سنجش از دور
و
GIS ایران

سنجش از دور و GIS ایران سال چهارم، شماره چهارم، زمستان ۱۳۹۱
Iranian Remote Sensing & GIS Vol.4, No.4, Winter 2013

۱-۲۰

ارائه روشی ابتکاری برای حل مسئله مسیریابی «فروشنده دوره‌گرد»

محمدرضا رجبی[۱]، علی منصوریان*[۲]، محمد طالعی[۳]، عباس علیمحمدی‌سراب[۴]

۱. کارشناس ارشد سیستم‌های اطلاعات مکانی، دانشگاه صنعتی خواجه نصیرالدین طوسی
۲و۳و۴. استادیار گروه GIS، دانشگاه صنعتی خواجه نصیرالدین طوسی

تاریخ دریافت مقاله: ۱۳۸۹/۱۱/۱۸ تاریخ پذیرش مقاله: ۱۳۹۰/۹/۱

چکیده

مسیریابی یکی از مسائل بسیار پرکاربرد GIS است که هدف اصلی آن یافتن بهترین مسیر گذرنده از یک سری موقعیت‌های از پیش تعیین شده است. این فرایند می‌تواند تأثیر بسزایی در تصمیم‌گیری‌های حساس مکانی داشته باشد. به همین دلیل از دیرباز تحقیقات بسیاری در مورد بهینه‌سازی این مسئله با استفاده از الگوریتم‌های مختلف صورت گرفته است. مسئله فروشنده دوره‌گرد یکی از مسائل بسیار کهن در علوم کاربردی است که پیش از پیدایش GIS نیز مطرح بوده است. این مسئله با ظهور فناوری‌های جدید مانند GIS کاربردهای بسیاری یافته و روش‌های جدیدی نیز برای حل آن پیشنهاد شده است. الگوریتم‌های تکاملی (ژنتیک) یکی از روش‌هایی هستند که برای حل مسائل بهینه‌سازی مختلف به کار گرفته می‌شوند. تحقیقات نشان داده است که تلفیق روش‌های جست‌وجوی محلی (Local Search) با عملگرهای ژنتیک می‌تواند منجر به نتایج بهتری در حل مسئله فروشنده دوره‌گرد شود. در نوشتار حاضر، روشی تازه و ابتکاری برای حل مسئله مسیریابی ارائه و پیاده‌سازی شده است. در این روش با بهره‌گیری از مفهوم مرکز هندسی به برازش چندضلعی‌ها با رئوس شهرها، به گونه‌ای پرداخته شده است که مسیر نهایی محدب‌ترین چندضلعی باشد. این الگوریتم با رویکردی پوششی با جهت بیرونی ـ درونی بزرگ‌ترین دایره محیطی شهرها را به کوچک‌ترین چندضلعی محدب ممکن تبدیل می‌کند. همچنین با استفاده از جست‌وجوی محلی مبتنی بر الگوریتم ژنتیک و روش نزدیک‌ترین همسایه (NN)، به حل مسئله مسیریابی فروشنده دوره‌گرد پرداخته شده است. ارزیابی نتایج حاصل از روش پیشنهادی با نتایج حاصل از روش‌های ژنتیکی، جست‌وجوی محلی و نزدیک‌ترین همسایه حاکی از این بود که روش پیشنهادی، سرعت و دقت بالایی را در تولید مسیرهای نهایی ارائه می‌کند. بررسی نتایج نهایی ژنتیک با روش ابتکاری نشان دادکه این الگوریتم همواره نمی‌تواند به جواب‌های بهتری برسد. مثلاً در تعداد ۲۵ بار اجرای جداگانهٔ جست‌وجوی ژنتیک، ۶۹/۳ درصد از جواب‌ها از جواب روش پیشنهادی، بهتر نبودند. از طرف دیگر روش پیشنهادی می‌تواند چندین هزار برابر سریع‌تر از الگوریتم قدرتمند ژنتیک جواب‌های نهایی را تولید کند.

کلیدواژه‌ها: GIS، مسیریابی، الگوریتم ژنتیک، جست‌وجوی محلی، نزدیک‌ترین همسایه (NN).

* نویسنده مکاتبه‌کننده: تهران، تقاطع خیابان ولی‌عصر و میرداماد، دانشگاه صنعتی خواجه نصیرالدین طوسی، دانشکده نقشه‌برداری،تلفن: ۸۸۷۸۶۲۱۲
Email: mansourian@kntu.ac.ir

۱- مقدمه

یکی از مهم‌ترین کاربردهایی که در مورد GIS[1] همواره مطرح بوده، پشتیبانی از تصمیم‌گیری‌های مرتبط با تعیین مسیر بهینه است. مسئله فروشنده دوره‌گرد[2] (TSP)، کوتاه‌ترین مسیر را برای یک فروشنده دوره‌گرد می‌یابد، به‌طوری‌که از یک شهر شروع می‌شود و از شهرهای دیگر به ترتیب خاصی عبور می‌کند و سرانجام به شهر مبدأ باز می‌گردد؛ با این شرط که از هر کدام از شهرها فقط و فقط یک بار عبور کند. هدف این مسئله بهینه‌سازی هزینه‌هاست. مشخص است مسافتی که این فروشنده دوره‌گرد می‌پیماید بستگی به ترتیبی دارد که از شهرها عبور می‌کند. بنابراین هدف این مسئله، یافتن مسیر بهینه برای عبور از شهرهاست، به گونه‌ای که اهداف هزینه‌ایِ در نظر گرفته‌شده، بهینه‌سازی شوند. با وجودی‌که درک این مسئله نیاز به مهارت‌های ریاضی پیچیده ندارد، اما به عنوان مسئله بهینه‌سازی ترکیبی دشوار از آن نام برده می‌شود، چرا که حل آن بسیار دشوار است.

چنین مسئله‌ای نخستین بار در سال ۱۷۵۹ میلادی از سوی اولرو با نام «مسئله مسیر شوالیه‌ها» مطرح شد. «مسیر شوالیه» عبارت بود از یک دور در گرافی که نودهای آن ۶۴ مربع صفحه شطرنج بودند که هر کدام با دو رأس دیگر مجاورت داشتند و شوالیه هر بار تنها می‌توانست به یک مربع مجاور حرکت کند. در قرن هیجدهم، این مسئله به عنوان زیرمجموعه‌ای از نظریه گراف از سوی ریاضی‌دان معروف ایرلندی، ویلیام هامیلتون، مورد مطالعه و بررسی قرار گرفت؛ که بعدها به احترام او «دور هامیلتونی» نام گرفت. عبارت «فروشنده دوره‌گرد» نخستین بار در کتابی آلمانی در مورد چگونگی شغل‌های دوره‌گردی به کار گرفته شد که مؤلف آن خود، فروشنده دوره‌گرد بود (Lawyer et al. 1985). نخستین کسی که این مسئله را مورد مطالعه و بررسی قرار داد، منگر (۱۹۳۲) بود که با ارائه الگوریتمی شبیه به «نزدیک‌ترین همسایه» سعی در حل این مسئله داشت اما نتوانست به جواب بهینه‌ای برسد. البته مطالعات اصولی روی TSP به عنوان مسئله بهینه‌سازی ترکیبی، به‌وسیله دانتزینگ و همکاران (۱۹۵۴) آغاز شد. مارینیکاس و همکاران (۲۰۰۷)، به شرح و بسط روش‌های مختلف جست‌وجوی همسایگی برای حل TSP، با استفاده از روش‌های اکتشافی و احتمالی پرداختند. آنها با ارائه الگوریتم Greedy، مسئله فروشنده دوره‌گرد را برای حالتی که تنها زیرمجموعه‌ای از شهرها می‌توانند احتمال بازدید داشته باشند، حل کردند. با وجود اینکه این الگوریتم می‌توانست جواب نهایی را روی زیرمجموعه‌ای از کل شهرها با سرعت مناسبی ارائه کند، اما کیفیت جواب‌های نهایی نسبتاً پایین بود ـ به‌ویژه وقتی که تعداد شهرها افزایش می‌یافت این موضوع بیشتر جلوه می‌نمود.

گلدبرگ (۱۹۸۹) با انجام مطالعاتی بر روی الگوریتم‌های تکاملی، آنها را به‌عنوان یکی از کارآمدترین الگوریتم‌های بهینه‌سازی معرفی کرد. در مورد TSP، جانسون و همکاران (۱۹۹۷) تحقیق مؤثری بر روی جزئیات بهینه‌سازی این مسئله با استفاده از الگوریتم‌های مختلف انجام دادند و به این نتیجه رسیدند که الگوریتم‌های تکاملی بهتر از روش‌های دیگر به جواب می‌رسند. به همین دلیل TSP یکی از مسائلی است که با استفاده از الگوریتم‌های تکاملی گوناگون مطالعات بسیاری در مورد حل آن صورت گرفته است (Bakhouya & Gaber, 2007). عملگرهای ژنتیک مختلفی برای حل بهتر TSP پیشنهاد شده که تعداد آنها همچنان رو به افزایش است. مثلاً عملگر inver-over (Tao, 1998)، عملگر DPX (Freisleben, 1996)، عملگر GX (Merz, 2002) و عملگر SPX (Soak, 2003) برای حل TSP طراحی و مورد استفاده قرار گرفته‌اند. لیو و همکاران (۲۰۰۵)، از تلفیق الگوریتم ژنتیک و روش‌های احتمالاتی برای حل مسئله

1. Geo-Spatial Information System
2. Traveling Salesman Problem

فروشنده دوره‌گرد بهره گرفتند. ماهیت ژنتیکی این روش در تلفیق با روش‌های احتمالاتی منجر به بروز اشکالاتی نیز گردید. با توجه به ضعف عملگرهای ژنتیکی، تمهید مناسبی برای رهایی از دام بهینه‌های نسبی در این روش اندیشیده نشد و با وجودی که الگوریتم گاه جواب‌های بهینه‌ای به دست می‌داد، اما تعداد همگرایی‌های نسبی بالا بود.

