‌دهد که به هر گره j فقط یک یال وارد می‌شود. هم‌چنین دسته محدودیت دوم به این نکته اشاره دارد که از هر گره i فقط یک یال خارج می‌شود. محدودیت سوم سبب می‌گردد که الگوریتم جواب‌هایی را که در آن یک زیردور فاقد انبار تشکیل شده است، به‌عنوان جواب قابل قبول در نظر نگیرد. باید توجه کرد که محدودیت اول و دوم سبب می‌گردند که به هر گره فقط یک یال وارد و نیز فقط یک یال خارج شود. بنابراین فقط حالتی مانند شکل ۱ می‌ماند که دوری بدون داشتن انبار تشکیل شود که در محدودیت‌های اول و دوم صدق می‌کند. اما چون این جواب در دسته محدودیت سوم صدق نمی‌کند، به‌عنوان یک جواب غیر قابل قبول معرفی می‌شود. در نهایت دسته محدودیت چهارم به شرایط دودویی x_{ij} اشاره می‌کند.

۳- تاریخچه مسأله

مانند اکثر مسائل بهینه‌سازی ترکیباتی، روش‌های حل TSP به دو دسته الگوریتم‌های دقیق و تخمینی تقسیم‌بندی می‌شوند. در الگوریتم‌های دقیق مانند تخفیف لاگرانژ [۸]، شاخه و کران [۹] و شبه‌تخصیص [۱۰] که بیشتر برای مسائل با اندازه تقریباً کوچک به‌کار می‌رود، جواب بهینه مسأله به‌دست می‌آید؛ علی‌رغم آن‌که زمان زیادی برای به‌دست آمدن این جواب باید صرف شود.

چون TSP یک مسأله NP-سخت[1] است [۱۱]، الگوریتم‌های دقیق کارایی خود را بر روی این مسأله در ابعاد بالا از دست می‌دهند و نمی‌توانند به جواب بهینه در یک زمان قابل قبول دست یابند. در نتیجه امروزه برای حل این‌گونه مسائل از الگوریتم‌های ابتکاری و فرا ابتکاری استفاده می‌شود که علی‌رغم این‌که نمی‌توانند جواب بهینه را به‌دست آورند اما می‌توانند جواب نزدیک به بهینه را در یک زمان قابل قبول به‌دست آورند. توجه به این نکته ضروری است که الگوریتم‌های ابتکاری در یک زمان اندک به جواب زیر بهینه می‌رسند اما چون دارای ساختار خوبی برای فرار از نقاط بهینه محلی نیستند، معمولاً به جواب‌های باکیفیت همگرا نمی‌شوند. به‌عنوان مثال می‌توان به روش کارپ [۱۲] اشاره کرد که از اولین الگوریتم‌ها برای حل این مسأله محسوب می‌شود.



شکل ۱: یک جواب غیر قابل قبول برای مساله TSP

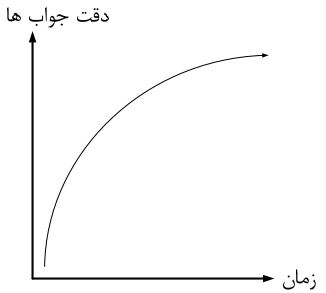
از طرف دیگر در چند دهه اخیر الگوریتم‌هایی پیشنهاد شدند که دارای ساختار تصادفی برای رسیدن به جواب مسأله هستند. این الگوریتم‌ها که فراابتکاری نامیده می‌شوند، می‌توانند تا حد امکان از بهینه‌های محلی فرار کنند و به جواب‌های بسیار خوبی همگرا شوند. به‌طور مثال برای این‌گونه روش‌ها می‌توان به شبکه‌های عصبی[2] [۱۳]، جستجوی ممنوع [۱۴]، ژنتیک[3] (GA) [۱۵]، ممتیک[4] [۱۶] و الگوریتم ICA [۱۷] اشاره کرد. علی‌رغم جستجوی تصادفی

1- NP-hard
2- Neural Networks
3- Genetic Algorithm (GA)
4- Memetic Algorithm

الگوریتم‌های فراابتکاری سبب می‌شود که فضای شدنی مسأله از جهات مختلف مورد بررسی قرار گیرد و در صورت افتادن الگوریتم در بهینه محلی، الگوریتم بتواند با استفاده از این ساختارها از بهینه محلی فرار کند ولی از طرف دیگر این ساختار سبب می‌شود که زمان این‌گونه الگوریتم‌ها نسبت به روش‌های ابتکاری افزایش یابد. باید اضافه کرد که در الگوریتم‌های فراابتکاری دقت جواب‌های به‌دست آمده و سرعت رسیدن به آن رابطه عکس با یکدیگر دارند و همان‌طور که در شکل ۲ نشان داده شده است، دارای نمودار محدبی هستند [۱۸]. همان‌طوری که از این شکل مشخص است سرعت پیشرفت دقت جواب‌ها در ابتدای الگوریتم بسیار خوب است و در یک زمان اندک رشد کیفیت جواب‌ها بسیار مطلوب می‌باشد ولی هر چه از شروع الگوریتم می‌گذرد، این مطلوبیت کاهش می‌یابد و سبب می‌شود که کیفیت جواب‌ها تقریباً ثابت بماند. بنابراین به‌علت حجم بسیار زیاد محاسبات، متغیرها و پارامترها تعداد تکرار الگوریتم به‌شدت افزایش می‌یابد و الگوریتم نمی‌تواند با سرعت مناسب کیفیت جواب‌ها را ارتقا دهد.

در الگوریتم‌های ترکیبی که از ترکیب الگوریتم‌های فراابتکاری و ابتکاری تشکیل شده‌اند، سعی شده است که از مزایای هر کدام از الگوریتم‌ها به‌نحو مطلوبی استفاده شود. به‌طور مثال الگوریتم‌های فراابتکاری دارای قابلیت خوبی برای جستجوی سراسری هستند و می‌توانند فضای جواب را در یک زمان قابل قبول مورد جستجو قرار دهند و مناطق مستعد را شناسایی کنند. بنابراین می‌توان در الگوریتم‌های ترکیبی از این روش‌ها برای جستجوی سراسری و یافتن جواب‌های خوب استفاده کرد. از طرف دیگر روش‌های ابتکاری دارای کارایی خوبی برای جستجوی محلی هستند و می‌توانند همسایگی‌های یک جواب را به‌طور مناسب مورد بررسی قرار دهند. بنابراین می‌توان در بعضی از الگوریتم‌های ترکیبی از این روش‌ها در مواقعی استفاده نمود که الگوریتم توانسته است جواب‌های باکیفیتی را شناسایی کند ولی برای یافتن جواب‌های بهتر در همسایگی‌های این جواب‌ها احتیاج به جستجوی محلی دارد. به‌عنوان نمونه‌ای از این روش‌ها، که امروزه توجه بسیار زیادی را به‌خود جذب کرده است، می‌توان به الگوریتم‌های ترکیبی جاروب[۱]، GA و نزدیک‌ترین جمعی[۲] [۱۹]، الگوریتم مورچگان[۳] و جستجوی پراکنده[۴] [۲۰]، روش پرندگان[۵] (PSO) و جستجوی محلی [۲۱] اشاره کرد.

