بسم الله الرحمن الرحیم

دانشکده ریاضی و علوم کامپیوتر

پایان‌نامه برای دریافت درجه کارشناسی ارشد در رشته ریاضی کاربردی

گرایش تحقیق در عملیات

حل مسئله فروشنده دوره گرد با انتخاب هتل با روش های فراابتکاری

استاد راهنما

دکتر محمود امین طوسی

استاد مشاور

دکتر محمدعلی پرتانیان

پژوهشگر:

فاطمه کیقبادی

تیر ۱۳۹۷

باسمه تعالی

فرم ارزشیابی و صورتجلسه دفاع از پایان‌نامه کارشناسی ارشد

فرم ۱۱۳-ت

جلسه دفاع از پایان‌نامه آقای /خانم **فاطمه کیقبادی** دانشجوی رشته ریاضی کاربردی گرایش تحقیق در عملیات به شماره دانشجویی ۹۴۲۳۱۳۳۰۲۳ با عنوان:

حل مسئله فروشنده دوره گرد با انتخاب هتل با روش های فراابتکاری

در مورخه ..... در دانشکده ریاضی و علوم کامپیوتر تشکیل و توسط هیات داوران مورد ارزشیابی قرار گرفت و نمره ..... برابر درجه ..... برای آن تعیین گردید .

به‌این ترتیب از این تاریخ آقای/ خانم فاطمه کیقبادی به عنوان کارشناس ارشد در رشته مذکور شناخته می‌شود .

موارد ارزشیابی	موارد	حداکثر نمره	نمره کسب شده
۱- کیفیت نگارش	رعایت اصول نگارش انسجام در تنظیم بخشهای مختلف ،کیفیت تصاویر ، جداول و اشکال ،تنظیم فهرست ها ، منابع و ماخذ	٤	
۲- کیفیت علمی	بررسی تاریخچه و سابقه تجربی و نظری موضوع انسجام منطقی دربخش های مختلف پایان نامه ،ابتکار و نوآوری ،اهمیت و ارزش علمی پایان نامه ،استفاده از منابع معتبر و جدید ،کیفیت تجزیه و تحلیل یافته ها و نتیجه گیری ،روشن بودن روش کار ،هدف ها و فرضیه های تحقیق ،جدید بودن روش تحقیق	۱۰	
۳-کیفیت ارایه در جلسه دفاع	تسلط بر موضوع و بیان واضح وتفهیم آن ،توانایی در پاسخگویی به سوالات مطرح شده در جلسه ، رعایت زمان ارایه , روش ارائه	٤	
٤-ارزشیابی گزارشات	گزارش های دوره ای پیشرفت کار(حداقل ٤مورد)	۱	
٥- خروجی پایان نامه	مقاله مستخرج از پایان نامه: این نمره به صورت زیر اختصاص می یابد ۱) چکیده کنفرانسی هر مورد ۰/۲۵ نمره تا سقف۰/۵ نمره ۲) مقاله کامل در مجموع مقالات همایشهای معتبر یا مقاله در مجلات علمی-ترویجی معتبر پذیرفته شده یا چاپ شده هر مورد ۰/۵ نمره تا سقف ۱نمره ۳)مقاله پذیرفته شده یا چاپ شده در مجلات علمی پژوهشی معتبر ۱ نمره ٤)مقاله ارسال شدهبه مجلات علمی پژوهشی معتبرهر مورد ۲۵/. نمره تا سقف ۰/۵ نمره ٥) دستگاه ساخته شده دارای گواهی ثبت اختراع یا به سفارش سازمان ها تا سقف ۱ نمره ٦) دستگاه ساخته شده کاربردی که به تایید رئیس دانشکده رسیده باشد تا سقف ۰/۵ نمره	۱	
جمع			

درجه معادل کسب شده: (از ۱۹تا ۲۰ عالی☐ از۱۸تا ۱۸/۹۹بسیارخوب ☐ از ۱٦تا ۱۷/۹۹خوب ☐ از ۱٤ تا ۱۵/۹۹قابل قبول☐ کمتر از ۱٤ غیر قابل قبول☐)

مشخصات هیات دوران

ردیف	نام و نام خانوادگی	سمت	مرتبه علمی	محل کار	امضـــــــــــا
۱	دکتر محمود امین طوسی	استاد راهنما	استادیار	دانشگاه حکیم سبزواری	
۲	دکتر محمدعلی پرتانیان	استاد مشاور	استادیار	دانشگاه حکیم سبزواری	
۳	دکتر علی‌اصغر مولوی	استاد داور	استادیار	دانشگاه حکیم سبزواری	
۴	دکتر غلامرضا مقدسی	نماینده تحصیلات تکمیلی	استادیار	دانشگاه حکیم سبزواری	

امضا

مدیر گروه

امضا

رئیس دانشکده

سوگند نامه دانش آموختگان دانشگاه حکیم سبزواری

به نام خداوند جان و خرد کزین برتر اندیشه بر نگذرد

اینک که به خواست آفریدگار پاک، کوشش خویش و بهره گیری از دانش استادان و سرمایه‌های مادی و معنوی این مرز و بوم، توشه‌ای از دانش و خرد گردآورده‌ام، در پیشگاه خداوند بزرگ سوگند یاد می کنم که در به کارگیری دانش خویش، همواره بر راه راست ودرست گام بردارم. خداوند بزرگ، شما شاهدان، دانشجویان و دیگر حاضران را به عنوان داورانی امین گواه می گیرم که ازهمه دانش و توان خود برای گسترش مرزهای دانش بهره گیرم و از هیچ کوششی برای تبدیل جهان به جایی بهتر برای زیستن، دریغ نورزم. پیمان می‌بندم که همواره کرامت انسانی را در نظر داشته باشم و همنوعان خود را در هر زمان و مکان تا سر حد امکان یاری دهم. سوگند می خورم که در به کارگیری دانش خویش به کاری که با راه و رسم انسانی، آیین پرهیزگاری، شرافت و اصول اخلاقی برخاسته از ادیان بزرگ الهی، به ویژه دین مبین اسلام، مباینت دارد دست نیازم. همچنین در سایه اصول جهان شمول انسانی و اسلامی، پیمان می‌بندم از هیچ کوششی برای آبادانی و سرافرازی میهن و هم میهنانم فروگذاری نکنم و خداوند بزرگ را به یاری طلبم تا همواره در پیشگاه او و در برابر وجدان بیدار خویش و ملت سرافراز، بر این پیمان تا ابد استوار بمانم.

نام و نام خانوادگی: فاطمه کیقبادی

تاریخ و امضا:

تأییدیه‌ی صحت و اصالت نتایج

باسمه تعالی

اینجانب فاطمه کیقبادی به شماره دانشجویی ۹۴۲۳۱۳۳۰۲۳ دانشجوی رشته ریاضی کاربردی مقطع تحصیلی کارشناسی ارشد تأیید می‌نمایم که کلیه‌ی نتایج این پایان‌نامه حاصل کار اینجانب و بدون هرگونه دخل و تصرف است و موارد نسخه برداری شده از آثار دیگران را با ذکر کامل مشخصات منبع ذکر کرده‌ام. درصورت اثبات خلاف مندرجات فوق، به تشخیص دانشگاه مطابق با ضوابط و مقررات حاکم (قانون حمایت از حقوق مؤلفان و مصنفان و قانون ترجمه و تکثیر کتب و نشریات و آثار صوتی، ضوابط و مقررات آموزشی، پژوهشی و انضباطی ...) با اینجانب رفتار خواهد شد و حق هرگونه اعتراض درخصوص احقاق حقوق مکتسب و تشخیص و تعیین تخلف و مجازات را از خویش سلب می‌نمایم. در ضمن، مسؤولیت هرگونه پاسخگویی به اشخاص اعم از حقیقی و حقوقی و مراجع ذی صلاح (اعم از اداری و قضایی) به عهده ی اینجانب خواهد بود و دانشگاه هیچ گونه مسؤولیتی در این خصوص نخواهد داشت.

نام و نام خانوادگی: فاطمه کیقبادی

تاریخ و امضا:

مجوز بهره برداری از پایان‌نامه

بهره برداری از این پایان‌نامه در چهارچوب مقررات کتابخانه و با توجه به محدودیتی که توسط استاد راهنما به شرح زیر تعیین می‌شود، بلامانع است:

☐ بهره برداری از این پایان‌نامه برای همگان بلامانع است.

☐ بهره برداری از این پایان‌نامه با اخذ مجوز از استاد راهنما، بلامانع است.

☐ بهره برداری از این پایان‌نامه تا تاریخ ..... ممنوع است.

استاد راهنما: دکتر محمود امین طوسی

تاریخ و امضا:

تقدیم به:

پدر و مادرم

سپاس خداوندگار حکیم را که با لطف بی کران خود، آدمی را زیور عقل آراست.
در آغاز وظیفه خود می دانم از زحمات بی دریغ استاد راهنمای خود، جناب آقای دکتر محمود امین طوسی، صمیمانه تشکر و قدردانی کنم که قطعاً بدون راهنمایی های ارزنده ایشان، این مجموعه به انجام نمی رسید.
از جناب آقای دکتر محمد علی پرتانیان که زحمت مطالعه و مشاوره این رساله را تقبل فرمودند و در آماده سازی این رساله، به نحو احسن اینجانب را مورد راهنمایی قرار دادند، کمال امتنان را دارم.
همچنین لازم می دانم از گروه پارسی‌لاتک در پاسخگویی به مشکلات کاربران کمال قدردانی را داشته باشم.
در پایان، بوسه می زنم بر دستان خداوندگاران مهر و مهربانی، پدر و مادر عزیزم و بعد از خدا، ستایش می‌کنم وجود مقدس شان را و تشکر می‌کنم از خانواده عزیزم به پاس عاطفه سرشار و گرمای امیدبخش وجودشان، که بهترین پشتیبان من بودند.

فاطمه کیقبادی

تیر ۱۳۹۷

فهرست مطالب

فهرست جداول

فهرست تصاویر

فرم چکیده ی پایان‌نامه ی دوره ی تحصیلات تکمیلی

مدیریت تحصیلات تکمیلی

دانشگاه حکیم سبزواری

نام خانوادگی دانشجو:کیقبادی	نام:فاطمه	ش. دانشجویی:۹۴۲۳۱۳۳۰۲۳
استاد راهنما: دکتر محمود امین طوسی		
استاد مشاور: دکتر محمدعلی پرتانیان		
دانشکده ریاضی و علوم کامپیوتر	رشته: ریاضی کاربردی	گرایش: تحقیق در عملیات
مقطع: کارشناسی ارشد	تاریخ دفاع:تیر ۱۳۹۷	تعداد صفحات: ؟؟
عنوان پایان‌نامه: حل مسئله فروشنده دوره گرد با انتخاب هتل با روش های فراابتکاری		
کلید واژه‌ها:الگوریتم ممتیک، مساله فروشنده‌دوره‌گرد با انتخاب هتل، جستجوی محلی، جستجوی ممنوعه		

چکیده: در این پایان‌نامه برخی از رویکردهای مسئله فروشنده‌دوره‌گرد با انتخاب هتل (TSPHS) مورد بررسی قرار گرفته‌است. TSPHS یک نوع مشهور از مسئله TSP است. TSP یک مسئله بهینه‌سازی است که هدف آن یافتن بهترین مسیر ممکن برای بازدید از همه شهرها و بازگشت به نقطه شروع با حداقل هزینه سفر است. TSPHS یک TSP با محدودیت اضافی که در مورد زمان سفر در روز است، می‌باشد. هدف این مسئله این‌است که یک فروشنده اغلب نمی‌تواند از تمام شهرها در یک روز بازدید کند، زیرا تنها ساعات محدودی در روز کار می‌کند یعنی فروشنده باید هرشب یک هتل را انتخاب کند، بطوریکه در نهایت یک دنباله بهینه از تمام شهرهای بازدید شده داریم. سفر هر روز باید در یکی از هتل‌های موجود شروع و به پایان یابد و اگر در روز خاص در یک هتل خاص پایان یابد روز بعد سفر باید از همین هتل شروع شود. هدف اصلی این مسئله این است که تعداد روزهای سفر به حداقل رسد در حالی‌که دف ثانویه به حداقل رساندن طول کل سفر نیز است. اگرچه این مسئله شبیه مسئله TSP است، اما به دلیل شرط اتخاب هتل بسیار دشوارتر است. در این پایان‌نامه برخی روش‌های حل TSP یا TSPHS شامل الگوریتم برنامه‌ریزی پویا، الگوریتم انشعاب‌وتحدید، الگوریتم ژنتیک، الگوریتم ممتیک، تابوسرچ و الگوریتم شبیه‌سازی تبریدی شرح داده شده‌است.

پیش‌گفتار

مسئله فروشنده‌دوره گرد TSP از جمله مسائل مهم و کاربردی در علوم کامپیوتر و تحقیق در عملیات است، این مسئله جز مسائل NP-Complete است. در این پایان نامه مسئله فروشندهدوره‌گرد خاص باانتخاب هتل را مورد بررسی قرار می‌دهیم. مسئله فروشنده‌دوره‌گرد با انتخاب هتل، اخیرا توسط ونستین وجن، [۱] وهمکارانش معرفی شده‌است [؟]. در این مسئله، با توجه به اینکه فروشنده تنها ساعات محدودی در روز کار می‌کند، ممکن است قادر به ملاقات تمام شهرها در یک روز نباشد، در این‌حالت فروشنده هر شب به انتخاب یک هتل احتیاج دارد. هدف، یافتن دنباله ای بهینه برای بازدید از تمام شهرها است، بطوریکه هرروز درصورت لزوم باید در یکی از هتل های موجود شروع و در هتلی پایان یابد.

هدف اصلی این مساله، کمینه کردن طول سفر و همزمان به حداقل رساندن تعداد روزها است. به خاطر به حداقل رساندن تعداد روزها و انتخاب هتل این مسئله خیلی دشوارتر از مسئله فروشنده دوره گرد [۲] است. در [؟] از الگوریتم ژنتیک استفاده شده است، به این صورت که یک جواب اولیه با دو روش انتخاب می‌شود در روش اول یک دنباله از شهر ها در نظر گرفته می‌شود، اما در روش دوم یک دنباله از هتل ها را داریم که در این روش با قرار دادن تک تک شهر ها در اولین سفر، تا زمانیکه در محدودیت حداکثر طول سفر برقرار باشد انجام می‌شود.

این پایان نامه شامل سه فصل است:

در فصل اول، مقدمه و تعاریف مورد نیاز مساله بیان می‌شود.

در فصل دوم، روش حلی برای مسئله فروشنده دوره گرد با انتخاب هتل بیان می‌شود.

در فصل سوم، به بیان مساله و روش های به کارگیری پرداخته می‌شود.

- Vansteenwegen, Pieter, Souffriau, Wouter, and Sörensen, Kenneth. The travelling salesperson problem with hotel selection. *Journal of the Operational Research Society*, 63(2):207–217, 2012.

- Mohsen, Abdulqader M and Al-Sorori, Wedad. A New Hybrid Discrete Firefly Algorithm for Solving the Traveling Salesman Problem. *Applied Computing and Information Technology*, 169–180,2017.

[۱] Vansteenwegen [۲] Travelling salesman problem

فصل ۱

معرفی مساله فروشنده دوره گرد با انتخاب هتل

این فصل دربرگیرنده تاریخچه، مفاهیم مورد نیاز پایان‌نامه است. در بخش اول به تعریف مفاهیم اولیه و شرح مسئله فروشنده‌دوره‌گرد پرداخته شده‌است. بخش دوم به تعریف چند الگوریتم برنامه نویسی پویا، انشعاب وتحدید، تپه نوردی، تبریدی و ژنتیک در مسئله فرووشنده‌دوره‌گرد، مقایسه چند تا از الگوریتمها و انتخاب بهترین آنها و حل مسئله فروشنده‌دوره‌گرد با بهینه‌سازی ازدحام ذرات پرداخته شده‌است. و در نهایت در بخش سوم به ذکر الگوریتم کرم شب تاب برای مسئله اختصاص داده شده‌است.

۱-۱ مقدمه

با گسترش روز افزون جوامع و رشد جمعیت، نیاز به صرفه جویی و یافتن روش‌هایی برای به حداقل رساندن زمان و هزینه در انجام امور صنعتی، عمرانی و غیره هر روز بیشتر ضرورت پیدا می‌کند. جهت نیل بدین مقصود، روش های بهینه‌سازی مورد استفاده قرار می‌گیرند. منظور از روش‌های بهینه‌سازی، روش‌هایی است که می‌توان توسط آن‌ها با حداقل زمان و هزینه به نتایج مطلوب رسید. مسائل مربوط به فروشنده دوره گردابتدا در سده ۱۸ توسط ویلیام هامیلتون [۱] و توماس کرکمن [۲] مطرح شد و سپس در دهه ۱۹۲۰ شکل عمومی آن توسط ریاضی دانانی چون کارل منگر [۳] از دانشگاه هاروارد مورد مطالعه قرار گرفت [؟]. این پایان نامه به بررسی مسئله فروشنده‌دوره‌گرد با وجود شرط اضافه درخصوص زمان سفر در روز می‌پردازد و روش حل با الگوریتم ژنتیک را مورد بررسی قرار می‌دهد.

در مسئله فروشنده‌دوره‌گرد تعدادی شهر داریم و هزینه رفتن مستقیم از هر شهر به شهر دیگر را می‌دانیم. مطلوب است یا کوتاه‌ترین مسیری که از یک شهر شروع شود و از تمام شهر ها یک بار عبور کند و به شهر شروع باز گردد.

در مسئله فروشنده‌دوره‌گرد با انتخاب هتل تعدادی شهر و تعدادی هتل و یک مدت زمان معین برای بازدید شهرها در

---

[۱] Wiliam Hamilton [۲] Thomas Kirkman [۳] Carl Menger

طول روز را داریم، در روز کاری به بازدید از شهرها پرداخته می‌شود و وقتی بازه زمانی به پایان رسید ملاقات کننده، به نزدیک ترین هتل نسبت به آن شهر می‌رود و استراحت می‌کند و روز بعد سفر خود را آغاز می‌کند.

۱-۱-۱ تاریخچه

روش‌های زیادی برای حل مسئله فروشنده‌دوره‌گرد ارائه شده‌است. از جمله این روش ها می توان به حل مسئله به روش اجتماع مورچه ها در سال ۱۹۹۷ [؟] ، با استفاده از الگوریتم بهینه‌سازی ذرات در سال ۲۰۱۲ [؟] و روش برنامه‌ریزی پویا در سال ۲۰۱۴ [؟] اشاره کرد. مسئله فروشنده‌دوره‌گرد باانتخاب هتل با روش الگوریتم ممتیک در سال ۲۰۱۵ [؟] ، با روش فراابتکاری سریع در سال ۲۰۱۵ [؟] ، با روش الگوریتم‌های دقیق و اکتشافی در سال ۲۰۱۵ [؟] و با روش الگوریتم پرندگان مهاجر در سال ۲۰۱۵ [؟] است.

۱-۱-۲ مفاهیم

در این بخش، مفاهیمی از مساله فروشندهدوره‌گرد با انتخاب هتل شرح داده می شود.

تعریف ۱-۱-۱. هر گراف G زوج مرتبی مانند (V, E) است که در آن V مجموعه‌ای متناهی و ناتهی و E زیرمجموعه‌ای از تمام زیرمجموعه‌های دو عضوی V می‌باشند. اعضای V را راس‌های G و اعضای E را یال‌های G می‌نامند. همچنین اگر بین دو راس یک گراف حداکثر یک یال موجود باشد و هیچ طوقه‌ای (یعنی یالی که راس را به خودش وصل می‌کند) وجود نداشته باشد، چنین گرفی را گراف ساده گویند.

تعریف ۱-۱-۲. گرافی که بین هر دو راس آن دقیقا یک یال وجود داشته باشد یک گراف کامل از مرتبه n است که دارای n راس و $\frac{n(n-۱)}{۲}$ یال است که آن را با k_n نشان می‌دهند. یک گراف کامل یک گراف منتظم از درجه $n-۱$ است.

تعریف ۱-۱-۳. یک دور در یک گراف که از همه رئوس دقیقا یک بار عبور کند را دور همیلتونی نامند.

تعریف ۱-۱-۴. در اکثر زمینه های تحقیقاتی می توان مسئله مورد نظر را به صورت یک مسئله بهینه سازی بیان کرد. هدف از مسائل بهینه‌سازی، پیدا کردن جواب بهینه است. در این نوع مسائل، یک یا چند تابع هدف خطی یا غیر خطی تعریف می شود که باید در عین حال که تمامی محدودیت‌ها برآورده می‌شوند، کمینه یا بیشینه شود. معمولا در مسائل بهینه‌سازی محدودیت هایی نیز تعریف می‌شوند که ممکن است شامل نامعادلات خطی یا غیر خطی و یا گسسته بودن متغیرهای تصمیم باشند. تمامی جواب های موجود باید در این محدودیت ها صدق کنند، در غیر این صورت جواب موجه نخواهد بود. مسائل بهینه‌سازی، از نظر نوع متغیرهای تصمیم به دو گروه مسائل گسسته که مسئله یافتن جواب بهینه در این دسته از مسائل، بهینه سازی ترکیبی [۱] است و مسائلی با متغیرهای پیوسته تقسیم بندی می‌شوند [؟].

[۱] combinatorial optimization

تعریف ۱-۱-۵. فاصله اقلیدسی بین دو نقطه $(x_۱, y_۱)$ و $(x_۲, y_۲)$ را با d_{ij} نشان می‌دهیم:

$$d_{ij} = \sqrt{(x_۱ - x_۲)^۲ + (y_۱ - y_۲)^۲}$$

تعریف ۱-۱-۶. دویال از گراف با دویال دیگر جابه‌جا می‌شوند به این جابه‌جایی $opt - ۲$ گوییم.

تعریف ۱-۱-۷. سه یال از گراف با سه یال دیگر جابه‌جا می‌شوند به این جابه‌جایی $opt - ۳$ گوییم.