مشاهده شده است که ترکیب الگوریتم‌های تکاملی با روش‌های ابتکاری جست‌وجوی محلی می‌تواند منجر به بهبود نتایج بشود (Lawrence et al., 2005). جانسون و همکاران (۱۹۹۷) به این نتیجه رسیدند که برای حل مسائل ترکیبی دشواری همچون TSP با استفاده از الگوریتم‌ها تکاملی، بایستی این الگوریتم‌های با روش‌های جست‌وجوی محلی تلفیق شوند. به چنین روش‌هایی معمولاً الگوریتم‌های «جست‌وجوی محلی ژنتیک» گفته می‌شود. نخستین بار TSP را برادی (۱۹۸۵) با استفاده از روش‌های ترکیبی تکاملی حل کرد. او از روش 2-opt به عنوان جست‌وجوی محلی استفاده کرد. نتایج وی نشان داد که این روش برای حل مسئله TSP قابلیت‌های مناسبی را ارائه می‌کند. کومار و سینگ (۲۰۰۷)، به حل مسئله فروشنده دوره‌گرد به صورت تکاملی چندهدفه پرداختند. آنها با ارائه روشی نوین در مرحله انتخاب در الگوریتم ژنتیک کوشیدند تا فرآیند جست‌وجو را به همگرایی برسانند. ارزیابی‌ها نشان می‌دهد که، روش آنها در ارائه جواب‌های تقریباً بهینه، به خوبی عمل می‌کرد اما از دو لحاظ دارای ایراد بود. نخست اینکه زمان پردازش با افزایش تعداد شهرها چندین برابر می‌شد و دوم بهینگی جواب‌ها با افزایش تعداد شهرها به طور محسوسی کاهش می‌یافت.

هدف اصلی این پژوهش، حل مسئله فروشنده دوره‌گرد با روشی است که بتواند این مسئله پیچیده را در زمانی کوتاه و با جوابی مناسب حل کند؛ از این رو با ارائه روشی ابتکاری برای حل این مسئله مسیریابی، توانایی آن با روش‌های NN و جست‌وجوی محلی ژنتیک مورد مقایسه قرار گرفته است.

در این مقاله پس از مقدمه، در بخش دوم به تعریف مسئله فروشنده دوره‌گرد و روش‌های مختلف حل آن پرداخته می‌شود. در بخش سوم چگونگی پیاده‌سازی و حل مسئله TSP با استفاده از سه روش جست‌وجوی محلی ژنتیک، NN و روش ابتکاری (MASS Center)MASS_C تشریح می‌شود و سرانجام نتیجه‌گیری و پیشنهادهای حاصل از پژوهش ارائه می‌گردد.

۲- مواد و روش‌ها

با وجود روش‌های گوناگونی که برای حل مسئله فروشنده دوره‌گرد ارائه شده، شکل کلی مسئله همواره ثابت بوده است. بنابراین می‌توان کلیت مسئله را با نمادگذاری مشابهی ارائه کرد. فرض کنید $G=(V,E)$ یک گراف باشد (جهتی یا غیرجهتی) که در آن مجموعه $V = \{1,2,...,n\}$ یک دسته از رئوس و مجموعه $E = \{e1, e2, ..., en\}$ یک دسته از یال‌های گراف هستند. اگر به هر کدام از یال‌ها یک مقدار به عنوان هزینه اختصاص بیابد (C_e) می‌توان گفت ماتریس $C = (C_{ij})_{n\times n}$ ماتریس هزینه‌ای است که در آن C_{ij} مقدار مربوط به هزینه یال متصل‌کننده رأس i به j در G است. دور هامیلتونی از G دور ساده‌ای است که شامل تمامی رأس‌های V می‌شود. هدف TSP یافتن یک دور (کوچک‌ترین دور هامیلتونی) در G است، به طوری‌که مجموع هزینه یال‌ها تا حد ممکن کمینه باشد (Soak, Ahn, 2003).

بنابراین TSP را می‌توان به عنوان مسئله جایگشتی نیز مورد بررسی قرار داد. اگر P_n مجموع تمام جایگشت‌های دسته $\{1,2,...,n\}$ باشد، TSP به یافتن جایگشت $\pi = (\pi(1), \pi(2), ..., \pi(n), [\pi(1)])$ در P_n می‌پردازد که در آن $C_{\pi(n)\pi(1)} + \sum_{i=1}^{n-1} C_{\pi(i)\pi(i+1)}$ کمینه است. در اینجا $(\pi(1), \pi(2), ..., \pi(n), [\pi(1)])$ ترتیب بازدید شهرها را ارائه می‌کند که از شهر $\pi(1)$ آغاز می‌شود. از آنجا که با تغییر یک واحدی روی π باز هم

همان دور قبلی تشکیل خواهد شد می‌توان گفت که روی هر دور هامیلتونی، به تعداد n جایگشت وجود دارند که نتیجهٔ آنها یکسان خواهد بود. با توجه به ماهیت ماتریس هزینه و در واقع ماهیت G، TSP به دو دسته طبقه‌بندی می‌شود. اگر C متقارن باشد به این معنی است که G غیرجهتی است و مسئله را با نام فروشندهٔ دوره‌گرد متقارن[1] می‌شناسند. اما اگر C متقارن نباشد به این معنی است که گراف G جهت‌دار است؛ که در این حالت به مسئله فروشنده دوره‌گرد غیرمتقارن[2] تبدیل می‌شود. با توجه به اینکه هر گراف غیرجهتی با دو برابر کردن یال‌هایش و دادن هزینه‌ای برابر به هر دو یال رفت و برگشت، تبدیل به گراف جهتی می‌شود، پس STSP در واقع نوعی از ATSP است (Kumar, Singh, 2007).

اگر تعداد شهرهای مورد نظر در مسئله TSP برابر با n باشد، تعداد کل مسیرهای ممکن برابر با $(n-1)!$ خواهد بود. بنابراین رابطه‌ای نمایی میان تعداد شهرها و تعداد مسیرهای ممکن در این مسئله وجود دارد که باعث می‌شود برای ۵ شهر ۲۴ مسیر، برای ۶ شهر ۱۲۰ مسیر و برای ۱۰ شهر ۳۶۲۸۸۰۰ مسیر ممکن وجود داشته باشد.

۲-۱- روش‌های حل TSP

از جنبهٔ فنی دیدگاه‌های مختلفی در مورد طبقه‌بندی روش‌های حل TSP وجود دارد. در یکی از این دیدگاه‌ها روش‌های حل مسئله TSP به طور کلی به دو دسته تقسیم‌بندی شده‌اند: الف) روش‌های تکمیلی متوالی (ایجادکننده مسیر)، و ب) روش‌های جست‌وجوی محلی. روش‌های تکمیلی متوالی با فرآیندی افزایشی، یک دور هامیلتونی ایجاد می‌کند و با یافتن نخستین جواب ممکن، پایان می‌یابد. اما روش‌های جست‌وجوی محلی روی یک جواب ممکن کار می‌کنند تا آن را بهبود بخشند. روش‌های تکمیلی متوالی در دیگر بهینه‌سازی‌های ترکیبی عملکرد ضعیف‌تری دارند؛ اما پژوهشگران دریافته‌اند که چنین روش‌هایی برای TSP عملکرد بهتری دارد و تقریباً ۱۵-۱۰ درصد از مسیرهای بهینه را در زمان نسبتاً کوتاهی می‌یابند. البته روش‌های بهینه‌سازی بر مبنای جست‌وجوی محلی کارکرد بهتری را ارائه می‌دهند (Johnson, 1997).

۲-۱-۱- روش‌های تکمیلی متوالی (ایجادکننده مسیر)

همان‌طور که اشاره شد روش‌های «ایجادکننده مسیر» به ساخت مسیر با استفاده از مجموعه‌ای قوانین می‌پردازند و با یافتن نخستین مسیر دارای شرایط اولیه، متوقف می‌شود. در این قسمت دو نوع از این روش‌های تکمیلی متوالی معرفی می‌شوند.

۲-۱-۱-۱- الگوریتم نزدیک‌ترین همسایه

این روش یکی از ساده‌ترین روش‌های حل TSP است. مبنای روش مذکور بر این است که هنگام تعیین مسیر در هر مرحله همیشه نزدیک‌ترین و ارزان‌ترین مسیر انتخاب می‌گردد. از آنجا که این الگوریتم به نوعی از الگوریتم‌های Greedy محسوب می‌شود، پیاده‌سازی آن ساده است و بسیار سریع نیز عمل می‌کند. اگرچه این الگوریتم در بعضی مواقع جواب‌های مناسبی ارائه می‌کند، اما به‌طور کلی بسیاری از مسیرهای پرهزینه و نامناسب را نیز می‌تواند تولید کند (Kumar, Singh, 2007).

۲-۱-۱-۲- الگوریتم GREEDY

این روش بسیار شبیه به الگوریتم نزدیک‌ترین همسایگی عمل می‌کند با این تفاوت که در هر گام به جای اینکه نزدیک‌ترین شهر را به گراف متصل کند، با کوتاه‌ترین یال شروع می‌کند، و هر بار کوچک‌ترین یال‌ها را از میان یال‌های باقی‌مانده به گراف می‌افزاید. یک یال را به عنوان «یال در دسترس» در نظر می‌گیریم که می‌تواند در ادامه الگوریتم مورد بررسی

1. Symmetric TSP
2. AsymmetricTSP

قرار بگیرد به شرطی که (۱) تاکنون ایجاد نشده باشد، (۲) افزودن آن منجر به رأس درجه ۳ نگردد و (۳) دور با طول کمتر از n ایجاد نکند (n تعداد رئوس) (Liu, 2005). در این الگوریتم شرط اتصال یال‌های قبلی و بعدی وجود ندارد و با توجه به قیود در نظر گرفته‌شده، مسیر ایجادشده در انتها، مسیری پیوسته خواهد بود.

۲-۱-۲- روش‌های جست‌وجوی محلی

در این قسمت روش‌های معمول و مؤثر در بهینه‌سازی مسیرها که در آنها با استفاده از تغییرات سعی می‌شود تا با تعویض یال‌ها، هر دور به دور بهتری تبدیل گردد، شرح داده می‌شود. با داشتن یک دور «ممکن» این الگوریتم مکرراً تعویض یال‌ها را انجام می‌دهد تا طول دور کنونی کاهش یابد. این عمل هنگامی‌که پس از چندین تکرار هیچ بهبودی در گراف ایجاد نشود، متوقف می‌گردد.

