علوم و مهندسی کامپیوتر
نشریه علمی پژوهشی انجمن کامپیوتر ایران
مجلد ۹، شماره ۱ و ۳ (الف)، ۱۳۹۰
صفحات ۱-۱۱
مقاله عادی

حل نمونه‌های بزرگ مسئله فروشنده دوره‌گرد متقارن با استفاده از یک الگوریتم جستجوی گرانشی گسسته ترکیبی

محمدباقر دولتشاهی[۱] حسین نظام‌آبادی‌پور[۲] ماشاالله ماشین‌چی[۱]

[۱]دانشکده ریاضی و علوم کامپیوتر، دانشگاه شهید باهنر کرمان، کرمان، ایران
[۲]دانشکده مهندسی برق، دانشگاه شهید باهنر کرمان، کرمان، ایران

چکیده

امروزه از الگوریتم‌های تصادفی مبتنی بر جمعیت بطور گسترده در حل مسائل بهینه‌سازی استفاده می‌شود. اغلب این الگوریتم‌ها با الهام از فرایندهای فیزیکی یا رفتارهای موجودات طبیعی به وجود آمده‌اند و هر کدام خصوصیات منحصر به خود را دارند. الگوریتم جستجوی گرانشی یکی از جدیدترین الگوریتم‌های تصادفی مبتنی بر جمعیت است که با الهام از قانون گرانش و قوانین حرکت و با ایجاد تعادلی مناسب بین قابلیت‌های کاوش و بهره‌گیری، برای حل مسائل بهینه‌سازی پیوسته معرفی شده است. در این مقاله، ابتدا با نگاهی بالا به پائین، سعی در شناخت مولفه‌های اصلی فرایند جستجو در فضای راه‌حل مسائل بهینه‌سازی توسط این الگوریتم خواهد شد. سپس، به بازتعریف این مولفه‌ها برای جستجو در فضای راه‌حل مسئله فروشنده دوره‌گرد متقارن که یکی از مشهورترین و سخت‌ترین مسائل بهینه‌سازی ترکیبیاتی است پرداخته می‌شود. نتایج پیاده‌سازی الگوریتم پیشنهادی در حل مسئله فروشنده دوره‌گرد و مقایسه آن با تعدادی از الگوریتم‌های حل تقریبی این مسئله، نشان دهنده کارایی مناسب الگوریتم پیشنهادی می‌باشد.

کلمات کلیدی: الگوریتم جستجوی گرانشی، بازتعریف عملگرها، عملگر جابجایی، مسئله فروشنده دوره‌گرد، مسائل NP-hard.

۱- مقدمه

امروزه چگونگی حل کارآمد گروه بزرگی از مسائل بهینه‌سازی ترکیبیاتی[۱] موسوم به مسائل NP-hard[۲] یکی از مهمترین چالش‌های اصلی ریاضیدانان، دانشمندان و مهندسان کامپیوتر تلقی می‌شود. حل این مسائل بهینه‌سازی به معنای پیدا کردن بهترین راه‌حل مسئله از میان یک مجموعه بسیار بزرگ (اما محدود) از راه‌حل‌های ممکن[۳] برای مسئله می‌باشد. با وجود تلاش‌های زیادی که در چندین دهه گذشته برای شناخت ویژگی‌های این دسته از مسائل صورت گرفته است، متاسفانه تا به حال یک الگوریتم که بتواند بهترین راه‌حل هر نمونه ورودی از یک مسئله NP-hard را در بدترین حالت در زمان چندجمله‌ای پیدا کرده و بهینه بودن راه‌حل را نیز ضمانت کند، ارائه نشده است. از طرف دیگر، در مجموعه دانش فعلی بشر هنوز هیچ اثبات قابل پذیرشی وجود ندارد که این ادعا را که "یک الگوریتم با پیچیدگی زمانی چندجمله‌ای وجود دارد که می‌تواند بهترین راه‌حل هر نمونه ورودی از یک مسئله NP-hard را به همراه اثبات بهترین بودن آن راه‌حل ارائه دهد"، تکذیب کند. از اینرو، امروزه مسائل NP-hard نقشی بسیار حساس و کلیدی را هم در تکامل نظریه پیچیدگی محاسباتی[۴] و هم در توسعه ابزارهای جدید محاسباتی ایفا می‌کنند [۳-۷].

در میان مجموعه مسائل NP-hard، مسئله فروشنده دوره‌گرد[۵] از محبوبیت بیشتری نسبت به دیگر مسائل در بین محققان برخوردار است. از دلایل مهم برای این موضوع می توان به سادگی آن در بُعد فهم و رابطه‌مند شدن، دشواری آن در بُعد حل و در نهایت نزدیکی آن به بسیاری از مسائل دنیای واقعی اشاره کرد. در این مسئله، فروشنده‌ای می خواهد با شروع از یک شهر و عبور از n شهر، اجناس خود را بفروشد بطوریکه از هر شهر یک بار و فقط یک بار عبور کند. در نهایت سفر

فروشنده به شهری ختم می‌شود که سفرش را از آنجا آغاز کرده است. در این مسئله هدف پیدا کردن کم هزینه‌ترین مسیر ممکن یا به عبارتی کوتاه‌ترین مسیر ممکن می‌باشد [۱، ۳ و ۸].

مسئله فروشنده دوره‌گرد و به خصوص نمونه‌های بسیار بزرگ آن به وفور در بسیاری از مسائل دنیای واقعی ظاهر می‌شود. به عبارت دیگر، بسیاری از مسائل ترکیبیاتی پیچیده‌ای که در دنیای واقعی با آنها سر و کار داریم، قابل مدل شدن به مسئله فروشنده دوره‌گرد هستند. بنابراین، تلاش برای حل نمونه‌های بزرگ مسئله فروشنده دوره‌گرد، موضوعی نیست که فقط مورد علاقه دانشمندان ریاضیات و علم کامپیوتر باشد، بلکه دلایل کافی برای علاقمند شدن محققان و مهندسان حوزه‌های مختلف نیز به حل نمونه‌های بزرگ این مسئله وجود دارد. با نگاه از دریچه مهندسی به مسئله فروشنده دوره‌گرد، مشاهده می‌شود که بیش از آنکه به روش‌هایی نیاز باشد که فقط به صورت تئوری قادر به حل نمونه‌های بزرگ این مسئله هستند (مثل تمام روش‌های با پیچیدگی زمانی بزرگتر از چندجمله‌ای) ، به روش‌هایی نیاز است که در عمل نیز قابلیت پیاده‌سازی و اجرا را داشته باشند. متاسفانه، از آنجاکه هنوز الگوریتمی ارائه نشده است که بتواند جواب بهینه نمونه‌های بزرگ این مسئله را در زمانی قابل قبول (حداکثر چندجمله‌ای) پیدا کرده و این بهینگی را نیز ضمانت کند، لذا متخصصان به این نتیجه رسیده‌اند که تا مشخص شدن پاسخ مسئله P=?NP [۷]، ناچارا برای حل نمونه‌های بزرگ مسئله فروشنده دوره‌گرد باید به جواب‌های نزدیک به بهینه[۷] اکتفا کرد. از اینرو، امروزه از نگاه مهندسی می‌توان به فرایند حل مسئله فروشنده دوره‌گرد نه به عنوان یک مسئله بهینه‌سازی کلاسیک که در آن هدف از حل مسئله پیدا کردن بهترین راه‌حل مسئله است، بلکه به عنوان فرایند یافتن یک مصالحه مناسب بین "بهینگی یک راه‌حل" و "زمان پیدا کردن آن راه‌حل" نگاه کرد [۹-۱۲].

روش‌های فرا ابتکاری[۸]، ابزارهایی با پیچیدگی زمانی معقول (معمولا در حد چندجمله‌ای) برای یافتن جواب یا جواب‌های نزدیک به بهینه مسائل بهینه‌سازی سخت و پیچیده‌ای هستند که معمولا روش‌های کلاسیک بهینه‌سازی در حل آنها تابحال موفقیت چندانی بدست نیاورده‌اند. این روش‌ها با بهره بردن از دو مفهوم کاوش[۹] و بهره‌گیری[۱۰] سعی در جستجوی کنترل شده فضای راه‌حل یک مسئله بهینه‌سازی را دارند. بدون شک هر چه قدرت یک الگوریتم فرا ابتکاری در کنترل مناسب این دو قابلیت بیشتر باشد، توانایی آن الگوریتم در پیدا کردن جواب‌های مناسب برای مسئله نیز افزایش خواهد یافت [۱۰-۱۲].

اخیرا، یک روش فرا ابتکاری جدید با نام الگوریتم جستجوی گرانشی[۱۱] [۱۳] با الهام از قانون گرانش و قوانین حرکت ارائه شده است. الگوریتم جستجوی گرانشی با ایجاد یک تعادل مناسب بین دو قابلیت کاوش و بهره‌گیری، قادر است فرایند جستجو در فضای راه‌حل مسائل بهینه‌سازی پیوسته[۱۲] [۱۳] و باینری [۱۴] پیچیده را برای یافتن جواب‌های نزدیک به بهینه با موفقیت انجام دهد. در [۲] برای اولین بار توسط مولفین نشان داده شده است که با بازتعریف عملگرهای اصلی این الگوریتم، می‌توان جستجوی نسبتا موفقیت‌آمیزی در فضای راه‌حل نمونه‌های با اندازه متوسط مسئله فروشنده دوره‌گرد را توسط این الگوریتم شاهد بود. در این مقاله در ادامه گسسته‌سازی انجام شده در مقاله [۲]، یک الگوریتم جستجوی گرانشی گسسته ترکیبی برای حل نمونه‌های با اندازه بزرگ مسئله فروشنده دوره‌گرد پیشنهاد خواهیم کرد.

ادامه این مقاله به این صورت سازماندهی شده است: در بخش ۲ مروری بر مسئله فروشنده دوره‌گرد و روش‌های مختلف حل آن خواهیم پرداخت. در بخش ۳ مروری بر نسخه اصلی (نسخه پیوسته) الگوریتم جستجوی گرانشی خواهیم داشت. در بخش ۴ روش پیشنهادی در بازتعریف عملگرهای اصلی الگوریتم جستجوی گرانشی برای جستجو در فضای راه‌حل مسئله فروشنده دوره‌گرد شرح داده خواهد شد. در بخش ۵ نتایج حاصل از پیاده‌سازی الگوریتم پیشنهادی در حل مسئله فروشنده‌گرد و همچنین مقایسه نتایج آورده شده است. نهایتا در بخش ۶ به نتیجه‌گیری در مورد روش ارائه شده پرداخته شده است.

