دانشکده علوم ریاضی

گروه ریاضی کاربردی

پایان‌نامه

برای دریافت درجه کارشناسی ارشد در رشته
ریاضی کاربردی، گرایش تحقیق در عملیات

عنوان

مساله مکان‌یابی پایانه‌های اتوبوس

استاد راهنما

دکترجعفر فتحعلی

استاد مشاور

دکتر برات‌اله غزنوی قصونی

پژوهشگر

فاطمه خراسانی

شهریور ۱۳۹۲


بسم الله الرحمن الرحیم

نام خانوادگی دانشجو: خراسانی نام: فاطمه

عنوان: مساله مکان‌یابی پایانه‌های اتوبوس

استاد راهنما: دکترجعفر فتحعلی
استاد مشاور: دکتر برات‌اله غزنوی قصونی

مقطع تحصیلی: کارشناسی ارشد رشته: ریاضی کاربردی گرایش: تحقیق در عملیات

دانشگاه: صنعتی شاهرود دانشکده علوم ریاضی
تاریخ فارغ‌التحصیلی: شهریور ۱۳۹۲ تعداد صفحات: ۹۰

واژگان کلیدی: مکان‌یابی ایستگاه اتوبوس، بهینه‌سازی ترکیبی، الگوریتم ژنتیک، جست‌وجوی موضعی، جست وجوی ممنوع

چکیده

در این پایان‌نامه یک روش ترکیبی و ابتکاری برای حل مساله مکان‌یابی پایانه‌های اتوبوس بیان می‌شود. در این روش مزایایی از الگوریتم ژنتیک و الگوریتم تکاملی جست‌وجوی موضعی ترکیب می‌شوند. سپس روش جدیدی برای حل مساله مکان‌یابی ایستگاه‌های اتوبوس ارائه می‌شود، در این روش از الگوریتم تکاملی جست‌وجوی ممنوع به جای الگوریتم جست‌وجوی موضعی استفاده می‌شود. در آخر نتایج عددی به دست آمده از این دو روش با هم مقایسه می‌شوند و روش بهتر معرفی می‌شود.

تقدیم به

خورشید پنهانم

امام زمانم

که جهان از نور وجودش روشن است و شقایق‌ها به بهانه‌ی دیدنش، جمعه‌ها را انتظار می‌کشند

پیشکش به

مادر مهربانم

که هم چون همیشه و بلکه بیشتر خستگی‌هایم را در آغوشش مداوا کرد

و پدر عزیزم

که کلامش هم چون مشعلی روشن و فروزان، در تاریکی راه‌های پر پیچ و خم زندگی‌ام است

هم چنین تقدیم به

خواهر و برادرانم که وجودشان گرمابخش زندگی‌ام است و دوست عزیزم فهیمه جوکار که همیشه

آرامش بخش من بوده و هست

سپاس می گویم خدای مهربانم راکه زیباترین راه زندگی را به من آموخت و با گرفتن دستان ناتوانم قدم های مرا در این راه محکم کرد، مشکلات و سختی ها را برایم آسان نمود و درهای بسته دانش را با نور وجودش به رویم گشود.

خدای مهربانم راشاکرم که تا این مرحله از مسیر علم و دانش را با موفقیت به اتمام رساندم و از او توفیق ادامه ی مسیر و حرکت در راستای خشنودیش را مسئلت دارم.

بر خود لازم می دانم از تمامی کسانی که من را در نوشتن این پایان نامه یاری کردند، قدردانی و تشکر نمایم. تشکر می کنم از استاد راهنمای محترم جناب آقای دکتر جعفر فتحعلی که با خلوص نیت و در کمال صبر و آرامش پاسخ گوی تمام سوالات من بودند. هم چنین تشکر می کنم از استاد مشاور محترم جناب آقای دکتر برات اله غزنوی قصونی و سپاسگزارم از جناب آقای دکتر مهدی زعفرانیه و جناب آقای دکتر علیرضا ناظمی که داوری این پایان نامه را بر عهده داشتند.

از خانواده محترمم به جهت تحمل شرایط سختی که در این مقطع تحصیلی داشتم، تشکر می کنم. از جناب آقای امین یوسفی به جهت راهنمایی های ارزنده ای که در نوشتن برنامه های کامپیوتری مربوطه داشتند سپاسگزارم و برای همکلاسی های عزیزم آقای محمدی و خانم ها پورآدینه، رسولی، حیدری، جوکار و رضایی آرزوی موفقیت در تمام مراحل زندگی را دارم.

فاطمه خراسانی
شهریور ۱۳۹۲

تعهد نامه

اینجانب فاطمه خراسانی دانشجوی دوره کارشناسی ارشد رشته ریاضی کاربردی دانشکده علوم ریاضی دانشگاه صنعتی شاهرود نویسنده پایان نامه مساله مکانیابی پایانه‌های اتوبوس تحت راهنمائی دکتر جعفر فتحعلی متعهد می شوم.

- تحقیقات در این پایان نامه توسط اینجانب انجام شده است و از صحت و اصالت برخوردار است.
- در استفاده از نتایج پژوهشهای محققان دیگر به مرجع مورد استفاده استناد شده است.
- مطالب مندرج در پایان نامه تاکنون توسط خود یا فرد دیگری برای دریافت هیچ نوع مدرک یا امتیازی در هیچ جا ارائه نشده است.
- کلیه حقوق معنوی این اثر متعلق به دانشگاه صنعتی شاهرود می باشد و مقالات مستخرج با نام « دانشگاه صنعتی شاهرود » و یا « Shahrood University of Technology » به چاپ خواهد رسید.
- حقوق معنوی تمام افرادی که در به دست آمدن نتایج اصلی پایان نامه تأثیرگذار بوده اند در مقالات مستخرج از پایان نامه رعایت می گردد.
- در کلیه مراحل انجام این پایان نامه ، در مواردی که از موجود زنده (یا بافتهای آنها) استفاده شده است ضوابط و اصول اخلاقی رعایت شده است.
- در کلیه مراحل انجام این پایان نامه، در مواردی که به حوزه اطلاعات شخصی افراد دسترسی یافته یا استفاده شده است اصل رازداری ، ضوابط و اصول اخلاق انسانی رعایت شده است .

تاریخ

امضای دانشجو

مالکیت نتایج و حق نشر

- کلیه حقوق معنوی این اثر و محصولات آن (مقالات مستخرج، کتاب، برنامه های رایانه ای، نرم افزار ها و تجهیزات ساخته شده است) متعلق به دانشگاه صنعتی شاهرود می باشد. این مطلب باید به نحو مقتضی در تولیدات علمی مربوطه ذکر شود.
- استفاده از اطلاعات و نتایج موجود در پایان نامه بدون ذکر مرجع مجاز نمی باشد.

* متن این صفحه نیز باید در ابتدای نسخه های تکثیر شده پایان نامه وجود داشته باشد .

دانشگاه صنعتی شاهرود

مدیریت تحصیلات تکمیلی

فرم شماره (۶)

باسمه تعالی

شماره:
تاریخ:
ویرایش:

فرم صورت جلسه دفاع از پایان نامه تحصیلی دوره کارشناسی ارشد

با تأییدات خداوند متعال و با استعانت از حضرت ولی عصر (عج) نتیجه ارزیابی جلسه دفاع از پایان نامه کارشناسی ارشد خانم فاطمه خراسانی رشته ریاضی کاربردی گرایش تحقیق در عملیات تحت عنوان مساله مکانیابی پایانه‌های اتوبوس
که در تاریخ ۱۳۹۲/۶/۲۳ با حضور هیأت محترم داوران در دانشگاه صنعتی شاهرود برگزار گردید به شرح ذیل اعلام می گردد:

قبول (با درجه : بسیار امتیاز ۱۸٫۴) ☐ دفاع مجدد ☐ مردود ☐

۱ـ عالی (۲۰ ـ ۱۹) ۲ـ بسیار خوب (۱۸/۹۹ ـ ۱۸)

۳ـ خوب (۱۷/۹۹ ـ۱۶) ۴ـ قابل قبول (۱۵/۹۹ ـ ۱۴)

۵- نمره کمتر از ۱۴ غیر قابل قبول

عضو هیأت داوران	نام ونام خانوادگی	مرتبهٔ علمی	امضاء
۱ـ استادراهنما	دکتر جعفر فتجعلی	دانشیار	
۲ـ استاد مشاور	دکتر برات اله غزنوی قصونی	استادیار	
۳ـ نماینده شورای تحصیلات تکمیلی	دکتر میثم علیشاهی	استادیار	
۴ـ استاد ممتحن	دکتر مهدی زعفرانیه	استادیار	
۵ ـ استاد ممتحن	دکتر علیرضا ناظمی	استادیار	