1- Sweep Algorithm
2- Nearest Addition Algorithm
3- Ant Colony Algorithm
4- Scatter Search
5- Particle Swarm Optimization (PSO)



شکل ۲: رابطه زمان و دقت جواب‌ها در الگوریتم‌های فراابتکاری

بنا به دلایل ذکر شده، در این مقاله از ترکیب روش‌های رقابت استعماری و جستجوی محلی سه‌گانه[1] (MICA) برای حل مسأله TSP استفاده و از نقاط قوت آن‌ها برای رسیدن به جوابی مناسب بهره گرفته می‌شود. این ترکیب، همان‌طور که در بخش‌های بعدی نشان داده شده است، سبب می‌شود که الگوریتم بتواند به هدف خود یعنی یافتن یک جواب با کیفیت در یک زمان مناسب دست‌یابد.

۴- الگوریتم پیشنهادی

الگوریتم ICA یکی از روش‌های جدید فراابتکاری است که توسط آتش‌پز و لوکاس [۲۲] ابداع شد. اگر چه تنها پنج سال از ابداع این الگوریتم می‌گذرد، اما تاکنون در مسائل زیادی مانند طراحی ساختار اسکلت [۲۳]، خوشه‌بندی داده‌ها [۲۴]، موتور القایی خطی [۲۵]، تعادل نش [۲۶] و غیره استفاده شده است. دلایل استقبال زیاد از این الگوریتم علاوه بر کارایی مناسب، سرعت همگرایی و توانایی بهینه سازی بالا در مقایسه با الگوریتم‌های موجود [۲۷]، بیشتر به جهت نوآوری و جذاب بودن آن برای متخصصین حوزه بهینه‌سازی است. در این بخش ایده اصلی روش ICA ابتدا در زیربخش اول مورد بررسی قرار می‌گیرد و سپس اصلاح و ترکیب پیشنهادی در زیربخش دوم ارائه می‌گردد.

۴-۱ الگوریتم رقابت استعماری

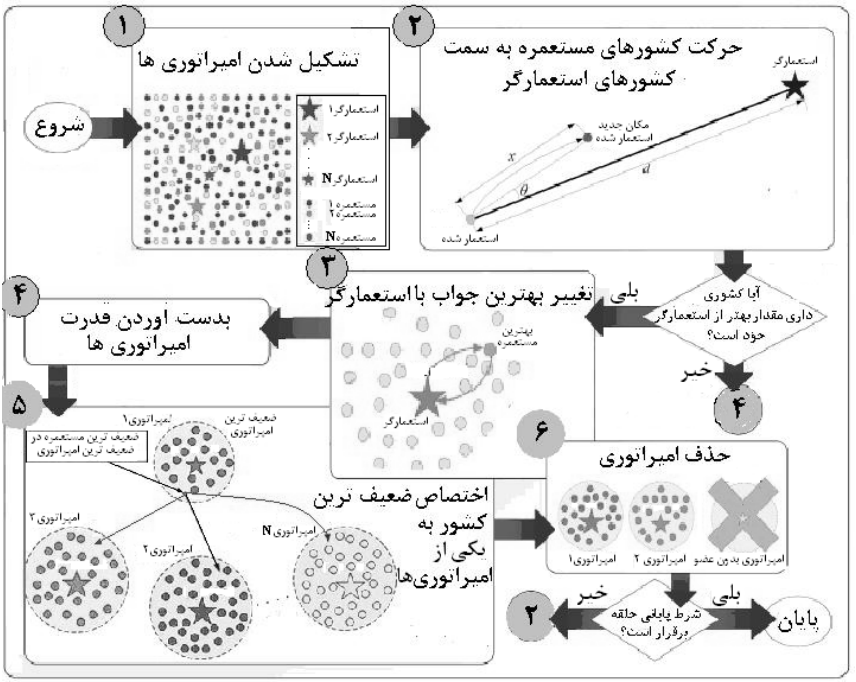
الگوریتم ICA که مانند روش‌های PSO و GA برای رسیدن به جواب از سیستم‌های تکاملی استفاده می‌کند، از نظر قابلیت تعمیم‌پذیری در مسائل بسیار موفق بوده است و در بسیاری از کاربردها ظاهر می‌شود. در این مسائل با توجه به این‌که الگوریتم اطلاعات اندکی مانند فضای

1- Modified Imperialist Competitive Algorithm (MICA)

جستجو و تعریف جواب شدنی برای مسأله اصلی دارد، می‌تواند با ایجاد جواب‌های تصادفی در مسیر یافتن جواب‌های بهتر حرکت کند و در فضای جستجو تا حد امکان به خوبی پیشروی نماید. الگوریتم کلی روش ICA به این روال است که در مرحله اول با تعدادی جمعیت اولیه، که در این الگوریتم کشور نامیده می‌شود، شروع می‌شود و سپس مقدار تابع هدف طبق مسأله مورد بررسی برای هر کدام از کشورها به‌دست می‌آید. با مقایسه مقادیر تابع هدف همه کشورها، تعدادی از بهترین آن‌ها انتخاب و کشورهای استعمارگر نامیده می‌شوند. سپس بقیه کشورها، کشورهای مستعمره نامیده می‌شوند. باید توجه کرد که در شرایط مساوی که چندین مقدار تابع هدف یکسان برای انتخاب کشورهای استعمارگر وجود دارد به تصادف از بین آن‌ها، کشورهای استعمارگر انتخاب می‌شوند. حال، کشورهای مستعمره به کشورهای استعمارگر اختصاص می‌یابند تا امپراتوری‌ها (هر امپراتوری از یک کشور استعمارگر و چند کشور مستعمره تشکیل شده است) تشکیل شوند. باید توجه کرد که هر چه قدرت یک کشور استعمارگر بیشتر باشد (تابع هدف آن بهتر باشد) تعداد بیشتری کشورهای مستعمره به آن اختصاص می‌یابد. به‌طور مثال در شکل ۳ نمونه‌ای از این تقسیم‌بندی نشان داده شده است. در این شکل ستاره‌های ۱, ۲, ..., N کشورهای استعمارگر را نشان می‌دهد؛ در حالی که دایره‌ها نشان‌دهنده کشورهای استعمارشده توسط هر کدام از کشورهای استعمارگر است. در اینجا کشور استعمارگر قوی‌تر که با ستاره بزرگ‌تر نشان داده شده است، دارای بیشترین تعداد کشور مستعمره با دایره‌های هم‌رنگ است و کشور استعمارگر از همه ضعیف‌تر که با ستاره کوچک‌تر نمایان است دارای کمترین کشورهای تحت نفوذ می‌باشد.