۲- مروری بر مسئله فروشنده دوره‌گرد و روش‌های حل آن

چنانکه گفته شد، مسئله فروشنده دوره‌گرد نقش مهمی در تکامل بخش عمده‌ای از تحقیقات علمی و حتی کاربردی، بازی می‌کند. این مسئله تابحال نقشی بسیار حساس و کلیدی را هم در تکامل نظریه پیچیدگی محاسباتی[۱۳] و هم در توسعه ابزارهای جدید محاسباتی ایفا کرده است [۱۰]. در ادامه این بخش، ابتدا برای درک بهتر این مسئله به تعریف ریاضی آن پرداخته و سپس مروری کلی بر روش‌های مختلف حل این مسئله خواهیم داشت.

۲-۱- تعریف ریاضی مسئله فروشنده دوره‌گرد کلاسیک

به بیان ریاضی، مسئله فروشنده دوره‌گرد به وسیله یک گراف جهت‌دار و وزن‌دار کامل به صورت گراف $G\ (V,\ E,\ d)$ قابل بیان است که در آن V شامل مجموعه‌ای از گره‌ها (شهرها)، E بیان‌گر مجموعه‌ای از یال‌ها و $d : E \to R$ یک تابع وزن‌دار است که عدد مثبت $d(c_i,\ c_j)$ را به یال متصل کننده گره iام (c_i) و گره jام (c_j) که با فاصله این گره‌ها متناسب است، نسبت می‌دهد. در این مسئله، هدف از بهینه‌سازی یافتن کوتاه‌ترین مسیر بسته‌ای است که از تمام شهرها و از هر یک تنها یک بار گذشته باشد (یک مسیر همیلتونی). گاها برای سادگی به کاندیدای جواب در این مسئله، سفر[۱۴] گفته می‌شود [۱].

در مسئله فروشنده دوره‌گرد متقارن[۱۵]، فاصله میان شهرها مستقل از جهت گذر از آن‌ها است یا به عبارتی $d(c_i,\ c_j) = d(c_j,\ c_i)$، در حالی که در مسئله فروشنده دوره‌گرد نامتقارن[۱۶]، حداقل برای یک جفت از گره‌ها داریم: $d(c_i,\ c_j) \neq d(c_j,\ c_i)$ در مسئله فروشنده دوره‌گرد قصد داریم کوتاه‌ترین سفر را که یک جایگشت[۱۷] π از شهرهای $\{c_1,\ c_2,\ \ldots,\ c_n\}$ است را به گونه‌ای بیابیم که $f(\pi)$ آن کمینه شود. $f(\pi)$ با استفاده از رابطه زیر محاسبه می‌شود [۱]:

$$f(\pi)= \sum_{i=1}^{n-1} d\left(c_{\pi(i)},\ c_{\pi(i+1)}\right) + d\left(c_{\pi(n)},\ c_{\pi(1)}\right) . \qquad (۱)$$

۲-۲- روش‌های حل مسئله فروشنده دوره‌گرد

همانند هر مسئله دیگر مسائل محاسباتی پیچیده، برای حل مسئله فروشنده دوره‌گرد نیز دو دسته الگوریتم وجود دارد: ۱) الگوریتم‌های دقیق[۱۸] و ۲) الگوریتم‌های ابتکاری[۱۹] [۱۱]. الگوریتم‌های دقیقِ ارائه شده برای حل مسئله فروشنده دوره‌گرد، قادرند راه‌حل بهینه هر نمونه از این مسئله را در زمانی نمایی پیدا کرده و بهینگی راه‌حل بدست آمده را نیز ضمانت کنند. هر چند احتمالِ وجود یک الگوریتم دقیق با درجه زمانی چندجمله‌ای برای حل هر نمونه از مسئله فروشنده دوره‌گرد غیرممکن نیست، اما متاسفانه تابحال چنین الگوریتمی توسط هیچ کسی ارائه نشده است [۷]. از مشهورترین الگوریتم‌های دقیق استفاده شده برای حل مسئله فروشنده دوره‌گرد، الگوریتم‌های مبتنی بر روش شاخه و حد[۲۰] هستند [۱۵، ۱۶]. در مقابل، الگوریتم‌های ابتکاریِ حل مسئله فروشنده دوره‌گرد، قادرند یک راه‌حلِ نزدیک به بهینه را در یک مدت زمان معقول (معمولا در حد چندجمله‌ای) برای این مسئله پیدا کنند. این الگوریتم‌ها خود نیز در دو دسته قرار می‌گیرند: ۱) الگوریتم‌های ابتکاری با قابلیت تضمین بدترین حالت و ۲)

الگوریتم‌های ابتکاری بدون قابلیت تضمین بدترین حالت. گروه اول، آنهایی هستند که درصد خطای راه‌حل خروجی را در بدترین حالت و همچنین زمان پیداکردن آن راه‌حل را در بدترین حالت ممکن ضمانت می‌کنند. در بسیاری از متون به این الگوریتم‌ها، الگوریتم‌های تقریبی[۲۱] گفته می‌شود. مرجع [۱۷] یکی از بهترین الگوریتم‌های تقریبی را برای مسئله فروشنده دوره‌گرد ارائه می‌دهد. در مقابل، در الگوریتم‌های گروه دوم هیچ‌گونه ضمانتی برای درصد خطا در بدترین حالت و همچنین در اغلب موارد برای زمان اجرا در بدترین حالت ممکن ارائه نمی‌شود. گرچه، به صورت تجربی و در عمل مشاهده شده است که راه‌حل‌هایی با کیفیت تقریبا مناسب توسط الگوریتم‌های ابتکاری دسته دوم پیدا می‌شود.

الگوریتم‌های ابتکاریِ بدون قابلیت تضمین بدترین حالت برای مسئله فروشنده دوره‌گرد، خود نیز در ۳ گروه قرار می‌گیرند:

- الگوریتم‌های ساخت سفر[۲۲]: این الگوریتم‌ها به طور تدریجی یک سفر را با اضافه کردن یک شهر به آن در هر گام ایجاد می‌کنند. از مهمترین الگوریتم‌های ساخت سفر برای مسئله فروشنده دوره‌گرد، می‌توان به الگوریتم‌های نزدیک‌ترین همسایه[۲۳] و حریصانه[۲۴] (آزمند) اشاره کرد. برای اطلاعات بیشتر در مورد الگوریتم‌های ساخت سفر می‌توان به [۱۸] مراجعه کرد.
- الگوریتم‌های بهبود دهنده سفر[۲۵]: این الگوریتم‌ها یک سفر مجاز را به عنوان ورودی گرفته و به تدریج با برخی جابجایی‌ها در یال‌های آن، کیفیت آن را بهبود می‌بخشند. از مهمترین الگوریتم‌های بهبود دهنده سفر می‌توان به خانواده الگوریتم‌های r-opt [۱۹] و "الگوریتم‌های فرا ابتکاری" [۲۰-۲۳] اشاره کرد.
- الگوریتم‌های ترکیبی: این الگوریتم‌ها از ترکیب خصوصیات دو گروه بالا برای پیدا کردن یک راه‌حل برای مسئله فروشنده دوره‌گرد استفاده می‌کنند. از مهمترین و کاراترین الگوریتم‌های این گروه می‌توان به الگوریتم GENIUS [۲۴] اشاره کرد.

۳- نسخه پیوسته الگوریتم جستجوی گرانشی

الگوریتم جستجوی گرانشی یک الگوریتم تصادفی[۲۶] مبتنی بر جمعیت[۲۷] است که اخیرا توسط محققان ایرانی با الهام از طبیعت معرفی شده است [۱۳]. این الگوریتم با استفاده از یک مجموعه از عامل‌های جستجوگر (موسوم به جرم[۲۸]ها) و شبیه‌سازی قانون گرانش نیوتون و قوانین حرکت بر روی این مجموعه از عامل‌ها، یک روش مناسب برای جستجوی راه‌حل‌های نزدیک به بهینه در فضای راه‌حل مسائل بهینه‌سازی پیوسته پیچیده فراهم می‌کند. در ادامه الگوریتم جستجوی گرانشی در دو قدم کلی توضیح داده می‌شود:

الف- تشکیل یک سیستم مصنوعی با زمان گسسته در محیط مسئله، موقعیت یابی اولیه برای عامل‌ها (اجرام)، وضع قوانین حاکم و تنظیم پارامترها.

ب- گذر زمان، حرکت اجرام و به روز رسانی پارامترها تا پیش آمدن زمان توقف.

۳-۱- تشکیل سیستم، وضع قوانین و تنظیم پارامترها

"فضای سیستم" در الگوریتم جستجوی گرانشی شامل یک دستگاه مختصات چند بُعدی در فضای تعریف مسئله است. هر نقطه از این فضا در واقع یک جواب مسئله است و از اینرو به این فضا معمولا فضای جواب[۲۹] یا فضای تصمیم[۳۰] گفته می‌شود. در الگوریتم جستجوی گرانشی، عامل‌های جستجوگر در فضای راه‌حل، مجموعه‌ای از عامل‌ها هستند. در طراحی این الگوریتم فرض شده است که تنها قوانین گرانش و حرکت حاکم هستند. صورت کلی این قوانین در الگوریتم جستجوی گرانشی، با تغییرات جزئی تقریباً شبیه همان قوانین طبیعت است و به صورت آنچه در زیر تعریف می‌شود در نظر گرفته شده‌اند.

قانون گرانش: هر عامل در سیستم مصنوعی تمام عامل‌های دیگر را به سمت خود جذب می‌کند. مقدار این نیرو برای یک عامل، متناسب است با حاصلضرب جرم گرانشی آن عامل، جرم گرانشی عامل دوم و عکس فاصله آن دو عامل.

قانون حرکت: سرعت فعلی هر عامل برابر است با مجموع ضریبی از سرعت قبلی عامل و تغییر سرعت آن. تغییر سرعت یا "شتاب" هر عامل نیز برابر است با نیروی وارد بر آن عامل، تقسیم بر جرم گرانشی آن عامل.