رئیس دانشکده : امضاء

دانشکده ریاضی
دانشگاه صنعتی شاهرود

فهرست مطالب

لیست تصاویر

لیست جداول

فصل ۱
مکان‌یابی پایانه‌های اتوبوس

۱.۱ مقدمه

طراحی شبکه‌های اتوبوس‌رانی یک مساله مهم در حمل و نقل عمومی است. یکی از گام‌های مهم در این راستا محاسبه‌ی تعداد مکان‌های بهینه پایانه‌های مورد نیاز است. در واقع این مساله به عنوان حالت خاصی از مساله مکان‌یابی تسهیلات، که یک مساله بهینه‌سازی ترکیباتی با مقیاس بزرگ است، نیاز به زمان زیادی برای حل دارد. تا کنون برای حل این مساله از روش‌های شاخه و کران، شمارش ضمنی، گرم و سرد کردن شبیه‌سازی شده، الگوریتم ژنتیک و جست‌وجوی موضعی ژنتیک استفاده شده است، که ما در این پایان‌نامه به بررسی روش جست‌وجوی موضعی ژنتیک می‌پردازیم.

یکی از مسائل مطرح در طراحی سیستم‌های اتوبوس‌رانی، مشخص کردن مکان‌های مناسب جهت ایجاد پایانه‌های اتوبوس‌رانی درون شهری است. در حالت کلی مساله‌ی مکان‌یابی تسهیلات نوعی مساله‌ی بهینه‌سازی است که هدف آن انتخاب زیر مجموعه‌ای از یک مجموعه محل‌های کاندید برای قرار دادن تسهیلات است که بیشترین خدمت‌دهی با کمترین هزینه را فراهم سازد.

تعیین مکان پایانه اتوبوس با هدف ماکزیمم کردن سرویس حمل و نقل عمومی، مساله مکان‌یابی پایانه اتوبوس نامیده می‌شود که به اختصار به صورت BTLP[۱] نمایش می‌دهیم. در زمینه مکان‌یابی و مسیریابی تسهیلات حمل و نقل جاده‌ای تا کنون تلاش‌های زیادی صورت گرفته است. لاپیر و همکاران [۳۴] چارچوبی برای طراحی سیستم توزیع ارائه کردند که درآن برای تعیین تعداد و مکان پایانه‌های بارگیری از یک الگوریتم جست‌وجوی ممنوع بهره گرفته شده است. بندر و وایت [۲] مدلی ابتکاری برای طراحی مسیر بهینه پیشنهاد دادند که مجموع هزینه حمل و نقل را کمینه می‌کند. بوزاین-ایاری و همکاران [۴] مدلی برای بهینه‌سازی میزان توقف اتوبوس‌ها در دو حالتی که محدودیت ظرفیت شبکه نرم یا سخت باشد ارائه کردند. اسچوبل

[۱] Bus terminal location problem

و گینکل [۱۸] از یک بهینه‌سازی دو معیاره برای ارضای دو هدف متناقض کاهش میزان تاخیر اتوبوس‌ها و کاهش تعداد افرادی که به علت حرکت یک اتوبوس آن اتوبوس را از دست می‌دهند بهره گرفتند.
برای حل این مسائل ابتدا آنها را به صورت مسائل مکان‌یابی p-سرویس‌دهنده‌های بدون ظرفیت که به اختصار به صورت p-UFLP[۲] نمایش می‌دهیم، فرمول‌بندی کرده و سپس با استفاده از الگوریتم‌های تکاملی روند حل را شروع می‌کنیم. برای توضیح مساله p-UFLP ابتدا به شرح مختصری از مسائل مکان‌یابی سرویس‌دهنده‌های بدون ظرفیت (UFLP) و p-میانه[۳] می‌پردازیم.

۲.۱ مساله مکان‌یابی سرویس‌دهنده‌های بدون ظرفیت

از مساله مکان‌یابی تسهیلات بدون ظرفیت در مقالات مختلف با عناوین متفاوتی نام برده شده است. UFLP یک مساله برنامه‌ریزی عدد صحیح مختلط ساده است، که می‌توان آن را به صورت برنامه‌ریزی صفر و یک نیز تبدیل کرد. ساختار خاص این مساله این اجازه را می‌دهد که تکنیک‌های خاصی برای حل آن مورد استفاده قرار بگیرد [۲۱]. مساله مکان‌یابی تسهیلات بدون ظرفیت به طور گسترده‌ای در مقالات مورد توجه قرار گرفته است [۳۳]. فرمول این مساله اولین‌بار به طور صریح توسط بالینسکی [۱] ارائه شده است. با این وجود هم‌چنان ابهاماتی در مقالات بعضی از محققان وجود دارد که حاکی از تاریخ مبهم و منابع نادرست UFLPها می‌باشد. کراروپ و پروزان [۳۳] با دقت بسیار به جست‌وجوی منشأ این مسائل پرداختند. افرویمسون و ری [۱۱] یک فرمول اصلاح شده از مساله FLP را ارائه کردند، که در آن از یک الگوریتم شاخه و کرانه برای حل آن استفاده می‌شد. سپس کوماوالا [۲۹] ساختارهای ویژه UFLP را برای توسعه الگوریتم شاخه و کرانه مورد استفاده قرار داد. بیلد و کراروپ [۳] و ارلن‌کوتر [۱۳] روش شاخه و کران دوگان را برای حل این مساله توسعه دادند. روش آن‌ها به عنوان کارآمدترین روش شناخته شده برای حل UFLP مورد استفاده قرار گرفت. گالواو و راجی [۱۵] یک روش برای حل یک فرمول کلی‌تر ارائه کردند که UFLP را به عنوان یک حالت خاص در بر دارد.

اکنون به توضیح این مسائل می‌پردازیم، فرض کنید $J = \{۱, \ldots, n\}$ مجموعه‌ای متناهی از n مکان کاندید برای تاسیس امکانات جدید و $I = \{۱, \ldots, m\}$ تعداد مشتریانی باشد که متقاضی کالای خاصی هستند، هزینه تاسیس سرویس‌دهنده در مکان $j \in J$ برابر f_j است و هزینه سفر از مکان سرویس‌دهنده $j \in J$ به مکان مشتری $i \in I$ برابر با C_{ij} است. بعضی از این مکان‌های احتمالی دارای شرایط مساعدتری جهت تاسیس می‌باشند که مکان‌های بالقوه نامیده می‌شوند و با Q نشان داده می‌شوند، شرایطی از قبیل مسافت حمل و نقل از مکان گزینش شده تا مشتری، هزینه حمل و نقل، هزینه تاسیس امکانات وغیره.

[۲] p-Uncapacitated facility location problem
[۳] p-Median problem

هدف مسائل مکان‌یابی تسهیلات بدون ظرفیت کم کردن هزینه‌ها به قسمی است که تمام تقاضاها پاسخ داده شود. بنابراین تابع هدف این مسائل به صورت زیر است:

$$Min_{Q\subseteq J}\Big\{\sum_{j\in Q} f_j + \sum_{i\in I} Min_{j\in Q} C_{ij}\Big\}$$

.

۳.۱ مساله p-میانه

مساله p-میانه که به اختصار به صورت p-MP نمایش داده می‌شود کاربردهای فراوانی دارد که از آن جمله می‌توان به مکان‌یابی مراکز شبکه‌های ارتباطی کامپیوتری، مراکز توزیع کالا، مراکز اداری، مراکز نظامی، مراکز پستی و غیره اشاره کرد.

این مساله می‌کوشد تا p سرویس‌دهنده را از بین n سرویس‌دهنده احتمالی به گونه‌ای انتخاب کند که هر مشتری دقیقاً به یکی از آن مراکز اختصاص یابد و هزینه متغیر کل(به عنوان مثال، هزینه رفت و آمد) می‌نیمم شود با این شرط که هزینه تاسیس سرویس‌دهنده‌ها که همان هزینه ثابت است صفر در نظر گرفته شود.