۲-۱-۲-۱- عملگرهای 2-OPT و 3-OPT

2-opt و 3-opt روش‌های ابتکاری هستند که نتایج مناسبی را در زمان‌هایی مناسب ارائه می‌کنند. الگوریتم 2-opt را نخستین بار کروز (۱۹۵۸) پیشنهاد داد. در این روش ابتدا یک دور به دو مسیر تقسیم می‌شود و سپس این دو مسیر دوباره و با یال‌هایی جدید متصل می‌گردند. اما در روش 3-opt دور اولیه به سه مسیر تقسیم می‌گردد و دوباره و با یال‌هایی متفاوت متصل می‌گردد تا بتواند منجر به دور کوتاه‌تری شود (Krasnogor, et al., 2006; Brady, 1985).

جدول ۱. مشخصات الگوریتم نزدیک‌ترین همسایگی در مسئله TSP

۱: ورودی: یک گراف کامل وزن‌دار
۲: خروجی: مجموعه‌ای از رئوس برچسب‌دار که ترتیب‌شان یک مسیر را تشکیل می‌دهد
۳: الگوریتم مسیریابی:
- از یک رأس دلخواه شروع کن و آن را به عنوان «رأس کنونی» برچسب‌گذاری کن
- تا هنگامی که رئوس برچسب‌دهی‌نشده وجود دارند انجام بده
  - ساده‌ترین و کم‌هزینه‌ترین یالی که یک رأس برچسب‌دهی‌نشدهٔ w را به رأس کنونی متصل می‌کند.
  - w را برچسب‌گذاری کن و آن را به عنوان «رأس کنونی» قرار بده
- پایان

۴: مسیر خروجی

جدول ۲. مشخصات الگوریتم Greedy در مسئله TSP

۱: ورودی: یک گراف کامل وزن‌دار
۲: خروجی: مجموعه‌ای از رئوس برچسب‌دار که ترتیب آنها یک مسیر را تشکیل می‌دهد
۳: الگوریتم
۴: کوتاه‌ترین یال را انتخاب کن و آن را از مجموعه یال‌ها حذف کن
۵: تا هنگامی که اندازه مجموعه یال‌های انتخاب‌شده کوچک‌تر از n است انجام بده:
- کوچک‌ترین یال (u,v) را انتخاب کن (به شرطی که یک رأس با درجه سه[1] یا دور با طول کمتر از n ایجاد نشود).
- یال (u,v) را حذف کن و دیگر مورد بررسی قرار نده.

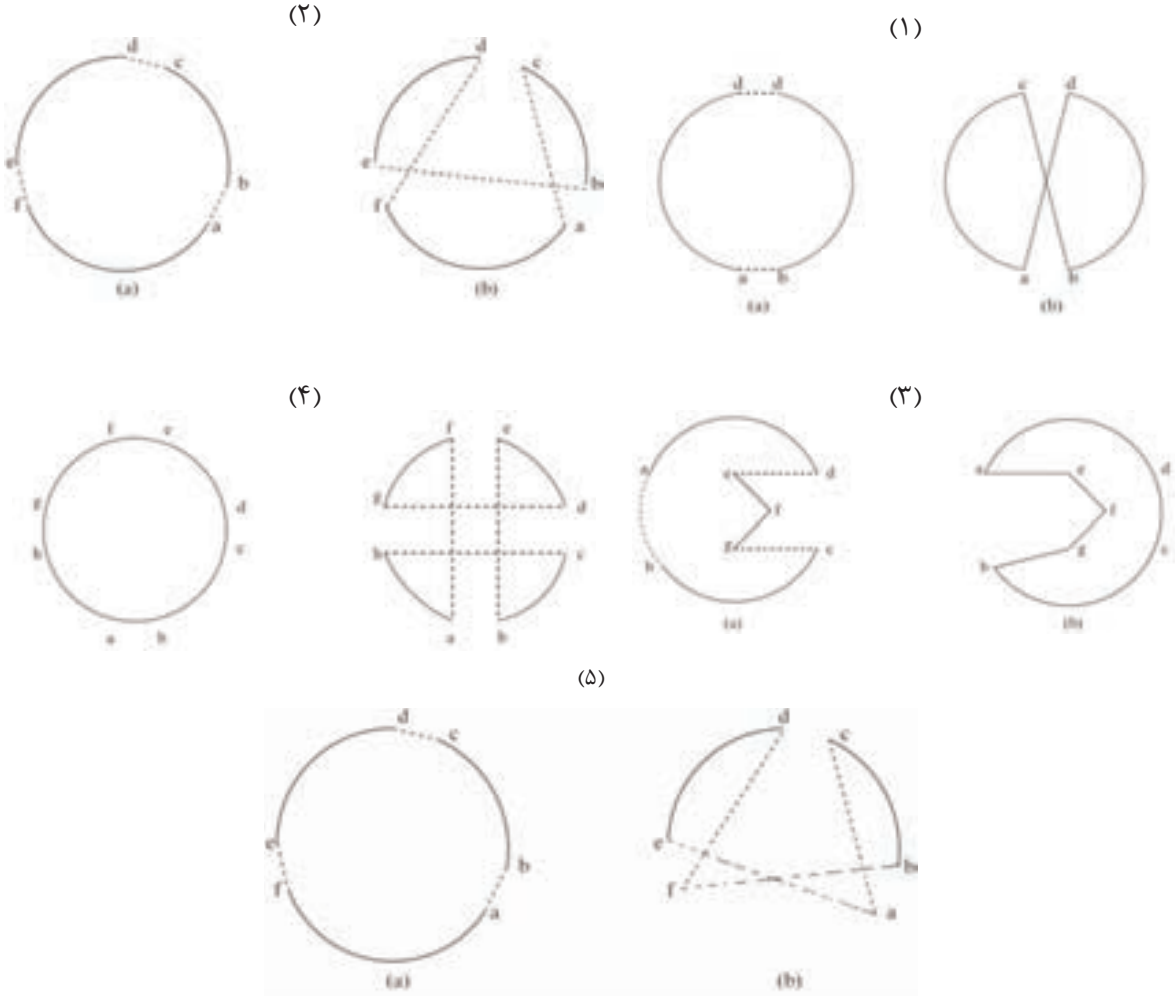
۶: پایان
۷: مسیر خروجی

۱. رأس درجه ۳، رأسی است که ۳ یال با آن تماس داشته باشد و در نتیجه بیش از یک بار مورد بازدید قرار گرفته باشد.

۲-۱-۲-۲- عملگرهـــای 2.5-OPT، OR-OPT و DOUBLE-BRIDGE

بنتلی (۱۹۹۲) روشی را ارائه کرد که بسیار شبیه به روش 2-opt بود اما تفاوت‌هایی نیز با آن داشت. در این روش یک شهر در موقعیتی قرار می‌گیرد (مثلاً بین دو شهر b و f) که بین دو شهر دیگر باشد؛ این روش معمولاً 2.5-opt نامیده می‌شود. اور (۱۹۷۶) الگوریتمی شبیه به 3-opt را با نام Or-opt ارائه کرد که در آن یک، دو و یا سه شهر متصل به هم که در همسایگی دو شهر c و d قرار گرفته‌اند در همسایگی دو شهر دیگر قرار بگیرند.

همان‌طور که مشاهده شد در روش‌های 2-opt و 3-opt، دو و سه یال تعویض می‌شدند پس می‌توان انتظار داشت که روشی برای مبادله چهار یال نیز وجود داشته باشد. این روش که «پل‌های دوگانه» نام دارد نخستین بار به‌وسیله لین ـ کرینگان (۱۹۷۳) مورد استفاده قرار گرفت و به این ترتیب عمل می‌کند که ابتدا چهار یال از دور به طور تصادفی جدا می‌شوند و سپس دوباره به گونه‌ای متفاوت متصل می‌گردند (Lin.S, Kernighan, 1973) (شکل ۱).



شکل ۱. عملگرهای جست‌وجوی محلی (۱) 2-opt (۲)3-opt (۳) Or-opt (۴) double-bridge (Kumar, Singh, 2007) (۵) 2.5 opt (مؤلفان)

۲-۲- الگوریتم ژنتیک

ایده اصلی الگوریتم‌های تکاملی[1] (EA) را ریچنبرگ در سال ۱۹۶۵ مطرح کرد. الگوریتم ژنتیک که منشعب از این نوع الگوریتم‌هاست، در واقع روش جست‌وجوی کامپیوتری بر پایه الگوریتم‌های بهینه‌سازی و براساس ساختار ژن‌ها و کروموزوم‌هاست، که پروفسور هلند (۱۹۸۹) در دانشگاه میشیگان آن را مطرح کرد و پس از وی جمعی از دانشجویانش از جمله گلدبرگ آن را شرح و بسط دادند. مبنای اصلی این الگوریتم‌ها بر اصل بقای بهترین‌هاست که طی آن در ساختاری تکاملی، مجموعه‌ای تصادفی از جواب‌های اولیه مسئله به تکامل می‌رسند و در واقع بهینه می‌شوند.

الگوریتم‌های ژنتیک با وجود تفاوت‌هایی که در نحوهٔ عملکردشان دارند، از اصول و عملگرهای مشابهی بهره می‌گیرند. همهٔ الگوریتم‌های تکاملی دارای تابعی برای ارزیابی شایستگی کروموزوم‌ها هستند. تابع ارزیابیِ شایستگی، به هر کدام از کروموزوم‌ها ـ یا همان جواب‌های تولیدشده ـ یک مقدار شایستگی نسبت می‌دهد. این مقدار برآوردی از میزان تناسب جواب‌های تولیدی را با استفاده از یک سری اطلاعات از پیش تعیین شده به‌دست می‌دهد. در واقع این مرحله تنها نقطه‌ای از الگوریتم است که مجموعه‌ای از اطلاعات وارد سیستم می‌شوند. کروموزوم‌ها با استفاده از مقادیر شایستگی انتخاب می‌شوند و به این ترتیب آنهایی که دارای مقادیر بهتری باشند بیشتر انتخاب می‌شوند. انتخاب کروموزوم‌ها براساس مقادیر شایستگی، فاکتور بسیار مهم در قدرت الگوریتم‌های ژنتیک است که به عنوان الگوریتم‌های جست‌وجوگر شناخته می‌شوند.