برای تشریح صوری قوانین الگوریتم جستجوی گرانشی، فرض کنید یک سیستم با s عامل جستجوگر وجود دارد و در آن موقعیت عامل i-اُم به صورت رابطه (۲) تعریف می‌شود:

$$X_i = \left(x_i^1, \ldots, x_i^d, \ldots, x_i^n\right) \;\; ; \quad i=1, 2, \ldots, s \,, \tag{۲}$$

که x_i^d مکان عامل i-اُم در بُعد d است و n نشان‌دهنده بُعد فضای جستجو است. لازم به ذکر است که موقعیت هر عامل بیانگر یک جواب ممکن از مسئله است. بر اساس [۱۳]، جرم گرانشی عامل i در زمان t (یعنی $M_i(t)$) پس از محاسبه شایستگی جمعیت فعلی با استفاده از رابطه‌های (۳) و (۴) محاسبه می‌شود:

$$q_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)} \;\; , \tag{۳}$$

$$M_i(t) = \frac{q_i(t)}{\sum_{j=1}^{s} q_j(t)} \;\; , \tag{۴}$$

که $M_i(t)$ و $fit_i(t)$ به‌ترتیب جرم گرانشی عامل i و شایستگی عامل i را در زمان t نشان می‌دهند. همچنین $worst(t)$ و $best(t)$ برای مسائل کمینه‌سازی با استفاده از رابطه‌های (۵) و (۶) محاسبه می‌شوند (در مسائل بیشینه‌سازی کافی است جای بهترین و بدترین عامل را در این رابطه‌ها عوض کرد):

$$best(t) = \min_{j\in\{1,\ldots,s\}} fit_j(t) \,, \tag{۵}$$

$$worst(t) = \max_{j\in\{1,\ldots,s\}} fit_j(t) \;. \tag{۶}$$

پس از محاسبه جرم گرانشی هر عامل، حال باید شتاب، سرعت و موقعیت جدید هر عامل را محاسبه کنیم. برای محاسبه شتاب عامل i در زمان t، برآیند نیروهای وارد شده از جانب K عامل بهتر بر روی عامل i بر اساس قانون گرانش محاسبه می‌شود (رابطه (۷) و (۸)). همچنین برای محاسبه سرعت یا میزان جابجایی عامل i در زمان $t+1$، کسری از سرعت این عامل در زمان t با شتاب آن در زمان t جمع می‌شود (رابطه (۹)). در نهایت موقعیت جدید عامل i، می تواند بر اساس رابطه (۱۰) محاسبه شود.

$$F_i^d(t) = \sum_{j\in Kbest,\, j\neq i} rand_j G(t) \frac{M_j(t) M_i(t)}{R_{ij}(t) + \varepsilon} \left(x_j^d(t) - x_i^d(t)\right) , \tag{۷}$$

$$a_i^d(t) = \frac{F_i^d(t)}{M_i(t)} = \sum_{j\in Kbest,\, j\neq i} rand_j G(t) \frac{M_j(t)}{R_{ij}(t) + \varepsilon} \left(x_j^d(t) - x_i^d(t)\right) , \tag{۸}$$

$$V_i^d(t+1) = rand_i \times V_i^d(t) + a_i^d(t) \,, \tag{۹}$$

$$x_i^d(t+1) = x_i^d(t) + V_i^d(t+1)\,. \qquad (۱۰)$$

برخی پارامترهای استفاده شده در رابطه‌های بالا، عبارتند از:

- $rand_i$ و $rand_j$ دو عدد تصادفی در بازه [۰و۱] هستند.
- $R_{ij}(t)$ فاصله اقلیدسی بین دو عامل i و j است که به‌صورت $\|x_i(t), x_j(t)\|_2$ تعریف می‌شود.
- ε یک مقدار کوچک برای جلوگیری از صفر شدن مخرج کسر رابطه (۸) در حالتی که فاصله اقلیدسی دو عامل صفر است، می‌باشد.
- $G(t)$ ضریب یا ثابت گرانش است و معمولا در ابتدای اجرای الگوریتم با مقدار G_0 مقداردهی می‌شود و در طول زمان مقدارش کاهش می‌یابد. این ضریب، یک پارامتر مناسب برای کنترل مصالحه بین قابلیت‌های کاوش و بهره‌گیری الگوریتم به حساب می‌آید. مقادیر بزرگ برای این پارامتر باعث تقویت توانایی کاوش الگوریتم و مقادیر کوچک آن باعث افزایش توانایی بهره‌وری الگوریتم می‌شود. لذا از آنجا که در مراحل اولیه جستجو لازم است که الگوریتم به جستجوی نقاط جدیدی در فضا مسئله پرداخته و در مراحل پایانی باید با افزایش توان بهره‌گیری به بهبود جواب‌های دیده شده بپردازد، گزینه مناسب برای ثابت گرانش آن است که با یک مقدار اولیه بزرگ شروع شده و با گذشت زمان مقدار آن کاهش یابد. روش‌های مختلفی برای کاهش مقدار G در طول زمان وجود دارد. با توجه به اهمیت این پارامتر در کنترل کاوش و بهره‌گیری، انتخاب روش مناسب برای کاهش مقدار آن باید با توجه به چگونگی مصالحه بین قابلیت‌های کاوش و بهره‌گیری برای حل مسئله مورد نظر انجام شود.
- $Kbest$ مجموعه‌ای از K عامل جمعیت فعلی با بهترین مقدار شایستگی می‌باشد. مقدار K ثابت نیست و در واقع تابعی از زمان است. در ابتدای اجرای الگوریتم با مقدار K_0 مقداردهی می‌شود و مقدارش در طول زمان معمولا بصورت خطی به عدد یک کاهش می‌یابد. دلیل انتخاب K عامل برتر از میان مجموعه عامل‌های مسئله را می‌توان در دو موضوع زیر خلاصه کرد: الف) کاهش پیچیدگی محاسباتی الگوریتم جستجوی گرانشی و ب) کمک به ایجاد تعادلی مناسب‌تر بین قابلیت‌های کاوش و بهره‌گیری.

۳-۲- گذر زمان، حرکت اجرام و به روز رسانی پارامترها

در ابتدای تشکیل سیستم، هر عامل به صورت تصادفی در یک نقطه از فضا که جوابی از مسئله است قرار می‌گیرد. در هر لحظه از زمان، عامل‌ها ارزیابی شده، تغییر مکان یا شتاب هر عامل پس از محاسبه روابط (۸) تا (۱۰) محاسبه شده، و در زمان بعد هر عامل در موقعیت جدیدش قرار می‌گیرد.

گام ۱)	تعیین محیط سیستم و مقداردهی پارامترها.
گام ۲)	مقداردهی اولیه به عامل‌ها به صورت تصادفی.
گام ۳)	ارزیابی شایستگی عامل‌ها.
گام ۴)	به روز رسانی پارامترهای $best$، $worst$ و محاسبه M برای هر عامل.
گام ۵)	به روز رسانی پارامترهای G و $Kbest$.
گام ۶)	محاسبه نیروی وارده شده به هر عامل از طرف عامل‌های عضو مجموعه $Kbest$.
گام ۷)	محاسبه شتاب و سرعت هر عامل.
گام ۸)	به روز رسانی موقعیت هر عامل.
گام ۹)	اگر شرط توقف برآورده نشده است به گام ۳ برو.
گام ۱۰)	برگرداندن بهترین راه‌حل پیدا شده.

شکل ۱- قالب کلی الگوریتم جستجوی گرانشی

مهمترین پارامترهای سیستم شامل: جرم گرانشی یک عامل، به روزرسانی مقدار K و به روز رسانی مقدار ضریب گرانش نیوتن می‌باشند که در هر مرحله محاسبه و به روز رسانی می‌شوند. همچنین شرط توقف یکی دیگر از پارامترهای مهم است که معمولا پس از رسیدن به زمانی خاص، شرایط آن برآورده می‌شود.

برای درک بهتر نحوه حل مسائل بهینه‌سازی توسط الگوریتم جستجوی گرانشی، الگوریتم آن در شکل ۱ آورده شده است.

۴- الگوریتم جستجوی گرانشی گسسته پیشنهادی

چنانکه در بخش قبل مشاهده شد، نسخه اصلی الگوریتم جستجوی گرانشی برای جستجو در فضای راه‌حل مسائل بهینه‌سازی پیوسته معرفی شده است. از اینرو نمی‌توان از این نسخه از الگوریتم مستقیما برای حل مسائل بهینه‌سازی گسسته استفاده کرد. برای اینکه الگوریتم جستجوی گرانشی بتواند در فضای راه‌حل مسائل بهینه‌سازی گسسته نیز فرایند جستجو را با موفقیت انجام دهد، لازم است مفاهیم و عملگرهایی از الگوریتم که مختص جستجو در فضای راه‌حل مسائل بهینه‌سازی پیوسته هستند، برای جستجو در فضاهای گسسته بازتعریف شوند.

در ادامه، ابتدا به شناخت و بررسی آن دسته از مفاهیم و عملگرهایی از الگوریتم جستجوی گرانشی که باعث حل مسائل بهینه‌سازی توسط این الگوریتم می‌شوند، پرداخته و سپس به بازتعریف این مفاهیم و عملگرها برای جستجو در فضای راه‌حل مسئله فروشنده دوره‌گرد خواهیم پرداخت. در نهایت، گام‌های اصلی الگوریتم جستجوی گرانشی گسسته ترکیبی تشریح خواهد شد.