۱-۱-۳ مسئله فروشنده دوره گرد

در مسئله فروشنده‌دوره‌گرد تعدادی شهر داریم و هزینه رفتن مستقیم از یکی به دیگری را می‌دانیم. مطلوب است کوتاه‌ترین مسیری که از یک شهر شروع شود و از تمامی شهرها دقیقا یکبار عبور کند و به شهر شروع باز گردد.
تعداد جواب های شدنی مسئله، برابر است با $\frac{۱}{۲}(n-۱)!$ برای $n > ۲$ که n تعداد شهرها است. در واقع این عدد برابر با تعداد دورهای همیلتونی در یک گراف کامل با n راس است. فرمول‌بندی مسئله بدین صورت است:

$$\min \sum_{i=۰}^{n} \sum_{j \neq i, j=۰} C_{ij} x_{ij} \qquad (۱-۱)$$

$$s.t \qquad ۰ \leq x_{ij} \leq ۱ \qquad i, j = ۰, \cdots, n \qquad (۲-۱)$$

$$\sum_{i=۰, i \neq j}^{n} x_{ij} = ۱ \qquad j = ۰, \cdots, n \qquad (۳-۱)$$

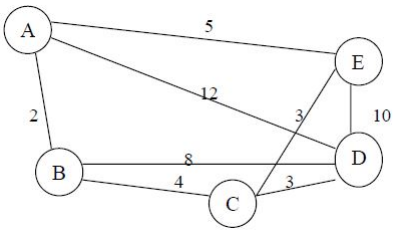
$$\sum_{j=۰, j \neq i}^{n} x_{ij} = ۱ \qquad i = ۰, \cdots, n \qquad (۴-۱)$$

که در اینجا C_{ij} مقدار هزینه یال i به j را معرفی می‌کند. تابع هدف (؟؟) هزینه کل سفر را کمینه می‌کند و محدودیت (؟؟) متغیر دودویی است. محدودیت (؟؟) نشان می دهد که به هر شهر فقط یکبار وارد می‌شویم و در نهایت محدودیت (؟؟) نشان دهنده این است که از هر شهر فقط یکبار خارج می‌شویم.

$$x_{ij} = \begin{cases} ۱ & \text{در تور، از شهر i به شهر j برویم} \\ ۰ & \text{در غیر این صورت} \end{cases}$$

مشکلی که این فرمول‌بندی دارد آن است که ممکن است تور حاصل شامل چندین زیرتور باشد.

مثال ۱-۱-۸. مجموعه شهر ها با گراف شکل ؟؟ را در نظر بگیرید. هدف پیدا کردن کوتاه‌ترین مسیر به طوری است که از همه رئوس دقیقا یکبار عبور کند و به نقطه شروع باز گردد.



شکل ۱-۱: گرافی با درج وزن روی یال ها

برای مثال دو مسیر A, B, C, E, D, A و A, B, C, D, E, A را در نظر می گیریم همانطور که میبینیم هزینه طول مسیر اول ۲۴ و هزینه طول مسیر دوم ۳۱ است پس مسیر اول بهتر از مسیر دوم است.

۱-۱-۴ پیچیدگی محاسباتی

یک الگوریتم برای حل مسئله، نیاز به دو منبع مهم دارد: زمان و فضای حافظه

تعریف ۱-۱-۹. پیچیدگی زمانی یک الگوریتم، برابر تعداد گام های مورد نیاز برای حل یک مسئله با اندازه n است که n بیانگر ابعاد مسئله است [؟].

پیچیدگی معمولا بر اساس بدترین حالت تعریف می شود. هدف از تعیین پیچیدگی محاسباتی یک الگوریتم، یافتن تعداد دقیق گام های الگوریتم نیست، بلکه هدف، یافتن یک حد بالا برای تعداد گام‌هاست. نماد O بزرگ، یکی از نمادهای رایج در تحلیل پیچیدگی الگوریتم ها است. بنابر تعریف، یک الگوریتم دارای پیچیدگی $f(n) = O(g(n))$ است، اگر ثابت های مثبت n_0 و c وجود داشته باشند، بطوریکه به ازای هر $n > n_0$ نامساوی $f(n) \leq c.g(n)$ برقرار باشد.در این حالت، $g(n)$ حد بالای $f(n)$ است. این نماد می تواند برای محاسبه پیچیدگی زمانی یافضای حافظه ای یک الگوریتم مورد استفاده قرار گیرد. در بحث پیچیدگی محاسباتی الگوریتم ها، می توان آن ها را در یکی از دو گروه زیر در نظر گرفت:

- الگوریتم با زمان چند جمله ای : پیچیدگی این الگوریتم $O(p(n))$ است که در آن $p(n)$ یک تابع چندجمله ای از درجه n است. یک تابع چندجمله ای با درجه k ، به صورت زیر تعریف می‌شود:

$$p(n) = a_k n^k + \cdots + a_j n^j + \cdots + a_1 n + a_0$$

که در آن $a_k > 0$ ، $1 \leq j \leq k - 1$. در نتیجه، این الگوریتم دارای پیچیدگی چند جمله ای $O(n^k)$ است.

- الگوریتم با زمان نمایی : پیچیدگی این الگوریتم $O(c^n)$ است که در آن c یک عدد حقیقی ثابت بزرگتر از یک است.

توجه داشته باشید که جستجوی زمانی یک الگوریتم با توجه به اندازه‌ی مسئله افزایش می یابد. مقایسه زمان محاسباتی الگوریتم‌ها، با توجه به پیچیدگی محاسباتی الگوریتم و اندازه مسئله، زمان محاسباتی بسیار بالا الگوریتم دارای پیچیدگی زمانی نمایی را در مقابل الگوریتم دارای پیچیدگی زمانی چندجمله‌ای نشان می‌دهد.

تعریف ۱-۱-۱۰. پیچیدگی محاسباتی یک مسئله، برابر است با پیچیدگی محاسباتی بهترین الگوریتمی که آن را حل می‌کند.

نظریه پیچیدگی مسائل، بر اساس مسائل تصمیم [۱] تعریف می‌شود. یک مسئله تصمیم، همیشه یک جواب 'بلی' یا 'خیر' دارد. هر مسئله بهینه سازی، قابل تبدیل به یک مسئله تصمیم است. از لحاظ پیچیدگی محاسباتی، مسائل تصمیم به دو کلاس مهم P و NP تقسیم می شوند.

تعریف ۱-۱-۱۱. کلاس P

کلاس پیچیدگی P ، شامل مجموعه ای از مسائل تصمیم است که الگوریتم‌های قطعی برای پیدا کردن جواب آن‌ها وجود دارد. به عبارت دیگر، تمامی مسئله‌های تصمیم که دارای جوابی از مرتبه چندجمله‌ای باشند در این کلاس قرار دارند. تعریف دیگری که از مسئله های کلاس P ارائه می‌شود این است که هرگاه الگوریتمی وجود داشته باشد که بتواند به ازای هر ورودی به طول n در حداکثر cn^k مرحله (k و c معرف اعداد ثابت مستقل از n) جواب درست بدهد، آنگاه گوییم مسئله می‌تواند در زمان چند جمله ای $O(n^k)$ حل شود و آن را در کلاس P قرار می‌دهیم.

تعریف ۱-۱-۱۲. کلاس NP

کلاس پیچیدگی NP ، شامل مجموعه‌ای از مسائل تصمیم است که میتوان آن ها را توسط الگوریتم‌های غیر قطعی چندجمله‌ای حل کرد. الگوریتم های غیر قطعی چندجمله‌ای، شامل الگوریتم‌های دو مرحله‌ای است. در مرحله اول، یک جواب کاندید به صورت تصادفی تولید می‌شود. در مرحله دوم، درستی جواب کاندید بررسی می‌شود. در صورتی‌که جواب کاندید بیان کننده‌ی یک جواب مسئله باشد، مقدار 'بله' و در غیر این صورت مقدار 'خیر' به عنوان جواب نهایی مسئله تصمیم برگردانده می‌شود. بنابراین، غیر قطعی بودن الگوریتم مربوط به مرحله ی اول و زمان چندجمله ای مربوط به مرحله دوم الگوریتم‌هاست. در حالت کلی، تمامی مسائلی‌که جواب آن‌ها توسط الگوریتم با زمان چندجمله‌ای قابل بررسی باشد مسائل NP هستند. بنابراین هر مسئله P یک مسئله NP است.

مسائل NP به دو دسته مسائل NP-Complete و NP-Hard تقسیم می‌شوند:

تعریف ۱-۱-۱۳. کلاس NP-Complete

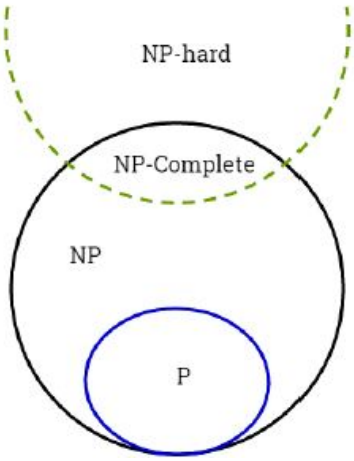
NP-Complete ها مسائلی هستند که به سرعت قابل حل نیستند. در نظریه پیچیدگی، مسائل NP-Complete دشوارترین مسائل NP هستند که قابل حل با الگوریتم @ های قطعی با زمان چندجمله‌ای نیستند. برای مشخص کردن این که آیا مسئله‌ای در این کلاس قرار دارد یا خیر روش اثباتی وجود ندارد و تنها با متناظر کردن مسئله ی جدید با یکی از مسئله هایی که قبلا

[۱] decision problems

در این کلاس قرار گرفته است به گونه ای که راه حل یکی برای دیگری نیز کاربرد داشته باشد، این مسئله جدید نیز در کلاس NP-Complete قرار خواهد گرفت.

تعریف ۱-۱-۱۴. کلاس NP-Hard

برای حل مسائل NP-Hard ، الگوریتم‌های قطعی چند جمله‌ای وجود ندارد. به عبارت دیگر، زمان حل آن ها با افزایش ابعاد مسئله به صورت نمایی افزایش می‌یابد. الگوریتم‌های فرا ابتکاری به عنوان الگوریتم‌های حل مسائل NP-Hard ارائه شده‌اند. این الگوریتم‌ها می‌توانند برای حل این گونه مسائل، یک جواب مناسب را در زمانی قابل قبول به دست آورند. در شکل ؟؟ رابطه بین این کلاس ها به نمایش گذاشته‌شده‌است.



شکل ۱-۲: رابطه بین مسائل کلاس P، NP، NP-Complete، NP-hard

۱-۲ الگوریتم های حل مسئله فروشنده دوره گرد

در این بخش به تشریح الگوریتم‌های اکتشافی مورد استفاده و روش برنامه‌نویسی پویا برای حل مسئله فروشنده‌دوره‌گرد و پیاده‌سازی آن‌ها می‌پردازیم. در تمامی روش‌ها، از یک بردار شامل شهر ها (بدون تکرار شهر) که شهر ابتدایی بردار با انتهای بردار برابر بوده و توالی شهرها، ترتیب ملاقات شهر ها توسط فروشنده را نشان می‌دهد، به عنوان یک جواب استفاده می‌کنیم. یک جواب به‌این صورت است:

$$Solution = [C_0, C_1, \cdots, C_{n-1}, C_0]$$

هدف از حل این مسئله، کمینه‌سازی هزینه انجام شده توسط فروشنده می باشد که به این صورت است:

$$Cost = Min \sum_{i=0}^{i=n-2} C_{i(i+1)} + C_{(n-1)0}$$

روش‌هایی که در ادامه به شرح آن‌ها می‌پردازیم عبارتنداز: الگوریتم برنامه‌ریزی پویا، الگوریتم انشعاب و تحدید، الگوریتم تپه‌نوردی، الگوریتم شبیه‌سازی تبریدی، الگوریتم ژنتیک و الگوریتم بهینه‌سازی ذرات.

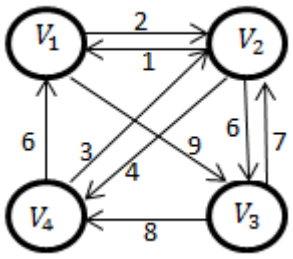
۱-۲-۱ الگوریتم برنامه ریزی پویا

در بسیاری از مسائل، که در آنها رشته‌ای از تصمیمات مرتبط با یکدیگر مطرح باشد، غالبا از برنامه‌ریزی پویا [۱] که دارای ماهیت ریاضی است استفاده می‌شود. این برنامه‌ریزی با بکار گیری فرایندی نظام گرا [۲] ترکیبی از تصمیمهای متوالی را تعیین می کند که به حداکثر شدن کارائی کلی [۳] منتهی شود [؟].

برخلاف برنامه ریزی خطی، چارچوب استانداردی برای فرموله کردن مسائل برنامه‌ریزی پویا وجود ندارد. در واقع آنچه برنامه ریزی پویا انجام می‌دهد ارائه روش برخورد کلی، جهت حل این نوع مسائل است. در هر مورد، باید معادلات و روابط ریاضی مخصوصی که با شرایط آن مسئله تطبیق نماید نوشته شود. از این رو، برای آنکه بتوان تشخیص داد که آیا اصولا می‌توان مسئله ای را با برنامه‌ریزی پویا حل کرد و اگر می‌شود، حل آن چگونه است، ضرورت دارد ساختار کلی برنامه‌ریزی پویا شناخته شود و علاوه بر آن، به خلاقیت نیز تا حدی احتیاج است. آشنائی با انواع کاربرد های برنامه‌ریزی پویا و بررسی ویژگی های مشترک آنها، احتمالا می تواندبه رشد چنین توانائیهایی کمک کند.

برنامه ریزی پویا، از یک جزء کوچک مسئله شروع نموده، جواب بهینه همان جزء را بدست می‌آورد. آنگاه، به تدریج این مسئله کوچک را بزرگتر کرده جواب بهینه آنرا با بهره گیری از جواب بهینه قبلی پیدا می‌کند، تا سرانجام جواب بهینه مسئله اصلی به‌دست آید. نحوه عمل بایک مثال نشان داده می‌شود.

گراف داده‌شده در شکل ؟؟ و نمایش ماتریس مجاورتی آن در ؟؟ را در نظر بگیرید.



شکل ۱-۳: تور بهینه $[V_1, V_3, V_4, V_2, V_1]$ است.

[۱]Dynamic Programming [۲]Systematic Procedure [۳]Overall Effectiveness

هزینه سه تور که از راس $V_۱$ شروع شوند عبارتنداز:

$$[V_۱, V_۲, V_۳, V_۴, V_۱] = ۲۲ \quad [V_۱, V_۳, V_۲, V_۴, V_۱] = ۲۶ \quad [V_۱, V_۳, V_۴, V_۲, V_۱] = ۲۱$$

باتوجه به هزینه به‌دست آمده می بینیم که مسیر $[V_۱, V_۳, V_۴, V_۲, V_۱]$ بهینه می باشد. اگر V_k نخستین راس پس از $V_۱$ روی هر تور بهینه باشد، زیر مسیر آن تور از V_k به $V_۱$ باید کوتاه‌ترین مسیر از V_k به $V_۱$ باشد که از هر کدام از رئوس دیگر دقیقا یک بار عبور می‌کند. یعنی، اصل بهینگی برقرار است و می‌توان از برنامه‌ریزی پویا استفاده کرد. گراف را با یک ماتریس مجاورتی C نشان می‌دهیم. ماتریس مجاورتی گراف ؟؟ به صورت ؟؟ است:

$$\begin{bmatrix} ۰ & ۲ & ۹ & \infty \\ ۱ & ۰ & ۶ & ۴ \\ \infty & ۷ & ۰ & ۸ \\ ۶ & ۳ & \infty & ۰ \end{bmatrix} \tag{۵-۱}$$

فرض کنید V مجموعه همه رئوس گراف، A زیر مجموعه ای از رئوس گراف و $D[V_i][A]$ طول کوتاه‌ترین مسیر از V_i به $V_۱$ که از هر راس مجموعه A دقیقا یکبار می‌گذرد، است. در گراف ؟؟ داریم

$$V = \{V_۱, V_۲, V_۳, V_۴\}$$

یک مجموعه با $\{V_۱, V_۲, V_۳, V_۴\}$ و یک مسیر با $[V_۱, V_۲, V_۳, V_۴]$ مشخص می‌شود. اگر $A = \{V_۳\}$ باشد داریم:

$$D[V_۲][A] = len([V_۲, V_۳, V_۱]) = \infty$$

که در آن len ، طول مسیر را مشخص می‌کند. اگر $A = \{V_۳, V_۴\}$ باشد داریم:

$$D[V_۲][A] = D[V_۲][V_۳, V_۴] = \min(len([V_۲, V_۳, V_۴, V_۱]), len([V_۲, V_۴, V_۳, V_۱])) = min(۲۰, \infty) = ۲۰$$

با توجه به تعریف فوق بدیهی است که:

$$D[V_i][\emptyset] = C_{i۱}$$

فرمول بازگشتی زیر حل مساله فروشنده دوره گرد را با روش برنامه نویسی پویا فراهم می کند:

$$\begin{cases} D[V_i][A] = \min(C_{ij} + D[V_j][A - V_j]) & A \neq \emptyset \\ D[V_i][\emptyset] = C_{i۱} & A = \emptyset \end{cases}$$

برای زیر مجموعهای از شهرها $S \subseteq \{۱, ۲, \cdots, n\}$ طول کوتاه‌ترین مسیر ملاقات شده در S را با $C_{S۱}$ نمایش می‌دهند. در این روش $C_{S۱} = \infty$ می‌شود، زیرا مسیر نمی‌تواند از شهر ۱ شروع شده و به شهر ۱ ختم شود. حال باید زیر مسئله ها را توسعه داد تا به حل مسئله اصلی رسید. پس از ملاقات شهر j باید شهر i ملاقات شود که این شهر بر اساس تابع مسافت که در رابطه زیرنشان داده شده، محاسبه می‌شود.

$$C_{Sj} = Min_{i \in S, i \neq j} C_{Sj} + d_{ij}$$

در این معادله، C_{Sj} طول مسیر از شهر ۱ تا شهر j بوده و d_{ij} طول مسیر از شهر i به شهر j است. شبه کد الگوریتم برنامه نویسی پویا برای حل مسئله فروشنده دوره گرد در الگوریتم ؟؟ نشان داده شده است.

الگوریتم ۱-۱ الگوریتم برنامه نویسی پویا برای مسئله فروشنده دوره گرد [؟].

```
1.  C_11 = 0
2.  for s=2 to n do
3.      for all subsets S ⊆{1, 2, ···, n} of size S and containing 1 do
4.          C_S1 = ∞
5.      end for
6.      for all j ∈ S − {1} do
7.          C_Sj = min{C_(S−{j})i + d_ij : i ∈ S − {j}}
8.      end for
9.      return minj C_({1,···,n})j + d_j1
10. end for
```

تعداد زیر مسئله ها در این روش برابر $n \times ۲^n$ می‌باشد و هر زیر مسئله با پیچیدگی زمانی $O(n)$ خطی قابل حل است. بنابراین هزینه اجرای این روش برابر $O(n^۲ \times ۲^n)$ می‌شود [؟].

۱-۲-۲ الگوریتم انشعاب و تحدید

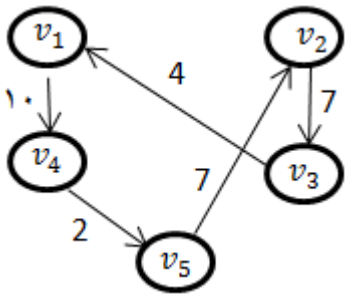
تکنیک $B\&B$ [۱] یک تکنیک برنامهنویسی است که در آن عملکرد تکنیک برگشت به عقب برای برخی مسائل می تواند بهبود یابد. با این مفهوم که با یک تخمین اولیه از هزینه یک جواب، به دنبال یک جواب بهتر هستیم. برای این منظور در هر کجای مسیر از جواب بعدی اگر میزان هزینه، بیشتر از هزینه تخمین اولیه باشد، آن مسیر حذف می‌شود [؟]. روش شاخه و حد اولا ما را به روش خاصی از پیمایش درخت محدود نمی‌کند و ثانیا، تنها برای مسائل بهینه‌سازی به‌کار می‌رود. همانطور که گفته شد، هدف ما در این مسئله یافتن کوتاه‌ترین مسیر در یک گراف جهت‌دار است که از یک راس معین آغاز شده و هر راس گراف را دقیقا یکبار ملاقات کرده و به همان راس آغازین ختم می‌شود. از آنجا که مهم نیست مسیرها از کدام راس آغاز

[۱] Branch and Bound

می‌شوند، ما می‌توانیم اولین راس را به عنوان راس آغازین در نظر بگیریم. ماتریسی که در ادامه آمده گرافی را نمایش می‌دهد که در آن برای هر راس به راس دیگر مسیری با طول مشخص شده وجود دارد. در روش شاخه و حد اولا ما را به روش خاصی از پیمایش درخت محدود نمی‌کند و ثانیا، تنها برای مسائل بهینه سازی بکار می‌رود. در روش شاخه و حد مرتبه زمانی در بدترین حالت تغییر نمی‌کند اما در عمل تعداد عملیات مورد نیاز کمتر خواهد بود. همانطور که گفته شد، هدف ما در این مسئله یافتن کوتاه ترین مسیر در یک گراف جهت دار است که از یک راس معین آغاز شده و هر راس گراف را دقیقا یکبار ملاقات نموده و به همان راس آغازین ختم می‌شود. از آنجا که مهم نیست مسیرها از کدام راس آغاز می شوند، ما می‌توانیم اولین راس را به عنوان راس آغازین در نظر بگیریم. ماتریس زیر گرافی را نمایش می‌دهد که در آن هر راس به راس دیگر مسیری با طول مشخص شده وجود دارد.

$$\begin{bmatrix} ۰ & ۱۴ & ۴ & ۱۰ & ۲۰ \\ ۱۴ & ۰ & ۷ & ۸ & ۷ \\ ۴ & ۵ & ۰ & ۷ & ۱۶ \\ ۱۱ & ۷ & ۹ & ۰ & ۲ \\ ۱۸ & ۷ & ۱۷ & ۴ & ۰ \end{bmatrix}$$

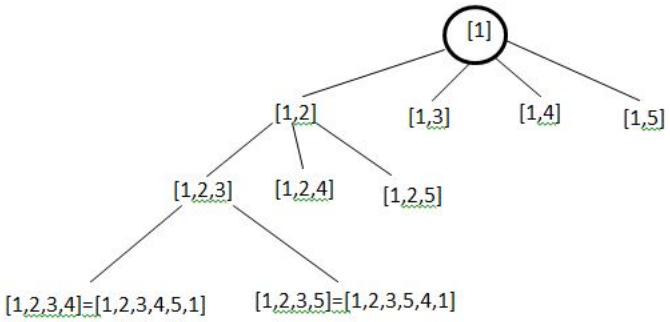
یک تور بهینه برای گراف فوق به شکل ؟؟ است.