تولیدمثل معمولاً نخستین عملی است که بر روی جمعیت اعمال می‌شود. در این روش یک سری کروموزوم از میان جمعیت[2] به عنوان والد انتخاب می‌شود که در نهایت با عمل ادغام منجر به تولید فرزندان می‌شود. براساس نظریهٔ حیاتِ بهترین‌ها، باید بهترین موارد انتخاب شوند تا نسل بعدی بهتری را تولید کنند. به همین دلیل به عملگر تولیدمثل[3]، عملگر انتخاب[4] نیز گفته می‌شود، روش‌های مختلفی در GA برای انتخاب کروموزوم‌ها وجود دارد، اما هدف اصلی در همهٔ آنها انتخاب رشته‌هایی با میانگین بالا از جمعیت فعلی و تولید کپی‌های چندگانه و قراردادن آنها در مکانی به نام استخر تولیدمثل[5] براساس فرم احتمالی است.

عملگر انتخاب دو تا از اعضای کنونی نسل را برای شرکت در عملیات‌های تکاملی ادغام و جهش برمی‌گزینند. در این مرحله دو والد برای تولید مثل انتخاب می‌شوند که بایستی منجر به تولید فرزندانی تکامل‌یافته بشوند. روش‌های انتخاب می‌توانند به شیوه‌های مختلفی اجرا بشوند که با توجه به شرایط مسائل می‌توانند نتایج مختلفی را نیز ارائه کنند. روش‌های انتخابی را که معمولاً در الگوریتم‌های تکاملی مورد استفاده قرار می‌گیرند می‌توان شامل روش چرخ رولت، روش رتبه‌بندی و روش‌های رقابتی[6] دانست. در روش رولت گاهی مشکلاتی از قبیل کندی و هم‌گرایی ناگهانی جست‌وجو به خاطر کوچک شدن سریع فضای جست‌وجو پدید می‌آیند. به همین دلیل معمولاً از روش دیگری به نام روش رقابتی استفاده می‌شود. برخلاف رولت در روش رقابتی که شبیه رقابت در طبیعت است، زیرمجموعهٔ کوچکی از کروموزوم‌ها (معمولاً دو یا سه) به صورت تصادفی انتخاب می‌شوند و به رقابت می‌پردازند. سرانجام در این رقابت براساس میزان متناسب بودن‌شان، یکی از آنها به پیروزی می‌رسید و به عنوان والد جدید در استخر تولیدمثل کپی می‌شود و این فرآیند تا تولید همهٔ والدها در جمعیت جدید تکرار می‌شود. در روش انتخاب رقابتی دودویی، برای گزینش هر کدام از والدها زیرمجموعه‌ای دوتایی از اعضای

1. Evolutionary Algorithms
2. population
3. reproduction
4. selection operator
5. Mating pool
6. Tournament

جمعیت انتخاب می‌شود و سپس هر کدام از این کروموزوم‌ها که بهترین شایستگی را داشته باشد به عنوان والد انتخاب می‌گردد تا در مراحل بعدی تولیدمثل شرکت داده شود.

پس از اینکه مرحلهٔ تولیدمثل تمام شد، جمعیتی از بهترین‌ها به‌وجود آمده است. عمل تولیدمثل مجموعه‌ای از بهترین‌ها را انتخاب می‌کند، اما کروموزوم‌های جدیدی را به‌وجود نمی‌آورد. به همین دلیل عملگر ادغام[1] با هدف تولید جواب‌های بهتر اعمال می‌شود. هدف از ادغام، جست‌وجوی فضای پارامتر و تا حد امکان حفظ اطلاعات نهفته در رشته‌هاست. روش‌های مختلفی برای انجام عمل ادغام وجود دارد (مانند ادغام تک‌نقطه‌ای و دونقطه‌ای، یکنواخت) براساس نظر دب (۲۰۰۱)، نمی‌توان گفت که کدام یک از روش‌های ادغام بهتر است.

در اینجا لازم است به مفهوم مهم دیگری به نام نرخ ادغام اشاره کنیم. طبق تعریف، نرخ ادغام بیانگر احتمال ادغام است که آن را با P_c نشان می‌دهند و مقدار آن بین صفر تا یک است. این نرخ در GA با پیدا کردن نسبت تعداد جفت‌های ادغام‌شده در جمعیت‌های ثابت به‌دست می‌آید و با فرض احتمال ادغام P_c می‌توان گفت که $100*P_c$ درصد از رشته‌ها در عملیات ادغام به کار رفته‌اند و $100*(1-P_c)$ درصد از جمعیت باقی‌مانده است. طبق تعریف دیگر، این نرخ بیانگر تعداد کروموزوم‌هایی است که وارد استخر تولیدمثل شده‌اند. هر چقدر این مقدار بیشتر باشد، یعنی کروموزوم‌های جدید و زیادتری وارد استخر تولیدمثل شده‌اند.

پس از عمل ادغام کروموزوم‌ها، نوبت به عمل جهش یا موتاسیون می‌رسد. به‌عبارت ساده عمل جهش در ژن‌های باینری شامل تبدیل عدد صفر به یک و برعکس است که براساس احتمالی کوچک مانند P_m به صورت بیت به بیت صورت می‌گیرد. عمل جهش به این ترتیب است که عددی تصادفی بین صفر تا یک تولید می‌شود. اگر عدد تولیدشده کوچک‌تر از P_m باشد مقدار خروجی برابر با درست (true) وگرنه برابر غلط (false) در نظر گرفته می‌شود. اگر برای هر بیت مقدار خروجی درست باشد بیت تغییر می‌کند وگرنه بیت بدون تغییر باقی خواهد ماند. بیت‌های یک رشته یا کروموزوم به صورت مستقل جهش می‌یابند، به این معنی که جهش یک بیت بر روی احتمال بیت‌های دیگر تأثیر نمی‌گذارد. این عمل در الگوریتم ژنتیک ساده به منزلهٔ عملگر ثانویه و به‌منظور حفظ اطلاعاتی که در حال از دست رفتن است، تلقی می‌گردد. این عمل برای جلوگیری از هم‌گرایی سریع و کمک به الگوریتم جست‌وجو به منظور فرار از به دام افتادن در مینیمم‌های موضعی مفید است. از سوی دیگر این عمل برای حفظ حالت متفاوت و متمایز بودن کروموزوم‌ها در جمعیت به کار می‌رود.

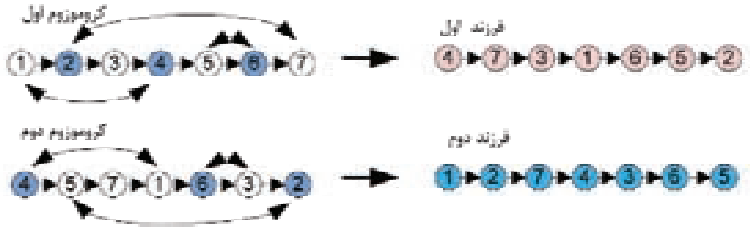
۳- بحث و نتایج

به منظور حل مسئله فروشنده دوره‌گرد در این پژوهش از سه شیوهٔ متفاوت بهره گرفته شد. در روش نخست با تلفیق عملگرهای جست‌وجوی محلی با الگوریتم ژنتیک به مسیریابی پرداخته شد. در روش دوم با استفاده از الگوریتم NN و همچنین تلفیق آن با عملگر 2-OPT به حل مسئله فروشنده دوره‌گرد پرداخته شد. و در نهایت روشی ابتکاری و جدید برای حل مسئله ارائه می‌گردد که در ادامه به تفصیل مورد بحث و بررسی قرار می‌گیرد.

۳-۱- الگوریتم ژنتیک برای حل TSP

به‌طور کلی الگوریتم‌های ژنتیک با توجه به شرایط مسائل مختلف می‌توانند به شیوه‌های متفاوتی طراحی و پیاده‌سازی شوند. از آنجا که هر کدام از عملگرهای ژنتیک (انتخاب، ادغام و جهش) دارای پارامترهای مختلف و روش‌های پیاده‌سازی گوناگونی هستند، برای هر مسئله بهینه‌سازی با تغییر این پارامترها می‌توان مجموعه متنوعی از جواب‌ها را ارائه کرد.

1. Cross Over



شکل ۲. اجرای عملگر توسعه‌یافته 3-OPT روی کروموزوم ترتیبی عبور از شهرها

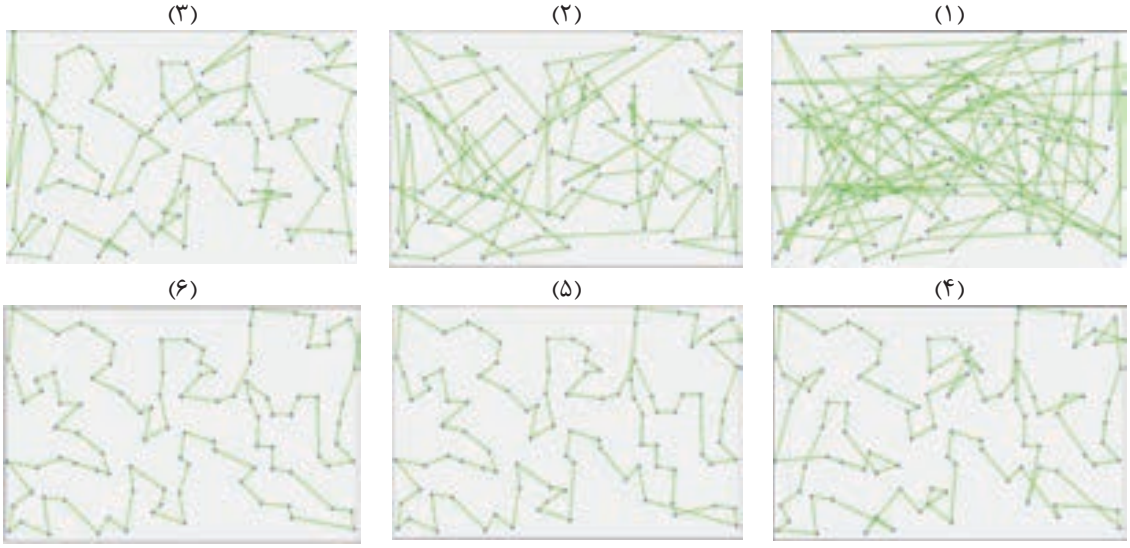
در پژوهش حاضر، الگوریتم ژنتیک برای ایجاد بهترین مسیری که از ۱۰۰ شهر مورد نظر عبور کند به کار گرفته شد. کدگذاری مسئله در قالب کروموزوم‌ها به این صورت پیاده‌سازی شد که هر کروموزوم شامل ۱۰۰ ژن می‌شود که شماره‌های میان ۱ تا ۱۰۰ را به خود اختصاص می‌دهند، و ترتیب آنها بیان‌کنندهٔ یک مسیر عبورکننده از ۱۰۰ شهر مورد نظر است. هر ژن بیانگر موقعیت ترتیبی هر شهر برای بازدید است و از این‌رو به هر شهر یک شماره تخصیص می‌یابد. بنابراین کدگذاری صحیحی (integer) برای این مسئله به کار گرفته شد. تعداد ژن‌ها برابر با تعداد شهرها خواهد بود و مقادیر ژن‌ها نباید خارج از محدوده تعداد شهرها باشد و هر شهر بایستی تنها یک بار در هر کروموزوم ظاهر شود. بهترین کروموزوم‌ها آنهایی در نظر گرفته شدند که در مقایسه با دیگران طول مسیر کمتری دارند. به عبارت دیگر برای ارزیابی شایستگی هر کدام از کروموزوم‌ها (جواب‌ها) از مسافت استفاده شد. برای انتخاب والدین شرکت‌کننده در تولیدمثل، از «تورنمت باینری» بهره‌گیری گردید.