۴-۱- مفاهیم و عملگرهای اصلی الگوریتم جستجوی گرانشی

برای شناخت و بررسی جامع مفاهیم و عملگرهای اصلی الگوریتم جستجوی گرانشی، به نظر می‌رسد که بهترین روش، بررسی "بالا به پائین[۳۱]" روند حل مسائل بهینه‌سازی توسط نسخه اصلی این الگوریتم می‌باشد. با یک نگاه بالا به پائین به روند حل مسائل بهینه‌سازی در این الگوریتم، این واقعیت آشکار می‌شود که الگوریتم جستجوی گرانشی در هر تکرار[۳۲] سعی در جابجا کردن کنترل شده عامل‌ها در فضای راه‌حل مسئله با در نظر گرفتن تعادلی مناسب بین قابلیت‌های کاوش و بهره‌گیری می‌کند. با ادامه نگاه بالا به پائین، حقایق جدیدی در مورد نحوه کار الگوریتم جستجوی گرانشی بدست می‌آید که در ادامه خواهیم دید:

- جابجا شدن هر عامل در فضای راه‌حل مسئله با استفاده از اِعمالِ پارامتر سرعتِ[۳۳] آن عامل به موقعیت فعلی آن، انجام می‌شود.
- پارامتر سرعت برای هر عامل، خود با استفاده از دو مقدار محاسبه می‌شود: ۱) سرعتِ مستقل[۳۴] (یا غیر وابسته) عامل و ۲) سرعتِ وابسته[۳۵] عامل.
- سرعتِ مستقل برای هر عامل، بدون تاثیرگذاری دیگر اعضای جمعیت فعلی بر روی آن عامل محاسبه می‌شود.
- سرعتِ وابسته برای هر عامل، با تاثیرگذاری دیگر اعضای جمعیت فعلی روی آن عامل و با استفاده از قانون گرانش و قوانین حرکت طبق رابطه (۸) محاسبه می‌شود.
- برای محاسبه سرعتِ وابسته هر عامل با استفاده از قانون گرانش و قوانین حرکت، به عملگرهایی برای محاسبه: ۱) فاصله[۳۶] دو عامل در فضای راه‌حل مسئله ۲) تفاوت[۳۷] دو عامل در فضای راه‌حل مسئله و ۳) ضرب یک عدد حقیقی در تفاوت بدست آمده برای دو عامل، نیازمندیم.

۴-۲- بازتعریف مفاهیم و عملگرهای الگوریتم جستجوی گرانشی برای مسئله فروشنده دوره‌گرد

با توجه به آنچه در بخش ۴-۱ تشریح شد، تمام مفاهیم و عملگرهایی از الگوریتم جستجوی گرانشی که برای حل مسئله فروشنده دوره‌گرد نیاز به بازتعریف دارند را می‌توان به سه دسته زیر تقسیم کرد:

الف) مفهوم فضای راه‌حل مسئله فروشنده دوره‌گرد و مفهوم جابجایی در این فضا.

ب) مفهوم سرعتِ مستقل (غیروابسته) یک عامل، چگونگی محاسبه آن برای یک عامل بدون تاثیرپذیری از دیگر عامل‌های جمعیت فعلی و عملگر اِعمالِ سرعتِ مستقل یک عامل به موقعیت فعلی آن عامل.

ج) مفهوم سرعتِ وابسته یک عامل، چگونگی محاسبه این سرعت برای یک عامل با استفاده از عامل‌های عضو مجموعه *Kbest* بر اساس قانون گرانش و قوانین حرکت و عملگر اِعمالِ سرعتِ وابسته یک عامل به موقعیت فعلی آن عامل.

در ادامه به تشریح چگونگی بازتعریف هر یک از موارد بالا در الگوریتم جستجوی گرانشی برای حل مسئله فروشنده دوره‌گرد پرداخته می‌شود.

۴-۲-۱- فضای راه‌حل مسئله فروشنده دوره‌گرد و مفهوم جابجایی در این فضا

همانطور که در تشریح مسئله فروشنده دوره‌گرد گفته شد، در این مسئله هدف پیدا کردن کم‌هزینه‌ترین سفر (تور) ممکن از میان مجموعه سفرهای مجاز مسئله است. با توجه به اینکه هر راه‌حل برای مسئله فروشنده دوره‌گرد یک سفر می‌باشد، لذا فضای راه‌حل این مسئله را می‌توان به عنوان "فضای سفرهای مسئله" در نظر گرفت.

همچنین چنانکه در تشریح الگوریتم جستجوی گرانشی گفته شد، حرکت در فضای راه‌حل یک مسئله به معنای جابجایی یک عامل از یک راه‌حل ممکن به یک راه‌حل ممکن دیگر است. بنابراین، جابجایی در فضای راه‌حل مسئله فروشنده دوره‌گرد را به معنای "حرکت از یک سفر مجاز به یک سفر مجاز دیگر" در نظر می‌گیریم.

با توجه به تعریف فضای راه‌حل و همچنین حرکت در فضای راه‌حل مسئله فروشنده دوره‌گرد، روند حل مسئله فروشنده دوره‌گرد با استفاده از الگوریتم جستجوی گرانشی را می‌توان در یک نگاه بالا به پائین اینگونه تعریف کرد: "جابجا کردن کنترل شده عامل‌ها در فضای سفرها در هر تکرار، با در نظر گرفتن قابلیت‌های کاوش و بهره‌گیری."

۴-۲-۲- سرعتِ مستقل یک عامل، چگونگی محاسبه آن برای یک عامل و چگونگی اِعمالِ آن به یک عامل

چنانکه قبلا نیز بیان شد، سرعتِ مستقل یک عامل بر خلاف سرعتِ وابسته که از تاثیرگذاری دیگر اعضای جمعیت فعلی بر این عامل محاسبه می‌شود، بدون دخالت دیگر اعضای جمعیت فعلی محاسبه می‌شود. پس از بررسی‌ها و آزمایشهای به عمل آمده، به نظر می‌رسد که یکی از بهترین گزینه‌های ممکن برای اِعمالِ سرعتِ مستقل به یک عامل در مسئله فروشنده دوره‌گرد، استفاده از جستجوگرهای محلی[۳۸] است. جستجوگرهای محلی، الگوریتم‌هایی هستند که یک راه‌حل را به عنوان ورودی گرفته و سعی می‌کنند مستقل از هر پارامتر دیگری و فقط با جایگزین کردن راه‌حل ورودی با راه‌حل‌های بهتری که در همسایگی آن راه‌حل قرار دارند، در صورت امکان کیفیت راه‌حل ورودی را بهبود بخشند.

در این مقاله، از جستجوگر محلی LK[۳۹] [۲۵] که یکی از مشهورترین الگوریتم‌های بهبود دهنده سفر برای مسئله فروشنده دوره‌گرد متقارن و از خانواده *r*-opt است، برای بازتعریف مفهوم سرعتِ مستقل در الگوریتم جستجوی گرانشی گسسته پیشنهادی استفاده شده است. الگوریتم LK با توجه به روش‌های استفاده شده در پیاده‌سازی آن، ممکن است کارایی متفاوتی را از خود نشان دهد. برای مثال، مرجع [۲۶] در بر گیرنده گزارشهایی از روش‌های پیاده‌سازی مختلف این الگوریتم می‌باشد. در این مقاله، از نسخه پیاده‌سازی شده الگوریتم LK در مرجع [۲۷] استفاده شده است.

۴-۲-۳- سرعتِ وابسته یک عامل، چگونگی محاسبه آن برای یک عامل و چگونگی اِعمال آن به یک عامل

برخلاف سرعتِ مستقل، سرعتِ وابسته برای یک عامل از تاثیرگذاری دیگر اعضای جمعیت فعلی بر آن عامل محاسبه می‌شود. بر اساس نسخه اصلی الگوریتم جستجوی گرانشی، سرعت وابسته یک عامل عبارت است از "شتابی" که عامل‌های مجموعه *Kbest* به آن عامل وارد می‌کنند. بنابراین، چنانکه مشاهده می‌شود، مفهوم سرعت وابسته به مفهوم شتاب کاهش[۴۰] می‌یابد.

بنابراین، با توجه به اهمیت مفهوم شتاب در جابجایی یک عامل در فضای راه‌حل مسئله، لازم است که مفهوم آن در فضای سفرها در مسئله فروشنده دوره‌گرد بازتعریف شود. با توجه به اینکه اِعمال شتاب به یک سفرِ مجاز باید سفر مجاز دیگری را تولید کند، لذا در وهله اول نیازمند تعریف مفهوم شتاب وارد شده به یک عامل، سپس چگونگی محاسبه شتاب برای آن عامل با استفاده از قانون گرانش و قوانین حرکت، و در نهایت تعریف عملگری که شتابِ یک عامل را به موقعیت فعلی عامل اِعمال کرده و موقعیت جدیدی که نشان‌دهنده یک سفر مجاز جدید است را تولید می‌کند، هستیم. در ادامه به تشریح هر یک از موارد گفته شده پرداخته می‌شود.

مفهوم شتاب وارد شده به یک عامل

چنانکه بیان شد، اِعمال شتاب به یک سفرِ مجاز باید موجب ایجاد سفرِ مجاز دیگری از فضای سفرها شود. با توجه به این موضوع، لازم است مفهوم شتاب بگونه‌ای بازتعریف گردد که تا حد امکان اِعمال آن به یک سفرِ مجاز برای ایجاد یک سفرِ مجازِ دیگر، ساده باشد. برای این منظور، در این مقاله از مفاهیم **عملگر جابجایی**[۴۱] و **دنباله جابجایی**[۴۲] [۳۵] برای بازتعریف مفهوم شتاب در الگوریتم جستجوی گرانشی گسسته پیشنهادی استفاده شده است.

فرض کنید π یک سفر مجاز از مسئله فروشنده دوره‌گرد و جایگشت $P = (c_{\pi(1)}, c_{\pi(2)}, \ldots, c_{\pi(n)})$ نشان‌دهنده دنباله شهرهای این سفر باشد. در این صورت یک **عملگر جابجایی** بر روی سفر π با نماد $SO = (c_{\pi(i)}, c_{\pi(j)})$ نشان داده شده و به این صورت تعریف می‌شود: "شهر $c_{\pi(i)}$ در سفر π را با شهر $c_{\pi(j)}$ در π جابجا کن". همچنین یک **دنباله جابجایی** بر روی سفر π با نماد $SS = (SO_1, SO_2, \ldots, SO_m)$ نشان داده شده و به این صورت تعریف می‌شود: "عملگرهای جابجایی SO_1، SO_2، ... و SO_m را به ترتیب روی سفر π اعمال کن" [۳۵].

حال با استفاده از مفهوم دنباله جابجایی، مفهوم شتاب به این صورت تعریف می‌شود: "شتاب وارد شده به یک عامل (سفر) از سوی عامل‌های عضو مجموعه

Kbest، عبارت است از یک دنباله جابجایی که عناصر این دنباله قطعا با استفاده از قانون گرانش و قوانین حرکت بدست آمده‌اند. همچنین اِعمال این دنباله جابجایی بر روی موقعیت فعلی یک عامل، موقعیت جدیدی که نشان‌دهنده یک سفر مجاز در فضای سفرها است برای آن عامل را نتیجه می‌دهد".