شکل ۱-۴: تور بهینه برای ماتریس

یک درخت فضای حالت برای این مساله می‌تواند به این صورت باشد که هر راس آغازین، به عنوان اولین راس در سطح آزمایش می‌شود. هر راس غیر از راس آغازین و راسی که در سطح ۱ انتخاب شده، به عنوان راس دوم در سطح ۲ آزمایش می‌شود و الی آخر. بخشی از این درخت فضای حالت در شکل ؟؟ نمایش داده شده است.

در ادامه لفظ گره، به گره ای در درخت فضای حالت، و لفظ راس به راسی در گراف اطلاق می شود. در هر گره از شکل ؟؟ ، مسیر انتخاب شده تا رسیدن به آن گره را در نظر گرفته‌ایم. برای سادگی کار، هر راس گراف را با اندیس آن مشخص می‌کنیم. گره ای که برگ نباشد نماینده همه تورهایی است که با مسیر ذخیره شده در آن گره شروع شده است. به عنوان مثال گره شامل [۱, ۲, ۳] نشانگر تمام تورهایی است که با مسیر [۱, ۲, ۳] آغاز می شوند. یعنی نشانگر تورهای [۱, ۲, ۳, ۴, ۵, ۱] و [۱, ۲, ۳, ۵, ۴, ۱] است. هر برگ، معرف یک تور است. بنابراین باید برگی را پیدا کنیم که شامل یک تور بهینه باشد. وقتی



شکل ۱-۵: درخت فضای حالت برای نمونه ای از مسئله فروشنده‌دوره‌گرد که در آن پنج راس وجود دارد. اندیس رئوس تور، در داخل گره آورده شده است.

که چهار راس در مسیر ذخیره شده در یک گره وجود داشته باشد، توسعه درخت را متوقف می کنیم زیرا پنجمین راس به خودی خود مشخص می‌شود.

برای آنکه از جستجوی اول بهترین استفاده شود، باید حدی برای هر گره پیدا کنیم. باید یک حد پایین برای طول هر توری که از بسط یک گره به دست می آید، پیداکنیم و گره را در صورتی وعده گاه می نامیم که حد آن کمتر از طول تور مینیمم کنونی باشد. یک حد را می‌توانیم به روش زیر پیدا کنیم.

در هر تور، طول لبه گرفته شده به هنگام گذر از یک راس باید حداقل به بزرگی طول کوتاه ترین لبه ای که در آن راس ناشی می‌شود، باشد. بنابراین یک حد پایین روی هزینه برای خروج از راس $V_۱$ عبارت از مقدار کمینه ی همه ی عناصر غیر صفر در سطح ۱ ماتریس هم جواری، حد پایین هزینه برای خروج از راس $V_۲$ عبارت از مقدار کمینه همه عناصر غیر صفر سطح ۲ از ماتریس هم جواری است و الی آخر. حدود پایین روی هزینه های رئوس گراف شکل ؟؟ عبارت است از:

$$v_۱ \quad minimum(۱۴, ۴, ۱۰, ۲۰) = ۴$$

$$v_۲ \quad minimum(۱۴, ۷, ۸, ۷) = ۷$$

$$v_۳ \quad minimum(۴, ۵, ۷, ۱۶) = ۴$$

$$v_۴ \quad minimum(۱۱, ۷, ۹, ۲) = ۲$$

$$v_۵ \quad minimum(۱۸, ۷, ۱۷, ۴) = ۴$$

چون یک تور باید هر راس را فقط یک بار ترک کند، حد پایینی طول تور، برابر حاصل جمع این مقدار کمینه است. بنابراین، حد پایینی تور عبارت است از: $۴ + ۲ + ۴ + ۷ + ۴ = ۲۱$. این بدان معنا نیست که توری با این طول وجود ندارد. فرض کنید گره حاوی [۱, ۲] در شکل ؟؟ را ملاقات کرده ایم. در آن حالت، قبلا $V_۲$ را به عنوان دومین راس تور مشخص

کرده ایم و هزینه رسیدن به $V_۲$ برابر وزن یال $V_۱$ به $V_۲$ است که برابر ۱۴ می‌باشد. بنابراین، هر توری که از گسترش این گره حاصل شود، با هزینه ی ترک رئوس زیر دارای حدود پایین مشخص شده زیر خواهد بود:

$$\begin{aligned} V_۱ &\qquad ۱۴ \\ V_۲ &\quad minimum(۷, ۸, ۷) = ۷ \\ V_۳ &\quad minimum(۴, ۷, ۱۶) = ۴ \\ V_۴ &\quad minimum(۱۱, ۹, ۲) = ۲ \\ V_۵ &\quad minimum(۱۸, ۱۷, ۴) = ۴ \end{aligned}$$

برای به دست آوردن مقدار کمینه برای $V_۲$ ، یالی را که به $V_۱$ می‌رود، لحاظ نمی کنیم زیرا $V_۲$ نمی‌تواند به $V_۱$ بازگردد. حد پایین طول هر تور که از گسترش دادن گره حاوی [۱, ۲] به دست می آید، برابر با حاصل جمع این مقادیر کمینه است:

$$۱۴ + ۷ + ۴ + ۲ + ۴ = ۳۱$$

برای آن که تکنیک تعیین حد روشن‌تر شود، فرض کنید گره حاوی [۱, ۲, ۳] از شکل ؟؟ را ملاقات کرده ایم. راس دوم را برابر $V_۲$ و راس سوم را $V_۳$ در نظر گرفته ایم. هر توری که از گسترش دادن این گره به دست آمده باشد، دارای حدود پایین زیر با هزینه ترک رئوس است:

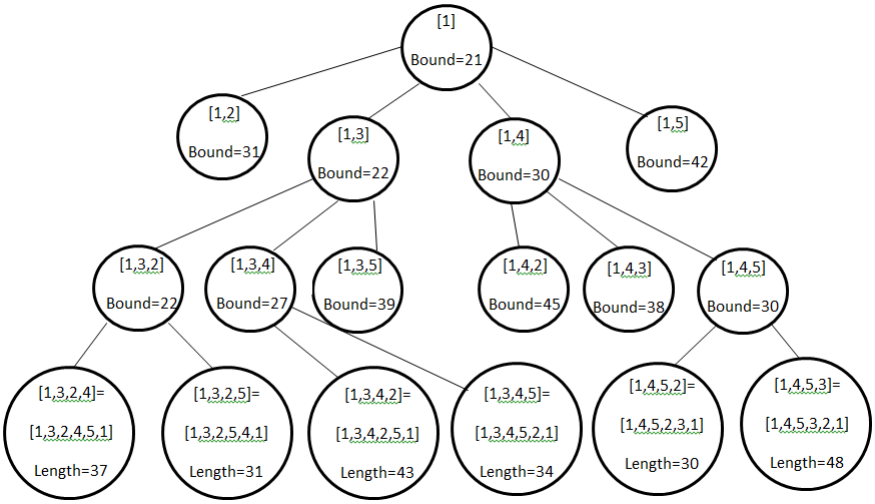
$$\begin{aligned} V_۱ &\qquad ۱۴ \\ V_۲ &\qquad ۷ \\ V_۳ &\quad minimum(۷, ۱۶) = ۷ \\ V_۴ &\quad minimum(۱۱, ۲) = ۲ \\ V_۵ &\quad minimum(۱۸, ۴) = ۴ \end{aligned}$$

برای به دست آوردن مقادیر کمینه $V_۴$ و $V_۵$ یال هایی را که به $V_۲$ و $V_۳$ می‌روند، لحاظ نمی کنیم زیرا قبلا در این رئوس بوده‌ایم. حد پایین طول هر تور را می توان با گسترش دادن گره حاوی [۱, ۲, ۳] به دست آورد:

$$۱۴ + ۷ + ۷ + ۲ + ۴ = ۳۴$$

به همین شیوه می توان حد پایین طول توری را به دست آورد که از توسعه دادن هر گره در درخت فضای حالت به دست می‌آید

و از این حدود پایین در جست وجو بهتر استفاده نمود. با این شیوه بهترین جست وجو با هرس کردن شاخه و حد، درخت شکل ؟؟ را تولید می‌کند. این حد در یک گره غیر برگ نگهداری می‌شود، حال آن که طول تور در یک برگ نگهداری می‌شود. به بهترین حل، مقدار اولیه ∞ (بی نهایت) را می‌دهیم، زیرا در ریشه هیچ حل کاندیدایی وجود ندارد (در درخت فضای حالت، حل های کاندیدا فقط در برگ ها وجود ندارند). درخت تشکیل شده برای حل گراف ؟؟ که در هر گره غیر برگ از درخت فضای حالت، تور ناقص در بالا و حدی که از طول هر تور با گسترش دادن کره به دست می‌آید، در پایین قراردارد به صورت ؟؟ است [؟].



شکل ۱-۶: درخت فضای حالت هرس شده ای که با به کارگیری بهترین جست وجو با هرس کردن شاخه و حد به دست می آید.

۱-۲-۳ الگوریتم تپه نوردی

الگوریتم جستجوی تپه نوردی از ساده ترین الگوریتم های جستجوی محلی است. در مرحله اول الگوریتم، از یک نقطه تصادفی در فضای جستجو به عنوان جواب اولیه شروع کرده و مقدار تابع هدف محاسبه می‌گردد. در مرحله بعد همسایه‌های جواب اولیه برای مقدار تابع هدف بررسی می‌شوند. چنانچه نقطه ای با مقدار تابع هدف کمتری وجود داشته باشد، این نقطه را به عنوان بهترین جواب ذخیره می‌کند و در صورتی که همسایه بهتری وجود نداشته باشد، نقطه فعلی را به عنوان بهترین جواب اعلام می‌نماید.

الگوریتم تپه نوردی، الگوریتمی است که برای یافتن بهترین جواب یک مسئله یا برای پیدا کردن جوابی از مسئله که به اندازه کافی مناسب و بهینه باشد، استفاده می‌شود. این الگوریتم بیشتر برای مسئله هایی مورد بحث قرار می‌گیرند که چندین جواب با ارزش برابر دارند و هدف یافتن یکی از آنهاست. برای بررسی این الگوریتم مسئله زیر را بررسی می‌کنیم:

تابع f به هر یک از اعضای مجموعه متناهی S یک عدد حقیقی نسبت می‌دهد. هدف یافتن عضوی از S است که کمترین (یا بیشترین) مقدار به آن نسبت داده‌شده‌است. همان گونه که گفته شد در اینجا فرض بر این است که مقدار تابع f برای

چندین عضو مختلف S کمینه است و هدف یافتن تنها یکی از آن‌هاست. (در غیر این صورت معمولا الگوریتم تپه نوردی، الگوریتم کار آمدی نیست.) شرح کار الگوریتم به صورت زیر است:

ابتدا یکی از اعضای مجموعه S را (به صورت تصادفی) انتخاب می‌کنیم و آن را A می‌نامیم. سپس در میان اعضایی از S که به A نزدیکند به دنبال جواب مناسب تری برای مسئله می‌گردیم. به عبارتی در میان اعضایی که به A نزدیک اند عضوی بیابیم که مقدار تابع f برای آن کم تر از f(A) باشد سپس این روند را ادامه داده تا به یک کمینه نسبی برای f دست یابیم. بدیهی است که کمینه نسبی به دست آمده لزوما پاسخ خواسته شده نیست. ولی اگر الگوریتم گفته شده چندین بار تکرار شود می‌توان به یافتن جوابی مطلوب امیدوار بود. (در هر بار اجرای الگوریتم اولین عضو به صورت تصادفی انتخاب می‌شود) عیب این الگوریتم این است که در یک تابع که مینیمم و یا ماکزیمم محلی زیادی دارد به راحتی در دام بهینه محلی گرفتار می‌شود و قدرت خروج از آنجا را ندارد. در این الگوریتم درخت جستجو را ذخیره نمی‌کند لذا ساختمان داده گره فعلی فقط باید حالت و مقدار تابع هدف را نگهداری کند. تپه نوردی به همسایه های حالت فعلی نگاه می‌کند. تپه نوردی گاهی جستجوی محلی حریصانه نام دارد زیرا بدون اینکه قبلا فکر کند به کجا برود، حالت همسایه خوبی را انتخاب می‌کند. تپه نوردی معمولا به سرعت به سمت جواب پیش می رود زیرا به راحتی می تواند حالت بد را بهبود ببخشد [؟]. برای حل مسئله فروشنده دوره گرد یک گراف ساده همبند را که یال های آن وزن دار است، در نظر بگیرید. هدف یافتن مسیری همیلتونی است که در آن مجموع وزن یال ها کمینه (یا به اندازه کافی کم) باشد. برای یافتن چنین مسیری می‌توان ابتدا یک مسیر همیلتونی تصادفی انتخاب نمود و سپس در هر گام با ایجاد تغییری اندک در مسیر، مجموع وزن یال های آن مسیر را بهینه نمود. مهمترین موضوع در روش تپه‌نوردی انتخاب تابع هدف می‌باشد که برای فروشنده‌دوره‌گرد تابع هدف، طول مسیر طی شده در نظر گرفته می‌شود. بدیهی است که طولهای کوتاه تر نتیجه ای است که به نتیجه نهایی نزدیکتر است. الگوریتم تپه نوردی به منظور حل مسئله فروشنده دوره گرد در الگوریتم ؟؟ نشان داده‌شده‌است.

الگوریتم ۱-۲ الگوریتم تپه نوردی برای مسئله فروشنده دوره گرد [؟].

```
1. Hill climbing( path: sequence of points)
2. Compute the initial length of path
3. loop
4.     choose the pair of points U, V such that swapping U with V in path has the shortest
       length
5.         if there is no improvment
6.           then return path
7.         end if
8.     swap U and V in the path and decrement the length by change
9. end loop
```

در این الگوریتم به اندازه یک جواب کنونی و جواب همسایه حافظه مصرف می‌شود. همانطور که اشاره شد، اندازه هر جواب برابر تعداد شهرها در مسئله است. بنابراین به اندازه $2n$ که n تعداد شهرهاست، حافظه نیاز دارد [؟].

۱-۲-۴ الگوریتم شبیه سازی تبریدی

واژه انگلیسی Simulated Annealing در لغت به معنای گداخته کردن جسم می‌باشد ولی در اصطلاح، یک فرایند فیزیکی برای بالا بردن دمای جسم تا رسیدن آن به نقطه ذوب و سپس سرد کردن آن طی شرایط مشخص می‌باشد که در طول این فرایند انرژی جسم به حداقل می‌رسد. این الگوریتم از مدل مونت کارلو (که رابطه‌ای‌است میان ساختار اتمی، آنتروپی و دما در طول فرآیند سرمایش یک ماده) الهام گرفته‌شده‌است. در این مدل ابتدا دمای ماده بسیار بالا بوده و کم کم دمای ماده پایین آورده می شود تا به دمای تعادل برسد. سرعت پایین آوردن دمای ماده بسیار مهم است و در صورتی که با سرعت زیادی دما کاهش یابد، ماده به تعادل نرسیده و دارای مشکلاتی خواهدشد. الگوریتم SA [۱] و سایر الگوریتم های ابتکاری، با یک جواب اولیه که به طور ابتکاری ایجاد می‌شود شروع به کار می‌کنند. سپس یک جواب همسایگی که بهبود در تابع هدف ایجاد نماید انتخاب می‌شود و تعداد تکرارهایی که دیگر بهبود در تابع هدف حاصل ایجاد نشود ادامه می‌یابد. معمولا الگوریتم های بهبود دهنده، که با یک جواب اولیه شروع شده و در طی مراحلی بهبود داده می‌شوند، ممکن است بعد از چند تکرار در نقطه بهینه محلی قرار بگیرند، که گاهی اوقات نیز از ناحیه جواب نهایی خیلی دور است. فرق الگوریتم تبرید تدریجی با الگوریتم های بهینه سازی محلی در این است، که در الگوریتم بهینه سازی محلی، یک جواب در همسایگی جواب قبلی ایجاد می‌شود، اگر تابع هدف به واسطه جواب جدید بهتر شود، جواب جدید قبول شده و در غیر این صورت جواب جدید رد می‌گردد. این عمل ممکن است منجر به قرار گرفتن در نقطه بهینه محلی شده و دیگر نتواند از آن خارج شود. در حالی که در روش تبرید تدریجی شبیه سازی شده، از توقف در ناحیه بهینه محلی اجتناب کرده و به طور گذرا از آن رد می‌شود. این حالت با پذیرفتن احتمالی جواب های بد انجام می‌شود، تا از نقطه بهینه محلی خارج شود [؟]. در ادامه از ایده کمینه سازی دما، در بهینه سازی مسئله فروشنده دوره گرد استفاده می‌کنیم. در پیاده سازی این الگوریتم، ابتدا یک جواب را به صورت کاملا تصادفی ایجاد می‌کنیم [؟].

---

[۱] Simulated Annealing

الگوریتم ۱-۳ الگوریتم شبیه سازی تبریدی برای مسئله فروشنده دوره گرد [؟].

1. $t \leftarrow 0$, initialize T
2. select a current point Vc at random and evaluate Vc
3. **Repeat**
4. **Repeat**
5. selectVn from the neighborhood of Vc
6. **if**$cost(Vc) < cost(Vn)$**then**
7. $Vc \leftarrow Vn$
8. **else if** random $[0, 1) < e^{((cost(Vn)-cost(Vc))/T)}$
9. $Vc \leftarrow Vn$
10. **end if**
11. **until** (termination criterion)
12. $T \leftarrow g(T)$
13. **until** (halting-criterion)

در گام بعدی نیاز به همسایه ای از جواب اولیه داریم که از جستجوی محلی استفاده می‌کنیم. با استفاده از احتمال موجود در خط ۸الگوریتم، جواب همسایه به عنوان جواب جاری درنظر گرفته‌می‌شود، حتی اگر تابع هدف را بهبود نبخشد. بدین ترتیب، شانس به دام افتادن در کمینه محلی کاهش می‌یابد. با گذشت زمان، احتمال پذیرش یک جوابی که تابع هدف را بهبود نمی‌دهد کم و کمتر می‌شود. همین روند تا رسیدن به شرط پایان برنامه ادامه می‌یابد.

در این الگوریتم همانند الگوریتم تپه نوردی، به اندازه یک جواب کنونی و جواب همسایه نیاز به حافظه داریم. بنابراین به اندازه $2n$ که n تعداد شهر هاست، حافظه نیاز دارد [؟].

۱-۲-۵ الگوریتم ژنتیک

ایده اصلی الگوریتم ژنتیک، انتقال خصوصیات موروثی توسط ژن ها است. در این الگوریتم، جواب ها با عنوان کروموزوم ها شناخته می‌شوند. فرآیندهایی که در این الگوریتم اتفاق می‌افتد، جهش (تغییر تصادفی کروموزوم ها) و ترکیب دو کروموزوم برای تولید کروموزوم جدید است. الگوریتم ژنتیک برای حل مسئله فروشنده‌دوره‌گرد در ؟؟ بیان شده‌است [؟].

الگوریتم ۱-۴ الگوریتم ژنتیک برای مسئله فروشنده دوره گرد [؟].

1. choose initial population
2. **Repeat**
3. Evaluate the individual cost of the population
4. select pairs of individuals to reproduce
5. Apply crossover operator
6. Apply mutation operator
7. **until** terminating condition

الگوریتم ژنتیک الگوریتمی مبتنی بر جمعیت است و کار خود را بر روی جمعیتی از جواب ها انجام می‌دهد. ابتدا تعداد ۱۰۰ جواب را به صورت تصادفی ایجاد می‌کند و مسیر طی شده در هر جواب را با استفاده از رابطه زیر بدست می‌آورد.

$$Cost = Min \sum_{i=۰}^{i=n-۲} C_{i(i+۱)} + C_{(n-۱)۰}$$

سپس باید عمل انتخاب والدین انجام شود که برای این کار چند روش وجود دارد. انتخاب دو جواب برتر، انتخاب دو جواب به صورت تصادفی از بین پنج جواب برتر و انتخاب دو جواب به صورت تصادفی روش های انتخاب والدین است. پس از انتخاب والدین با یکی از روش های معرفی شده، دو جواب باهم ترکیب می‌شوند. عمل ترکیب در شکل ؟؟ نشان داده شده است.

یک نوع ترکیب، ترکیب PMX [۱] است که با داشتن والدین $P_۱$ و $P_۲$ به تولید فرزندان $O_۱$ و $O_۲$ منجر می‌شود. شرح روش به صورت زیر است:

- بخشی به طور تصادفی از والد اول در فرزند کپی شود.
- با نگاهی به موقعیت های مشابه در والد دوم، هر مقدار را که به فرزند داده نشده انتخاب کنید. برای هر یک از این مقادیر:

  ۱. شاخص این مقدار در والد دوم، مقدار x را از والد اول در همان موقعیت تعیین کنید.

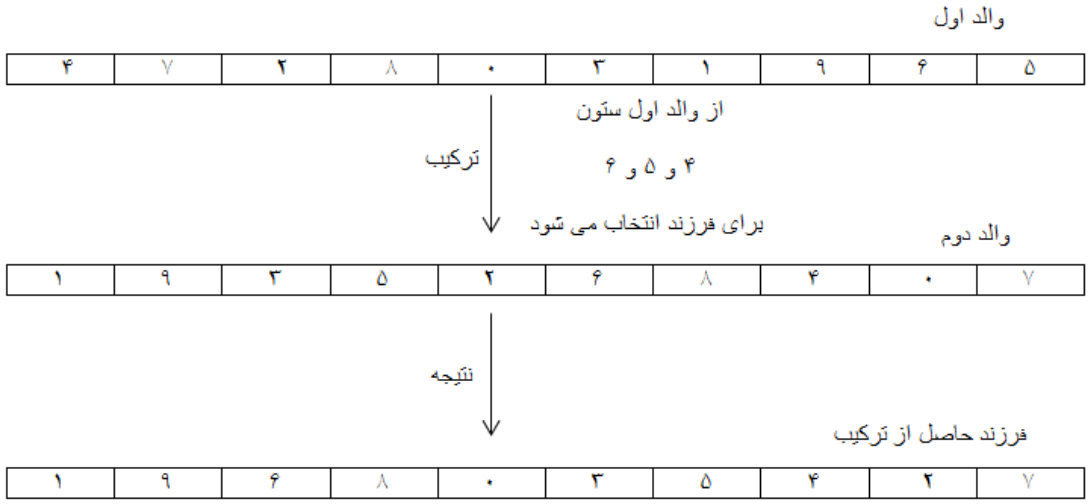
  ۲. همان مقدار را در والد دوم تعیین کنید.

  ۳. اگر شاخص این مقدار در والد دوم بخشی از ردیف اصلی باشد، به مرحله ۱ بر می گردد.

  ۴. اگر شاخص این مقدار در والد دوم بخشی از ردیف اصلی نباشد، مقدار y به فرزند در همان موقعیت وارد می شود.