در مرحله دوم کشورهای استعمارگر سعی در وابسته کردن کشور مستعمره با دگرگونی فرهنگ و رسوم آن کشور می‌کنند که تابع جذب نامیده می‌شود. این عمل سبب می‌گردد که کشور مستعمره در جهت همسان‌شدن فرهنگی با کشور استعمارگر حرکت کند. به‌عبارت دیگر هر کدام از کشورهای مستعمره، به ترتیب با استفاده از یک روال معین به کشور استعمارگر نزدیک می‌شوند. در مرحله سوم براساس روال طبیعی، بعضی از کشورهای مستعمره ممکن است انقلاب کنند و بتوانند قدرت امپراتوری را در دست بگیرند. در نتیجه ρ درصد از کشورهای مستعمره انتخاب و دچار انقلاب می‌شوند. بعد از این‌که توابع جذب و انقلاب انجام شد، تابع هدف برای هر کدام از کشورهای مستعمره دوباره محاسبه می‌شود. حال، برای هر کدام از امپراتوری‌ها اگر بهترین تابع هدف کشورهای مستعمره از تابع هدف کشور استعمارگر مقدار بهتری داشت، آن‌گاه جای آن دو کشور باهم عوض می‌شود. در ضمن، بعد از انجام این عمل برای تمامی امپراتوری‌ها بهترین جواب و مقدار همه کشورهای استعمارگر به‌عنوان بهترین جواب و مقدار جاری در این تکرار الگوریتم ذخیره می‌شود. حال، اگر مقدار تابع هدف جواب

جاری نسبت به بهترین مقداری که تاکنون به‌دست آمده دارای کیفیت بهتری باشد، آن‌گاه جواب و مقدار جدید جایگزین جواب قبلی می‌شود. از طرف دیگر باید توجه کرد که بعد از خاتمه الگوریتم، این جواب و مقدار به‌عنوان جواب مساله ارائه می‌شود.



شکل ۳: روند الگوریتم ICA

در مرحله چهارم قدرت کل یک امپراتوری به‌صورت مجموع تابع هدف کشور استعمارگر به اضافه ضریبی مانند $0 \leq \lambda \leq 1$ از میانگین توابع هدف مستعمرات آن تعریف می‌شود. توجه به این نکته ضروری است که این ضریب سبب می‌گردد که کاربر بتواند تأثیر اهمیت کشور استعمارگر یا میانگین کشورهای مستعمره را در قدرت نهایی هر امپراتوری معین کند. هم‌چنین طی رقابت استعماری که براساس مقایسه قدرت امپراتوری‌ها صورت می‌گیرد، ضعیف‌ترین عضو از ضعیف‌ترین امپراتوری انتخاب شده و به یکی از امپراتوری‌ها انتقال می‌یابد. باید توجه کرد که اگر امپراتوری دارای هیچ کشور مستعمره نباشد، آن‌گاه کشور استعمارگر آن به کشوری مستعمره تبدیل می‌شود و به یکی از امپراتوری‌ها انتقال می‌یابد. نکته بسیار مهم در مرحله چهارم در این است که انتقال ضعیف‌ترین عضو از ضعیف‌ترین امپراتوری به امپراتوری دیگر به‌صورت احتمالی صورت می‌گیرد و هرچه امپراتوری دارای قدرت بیشتری باشد به‌احتمال

بیشتری صاحب آن می‌شود. بنابراین در طی مراحل الگوریتم، امپراتوری‌های ضعیف که دارای قدرت کمتری نسبت به سایر امپراتوری‌ها هستند به تدریج قدرت خود را از دست می‌دهند و به‌مرور زمان از بین می‌روند. بعد از این مرحله شرایط پایانی الگوریتم مورد آزمایش قرار می‌گیرد، اگر شرایط پایانی، که یکی از آن‌ها می‌تواند باقی ماندن فقط یک امپراتوری باشد، برقرار شوند، الگوریتم به پایان می‌رسد؛ در غیر این‌صورت الگوریتم به مرحله دوم بر می‌گردد.

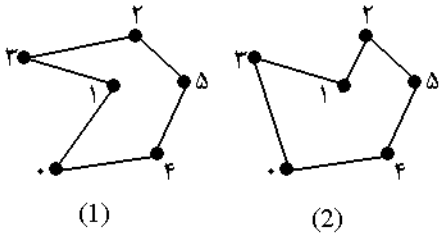
۲-۴ الگوریتم رقابت استعماری اصلاحی

چون ICA یک روش تکاملی است، پس در ابتدا باید به‌وسیله تعدادی کشور (جمعیت اولیه) شروع به‌کار کند. بنابراین p جمعیت اولیه تصادفی برای مسأله ایجاد می‌شود و مقادیر تابع هدف f_i به‌ازای هر کشور $i = ۱,۲,...,p$ برای مسأله TSP به‌دست می‌آید. حال، $m < p$ کشور که دارای مقادیر تابع هدف کمتر برای مسأله مربوطه هستند، به‌عنوان کشورهای استعمارگر در نظر گرفته می‌شوند و با جابجایی، اندیس‌های ۱ تا m جمعیت اولیه را تشکیل می‌دهند. سپس به‌هر کدام از کشورهای استعمارگر، تعدادی مساوی از کشورهای مستعمره اختصاص می‌یابند (با قسمت صحیح تقسیم p-m بر m). به‌علاوه، به‌علت خاصیت تابع جزء-صحیح، بقیه کشورها نیز به قوی‌ترین امپراتوری اختصاص می‌یابند. باید توجه داشت که این فرمول فقط تعداد کشورهای مستعمره اختصاص یافته به‌هر کشور استعمارگر را مشخص می‌کند اما اختصاص یافتن هر کشور مستعمره به کشور استعمارگر به‌طور تصادفی و با احتمال برابر است.

طبق حالت طبیعی که وجود دارد باید کشورهای مستعمره از لحاظ فرهنگی و اجتماعی به-سمت کشورهای استعمارگر با استفاده از تابع جذب حرکت کنند. بنابراین در این الگوریتم، از روش نزدیک‌ترین همسایه تصادفی برای تابع جذب استفاده می‌شود [۲۸]. به‌طور مثال اگر در این روش [۴ ۲ ۵ ۳ ۱] نشان‌دهنده کشور مستعمره و [۵ ۲ ۱ ۴ ۳] نشان‌دهنده کشور استعمارگر برای مسأله TSP با ۵ گره باشد، آن‌گاه الگوریتم از اولین گره کشور مستعمره که ۱ است، شروع به حرکت می‌کند. سپس $i = ۱$ در نظر گرفته می‌شود و همسایه‌های ملاقات‌نشده ۱ در دو کشور با گره‌های ۳، ۴ و ۲، را در مجموعه S قرار می‌دهد. حال، اگر c_{ij} نشان‌دهنده فاصله بین دو گره i و j باشد، آن‌گاه گره‌های متعلق به S به احتمال V_j در فرمول (۵) مورد ملاقات قرار می‌گیرند.

$$V_j = \frac{1/c_{ij}}{\sum_{j \in S}(1/c_{ij})} \qquad \forall j \in S. \tag{۵}$$

فرض کنیم گره ۲ برگزیده شود؛ پس در جمعیت اولیه تاکنون [- - - ۲ ۱] به‌وجود می‌آید. به‌علاوه برای ۲، سه گره ۱ ،۴ و ۵ به عنوان همسایه محسوب می‌شوند. اما چون قبلاً ۱ مورد ملاقات قرار گرفته شده است، از بین ۴ و ۵ که مجموعه S را تشکیل می‌دهند، طبق فرمول (۱) نزدیک‌ترین همسایه تصادفی انتخاب می‌گردد (باید توجه کرد که اگر مجموعه S در مرحله‌ای تهی شد، آن‌گاه گره‌های تاکنون ملاقات نشده در آن قرار می‌گیرند). این روش سبب می‌گردد که به علت ممنوع شدن انتخاب کشورهای قبلاً ملاقات‌شده، حتماً یک جواب شدنی برای مسأله به‌دست آید. باید توجه کرد که جواب به‌دست آمده در صورتی جایگزین جواب کشور مستعمره می‌شود که دارای مقدار تابع هدف بهتر باشد. از طرف دیگر استفاده از این روش سبب می‌گردد که علاوه بر این‌که کیفیت مقادیر تابع هدف کشورهای مستعمره طبق این روال افزایش یابد، تنوع کشورهای مستعمره به علت ساختار تصادفی تابع جذب حفظ شود.