محاسبه شتاب وارد شده به یک عامل

چنانکه بیان شد، در الگوریتم جستجوی گرانشی شتاب وارد شده به یک عامل (که در الگوریتم پیشنهادی یک دنباله جابجایی خواهد بود) از طریق تاثیرپذیری آن عامل از عامل‌های عضو مجموعه *Kbest* و با استفاده از قانون گرانش و قوانین حرکت محاسبه می‌شود. با دقت در رابطه (۸)، مشاهده می‌شود که برای محاسبه شتاب یک عامل در مسئله فروشنده دوره‌گرد توسط الگوریتم جستجوی گرانشی، نیاز است که برخی از عملگرهای این رابطه برای فضای سفرها بازتعریف شوند. این عملگرها به همراه تعریف مجدد آنها در الگوریتم پیشنهادی در ادامه آورده خواهند شد.

- **عملگر محاسبه تفاوت موقعیت دو عامل در فضای سفرها:** این عملگر را با نماد $SS = T_1 \ominus T_2$ نشان داده و معنای آن این است که تفاوت دو سفر T_1 و T_2 دنباله جابجایی SS است که اِعمال آن به سفر T_1، سفر T_2 را نتیجه می‌دهد.
- **عملگر محاسبه فاصله بین موقعیت دو عامل در فضای سفرها:** این عملگر را با نماد $I = \lvert T_1 \ominus T_2 \rvert$ نشان داده و معنای آن این است که فاصله دو سفر T_1 و T_2 عدد صحیح I است که نشان دهنده تعداد عملگرهای جابجایی موجود در دنباله جابجایی $SS = T_1 \ominus T_2$ است.
- **عملگر محاسبه ضرب یک عدد حقیقی در یک دنباله جابجایی:** این عملگر را با نماد $SS_2 = c \otimes SS_1$ نشان داده و به صورت زیر تعریف می‌کنیم: ضرب عدد حقیقی c در دنباله جابجایی SS_1، دنباله جابجایی SS_2 را نتیجه می‌دهد که اعضای آن، $\lceil c \times \lvert SS_1 \rvert \rceil$ عضو اول SS_1 می‌باشند. لازم به ذکر است که برای $c>1$ فرض می‌کنیم $c=1$ و برای $c<0$ فرض می‌کنیم $c=0$.
- **عملگر جمع m دنباله جابجایی:** این عملگر را با نماد $SS' = SS_1 \odot SS_2 \odot \ldots \odot SS_m$ نشان داده و به صورت زیر تعریف می‌کنیم: جمع m دنباله جابجایی SS_1، SS_2، ... و SS_m، دنباله جابجایی جدید SS' است که اعضای آن اجتماع تمامی عملگرهای جابجایی موجود در m دنباله جابجایی ورودی است.

برای مثال فرض کنید موقعیت عامل‌های M_1 و M_2 به ترتیب برابر با سفرهای $T_1 = (1, 2, 3, 4, 5)$ و $T_2 = (3, 1, 2, 4, 5)$ باشد. در این صورت، تفاوت موقعیت M_1 از M_2 (یعنی $T_1 \ominus T_2$) عبارت است از دنباله جابجایی $SS = ((1, 2), (2, 3))$. همچنین، فاصله موقعیت M_1 از M_2 (یعنی $\lvert T_1 \ominus T_2 \rvert$) برابر است با طول SS یعنی ۲. علاوه بر این، ضرب عدد حقیقی $c=0.3$ در دنباله جابجایی SS برابر است با: $SS_1 = ((1, 2))$. در نهایت، جمع سه دنباله جابجایی (شتاب) SS_1، $SS_2 = ((3, 4), (1, 2))$ و $SS_3 = ((3, 4), (2, 5), (3, 2))$ برابر است با دنباله جابجایی: $SS_4 = ((1, 2), (3, 4), (2, 5), (3, 2))$.

با توجه به تعاریف جدید عملگرها، به راحتی می‌توان رابطه (۸) را برای محاسبه شتاب یک عامل در فضای سفرها بازنویسی کرد. رابطه (۱۱) رابطه بازنویسی شده را نشان می‌دهد (لازم به ذکر است که برای خوانایی بیشتر، در این رابطه نماد سیگما جایگزین نماد $\odot$ شده است):

$$a_i(t) = \sum_{j \in Kbest,\, j \neq i} (rand_j G(t) \frac{M_j(t)}{R_{ij}(t) + \varepsilon}) \otimes (x_j(t) \ominus x_i(t)) , \qquad (۱۱)$$

که در آن $R_{ij}(t)$ فاصله موقعیت‌های x_j و x_i در زمان t است، یعنی: $R_{ij}(t) = \lvert x_j(t) \ominus x_i(t) \rvert$.

اِعمال شتابِ یک عامل به موقعیت فعلی آن

پس از تعریف مفهوم شتاب (که در واقع تقلیل‌یافته مفهوم سرعتِ وابسته در الگوریتم جستجوی گرانشی است) برای فضای سفرها توسط مفهوم دنباله جابجایی و همچنین تشریح روش محاسبه آن برای یک عامل، لازم است تا چگونگی اِعمال شتاب به یک عامل و عملگری که انجام این وظیفه را بر عهده دارد نیز بیان شود. خوشبختانه، در تعریف مفهوم شتاب تمام سعی خود را بر این گذاشته‌ایم که مفهوم شتاب را با الهام از ساختار سفرها تعریف کنیم. از اینرو، تعریف عملگری که موقعیت یک عامل را به همراه شتابِ آن عامل در تکرار فعلی به عنوان ورودی گرفته و باید سفر مجاز جدیدی را به عنوان موقعیت جدید آن عامل تولید کند، امری ساده خواهد بود. فرض کنید موقعیت فعلی عاملِ M_1، سفرِ T_1 و شتابِ محاسبه شده برای این عامل در تکرار فعلی، دنباله جابجایی SS_1 باشد. در این صورت عملگر جابجا کردن موقعیت عامل M_1 در فضای سفرها را با نماد $T_2 = T_1 \oplus SS_1$ نشان داده و به صورت زیر تعریف می‌کنیم: تک تک عملگرهای جابجایی شتابِ SS_1 را به ترتیب به سفر T_1 اِعمال کرده و سفر مجاز T_2 را به عنوان موقعیت جدید عامل M_1 نتیجه می‌دهد. برای مثال، فرض کنید موقعیت فعلی عامل M_1، سفرِ $T_1 = (1, 2, 3, 4, 5)$ و شتاب محاسبه شده برای این عامل، دنباله جابجایی $SS_1 = ((1, 2), (2, 3))$ باشد. در این صورت، اِعمال تک تک عملگرهای جابجایی شتابِ SS_1 بر روی موقعیت فعلی عامل M_1، موقعیت این عامل را به سفر $T_2 = (3, 1, 2, 4, 5)$ تغییر خواهد داد.

۴-۳- الگوریتم جستجوی گرانشی گسسته ترکیبی پیشنهادی برای حل مسئله فروشنده دوره‌گرد متقارن

در بخش ۴-۱ و ۴-۲، به ترتیب به شناخت و بازتعریف آن دسته از مفاهیم و عملگرهایی از الگوریتم جستجوی گرانشی که برای حل مسئله فروشنده دوره‌گرد متقارن نیاز به تعریف مجدد داشتند، پرداخته شد.

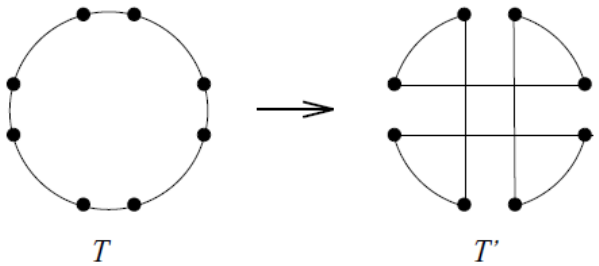
گام ۱)	تولید جمعیت اولیه از عامل‌ها به طور تصادفی.
گام ۲)	ارزیابی شایستگی هر عامل (سفر) و پیدا کردن بهترین و بدترین عامل‌های جمعیت فعلی.
گام ۳)	محاسبه مقدار M_i برای هر عامل i.
گام ۴)	به روز رسانی پارامترهای G و K.
گام ۵)	محاسبه شتاب برای هر عامل با استفاده از رابطه (۱۱).
گام ۶)	اِعمال شتاب (سرعتِ وابسته) هر عامل به آن.
گام ۷)	بهبود کیفیت هر عامل با استفاده از الگوریتم LK (اِعمال سرعتِ مستقل هر عامل به آن).
گام ۸)	اگر میزان تنوع[۴۳] جمعیت با توجه به تکرار فعلی الگوریتم در حد قابل قبولی است، به گام ۱۰ برو.
گام ۹)	اِعمال عملگر double bridge بر روی یکی از عامل‌های تکراری و سپس اِعمال الگوریتم LK بر روی آن عامل.
گام ۱۰)	اگر شرایط مطلوب حاصل نشده است، به گام ۲ برو.
گام ۱۱)	ارائه بهترین راه‌حل پیدا شده به عنوان خروجی.

شکل ۲- گام‌های اصلی الگوریتم پیشنهادی برای حل مسئله فروشنده دوره‌گرد متقارن

در این بخش به تشریح گام‌های کلی الگوریتم پیشنهادی پرداخته می‌شود. شکل ۲ گام‌های اصلی الگوریتم جستجوی گرانشی گسسته ترکیبی پیشنهادی را نشان می‌دهد.