در بسیاری از مواقع، الگوریتم‌های تکاملی و ژنتیک جمعیت‌شان را به صورت تصادفی به‌وجود می‌آورند. در این مسئله با توجه به قیود و شروط مسئله اگر کروموزوم‌ها به طور کاملاً تصادفی ایجاد شوند، امکان اینکه کروموزوم‌های جدید قیود را برآورده کنند بسیار کم خواهد بود. به‌طور کلی قیود موجود در مسئله فروشنده دوره‌گرد شامل (۱) بازدید از شهر فقط و فقط یک بار، و (۲) بازگشت مجدد به نقطهٔ شروع می‌گردد. به همین دلیل در این پژوهش کروموزوم‌ها با استفاده از آرایه‌ای از پیش تعریف شده، ایجاد شدند. این آرایهٔ تک‌بعدی که اندازهٔ آن برابر است با تعداد شهرهای مورد بازدید قرار داده شده، ابتدا با استفاده از شماره شهرها مقداردهی می‌شود و هنگامی‌که بایستی کروموزوم جدیدی تولید شود به طور تصادفی ترتیب مقداردهی اولیه خود را با رعایت قیود تعویض می‌کند تا مسیر جدیدی تولید شود.

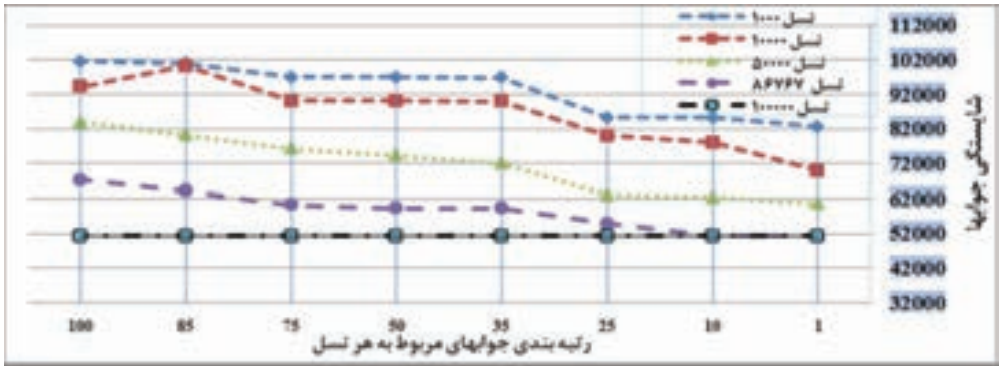
عملگر ادغام با استفاده از جست‌وجوی محلی 3-opt به تعویض و تولید یال‌های جدید پرداخت تا الگوریتم را به کروموزوم‌های با شایستگی بهتر هم‌گرا کند. نرخ ادغام برابر با ۰/۷ در نظر گرفته شد. بنابراین ۷۰ درصد از نسل در ادغام شرکت داده می‌شوند. برای به‌کارگیری عملگر 3–OPT در قالب ادغام در الگوریتم ژنتیک با توجه به نوع کروموزوم‌های در نظر گرفته‌شده برای حل مسئله ابتدا از والد یکم سه تا از رئوس به‌طور تصادفی انتخاب شدند. سپس محل این رئوس در والد دوم شناسایی گردید (شکل ۲). مثلاً در شکل ۲ سه رأس (ژن) تصادفی عبارت‌اند از ۲، ۴ و ۶. رأس ۲ که در کروموزوم یکم دومین ژن است در کروموزوم دوم، هفتمین ژن خواهد بود. بنابراین محل رأس ۲ در کروموزوم یکم به «هفتمین ترتیب» تغییر می‌یابد و در کروموزوم دوم نیز رأس ۲ به محل ژن دوم انتقال می‌یابد. این مراحل برای دو رأس دیگر نیز تکرار می‌گردد. برای پیاده‌سازی عملگر جهش بدین صورت عمل شد که به ازای هر دو تا از ژن‌ها که مقداری کمتر از ۰/۰۵ برای آنها به‌دست بیاید، جای آنها در کروموزوم‌ها با یکدیگر تعویض می‌شود و مسیری جدید تولید می‌گردد. توجه شود که این عملگر در کروموزومی که فقط یک ژن کمتر از ۰/۰۵ داشته باشد اجرا نمی‌شود و برای کورموزوم با پنج ژن جهش‌یافته، فقط

دو بار اجرا می‌شود. بعد از تولید نسل جدید با استفاده از عملگرهای ژنتیک، آنها از لحاظ شایستگی با والدین‌شان مقایسه می‌شوند و بهترین ۱۰۰ کروموزوم برای نسل بعدی انتخاب می‌گردد. پس با توجه به اندازه جمعیت ـ که برابر ۱۰۰ در نظر گرفته شد ـ در این مرحله از میان ۲۰۰ کروموزوم والد و فرزند، ۱۰۰ کروموزوم برتر انتخاب می‌گردند. این روش باعث حفظ کروموزوم‌های شایسته‌ای می‌شود که در نسل‌های اولیه تولید می‌گردند.

پس از اجرای تنظیمات اولیه و اجرای الگوریتم، ابتدا یک مسیر تصادفی اولیه با رعایت قیود تولید می‌شود و سپس با استفاده از عملگرهای ژنتیک مقدار شایستگی (طول) آن کاهش می‌یابد تا هنگامی که به بهترین جواب‌ها برسد. پس از اینکه تمامی کروموزوم‌های تولیدشده از بقیه کروموزوم‌ها بهتر بودند و هیچ کروموزوم جدیدی بهتر از کروموزوم‌های نسل کنونی نبود، نسل تولیدی بدون هیچ تغییری تا پایان تکرارهای تعیین‌شده باقی می‌ماند. در شکل ۳، روند تکاملی مسیر اولیهٔ تولیدشده نمایش داده شده است. همان‌طور که مشاهده می‌شود با افزایش تکرارها مسیری که تولید می‌شود بهبود می‌یابد و درواقع کوتاه‌تر می‌شود تا به بهینه‌ترین حالت برسد. بعد از تولید هر نسل کروموزوم‌های آن از لحاظ شایستگی از بهترین به بدترین مرتب می‌شوند و به ترتیب رتبهٔ ۱ تا ۱۰۰ به آنها اختصاص می‌یابد.



شکل۳- (۱) مسیر تولیدشده TSP بعد از ۱۰۰ تکرار، (۲) مسیر تولیدشده TSP بعد از ۱۰۰۰ تکرار، (۳) مسیر تولیدشده TSP بعد از ۱۰۰۰۰ تکرار، (۴) مسیر تولیدشده TSP بعد از ۵۰۰۰۰ تکرار، (۵) مسیر TSP تولیدشده بعد از ۷۵۰۰۰ تکرار، (۶) مسیر TSP تولیدشده بعد از ۱۰۰۰۰۰ تکرار



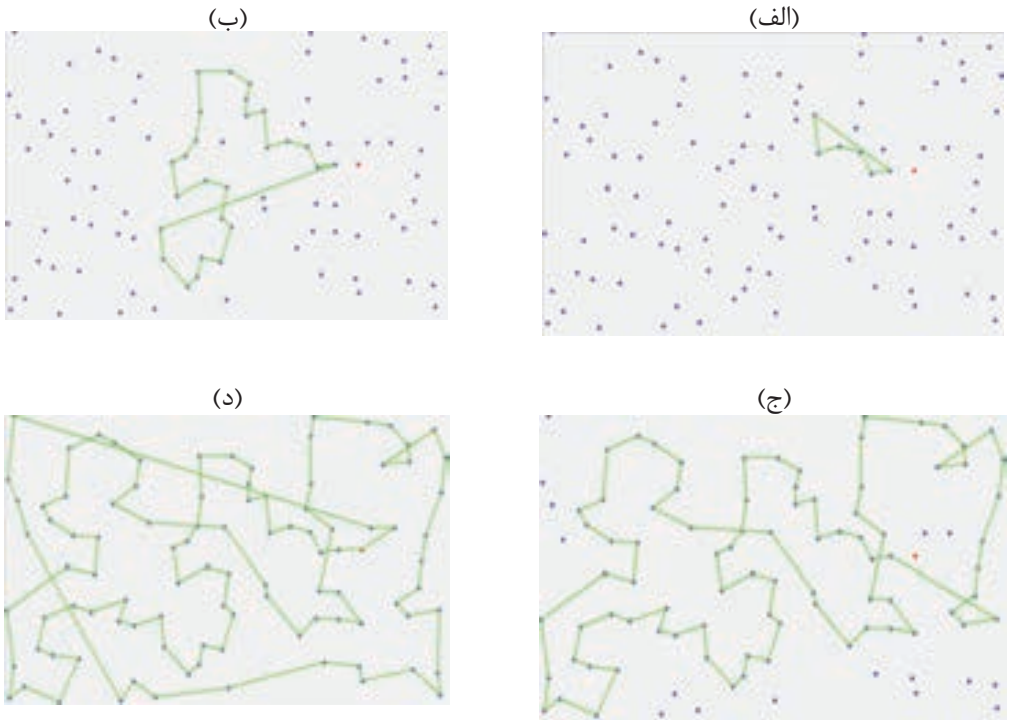
شکل ۴. روند تکاملی مربوط به جواب‌های حاصل از حل مسئله TSP با الگوریتم ژنتیک

در شکل ۴ کروموزوم‌های مختلف تولیدشده در نسل‌های مختلف، در جست‌وجوی محلی ژنتیک با یکدیگر مورد مقایسه قرار گرفته‌اند. روند تکاملی به‌خوبی با افزایش تعداد نسل‌ها طی می‌شود و جواب‌هایی که در نسل‌های بالاتر تولید می‌شوند شایسته‌ترند.