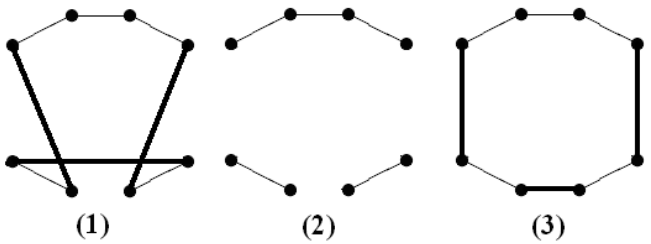
شکل ۴: حرکت بهبوددهنده دو نقطه‌ای

بعد از این‌که تابع جذب برای همه کشورهای مستعمره نسبت به کشورهای استعمارگر انجام شد، ρ درصد از کشورهای مستعمره دچار انقلاب می‌شوند. برای این عمل از روش بهبوددهنده دو نقطه‌ای[1] استفاده می‌شود. این روش همان‌طور که در شکل ۴ نشان داده شده است، براساس انتخاب دو گره از مساله و تعویض آن‌ها با هم کار می‌کند. باید توجه کرد که ρ درصد کشورهای هر امپراتوری به‌تصادف انتخاب و این روش روی آن‌ها انجام می‌شود. حال، جواب‌های جدید به‌دست آمده از انقلاب به همراه کشورهای مستعمره امپراتوری jام در نظر گرفته می‌شوند و بهترین جواب‌ها به‌عنوان کشورهای مستعمره امپراتوری jام انتخاب می‌شوند. بعد از این‌که مقدار تابع هدف برای تمامی کشورهای مستعمره به‌دست آمد، امکان دارد که این کشورها دارای تابع هدف بهتری نسبت به کشورهای استعمارگر امپراتوری خود شوند. بنابراین

1- Two-Point

بهترین کشورهای مستعمره در هر امپراتوری انتخاب شده و در صورت داشتن تابع هدف بهتر، جایگزین کشور استعمارگر می‌شوند.

توجه به این نکته ضروری است که در اجرای الگوریتم دو متغیر وجود دارند که مقادیر بهترین جواب و مقدار تابع هدف را ذخیره می‌کنند. این دو متغیر بعد از بروز شدن استعمارگرها بررسی می‌شوند. در این مرحله بهترین جواب و مقدار تابع هدف کشورهای استعمارگر انتخاب شده و الگوریتم بهبوددهنده سه‌گانه طبق شکل ۵ بر روی آن انجام می‌شود و سپس مقدار به‌دست آمده اگر از بهترین مقدار به‌دست آمده در تکرارهای قبلی بهتر باشد، آن جواب و مقدار تابع هدف جایگزین می‌شوند.



شکل ۵: حرکت بهبوددهنده سه‌گانه

روش بهبود دهنده سه‌گانه براساس حذف کردن سه یال از تور و دوباره متصل‌کردن آن سه یال از طریقی دیگر، کار می‌کند. باید توجه کرد که برای متصل‌کردن مسیر و ایجاد دوباره تور، چندین راه وجود دارد. اما فقط حالتی پذیرفته می‌شود که در محدودیت‌های مسأله صدق کند و تور جدید مقدار بهتری را برای مسأله به‌دست آورد. توجه به این نکته ضروری است که عمل حذف سه کمان و متصل‌کردن دوباره آن‌ها به‌طور متوالی تا آنجا ادامه می‌یابد که دیگر هیچ حرکت بهبوددهنده سه‌گانه یافت نشود.

در مرحله بعدی قدرت هر امپراتوری باید سنجیده شود تا بدین ترتیب مشخص گردد کدام امپراتوری از قدرت بیشتری برخوردار است. برای این منظور از فرمول زیر استفاده می‌شود:

$$w_j = f_j + \lambda(s_j) \qquad j = 1,...,m. \qquad (۶)$$

در این فرمول

w_j : قدرت کل امپراتوری jام است که شامل کشور استعمارگر و کشورهای مستعمره توسط آن کشور می‌باشد.

s_j : میانگین تابع هدف کشورهای مستعمره در هر امپراتوری است.

λ : ضریبی بین منفی یک و صفر است که به‌وسیله آن می‌توان تأثیر تابع هدف کشور استعمارگر را نسبت به میانگین تابع هدف کشورهای استعمارشده در قدرت یک امپراتوری تعیین کرد.

حال که قدرت امپراتوری‌ها مشخص شد، باید هر امپراتوری دارای تابع هدف بزرگ‌تر (چون مساله TSP مساله مینیمم‌سازی است) رو به زوال حرکت کند و قدرت خود را با از دست دادن مستعمرات خود از دست بدهد. بنابراین ضعیف‌ترین کشور از ضعیف‌ترین امپراتوری انتخاب شده و به یکی دیگر از امپراتوری‌ها ملحق می‌شود. باید توجه کرد که این الحاق همیشه به بهترین امپراتوری تعلق نمی‌گیرد، بلکه احتمال آن بر اساس فرمول (۷) برای امپراتوری jام به‌دست می‌آید.

$$\beta_j = \frac{1/w_j}{\sum_{j=1}^{m} 1/w_j} \qquad j = 1,...,m. \tag{۷}$$

بنابراین هر امپراتوری که دارای قدرت بیشتری باشد با احتمال بیشتری این کشور مستعمره را تصاحب می‌کند. به‌علاوه، در اینجا اگر ضعیف‌ترین امپراتوری هیچ مستعمره‌ای نداشته باشد، آن‌گاه آن امپراتوری حذف می‌شود و کشور استعمارگر به قوی‌ترین امپراتوری انتقال می‌یابد. در غیر این‌صورت الگوریتم دوباره تکرار می‌گردد (هر مستعمره به سمت کشور استعمارگر خود حرکت می‌کند) تا این‌که شرط پایانی حلقه برقرار گردد. برای شرایط پایانی حلقه در اینجا از دو شرط استفاده می‌شود که به‌طور هم‌زمان در پایان هر تکرار الگوریتم مورد بررسی قرار می‌گیرد. این دو شرط تکرار الگوریتم به تعدادی معین t بار و باقی ماندن فقط یک امپراتوری می‌باشند. حال، هر کدام از این دو شرط در هر زمان اتفاق بیفتد الگوریتم به پایان می‌رسد و بهترین جواب و مقداری که تاکنون به‌دست آمده است به‌عنوان بهترین جواب و مقدار معرفی می‌گردد.