چنانکه در شکل ۲ دیده می‌شود، الگوریتم پیشنهادی از ۱۱ گام تشکیل شده است. در گام ۱ از الگوریتم، جمعیت اولیه عامل‌ها به طور کاملا تصادفی ساخته می‌شود. سپس در گام ۲، شایستگی هر عضو از جمعیت ارزیابی شده و بر اساس میزان شایستگی هر عضو از جمعیت، بهترین و بدترین عضو جمعیت مشخص می‌شود. در گام ۳، جرم گرانشی عامل i (یعنی M_i) محاسبه می‌شود. در گام ۴، از روش کاهش خطی برای به روز رسانی مقدار پارامترهای G و K استفاده شده است. در گام ۵، شتاب هر عامل (یا سرعت وابسته هر عامل) با استفاده از رابطه (۱۱) محاسبه خواهد شد. در گام ۶، شتاب محاسبه شده برای هر عامل به آن اِعمال می‌شود (طریقه اِعمال آن به طور کامل در بخش ۴-۲-۳ شرح داده شده است). در گام ۷، سرعت مستقل هر عامل به آن اِعمال خواهد شد (طریقه اِعمال آن به طور کامل در بخش ۴-۲-۲ شرح داده شده است). چنانکه قبلا نیز گفته شد، اِعمال سرعت مستقل هر عامل به آن توسط الگوریتم LK انجام خواهد شد.

گام ۸ از الگوریتم، به بررسی این موضوع می‌پردازد که آیا تنوع جمعیت در تکرار فعلی با توجه به مصالحه‌ای که باید بین قابلیت‌های کاوش و بهره‌گیری صورت گیرد، در شرایط مناسب هست یا خیر. بدون شک اگر در تکرارهای اولیه تنوع جمعیت پائین باشد نیاز است که با روشی مناسب تنوع جمعیت را برای حفظ قابلیت کاوش الگوریتم بالا نگه داشت. برای این منظور، در صورتی که تنوعِ جمعیت وضع مطلوبی نداشته باشد، در گام ۹ ابتدا عملگر double bridge [۲۸] را بر روی یکی از عامل‌های تکراری جمعیت اِعمال کرده و سپس برای بهبود کیفیت عامل مورد نظر، الگوریتم LK را بر روی آن اِعمال می‌کنیم. وظیفه عملگر double bridge به هم ریختن حساب شده‌ی یک سفر است و با انجام این کار می‌تواند شکل کلی آن سفر را تغییر دهد. شکل ۳ یک سفر را قبل و بعد از اِعمال عملگر double bridge روی آن نشان می‌دهد.



شکل ۳- تولید سفر T' از سفر T پس از اِعمال عملگر double bridge بر روی آن

گام ۱۰ از الگوریتم پیشنهادی، به بررسی اینکه آیا شرایط توقف الگوریتم فرا رسیده است یا خیر، می‌پردازد. برای بررسی شرط توقف یک الگوریتمِ مبتنی بر تکرار[۴۴]، پارامترهای مختلفی وجود دارد که می‌توان هر یک از آنها و یا ترکیبی از آنها را در نظر گرفت (شرایط توقف الگوریتم پیشنهادی در قسمت پیاده‌سازی الگوریتم شرح داده خواهد شد). اگر شرایط توقف برآورده شده بود، الگوریتم به گام ۱۱ و اگر این شرایط برآورده نشده بود، الگوریتم به گام ۲ خواهد رفت. در نهایت، گام ۱۲ بهترین راه‌حل پیدا شده را به عنوان خروجی بر می‌گرداند.

۵- نتایج

برای بررسی کارایی الگوریتم جستجوی گرانشی پیشنهادی، آزمایش‌هایی بر روی تعدادی از نمونه‌های متقارن مسئله فروشنده دوره‌گرد کتابخانه محک استاندارد TSPLIB [۲۹] صورت گرفته است. در تمام این آزمایش‌ها، الگوریتم پیشنهادی با زبان C پیاده‌سازی و بر روی یک کامپیوتر دارای پردازنده پنتیوم ۴ با فرکانس کاری ۳.۰ گیگاهرتز و حافظه اصلی ۵۱۲ مگابایت اجرا شده است. در ادامه این بخش، به بررسی نمونه‌های استفاده شده برای ارزیابی الگوریتم پیشنهادی، شرایط پیاده‌سازی الگوریتم پیشنهادی، بررسی نتایج بدست آمده توسط الگوریتم پیشنهادی و مقایسه آن با دیگر الگوریتم‌ها پرداخته می‌شود و در نهایت تحلیلی از فرایند حل مسئله فروشنده دوره‌گرد توسط الگوریتم پیشنهادی آورده خواهد شد.

۵-۱- نمونه‌های استفاده شده برای ارزیابی کارایی الگوریتم پیشنهادی

برای ارزیابی کارایی الگوریتم جستجوی گرانشی گسسته ترکیبی، ۲۳ نمونه متقارن از نمونه‌های تقریبا بزرگ کتابخانه استاندارد TSPLIB انتخاب شده است. این نمونه‌ها به همراه مقدار بهینه‌شان در جدول ۱ آورده شده‌اند. چنانکه در جدول ۱ دیده می‌شود، نام هر نمونه با یک عدد خاتمه می‌یابد که این عدد بیانگر تعداد شهرهای آن نمونه می‌باشد. برای مثال، نمونه dsj1000 یک نمونه از مسئله فروشنده دوره‌گرد با ۱۰۰۰ شهر می‌باشد. تنوع نمونه‌های مورد بررسی از ۱۰۰۰ شهر تا ۷۳۹۷ شهر می‌باشد.

جدول ۱- نمونه‌های استفاده شده برای ارزیابی الگوریتم پیشنهادی

نام نمونه	مقدار بهینه
dsj1000	۱۸۶۵۹۶۸۸
pr1002	۲۵۹۰۴۵
u1060	۲۲۴۰۹۴
vm1084	۲۳۹۲۹۷
pcb1173	۵۶۸۹۲
d1291	۵۰۸۰۱
rl1304	۲۵۲۹۴۸
rl1323	۲۷۰۱۹۹
nrw1379	۵۶۶۳۸
fl1400	۲۰۱۲۷
fl1577	از ۲۲۲۰۴ تا ۲۲۲۴۹
vm1748	۳۳۶۵۵۶
u1817	۵۷۲۰۱
rl1889	۳۱۶۵۳۶
d2103	از ۷۹۹۵۲ تا ۸۰۴۵۰
u2152	۶۴۲۵۳
pr2392	۳۷۸۰۳۲
pcb3038	۱۳۷۶۹۴
fl3795	از ۲۸۷۲۳ تا ۲۸۷۷۲
fnl4461	۱۸۲۵۶۶
rl5915	از ۵۶۵۰۴۰ تا ۵۶۵۵۳۰
rl5934	از ۵۵۴۰۷۰ تا ۵۵۶۰۴۵
pla7397	۲۳۲۶۰۷۲۸

برای برخی از نمونه‌های جدول ۱، مقدار دقیق بهینه برای آن نمونه مشخص نیست. از اینرو، در قسمت مقدار بهینه برای آن نمونه، کران پائین و بالای شناخته شده برای طول سفر بهینه آن نمونه آورده شده است. برای مثال، عبارت " از ۲۲۲۰۴ تا ۲۲۲۴۹" برای بیان مقدار بهینه یک نمونه، به این معناست که طول سفر بهینه برای آن نمونه حداقل ۲۲۲۰۴ و حداکثر ۲۲۲۴۹ می‌باشد.

۵-۲- شرایط پیاده‌سازی الگوریتم پیشنهادی

پیاده‌سازی الگوریتم پیشنهادی با زبان C و اجرای آن بر روی یک کامپیوتر دارای پردازنده پنتیوم ۴ با فرکانس کاری ۳.۰ گیگاهرتز و حافظه اصلی ۵۱۲ مگابایت صورت گرفته است. در پیاده‌سازی انجام شده، اندازه جمعیت برای الگوریتم برابر با ۲۰ ($S = 20$) و مقدار اولیه تعداد عامل‌های تاثیرگذار بر روی دیگر عامل‌ها برابر با ۱۰ ($K_0 = 10$) در نظر گرفته شده است. همچنین، مقدار اولیه برای پارامتر ضریب گرانش برابر با $\frac{n}{5}$ ($G_0 = \frac{n}{5}$) در نظر گرفته شده است که n تعداد شهرهای نمونه مورد آزمایش می‌باشد.

در نهایت، برای شرایط توقف الگوریتم پیشنهادی، ترکیب فصلی دو پارامتر زیر در نظر گرفته شده است:

- رسیدن تعداد تکرار الگوریتم به ۲۰۰۰بار
- رسیدن زمان اجرای الگوریتم به: ۳۰۰ ثانیه برای نمونه‌هایی با اندازه بین ۱۰۰۰ تا ۵۰۰۰ شهر، ۱۰۰۰ ثانیه برای نمونه‌هایی با اندازه بین ۵۰۰۰ تا ۷۰۰۰ شهر، و ۲۰۰۰ ثانیه برای نمونه‌هایی با اندازه بزرگتر از ۷۰۰۰ شهر.

۵-۳- نتایج پیاده‌سازی و مقایسه

جدول ۲ نتایج پیاده‌سازی الگوریتم پیشنهادی بر روی ۲۳ نمونه مسئله فروشنده دوره‌گرد نشان داده شده در جدول ۱ را نشان می‌دهد. در جدول ۲، برای هر الگوریتم دو ستون دیده می شود که در ادامه به بررسی آنها پرداخته می‌شود:

- "بهترین" یا *Err*1: این ستون خطای نسبی بهترین سفر دیده شده در ۲۰ اجرای مستقل الگوریتم برای یک نمونه از مسئله فروشنده دوره‌گرد را نشان می‌دهد. این خطا با استفاده از رابطه (۱۲) محاسبه می‌شود:

$$Err1 = \frac{(best - \mathrm{Opt})}{\mathrm{Opt}} \times 100\% \ , \qquad (۱۲)$$

که در آن *best* عبارت است از طول بهترین راه‌حل بدست آمده توسط الگوریتم برای نمونه مورد بررسی در ۲۰ اجرای مستقل الگوریتم و Opt عبارت است از طول بهترین راه‌حل نمونه مورد بررسی.

- "میانگین" یا *Err*2: این ستون خطای نسبی میانگینِ بهترین سفرهای دیده شده توسط الگوریتم در ۲۰ اجرای مستقل الگوریتم برای یک نمونه از مسئله فروشنده دوره‌گرد را نشان می‌دهد. این خطا با استفاده از رابطه (۱۳) محاسبه می‌شود:

$$Err2 = \frac{(average - \mathrm{Opt})}{\mathrm{Opt}} \times 100\% \ , \qquad (۱۳)$$

که در آن *average*، میانگین طول بهترین راه‌حل بدست آمده برای نمونه مورد بررسی در هر اجرای مستقلِ الگوریتم، و Opt طول بهترین راه‌حل نمونه مورد بررسی می‌باشد.