بررسی نتایج نشان داد که الگوریتم جست‌وجوی محلی ژنتیک در نسل ۸۶۷۶۷ به جوابی با ۵۱۳۴۳/۲۳ واحد می‌رسد که تا نسل ۱۰۰,۰۰۰ به‌عنوان بهترین جواب باقی می‌ماند. صدهزار تکراری که برای تعیین مسیر عبوری فروشنده دوره‌گرد از صد شهر اجرا گردید، در زمانی در حدود ۴۹۱ دقیقه (بیش از ۸ ساعت) به بهترین جواب (هم‌گرایی) رسید و کل تکرارها نیز ۵۷۴ دقیقه به طول انجامید. توجه شود که با توجه به ماهیت تصادفی الگوریتم ژنتیک این مقادیر در تکرارهای متفاوت تا ۱۵ درصد تغییر می‌کردند. مقایسه الگوریتم ژنتیک برای تعداد صد شهر با شرایطی با تعداد شهر کمتر، نشان داد که با افزایش تعداد شهرها زمان محاسباتی الگوریتم ژنتیک به صورت محسوسی طولانی‌تر می‌شود. مثلاً الگوریتم ژنتیک فوق برای تعداد ۳۰ شهر چندین بار اجرا گردید. نتایج حاکی از این بود که هم‌گرایی همواره ۱۰ تا ۱۵ دقیقه پس از اجرای الگوریتم، با حدود ۴۲۰۰ تکرار رخ می‌دهد.

۳-۲- الگوریتم NN برای حل TSP

برای حل مسئله فروشنده دوره‌گرد با استفاده از الگوریتم NN، نخستین شهری که بایستی از آن عبور کرد به عنوان «شهر کنونی» تعیین و جزء شهرهای «بازدید شده» قرار داده می‌شود. پس از این مرحله بقیه شهرها مورد بررسی قرار می‌گیرند تا نزدیک‌ترین شهر به‌عنوان شهر بعدی انتخاب گردد. مادامی که شهر بازدیدنشده‌ای وجود نداشته باشد، این مراحل تکرار می‌شود و هر بار یک شهر به مجموعه شهرهای بازدیدی افزوده می‌شود و در نهایت مجموعه‌ای از شهرهای مرتب‌شده، ارائه می‌شود که مسیر عبوری فروشنده دوره‌گرد را از تعداد معلومی شهر (در اینجا ۱۰۰ شهر) ارائه می‌کند (شکل ۵).



شکل۵. حل مسئله فروشنده دوره‌گرد با استفاده از الگوریتم NN برای ۱۰۰ شهر

شکل ۶. بهبود جواب نهایی الگوریتم NN با استفاده از عملگر 2-opt (الف) جواب نهایی NN (ب) بهبود جواب NN

جدول ۳. مشخصات الگوریتم ابتکاری MASS_C در حل TSP

۱: ورودی: یک گراف کامل وزن‌دار
۲: خروجی: مجموعه‌ای از رئوس برچسب‌دار که ترتیب آنها یک مسیر را تشکیل می‌دهد
۳: الگوریتم
۴: مرکز هندسی حاصل از تمامی شهرها را محاسبه کن.
۵: دورترین شهر از مرکز هندسی انتخاب شود
۶: دورترین شهر از مرکز هندسی به‌عنوان "شهر بازدیدی" قرار داده شود.
۷: اگر دور هامیلتونی تشکیل نشده است، برو به ۵ وگرنه برو به ۸
۸: دورترین شهر غیربازدیدی از مرکز هندسی را بیاب و آن را مابین دو تا از نزدیک‌ترین شهرهای بازدیدی قرار بده و مسیر هامیلتونی جدید تشکیل بده.
۹: به شهر غیربازدیدی جدید، برچسب "بازدیدشده" بده.
۱۰: اگر شهر بازدیدنشده‌ای وجود دارد برو به ۸ وگرنه برو به ۱۱.
۱۱: پایان

بررسی نتایج حاصل از الگوریتم NN در حل مسئله فروشنده دوره‌گرد حاکی از پرهزینه بودن مسیرهای پیشنهادی (مسیر طولانی‌تر) بود. مطالعات نشان داد که می‌توان عملگرهای «جست‌وجوی محلی» را در این روش وارد کرد. برای بررسی بیشتر نتایج حاصل از این تلفیق، عملگرهای 2-OPT روی مسیر نهایی حاصل از اجرای الگوریتم NN، اعمال شدند. این عملگرها با تعویض دوبه‌دوی یال‌های به‌دست‌آمده، هر بار طول کلی مسیر را مورد ارزیابی قرار می‌دهند و در صورتی‌که این تعویض یال منجر به مسیر نهایی کم‌هزینه بشود، این تغییر مسیر روی جواب نهایی ذخیره می‌گردد.

بررسی نتایج حاکی از این بود که حدوداً تا ۲۰ درصد از طول مسیر نهایی تولیدشده به‌وسیله الگوریتم NN را می‌توان به‌وسیله ترکیب عملگرهای جست‌وجوی محلی با آن بهبود بخشید. اگرچه این مسیر بهبودیافته هنوز نمی‌تواند مسیر بهینه‌ای باشد ولی در کارکردهایی که الگوریتم NN را به خاطر سادگی و سرعت آن در یافتن جواب، به کار می‌گیرند، تلفیق با عملگرهای جست‌وجوی محلی می‌تواند تا حدودی بر کارایی جواب‌های نهایی NN بیفزاید (شکل ۶).

۳-۳- روشی ابتکاری برای حل TSP

به‌طور کلی برای ایجاد کوتاه‌ترین مسیری که از تعداد مشخصی شهر (در اینجا ۱۰۰ شهر) عبور کند، از روش‌ها و تکنیک‌های مختلفی می‌توان استفاده کرد. در این پژوهش با استفاده از ایده‌ای ابتکاری از مفهوم مرکز هندسی بهره‌گیری گردید تا روشی جدید از نوع روش‌های «ایجادکننده مسیر» را که بتواند با دقتی مناسب در یک مرحله مسیری مورد قبول که از تعداد مشخصی شهر عبور کند، به وجود آورد. در روش ابتکاری این پژوهش، ابتدا مرکز هندسی مربوط به مجموعه تمامی شهرهای ارائه‌شده محاسبه می‌گردد. در مرحله بعد، دورترین شهر از این نقطه محاسبه و جزء «شهرهای بازدیدی» قرار داده می‌شود، این کار سه بار ادامه پیدا می‌کند تا با سه رأس نخستین دور هامیلتونی بتواند تشکیل شود. از این قسمت به بعد الگوریتم بایستی هم‌زمان دو نوع محاسبه را انجام دهد؛ نخست اینکه دورترین شهر جدید از مرکز هندسی را بیابد و سپس آن را به گونه‌ای در ترتیب عبور از شهرها قرار دهد که کوتاه‌ترین مسیر ایجاد گردد. برای این مرحله، الگوریتم با جست‌وجو روی شهرهای بازدیدی، نزدیک‌ترین دو شهر عبوری را می‌یابد، و با افزودن یک عضو به دنبالهٔ مربوط به شهرهای بازدیدی، شهر جدید را در موقعیتی قرار می‌دهد که کمترین هزینه (در اینجا فاصله است) به مسیر قبلی افزوده گردد (جدول ۳).

با این شیوه هربار محدب‌ترین چندضلعی به شهرهای بازدیدی برازش داده می‌شود که منجر می‌گردد مسیر تولیدی بهینه باشد. توجه شود که چندضلعی برازش داده‌شده پس از تشکیل اولین دور هامیلتونی به‌تدریج از حالت محدب خارج می‌گردد اما همواره در این الگوریتم از میان چندین هزار چندضلعی که می‌توان برازش داد، چندضلعی‌ای انتخاب می‌شود که نزدیک‌ترین ویژگی‌ها به چندضلعی محدب را داشته باشد.