مثال: برای درک بیشتر الگوریتم مثالی با ۵ گره (۰، ۰)، (۴، ۳)، (۳، ۴)، (۳، ۰) و (۰، ۴) و به‌ترتیب با شماره‌های ۱ تا ۵ را در نظر بگیرید. هدف یافتن ترتیبی برای این پنج گره است که بهترین جواب را برای مسأله TSP در یک تکرار الگوریتم به‌دست آورد. برای این منظور پارامترهای الگوریتم به صورت $P = 4$، $m = 2$، $\rho = 0/30$ و $\lambda = -0/2$ در نظر گرفته‌می‌شود.

با اجرای الگوریتم چهار جواب تصادفی به‌ترتیب اول تا چهارم برای مسأله TSP به‌صورت [۵ ۴ ۳ ۲ ۱]، [۵ ۲ ۴ ۳ ۱]، [۳ ۲ ۴ ۵ ۱] و [۳ ۵ ۲ ۴ ۱] به‌دست آمد که دارای مقدار تابع هدف ۱۹/۴۱، ۱۸/۵۸، ۱۸/۵۸و ۱۸/۲۹ بودند. در اجرای الگوریتم جواب‌های دوم و چهارم به‌

عنوان کشورهای استعمارگر به‌ترتیب دارای کشورهای مستعمره اول [۵ ۴ ۳ ۲ ۱] و سوم [۳ ۲ ۴ ۵ ۱] شدند. با اجرای تابع جذب برای این دو کشور مستعمره با کشورهای استعمارگر متناظر به‌ترتیب دو کشور جدید [۲ ۳ ۴ ۵ ۱] و [۴ ۲ ۳ ۵ ۱] با مقادیر ۱۹/۴۱ و ۱۴/۵۸ به‌دست آمد. از طرف دیگر با در نظر گرفتن $\rho = ۰/۳۰$ هیچ‌کدام از کشورهای جدید به‌دست آمده دچار انقلاب نشدند. حال، چون کشور سوم از جواب [۳ ۲ ۴ ۵ ۱] به جواب بهتر [۴ ۲ ۳ ۵ ۱] تبدیل شد، جای این دو با هم عوض می‌شود. از طرف دیگر همین جواب با مقدار ۱۴/۵۸ به‌عنوان کشور استعمارگر امپراتوری دوم و هم به‌عنوان بهترین جواب و مقدار تاکنون به‌دست آمده، ذخیره گشت. حال، طبق فرمول (۲) قدرت امپراتوری‌ها به‌دست آمد که امپراتوری اول دارای مقدار بیشتری از امپراتوری دوم شد و بدین ترتیب کشور مستعمره خود را به امپراتوری دوم داد (مسأله کمینه‌سازی است). بنابراین پس از تکرار اول، امپراتوری اول فقط شامل کشور استعمارگر [۵ ۴ ۲ ۳ ۱] و امپراتوری دوم شامل کشور استعمارگر [۴ ۲ ۳ ۵ ۱] و کشورهای مستعمره [۵ ۴ ۳ ۲ ۱] و [۳ ۵ ۲ ۴ ۱] شدند. از طرفی چون شرط پایانی الگوریتم که باقی‌ماندن یک امپراتور یا اجرای الگوریتم به‌تعداد $t = ۱۰$ برقرار نیست، الگوریتم به پایان نمی‌رسد و به ابتدای حلقه اصلی برمی‌گردد.

۵- محاسبات عددی

در این بخش ابتدا خصوصیات مثال‌های مورد آزمایش و نتایج الگوریتم پیشنهادی برای این مثال‌ها ارائه می‌شود و سپس نتایج به‌دست آمده به‌وسیله الگوریتم ICA با تعدادی از نتایج مشهورترین الگوریتم‌های فراابتکاری مورد مقایسه قرار می‌گیرد. باید اضافه کرد که تمام کدهای الگوریتم پیشنهادی به زبان *Matlab* ۷ نوشته شده است و کامپیوتری که این برنامه‌ها بر روی آن اجرا شده است از نوع پنتیوم ۴ با قدرت $۲.۴\ GHz$ و با دو گیگا بایت حافظه می‌باشد.

در جدول ۱، تعداد ۱۹ مثال استاندارد TSP در نظر گرفته شده است که دارای بازه بسیار خوبی از تعداد گره‌ها می‌باشد و مقادیری بین ۲۴ تا ۲۰۰ را اختیار می‌کند. این مثال‌ها در آدرس اینترنتی زیر وجود دارند:

http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp.

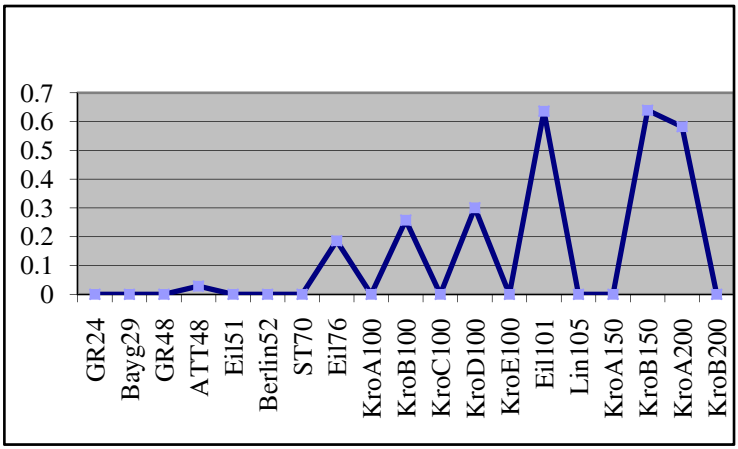
خصوصیات کامل مثال‌ها مانند تعداد گره‌های هر مثال (n) به‌همراه تعداد تکرارهای حلقه اصلی الگوریتم برای خاتمه روش (t)، تعداد اجرای مستقل الگوریتم برای یافتن بهترین مقدار (*tbest*)، بهترین مقادیر به‌دست آمده توسط الگوریتم پیشنهادی (MICA) و بهترین جواب‌های تاکنون به‌دست آمده (*Best*) در جدول ۱ نشان داده شده است.

جدول ۱: مشخصات مثال‌ها و نتیجه الگوریتم پیشنهادی

Best	*MICA*	*tbest*	*t*	*n*	مثال	*Best*	*MICA*	*tbest*	*t*	*n*	مثال
۲۰۷۴۹	۲۰۷۴۹	۱۰	۳۰۰	۱۰۰	KroC۱۰۰	۱۲۷۲	۱۲۷۲	۱۰	۷۲	۲۴	GR۲۴
۲۱۲۹۴	۲۱۳۵۸	۱۰	۳۰۰	۱۰۰	KroD۱۰۰	۱۶۱۰	۱۶۱۰	۱۰	۸۷	۲۹	Bayg۲۹
۲۲۰۶۸	۲۲۰۶۸	۱۰	۳۰۰	۱۰۰	KroE۱۰۰	۵۰۴۶	۵۰۴۶	۱۰	۱۴۶	۴۸	GR۴۸
۶۲۹	۶۳۳	۱۰	۳۰۳	۱۰۱	Eil۱۰۱	۱۰۶۲۸	۱۰۶۳۱	۱۰	۱۴۶	۴۸	ATT۴۸
۱۴۳۷۹	۱۴۳۷۹	۱۰	۳۱۵	۱۰۵	Lin۱۰۵	۴۲۶	۴۲۶	۱۰	۱۵۳	۵۱	Eil۵۱
۲۶۵۲۴	۲۶۵۲۴	۱۰	۴۵۰	۱۵۰	KroA۱۵۰	۷۵۴۲	۷۵۴۲	۱۰	۱۵۶	۵۲	Berlin۵۲
۲۶۱۳۰	۲۶۲۹۷	۱۰	۴۵۰	۱۵۰	KroB۱۵۰	۶۷۵	۶۷۵	۱۰	۲۱۰	۷۰	ST۷۰
۲۹۳۶۸	۲۹۵۳۹	۱۰	۶۰۰	۲۰۰	KroA۲۰۰	۵۳۸	۵۳۹	۱۰	۲۲۸	۷۶	Eil۷۶
۲۹۴۳۷	۲۹۴۳۷	۱۰	۶۰۰	۲۰۰	KroB۲۰۰	۲۱۲۸۲	۲۱۲۸۲	۱۰	۳۰۰	۱۰۰	KroA۱۰۰