روش‌های تقریبی مختلفی برای حل مسئله p-میانه به کار برده شده است که از آن جمله می‌توان روش‌های مبتنی بر برنامه ریزی خطی، روش‌هایی با استفاده از گراف‌ها و روش‌های ابتکاری را نام برد. یکی از روش‌های ابتکاری برای مسئله p-میانه که توسط مارنزانا [۳۵] ارائه شده است روش جستجوی همسایگی[۴] می‌باشد. در این روش از یک جواب شدنی برای مسئله، یعنی مجموعه‌ای شامل مکان p وسیله شروع کرده و نقاط تقاضا را به نزدیک‌ترین وسیله نسبت می‌دهیم. مجموعه گره‌های نسبت داده شده به هر وسیله یک همسایگی اطراف آن وسیله تشکیل می‌دهند. در هر همسایگی، مسئله ۱-میانه را حل کرده و وسیله را به این مکان جدید تغییر مکان می‌دهیم. سپس هنگامی که برای تمام همسایگی‌ها مکان وسایل دوباره پیدا شدند، همسایگی‌های جدید را به‌دست آورده و عمل را تکرار می‌کنیم تا زمانی‌که دیگر تغییری در همسایگی‌ها یا مکان وسایل صورت نگیرد.

بر اساس روش فوق تیتز و بارت [۴۳] نیز الگوریتمی ارائه کرده‌اند که در آن برای تغییر مکان یک وسیله، رأس‌های دیگر به ترتیب بررسی شده و در صورتی که تابع هدف بهبود یافت تغییر مکان انجام می‌گیرد. این روند تا زمانی‌که دیگر بهبودی در جواب حاصل نشود ادامه می‌یابد. با این روش می‌توان یک جواب داده شده را بهبود بخشید. البته نحوه انتخاب مکان‌های جدید و جواب اولیه در سرعت الگوریتم مؤثر است. هانسن

[۴] Neighborhood search

و ملادنویچ [۲۳] مسئله p-میانه را با روش جستجوی همسایگی متغیر حل کرده‌اند که در آن دو جواب که در $k(k \leq p)$ عضو اختلاف دارند به عنوان همسایه در نظر گرفته می‌شوند.

یک روش ابتکاری نیز که به صورت الگوریتمی حریصانه[۵] و بر اساس روش تغییر مکان وسایل می‌باشد، در [۶] آمده است که در آن ابتدا نقطه ۱-میانه به‌دست آمده و در مجموعه جواب قرار می‌گیرد. سپس در هر گام یک رأس به این مجموعه اضافه شده و بررسی می‌شود که آیا می‌توان رأس دیگری را جایگزین رأس‌های انتخاب شده کرد تا تابع هدف کاهش یابد یا خیر. در صورت مثبت بودن جواب، رأس را جایگزین کرده و دوباره عمل بررسی انجام می‌شود. پس از p گام یک جواب تقریبی برای مسئله به‌دست می‌آید. این الگوریتم یک مجموعه مانند S شامل p رأس را پیدا می‌کند که لزوماً جواب بهینه نیست ولی برای هر مجموعه دیگر مانند X که شامل p رأس از شبکه باشد اگر X و S دارای $p-1$ رأس مشترک باشند آنگاه داریم $F(S) \leq F(X)$ که F نشان‌دهنده مقدار تابع هدف است.

از دیگر روش‌های ابتکاری که برای مسئله p-میانه به‌کار رفته است روش‌های الگوریتم ژنتیک[۶] و جستجوی ممنوع[۷] است. اولین کسانی که الگوریتم ژنتیک را برای مسئله p-میانه به کار بردند، هاسج و گودچایلد [۲۶] بودند. سپس دیبل و دشام [۱۰]، ارکوت و همکاران [۱۲] و کورا و همکاران [۷] روش‌هایی با استفاده از الگوریتم‌های ژنتیک برای مسئله p-میانه ارائه کردند. با استفاده از جستجوی تابو نیز مورنو و همکاران [۳۹] و رلند و همکاران [۴۱] روش‌هایی برای مسئله p-میانه ارائه کرده‌اند.

برای بررسی کارایی یک روش ابتکاری اگر بتوانیم کران‌هایی برای جواب بهینه بیابیم آنگاه می‌توانیم تعیین کنیم که آیا جواب یک روش ابتکاری به جواب بهینه نزدیک شده است یا خیر. یک روش برای تعیین کرانی برای جواب بهینه، روش آزادسازی لاگرانژ[۸] است. در این روش با انتقال یک یا چند محدودیت همراه با ضرایبی به نام ضرایب لاگرانژ، به تابع هدف یک مسئله جدید جایگزین مسئله اصلی می‌شود که جواب آن کرانی برای مسئله اصلی است. روش آزادسازی لاگرانژ برای مسئله p-میانه در [۱۴] و [۹] آمده است. در [۲۲] نیز بر روی روش‌های تقریبی گوناگون (ابتکاری و برنامه ریزی ریاضی) که توسط افراد مختلف ارائه شده بررسی‌هایی به عمل آمده است.

برای مسئله p-میانه روی درخت، کریو و حکیمی [۲۸] الگوریتمی دقیق با پیچیدگی زمانی $O(p^۲n^۲)$ ارائه کرده‌اند که آن را تمیر [۴۲] به الگوریتمی با پیچیدگی زمانی $O(pn^۲)$ بهبود بخشیده است.

اکنون به بیان مدل ریاضی این مسائل می‌پردازیم. فرض کنید $J = \{۱, \ldots, n\}$ مجموعه‌ای متناهی از n مکان احتمالی برای تاسیس امکانات جدید و $I = \{۱, \ldots, m\}$ تعداد مشتریانی باشد که متقاضی کالای خاصی

[۵] Greedy
[۶] Genetic algorithm
[۷] Tabu search
[۸] Lagrangean relaxation

هستند و هزینه سفر از مکان سرویس‌دهنده $j \in J$ به مکان مشتری $i \in I$ یا به عبارتی هزینه متغیر برابر با C_{ij} است، هم‌چنین فرض کنید p تعداد تسهیلاتی است که باید مکان‌یابی شوند، به طوری‌که $۱ \leq p \leq n$. در این مساله می‌کوشیم هر مشتری را دقیقاً به یکی از این p تسهیل به گونه‌ای تخصیص دهیم، که هزینه متغیر کل مینیمم شود. این تخصیص توسط تاسیس تسهیل متناظر با کمترین C_{ij} می‌باشد. در نتیجه هزینه متغیر کل برابر است با $\sum_{i \in I} Min_{j \in Q} C_{ij}$، به طوری‌که $Q \subseteq J$. بنابراین تابع هدف این مسائل به صورت زیر خواهد بود،

$$p - MP : Min_{\substack{Q \subseteq J \\ |Q|=p}} \left\{ \sum_{i \in I} Min_{j \in Q} C_{ij} \right\}$$

مدل برنامه‌ریزی عدد صحیح صفر و یک این مسائل به صورت زیر می‌باشد.فرض کنید،

$$x_j = \begin{cases} ۱ & \text{اگر سرویس‌دهنده } j \text{ تأسیس شود} \\ ۰ & \text{در غیر این صورت} \end{cases}$$

و

$$y_{ij} = \begin{cases} ۱ & \text{اگر مشتری } i \text{ تمام تقاضای خود را ازمرکز } j \text{ دریافت کند} \\ ۰ & \text{در غیر این صورت} \end{cases}$$

اکنون خواهیم داشت

$$\text{Min} \qquad \sum_{i \in I} \sum_{j \in J} C_{ij} y_{ij} \tag{۱.۱}$$

subject to

$$\sum_{j \in J} y_{ij} = ۱ \qquad \forall\, i \in I \tag{۲.۱}$$

$$x_j - y_{ij} \geq ۰ \qquad \forall\, i, j \tag{۳.۱}$$

$$\sum_{j \in J} x_j = p \tag{۴.۱}$$

$$x_j \in \{۰, ۱\} \qquad \forall\, j \in J \tag{۵.۱}$$

$$y_{ij} \in \{۰, ۱\} \qquad \forall\, i, j \tag{۶.۱}$$

۱.۳.۱ دو ویژگی تشخیص $UFLP$ از $p - MP$

- بر خلاف p-MP، برای مسائل مکان‌یابی تسهیلات بدون ظرفیت یک هزینه ثابت و نامنفی وابسته به هرسرویس‌دهنده بالقوه در نظر گرفته می‌شود که این هزینه تنها در زمانی خرج می‌شود که در محل موردنظر سرویس‌دهنده تأسیس شود.

- در UFLP تعداد سرویس‌دهنده‌ها از قبل معلوم نیست، ولی در p-MP تعداد آن‌ها p است.