همچنین، در جدول ۳ به مقایسه میانگین نتایج بدست آمده توسط الگوریتم پیشنهادی با میانگین نتایج بدست آمده توسط تعدادی از جدیدترین الگوریتم‌های حل مسئله فروشنده دوره‌گرد پرداخته شده است. این الگوریتم‌ها عبارتند از:

- یک الگوریتم ادغام کننده سفر[۴۵] [۳۰]،
- یک الگوریتم زنبور عسل [۳۱]،
- یک الگوریتم ذوب شبیه‌سازی شده تطبیقی با جستجوی حریصانه[۴۶] [۳۲]،
- یک الگوریتم ممتیک[۴۷] [۳۳]،
- یک شبکه عصبی ممتیک [۳۴]،
- یک الگوریتم بهینه‌ساز جمعیت ذرات[۴۸] گسسته [۳۶]، و
- یک شبکه عصبی سازنده-بهینه‌ساز[۴۹] [۳۷].

جدول ۲- نتایج بدست آمده توسط الگوریتم جستجوی گرانشی گسسته ترکیبی پیشنهادی

نمونه	الگوریتم پیشنهادی	
	بهترین (*Err*1)	میانگین (*Err*2)
dsj1000	۰.۰۰۱۲	۰.۰۰۲۴
pr1002	۰	۰.۰۰۱۰
u1060	۰	۰
vm1084	۰	۰
pcb1173	۰	۰.۰۰۰۷
d1291	۰	۰
rl1304	۰	۰
rl1323	۰	۰.۰۰۷۶
nrw1379	۰	۰.۰۰۵۰
fl1400	۰	۰
fl1577	۰	۰.۰۱۶۷
vm1748	۰	۰
u1817	۰	۰.۰۶۵۰
rl1889	۰	۰.۰۰۹۴
d2103	۰	۰.۰۰۳۷
u2152	۰	۰.۰۵۰۴
pr2392	۰	۰.۰۰۱۲
pcb3038	۰.۰۰۷۱	۰.۰۳۷۱
fl3795	۰	۰.۰۰۹۰
fnl4461	۰.۰۲۱۴	۰.۰۳۵۰
rl5915	۰.۰۱۴۸	۰.۰۴۶۸
rl5934	۰.۰۰۱۱	۰.۰۳۴۵
pla7397	۰.۰۰۴۹	۰.۰۱۸۹
میانگین	۰.۰۰۲۵	۰.۰۱۷۹

در انتها لازم است یادآور شویم که هرچند امکان رتبه‌بندی این الگوریتم‌ها بر اساس میانگین سرعت پیدا کردن سفرها نیز وجود دارد، اما به علت اینکه سرعت سفرهای پیدا شده توسط هر یک از این الگوریتم‌ها، علاوه بر خود الگوریتم به عامل‌های دیگر از جمله کامپایلر و سخت‌افزار کامپیوتر نیز بستگی دارد، لذا به نظر

می‌رسد معیار میانگین سرعت برای رتبه‌بندی این الگوریتم‌ها معیار چندان مناسبی نباشد.

جدول ۳- رتبه‌بندی بین الگوریتم پیشنهادی و دیگر الگوریتم‌های حل مسئله فروشنده دوره‌گرد

Rank	Method	Average
1	**Proposed DGSA**	0.0179
2	PSO-LK [36]	0.0201
3	Tourmerge [30]	0.0466
4	HBMO-TSP [31]	0.0644
5	Inver-over & LK MA [33]	0.2464
6	ASA-GS [32]	2.5025
7	CONN [37]	8.2081
8	Memetic SOM [34]	8.8407

۵-۴- تحلیل کارایی الگوریتم جستجوی گرانشی پیشنهادی

چنانکه در رتبه‌بندی جدول ۳ دیده شد، میانگین خطای نسبی بدست آمده توسط الگوریتم جستجوی گرانشی پیشنهادی در حل نمونه‌های جدول ۱، بهتر از سایر الگوریتم‌ها است. نکته مهمی که در این رتبه‌بندی دیده می‌شود این است که در تعدادی از الگوریتم‌های جدول ۳ نیز همانند الگوریتم پیشنهادی ما از الگوریتم LK به عنوان یک بهبود دهنده سفر استفاده شده است، اما چنانکه می‌بینیم، میانگین خطای نسبی الگوریتم پیشنهادی ما بهتر از تمام این الگوریتم‌هاست. این موضوع خود بیانگر این نکته است که الگوریتم پیشنهادی به خوبی توانسته است فرایند کاوش در فضای راه‌حل‌های مسئله فروشنده دوره‌گرد را انجام دهد.

چنانکه قبلا نیز اشاره شد، در الگوریتم جستجوی گرانشی پیشنهادی سه گام زیر به طور تکراری اجرا می‌شود:

۱) محاسبه سرعتِ وابسته برای هر عامل و اِعمال این سرعت به عامل.

۲) اِعمال سرعتِ مستقل به هر عامل (باعث قرار گرفتن عامل‌ها در بهینه‌های محلی می‌شود).

۳) تنظیم تنوع جمعیت در صورتیکه تنوع جمعیت به اندازه کافی نباشد.

اجرای این سه گام در طول تکرارهای متوالی، باعث می‌شود که الگوریتم کاوش تقریبا موفقیت آمیزی را در فضای راه‌حل‌های بهینه محلی مسئله فروشنده دوره‌گرد داشته باشد. از اینرو، الگوریتم سعی می‌کند با حفظ قابلیت کاوش، جمعیت را به سمت راه‌حل‌های بهینه محلی با کیفیت بالا حرکت دهد. لذا احتمال پیدا کردن جواب‌های بسیار نزدیک به بهینه و حتی جواب بهینه افزایش می‌یابد. بنابراین، بطور کلی علت اصلی موفقیت الگوریتم جستجوی گرانشی پیشنهادی را می‌توان در یک جمله خلاصه کرد: "کاوش مناسب در فضای راه‌حل‌های بهینه محلی[۵۰] مسئله فروشنده دوره‌گرد".

۶- نتیجه‌گیری

تمام الگوریتم‌هایی که تا امروز برای حل دقیق نمونه‌های مسائل NP-hard ارائه شده‌اند، دارای پیچیدگی زمانی نمایی هستند. با توجه به اینکه در اغلب موارد، پیدا کردن جواب بهینه‌ی نمونه‌های بزرگ مسائل NP-hard ضرورتی ندارد، می‌توان برای حل این مسائل از روش‌های حل تقریبی مثل الگوریتم‌های فرا ابتکاری استفاده کرد که معمولا جواب‌های نزدیک به بهینه را در یک مدت زمان معقول برای مسئله ارائه می‌کنند. در این مقاله، نشان داده شده است که با گسسته‌سازی عملگرهای اصلی الگوریتم جستجوی گرانشی که یکی از جدیدترین روش‌های فرا ابتکاری است، می‌توان از آن در حل نمونه‌های با اندازه بزرگ مسئله فروشنده دوره‌گرد که یکی از سخت‌ترین مسائل با فضای جستجوی ترکیبیاتی است، استفاده کرد. نتایج پیاده‌سازی الگوریتم پیشنهادی و مقایسه آن با سایر الگوریتم‌ها، نشان دهنده کارایی مناسب الگوریتم پیشنهادی می‌باشد. در کارهای آینده بر روی افزایش کارایی الگوریتم پیشنهادی و استفاده از آن در حل دیگر مسائل بهینه‌سازی ترکیبیاتی متمرکز خواهیم شد.

مراجع

[۱] ح. نظام‌آبادی‌پور، الگوریتم وراثتی: مفاهیم پایه و مباحث پیشرفته، انتشارات دانشگاه شهید باهنر کرمان، ۱۳۸۹.

[۲] م. ب. دولتشاهی، ح. نظام‌آبادی‌پور و م. ماشین‌چی، "حل مسئله فروشنده دوره‌گرد با استفاده از الگوریتم جستجوی گرانشی،" شانزدهمین کنفرانس ملی انجمن کامپیوتر ایران، دانشگاه صنعتی شریف، ۱۳۸۹.

[3] M. R. Garey, and D. S. Johnson, *Computers and Intractability: AGuide to the Theory of NP-Completeness*, W. H. Freeman press, 1979.

[4] H. P. Christos, *ComputationalComplexity*, Addison-Wesley Publishing, 1995.

[5] O. Goldrich, *P, NP, and NP-Completeness: The Basic of Computational Complexity*, Cambridge University Press, 2010.

[6] F. V. Fomin, and D. Kratsch, *Exact Exponential Algorithms*, Springer, 2010.

[7] R. J. Lipton, *The P=NP Question and Godel's Lost Letter*, Springer, 2010.

[8] G. Gutin, and A. P. Punnen, *Traveling Salesman Problem and Its Variations*, Kluwer Academic Publishers, Dordrecht, 2002.

[9] J. Hromkovic, *Algorithmicsfor Hard Problems: Introductionto Combinatorial Optimization, Randomization, Approximation, and Heuristics*, Springer, 2004.

[10] H. H. Hoos, and T. Stutzle, *StochasticLocal Search: Fundations and Applications*, Morgan Kaufmann Publishers, 2005.

[11] Talbi, and El-Ghazali, *Metaheuristics: From Design to Implementation*, John Wiley & Sons, 2009.

[12] D. Karapetyan, *Design, Evaluation and Analysis of Combinatorial Optimization Heuristic Algorithms*, Ph.D. Thesis, University of London, Royal Holloway College, 2010.

[13] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, *"GSA: A Gravitational Search Algorithm,"* Information Sciences, vol. 179, pp. 2232–2248, 2009.

[14] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, *"BGSA: binary gravitational search algorithm,"* Natural Computing, vol. 9, pp. 727-745, 2010.

[15] G. Carpaneto, and P. Toth, *"Some new branching and bounding criteria for the asymmetric travelling salesman problem,"* Management Science, vol. 26, pp. 736-743, 1980.

[16] D. L. Miller, and J. F. Pekny, *"Exact solution of large asymmetric traveling salesman problems,"* Science, vol. 251, pp. 754-761, 1991.

[17] S. Arora, *"Polynomial Time Approximation Schemes for Euclidian Traveling Salesman and Other Geometric Problems,"* Journal of the ACM, vol. 45, no. 5, pp. 753-782, 1998.