از طرف دیگر برای این‌که روند الگوریتم ارائه شده در جدول ۱ برای TSP به‌طور واضح‌تر نشان داده شود در شکل ۶، درصد انحراف نسبت به بهترین جواب‌های تاکنون به‌دست آمده، آورده شده است. درصد انحراف نسبت به بهترین جواب شناخته‌شده از فرمول (۸) به‌دست می‌آید:

$$gap = ۱۰۰ \times \frac{[c(s^{*}) - c(s^{**})]}{c(s^{**})}. \qquad (۸)$$



شکل ۶: میزان درصد انحراف جواب‌های الگوریتم پیشنهادی نسبت به *Best*

در این فرمول s^* بهترین جواب پیداشده به‌وسیله الگوریتم مورد نظر و s^{**} بهترین جواب شناخته‌شده برای مثال مورد نظر (*Best*) است. باید توجه کرد که در این شکل محور افقی نشان‌دهنده ۱۹ مثال مورد آزمایش است در حالی‌که محور عمودی نشان‌دهنده *gap* الگوریتم MICA نسبت به بهترین جواب تاکنون به‌دست آمده است.

در جدول ۲ نتایج شش الگوریتم فراابتکاری برای این مثال‌ها آورده شده است. در این جدول هر کدام از الگوریتم‌ها از دو زیرستون تشکیل شده‌اند که به‌ترتیب از راست به چپ بهترین جواب به‌دست آمده توسط الگوریتم (s) و *gap* را نشان می‌دهد.

جدول ۲: مقایسه الگوریتم‌های فراابتکاری با الگوریتم پیشنهادی

مثال	*GA*		*EA*		*PSO*		*BCO*		*ICA*		*MICA*		*Best*
	s	*Gap*	*s*	*Gap*	*s*	*Gap*	*s*	*Gap*	*s*	*Gap*	*s*	*Gap*	
GR۲۴	۱۲۷۲	۰/۰۰	۱۲۷۲	۰/۰۰	-	-	-	-	-	-	۱۲۷۲	۰/۰۰	۱۲۷۲
Bayg۲۹	۱۶۱۰	۰/۰۰	۱۶۱۰	۰/۰۰	-	-	-	-	-	-	۱۶۱۰	۰/۰۰	۱۶۱۰
GR۴۸	۵۰۴۶	۰/۰۰	۵۰۴۶	۰/۰۰	-	-	-	-	-	-	۵۰۴۶	۰/۰۰	۵۰۴۶
ATT۴۸	-	-	۱۰۶۲۸	۰/۰۰	-	-	۱۰۶۶۱	۰/۰۰	-	-	۱۰۶۳۱	۰/۰۰	۱۰۶۲۸
Eil۵۱	-	-	۴۲۶	۰/۰۰	۴۲۷	۰/۰۰	۴۲۸	۰/۰۰	۴۳۱/۹	۱/۳۹	۴۲۶	۰/۰۰	۴۲۶
Berlin۵۲	-	-	۷۵۴۲	۰/۰۰	۷۵۴۲	۰/۰۰	-	-	۷۵۴۸/۸	۰/۰۹	۷۵۴۲	۰/۰۰	۷۵۴۲
ST۷۰	۶۸۵	۰/۰۱	۶۷۵	۰/۰۰	-	-	-	-	۶۷۸	۰/۴۴	۶۷۵	۰/۰۰	۶۷۵
Eil۷۶	-	-	۵۳۸	۰/۰۰	۵۴۰	۰/۰۰	۵۳۹	۰/۰۰	۵۴۳/۳	۰/۹۹	۵۳۹	۰/۰۰	۵۳۸
KroA۱۰۰	۲۱۵۰۴	۰/۰۱	۲۱۲۸۲	۰/۰۰	۲۱۲۹۶	۰/۰۰	۲۱۷۶۳	۰/۰۲	۲۱۳۰۳/۳	۰/۱۰	۲۱۲۸۲	۰/۰۰	۲۱۲۸۲
KroB۱۰۰	-	-	۲۲۱۴۱	۰/۰۰	-	-	۲۲۶۳۷	۰/۰۲	۲۲۲۲۵/۱	۰/۳۸	۲۲۱۹۸	۰/۰۰	۲۲۱۴۱
KroC۱۰۰	-	-	۲۰۷۴۹	۰/۰۰	-	-	۲۰۸۵۴	۰/۰۱	۲۰۸۴۶/۵	۰/۴۷	۲۰۷۴۹	۰/۰۰	۲۰۷۴۹
KroD۱۰۰	-	-	۲۱۲۹۴	۰/۰۰	-	-	۲۱۶۴۳	۰/۰۲	-	-	۲۱۳۵۸	۰/۰۰	۲۱۲۹۴
KroE۱۰۰	-	-	۲۲۰۶۸	۰/۰۰	-	-	۲۲۴۵۰	۰/۰۲	-	-	۲۲۰۶۸	۰/۰۰	۲۲۰۶۸
Eil۱۰۱	-	-	۶۲۹	۰/۰۰	-	-	۶۳۵	۰/۰۱	-	-	۶۳۳	۰/۰۱	۶۲۹
Lin۱۰۵	-	-	۱۴۳۷۹	۰/۰۰	-	-	۱۵۲۸۸	۰/۰۱	-	-	۱۴۳۷۹	۰/۰۰	۱۴۳۷۹
KroA۱۵۰	-	-	۲۶۵۲۴	۰/۰۰	-	-	۲۷۸۵۸	۰/۰۱	-	-	۲۶۵۲۴	۰/۰۰	۲۶۵۲۴
KroB۱۵۰	-	-	۲۶۱۳۰	۰/۰۰	-	-	۲۶۵۳۵	۰/۰۱	-	-	۲۶۲۹۷	۰/۰۱	۲۶۱۳۰
KroA۲۰۰	-	-	۲۹۳۶۸	۰/۰۰	۲۹۵۶۳	۰/۰۱	۲۹۹۶۱	۰/۰۲	-	-	۲۹۵۳۹	۰/۰۱	۲۹۳۶۸
KroB۲۰۰	-	-	۲۹۴۳۷	۰/۰۰	-	-	۳۰۳۵۰	۰/۰۳	-	-	۲۹۴۳۷	۰/۰۰	۲۹۴۳۷