۴.۱ مساله مکان‌یابی p-سرویس‌دهنده بدون ظرفیت ($p-UFLP$)

p-UFLP یک مدل ترکیبی است که خصوصیات معینی از p-MP و UFLP را ترکیب می‌کند. می‌توان گفت p-MP یک حالت خاص از این دسته مسائل می‌باشد، زیرا اگر برای مسائل p-MP هزینه‌های ثابت در نظر گرفته شوند این مسائل به مسائل p-UFLP بسط داده می‌شوند. هم‌چنین p-UFLPها همان UFLPها هستند با این قید اضافی که تعداد مراکز تاسیس برابر عدد مشخص p هستند. اکنون با ترکیب دو مساله فوق مدل برنامه‌ریزی عدد صحیح را برای حل مسائل مکان‌یابی p-سرویس‌دهنده بدون ظرفیت ارائه می‌کنیم،

تعریف می‌کنیم:

$$x_j = \begin{cases} ۱ & \text{اگر سرویس‌دهنده } j \text{ تأسیس شود} \\ ۰ & \text{در غیر این صورت} \end{cases}$$

و

$$y_{ij} = \begin{cases} ۱ & \text{اگر مشتری } i \text{ تمام تقاضای خود را ازمرکز } j \text{ دریافت کند} \\ ۰ & \text{در غیر این صورت} \end{cases}$$

خواهیم داشت

$$\text{Min} \quad \sum_{j\in J} f_j x_j + \sum_{i\in I}\sum_{j\in J} C_{ij} y_{ij} \tag{۷.۱}$$

subject to

$$\sum_{j\in J} y_{ij} = ۱ \qquad \forall\, i \in I \tag{۸.۱}$$

$$x_j - y_{ij} \geq ۰ \qquad \forall\, i,j \tag{۹.۱}$$

$$\sum_{j\in J} x_j = p \tag{۱۰.۱}$$

$$x_j \in \{۰,۱\} \qquad \forall\, j \in J \tag{۱۱.۱}$$

$$y_{ij} \in \{۰,۱\} \qquad \forall\, i,j \tag{۱۲.۱}$$

شرط (۸.۱) نشان می‌دهد که تمام تقاضای مشتری i برآورده شده است و شرط (۹.۱) نشان می‌دهد که هر مشتری در صورتی از یک سرویس‌دهنده تقاضا می‌کند که آن سرویس‌دهنده واقعاً تأسیس شده باشد زیرا در غیر این صورت هیچ هزینه ثابتی پرداخت نشده است، شرط (۱۰.۱) نشان می‌دهد که تنها p سرویس‌دهنده تاسیس شده‌اند.

(x_j, y_{ij})برای تمام i و j یک جواب شدنی p-UFLP است مگر اینکه مفروضات مشخصی بر روی مقادیر

هزینه‌های ثابت و متغیر در نظر گرفته شود.
با استفاده از این مدل می‌کوشیم تا زمینه را برای بهبود طرح پیشنهادی مکان‌های تسهیلاتی از قبیل دبیرستان‌ها، بیمارستان‌ها، سوله‌ها، کشتارگاه‌ها و غیره ارتقا بخشیم. در این پایان‌نامه مساله مکان‌یابی پایانه‌های اتوبوس با استفاده از این مدل، فرمول‌بندی شده و سپس با استفاده از الگوریتم‌های ابتکاری و تکاملی روند حل را ادامه می‌دهیم که در فصل ۴ توضیح داده خواهد شد.

۵.۱ تعریف مساله مکان‌یابی پایانه‌های اتوبوس

رشد روزافزون جمعیت، تمرکز اکثریت آنها در محل‌های مناسب‌تر و در عین حال پراکندگی سایر افراد در نقاط دیگر، افزایش تعداد اتومبیل‌ها در خیابان‌ها و ترافیک سنگین در خیابان‌های شهرهای بزرگ، گسترش سیستم حمل و نقل عمومی درون شهری را به عنوان یک نیاز اساسی پیش روی دولت‌ها قرار داده است که در این میان گسترش شبکه اتوبوس رانی نقش بسزایی در کاهش مشکلات فوق خواهد داشت. مساله‌ی طراحی سیستم حمل و نقل عمومی، از جمله اتوبوس رانی یکی از مسائل پیچیده در برنامه ریزی حمل و نقل است که شامل زیر مساله‌های مختلفی می‌شود.
این مساله از یک دیدگاه به دو بخش تعیین ساختار شبکه و تعیین برنامه عملیاتی سیستم تقسیم می‌شود. تعیین ساختار شبکه شامل تعیین تعداد و محل پایانه‌های شبکه و تعیین مسیر خطوط می‌شود و برنامه عملیاتی شامل تعداد اتوبوس‌های هر مسیر، زمان‌بندی حرکت و مدیریت تاخیرها می‌شود.
در این پایان‌نامه قسمتی از بخش اول مساله یعنی تعیین محل پایانه‌های شبکه مورد نظر قرار می‌گیرد.
از دیدگاه دیگر توابع هدف در مسائل مکان‌یابی به دو دسته تقسیم می‌شوند، دسته اول، سعی در افزایش کارایی کل سیستم دارند و به تک تک نقاط تقاضا توجه چندانی ندارند. دسته‌ی دیگر که معمولاً برای مکان‌یابی تسهیلات اورژانسی و یا تجهیزات خطر آفرین کاربرد دارند، سعی می‌کنند تسهیل یا تسهیلات جدید را در مکانی مستقر کنند که میزان بهره‌مندی یا گله‌مندی نقاط تقاضا از آن یکسان باشد مانند تابع مینی‌ماکس[۹] یا ماکسی‌مین[۱۰].
تابع هدفی که در این پایان‌نامه استفاده می‌شود نیز سعی در افزایش کارایی کل شبکه اتوبوس رانی دارد.
برای بیان مساله‌ی مکان‌یابی پایانه‌های سیستم اتوبوس‌رانی، یک شبکه خیابانی مد نظر قرار می‌گیرد و فرض می‌شود تمام گره‌های شبکه می‌توانند به عنوان ایستگاه‌های سیستم اتوبوس رانی در نظر گرفته شوند. همچنین فرض می‌شود که میزان سوار و پیاده شدن مسافرین در هر گره (پتانسیل گره) در دست است، برای تعیین

[۹] Minimax
[۱۰] Maximin

این پارامتر می‌توان یک سیستم فرضی اتوبوس‌رانی تعریف کرد، به گونه‌ای که از تمام خیابان‌های اصلی شبکه حداقل یک خط اتوبوس عبور کند و تا حد امکان تمام شبکه پوشش داده شود. با استفاده از نتایج حاصل از این تخصیص پتانسیل سفر برای گره‌های شبکه مشخص می‌شود، به علاوه فرض می‌شود که چنانچه پایانه‌ای در یک گره ایجاد شود نه تنها می‌تواند به تمام مسافرین خود آن گره خدمت‌دهی کند بلکه می‌تواند به مسافرین تمام گره‌های همسایه نیز خدمت دهد.

فصل ۲

الگوریتم‌های جست‌وجوی موضعی و ژنتیک

از آن‌جا که روش استفاده شده در این پایان‌نامه در خصوص حل مساله مکان‌یابی پایانه اتوبوس ترکیبی از دو الگوریتم تکاملی جست‌وجوی موضعی و ژنتیک است، در این فصل به بررسی و توضیح این الگوریتم‌ها می‌پردازیم.

۱.۲ جست‌وجوی موضعی

در طول دو دهه گذشته کاربرد بهینه‌سازی در زمینه‌های مختلفی چون مهندسی صنایع، برق، رایانه، ارتباطات و حمل و نقل گسترش یافته است. بهینه سازی خطی و غیرخطی با متغیرهای پیوسته در دهه‌های پنجاه و شصت از اصلی‌ترین جنبه‌های توجه به بهینه‌سازی بود، بهینه‌سازی ترکیبی که مربوط به مسائل بهینه‌سازی با متغیرهای گسسته می‌شود در دهه‌ی هفتاد مورد توجه قرار گرفت.

یک روش ناشیانه برای حل یک مساله بهینه‌سازی ترکیبی، شمارش کامل است که در آن برای تمامی ترکیب‌های ممکن مقدار تابع هدف محاسبه و در نهایت بهترین جواب امکان‌پذیر انتخاب می‌شود. روشن است که شیوه شمارش کامل نهایتاً به جواب دقیق مساله منجر می‌شود، اما در عمل به دلیل زیاد بودن تعداد ترکیب‌های ممکن در مسائل واقعی استفاده از آن بی‌نتیجه است. با توجه به مشکلات مربوط به روش شمارش کامل، تلاش بسیاری در توسعه روش‌های مؤثرتر و کاراتر صورت گرفته است. در همین راستا الگوریتم‌های مختلفی به وجود آمده است که مشهورترین نمونه‌های آن شاخه و کرانه و شمارش ضمنی‌اند.