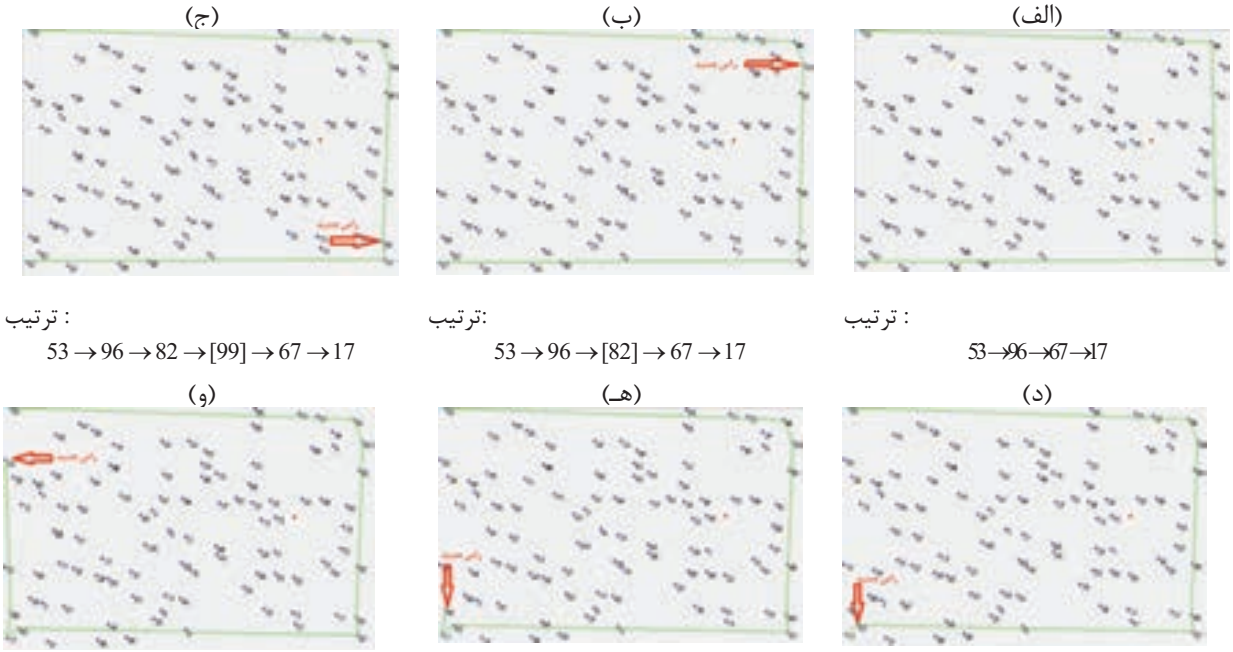
در شکل ۷، بخشی از ایجاد یک مسیر در الگوریتم ابتکاری MASS_C ارائه شده است که نشان می‌دهد که دورترین شهر برای ورود به مسیر موقت انتخاب می‌شود، الگوریتم به گونه‌ای مسیر را ایجاد می‌کند که کم‌هزینه‌ترین باشد. طراحی این قسمت از الگوریتم با این شیوه صورت گرفت که به محض ورود رأس جدید، فاصله تمام رأس‌های مسیر قبلی با آن مورد ارزیابی قرار می‌گیرد تا دو تا از نزدیک‌ترین رئوس به آن انتخاب شوند و مسیر جدید به گونه‌ای ایجاد شود که رأس جدید مابین نزدیک‌ترین رئوس به خود قرار بگیرد. این روش از ابتدا تا انتهای روند حل مسئله حفظ می‌شود و باعث می‌گردد که از ابتدا مسیر به بهترین شکل ایجاد شود و در هر مرحله مسیر ایجادشدهٔ قبلی از دست نرود و با ترمیمی جزئی و بهینه سعی بر تکمیل خود داشته باشد (شکل ۸).

مقایسه نتایج حاصل از سه الگوریتم پیاده‌سازی‌شده حاکی از این بود که الگوریتم ژنتیک تلفیق‌شده با «جست‌وجوی محلی» توانست در برخی تکرارها جواب‌های بهتری را در مقایسه با الگوریتم ابتکاری MASS_C به‌دست بیاورد، اما این برتری نیز چشمگیر نبود. با وجودی‌که جواب‌های «جست‌وجوی محلی ژنتیک» همواره بهتر از جواب‌های الگوریتم NN و حتی NN-2OPT بود، بررسی بیشتر روی ۲۵ بار اجرای جداگانهٔ الگوریتم ژنتیک حاکی از آن بود که الگوریتم ابتکاری از «میانگین جواب‌های ژنتیکی»، جواب‌های بهتری را ارائه می‌کند (شکل ۹).

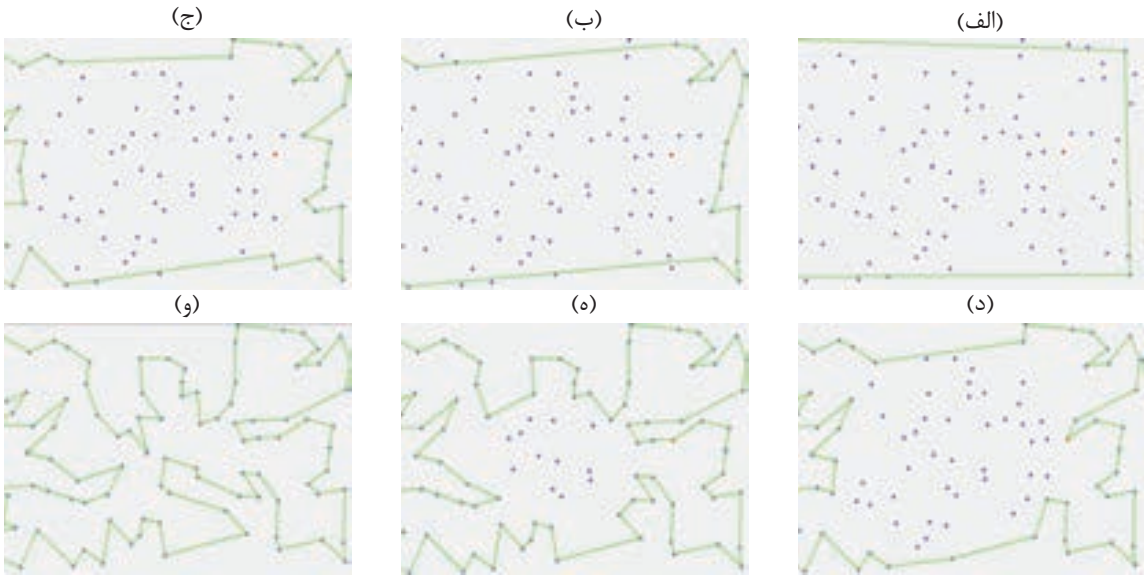
نکتهٔ دیگر یک‌مرحله‌ای بودن الگوریتم MASS_C است که ساختار «ایجادکننده» آن موجب می‌شود در یک تکرار و زمانی بسیار کمتر از الگوریتم ژنتیک، جواب‌های مورد قبولی برای مسئله فروشنده دوره‌گرد ارائه شود. به عنوان نمونه در حل مسئله فروشنده دوره‌گرد برای ۳۰ شهر، الگوریتم MASS_C در کمتر از یک ثانیه جواب را ارائه می‌کند و الگوریتم ژنتیک ۱۱ دقیقه طول می‌کشد تا به هم‌گرایی برسد. از طرف دیگر، مدت زمانی که طول می‌کشد تا الگوریتم ژنتیک به جوابی به شایستگی جواب MASS_C برسد برای

۳۰ شهر بیش از ۹ دقیقه است. افزون بر این برای تعداد بیشتر شهرها ـ مثلاً برای ۱۰۰ شهر ـ الگوریتم MASS_C در کمتر از ۲ ثانیه جوابی را به‌دست می‌دهد که جست‌وجوی محلی ژنتیک پس از ۴۲۰ دقیقه به آن می‌رسد. بررسی نتیجه نهایی ژنتیک با MASS_C نشان داد که با توجه به خاصیت جست‌وجوی تصادفی ژنتیک، این الگوریتم همواره نمی‌تواند به جواب‌های بهتری از MASS_C برسد. مثلاً در تعداد ۲۵ بار اجرای جداگانهٔ جست‌وجوی ژنتیک، ۶۹/۳ درصد از جواب‌ها از جواب MASS_C بهتر نبودند. همچنین میزان برتری جواب‌های بهتر نیز غالباً بسیار خفیف و اندک بود. پس از بررسی نتایج حاصل از الگوریتم MASS_C برای بهبود هر چه بیشتر جواب‌های نهایی، مسیر نهایی MASS_C وارد عملگر جست‌وجوی محلی 2-OPT گردید. این عملگر برای مسیر تعداد شهرهای کمتر (مثلاً ۳۰ شهر) نتوانست بهبودی روی مسیر MASS_C اعمال کند، اما با افزایش تعداد شهرها (۱۰۰ شهر) مشاهده گردید که به میزان ۵ تا ۱۰ درصد بهبود مسیر را منجر می‌گردد. اعمال 2-OPT روی ۱۰۰ شهر نشان داد که مسیر در دو مرحله اجرای 2-OPT دو بار بهبود می‌یابد (شکل ۱۰). اما با بیش از دو بار تکرار، تغییری در مسیر تولیدی رخ نمی‌دهد.

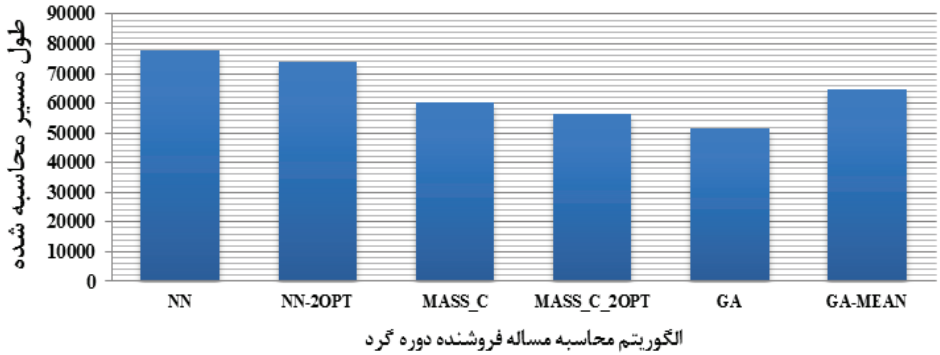
الگوریتم تلفیقی MASS_C و 2-OPT برای ۱۰۰ شهر در زمانی در حدود چهار ثانیه مسیرهای بسیار مناسبی را تولید می‌کند. مقایسه روش‌های پیاده‌سازی‌شده در این پژوهش نشان داد که الگوریتم MASS-C با سرعتی بالا می‌تواند خروجی‌های مناسبی را برای کارکردهای مختلف که نیازمند تولید مسیرهای مناسب در زمان‌های مناسب هستند، ارائه کند (شکل ۹).



شکل ۷. مرحله واردسازی رأس جدید (دورترین از مرکز هندسی) به ترتیبی که مسیر تولیدی کمترین هزینه را داشته باشد.