به‌علاوه در همین جدول، در ستون دوم الگوریتم ژنتیک (GA) آورده شده است [۲۹]، در حالی که ستون‌های سوم و چهارم به‌ترتیب نشان‌دهنده الگوریتم‌های تکاملی (EA) [۳۰] و بهینه‌سازی ذرات (PSO) [۳۱] می‌باشند. چهارمین الگوریتمی که نتایج آن در جدول ۲ ارائه شده است، الگوریتم جمعیت زنبورها (BCO) است که در ستون پنجم آورده شده است [۳۲]. از طرف دیگر الگوریتم ICA پنجمین الگوریتمی است که در ستون ششم نشان داده شده است [۱۷]. توجه به این نکته ضروری است که الگوریتم ICA تفاوت‌های اساسی با الگوریتم MICA دارد. در الگوریتم پیشنهادی برخلاف روش ICA از توابع جدید جذب و انقلاب استفاده شده است. به‌علاوه برای افزایش کارایی آن از الگوریتم بهبود دهنده سه‌گانه استفاده شده است. این تغییرات سبب شد که الگوریتم از کارایی بیشتری برخوردار شود که در ادامه به آن پرداخته می‌شود. سرانجام ستون‌های هفتم و هشتم نشان‌دهنده نتایج MICA و بهترین مقادیری می‌باشند که تاکنون برای هر مسأله به وسیله الگوریتم‌های دیگر به‌دست آمده‌اند.

با مقایسه نتایج به‌دست‌آمده در این جدول می‌توان نتیجه گرفت که الگوریتم پیشنهادی نسبت به الگوریتم اصلاحی GA توانسته است که به جواب‌های با کیفیت‌تری دست یابد به‌طوری که در دو مثال *St*۷۰ و *KroA*۱۰۰ کیفیت جواب‌ها را افزایش داده است. از طرف دیگر در سه مثال *GR*۲۴، *Bayg*۲۹ و *Gr*۴۸ الگوریتم پیشنهادی جواب‌های یکسانی را به‌دست آورده است. بنابراین می‌توان نتیجه گرفت که اگرچه الگوریتم GA دارای کارایی خوبی برای مثال‌های با تعداد گره پایین است، اما هنگامی‌که تعداد گره‌ها افزایش پیدا می‌کند الگوریتم GA برخلاف الگوریتم پیشنهادی نمی‌تواند کارایی خود را حفظ کند و به جواب‌هایی با کیفیت پایین‌تر دست پیدا می‌کند.

دومین الگوریتمی که نتایج آن با روش پیشنهادی مقایسه شده است، الگوریتمPSO است. این الگوریتم در پنج مثالی که مورد آزمایش قرار گرفته است دارای ثبات بوده و توانسته است که جواب‌ها را با مقداری کمتر از یک درصد خطا به‌دست آورد. علی‌رغم این‌که این الگوریتم در مثال *Berlin*۵۲ به بهترین جواب تاکنون به‌دست‌آمده دست یافته است اما در چهار مثال باقیمانده نتوانسته است که با این دقت به جواب برسد. هم‌چنین در همین چهار مثال الگوریتم پیشنهادی توانسته است که جواب‌های بهتری را نسبت به این الگوریتم به‌دست آورد. بنابراین می‌توان نتیجه گرفت که الگوریتم MICA در مقایسه با الگوریتم PSO از قدرت بیشتری برخوردار است و توانا فرار از نقاط بهینه محلی را دارد.

در مقایسه الگوریتم پیشنهادی با الگوریتم BCO می‌توان مشاهده کرد که الگوریتم BCO فقط در مثال *Eil*۷۶ از چهارده مثال مورد آزمایش توانسته است که جوابی یکسان با الگوریتم پیشنهادی به‌دست آورد و در سیزده مثال باقیمانده الگوریتم پیشنهادی جواب‌های بهتری را نسبت به این الگوریتم به‌دست آورده است. باید توجه کرد که الگوریتم BCO به‌طور کلی در

مقایسه با سایر الگوریتم‌های ارائه‌شده در جدول ۲ از کارایی لازم برخوردار نبوده و در هیچ‌یک از مثال‌ها نتوانسته است که به بهترین جواب‌ها در ستون *Best* دست یابد. به‌علاوه با مقایسه الگوریتم ICA با الگوریتم MICA می‌توان مشاهده کرد که الگوریتم پیشنهادی از کارایی بسیار بیشتری نسبت به این الگوریتم برخوردار است به‌طوری که الگوریتم پیشنهادی توانسته است که در هر هفت مثال مربوطه جواب‌های باکیفیت‌تری را تولید کند. هم‌چنین برخلاف الگوریتم ICA که در هیچ‌کدام از هفت مثال نتوانسته است به بهترین جواب‌ها دست یابد، الگوریتم MICA در پنج مثال توانسته است که به بهترین جواب‌ها دست یابد.

آخرین الگوریتمی که نتایج آن در جدول ۲ با الگوریتم پیشنهادی مقایسه شده است، الگوریتم تکاملی EA است که توانسته است کارایی بسیار خوبی را نشان دهد و در همه مثال‌ها به بهترین جواب‌های تاکنون به‌دست‌آمده، دست یابد. لذا این الگوریتم بهترین الگوریتم ارائه‌شده در این جدول است که توانسته است جواب‌های بهتری را نسبت به الگوریتم پیشنهادی به‌دست آورد. بنابراین می‌توان از نظر قدرت یافتن بهترین جواب، الگوریتم‌های جدول ۲ را به‌ترتیب از ضعیف به قوی به‌صورت ICA، BCO، GA، PSO، MICA و EA مرتب کرد.

۶- نتیجه‌گیری

در این مقاله، نوعی اصلاح روش ICA ارائه شد که در آن برای افزایش کارایی الگوریتم از تابع جذب نزدیک‌ترین همسایه اصلاحی و روش انقلاب بهبوددهنده دو نقطه‌ای استفاده شد. هم‌چنین برای افزایش کیفیت جواب‌های به‌دست‌آمده روش بهبود دهنده سه‌گانه به‌کار گرفته شد. به‌نظر می‌رسد که ترکیب الگوریتم مربوطه با سایر روش‌های فراابتکاری مانند الگوریتم مورچگان، جستجوی ممنوع و یا استفاده از الگوریتم‌های قوی‌تر محلی مانند لین-کرنیگان به‌جای الگوریتم بهبوددهنده سه‌گانه نتایج بهتری داشته باشد. از طرف دیگر می‌توان این الگوریتم را بر روی مسائل دیگر بهینه‌سازی مانند مسأله مسیریابی وسیله نقلیه و مسأله خوشه‌بندی ظرفیت‌دار به‌کار برد. کار بر روی این ایده‌ها و کاربردی کردن آن‌ها به مقاله‌های بعدی موکول می‌شود.

مراجع

[1] Park, Y.B. (2001), A hybrid genetic algorithm for the vehicle scheduling problem with due times and time deadlines, *International Journal of Productions Economics*, **73(2)**, 175–188.

[2] Saleh, H.A. and Chelouah R. (2004), The design of the global navigation satellite system surveying networks using genetic algorithms, *Engineering Applications of Artificial Intelligence*, **17**, 111–122.

[3] Chan, D. and Mercier, D. (1989), IC insertion: An application of the traveling salesman problem, *International Journal of Production Research*, **27**, 1837–1841.