شاخه و کرانه در اصل یک استراتژی تقسیم و تسخیر است. ایده‌ی آن تقسیم ناحیه شدنی به زیربخش‌های قابل کنترل‌تر و در صورت نیاز تقسیم آنها به زیربخش‌های کوچک‌تر است. در حالت کلی راه‌های زیادی برای تقسیم ناحیه شدنی وجود دارد و در نتیجه الگوریتم‌های شاخه و کرانه متعددی وجود دارند. همچنین نرم‌افزارهای متعددی بر پایه الگوریتم شاخه و کران برای حل مسائل بهینه‌سازی تهیه شده‌اند که نرم‌افزار

GAMS از آن جمله است. در روش عمومی شاخه و کرانه برای حل مسائل برنامه‌ریزی مختلط پس از مشخص شدن مقادیر تعدادی از متغیرهای صحیح در هر شاخه یک برنامه خطی برای تعیین مقادیر بهینه متغیرها حل می‌شود. روش شمارش ضمنی حالت خاصی از روش شاخه و کرانه است که در هر شاخه مقادیر تمام متغیرهای صحیح مشخص‌اند. این روش وقتی کارایی دارد که پس از مشخص شدن مقادیر متغیرهای صحیح، حل باقی‌مانده مساله برای تعیین متغیرهای پیوسته ساده باشد و نیازی به حل یک برنامه خطی نباشد.

باید توجه داشت که روش‌های دقیق برای مسائل با ابعاد بزرگ کارایی خود را از دست می‌دهند و عملکردی بهتر از شمارش کامل نخواهند داشت، به همین دلیل اخیراً تمرکز بیشتری بر روش‌های ابتکاری صورت گرفته است. برای روش‌های ابتکاری نمی‌توان تعریفی جامع و مانع ارائه کرد، با وجود این در اینجا کوشش می‌شود تعریفی تا حد امکان مناسب برای آن عنوان شود.

روش جست‌وجوی ابتکاری، روشی است که با جست‌وجو در بین جواب‌های ممکن، جوابی خوب (نزدیک به بهینه) در زمانی محدود برای یک مساله ارائه می‌کند. معمولاً هیچ تضمینی برای بهینه بودن جواب وجود ندارد و حتی نمی‌توان میزان نزدیکی جواب به دست آمده به جواب بهینه را نیز تعیین کرد.

الگوریتم جست‌وجوی همسایه یا جست‌وجوی موضعی یکی از روش‌های جست‌وجوی ابتکاری مبتنی بر تکرار است و انجام هر تکرار آن مستلزم وجود یک مکانیزم تولید جواب است. مکانیزم تولید جواب، برای هر جواب i یک همسایه R_i به وجود می‌آورد که می‌توان از i به آن منتقل شد.

الگوریتم از یک نقطه شروع می‌شود و در هر تکرار از نقطه‌ی جاری به یک نقطه همسایه جابه‌جایی صورت می‌گیرد، اگر جواب همسایه هزینه کمتری داشته باشد جایگزین جواب جاری می‌شود (در مساله حداقل‌سازی) و در غیر این صورت نقطه‌ی همسایه دیگری انتخاب می‌شود، هنگامی که هزینه یک جواب از هزینه تمام نقاط همسایه آن کمتر باشد الگوریتم پایان می یابد.

مفهوم روش جست‌وجوی همسایه از حدود چهل سال پیش مطرح شده است، از جمله اولین موارد آن کارهای کرز [۸] است که مفهوم جست‌وجوی همسایه برای حل مساله فروشنده دوره‌گرد مورد استفاده قرار گرفته است، در کارهای اخیر گزارش شده توسط ریوز [۴۰] نیز جنبه‌هایی از این شیوه یافت می‌شود.

ایده روش جست‌وجوی موضعی عبارت است از تعریف همسایگی برای جواب و حرکت از یک جواب به بهترین همسایگی در هر گام. در این روش نیاز به یک جواب اولیه می‌باشد که این جواب اولیه می‌تواند از هر روشی به دست آید (به عنوان مثال، از روش حریصانه) .

مثال ۱.۱.۲. مساله مکان‌یابی سرویس‌دهنده‌های بدون ظرفیت را در نظر بگیرید، فرض کنید تعداد مشتری‌ها $m = ۶$ و تعداد سرویس‌دهنده‌ها $n = ۴$ باشد و ماتریس هزینه‌های سفر و بردار هزینه‌های تاسیس به صورت زیر باشند.

$$C = \begin{Bmatrix} ۶ & ۲ & ۳ & ۴ \\ ۱ & ۹ & ۴ & ۱۱ \\ ۱۵ & ۲ & ۶ & ۳ \\ ۹ & ۱۱ & ۴ & ۸ \\ ۷ & ۲۳ & ۲ & ۹ \\ ۴ & ۳ & ۱ & ۵ \end{Bmatrix}$$

$$f = \{۲۱ \quad ۱۶ \quad ۱۱ \quad ۲۴\}$$

هم‌چنین فرض کنید $N = \{۱, ۲, ۳, ۴\}$ مجموعه سرویس‌دهنده‌ها باشد. یک جواب برای مساله، یک مجموعه مانند S است، به‌طوری‌که $S \subseteq N$ و $(S \neq \emptyset)$. فرض کنید یک همسایگی برای S به صورت زیر تعریف می‌شود؛

$$Q(S) = \{T \subseteq N | T = S \cup \{j\} \; for \; j \notin S \; or \; T = S \setminus \{i\} \; for \; i \in S\}.$$

یعنی مجموعه‌هایی که یا یک راس به S اضافه شده است یا یک راس از آن کم شده است به عنوان همسایه S در نظر گرفته می‌شود. برای هر جواب S، هزینه متناظر به صورت زیر محاسبه می‌شود؛

$$c(S) = \sum_{i=۱}^{۶} Min_{j \in S} C_{ij} + \sum_{j \in S} f_j.$$

به عنوان مثال فرض کنید از جواب اولیه $S_۰ = \{۱, ۲\}$ شروع کنیم، خواهیم داشت:

$$c(S_۰) = (۲ + ۱ + ۲ + ۹ + ۷ + ۳) + (۲۱ + ۱۶) = ۶۱.$$

همان‌طور که مشاهده می‌شود مشتری‌های ۲، ۴، ۵ از سرویس‌دهنده ۱ و مشتری‌های ۱،۳ و ۶ از سرویس‌دهنده ۲ سرویس‌دهی می‌شوند. با توجه به همسایگی تعریف شده مجموعه تمام همسایه‌های $S_۰$ به صورت زیر خواهند بود:

$$Q(S_۰) = \{\{۱\}, \{۲\}, \{۱, ۲, ۳\}, \{۱, ۲, ۴\}\}.$$

هزینه متناظر با هر یک از این همسایه‌ها عبارت است از:

$$c(\{۱\}) = ۶۳, \qquad c(\{۲\}) = ۶۶, \qquad c(\{۱, ۲, ۳\}) = ۶۰, \qquad c(\{۱, ۲, ۴\}) = ۸۴.$$

از آن‌جایی که در صدد کم کردن هزینه‌ها هستیم $S_۱ = \{۱, ۲, ۳\}$ با هزینه $c(S_۱) = ۶۰$ به عنوان بهترین همسایه انتخاب می‌شود.مجموعه همسایگی‌های جواب جاری به صورت زیر خواهد بود؛

$$Q(S_۱) = \{\{۱, ۲\}, \{۱, ۳\}, \{۲, ۳\}, \{۱, ۲, ۳, ۴\}\}.$$

بعد از محاسبه هزینه هر کدام از جواب‌ها $S_۲ = \{۲, ۳\}$ با مقدار هزینه $c(S_۲) = ۴۲$ به عنوان بهترین همسایه انتخاب شده و مجموعه تمام همسایگی‌ها به صورت زیر مرتب می‌شوند؛

$$Q(S_۲) = \{\{۲\}, \{۳\}, \{۱, ۲, ۳\}, \{۲, ۳, ۴\}\}.$$

بهترین همسایه $S_۲$، جواب $S_۳ = \{۳\}$ با مقدار هزینه $c(S_۳) = ۳۱$ است که مجموعه همسایگی‌های آن به صورت زیر است؛

$$Q(S_۳) = \{\{۱,۳\},\{۲,۳\},\{۳,۴\}\}.$$

مقدار هزینه‌های این جواب‌ها به صورت زیر است:

$$c(\{۱,۳\}) = ۴۹ \qquad c(\{۲,۳\}) = ۴۲ \qquad c(\{۳,۴\}) = ۵۲$$

همان‌طور که مشاهده می‌شود هیچ‌کدام از جواب‌ها بهتر از جواب جاری نیستند، بنابراین $S_۳ = \{۳\}$ به عنوان جواب بهینه موضعی نسبت به همسایگی تعریف شده معرفی می‌شود.