- هر موقعیت باقی مانده از والد دوم را در فرزند کپی کنید.

[۱] Partially Mapped Crossover



شکل ۱-۷: عمل ترکیب با روش PMX

عمل جهش با یک احتمالی بر روی کروموزوم ایجاد شده صورت می گیرد. عمل جهش به صورت جابجایی دو شهر در یک جواب انجام شده‌است. مراحل الگوریتم تا رسیدن به شرط پایانی ادامه می‌یابند. در هر مرحله بهترین مسیرها مشخص شده و در نهایت پس از اتمام الگوریتم، بهترین مسیر که دارای کمترین هزینه (اندازه مسیر) باشد، انتخاب می‌شود.

در الگوریتم ژنتیک با توجه به اندازه جمعیت نیاز به حافظه داریم. اگر اندازه جمعیت را m و حافظه مصرفی برای هر جواب را n در نظر بگیریم، به اندازه $m \times n$ حافظه مصرفی این الگوریتم است [؟].

در [؟] به بررسی و مقایسه کارایی چهار الگوریتم اکتشافی تپه نوردی، شبیه‌سازی‌تبریدی، ژنتیک و برنامه‌نویسی‌پویا برای حل مسئله فروشنده‌دوره‌گرد پرداخته‌شده‌است. نتایجی که از این بررسی بر روی دو مجموعه داده به‌دست آمده‌است، بدین صورت است که استفاده از روش برنامه نویسی پویا بسیار زمان بر بوده و به منظور رسیدن به جواب بهینه در زمان کوتاه، از الگوریتم‌های اکتشافی استفاده شده‌است، که با این الگوریتم‌ها جواب بهینه به‌دست می‌آید.

۱-۲-۶ الگوریتم بهینه سازی ازدحام ذرات

الگوریتم بهینه سازی ازدحام ذرات PSO [1] یک الگوریتم فراابتکاری است که از حرکت گروهی که به شکل دسته جمعی زندگی می‌کنند مانندسته ای از پرندگان، گروه ماهی‌ها و جوامع انسانی الگو گرفته شده است. این الگوریتم مبتنی بر جمعیت است که از روی رفتار اجتماعی پرندگان مدل شده‌است. در ابتدا این الگوریتم با الهام از لگوهای حاکم بر پرواز همزمان پرندگان و تغییر ناگهانی مسیر آنها و تغییر شکل بهینه‌ی دسته، به کار گرفته‌شده‌است. این الگوریتم در مقایسه با الگوریتم ژنتیک دارای سرعت همگرایی بالایی است. در PSO ذرات در فضای جستجو پراکنده می‌شوند. تغییر مکان ذرات در فضای جستجو تحت تاثیر تجربه و دانش خودشان و همسایگان است.در ابتدا یک جمعیت از جواب (ذرات) ایجاد می‌شود.

[1] Particle Swarm Optimization

این ذرات براساس چند فرمول در فضای جستجو حرکت می‌کنند، حرکت ذرات توسط بهترین موقعیت شناخته‌شده خود در فضای جستجو و همچنین بهترین موقعیت شناخته‌شده نسبت به تمام ذرات است. فرض کنید $f : R^n \rightarrow R$ تابع هزینه باشد که باید کمینه شود. این تابع یک جواب کاندید را به صورت آرگومان در قالب بردار از اعداد حقیقی می‌گیرد و یک عدد حقیقی به عنوان خروجی تولید می‌کند که نشان دهنده ارزش تابع هدف، جواب داده‌شده‌است. هدف پیدا کردن جواب کمینه تابع f است. در PSO ، هر جواب مسئله بهینه سازی به عنوان یک ذره در فضای جستجو درنظر گرفته می‌شود و هر ذره دارای ارزشی است که توسط تابع هدف تعیین می‌شود و همچنین سرعت آن تعیین کننده مقصد و فاصله اش است. تمام ذرات در فضای جواب به‌دنبال بهترین جواب هستند. در ابتدا PSO یک مجموعه جواب تصادفی از ذرات را می‌گیرد سپس جواب های بهینه را با جستجو مکرر پیدا می کند. در هر تکرار، یک ذره بهترین موقعیت خود را پیدا می‌کند. بهترین موقعیتی که خود ذره تاکنون داشته (مولفه شناختی) p_{best} و بهترین موقعیتی که تاکنون توسط کل ذرات به‌وجود آمده‌است (مولفه جمعی) g_{best} نامیده می‌شوند. از نظر مفهومی p_{best} برای هر ذره در واقع حافظه اتوبیولوژیکی آن ذره محسوب می‌شود و تغییر موقعیت ذره بر اساس p_{best} در واقع پاسخ به احساس غربتی است که ذره هنگام دوری از محلی که در آن بیشتر ارضا می‌شوند، دارند. g_{best} همان دانش عمومی جمعیت است وقتی که ذرات موقعیت خود را بر اساس g_{best} تغییر می‌دهند در واقع تلاش می‌کنند که سطح دانش خود را به سطح دانش جمعیت برسانند. تمام ذرات مرحله های زیر را با توجه به بهترن موقعیت نسبت به خود و دیگر ذرات می‌گذرانند. سرعت و موقعیت ذره id به صورت روابط ؟؟ و ؟؟ در هر مرحله به‌روزرسانی می‌شوند.

$$V'_{id} = wV_{id} + \eta_1 r_1(p_{best} - X_{id}) + \eta_2 r_2(g_{best} - X_{id}) \qquad (۶-۱)$$

$$X'_{id} = X_{id} + V'_{id} \qquad (۷-۱)$$

بطور کلی w وزن ایستایی نامیده می‌شود، این یک عامل نسبت است که با سرعت مرتبط است $0 < w < 1$ ، η_1 و η_2 ثابت هستند و به طور معمول عوامل شتاب دهنده نامیده می‌شوند $\eta_1 = \eta_2 = 2$ ، r_1 و r_2 اعداد تصادفی هستند، X_{id} نشان دهنده موقعیت ذره و V_{id} نشان دهنده سرعت ذره id است.

در فرمول (؟؟) بخش اول نشان دهنده سرعت اولیه ذره است، بخش دوم آن نشان دهنده موقعیت ذره نسبت به بهترین موقعیت خودش است و در نهایت بخش سوم آن بخش وابسته (جمعی) نامیده می‌شود، یعنی موقعیت ذره را نسبت بهترین موقعیت تمام ذرات مشخص می‌کند.

۱-۲-۶-۱ الگوریتم PSO برای حل مسئله فروشنده‌دوره گرد

در [؟] از PSO برای حل مسئله فروشنده‌دوره گرد استفاده شده است که از جایگشت برای نشان دادن هر ذره استفاده می‌کند. هر ذره به عنوان یک مسیر محسوب می شود (تمام مسیرها باید به عنوان حلقه یا مسیر بسته تعریف شوند). طول هر مسیر را به عنوان تابع f در نظر می‌گیریم. هر چه طول کوتاه تر باشد مقدار تابع کمتر است. تابع f به‌صورت رابطه (؟؟) است.

$$f(S_i) = \frac{۱}{\sum_{i=۱}^{N} d[C_{n(i)}, C_{n(i+۱)modN}]} \quad (۸-۱)$$

در جدول ؟؟ نتایج مقایسه دو الگوریتم GA و PSO برای ده سری داده مورد آزمایش قرار گرفته شده، نشان داده‌شده‌است. به‌طوریکه ستون اول و دوم به‌ترتیب نمونه‌های مورد آزمایش و جواب بهینه آن‌هاست. [۱] و ستون سوم جواب بهینه همان نمونه

جدول ۱-۱: مقایسه نتایج بهینه الگوریم ژنتیک و بهینه‌سازی ذرات

Test Cases	Optimal in TSPLIB	GA	PSO
$Berlin۵۲$	۷۵۴۲	۷۵۴۲	۷۵۴۲
$kroA۱۰۰$	۲۱۲۸۲	۲۱۳۱۵	۲۱۳۱۰
$kroA۲۰۰$	۲۹۳۶۸	۳۰۱۶۸	۲۹۹۶۸
$Pr۲۹۹$	۴۸۱۹۱	۴۸۵۶۸	۴۸۵۴۰
$Rd۴۰۰$	۱۵۲۸۱	۱۵۱۳۵	۱۵۱۳۵
$Ali۵۳۵$	۲۰۲۳۱۰	۲۴۲۳۱۰	۲۳۱۱۲۰
$D۶۵۷$	۴۸۹۱۲	۵۰۹۱۲	۵۰۶۱۲
$Rat۷۸۳$	۸۸۰۶	۸۹۶۵	۸۹۰۵
$Ul۰۶۱$	۲۲۴۰۹۴	۲۷۹۰۹۴	۲۶۹۹۰۸
$Ul۴۳۴$	۱۵۲۹۷۰	۱۸۲۷۸۰	۱۷۷۸۹۰

ها با الگریتم ژنتیک داده شده و در نهایت ستون آخر جواب بهینه با الگوریتم بهینه سازی ذرات داده‌شده‌است. در [؟] شرایط مسئله ذکر نشده‌است و ظاهرا الگوریتم PSO در دقت و سرعت همگرایی نسبت به الگوریتم GA بهتر است.

۱-۳ الگوریتم کرم شب تاب برای حل مسئله فروشنده دوره‌گرد

الگوریتم کرم شب تاب یک الگوریتم فراابتکاری جدید است که از پدیده طبیعی نور چشمک زن الهام گرفته شده‌است. این الگوریتم باز هم برخی نقطه ضعف ها مانند، سرعت همگرایی کم و افتادن در دام بهینه محلی را دارد. در این الگوریتم برای بهبود جواب از اپراتورهای الگوریتم ژنتیک، تقاطع و جهش در جستجو محلی استفاده می‌شود.

[۱](http://www.iwr.uniheidelberg.de/groups/comopt/software/TSPLIB959

۱-۳-۱ الگوریتم کرم شب تاب

الگوریتم کرم شب تاب (FA) [۲] توسط سین شی یانگ [۳] پیشنهاد شده است [؟][؟]. این الگوریتم متعلق به گروه الگوریتم‌های تصادفی می‌باشد یعنی یک نوع جستجو تصادفی برای رسیدن به مجموعه‌ای از جواب‌ها به کار برده می‌شود. الگوریتم کرم شب‌تاب در پایین‌ترین سطح خود به تولید جواب‌ها درون یک فضای جستجو تمرکز می‌کند و بهترین را برای بقا انتخاب می‌نماید. جستجو تصادفی از گیر افتادن در دام بهینه محلی اجتناب می‌کند. رفتار چشمک زن کرم‌های شب‌تاب بر اساس معیارهای جذابیت، روشنایی و فاصله تغییر می‌کند. بدین صورت که هرچه فاصله افزایش یابد، شدت نور کاهش می‌یابد. این الگوریتم با استفاده از نور چشمک زن الهام گرفته شده است. این روش، از قانون مربع معکوس برای شدت نور استفاده می‌کند که در آن I بامربع فاصله از منبع نور رابطه معکوس دارد. جریان کارکرد FA در مراحل زیر خلاصه می شود:

گام اول (جمعیت اولیه) :

جمعیت اولیه کرم شب تاب به طور تصادفی باتوجه به تمام مقادیر ممکن در محدوده جواب‌ها ایجاد می‌شود.

گام دوم (تابع شایستگی)

محاسبه شدت نور I برای هر یک از کرم‌های شب تاب در جمعیت و ارزیابی آن کرم با توجه به تابع شایستگی [۱] است. شدت نور $I(x_i)$ کرم شب تاب در موقعیت x_i متناسب با مقدار تابع شایستگی است. شدت نور باتوجه به معادله ؟؟ بیان می‌شود:

$$I = I_{\circ} e^{(-\gamma r^{۲})} \qquad (۹-۱)$$

که در آن $I_{\circ}$ منبع شدت نور و γ نشان دهنده ضریب جذب نور، ثابت هستند.

گام سوم (به روز رسانی جمعیت):

کرم شب تابی که جذابیت نور آن بیشتر باشد جایگزین می‌شود. جذابیت متانسب با شدت نور است که در معادله ؟؟ تعریف شده است:

$$\beta = \beta_{\circ} e^{(-\gamma r^{۲})} \qquad (۱۰-۱)$$

که $\beta_{\circ}$ جذابیت در فاصله $r = \circ$ است. حرکت کرم شب تاب i به سمت j در معادله ؟؟ تعریف می‌شود:

$$x_i^{t+۱} = x_i^t + \beta_{\circ} e^{-\gamma r_{ij}^{۲}} (x_j^t - x_i^t) + \alpha \epsilon_i^t \qquad (۱۱-۱)$$

که α یک پارامتر تصادفی، ϵ_i^t یک بردار از اعداد تصادفی در تکرار t و γ ضریب جذب نور است. فاصله اقلیدسی بین دو کرم i و j در موقعیت x_i و x_j توسط r_{ij} مشخص می‌شود.

---

[۱] fitness function [۲] Firefly Algorithm [۳] Xin-She-Yang

گام چهارم (یافتن بهترین کرم شب تاب):

بهترین کرم شب تاب در میان کل جمعیت را در تکرار t تعیین کنید.

گام پنجم (معیار توقف):

گام های دوم، سوم و چهارم را تا رسیدن به جواب مطلوب تکرار کنید.

بهترین کرم شب تاب را در سراسر جمعیت با x^* نشان داده می‌شود.

۱-۴ الگوریتم گسسته کرم شب تاب برای مسئله فروشنده دوره گرد

FA برای حل TSP که در آن هر کرم شب تاب به عنوان جایگزینی از تمام شهرهای تور است، نمایش داده می‌شود. برای نشان دادن حرکت کرم ها، ژو [۱] و همکارانش با حرکت k-opt فاصله بین دو کرم i و j را محاسبه می‌کند [؟]. هر یک از کرم های شب تاب از موقعیت $x_i - ۱$ به موقعیت جدید x_i در تکرار t از معادلات زیر بهدست می‌آیند:

$$x_i^t = ۲ - opt(x_i^t - ۱) \quad (۱۲-۱)$$

$$x_i^t = ۳ - opt(x_i^t - ۱) \quad (۱۳-۱)$$

این حرکت با مقدار شدت نور از کرم شب تاب کنترل می‌شود. شدت نور I با استفاده از فرمول ؟؟ محاسبه می شود :

$$I_i = random(۱, r_{ij}) \quad (۱۴-۱)$$

که I_i شدت نور کرم شب تاب i است که به صورت تصادفی از ۱ تا r_{ij} انتخاب می‌شود. r_{ij} فاصله اقلیدسی بین کرم شب تاب i و بهترین کرم شب تاب در جمعیت j است.

با توجه به مقدار شدت نور، هر یک از کرم های شب تاب حرکت کوتاه یا بلند با استفاده از معادله‌های ؟؟ به‌دست می‌آیند.

۱-۵ الگوریتم ترکیبی گسسته کرم شب تاب برای مسئله فروشنده دوره گرد

در این بخش، یک الگوریتم جدید ترکیبی گسسته FA که HDFA [۲] نامیده شده، معرفی می‌شود. هدف HDFA ، رفع نقطه ضعف های الگوریتم کرم شب تاب ذکر شده با اضافه کردن اجزایی از الگوریتم های دیگر است. HDFA می‌تواند تنوع

[۱] zhou [۲] Hybrid Discrete Firefly Algorithm

جمعیت را حفظ کند و موجب بروز همگرایی زودرس شود. HDFA در هشت گام به صورت زیر خلاصه شده است.

گام اول (پارامترهای ابتدایی)

پارامترهای ابتدایی HDFA برای کنترل عملکرد است. این پارامترها عبارتنداز: شدت نور (روشنایی)، اندازه جمعیت، نرخ تقاطع و جهش

گام دوم (جمعیت اولیه)

ایجاد جمعیت اولیه کرم شب تاب به صورت تصادفی با در نظر گرفتن تمام ارزش ها از طیف وسیعی از مقادیر برای اطمینان از طیف گسترده ای از جواب ها است.

گام سوم (محاسبه شدت نور)

محاسبه شدت نور برای هر یک از کرم های شب تاب در جمعیت ابتدا توسط فاصله اقلیدسی خود از بهترین جواب فعلی، و سپس با انتخاب شدت نور آن I_i به عنوان یک عدد تصادفی در محدوده بین ۱ و فاصله اقلیدسی باتوجه به معادله ؟؟ بدست می‌آید:

$$I_i = rand[۱, r_{ij}] \qquad (۱۵-۱)$$

گام چهارم (تقاطع)

این اپراتور بین بهترین موقعیت کرم شب تاب x^* و موقعیت فعلی x_i عمل می‌کند. آنگاه تناسب موقعیت کرم فعلی x_i با موقعیت دو فرزند تولید شده توسط تقاطع مقایسه شده، تا بهترین موقعیت را به عنوان موقعیت جدید $x_i^{\cdot}$ انتخاب شود.

گام پنجم (جستجو محلی)

برای هر کرم i یک موقعیت جدید $x_i^{\cdot}$ با حرکت دادن آن کرم به سمت موقعیت بهتر j با توجه به شدت نور و جذابیت بین آنها ایجاد می‌شود یعنی I_i کمتر مساوی تعداد شهرها تقسیم بر دو که با $opt - ۲$ اعمال شده، است در غیر این صورت روش $opt - ۳$ اعمال می‌شود. بر طبق آن، تابع شایستگی را برای کرم شب تاب محاسبه کنید.

گام ششم (جهش)

برای افزایش تنوع، اپراتور جهش برای هر نسل در جمعیت با احتمال برابر نرخ جمعیت از پیش تعیین شده، انجام می‌شود. در این فرآیند، اگر یک عدد تصادفی در محدوده [۰, ۱] کمتر از نرخ جهش باشد، الگوریتم موقعیت کرم x_i را تغییر می‌دهد این کار با تبادل محتوا دو بیت که به صورت تصادفی انتخاب شده و بیت های دیگر بدون تغییر می‌مانند، انجام می‌شود و برطبق آن تابع ارزیابی محاسبه می‌شود.

گام هفتم (یافتن بهترین جواب سراسرس)

تعیین بهترین کرم از بین تمام کرم ها در تکرار t

گام هشتم (معیار توقف)

اگر بهترین جواب با حداکثر تعداد تکرارها به‌دست آمده باشد، بهترین کرم شب تاب سراسری را برمی‌گرداند، در غیر

اینصورت گام های سوم تا هفتم را تکرار کنید.

مراحل HDFA در الگوریتم ؟؟ ارائه شده‌است.

الگوریتم ۱-۵ الگوریتم HDFA [؟].

1. Input: objective function $f(x)$;
2. Output: the best solution x^*;
3. t=0;
4. Initialize the population P: $X = x_1, x_2, \cdots, x_n$;
5. **for** $i = 1$ to No-of-fireflies do
6. Calculate the objective function $f(x_i)$ for firefly position x_i of firefly i;
7. Light intensity I_i at x_i is determined by $f(x_i)$;
8. Find the current best firefly x^*
9. **end for**
10. **While** termination criterion not reached **do**
11. **For** $i = 1$ to No-of-fireflies **do**
12. **crossover**(x_i, x^*)
13. **For** $j = 1$ to No-of-fireflies **do**
14. **if**$I_i > I_j$**then**
15. Compute the attractiveness According Eq **??**;
16. Move firefly i from position x_i toward position x_j;
17. According to Eq **??**;
18. **end if**
19. **end for**
20. **if** rand< mutation-rate then
21. Mutate(x_i)
22. **end if**
23. **end for**
24. Evaluate the current firefly position x_i;
25. Find the current best position x^* among all fireflies;
26. t=t+1
27. **end while**
28. Output the best tour x^* from all fireflies in the population
29. **end for**

نتایج بررسی این الگوریتم جدید در [؟] نشان می‌دهد که HDFA می‌تواند برای حل مقادیر بزرگ TSP استفاده شود و درنتیجه باعث حل مسائل پیچیده تر بهینه سازی در دنیای واقعی می‌شود.

۱-۵-۱ عملگر تقاطع

این اپراتور توسط الگوریتم ژنتیک برای ترکیب دو جواب که منجر به یک جواب بهتر است استفاده می‌شود. بنابراین این عملیات برای بهره برداری از جواب های فضای جستجو برای جستجو محلی است. اپراتور تقاطع انواع مختلفی دارد، یکی از این عملیات PMX است.

گام های عملگر تقاطع PMX در الگوریتم ؟؟ آمده‌است.

الگوریتم ۱-۶ الگوریتم PMX برای مسئله فروشنده‌دوره‌گرد [؟] .

1. Select the start crossover point cp1 randomly in the range$[0, num-of-cities-1]$;
2. cp2=$(cp1 + current-lightintensity)\%(number-of-cities-1)$;
3. parent1=x^*;
4. parent2=x_i;
5. Copy the swab,i.e., the region between cp1 and cp2, from parent1 into the first offspring;
6. **for** each index$i$ in the swab **do**
7. **if**parent2$[i] \neq$ offspring[i] **then**
8. add parent2$[i]$to the set Elements;
9. **end if**
10. **end for**
11. **for** each item e in Elements do
12. Find index$i$ of e in parent2;
13. Find the item e' in parent1 in index i;
14. Find e' in parent2 and its index k;
15. **if**index k in parent2 is in the swab then
16. e=e';
17. **go to** 12;
18. **else**
19. offspring$[j]$=e;
20. **end if**
21. **end for**
22. Fill all the remaining positions of the first offspring from parent2.
23. Analogously, **go to** 5 to create second child using parent1=x_i and parent2=x^*

۱-۵-۲ عملگر جهش

اپراتور جهش به‌منظور حفظ تنوع جمعیت و جلوگیری از همگرایی محلی جواب های بعدی استفاده می‌شود.

در جهش، هر جواب در جمعیت ممکن است برای تولید یک جواب جدید با احتمال یک جهش ژنتیکی اصلاح شود. اضافه کردن جهش به FA ممکن است باعث شود موقعیت کرم شب تاب با والدش متفاوت باشد درنتیجه مناطق جدید را در فضای جستجو، جستجو می‌کند. جهش تعاملی (IM) [1] به طور معمول برای مسائل با نمایش های جایگزینی ماند TSP مورد استفاده قرار می‌گیرد. برای کشف جواب های جدید ممکن، IM برای TSP با انتخاب دو موقعیت از دو شهر به طور تصادفی در تور، برای تولید جواب های جدید استفاده می‌شود. گام های اصلی IM در الگوریتم ؟؟ آمده است.

الگوریتم ۱-۷ گام های الگوریتم IM برای مسئله فروشنده دوره گرد [؟] .