[4] Lawler, E.L., Lenstra, J.K. and Shmoys D.B. (1985), The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization, Wiley, New York.

[5] Brummit, B. and Stentz, A. (1996), Dynamic mission planning for multiple mobile robots, *Proceedings of the IEEE International Conference on Robotics and Automation*, **3**, 2396-2401.

[6] Zhang, W. (1993), *Truncated branch-and-bound: A case study on the asymmetric tsp*, in: Working Note of AAAI 1993 Spring Symposium: AI and NP-Hard Problems, Stanford, CA, 160–166.

[7] Bektas, T. (2006), The multiple traveling salesman problem: an overview of formulations and solution procedures, *Omega*, **34**, 209 – 219.

[8] Yadlapalli S., Malik W.A., Darbhaa S. and Pachter M. (2004), A Lagrangian-based algorithm for a Multiple Depot, Multiple Traveling Salesmen Problem, *Nonlinear Analysis: Real World Applications*, **10(4)**, 1990-1999.

[9] Gavish B. and Srikanth K. (1986), An optimal solution method for large-scale multiple traveling salesman problems, *Operations Research*, **34(5)**, 698–717.

[10] Gromicho J., Paixao J. and Branco I. (1992), Exact solution of multiple traveling salesman problems, In: *MustafaAkgül, et al., editors.Combinatorial optimization*. NATO ASI Series, Berlin: Springer, **F82**, 291–292.

[11] Garey, M.R. and Johnson, D.S. (1979), *Computers and intractability: A guide to the theory of NP-completeness*, San Francisco: W.H. Freeman.

[12] Karp, R.M. (1977), Probabilistic analysis of partitioning algorithms for the traveling salesman problem in the plane, *Mathematics of Operations Research*, **2**, 209-224.

[13] Qu, H., Yi, Z. and Tang, H. (2007), A columnar competitive model for solving multi-traveling salesman problem, *Chaos Solitons and Fractals*, **31**, 1009–1019.

[14] Ryan, J.L., Bailey, T.G., Moore J.T. and Carlton W.B. (1998), Reactive Tabu search in unmanned aerial reconnaissance simulations, *Proceedings of the 1998 winter simulation conference*, **1**, 873–879.

[15] Zhang, T., Gruver, W. A. and Smith, M. H. (1999), Team scheduling by genetic search, Proceedings of the second international conference on intelligent processing and manufacturing of materials, **2**, 839–844.

[16] Prins, C. (2009), Two memetic algorithms for heterogeneous fleet vehicle routing problems, *Engineering Applications of Artificial Intelligence*, **22**, 916–928.

[17] Nemati, K., Shamsuddin, S.M. and Saberi Kamarposhti, M. (2011), Using Imperial Competitive Algorithm for Solving Traveling Salesman Problem and Comparing the Efficiency of the Proposed Algorithm with Methods in Use, *Australian Journal of Basic and Applied Sciences*, **5(10)**, 540-543.

[18] Vasquez, M., Dupont, A. and Habet. D. (2005), *Metaheuristics: Progress as real Problem Solvers*, MIC-Kluwer.

[19] Wang, C.H. and Lu, J.Z. (2009), A hybrid genetic algorithm that optimizes capacitated vehicle routing problems, *Expert Systems with Applications*, **36**, 2921–2936.

[20] Zhang, X. and Tang, L. (2009), A new hybrid ant colony optimization algorithm for the vehicle routing problem, *Pattern Recognition Letters*, **30**, 848–855.

[21] Ai, T. J. and Kachitvichyanukul, V. (2009), Particle swarm optimization and two solution representations for solving the capacitated vehicle routing problem, *Computers & Industrial Engineering*, **56**, 380–387.

[22] Atashpaz-Gargari, E. and Lucas, C. (2007), Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition, In: *Proceedings of the IEEE Congress on Evolutionary Computation*, Singapore, 4661–4667.

[23] Kaveh, A. and Talatahari, S. (2010), Optimum design of skeletal structures using imperialist competitive algorithm, *Computers & Structures*, **88 (21-22)**, 1220-1229.

[24] Niknam, T., Taherian Fard, E., Pourjafarian, N. and Rousta, A. (2011), An efficient hybrid algorithm based on modified imperialist competitive algorithm and K-means for data clustering, *Engineering Applications of Artificial Intelligence*, **24(2)**, 306-317.

[25] Lucas, C., Nasiri-Gheidari, Z. and Tootoonchian, F. (2010), Application of an imperialist competitive algorithm to the design of a linear induction motor, *Energy Conversion and Management*, **51(7)**, 1407-1411.

[26] Rajabioun, R., Atashpaz-Gargari, E. and Lucas, C. (2008), Colonial Competitive Algorithm as a Tool for Nash Equilibrium Point Achievement, *Lecture Notes in Computer Science*, **5073**, 680-695.

[27] Khabbazi, A., Atashpaz-Gargari, E. and Lucas, C. (2009), Imperialist competitive algorithm for minimum bit error rate beamforming, *International Journal of Bio-Inspired Computation,* **1(1)**, 125–133.

[28] Pellegrini, P. (2005), *Application of two nearest neighbor approaches to a rich vehicle routing problem*, TR/IRIDIA/2005-15, IRIDIA, Universite Librede Bruxelles, Brussels, Belgium.

[29] Ray, S.S., Bandyopadhyay, S. and Pal, S.K. (2004), New operators of genetic algorithms for traveling salesman problem, *Proceedings of the 17th International Conference on Pattern Recognition*, **2**, 497–500.

[30] Shen, G. and Zhang, Y.Q. (2011), A new evolutionary algorithm using shadow price guided operators, *Applied Soft Computing*, **11(2)**, 1983-1992.

[31] Zhong, W., Zhang, J. and Chen, W. (2007), A novel discrete particle swarm optimization to solve traveling salesman problem, Evolutionary Computation, IEEE Congress on, 3283–3287.

[32] Wong, L.P., Low, M.Y.H. and Chong, C.S. (2008), A bee colony optimization algorithm for traveling salesman problem, Modeling & Simulation, AICMS 08. Second Asia International Conference on, 818–823.

Application a Modified Imperialist Competitive Algorithm for Solving the Traveling Salesman Problem

Majid Yousefikhoshbakht[1], Farzad didehvar[2] and Farhad Rahmati[2]

[1]Young Researchers Club, Hamedan Branch, Islamic Azad University, Hamedan, Iran

[2]College of Mathematics and Computer Science, Amirkabir University of Technology, Tehran, Iran

Abstract

This paper proposes a modified Imperialist Competitive Algorithm (MICA) for solving the Traveling Salesman Problem (TSP) that is different with common Imperialist Competitive Algorithm (ICA) in assimilation policy between Imperialist and colonies countries and revolution of colonies. Furthermore, the 3-opt local search is used for increasing performance of the algorithm. The new ICA algorithm is tested on nineteen instances of TSBLIB and its performance is compared with ICA, Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Evolutionary Algorithm (EA) and Bee Colony Optimization (BCO). Extensive computational tests confirm the effectiveness of the proposed approach.

Keywords: Traveling Salesman Problem, Imperialist Competitive Algorithm, NP-Hard Problems.

Mathematics Subject Classification (2000): 68T20, 65Y20