الگوریتم جست‌وجوی همسایه معمولاً با چند اشکال مواجه می‌شود، ممکن است الگوریتم در یک بهینه محلی متوقف شود و مشخص نشود که آیا جواب به دست آمده یک بهینه محلی است یا یک بهینه سراسری. بهینه محلی به دست آمده وابسته به جواب اولیه است و در مورد چگونگی انتخاب جواب اولیه هیچ راه‌حلی در دسترس نیست، به علاوه معمولاً نمی‌توان یک حد بالا برای زمان اجرا تعیین کرد.
البته الگوریتم مبتنی بر جست‌وجوی همسایه مزایایی نیز دارد، از جمله این‌که یافتن جواب اولیه،تعیین تابع هدف و مکانیزم تولید جواب همسایه به طور معمول ساده است با وجود آن‌که تعیین حد بالا برای زمان اجرا امکان‌پذیر نیست ولی با اطمینان می‌توان گفت که هر تکرار از الگوریتم در زمان مشخصی قابل اجراست. برای اجتناب از مشکلات موجود در روش جست‌وجوی همسایه، الگوریتم‌هایی توسط کرک پاتریک و همکارانش [۳۰]، سرنی [۵] و متراپلیس و همکارانش [۳۸]، ابداع شد که به دلیل شباهت به پدیده‌ی سرد کردن جامدات مذاب به نام گرم و سرد کردن شبیه‌سازی شده شناخته می شوند.

۲.۲ الگوریتم ژنتیک

محققان با مشاهده سازگاری، بقا، خودترمیمی، هدایت، تولیدمثل و دیگر خصوصیات سیستم‌های طبیعی و این نکته که طبیعت مسائل خود را چگونه حل می‌کند، به فکر تقلید از روش‌های طبیعی در حل مسائل سخت و طراحی سیستم‌ها افتادند. ایده اصلی الگوریتم ژنتیک هم بر اساس این فکر و بر پایه تئوری تکامل داروین[۱] شکل گرفت.
اصولاً الگوریتم ژنتیک یکی از مهم‌ترین الگوریتم‌های جست‌وجوگر ابتکاری است که از آن برای بهینه‌سازی توابع مختلف استفاده می شود، الگوریتم ژنتیک جزیی از محاسبات تکاملی است که خود جزیی از هوش مصنوعی می‌باشد.
در سال ۱۹۷۵ هالند [۲۵] به مبانی ریاضی الگوریتم ژنتیک در کتاب مشهور خود پرداخت. با توجه به

[۱] Darwin

موفقیت الگوریتم ژنتیک، جان کوزا [۳۲] در سال ۱۹۹۱ برای پیدا کردن الگوریتمی که بتواند جواب هر صورت مساله‌ای را بیابد، مفهوم برنامه‌ریزی تکاملی و برنامه‌ریزی ژنتیکیGP [۲] را ارائه کرد. تعریف زیر از گلدبرگ [۲۰] است.

تعریف ۱.۲.۲. الگوریتم‌های ژنتیکی، تکنیک‌های جست‌وجو تصادفی هستند که بر اساس انتخاب طبیعی و نسل‌شناسی طبیعی کار می‌کنند.

۱.۲.۲ زمینه‌های بیولوژیکی ژنتیک

واژگان ژنتیک

۱. **ژن**[۳]: بلوک‌های DNA یا اسیدهای نوکلئیک هستند که یک خصیصه (مثل رنگ چشم) را کد می‌کنند، هر ژن از ۴ باز نیتروژن‌دار تشکیل شده است. به ژن، ترکیب، نشان و کاشف نیز می‌گویند.

۲. **کروموزوم**[۴]: یک رشته طولانی به هم پیچیده از ژن‌ها یا همان رشته‌های DNA است، که اطلاعات ساختاری و رفتاری موجودات را در خود به صورت کد شده داراست.

۳. **آلل**[۵]: به مجموعه‌های ممکن برای یک خصیصه می‌گویند.

۴. **مکان**[۶]: هر ژن در کروموزوم موقعیت خاص خودش را دارد، به این موقعیت ژن در کروموزوم، مکان می‌گویند.

۵. **فنوتایپ**[۷]: بعد از تکامل، همان خصوصیات فیزیکی فرزند می‌باشند.

۶. **جمعیت**[۸]: مجموعه کروموزوم‌ها را جمعیت می‌گویند.

۲.۲.۲ نظریه داروین و تشریح کامل

داروین حفظ تغییرات فردی مفید و حذف تغییرات فردی زیان‌آور را انتخاب طبیعی نامید و تغییرات حاصل در موجودات زنده را به انتخاب طبیعی نسبت داد. به طور خلاصه نظریه داروین را می‌توان به صورت زیر

[۲] Genetic programming
[۳] Gene
[۴] Choromosome
[۵] Allel
[۶] Locus
[۷] Phenotype
[۸] Population

بیان کرد،

- شرایط محیط تغییر می‌کند.
- اندام‌هایی به کار افتاده و اندام‌هایی از کار می‌افتند.
- صفات جدیدی در موجودات زنده ظاهر می‌شود.
- صفت اکتسابی، موروثی می‌شود.
- موجودات زنده متنوع می‌شوند.
- بین موجودات زنده رقابت‌هایی ایجاد می‌شود (تنازع بقا).
- موجودات زنده غالب، باقی می‌مانند (انتخاب طبیعی).
- تغییرات کوچک وراثت در نتیجه انتخاب طبیعی انباشته می‌شود.
- موجودات زنده به تدریج تکامل حاصل پیدا می‌کنند.

در طبیعت فرایند تکامل هنگامی ایجاد می‌شود که چهار شرط زیر برقرار باشند:

۱. یک موجود توانایی تکثیر خود را داشته باشد.
۲. یک جمعیت از چنین موجوداتی که قابلیت تولید خود را دارند، وجود داشته باشد.
۳. تنوع در بین این موجودات وجود داشته باشد.
۴. انواع این موجودات توسط بعضی از قابلیت‌ها در زندگی محیط اطرافشان از هم مجزا می‌شوند.

تنوع موجودات در طبیعت در اختلافی است که در کروموزوم‌های آن‌ها وجود دارد و این تفاوت در ساختار و رفتار افراد یک جمعیت باعث قدرت بقای متفاوت می‌شود، چرا که تنوع ساختار و رفتار به‌نوبه خود نرخ زاد و ولد بالاتری را موجب می‌شود و طبعاً موجوداتی که برازندگی[۹] کمتری با محیط دارند دارای نرخ پایین‌تری از زاد و ولد هستند. بنابراین با گذشت زمان تعداد نفراتی که رفتار و ساختار مناسب‌تری دارند افزایش می‌یابد و جمعیت رو به بهبودی و تکامل می‌رود. با توجه به نظریه داروین بعد از چند دوره زمانی و گذشت چند

[۹] Fitness

نسل، کل جمعیت تمایل دارد موجوداتی را بیشتر در خود داشته باشد که با تغییر کروموزوم‌های جمعیت، ساختار و رفتار جمعیت تکامل می‌یابد و موجبات انجام بهتر کارها در محیط و تولیدمثل بهتر فراهم می‌شود. یکی از عواملی که پدیدآورنده تحول در موجودات می‌باشد عملگر تقاطع[۱۰] است. تقاطع در کروموزوم به معنی اتصال قسمتی (هایی) از کروموزوم A با قسمتی (هایی) از کروموزوم B است. بر طبق نظریه ثبات انواع[۱۱]، گاهی ترکیب شیمیایی ژن‌های فرزند در اثر عواملی که به طور مستقیم یا غیر مستقیم با آن‌ها ارتباط دارند تغییر می‌کند، به این عمل، جهش[۱۲] می‌گویند.

۳.۲.۲ شمای کلی الگوریتم ژنتیک

ابتدا واژگان ژنتیک را در الگوریتم ژنتیک معنا می‌کنیم،
کروموزوم: به رمز در آمده یا کد شده یک جواب یا قسمتی از یک جواب را کروموزوم می‌گویند.
ژن: عناصر تشکیل دهنده یک کروموزوم هستند که معمولاً عدد یا کلمه یا حتی دستورات برنامه‌نویسی هستند.