1. Step1: For solution X_i, select two cities, m_1 and m_2, in the solution at random;
2. Step2: Swap $X_i(m_1)$ and $X_i(m_2)$;
3. Step3: Return new mutated solution X_i';

۱-۶ مروری بر الگوریتم جستجو ممنوعه

در این اگوریتم یک لیست ممنوعه داریم که برای جلوگیری از دوری شدن استفاده می‌شود که حرکت یا خصوصیاتی از حرکت‌های ممنوعه در آن ثبت می‌شود و همچنین یک لیست کاندیدا که برای کاهش همسایه‌ها است و معیار تنفس جواب ممنوعی که از تمام جواب‌هایی که تا حال یافت شده بهتر است را ذخیره می‌کند.

روال کار الگوریتم جستجو ممنوعه به این صورت است که از یک جواب تصادفی شروع به حرکت می‌کند سپس بهترین جواب همسایه را از میان همسایه‌های جواب فعلی را انتخاب می‌کند در صورتی که این جواب در فهرست ممنوعه نباشد به سمت جواب همسایه حرکت می‌کند در غیراین‌صورت معیار تنفس را چک می‌کند براساس معیار تنفس اگر جواب همسایه از بهترین جواب یافت‌شده تاکنون بهتر باشد الگوریتم به سمت آن حرکت می‌کند حتی اگر جواب در فهرست ممنوعه باشد وسپس فهرست ممنوعه به روز می‌شود.

در فصل بعد مسئله فروشنده‌دوره‌گرد باانتخاب هتل مورد بررسی قرار می‌گیرد.

[1] Interchanging Mutation

فصل ۲

مسئله فروشنده دوره گرد با انتخاب هتل

در این فصل ابتدا صورت کلی مسئله فروشنده‌دوره‌گرد باانتخاب هتل معرفی می‌شود سپس مسئله با الگوریتم مهاجرت پرندگان حل می‌شود و در نهایت کاربرد توریستی مسئله ذکر می‌شود.

مسئله فروشنده‌دوره‌گرد باانتخاب هتل (TSPHS) [۱] یک نوع مسئله فروشنده‌دوره‌گرد است که در آن حداکثر طول سفر برای هر روز محدود است و فروشنده باید یکی از هتل های موجود را در پایان هر روز انتخاب کند. هدف از مسئله فروشنده‌دوره‌گرد با انتخاب هتل کمینه کردن تعداد روزها سفر و طول کل سفر است. مسئله چندین کاربرد عملی دارد مثلا برنامه‌ریزی چند روز تورهای فروشندگان یا مسیریابی وسایل نقلیه الکتریکی که نیاز به پیدا کردن یک ایستگاه برای شارژ وسیله ندارد.

در مسئله فروشنده دوره‌گردباانتخاب هتل سفر مربوط به یک روز کاری است یعنی دنباله‌ای از بازدید شهرها که شروع و پایانش در یک هتل است، درحالی‌که تور مجموعه‌ای از سفرهای متصل است که به‌طور کلی از همه شهرها بازدید می‌شود. هر سفر باید در یکی از هتل های موجود شروع و پایان یابد و نباید بیشتر از طول سفر در نظرگرفته‌شده باشد البته هتل اولیه هر سفر باید هتل نهایی سفر قبلی باشد.علاوه بر این هتل ابتدایی و انتهایی تور یکسان است.

۲-۱ توصیف و فرمول‌بندی مسئله فروشنده‌دوره‌گرد با انتخاب هتل

فرض کنید $H = \{۰, ۱, \cdots, s\}$ و $C = \{s+۱, \cdots, s+n\}$ به‌ترتیب مجموعه‌هایی از $s+۱$ هتل و n شهر باشند که به‌صورت یک گراف کامل $G = (V, A)$ تعریف شده است، بطوری‌که V و A بصورت زیر تعریف شده‌اند:

$$V = H \cup C, \quad A = \{(i,j) | i,j \in V, i \neq j\}$$

[۱] Traveliing Salesperson Problem with Hotel Selection

هر شهر $i \in C$ نیاز به یک زمان سرویس دهی τ_i ($\forall i \in H, \quad \tau_i = ۰$) دارد. C_{ij} مدت زمان مورد نیاز برای سفر از موقعیت i به j است. فرض کنید T مجموعه سفرهای شدنی و $\bar{\kappa}$ زیرمجموعه‌ای از سفرهای باز باشد. برای هر سفر $t \in T$ سه پارامتر $\lambda_t, \beta_{ht}, \alpha_{it}$ تعریف می‌شود که α_{it} مقدار ۱ می‌گیرد اگر در سفر t از شهر $i \in C$ بازدید شود درغیر این صورت مقدار ۰ می‌گیرد و β_{ht} تعداد دفعات استفاده از هتل h در سفر t است از اینرو یکی از مقادیر ۰, ۱, ۲ را می‌گیرد و λ_t طول کل t سفر را نشان می‌دهد ازطرفی w_h یک متغیر عدد صحیح است که نشان دهنده تعداد دفعاتی است که به هتل h می‌رود و روز بعد سفر خود را آغاز می‌کند، است و در نهایت x_t یک متغیر باینری است که در صورتی که سفر t انتخاب شود مقدار یک می‌گیرد در غیر این صورت مقدارش صفر است [؟]. حالت کلی فرمول‌بندی این مسئله بدین صورت است:

$$\min \sum_{t \in T} (M + \lambda_t) x_t \tag{۱-۲}$$

$$s.t \quad \sum_{t \in T} \alpha_{it} x_t = ۱ \quad i \in C \tag{۲-۲}$$

$$\sum_{t \in \bar{\kappa}} \beta_{ht} + x_t = ۲ w_h \quad h \in H \tag{۳-۲}$$

$$\sum_{t \in \Delta(s)} x_t \leq |\Delta(s)| (\sum_{t \in \psi(s)} x_t) \quad s \subseteq H \setminus \{۰\} \tag{۴-۲}$$

$$x_t \in \{۰, ۱\}, w_h \in Z \tag{۵-۲}$$

تابع هدف ؟؟ تعداد سفر و طول کل سفر را کمینه می‌کند که در آن یک عدد بزرگ M به هر یک از سفر های انتخاب شده اضافه می‌شود.

محدودیت ؟؟ نشان می‌دهد که تمام شهرها دقیقا یکبار بازدید می‌شوند.

محدودیت ؟؟ نشان می‌دهد شروع و پایان هر سفر در یک هتل خاص است.

محدودیت ؟؟ از تشکیل زیرتور جلوگیری می‌کند، که در آن $\Delta(s)$ سفرهایی است که شروع و پایان آنها در مجموعه S است اما $\psi(s)$ مجموعه سفرهایی است که یا شروع یا پایان آن در مجموعه S است و $S \subseteq H$ [؟]. حال به توضیح حالت کلی الگوریتم بهینه سازی پرندگان پرداخته می‌شود و در بخش بعدی این الگوریتم برای مسئله فروشنده دوره گرد با انتخاب هتل بررسی می‌شود.

۲-۲ حالت کلی الگوریتم بهینه‌سازی پرندگان مهاجر MBO

الگوریتم بهینه سازی پرندگان مهاجر [۱] یک روش جستجوی محلی است که با درنظر گرفتن یک جواب اولیه مربوط به پرندگانی که به صورت V دور هم قرار می‌گیرند، آغاز می‌شود. با شروع از جواب اولیه و حرکت هر پرنده به سمت دم

[۱] Migrating Birds Optimization

پرنده دیگر جواب نسبت به جواب همسایگی بهبود می‌یابد، بطوری‌که اگر جواب همسایه بهتر باشد جایگزین جواب فعلی می‌شود. زمانیکه تمام جواب‌ها توسط همسایگی‌ها بهبود یافتند این روش چندین بار تکرار می‌شود تا جواب اولیه بهینه شود و جواب بعدی وارد حلقه بعد می‌شود و الگوریتم بعد از تعدادی تکرار متوقف می‌شود . ابتدا فرض کنید n تعداد جواب‌های اولیه پرندگان، k تعداد جواب‌های همسایه، x تعداد جواب‌های همسایه که با جواب‌های بعدی به اشتراک گذاشته می‌شود، m تعداد تورها و K محدودیت تکرار است. الگوریتم ؟؟ روال کلی روش بهینه سازی مهاجرت پرندگان را نشان می‌دهد.

الگوریتم ۲-۱ الگوریتم بهینه‌سازی مهاجرت پرندگان [؟].

1. Generate n initial solutions in a random manner and place them on an hypothetical V formation arbitrarily
2. i=0
3. **while** (i<k) **do**
4. **for** (j=0; j<m; j++) **do**
5. Try to improve the leading solution by generating and evaluating k neighbors of it
6. i=i+K
7. **for** each solution s_r in the flock (except leader) **do**
8. Try improve s_r by evaluating (k−x) neighbors of it and x unused best neighbors from the solution in the front
9. i= i+ (k− x)
10. **end for**
11. **end for**
12. Move the leader solution to the end and forward one of the solutions following it to the leader position
13. **end while**
14. return the best solution in the flock

الگوریتم MBO شباهت زیادی به حرکت دست جمعی پرندگان دارد. شایع ترین سبک پرواز پرندگان به صورت V است. دلیل این سبک پرواز این است که در انرژی صرفه‌جویی می‌شود. پرنده رهبر بیشترین انرژی را صرف می‌کند و دیگر پرندگان با انرژی باد تولید شده توسط حرکت بال‌هایشان حرکت می‌کنند. پرندگان به دلیل طول بال‌های متفاوت به اشکال مختلف در حالت V حرکت می‌کنند به اینصورت که پرنده رهبر در جلو مسیر را انتخاب کرده و دیگر پرندگان در فاصله و زاویه خاصی از پرنده رهبر و با بررسی بهترین جا نسبت به خود و دیگر پرندگان جواب بهینه تولید می‌کنند [؟].

۲-۳ حل مسئله فروشنده دوره گرد با انتخاب هتل با الگوریتم مهاجرت پرندگان

برای حل مسئله فروشنده‌دوره‌گرد با انتخاب هتل از الگوریتم مهاجرت پرندگان (MBO) [۱] که توسط دامن[۲] ارئه شده‌است، استفاده می‌شود [؟]. مدل‌بندی مسئله همان مدل‌بندی ونستن وجن [۳] و همکارانش است [؟] با این تفاوت که محدودیت ؟؟ به صورت زیر تغییر می کند و در این مدل‌بندی x_{ij}^d همان x_{kld} در مدل‌بندی ونستن وجن است. که در آن τ_j زمانی برای سرویس دهی در نظر گرفته شده است.

$$\sum_{i,j\in\mathbf{A}} (C_{ij}+\tau_j)x_{ij}^d - \sum_{i\in\mathbf{A}} C_{ih}x_{ih}^d \leq L \qquad d = ۱,\cdots,d \tag{۶-۲}$$

دلیل این تغییر آن است که در دنیای واقعی زمان رفتن از آخرین شهر به هتل به عنوان زمان کاری در نظر گرفته نمی شود بنابراین باید نادیده گرفته شود.

۱-۳-۲ الگوریتم مهاجرت پرندگان برای TSPHS

الگوریتم MBO برای TSPHS در تکرار اول شبیه به الگوریتم ؟؟ عمل می کند. به این صورت که یک جواب اولیه تولید می کند در ابتدا الگوریتم مهاجرت پرندگان را برای مسئله فروشنده دوره گرد در یک تکرار اجرا می‌کند در ادامه برای هر تور یک هتل انتخاب می‌کند بررسی می‌کند که جواب شدنی باشد اگر شدنی نبود با عملگر جهش جواب را به سمت شدنی حرکت می‌دهد بعد از چند تکرار جواب بهینه بدست می‌آید. این الگوریتم به صورت ؟؟ است:

---

[۱] Migrating birds optimization [۲] Duman [۳] Vensteenwegen

الگوریتم ۲-۲ الگوریتم مهاجرت پرندگان برای مسئله فروشنده دوره گرد با انتخاب هتل [؟].

```
1. Generate initial solutions (P)
2. while stopping criterion not met do do
3.    implement MBO for tsp in one iteration
4.    for  each tour, select its related hotels (ph) do
5.       while  each ph is infeasible do
6.                mutate p and make new ph
7.       end while
8.    end for
9.    replace the best solution in the head
10.   change Ph to p
11. end while
12. select hotel for the tour on the head
```

در [؟] نمونه‌ای از مسئله فروشنده‌دوره‌گرد با انتخاب هتل را با الگوریتم مهاجرت پرندگان حل شده‌است و با جواب های دقیق مقایسه شده‌اند. جدول ؟؟ ستون اول نمونه های مدنظر در [؟] است و ستون دوم جواب دقیق (ES) هزینه ها،

جدول ۲-۱: مقایسه جواب دقیق (ES) و MBO (زمان در ثانیه)

Instance	ES Cost	ES Time	MBO Cost	MBO Tme
$berlin$۱۰	۲۰۰۱۸	۱۰٫۸۶۱۱۵	۲۰۱۳	۳٫۱۵۱
$berlin$۱۲	۳۰۰۲۴	۱۱٫۰۶۹۳	۳۰۰۱۶	۶٫۴۰۶
$berlin$۱۳	۳۰۰۲۶	۱۱٫۴۶۸۲۱	۳۰۰۱۷	۵٫۵۸۹
$berlin$۱۵	۴۰۰۳۴	۱۱٫۵۱۲۲۳	۴۰۰۲۲	۱۴٫۳۱۳
$berlin$۱۹	۵۰۰۵۳	۱۲٫۳۰۸۹۸	۵۰۰۳۴	۵۶٫۸۵۳
$berlin$۲۰	۵۰۰۵۱	۱۲٫۹۵۲۹۹	۵۰۰۳۷	۴۸٫۹۴۳
$berlin$۲۵	۶۰۰۵۹	۱۴٫۱۹۴۶	۶۰۰۳۸	۱۹۱٫۵۸۹
$berlin$۲۶	۷۰۰۶۲	۱۷٫۶۷۲۱۶	۷۰۰۳۷	۱۰۰۰٫۴۷
$berlin$۲۷	۷۰۰۰۶۳	۲۳٫۴۴۱۸۱	۷۰۰۳۶	۱۰۰۱٫۴۱

ستون سوم زمان صرف شده برای جواب های دقیق، ستون چهارم هزینه به دست آمده از الگوریتم مهاجرت پرندگان و ستون پنجم زمان صرف شده برای اجرای الگوریتم مهاجرت پرندگان است.

۲-۴ کاربردی از مسئله فروشنده دوره گرد با انتخاب هتل

در [؟] کاربرد توریستی مسئله فروشنده‌دوره‌گرد بیان شده است به این صورت که به بازدید از ۱۹ مقصد توریستی در لنکاوی پرداخته‌می‌شود. از آنجایی که به دنبال یافتن بهترین مسیر به منظور به حداقل رساندن هزینه های سفر بودند یک انتخاب

خوب مسیر برای توریست ها اهمیت ویژه ای در مقدار هزینه سفر و زمان سفر دارد. هزینه های موجود در سفر عبارت اند از: هزینه کرایه وسیله نقلیه، بنزین واسکان برای استراحت. شکل کلی این جزیره به صورت شکل ؟؟ است:

شکل ۲-۱: جزیره کداه لنکاوی

آنها یک مدل ریاضی مشابه با مساله فروشنده دوره گرد (TSP) برای مسئله را در نظر گرفتند. مسئله را بدین صورت فرض کردند که ۱۹ مقصد موجود است که مقصد اول و آخر آن یکسان و جزیره کداه لنکاوی است، زمان رفتن به هر مقصد را بر حسب دقیقه می‌دانستند برای این بازدید چهار روز مدنظرشان بود که به جز روز آخر که به جزیره برمی‌گشتند سه شب دیگر برای استراحت نیاز به اسکان داشتند. فرمول بندی آن بدین صورت است:

$$\min\Big(\sum_{i=۱}^{n}\sum_{j=۱}^{n} d_{ij}x_{ij}\Big)/۶۰۰ \qquad (۷-۲)$$

$$s.t \quad \sum_{i=۱}^{n} x_{ij} = ۱; \quad j = ۱,۲,\cdots,n, \quad i \neq j \qquad (۸-۲)$$

$$\sum_{j=۱}^{n} x_{ij} = ۱; \quad i = ۱,۲,\cdots,n, \quad j \neq i \qquad (۹-۲)$$

$$u_i - u_j + nx_{ij} \leq n-۱; \quad i \neq j; i = ۲,۳,\cdots,n; \quad j = ۲,۳,\cdots,n \qquad (۱۰-۲)$$

$$x_{ij} \in \{۰,۱\}; \quad \forall i,j = ۱,۲,\cdots,n \qquad (۱۱-۲)$$

با

$$x_{ij} = \begin{cases} ۱ & \textbf{در صورتی که یک مسیر از مقصد } i \textbf{ به } j \textbf{ موجود باشد} \\ ۰ & \textbf{در غیر این صورت} \end{cases}$$

که متغیر های بالا به صورت زیر تعریف می‌شوند:

$d_{ij} =$ سفر و زمان سفر از مقصد توریستی i به مقصد توریستی j

$u_i =$ شماره دنباله مقصد توریستی i در این سفر

$u_j =$ شماره دنباله مقصد توریستی j در این سفر

معادله (؟؟) کل سفر و زمان بازدید از سفر در یک روز را کمینه می‌کند. یعنی پس از ۱۰ ساعت (۶۰۰ دقیقه) گردشگران هتلی برای استراحت انتخاب می‌کنند.

محدودیت (؟؟) نشان می دهد که گردشگران یک بار به هر مقصد توریستی وارد می‌شوند.

محدودیت (؟؟) نشان می دهد که گردشگران هر مقصد توریستی را یک بار ترک می‌کنند.

و در نهایت محدودیت (؟؟) برای جلوگیری از تشکیل زیر تور است.

در صورتی که گردشگران با استفاده از مسیر منظم و بدون استفاده از مدل TSP و با بکارگیری ازتجارب و اطلاعات فردی مسیر خود را انتخاب کنند کل هزینه سفر RM۱۱۹۱٫۱۰ می‌شد. اما با استفاده از مدل TSP هزینه گردشگران برای بازدید از تمام مقصد های توریستی در چهار روز با هزینه‌های کلی سفر که شامل هزینه کرایه وسیله نقلیه، بنزین و محل اقامت می‌شود به مقدار RM۸۵۶٫۵۵ شد. بنابراین مدل TSP برای کمک به گردشگران در انتخاب بهترین مسیر که کل هزینه سفر را به حداقل برساند مناسب است. اما در انتخاب اسکان برای هرروز به خوبی عمل نمی‌کند و برای آن باید محدودیت‌هایی تعریف کرد که در انتهای هرروز یک اسکان نزدیک به آخرین مکان بازدید شده یافت و روز بعد ادامه مکان‌ها را بازدید کرد. برای حل این مساله می توان از مدل مساله فروشنده دوره گرد با انتخاب هتل که در فصل بعد ذکر شده، استفاده کرد [؟].

فصل ۳

الگوریتم ممتیک برای مسئله فروشنده دوره گرد با انتخاب هتل

این فصل دربرگیرنده توضیح روش حل مسئله فروشنده‌دوره‌گرد با انتخاب هتل است [؟] . در بخش اول به توصیف مسئله و مدل‌سازی ریاضی آن پرداخته شده‌است. بخش دوم الگوریتم ممتیک برای مسئله توضیح داده‌شده‌است بخش سوم عملگرهای الگوریتم ممتیک که خود شامل زیربخش هایی از جمله روش ساخت، تولید جمعیت اولیه، انتخاب و تقاطع، تنوع و جهش، بهبود فرزندان و بروز رسانی جمعیت و چک نسل ها است. در بخش چهارم جستجوی ممنوعه برای مسئله فروشنده‌دوره‌گرد با انتخاب هتل ارائه شده‌است.

۳-۱ توصیف مسئله

این مساله شامل یک مجموعه ناتهی H از هتل‌ها و یک مجموعه C از شهرها است که به صورت یک گراف کامل $G = (V, A)$ تعریف شده است، به‌طوری‌که V و A بهصورت زیر تعریف شده‌اند:

$$V = H \cup C, \quad A = \{(i,j) | i,j \in V, i \neq j\}$$

هر شهر $i \in C$ نیاز به یک زمان سرویس‌دهی τ_i (با $\forall i \in H, \quad \tau_i = 0$) دارد. C_{ij} مدت زمان مورد نیاز برای سفر از موقعیت i به j است. هدف به حداقل رساندن زمان رفت و برگشت برای بازدید تمام شهرها وهمچنین به حداقل رساندن کل زمان تور است. هتل ابتدایی و انتهایی از تور یکسان در نظر گرفته‌می‌شوند ($i = 0$)، همچنین از این هتل می‌توان در طول تور نیز استفاده کرد. منظور از 'سفر' تور یک روز است که دارای ویژگی‌های زیر است:

- در هر سفر شروع و پایان باید در یکی از هتل های موجود باشد.
- سفر نباید از زمان در نظر گرفته شده بیشتر باشد.
- هر سفر باید در هتلی که سفر قبلی در آن پایان گرفته شده شروع شود.

از آنجایی که هیچ محدودیتی در انتخاب هتل وجود ندارد، یک جواب برای TSPHS ، [۱] لزوما یک دور نیست.