[18] D. J. Rosenkrantz, R. E. Stearens, and P. M. Lowis, *"An analyze of several heuristics for the traveling salesman problems,"* SIAM Journal on computing, vol. 6, pp. 563-581, 1977.

[19] S. Lin, *"Computer solutions of the traveling salesman problem,"* Bell System Computer Journal, vol. 44, pp. 2245-2269, 1965.

[20] S. Kirpatrick, C. D. Gelatt, and M. P. Vecchi, *"Optimization by simulated annealing,"* Science, pp. 671-680, 1983.

[21] F. Glover, *"Tabu search,"* Part II, ORSA Journal on Computing, pp. 4-32, 1990.

[22] K. Mathias, and D. Whitley, *"Genetic operators, the fitness landscape, and the traveling salesman problem,"* Parallel Problem Solving from Nature, Elsevier, pp. 219–228, 1992.

[23] M. Clerc, *"Discrete particle swarm optimization, illustrated by the traveling salesman problem,"* Studies in Fuzziness and Soft Computing New optimization techniques in engineering, Springer, Berlin, vol. 141, pp. 219–239, 2004.

[24] M. Gendreau, A. Hertz, and G. Laporte, *"New insertion and post-optimization procedures for the traveling salesman problem,"* Operation Research, forthcoming, 1992.

[25] S. Lin, and B. Kernighan, *"An effective heuristic algorithm for the traveling-salesman problem,"* Operations Research, vol. 21, pp. 498-516, 1973.

[26] D. S. Johnson, and L. A. McGeoch, *"Experimental analysis of heuristics for the STSP,"* In: Traveling Salesman Problem and Its Variations, Guttin, G., Punnen, A. P. (Eds.), pp. 369-443, ISBN: 1402006640, Kluwer Academic, Dordrecht, 2002.

[27] D. Applegate, R. Bixby, V. Chvatal, and W. Cook, *"Finding Tours in the TSP,"* Technical Report TR99-05, Department of Computational and Applied Mathematics, Rice University, 1999.

[28] O. Martin, S. W. Otto, and E. W. Felten, *"Large-step Markove Chains for the traveling salesman problem,"* Complex Systems, vol. 5, pp. 299-326, 1991.

[29] G. Reinelt, *"TSPLIB—A traveling salesman problem library,"* ORSA Journal on Computing, vol. 3, pp. 376–384, 1991.

[30] W. J. Cook, and P. Seymour, *"Tour Merging via Branch-decomposition,"* INFORMS Journal on Computing, vol. 15, pp. 233-248, 2003.

[31] Y. Marinakis, M. Marinaki, and G. Dounias, *"Honey bees mating optimization algorithm for the Euclidian traveling salesman problem,"* Information Sciences, vol. 181, pp. 4684-4698, 2011.

[32] X. Geng, Z. Chen, W. Yang, D. Shi, and K. Zhao, *"Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search,"* Applied Soft Computing, vol. 11, pp. 3680-3689, 2011.

[33] Y. T. Wang, J. Q. Li, K. Z. Gao, and Q. K. Pan, *"Memetic algorithm based on improved Inver-Over operator and Lin-Kernighan local search for the Euclidean traveling salesman problem,"* Computer and Mathematics with Applications, vol. 62, pp. 2743-2754, 2011.

[34] J. C. Creput, and A. Koukam, *"A memetic neural network for the Euclidean traveling salesman problem,"* Neurocomputing, vol. 72, pp. 1250-1264, 2009.

[35] K. P. Wang, L. Huang, C. G. Zhou, and W. Pang, *"Particle swarm optimization for traveling salesman problem,"* Proceeding of the 2nd ICMLC, Xi'an, China, vol. 3, IEEE Press, pp. 1583–1585, 2003.

[36] E. F. G. Goldbarg, G. R. Souza, and M. C. Goldbarg, *"Particle swarm for the traveling salesman problem,"* Proceedings of the EvoCOP 2006, Gottlieb, J, Raidl, G. R. (Ed.), Lecture Notes in Computer Science, vol. 3906, pp. 99-110, ISBN: 3540331786, Budapest, Hungary, Springer, Berlin, 2006.

[37] M. Saadatmand-Tarzjan, M. Khademi, M. R. Akbarzadeh-T, and H. A. Moghaddam, *"A novel constructive-optimizer neural network for the traveling salesman problem,"* IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics, vol. 37, pp. 754–770, 2007.

محمدباقر دولتشاهی در سال ۱۳۶۶ در شهر خرم‌آباد متولد شد. او فارغ‌التحصیل کارشناسی ارشد رشته علوم کامپیوتر گرایش سیستم‌های هوشمند از دانشگاه شهید باهنر کرمان در سال ۱۳۹۰ می‌باشد. پایان‌نامه کارشناسی ارشد نامبرده در زمینه حل مسائل بهینه‌سازی ترکیبیاتی با استفاده از الگوریتم‌های فرا ابتکاری و تحت راهنمایی آقایان دکتر ماشاالله ماشین‌چی و دکتر حسین نظام‌آبادی‌پور انجام شده است. زمینه‌های پژوهشی مورد علاقه وی عبارتند از: نظریه پیچیدگی محاسباتی، مسائل بهینه‌سازی ترکیبیاتی، رایانش نرم، و شبکه‌های کامپیوتری.

آدرس پست‌الکترونیکی ایشان عبارت است از:
mb.dowlatshahi@yahoo.com

حسین نظام‌آبادی‌پور کارشناسی و کارشناسی ارشد خود را در مهندسی برق- الکترونیک به ترتیب از دانشگاه شهید باهنر کرمان و دانشگاه تربیت مدرس در سال‌های ۱۳۷۷ و ۱۳۷۹ دریافت کرد. او سپس مدرک دوره دکترای خود را در مهندسی برق- الکترونیک از دانشگاه تربیت مدرس در سال ۱۳۸۳ اخذ کرد و در همان سال در سمت استادیاری در بخش مهندسی برق دانشگاه شهید باهنر کرمان مشغول به فعالیت شد. وی در حال حاضر در سمت دانشیاری در این دانشگاه مشغول به خدمت است. زمینه‌های پژوهشی مورد علاقه نامبرده عبارتند از: الگوریتم‌های ابتکاری، پردازش تصویر، بازشناسی الگو، و رایانش نرم.

آدرس پست آدرس پست‌الکترونیکی ایشان عبارت است از:
nezam@mail.uk.ac.ir

ماشاالله ماشین‌چی در سال ۱۳۲۹ در شهر کرمان متولد شد. او کارشناسی و کارشناسی ارشد خود را در رشته آمار به ترتیب از دانشگاه فردوسی مشهد و دانشگاه شیراز در سال‌های ۱۳۵۳ و ۱۳۵۵ دریافت کرد. او سپس مدرک دکترای خود را در رشته ریاضیات از دانشگاه واسدا در ژاپن در سال ۱۳۶۶ دریافت کرد. وی در حال حاضر در سمت استاد تمامی در دانشکده ریاضی و کامپیوتر دانشگاه شهید باهنر کرمان مشغول به خدمت است. زمینه‌های پژوهشی مورد علاقه نامبرده عبارت است از: ریاضیات فازی به خصوص در آمار، تصمیم‌گیری، و سیستم‌های جبری.

آدرس پست‌الکترونیکی ایشان عبارت است از:
mashinchi@mail.uk.ac.ir

اطلاعات بررسی مقاله:

تاریخ ارسال: ۹۰/۲/۲۰
تاریخ اصلاح: ۹۰/۱۲/۱۵
تاریخ قبول شدن: ۹۰/۱۲/۲۰
نویسنده مرتبط: محمدباقر دولتشاهی، دانشکده ریاضی و علوم کامپیوتر، دانشگاه شهید باهنر کرمان، کرمان، ایران.

[1] Combinatorial Optimization
[2] Non-Deterministic Polynomial-Time Hard
[3] Feasible Solution
[4] Computational Complexity Theory
[5] Traveling Salesman Problem (TSP)
[۶] این مسئله یکی از مهمترین مسائل پاسخ داده نشده در زمینه "نظریه پیچیدگی" می‌باشد. این مسئله به این موضوع می‌پردازد که: آیا کلاس پیچیدگی زمانی چندجمله‌ای غیرقطعی (non-deterministic polynomial-time) با کلاس پیچیدگی زمانی چندجمله‌ای قطعی (یا به اختصار پیچیدگی زمانی چندجمله‌ای) برابر است یا خیر؟ مسئله فروشنده دوره‌گرد یکی از مشهورترین مسائلی است که وجود یک الگوریتم غیرقطعی با پیچیدگی زمانی چندجمله‌ای برای آن اثبات شده است، اما وجود یک الگوریتم قطعی با پیچیدگی زمانی چندجمله‌ای برای آن تابحال نه اثبات و نه رد شده است.
[7] Near-Optimal
[8] Meta-Heuristic
[9] Exploration
[10] Exploitation
[11] Gravitational Search Algorithm (GSA)
[12] Continuous Optimization
[13] Computational Complexity Theory
[14] Tour
[15] Symmetric TSP (STSP)
[16] Asymmetric TSP (ATSP)
[17] Permutation
[18] Exact Algorithms
[19] Heuristic Algorithms
[20] Branch and Bound
[21] Approximation Algorithms
[22] Tour Construction Algorithms
[23] Nearest Neighbour Algorithm
[24] Greedy Algorithm
[25] Tour Improvement Algorithms
[26] Stochastic
[27] Population-based
[28] Mass
[29] Solution Space
[30] Dicision Space
[31] Top-Down
[32] Iteration
[33] Velocity
[34] Independent Velocity
[35] Dependent Velocity
[36] Distance
[37] Subtraction
[38] Local Searchers
[39] Lin-Kernighan
[40] Reduce
[41] Swap Operator (SO)
[42] Swap Sequence (SS)
[43] Diversity
[44] Iteration-based Algorithm
[۴۵] نتایج پیاده‌سازی این الگوریتم از صفحه اینترنتی http://www2.research.att.com/~dsj/chtsp/index.html آمده است.
[46] Simulated Annealing Algorithm With Greedy Search
[47] Memetic Algorithm
[48] Particle Swarm Optimization (PSO)
[49] Constructive-Optimizer Neural Network
[50] Local Optimum