۴.۲.۲ ساختار کلی الگوریتم ژنتیک

در الگوریتم ژنتیک باید اندازه جمعیت N، مقادیر P_C احتمال وقوع تقاطع (نرخ تقاطع) و P_M احتمال وقوع جهش (نرخ جهش) در هر تکرار معلوم باشند. در اکثر پیاده‌سازی‌ها این مقادیر در کل اجرا ثابت هستند. ساختار کلی الگوریتم ژنتیک به صورت زیر است:

۱. تولید جمعیت اولیه به اندازه N

۲. محاسبه تابع برازندگی هر کروموزوم در جمعیت

۳. (عمل تولید نسل) ایجاد جمعیت جدید، برای این منظور N بار قدم‌های زیر را انجام می‌دهیم،

(آ) انتخاب دو کروموزوم والد از جمعیت جاری با استفاده از عملگر انتخاب،

(ب) انجام عملگر تقاطع روی والدین با احتمال وقوع تقاطع P_C و ایجاد فرزند،

(ج) با احتمال وقوع جهش P_M، انجام عمل جهش روی فرزند،

[۱۰] Crossover
[۱۱] Fixism
[۱۲] Mutation

(د) اضافه کردن فرزند به جمعیت جدید.

۴. انتخاب جمعیت جدید و جایگذاری جمعیت جاری با جمعیت جدید به وسیله عملگر انتخاب

۵. آزمون شرایط توقف و برگرداندن بهترین جواب در صورت توقف

۶. بازگشت به گام ۲

همان‌طور که مشاهده می‌شود اصول پایه‌ای الگوریتم ژنتیک بسیار عمومی است. بنابراین، برای پیاده‌سازی الگوریتم ژنتیک روی مسائل مختلف، عوامل زیادی باید تعیین و مورد بررسی قرار بگیرند. ابتدا پیش از هر چیز باید مکانیزمی برای تبدیل هر جواب مساله به یک کروموزوم تعیین کرد، پس از آن یک مجموعه از کروموزوم‌ها به عنوان جمعیت آغازین تهیه می‌گردند. اندازه جمعیت معمولاً به طور تجربی به دست می‌آید، سپس باید عملگرهای انتخاب، جهش و تقاطع را متناسب با کروموزوم‌ها و نوع مساله تعیین کرد.

۵.۲.۲ مفاهیم کلیدی الگوریتم ژنتیک:

کدگذاری

الگوریتم ژنتیک به‌جای این‌که بر روی متغیرهای مساله کار کند، همان‌طور که در بخش قبل عنوان شد با کروموزوم‌ها یا همان شکل کدشده آن‌ها سر و کار دارد. در حقیقت عمل کدکردن جواب، یکی از اصلی‌ترین کارها در پیاده‌سازی الگوریتم ژنتیک بر روی مسائل مختلف است. یکی از مهم‌ترین سؤالات در زمینه الگوریتم ژنتیک این است که بهترین کدگذاری[۱۳] برای یک مساله خاص چیست؟ به‌طور کلی رویکرد کدگذاری در الگوریتم ژنتیک به دو صورت می‌باشد:

۱. کدگذاری مستقیم: در این روش کل یک جواب به عنوان یک کروموزوم در نظر گرفته می‌شود، لازم به ذکر است که برای مسائل پیچیده عملگرهای ژنتیکی روی کروموزوم‌هایی که بدین طریق ساخته می‌شوند به خوبی عمل نمی‌کنند.

۲. کدگذاری غیر مستقیم: در این روش قسمتی از یک جواب به صورت کروموزوم، کد می‌شود.

روش‌های کدگذاری

۱. کدگذاری دودویی: این نوع کدگذاری که در این پایان‌نامه مورد استفاده قرار گرفته است، متداول‌ترین نوع کدگذاری است. در این نوع کدگذاری هر کروموزوم رشته‌ای از صفر و یک می‌باشد، کدگذاری

[۱۳] Coding

دودویی می‌تواند حالت‌های زیادی را پوشش دهد و به راحتی با عملگرهای بیتی در زبان‌های برنامه‌سازی، پیاده‌سازی شود.

برای مثال، یک مساله کوله‌پشتی را در نظر بگیرید. مساله کوله‌پشتی به صورت ساده، عبارت است از مساله‌ای که یک کوه‌نورد در هنگام انتخاب اقلام برای کوه‌نوردی با آن مواجه است. کوله‌پشتی این کوه‌نورد فضای محدودی داشته و کوه‌نورد مجبور است تعداد شیء محدودی را از بین n شیء در اختیار انتخاب کند. به دلیل محدودیت فضا، امکان انتخاب همه اشیا وجود ندارد. مساله این است که کدام اشیا انتخاب گردد که بیشترین مطلوبیت را برای کوه‌نورد داشته باشد. کدگذاری باینری، یک روش مناسب برای نشان دادن جواب‌های این مساله می‌باشد. آرایه [۱ ۱ ۰ ۰ ۱ ۱ ۰ ۰ ۱ ۱] ، یک مثال از کدگذاری باینری را برای این مساله نمایش می‌دهد. همان‌طوری که در این آرایه مشاهده می‌شود، این کد مشخص می‌کند که از بین ۱۰ شیء، اشیا شماره ۱،۴،۵،۶،۹،۱۰ انتخاب شده‌اند.

۲. کدگذاری حقیقی(ارزشی): این نوع کدگذاری در مسائلی که در آن‌ها مقادیر پیچیده نظیر اعداد حقیقی به کار می‌روند، استفاده می‌شود. این روش معمولاً به صورت یک رشته، مجموعه، ماتریس و یا غیره از اجزا بیان می‌گردد. در کدگذاری ارزشی، هر جزء ارزش خاصی دارد. این پارامتر باارزش می‌تواند عدد، حرف و یا کلمه باشد. در صورت استفاده از این نوع کدگذاری، نیاز به توسعه روش‌های خاصی برای ایجاد همسایگی، حرکت و غیره در الگوریتم‌ها می‌باشد.

۳. کدگذاری ترتیبی: این نوع کدگذاری عموماً برای حل مسائل بهینه‌سازی ترکیباتی، زمان‌بندی خودروها و غیره به کار می‌رود. در اینجا هر کروموزوم شامل اعدادی است که معمولاً ترتیب خاصی را بیان می‌کنند.

۴. کدگذاری درختی: کدگذاری درختی در برنامه‌ریزی ژنتیکی به کار می‌رود. در کدگذاری درختی، هر کروموزوم یک درخت از اشیایی نظیر توابع یا دستورها در زبان برنامه‌نویسی می‌باشند.

ایجاد جمعیت اولیه

در این مرحله، جمعیت اولیه از جواب‌هایی که معمولاً به‌صورت تصادفی تولید می‌شوند، شکل می‌گیرد. البته در بعضی از موارد با توجه به نوع مساله و برای بالا بردن سرعت همگرایی الگوریتم از روش‌های ابتکاری نیز استفاده گردیده است. محققان اخیراً مفهوم بذرافشانی[۱۴] را ارائه کرده‌اند.

[۱۴] Seeding

اعمال ژنتیک

همان‌طور که در شمای کلی الگوریتم ژنتیک دیده شد، اعمال ژنتیک برای انتخاب والدین، ایجاد فرزندان و گزینش نسل جدید می باشند. انتخاب و معرفی عملگرهای ژنتیک برای پیاده‌سازی بر روی مساله خاص یکی از مهم‌ترین مباحث مربوط به الگوریتم ژنتیک است. انتخاب صحیح عملگر باعث قدرت و سرعت الگوریتم ژنتیک می شود.

۱. **عملگر انتخاب**[۱۵]

وظیفه اصلی عملگر انتخاب، هدایت الگوریتم به نواحی امیدبخش فضای جواب است. چون در الگوریتم ژنتیک اندازه جمعیت ثابت است، عملگر انتخاب با

(آ) شناسایی جواب‌های خوب

(ب) ساختن کپی از آن‌ها

(ج) حذف جواب‌های بد

علاوه بر حفظ اندازه جمعیت، باعث تکثیر و افزوده شدن جواب های خوب در جمعیت می‌شود. عملگر انتخاب دارای سه بخش مهم زیر می‌باشد که جداگانه مورد بررسی قرار می‌گیرند.