قبل از بیان مدل برنامه‌ریزی خطی کاستور به بیان مدل برنامه‌ریزی ونستن وجن و همکارینش می‌پردازیم [؟]. که در مدل ریاضی ونستن وجن یک مجموعه با $s+۱$ هتل $(i = ۰, \cdots, s)$ و n شهر $(s+۱, \cdots, s+n)$ داریم. تعداد سفرها از فرمول زیر بدست می‌آید:

$$m = \sum_{i=s+۱}^{s+n} \frac{T_i}{C} \qquad (۱-۳)$$

که در آن T_i زمان اختصاص داده‌شده برای بازدید از هر شهر و C محدودیت زمانی در نظر گرفته‌شده برای هر سفر $d = ۱, \cdots, m$ است. که T_i متناظر با τ_i و C متناظر با L که حد بالا برای سفر در مدل برنامه‌ریزی خطی کاستور است. مدل

[۱] Travelling salesperson problem with hotel selection

برنامه‌ریزی خطی صفرو یک مسئله به صورت زیر است:

$$\min \sum_{d=۱}^{m} \sum_{k=۰}^{s+n} \sum_{l=۰}^{s+n} x_{kld} c_{kl} \quad (۲-۳)$$

$$s.t \quad \sum_{d=۱}^{m} \sum_{k=۰}^{s+n} x_{kid} = ۱, i = s+۱, \cdots, s+n \quad (۳-۳)$$

$$\sum_{k=۰}^{s+n} x_{kid} = \sum_{l=۰}^{s+n} x_{ild}, d = ۱, \cdots, m; i = s+۱, \cdots, s+n \quad (۴-۳)$$

$$\sum_{h=۰}^{s} \sum_{k=۰}^{s+n} x_{khd} = \sum_{h=۰}^{s} \sum_{l=۰}^{s+n} x_{hld} = ۱, d = ۱, \cdots, m \quad (۵-۳)$$

$$\sum_{k=۰}^{s+n} \sum_{l=۰}^{s+n} (x_{kld}(c_{kl} + T_l)) \leq C, d = ۱, \cdots, m \quad (۶-۳)$$

$$\sum_{l=۰}^{s+n} x_{۰l۱} = \sum_{k=۰}^{s+n} x_{k۰m} = ۱ \quad (۷-۳)$$

$$\sum_{k=۰}^{s+n} x_{khd} = \sum_{l=۰}^{s+n} x_{hld+۱}, d = ۱, \cdots, m-۱; h = ۰, \cdots, s \quad (۸-۳)$$

$$u_i - u_j + ۱ \leq (n-۱)(۱ - \sum_{d=۱}^{m} x_{ijd})$$

$$, i = s+۱, \cdots s+n; j = s+۱, \cdots s+n \quad (۹-۳)$$

$$x_{ijd} \in \{۰, ۱\}, d = ۱, \cdots, m \quad i = ۰; \cdots, s+n \quad j = ۰, \cdots, s+n \quad (۱۰-۳)$$

$$u_i \in ۱, \cdots, n \quad i = s+۱, \cdots, s+n \quad (۱۱-۳)$$

که در آن x_{kld} برابر یک است اگر در سفر d از یک شهر یا هتل k یا از یک شهر یا هتل l ملاقات شود، در غیراین‌صورت برابر صفر است. x_{kld} همان x_{ij}^{d} در مدل‌بندی کاستور است که بعدا به معرفی آن می‌پردازیم. u_{id} نشان‌دهنده موقعیت شهر i در سفر d است. مدل سازی بالا به شرح زیر است:

تابع هدف (؟؟) تعداد سفر و کل مسافت را کمینه می‌کند.

محدودیت (؟؟) نشان می‌دهد از هر شهر فقط یکبار بازدید می‌شود.

محدودیت (؟؟) رفت وبرگشت از هر سفر را تضمین می‌کند.

محدودیت (؟؟) نشان می‌دهد که شروع هر سفر و پایانش در یکی از هتل های موجود است.

محدودیت (؟؟) یک حد بالا برای طول سفر مشخص می‌کند.

محدودیت (؟؟) نشان می‌دهد که شروع و پایان تور در هتل صفر است.

محدودیت (؟؟) نشان می‌دهد که اگر یک سفر در یک هتل مشخص به پایان برسد، سفر بعدی از همان هتل شروع می‌شود.

محدودیت (؟؟) از تشکیل زیرتور جلوگیری می‌کند. این محدودیت با توجه به فرمولبندی میلر-تاکر-زملین، [1] است.

مدل‌سازی که در ادامه خواهدآمد همان مدل‌سازی بالا با این تفاوت است که برای هر یک از محدودیت‌های (؟؟)، (؟؟) و (؟؟) دو محدودیت تعریف کرده و یک محدودیت دیگر برای نشان دادن اینکه سفرها در روزهای متوالی شکل می‌گیرد، دارد.

فرض کنید D حداکثر تعداد سفر های موجود در جواب باشد. برای به حداقل رساندن تعداد سفرها، یک ضریب M به طور کافی بزرگ، در تابع هدف ضرب می‌شود.

فرض کنید x_{ij}^d یک متغیر دودیی با مقدار یک باشد، اگر در سفر d یک شهر یا هتل i یا یک شهر یا هتل j ملاقات شود، در غیر اینصورت مقدار آن صفر است. همچنین فرض کنید متغیر دودویی y^d ، مقدار یک داشته‌باشد، اگر در سفر d حداقل یک شهر یا هتل ملاقات شده‌باشد، در غیر اینصورت مقدار آن صفر می‌شود، اگر هیچ سفری در روز d به منظور ملاقات از تمام شهرها نیاز نباشد. در تابع هدف فرمول‌بندی IP موجود به ترتیب متغیر های y^d و x_{ij}^d برای عبارات ریاضی استفاده

[1] Miller-Tucker-Zemlin

می‌کند و این مسئله به صورت زیر مدل‌بندی می‌شود:

$$\min M \sum_{d=1}^{D} y_d + \sum_{d=1}^{D} (\sum_{(i,j)\in \mathbf{A}} C_{ij} x_{ij}^d) \quad (۱۲-۳)$$

$$s.t \quad \sum_{d=1}^{D} \sum_{i\in \mathbf{V}} x_{ij}^d = 1, j \in \mathbf{C} \quad (۱۳-۳)$$

$$\sum_{i\in \mathbf{V}} x_{ij}^d = \sum_{i\in \mathbf{V}} x_{ji}^d, j \in \mathbf{C}, d = 1, \cdots, D \quad (۱۴-۳)$$

$$\sum_{h\in \mathbf{H}} \sum_{j\in \mathbf{V}\setminus\{h\}} x_{hj}^d = y^d, d = 1, \cdots, D \quad (۱۵-۳)$$

$$\sum_{h\in \mathbf{H}} \sum_{j\in \mathbf{V}\setminus\{h\}} x_{ih}^d = y^d, d = 1, \cdots, D \quad (۱۶-۳)$$

$$\sum_{(i,j)\in \mathbf{A}} (C_{ij} + \tau_j) x_{ij}^d \le L, d = 1, \cdots, D \quad (۱۷-۳)$$

$$\sum_{V\setminus\{0\}} x_{0j}^1 = 1 \quad (۱۸-۳)$$

$$\sum_{i\in \mathbf{V}\setminus\{0\}} x_{i0}^d \ge y^d - y^{d+1}, d = 1, \cdots, D \quad (۱۹-۳)$$

$$\sum_{i\in \mathbf{V}} x_{ih}^d + y^d \ge \sum_{i\in \mathbf{V}} x_{hi}^{d+1} + y^{d+1}. h \in \mathbf{H}, d = 1, \cdots, D \quad (۲۰-۳)$$

$$\sum_{i\in \mathbf{V}} x_{ih}^d - \sum_{i\in \mathbf{V}} x_{hi}^{d+1} \le 1 - y^{d+1}, h \in \mathbf{H}, d = 1, \cdots, D-1 \quad (۲۱-۳)$$

$$x_{ij}^d \le y^d. (i,j) \in \mathbf{A}, d = 1, \cdots, D \quad (۲۲-۳)$$

$$y_d \ge y^{d+1}, d = 1 \cdots, D-1 \quad (۲۳-۳)$$

$$\sum_{i\in \kappa} \sum_{j\in \kappa\setminus\{i\}} x_{ij}^d \le |\kappa| - 1, \kappa \subset C, 2 \ge |\kappa| \le |C| - 1, d = 1 \cdots, D \quad (۲۴-۳)$$

$$x_{ij}^d \in 0, 1, (i,j) \in \mathbf{A}, d = 1, \cdots, D \quad (۲۵-۳)$$

$$y^d \in \{0, 1\}, d = 1, \cdots, D \quad (۲۶-۳)$$

مدل‌سازی بالا به شرح زیر است:

تابع هدف (؟؟) تعداد سفر وکل مسافت را به حداقل می‌رساند.

محدودیت (؟؟) نشان می‌دهد که هر شهر یک بار بازدید دارد.

محدودیت (؟؟) رفت و برگشت از هر سفر را تضمین می‌کند.

محدودیت (؟؟) و (؟؟) نشان می‌دهد که شروع هر سفر و پایانش در یکی از هتلهای موجود است.

محدودیت (؟؟) یک حد بالا برای طول سفر مشخص می‌کند.
محدودیت (؟؟) و (؟؟) نشان می‌دهد که شروع و پایان تور در هتل صفر است.
محدودیت (؟؟) و (؟؟) نشان می‌دهد اگر سفری در هتلی پایان یابد سفر بعدی در همان هتل شروع می‌شود.
محدودیت (؟؟) یک سفر انتخاب می‌شود اگر وتنها اگر مراجعه به شهر یا هتل در طول یک روز وجود داشته‌باشد. محدودیت (؟؟) نشان می‌دهد که سفر ها در روزهای متوالی شکل می‌گیرد.
محدودیت (؟؟) یک زیرتور کلاسیک شامل زیرمجموعه‌های κ از مجموعه شهرهای C ، با حذف محدودیت‌های به کار برده شده در هر سفر بدست می‌آید. در نظر داشته باشید که چون یک جواب شدنی از TSPHS ممکن است شامل دوره شروع و پایان در همان هتل باشد، زیر مجموعه‌های κ با حذف زیر تور در محدودیت (؟؟) فقط شامل شهرها است.
در نهایت محدودیت (؟؟) و (؟؟) ثابت های دودویی برای متغیرهای x_{ij}^d و y^d است. حال این مساله را با یک روش فوق ابتکاری بررسی می‌کند، این بحث در دو سطح تصمیم گیری انتخاب هتل و مسیریابی عمل می‌کند که تصمیم انتخاب هتل با استفاده از الگوریتم ممتیک و مسیر یابی یا یک روش جستجوی ممنوعه ساخته می‌شود.

۳-۲ الگوریتم ممتیک برای TSPHS

در این بخش، روش فراابتکاری برای حل مسئله فروشنده‌دوره‌گرد باانتخاب هتل بیان می‌شود. این روش، دو هدف مختلف انتخاب هتل و مسیربابی شهرها را مورد بررسی قرار می‌دهد. همانطور که در بخش ؟؟ ، گفته شد یک تور شروع و پایانش در هتل از قبل تعریف شده،است. انتخاب هتل در انتهای هر سفر بسیار مهم است، زیرا باید با یک انتخاب درست زمان سفر را تا جای ممکن به حداقل رساند.هتل ها به صورت یک دنباله $(H_s =< h_r >, h_r \in H)$ است، در این دنباله در هر تکرار ممکن است بیش از یکبار از هر هتل استفاده شود. به نظر می‌رسد که انتخاب هتل تاثیر زیادی در کیفیت جواب نهایی دارد [؟] [؟]. در اینجا، انتخاب هتل با استفاده از الگوریتم ممتیک (MA) ، [۱] [؟] و مسیریابی واقعی ازشهرها با یک روش جستجوی ممنوعه TS ، [۲] در MA استفاده می‌شود [؟] [؟].

هرگاه یک الگوریتم مبتنی بر جمعیت همانند الگوریتم ژنتیک با یک الگوریتم جستجو محلی ترکیب شود، به الگوریتم حاصل، الگوریتم ممتیک گویند. الگوریتم ممتیک، یک الگوریتم تکاملی است که در آن جواب بااستفاده از یک یا چند عملگر جستجوی محلی بهبود می‌یابد. این الگوریتم به طور معمول کروموزوم های ادغام شده را که جمعیت نامیده می‌شود و با P نمایش داده‌می‌شود، نگه می دارند یعنی جواب‌ها را نمایش می‌دهند. در الگوریتم ممتیک یک کروموزوم به صورت دنباله‌ای از هتل‌ها است که در آن جواب، بازدید از هتل‌ها است. کیفیت کروموزوم به طور مستقیم متناظر با کیفیت تور TSPHS تولید شده توسط جستوجوی ممنوعه است.

الگوریتم ؟؟ ، جزئیات الگوریتم ممتیک با استفاده از شبه کد است، که نیاز به دو پارامتر مشخص دارد.

۱. G_{max} : حداکثر تعداد نسل که تعریف معیار توقف الگوریتم است.

---

[۱] Memetic Algorithm [۲] Tabu Search

۲. P_s : اندازه جمعیت.

الگوریتم ۳-۱ الگوریتم ممتیک برای مسئله فروشنده دوره گرد با انتخاب هتل.

```
1. initialisation
2. genration of initital population (P)
3. while stopping criterion not met do
4.         select: p1 and p2 from P
5.     crossover:    p1 ⊗ ps → o1, o2
6.     for each offspring o do
7.         while o is not diverse enough do
8.             mutate o
9.         end while
10.        improve with tabu search
11.        replace according to established criteria
12.    end for
13. end while
```

۳-۲-۱ عملگرهای الگوریتم ممتیک

در این بخش، دو روش ساخت برای تولید تورهای TSPHS ارائه شده‌است. روش اول به ساخت یک تور از دنباله ای از شهرها و روش دوم به ساخت یک تور از دنباله ای از هتل ها می‌پردازد. هر دو روش توسط MA در مراحل مختلف الگوریتم مورد استفاده قرار می‌گیرد.

روش ساخت C1 : یک دنباله از شهرها داریم، بدون درنظر گرفتن محدودیت زمانی با بازدید شهرها پرداخته‌می‌شود. روش ساخت اول، به تولید یک تور شدنی TSPHS که نسبت به تور TSP بهتر است و شروع و پایان تور در هتل از پیش تعریف شده و تمام شهرها ملاقات می‌شوند، در حالیکه محدودیت زمانی نادیده گرفته می‌شود، می‌پردازد. این تور، به صورت $\theta =< s_0, s_1, \cdots, s_n >$ تعریف می‌شود (که s_0 هتل ابتدایی و انتهایی و $s_1, \cdots, s_n$ شهرها هستند) که بطور کلی برای TSPHS شدنی است.

روش ساخت C1 بهترین دنباله از هتل‌های بهینه تور TSP افراز شده به سفر های ممکن را تعیین می‌کند. روش ساخت افراز بندی شده گراف حاصل از ساخت روش اول، حاوی حداکثر $mn + 1$ گره است که $m = |H|$ و $n = |C|$. اولین گره در گراف تولید شده، هتل شروع با اندیس صفر است. تمام گره های دیگر مرتبط با ترکیبی از شهرها و هتل هاست. گره i^p را متناظر با توقف در هتل p برای ملاقات i امین شهر در تور s_i تعریف می‌کنیم. گره نهایی در گراف معین، n° است، برای ورود به آخرین هتل (هتل صفر)، بعد از ملاقات n امین شهر در تور s_n است. اگر یک سفر از هتل p برای بازدید از شهرهای تور s_{i+1} به s_j پس از رسیدن به هتل q امکان پذیر باشد یال (i^p, j^q) به گراف اضافه می‌شود. طول یال (i^p, j^q) مجموعه

زمان مورد نیاز برای سفر از i^p به j^q از طریق شهرها تور s_{i+1} به s_j و زمان بازدید است، که به‌صورت زیر است.

$$C_{ps_{i+1}} + \sum_{v=i+1}^{j-1} (C_{s_v s_{v+1}} + \tau_{s_v}) + \tau_{s_j} + C_{s_j q}.$$

پس از آن روش افراز شده یک مسیر از صفر به n_0 را با استفاده از یال (i^p, j^q) تعیین می‌کند. روش افرازبندی، از الگوریتم دیجسترا به صورت متوالی با به حداقل رساندن تعداد یال ها و طول کل سفر استفاده می‌کند.

روش ساخت C2 : در این روش دنباله‌ای از هتل‌ها را داریم، با توجه به محدودیت زمانی شهرها را در مسیر بین هتل‌ها جا داده و در هر سفر تعدادی شهر را بازدید می‌کنیم.

ساخت روش دوم، یک تور TSPHS از یک دنباله داده‌شده از هتل‌های H_s ایجاد می‌کند. تعداد سفر ها، جواب TSPHS است که با k تعریف می‌شود، بطوریکه $k = |H_s| + 1$. باتوجه به کافی نبودن تعداد هتل‌ها ممکن است که یک دنباله از هتل‌ها باتوجه به نتایج جواب TSPHS نشدنی باشد. در این صورت یک جواب نشدنی بدست آمده‌است، که با روش جستجوی ممنوعه بهبود می‌یابد.

روش C2 برای اولین بار یک مجموعه ای از K سفرهای خالی را ایجاد می‌کند که توسط H_s اعمال می‌شود. شهرها با استفاده از اکتشافات متوالی کریستوفیدس، [1] و همکارانش به تور اضافه می‌شود [۸].

این روش با قرار دادن تک تک شهرها در اولین سفر، تازمانیکه محدودیت زمانی برقرار است انجام می‌شود، به همین ترتیب با سفر دوم ادامه می‌دهد. درنهایت، هر سفر بااستفاده از عملگر opt -3 بهبود پیدا می‌کند [؟] . این عملگر سه یال از جواب فعلی را حذف می‌کند و از بین هشت روش ممکن، بهترین سفر را ارائه می‌دهد. در ادامه تولید جمعیت اولیه و همین طور تقاطع و جهش و بهبود و به روز رسانی جمعیت اولیه را شرح می‌دهیم.

۳-۲-۱-۱ تولید جمعیت اولیه

اولین مرحله از الگوریتم ممتیک، تولید کروموزوم P_s برای جمعیت اولیه است. برای هر عضو j از جمعیت، نمایش دنباله برداری از هتل‌ها H_s^j، یک تور S^j وابسته وجود دارد.

اولین عضو جمعیت H_s^1 با استفاده از روش ساخت C1 که جواب شدنی برای TSPHS را تضمین می‌کند، تولید می‌شود.جواب، اغلب یک جواب خوب است، بنابراین حد بالای خوبی را فراهم می‌کند، یعنی $|H_s^1| + 1$ حد بالایی برای سفرهایی که جواب بهینه دارند، است. برای بعضی موارد، روش ساخت C1 یک جواب شامل تنها یک سفر تولید می‌کنند، بطوریکه جواب بهینه TSPHS بطور اتفاقی با یک تور ساده TSP برابر است، الگوریتم ممتیک بلافاصله پس از جواب بدست آمده به پایان می‌رسد. بدون بهینه سازی بیشتر، تور TSP از روش ساخت C1 بدست آمده‌است، در صورتیکه اولین عضو از جمعیت، H_s^1 شامل بیش از یک سفر است، $P_s - 1$ کروموزوم شامل $|H_s^1|$ هتل‌ها، با جمعیت اضافی است. با توجه به این واقعیت که روش ساخت C1 قطعی است، نمی توان آن را برای تولید جواب های دیگر H_s^1 مورد استفاده قرارداد.

[1] Christofides

بنابراین جمعیت اولیه با استفاده از روش ساخت C2 بدست می‌آید. که در این روش ۱ $-$ P_s دنباله تصادفی از هتل $|H_s^1|$ است، نتایج در ۱ $-$ P_s جواب TSPHS مختلف، استفاده می‌شوند. در نهایت هر جواب در جمعیت با استفاده از روش جستجوی ممنوعه بهبود یافته‌اند.

۳-۲-۱-۲ انتخاب و تقاطع

پس از تشکیل جمعیت اولیه، نسل جدید توسط جفت گیری اعضای جمعیت تولید می‌شوند. اعضای جمعیت جفت گیری، والدین نام دارند. در الگوریتم ممتیک، انتخاب والدین با استفاده از روش باینری انجام می‌شود. به طور تصادفی دو عضو از جمعیت، h_1 و h_2 به عنوان اولین والد P_1 در نظر گرفته می‌شوند. سپس دو عضو دیگر به طور مشابه به عنوان دومین والد P_2 انتخاب می‌شوند. عملگر تقاطع تک نقطه‌ای است بطوریکه دنباله هتل های اعمال شده به والدین، به تولید فرزندان جدید O_1 و O_2 منجر می‌شود. عملگر تقاطع در شکل ؟؟ نشان داده شده‌است، که $h_{p_j}^i$ ، i امین هتل از دنباله هتل‌ها برای والدین P_j است.

در این عملگر تنها یک زیرمجموعه از هتل ها بین دو والد عوض می‌شوند، بعبارت دیگر یک دنباله از شهرها برای فرزند O_1 از والدین P_1 گرفته شده‌است در حالیکه دنباله دیگر از شهرها برای فرزند O_2 از والدین P_2 گرفته شده‌است. عملگر تقاطع ممکن است منجر به سفر های نشدنی به‌دلیل تغییر هتل اولیه یا هتل پایانی از یک سفر شود که احتمال اینکه منجر به بیشتر شدن حداکثر زمان شود، وجود دارد که جستجو ممنوعه این جواب‌های نشدنی را بهبود می‌دهد.

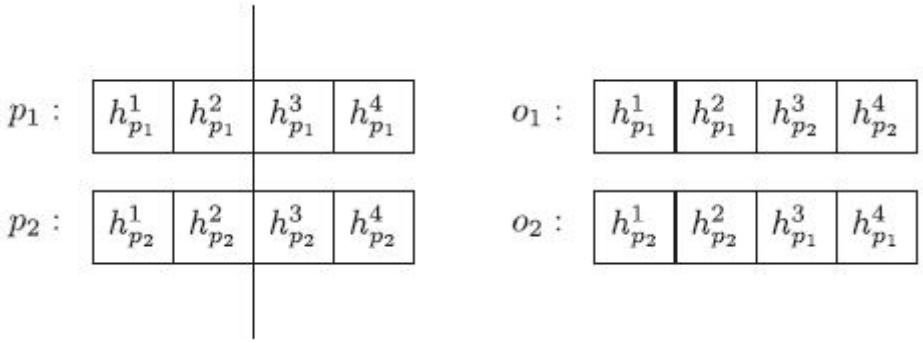
۳-۲-۱-۳ مدیریت جمعیت: تنوع و جهش

در حالت کلی معمولا احتمال کمی وجود دارد که فرزندی خصوصیتی را دارا باشد که آن مشخصه جز خصوصیات ژنتیکی والدین نباشد. در این وضعیت، حالتی از جهش به وجود آمده‌است. اگر در هنگام عمل جهش خصوصیتی برنده شود، این خصوصیت جدید شانس بروز در نسل های بعدی را پیدا می‌کند.

در این بخش فاصله به حداقل تعداد عملیات‌های ابتدایی مورد نیاز برای تبدیل یک دنباله هتل به دیگری برابر است با عملیات ابتدایی با حذف یک هتل و یا جایگزینی یک هتل با هتل دیگر تعریف شده‌است. فاصله بین دو کروموزوم H_i و H_j توسط $d(H_i, H_j)$ تعریف می‌شود و فاصله یک فرزند با دنباله هتل‌های H_0 از جمعیت P از رابطه زیر بدست می‌آید:

$$d_p(H_0) = \min_{i \in P} d(H_0, H_i)$$

اخیرا تولید دنباله هتل H_0 در صورتی که $d_p|H_0| > \Delta$ به طور آزمایشی مورد قبول واقع شده‌است و Δ یک آستانه تنوع است. اگر $d_p|H_1| \leq \Delta$ ، دنباله H_0 جهش یافته به $d_p|H_0| > \Delta$ است، درحالیکه بطور تصادفی بجای یکی از هتل‌ها در H_0 هتل‌های دیگر جایگزین می‌شود. پارامتر Δ ، بستگی به تعداد سفرهای در جواب K و مجموعه 0.5 K در طی الگوریتم



شکل ۳-۱: تقاطع تک نقطه‌ای

ممتیک است.