شکل ۸. حل مسئله فروشنده دوره‌گرد با استفاده از الگوریتم ابتکاری MASS_C برای ۱۰۰ شهر



شکل ۹. مقایسه روش‌های مختلف در حل مسئله فروشنده دوره‌گرد

شکل ۱۰. بهبود مسیر تولیدی به‌وسیله الگوریتم MASS_C با استفاده از عملگر 2-OPT

۴- نتیجه‌گیری

مسئله مسیریابی فروشنده دوره‌گرد در این پژوهش با چندین روش تلفیقی و ابتکاری بررسی و حل گردید. هدف کلی عبارت بود از یافتن مسیر بهینه‌ای که از ۱۰۰ شهر مورد نظر عبور کند. میزان شایستگی هر مسیر با استفاده از محاسبه طول کل مسیر اختصاص‌دهی شد. در الگوریتم ژنتیک از عملگرهای جست‌وجوی محلی 3-OPT برای اجرای عملگر جهش بهره‌گیری گردید. پیاده‌سازی با روش NN به دو شیوه صورت گرفت. ابتدا NN اجرا شد و سپس برای بهبود نتایج آن یک عملگر جست‌وجوی محلی 2-OPT نیز به آن اضافه گردید. علاوه بر اینها روشی ابتکاری نیز برای حل مسئله فروشنده دوره‌گرد در این مقاله پیشنهاد گردید که براساس محاسبه فاصلهٔ مرحله به مرحله از مرکز هندسی می‌کوشد بهترین و محدب‌ترین چندضلعی را به عنوان مسیر پیشنهادی به تمامی شهرهای پیشنهادی به گونه‌ای برازش دهد که قیود مسئله همواره رعایت شده باشند.

حل مسئله فروشنده دوره‌گرد با استفاده از جست‌وجوی محلی ژنتیک، مسیرهای کوتاه‌تری را در مقایسه با الگوریتمNN و NN-2OPT ارائه می‌کند. مقایسه نتایج حاصل از الگوریتم ژنتیک روی مسیریابی فروشنده دوره‌گرد با روش ابتکاری پژوهش حاضر، حاکی از این بود که الگوریتم ژنتیک با توجه به ماهیت «جست‌وجوی تصادفی» خود در برخی تکرارها مسیرهای کوتاه‌تری را در مقایسه با الگوریتم MASS_C ارائه کرد. اما مقایسه نتایج مربوط به مسیر پیشنهادی برای میانگین تکرارهای ژنتیک با الگوریتم ابتکاری حاکی از کوتاه‌تر بودن مسیر پیشنهادی از متوسط طول مسیرهای به‌دست‌آمده از جست‌وجوی محلی ژنتیک است. جست‌وجوی محلی ژنتیک به مسیرهای کوتاه‌تری منتج می‌شود، اما این جواب‌ها در مقایسه با روش‌های دیگر، در زمان بسیار طولانی‌تری به دست می‌آید. با توجه به «یک‌مرحله‌ای بودن» و در نتیجه سریع بودن الگوریتم شرح و بسط داده شده (MASS-C) می‌توان از آن به عنوان روش جدیدی در مسیریابی بهره‌گیری کرد. البته توجه به این نکته نیز ضروری است که الگوریتم طراحی‌شده قابلیت مسیریابی برای تعداد شهرهای بیشتر را نیز داراست.

مدت زمان حل مسئله فروشنده دوره‌گرد با الگوریتم MASS_C روی رایانه‌ای با مشخصات Intel Core 2 Duo CPU T7300 2.00GHz و 2.00 GB RAM، برابر با یک ثانیه و ۸۷ صدم ثانیه بود، که چندین هزار برابر سریع‌تر از الگوریتم قدرتمند ژنتیک روی صد شهر توانست جواب‌های بهینه را ارائه کند. الگوریتم MASS_C با توجه به کارایی مناسب در تولید جواب‌های نهایی مسئله فروشنده دوره‌گرد با سرعت و دقت مناسبی می‌تواند در کاربردهای سرویس‌های تحت وب مسیریابی بسیار مورد استفاده قرار بگیرد.

مسئله فروشنده دوره‌گرد یکی از مسائل پرکاربرد مهندسی قلمداد می‌شود. این مسئله در طراحی سخت‌افزار، وسایل رادیوالکتریک، مخابرات، ساختاردهی شبکه‌های رایانشی و مانند اینها به کار گرفته می‌شود برخی مسائل صنعتی مانند Machine Scheduling، Cellular Manufacturing و مسائل تخصیص فرکانس را می‌توان به عنوان TSP فرمول‌بندی کرد. نگارندگان در این پژوهش مسئله TSP را در حالت کلی و بدون هیچ شرط اضافی حل کرده‌اند؛ چرا که در صورت حل مسئله در حالت کلی می‌توان به سمت سفارشی‌سازی آن برای کاربردهای مختلف پیش رفت. به عنوان نمونه برای مسیریابی روی شبکه موجود راه ـ که گراف کامل نیست ـ در الگوریتم MASS_C در مرحله برازش چندضلعی، شرط «انتخاب از یال‌های موجود» اضافه می‌گردد. نگارندگان پس از ارائه روش جدید خود برای حل مسئله بنیادی فروشندهٔ دوره‌گرد، در پژوهش‌های آینده سعی در ارائه نسخه‌های مختلف آن در کاربردهای مکان‌محور خواهند کرد.

در این مقاله TSP تک‌هدفه مورد بررسی قرار گرفت اما می‌توان در مسائل مسیریابی علاوه بر بعد مسافت، موارد و هزینه‌های دیگری را نیز با توجه به نوع

مسئله در نظر گرفت. در این حالت می‌توان از الگوریتم‌های تکاملی چندهدفه (MOEA) برای حل مسئله مسیریابی استفاده کرد.

۵- منابع

Bakhouya. M., and J. Gaber, 2007**, An Immune Inspired-based Optimization Algorithm: Application to the Traveling Salesman Problem, AMO**, Advanced Modeling and Optimization, Vol. 9, No. 1.

Bentley JL, 1992, **Fast Algorithms for Geometric Traveling Salesman Problems**, ORSA Journal on Computing, 4, 387–411.

Bock F., 1958, **An Algorithm for Solving Traveling Salesman and Related Network Optimization Problems**, unpublished Manuscript Associated with talk presented at the 14th ORSA National Meeting.

Brady RM, 1985, **Optimization Strategies Gleaned From Biological Evolution**, Nature, 317, 804–806.

Croes GA., 1958, **A Method for Solving Traveling Salesman Problems**, Operations Research, 6, 791–812.

Dantzig GB, Fulkerson DR, Johnson SM, 1954, **Solution of a Large Scale Traveling Salesman Problem**, Operations Research, 2, 393–410.

Deb K., 2001, **Multi-Objective Optimization using Evolutionary Algorithms**, Wiley, Chichester.

Freisleben B, Merz P., 1996, **New Local Search Operators for Traveling Salesman Problem**, 4th International Conference on Parallel Problem Solving from Nature, PPSN IV, September, Vol. 1141 of LNCS, pp. 22–26, Springer, Berlin Heidelberg New York.

Goldberg DE., 1989, **Genetic Algorithms in Search, Optimization, and Machine Learning**, Addison-Wesley, New York.

Holland, H.J et. al., 1989, **Induction: Processes of Inference, Learning, and Discovery**, MIT Press.

Johnson DS, McGeoch LA, 1997, **The Traveling Salesman Problem: A Case Study in Local Optimization**, Aarts EHL, Lenstra JK (ed.) Local Search in Combinatorial Optimization, pp. 215–310, Wiley New York.

Krasnogor N., Aragon A., Pacheco J., 2006, **Memetic algorithms, Metaheuristics in Neural Networks Learning**, Kluwer, Dordrecht.

Kumar. R, Singh. P.K., 2007, **Pareto Evolutionary Algorithm Hybridized with Local Search for Biobjective TSP**, Studies in Computational Intelligence (SCI)75, 361-398.

Lawrence. V., A. Snyder and M. S. Daskin, 2005, **A Random-key Genetic Algorithm for the Generalized Traveling Salesman Problem**, Annals of Operations Research, 21, 59–84.

Lawyer EL, Lenstra JK, Rinnooy Kan AHG, Shmoys DB, 1985, **The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization**, Wiley, New York.

Li W., 2005, **Finding Pareto-optimal Set by Merging Attractors for a Biobjective Traveling Salesman Problem**, 3rd International Conference on Evolutionary Multicriterion Optimization (EMO 2005), 9–11. March, vol. 3410 of LNCS, pp. 797–810, Springer, Berlin Heidelberg New York.

Lin S., 1965, **Computer Solutions of the Traveling Salesman Problem**. Bell System Technical Journal, 44, 2245–2269.

Lin S, Kernighan BW., 1973, **An Effective Heuristic Algorithm for the Traveling Salesman Problem**, Operations Research, 21, 498–516.

Liu, Y.H., Jou, R.C., Wang, C., 2005, **Genetic Algorithms for the Probabilistic Traveling Salesman Problem**, In: Proceedings of the conference on E-logostics. Taoyuan, Taiwan, pp. 77-82.

Miyamoto K, Yasuda K., 2005, **Multipoint Based Tabu Search Using Proximate Optimality Principle**, International Conference on Systems, Man and Cybernatics, 10–12 October, vol. 4, pp. 3094–3099, IEEE.

Marinikas. Y, Migdalas. A, Pardalos. P., 2007, **Expanding Neighborhood Search – GRASP for the Probabilistic Traveling Salesman Problem**, Optimization letters 2: 351-361. Springer-Verlag.

Menger K., 1932, **Das Botenproblem**, Ergebnisse Eines Mathematischen Kolloquiums, 2, 11–12.

Merz P., 2002, **A Comparison of Memetic Recombination Operators for the Traveling Salesman**, Genetic and Evolutionary Computation Conference (GECCO'02), New York, USA, July, pp. 472–479.

Kaufmann, M.Los Altosor I., 1976, **Traveling Salesman-Type Combinatorial Problems and their Relation to the Logistics of Regional Blood Banking**, Ph.D. thesis, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL.

Or I., 1976, **Traveling Salesman-Type Combinatorial Problems and their Relation to the Logistics of Regional Blood Banking**, Ph.D. thesis, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL.

Rechenberg, I., 1965, **Cybernetic Solution Path of an Experimental**, Problem library translation No. 1122. Royal Aircraft Establishment.

Soak SM, Ahn BH, 2003, **New Subtour-based Operator for Tsp. Genetic and Evolutionary Computation Conference (GECCO'03)**, vol. 2724 of LNCS, pp. 1610–1611, Springer, Berlin Heidelberg New York.

Tao G, Michalewicz Z., 1998, **Inver-over Operator for the Tsp. Parallel Problem Solving from Nature**, PPSN V, vol. 1498 of LNCS, pp. 803–812, Springer, Berlin Heidelberg New York.