۳-۲-۱-۴ بهبود فرزندان و به روزرسانی جمعیت

پس از پذیرفته‌شدن هر دو فرزند O_1 و O_2 ، دنباله‌های جدید تولید شده و تورهای مربوطه با استفاده از روش جستجوی ممنوعه بهبود می‌یابند. با این حال،این جواب‌ها بهبود یافته، از بدترین جواب‌های موجود بهتر هستند. ب این صورت که بعد از تولید فرزندان قبل از اینکه وارد تکرار بعدی شود مشخص می‌کند که کدام فرزندان حق رفتن به نسل بعد را دارند. کروموزوم های نسل فعلی را در کنار فرزندان حاصل در نظر گرفته و آن‌هایی را که برحسب میزان تابع برازندگی که در اینجا کمینه کردن تعداد سفرها وطول کل تور محسوب می‌شود، را بهصورت نزولی (از بزرگ به کوچک) مرتب می‌کند آنگاه به تعداد اندازه جمعیت کروموزوم ابتدایی لیست مرتب شده را به نسل بعد منتقل و باقی را از بین می‌برند.

۳-۲-۱-۵ چک کردن نسل ها

پس از هر تکرار الگوریتم ممتیک، بررسی می‌شود که یک جواب شامل تعداد کمتری از سفرها که k^* نامیده می‌شوند، باشد. در این صورت تمام اعضای جمعیت شامل $k^* + 1$ یا سفرهایی که بیشتر باشد، حذف می‌شوند و اعضای جدید جمعیت که شامل سفرهای k^* هستند، با استفاده از روش ساخت C2 تولید شده و بااستفاده از روش جستجوی ممنوعه بهبود می‌یابند.

۳-۳ جستجوی ممنوعه

همانطور که در ابتدای این بخش گفته شد، روش جستجوی ممنوعه برای مسیریابی استفاده می‌شود. باتوجه به جواب S ، جستجوی ممنوعه در اینجا شرح داده شده و هدف به حداقل رساندن تابع وزن است.

$$F(S) = \sum_{d=1}^{D} \sum_{(i,j)\in A} C_{ij} x_{ij}^{d} + \varphi \sum_{d=1}^{D} \left[\sum_{(i,j)\in A} (C_{ij} + \tau_j) x_{ij}^{d} - L \right]^{+} \qquad (۲۷-۳)$$

که در آن x_{ij}^{d} متغیر دودویی با مقدار یک است، اگر یال (i,j) در طول روز سفر d موجود باشد، در غیر این صورت مقدار آن صفر است و $[Z]^{+} = \max(Z, 0)$. تابع وزن، مسافت کل طی شده در نظر گرفته می‌شود و یک جریمه در صورتی‌که محدودیت زمانی درنظر گرفته‌شده بیشتر شود، به تابع هدف اضافه می‌شود.

الگوریتم ؟؟ ، شبه کد مربوط به روال جستجو ممنوعه را که یک پیاده سازی جستجوی ممنوعه استاندارد است، نشان می‌دهد. در شبه کد، $\hat{S}$ نشان دهنده بهترین جواب شدنی یافت شده، است در حالیکه S^{*} مربوط به جواب، بابهترین مقدار تابع وزن $F(S)$ تاکنون یافت شده، است. ساختار همسایگی $N(S)$ بررسی توسط جستجوی ممنوعه است، که با اپراتورهای Relocate و Exchange تعریف شده‌است. مجموعه Γ ، نشان دهنده لیست کاندید هایی که ممکن است در یک تکرار حذف شوند و مجموعه Λ ، لیست حرکت ممنوعه در نظر گرفته شده‌است. لیست ممنوعه و لیست کاندیدا، یک مجموعه از تمام حرکت های ممکن است. روال جستجوی ممنوعه نیاز به یک جواب S اولیه و همچنین تعریف پنج پارامتر زیر را دارد:

- i_{max} : حداکثر تعداد تکرار بدون بهبود $\hat{S}$ و S^{*} است.
- θ^{-}, θ^{+} : تعداد حداقل و حداکثر تکرارهای حرکت ممنوعه در نظر گرفته‌شده‌است.
- u_p : تکرار با عامل جریمه φ ، به روز شده‌است.
- u_c : تکرار با هر یک از سفرهای موجود در جواب فعلی دوباره بهینه‌سازی شده‌است.

الگوریتم ۳-۲ الگوریتم جستجو ممنوعه برای مسئله فروشنده دوره گرد با انتخاب هتل.

1: **procedure** TABU SEARCH
2: initialisation: time= 0, $\Lambda = \emptyset$,$S^* = S$.
3: **if** S is feasible **then**
4: $\widehat{S} = S$
5: **end if**
6: **while** stopping criterion not met **do**
7: increase iteration counter: time=time+1
8: reset candidate list: $\Gamma = \emptyset$
9: **for** each possible move $\mu \in N(S)$ **do**
10: add μ to the candidate list Γ according to tabu and spiratin conditions
11: **end for**
12: select best move: $\mu_{best} = \arg\min_{\mu \in \Gamma} F(S \oplus \mu)$
13: update solution: $S = S \oplus \mu_{best}$
14: update tabu list: Λ
15: **if** time mod $u_c = 0$ **then**
16: improve trips in S using 3_opt operator
17: **end if**
18: reduce number of trips using $JoinTrips$ operator
19: update S^* and$\widehat{S}$ if applicable
20: **if**time mod $u_p = 0$ then update penalisation factor φ
21: **end if**
22: **end while**
23: **end procedure**

۳-۳-۱ عملگرهای جستجوی محلی

در این بخش با استفاده از سه نوع عملگر مختلف، جستجوی ممنوعه توسعه می‌یابد:

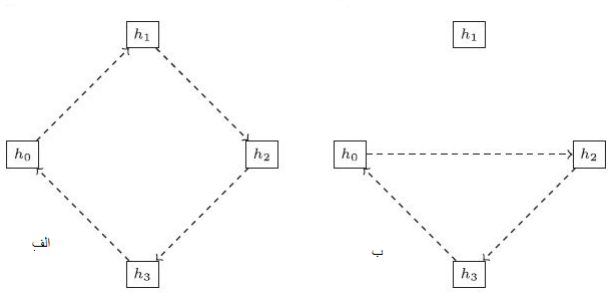
۱. سفرهای میانی عملگر سفرها 3-opt برای کوتاه کرن سفرهای موجود در جواب.

۲. دو عملگر سفر، Relocate و Exchange ، برای یک یا دو مجموعه از شهرها.

۳. یک عملگر کاهش سفر، JoinTrips ، برای ادغام دو سفر متوالی

عملگر $Relocate$ ، زنجیره ای از κ تا شهر متوالی را از یک سفر حذف می‌کند و آن را در سفر دیگر قرار می‌دهد. عملگر Exchange ، دو زنجیره از κ تا شهرهای متوالی را به واسطه دو سفر مختلف جابجا می‌کند. در روش جستجوی ممنوعه برای هر دو عملگر مقدار κ برابر ۳ در نظر گرفته‌شده‌است. وظایف عملگر جانسن به شرح زیر است:

یک تور شامل، k سفر و یک دنباله از هتل‌های مشابه $H_s =< h_1, \cdots, h_{k-1} >$ می‌شود. عملگر یک یا چند سفر متوالی را به هم متصل می‌کند، در حالیکه تضمین می‌شود جواب جدید شدنی است. در ساده ترین حالت، عملگر دو سفر متوالی i و $i+1$ را با حذف هتل میانی h_i به هم متصل می‌کند. در نتیجه تور بدست‌آمده شامل تنها k+1 سفر و دنباله ای از هتل‌های جدید $H_s =< h_1, \cdots, h_{i-1}, h_{i+1}, \cdots, h_{k-1} >$ است.بنابراین هنگام پیوستن به سفر، عملگر JoinTrip با حفظ هتل‌های باقی‌مانده برای بازدید از شهرها است. شکل ؟؟ یک تصویر گرافیکی از پیوستن دو سفر متوالی در یک مثال ساده که شامل شروع در هتل h_0 و سه هتل h_1, h_2, h_3 است، را فراهم می‌آورد. در شکل، مربع نشان دهنده هتل‌هاست در حالی که پیکان نشان دهنده مجموعه ای از شهرها که در هر سفر ملاقات می‌شوند، است. جواب اصلی در شکل ؟؟ الف نشان داده‌شده‌است در حالیکه جواب پس از پیوستن به سفر اول و دوم با حذف هتل h_1 بدست‌آمده و در شکل ؟؟ ب نشان داده‌شده‌است. جواب اصلاح شده به وضوح شامل سه سفر از چهار سفر است.



شکل ۳-۲: مثالی از $JoinTrips$ الف) تور قبل از جهش ب) تور بعد جهش

۳-۳-۲ معیار توقف

اگر جستجوی ممنوعه، در طول تکرار i_{max} ، جواب بدتری نسبت به S^* یا $\hat{S}$ داشته‌باشد، متوقف می‌شود.

۳-۳-۳ شرایط ممنوعه و معیار تنفس

لیست ممنوعه Λ ، مجموعه‌ای از جفت‌های (v, r) که در آن v شهر و r تعداد سفرهایی که ممنوع در نظر گرفته‌شده‌اند، است. اگر در هر تکرار، یک حرکت μ تغیر مکان شهرها v از سفر r به سفر دیگری باشد آنگاه جفت (v, r) را به لیست ممنوعه Λ ، برای θ تکرار بعدی اضافه می‌کنیم. پارامتر θ یک عدد صحیح تصادفی بین دو پارامتر θ^- و θ^+ مشخص شده، توسط کاربر است.

اگر سفری که ممنوع درنظر گرفته‌شده از تمام سفرهایی که تا حال یافت شده بهتر باشد، از ممنوعیت خارج شده و به عنوان بهترین سفر در نظر گرفته می‌شود و لیست ممنوعه به روز می‌شود. به عبارتی، هر حرکت μ ممکن، شامل حداقل یکی از جفت‌های موجود در Λ ، در θ تکرارهای بعدی، در نظر گرفته نمی‌شود. با این حال اگر جواب، نتیجه حرکت ممنوع به یک

جواب شدنی بهتر از $\hat{S}$ از نظر تابع هدف ؟؟ ویا به یک جواب بهتر از S^* از نظر تابع هدف ؟؟ باشد آنگاه آن حرکت به لیست کاندید Γ اضافه می‌شود، در این حالت می‌گوییم که حرکت μ معیار توقف را برآورد می‌کند.

۳-۳-۴ حرکت های اکتشافی

ساختار همسایگی کامل N(S) با جستجو ممنوعه در هر تکرار از همسایگی توسط عملگرهای Relocate و Exchange تعریف شده‌است. به عبارت دیگر در هر تکرار از روش جستجوی ممنوعه، هر دو همسایگی در دوره $O(n^2)$ بهترین حرکت شناسایی شده‌است.

۳-۳-۵ بهبود سفر و کاهش عملکرد سفر

سفرهای موجود، در جواب S فعلی بهینه‌سازی شده‌است. دوره تناوب توسط پارامتر u_c: هر تکرار u_c ، سفرهایی که با استفاده از عملگرهای Relocate و Exchange اصلاح شده، بطور جداگانه با کمک عملگر 3-opt بهبود می‌یابد.

برخلاف عملگر 3-opt عملگر JoinTrips ، در هر تکرار اعمال می‌شود. بطور خاص در هر تکرار بررسی می‌شود که آیا امکان پیوستن به دو یا چند سفر متوالی در هر زمان امکان‌پذیر است.

۳-۳-۶ به روز رسانی عامل جریمه

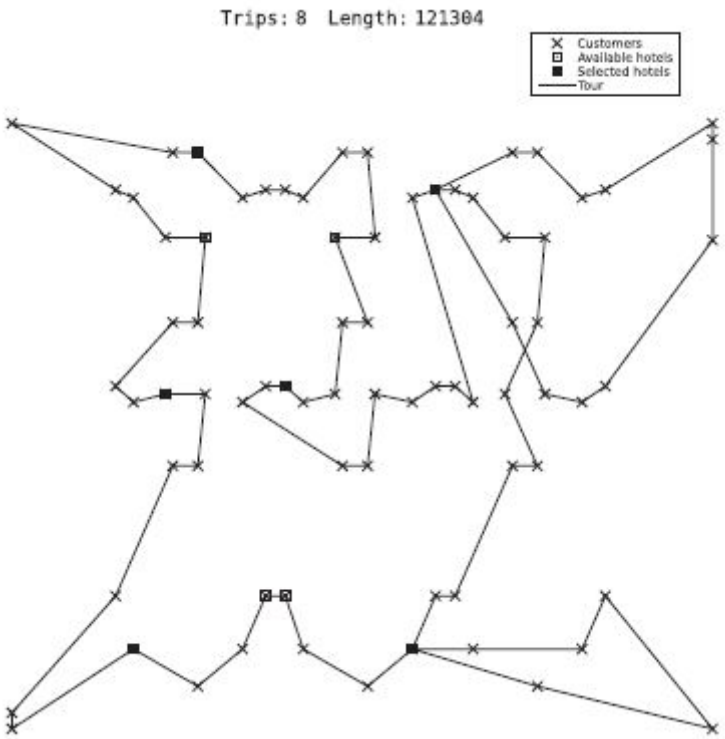
عامل φ ، جریمه در تابع وزن ؟؟ به روز رسانی دوره ای شده است که کارکرد جندرا و همکارانش برای CVRP به شرح زیر است. هر تکرار u_p وضعیت جواب u_p قبلی را مورد بررسی قرار می‌دهد. اگر تمام این جواب ها شدنی باشند، آنگاه φ توسط یک عامل از ۲ کاهش می‌یابد. اگر جواب های قبلی u_p غیر شدنی باشند، آنگاه φ توسط یک عامل از ۲ افزایش می‌یابد. این به روز رسانی پویا از عامل جریمه اجازه می‌دهد که راهی بین فضای جواب های شدنی و نشدنی برگزینند.

آزمایشاتی کاستور روی چهار مجموعه از موارد معرفی شده توسط ونستین وجن و همکارانش انجام داده‌است. نمونه‌های آزمایش شده در سایت $http://antor.ua.ac.be/tsphs$ است. کاری که ونستین وجن و همکارانش بر روی داده‌ها و به صورت کم و زیاد کردن شهرها در روش‌های مختلف انجام دادند به این صورت است:

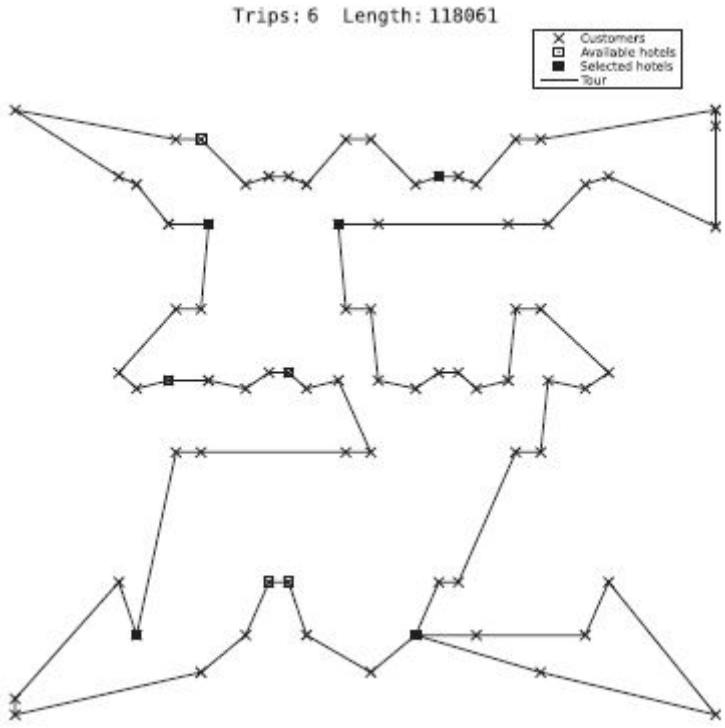
برای حل مسئله از الگوریتم ممتیک استفاده کرده‌اند، که این الگوریتم با جستجو ممنوعه بهبود داده‌است. الگوریتم ممتیک TSPHS را در دو سطح تصمیم گیری انتخاب هتل و مسیر یابی شهرها حل می‌کند. انتخاب هتل یک تصمیم کلیدی برای مسئله است زیرا یک انتخاب نامناسب هتل در کیفیت نهایی جواب تاثیر دارد. ونستین وجن و همکارانش با روش جستجو ممنوعه هتل را انتخاب می‌کند. در جدول ؟؟ نتایج به صورت زیر است:

جدول ۳-۱: نتایج محاسباتی برای نمونه معیارها در SET 1.

Instance	n	Trips	Length	Time(s)
c۱۰۱	۱۰۰	۹	۹۵۹۵٫۶	۲۴٫۱
r۱۰۱*	۱۰۰	۸	۱۷۰۴٫۶	۲۴٫۰
rc۱۰۱	۱۰۰	۸	۱۶۷۴٫۱	۲۹٫۵
c۲۰۱	۱۰۰	۳	۹۵۶۰٫۰	۱۶٫۲
r۲۰۱	۱۰۰	۲	۱۶۴۳٫۴	۱۱٫۶
rc۲۰۱	۱۰۰	۲	۱۶۴۲٫۷	۱۲٫۴
pr۰۱	۴۸	۲	۱۴۱۲٫۲	۲٫۸
pr۰۲	۹۶	۳	۲۵۴۳٫۳	۱۸٫۱
pr۰۳	۱۴۴	۴	۳۴۱۵٫۱	۴۸٫۴
pr۰۴	۱۹۲	۵	۴۲۱۷٫۴	۱۶۵٫۸
pr۰۵*	۲۴۰	۵	۴۶۵۸٫۷	۳۳۱٫۸
pr۰۶	۲۸۸	۷	۵۹۶۳٫۱	۳۲۷٫۷
pr۰۷	۷۲	۳	۲۰۷۰٫۳	۱۳٫۱
pr۰۸	۱۴۴	۴	۳۳۷۲٫۰	۶۴٫۹
pr۰۹	۲۱۶	۵	۴۴۲۰٫۳	۲۲۸٫۰
pr۱۰	۲۸۸	۷	۵۹۴۰٫۵	۴۰۹٫۰



شکل ۳-۳: روش ساخت c۱ جستجو ممنوعه



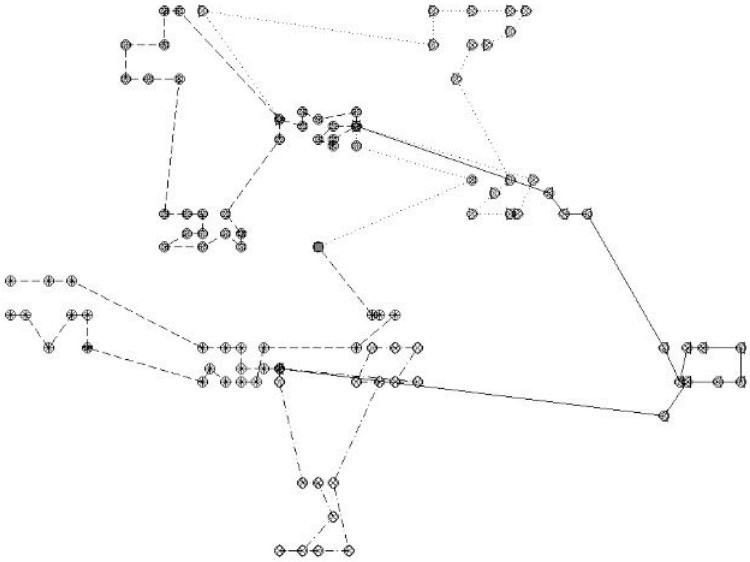
شکل ۳-۴: $MA + TS$

۳-۴ نتایج

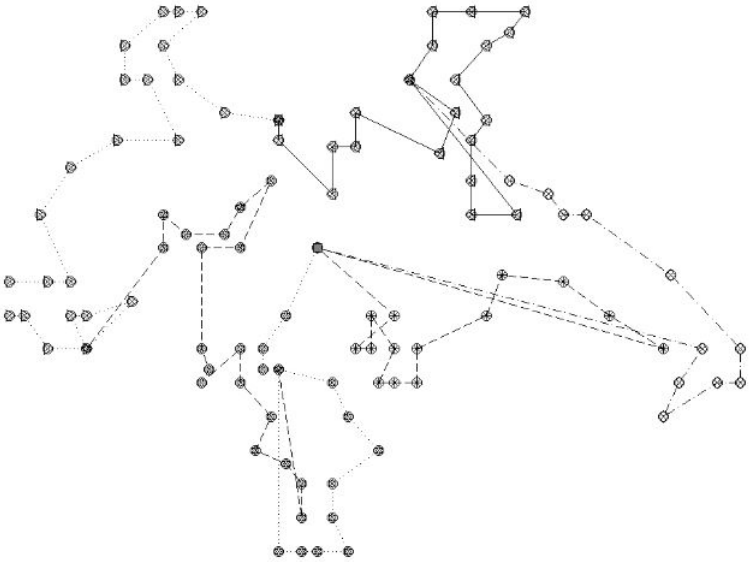
مسئله فروشنده دوره گرد با انتخاب هتل با چندین روش حل شده‌است اما سیر کاری ما به شرح زیر است:
ابتدا یک جواب اولیه به روش ساخت C1 با الگوریتم ژنتیک پیدا کردیم به این صورت که فقط شهر اول را به عنوان هتل در نظر می‌گیریم و از همه شهرها دقیقا یکبار عبور کره و به هتل باز می‌گردیم این مسیر کوتاهترین مسیر است و در نهایت این جواب اولیه را به بهترین جواب رساندیم، جواب های به دست آمده در جدول زیر و اجرای هر کدام از نمونه ها در شکل های ؟؟ ، ؟؟ ، ؟؟ و ؟؟ آمده است:

جدول ۳-۲: نتایج ما از اجرا برنامه برای نمونه معیارها در SET 1

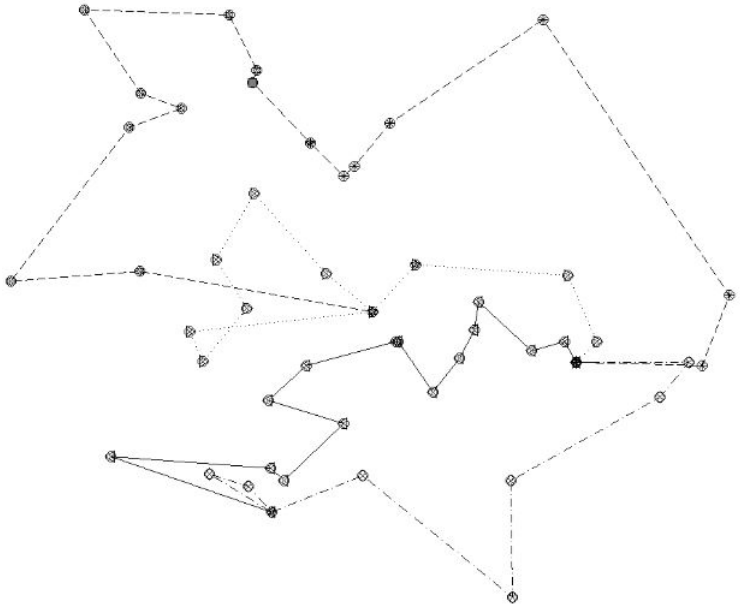
Instance	n	Trips	Length	Time(s)
c۱۰۱	۱۰۰	۶	۷۴۶٫۴	۸٫۳
r۱۰۱	۱۰۰	۵	۸۱۰٫۹	۹٫۸
rc۱۰۱	۱۰۰	۵	۸۱۶٫۸	۸٫۱
c۲۰۱	۱۰۰	۶	۸۲۲٫۷	۹٫۴
r۲۰۱	۱۰۰	۵	۸۱۰٫۹	۸٫۹
rc۲۰۱	۱۰۰	۶	۹۴۴٫۶	۹٫۲
pr۰۱	۴۸	۵	۱۱۱۲٫۷	۳٫۰
pr۰۲	۹۶	۶	۱۷۷۲٫۵	۸٫۰
pr۰۳	۱۴۴	۶	۲۵۸۳٫۲	۹٫۷
pr۰۴	۱۹۲	۵	۲۷۳۲۳٫۱	۱۰٫۴
pr۰۵	۲۴۰	۵	۳۲۵۴٫۷	۱۰٫۹
pr۰۶	۲۸۸	۵	۴۲۳۰٫۳	۱۲٫۱
pr۰۷	۷۲	۶	۱۵۵۹٫۸	۴٫۷
pr۰۸	۱۴۴	۶	۲۲۵۵٫۶	۹٫۷
pr۰۹	۲۱۶	۵	۲۹۶۱٫۱	۱۰٫۹
pr۱۰	۲۸۸	۵	۴۶۱۸٫۷	۱۱٫۹



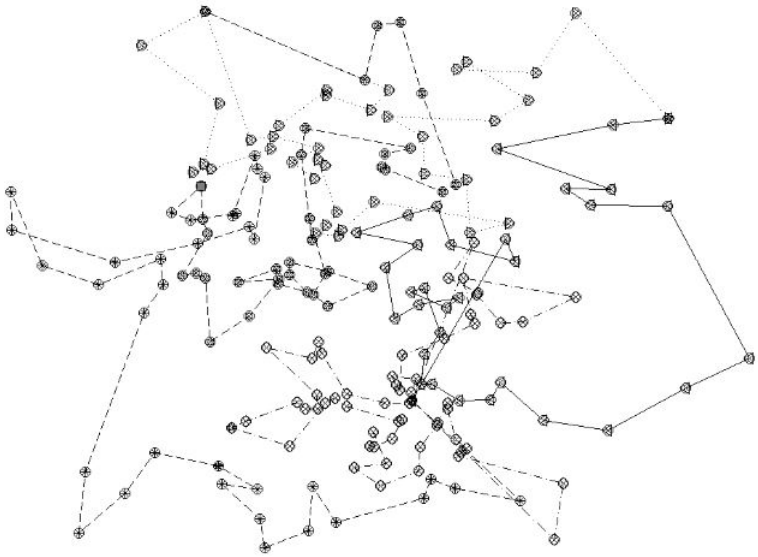
شکل ۳-۵: اجرا برای داداهای c۱۰۱



شکل ۳-۶: اجرا برای داده های c۲۰۱



شکل ۳-۷: اجرا برای داده های $pr01$



شکل ۳-۸: اجرا برای داده های $pr04$

فهرست منابع

[1] Vansteenwegen, Pieter, Souffriau, Wouter, and Sörensen, Kenneth. The travelling salesperson problem with hotel selection. *Journal of the Operational Research Society*, 63(2):207–217, 2012.

[2] Biggs, Norman, Lloyd, E Keith, and Wilson, Robin J. *Graph Theory, 1736-1936*. Oxford University Press, 1976.

[3] Al Zoubi, Reem and Salam, Asiya Abdus. Travelling salesman problem using dynamic approach. *International Journal of Computer Applications*, 94(16):20–23, 2014.

[4] Yan, Xuesong, Zhang, Can, Luo, Wenjing, Li, Wei, Chen, Wei, and Liu, Hanmin. Solve traveling salesman problem using particle swarm optimization algorithm. *International Journal of Computer Science*, 9(2012):264–271, 2012.

[5] Dorigo, Marco and Gambardella, Luca Maria. Ant colonies for the travelling salesman problem. *biosystems*, 43(2):73–81, 1997.

[6] Castro, Marco, Sörensen, Kenneth, Vansteenwegen, Pieter, and Goos, Peter. A memetic algorithm for the travelling salesperson problem with hotel selection. *Computers & Operations Research*, 40(7):1716–1728, 2013.

[7] Castro, Marco, Sörensen, Kenneth, Vansteenwegen, Pieter, and Goos, Peter. A fast metaheuristic for the travelling salesperson problem with hotel selection. *4OR*, 13(1):15–34, 2015.

[8] Baltz, Andreas, El Ouali, Mourad, Jäger, Gerold, Sauerland, Volkmar, and Srivastav, Anand. Exact and heuristic algorithms for the travelling salesman problem with multiple time windows and hotel selection. *Journal of the Operational Research Society*, 66(4):615–626, 2015.

[9] Karimi, Amirhossein and Bashiri, Mahdi. Amigrating birds algorithm for the travelling salesperson problem with hotel selection (tsphs).

[۱۰] نازنوش اطمینان، الهام پروین نیا، محمد صفری، مهدی مسعودی چله گاهی. مقایسه الگوریتم های فراابتکاری به منظور حل مسئله فروشنده دوره گرد. در چهاردهمین کنفرانس ملی سالانه انجمن کامپیوتر ایران، شیراز، ایران، اسفند ۱۳۹۵. دانشگاه آزاد اسلامی واحد شیراز.

[۱۱] یقینی، مسعود و اخوان کاظم‌زاده، محمد رحیم. الگوریتم‌های بهینه‌سازی فراابتکاری. انتشارات جهاد دانشگاهی (واحد صنعتی امیرکبیر)، تهران، ویرایش چاپ سوم، ۱۳۹۰.

[12] Neapolitan, Richard E and Naimipour, Kumarss. *Foundations of algorithms*. Jones & Bartlett Learning, 2010.

[13] Bykov, Yuri and Petrovic, Sanja. A step counting hill climbing algorithm applied to university examination timetabling. *Journal of Scheduling*, 19(4):479–492, 2016.

[14] Kirkpatrick, Scott, Gelatt, C Daniel, Vecchi, Mario P, et al. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[15] Zhan, Shi-hua, Lin, Juan, Zhang, Ze-jun, and Zhong, Yi-wen. List-based simulated annealing algorithm for traveling salesman problem. *Computational intelligence and neuroscience*, 2016:8, 2016.

[16] Sharapov, RR. Genetic algorithms: basic ideas, variants and analysis. in *Vision systems: segmentation and pattern recognition*. InTech, 2007.

[17] Yang, Xin-She and Press, Luniver. Nature-inspired metaheuristic algorithms second edition. 2010.

[18] Yang, Xin-She. *Engineering optimization: an introduction with metaheuristic applications*. John Wiley & Sons, 2010.

[19] Zhou, Lingyun, Ding, Lixin, Qiang, Xiaoli, and Luo, Yihan. An improved discrete firefly algorithm for the traveling salesman problem. *Journal of Computational and Theoretical Nanoscience*, 12(7):1184–1189, 2015.

[20] Mohsen, Abdulqader M and Al-Sorori, Wedad. A new hybrid discrete firefly algorithm for solving the traveling salesman problem. in *Applied Computing and Information Technology*, pp. 169–180. Springer, 2017.

[21] Duman, Ekrem, Uysal, Mitat, and Alkaya, Ali Fuat. Migrating birds optimization: A new metaheuristic approach and its performance on quadratic assignment problem. *Information Sciences*, 217:65–77, 2012.

[22] Hashim, Zakiah and Ismail, Wan Rosmanira. Applications of travelling salesman problem in optimizing tourist destinations visit in langkawi. in *Regional Conference on Science, Technology and Social Sciences (RCSTSS 2014)*, pp. 265–273. Springer, 2016.

[23] Kim, Byung-In, Kim, Seongbae, and Sahoo, Surya. Waste collection vehicle routing problem with time windows. *Computers & Operations Research*, 33(12):3624–3642, 2006.

[24] Friend, John. The strategic choice approach. *Wiley Encyclopedia of Operations Research and Management Science*, 2011.

[25] Glover, Fred. Tabu search—part i. *ORSA Journal on computing*, 1(3):190–206, 1989.

[26] Glover, Fred. Tabu search—part ii. *ORSA Journal on computing*, 2(1):4–32, 1990.

[27] Lin, Shen. Computer solutions of the traveling salesman problem. *The Bell system technical journal*, 44(10):2245–2269, 1965.

پیوست آ

درج کد

برنامه $MATLAB$ برای حل مسئله فروشنده دوره گرد (ایجاد یک m تور $TSPHS$)

```
function pop = create_TSPHS(NVARS,FitnessFcn,options)
%CREATE_PERMUTATIONS Creates a population of permutations.
%   POP = CREATE_PERMUTATION(NVARS,FITNESSFCN,OPTIONS) creates a
    population
%  of permutations POP each with a length of NVARS.
%
%   The arguments to the function are
%     NVARS: Number of variables
%     FITNESSFCN: Fitness function
%     OPTIONS: Options structure used by the GA

%   Copyright 2004-2007 The MathWorks, Inc.

totalPopulationSize = sum(options.PopulationSize);
n = NVARS;

pop = cell(totalPopulationSize,1);
for i = 1:totalPopulationSize
    pop{i} = randperm(nH);
end
```

برنامه $MATLAB$ برای حل مسئله فروشنده دوره گرد (تابع مسافت)

```
function distances=dist(locations,cities)

distances = zeros(cities);
for count1=1:cities
    for count2=1:count1
        x1 = locations(count1,1);
        y1 = locations(count1,2);
```

```
        x2 = locations(count2,1);
        y2 = locations(count2,2);
        distances(count1,count2)=sqrt((x1-x2)^2+(y1-y2)^2);
        distances(count2,count1)=distances(count1,count2);
    end
end

end
```

برنامه $MATLAB$ برای حل مسئله فروشنده دوره گرد(خواندن مسئله)

```
function distances=dist(locations,cities)
%  Read TSHS
%% M.Amintoosi, HSU, 2017
% Usage: [DH,DC,t] = read_TSHS_problem('c101')
function [DH,DC,t] = read_TSHS_problem(problem)
% clc
% problem='c101';
fp=fopen([ problem '.tsphs']);
nH = fscanf(fp,'%d',1);
nC = fscanf(fp,'%d',1);
t = fscanf(fp,'%f',1);

DH=zeros(nH,2); % Flow Matrix: the amount of flow between each
   pair of nodes
DC=zeros(nC,3); % Cost Matrix: the transportation cost per unit
   of flow between nodes

% Reading nodes' informations
for  k=1:nH
    i = fscanf(fp,'%d',1);
    DH(k,1) = fscanf(fp,'%f',1);
    DH(k,2) = fscanf(fp,'%f',1);
end
for  k=1:nC
    i = fscanf(fp,'%d',1);
    DC(k,1) = fscanf(fp,'%f',1);
    DC(k,2) = fscanf(fp,'%f',1);
    DC(k,3) = fscanf(fp,'%f',1);
end
fclose('all');
```

برنامه $MATLAB$ برای حل مسئله فروشنده دوره گرد (برای رسم)

```
function distances=dist(locations,cities)
function state = traveling_salesman_plot2(options,state,flag,
   locations)
%global locations
global idxH
```

```
[unused,i] = min(state.Score);
tour= state.Population{i};
%global idxC
tour(end+1) = tour(1); % For displayin with plot
% plot(locations(idxH,1),locations(idxH,2),'r*');
plot(locations(1,1),locations(1,2),'r*');
hold on
plot(locations(tour,1),locations(tour,2),'bo-');
hold off
```

برنامه $MATLAB$ برای حل مسئله فروشنده دوره گرد (حل مسئله کوچک)

```
function distances=dist(locations,cities)
function state = traveling_salesman_plot2(options,state,flag,
   locations)
function [x,fval]=tsphs0(distances,cities)
% type create_TSPHS.m

% type crossover_permutation.m

% type mutate_permutation.m

% type traveling_salesman_fitness.m

FitnessFcn = @(x) traveling_salesman_fitness(x,distances);

% type traveling_salesman_plot.m

% my_plot = @(options,state,flag) traveling_salesman_plot2(
   options, ...
%     state,flag,locations);

options = gaoptimset('PopulationType', 'custom');%,'PopInitRange
   ',[1;cities]);

% options = gaoptimset(options,'CreationFcn',
   @create_permutations, ...
%     'CrossoverFcn',@crossover_permutation, ...%
   @crossover_order1,
%     'MutationFcn',@mutate_permutation, ...
%     'PlotFcn', my_plot, ...
%     'Generations',500,'PopulationSize',60, ...
%     'StallGenLimit',200,'Vectorized','on');
options = gaoptimset(options,'CreationFcn',@create_permutations
    ...
    'CrossoverFcn',@crossover_permutation, ...%@crossover_order1
       ,
    'MutationFcn',@mutate_permutation, ...
    'Generations',2500,'PopulationSize',60, ...
```

```
    'StallGenLimit',200,'Vectorized','on');
numberOfVariables = cities;
% x = ga(fitnessfcn,nvars,A,b,Aeq,beq,LB,UB,nonlcon,options)
[x,fval] = ga(FitnessFcn,numberOfVariables,options);
end
```

برنامه $MATLAB$ برای حل مسئله فروشنده دوره گرد (اصلی)

```
function distances=dist(locations,cities)
function state = traveling_salesman_plot2(options,state,flag,
   locations)
function [x,fval]=tsphs0(distances,cities)
clc
clear
problem='c101';
[DH,DC,t] = read_TSHS_problem(problem);
tic
% locations = [DH;DC(1:7,1:2)];
locations = [DH;DC(:,1:2)];
nH = size(DH,1);
nC = size(DC,1);
idxH = 1:nH;
idxC = nH+1:nH+nC;
cities = size(locations,1);
distances=dist(locations,cities);

%%
G = sparse(distances);
km = graphallshortestpaths(G);
kfbhm = km(1:nH,nH+1:end)
max(kfbhm(:))
%%
loc=locations([1 nH+1:nH+nC],:);
cit=cities-nH+1;
dis=dist(loc,cit);
[xx,fval]=tsphs0(dis,cit);

a=xx{1};
indH1=find(a==1);
y=[a(indH1:end) a(1:indH1-1)];
x=y;
x(end+1) = x(1);
x(2:end-1)=x(2:end-1)+nH-1;
% (x)

clf
hold on
plot(locations(x,1),locations(x,2),'bo');
```

```
plot(locations(1:nH,1),locations(1:nH,2),'kp');
pause(1);
hold off

num=length(x);
S=0;
for i=1:num-1
    S=S+distances(x(i),x(i+1));
end
Maxdis=ceil(S/4);
X=cell(1,1);
k=1;
i=1;
I=0;
ext=0;
dx=zeros(1,1);
j0=0;
S=zeros(1,1);
while(i<=num)
    s=ext;
    j=j0;
    while(s<Maxdis)
        ext=0;
        if(i==num)
            break;
        end
        dx(i)=distances(x(i),x(i+1));
        if(s+dx(i)<Maxdis)
            s=s+dx(i);
            j=j+1;
            X{k}(j)=x(i);
            i=i+1;
        else
            i=i-1;
            if(sum(x(i)==(1:nH))~=0)
                break;
            end
            [ext,ind]=min(distances(X{k}(j),1:nH));
            while(s+ext>Maxdis)
                s=s-dx(i);
                i=i-1;
                if(j==1)%(i==I)
                    error('Distance more than Maxdis!Please
                       increase Maxdis.');
                end
                j=j-1;
                X{k}=X{k}(1:j);
                [ext,ind]=min(distances(X{k}(j),1:nH));
            end
```

```
            X{k}(j+1)=ind;
            s=s+ext;
            i=i+1;
            break;
        end
    end
    if(i==num)
         break;
    end
    if numel(X{k})==2
        error('Giiiiir darim')
    end

    S(k)=s;
    I=i;
    k=k+1;
    j0=1;
    X{k}(j0)=ind;
    ext=distances(X{k}(j0),x(i));
end
S(k)=s;
X{k}(end+1)=x(end);

fprintf('\nMaxdis=%g\n\n',Maxdis);
for kk=1:k
    for ii=1:length(X{kk})
        fprintf('%g -> ',X{kk}(ii))
    end
    fprintf('\nLentour(%g)=%g\n\n',kk,S(kk));
end
fprintf('\n\ntotalLen=%g\n',sum(S));
toc

ch{1}='--k+';
ch{2}='-.kd';
ch{3}='-k<';
ch{4}=':k>';
ch{5}='--kh';
ch{6}=':kh';
nx=0;
hold on
plot(locations(x,1),locations(x,2),'wo');
for i=1:k
    plot(locations(X{i},1),locations(X{i},2),ch{i});
    for j=1:length(X{i})
        nx=nx+1;
        text(locations(X{i}(j),1),locations(X{i}(j),2)+1,num2str
           (X{i}(j)));
        pause(0.2);
```

```
    end
    pause(0.5);
end
plot(locations(x(1),1),locations(x(1),2),'wp');
plot(locations(x(1),1),locations(x(1),2),'ks','MarkerFaceColor'
   ,[0.5,0.5,0.5]);
hold off
%%
axis off
print(gcf,'-dpng','Tour1.png')
```

واژه‌نامه فارسی به انگلیسی

الگوریتم برنامه‌نویسی پویا . . . . . Dynamic programming algorithm

الگوریتم بهینه‌سازی ذرات . . . . . Particle optimization algorithm

الگوریتم تپه‌نوردی . . . . . Hill climbing algorithm

الگوریتم ژنتیک . . . . . Genetic algorithm

الگوریتم شاخه‌وکران . . . . . Branch and bound algorithm

الگوریتم شبیه‌سازی تبریدی . . . . . Simulated Annealing algorithm

الگوریتم ممتیک . . . . . Memetic algorithm

بهینه‌سازی . . . . . Optimization

پیچیدگی محاسباتی . . . . . Computational complexity

تقاطع . . . . . Crossover

جهش . . . . . Mutation

جمعیت اولیه . . . . . initial Population

جستجو ممنوعه . . . . . Tabu search

جستجو محلی . . . . . Local search

فروشنده‌دوره‌گرد . . . . . Travelling salesperson

فراابتکاری . . . . . Metaheuristic

مسیریابی . . . . . Routing

معیار توقف . . . . . Stoping criterion

واژه‌نامه انگلیسی به فارسی

الگوریتم برنامه‌نویسی پویا . . . . . Dynamic programming algorithm

الگوریتم بهینه‌سازی ذرات . . . . . Particle optimization algorithm

الگوریتم تپه‌نوردی . . . . . Hill climbing algorithm

الگوریتم ژنتیک . . . . . Genetic algorithm

الگوریتم شاخه‌وکران . . . . . Branch and bound algorithm

الگوریتم شبیه‌سازی تبریدی . . . . . Simulated Annealing algorithm

الگوریتم ممتیک . . . . . Memetic algorithm

بهینه‌سازی . . . . . Optimization

پیچیدگی محاسباتی . . . . . Computational complexity

تقاطع . . . . . Crossover

جهش . . . . . Mutation

جمعیت اولیه . . . . . initial Population

جستجو ممنوعه . . . . . Tabu search

جستجو محلی . . . . . Local search

فروشنده‌دوره‌گرد . . . . . Travelling salesperson

فراابتکاری . . . . . Metaheuristic

مسیریابی . . . . . Routing

معیار توقف . . . . . Stoping criterion

Hakim Sabzevari University

An Outline of MSc. Thesis

Surname:Keyghobadi	Name:Fateme	Student No.:9423133023

Supervisor: Dr. Mahmood Amintoosi

Advisor: Dr. Mohammadali Partanian

Faculty of Mathematics and Computer Science

Program: Applied Mathematics Field:Operational Research

Title of thesis: Solving the travelling salesman problem with hotel selection using metaheuristic approaches

Keywords:Memetic algorithm, Travelling salesman problem with hotel selection, Local search, Tabu search

Abstract: In this thesis some approaches of travelling salesperson problem with hotel selection (TSPHS) is investigated. TSPHS is a kind of famous Traveling Salesman Problem (TSP). TSP is an optimization problem that aims to find the best possible route to visit all cities and return to the starting point with minimum traveling cost. TSPHS is a TSP with an extra constraint, which is about the time of travel per day. The motivation for this problem is that a salesperson often cannot visit all customers in a single day, due to the fact that he/she can only work for a limited number of hours per day. This implies that the salesperson needs to select a hotel each night, on top of determining the optimal sequence in which to visit all customers. Every day should start and end in one of the available hotels and, if a given day ends in a certain hotel, the next day should start in the same hotel. The primary goal of this problem is to minimize the required number of days, while the secondary goal is to minimize the total travelled length. Although this problem appears to be very similar to the (regular) travelling salesperson problem (TSP), it is inherently more difficult due to the hotel selection requirement. In this thesis some methods for solving TSP and/or TSPHS including dynamic programming, branch and bound, genetic algorithm, memetic algorithm, tabu search and simulated annealing is explained.

Hakim Sabzevari University
Faculty of Mathematics and Computer Science

A Thesis Submitted in Partial Fulfilment of the Requirement for the Degree of Master of Science in Applied Mathematics

Solving the travelling salesman problem with hotel selection using metaheuristic approaches

Supervisor:
Dr. Mahmood Amintoosi

Advisor:
Dr. Mohammadali Partanian

By:
Fateme Keyghobadi

July 2018