(آ) فضای نمونه‌گیری[۱۶]

(ب) مکانیزم نمونه‌گیری

(ج) احتمال انتخاب

فضای نمونه‌گیری

عملگر انتخاب برای ایجاد نسل بعد، یا از همه نوزادان و والدین استفاده می‌کند و یا از بخشی از آن‌ها. به‌طور کلی دو نوع فضای نمونه گیری وجود دارد:

(آ) فضای نمونه‌گیری عادی[۱۷]

در این روش هر نوزاد بلافاصله پس از تولید جایگزین والد خود می‌شود. اشکال اصلی این روش در این است که هیچ تضمینی برای این‌که نوزادان تولید شده بهتر از والدین خود باشند وجود ندارد.

[۱۵] Selection operator
[۱۶] Sampling space
[۱۷] Regulator sampling space

(ب) فضای نمونه‌گیری توسعه یافته[۱۸]

در این روش کلیه والدین و نوزادان دارای شانس برابر برای انتخاب شدن هستند. اندازه فضای نمونه‌گیری برابر مجموع فرزندان و والدین است. مزیت عمده این روش در این است که کارایی الگوریتم را می‌توان با افزایش نرخ‌های جهش و تقاطع بالا برد.

مکانیزم نمونه‌گیری

برای شناسایی و انتخاب جواب‌های خوب معمولاً سه رویکرد کلی زیر مورد استفاده قرار می‌گیرند

(آ) نمونه‌گیری تصادفی (چرخ رولت[۱۹])

در این روش، سطح چرخ به بخش‌هایی تقسیم می‌شود که تعداد آن‌ها برابر تعداد اعضای جمعیت (N) است. سطح هر بخش متناسب با مقدار برازندگی هر جواب است. سپس چرخ به گردش در می‌آید، زمانی که چرخ متوقف می‌شود جواب مقابل پیکان انتخاب می‌شود، فرایند بالا N بار انجام می‌شود و جمعیت جدید انتخاب می‌شود.

احتمال انتخاب جواب kام به صورت زیر محاسبه می‌شود

$$p_k = \frac{f_k}{\sum_{i=1}^{N} f_i}, \qquad (N = \text{اندازه جمعیت}),$$

که در آن f_k مقدار تابع هدف برای جواب k و f_i مقدار تابع هدف به ازای جواب i می‌باشد. برای پیاده‌سازی مدل چرخ رولت در کامپیوتر ابتدا تمام جواب‌های جمعیت به صورت دنباله درآورده می‌شوند و احتمال انتخاب هر کدام p_k محاسبه می‌شود سپس مقدار تجمعی q_k برای هر جواب از رابطه

$$q_k = \sum_{i=1}^{k} p_i,$$

محاسبه می‌گردد، عدد تصادفی r بین صفر و یک ایجاد می‌شود، اندیس k به صورت زیر تعیین می شود

$$k = \min\{j : q_j > r\},$$

یا به عبارت دیگر کوچک‌ترین اندیسی که احتمال تجمعی بیشتر از r داشته باشد، انتخاب می شود.

تحلیل چرخ رولت

مکانیزم چرخ رولت و پیاده‌سازی آسان آن باعث شده است تا در بسیاری از الگوریتم‌های پیشنهاد

[۱۸] Enlorged sampling space
[۱۹] Roulette wheel

شده از چرخ رولت استفاده شود، همان‌طور که دیده شد در این روش به تولید N عدد تصادفی و $O(N)$ عملیات مقایسه نیاز است. این مساله باعث افزایش زمان محاسبه، در حل مسائل بزرگ می‌گردد. دو روش برای رفع این مشکل طراحی شده است، در روش اول تعداد $[p_k \times N]$ پس از هر جواب انتخاب می‌شود ([] تابع جزء صحیح است) برای انتخاب بقیه جمعیت هم از چرخ رولت معمولی استفاده می‌شود.

در روش دوم که توسط بیکر ارائه شد و به انتخاب کلی تصادفی[۲۰] (SUS) معروف است به صورت زیر عمل می‌شود.

- عدد تصادفی r انتخاب می‌شود،
- مجموعه زیر را در نظر می‌گیریم؛

$$R = \{r, r + \frac{۱}{N}, \ldots, r + \frac{N - ۱}{N}\}.$$

اکنون مانند روش چرخ رولت از اعداد مجموعه R استفاده می‌گردد. صرف‌نظر از پیچیدگی محاسبات، این روش به مقیاس وابسته است.

این روش بر اساس مقدار دقیق تابع برازندگی اجرا می‌گردد. برای مثال اگر در جمعیت یک جواب دارای برازش بسیار زیاد باشد احتمال انتخاب این جواب تقریباً ۱ است، پس نسل بعد شامل این جواب می‌باشد، یا اگر تقریباً برازندگی تمام جواب‌ها مساوی باشد عملاً انتخاب، انتخاب تصادفی ساده است. برای رفع این مشکل از انتخاب بر اساس رتبه‌بندی[۲۱] استفاده می‌شود، در این شیوه جواب‌ها مرتب می‌شوند، به بهترین جواب رتبه ۱ و به بدترین جواب رتبه N داده می‌شود، هر عنصر در لیست یک مقدار برازندگی مساوی با رتبه‌اش در لیست دریافت می کند، حال چرخ رولت بر اساس رتبه به‌کار گرفته می‌شود.

(ب) نمونه‌گیری قطعی

- انتخاب کسر: در این روش یک عدد کوچک‌تر از صد به نام T تعریف می‌شود، سپس کروموزوم‌ها را بر مبنای مقادیر برازندگی مرتب و T درصد برتر را انتخاب می‌کنند. اکنون از هر یک از آن‌ها $(۱۰۰/T)$ کپی به نسل بعد انتقال می‌دهند. به عنوان مثال $T = ۲۰$ و $N = ۲۰$ را در نظر بگیرید، بین ۲۰ کروموزوم ۴ تای اول را از لیست مرتب شده انتخاب کرده و از هر کدام $۱۰۰/۲۰ = ۵$ کپی به نسل بعد منتقل می‌شوند.

[۲۰] Stochastic universal sampling
[۲۱] Ranking

- انتخاب نخبه‌گرا[۲۲]:
  در این روش بهترین کروموزوم به تعداد N به نسل بعد منتقل می‌شود.
- انتخاب جایگزینی نسلی اصلاح شده[۲۳]: در این روش ابتدا کروموزوم‌ها بر اساس مقدار برازش منظم شده، سپس به تعداد نوزادان تولید شده، از انتهای لیست کروموزوم‌ها حذف می‌گردند و سپس نوزادان جایگزین کروموزوم‌های حذف شده می شوند.
- انتخاب $(\mu+\lambda)$: در این روش λ نوزاد تولید و این تعداد با μ کروموزوم که در حقیقت همان والدین هستند جهت بقا رقابت می‌کنند. بعد از رقابت این دو گروه، کروموزوم‌هایی که جزء بهترین‌ها بوده‌اند انتخاب می‌شوند. این روش از ورود کروموزوم‌های مشابه جلوگیری می‌کند.

(ج) انتخاب مسابقه

رقابتی بین دو جواب انتخاب شده شکل می‌گیرد. جواب بهتر، برنده می شود و به مرحله بعد راه پیدا می‌کند، این فرایند ادامه پیدا می‌کند تا N جواب بهتر انتخاب شوند.
روش بهتر این است که ما هر جواب را در دو مسابقه شرکت دهیم، در این صورت جواب‌های خوب با احتمال بیشتری به مرحله بعد می‌رسند و جواب‌های بد احتمال حذفشان بیشتر می‌شود، واضح است که ما در مرحله بعد از هر کدام از جواب‌ها صفر، یک یا دو نسخه داریم.

۲. عملگر تقاطع

این عملگر ترکیب قسمت‌های خوب جواب را بر عهده دارد و باعث ارث‌بری نسل جدید از نسل قدیم می‌شود. تقریباً تمام عملگرهای تقاطعی پیشنهاد شده، دو جواب را به‌طور تصادفی انتخاب می‌کنند و به‌طور تصادفی قسمتی از دو جواب والد را با هم عوض می‌کنند. احتمال وقوع تقاطع با P_C نمایش داده می‌شود.
نکته مهم در عمل تقاطع این است که P_C، نقش مهمی در سرعت و بهبود الگوریتم ژنتیک دارد. اگر احتمال تقاطع بالا باشد ترکیبات مختلف در جمعیت زیاد می‌شود ولی احتمال خراب شدن جواب‌های خوب و رفتن به نواحی غیر ضروری هم افزایش می‌یابد.

انواع عملگرهای تقاطعی

(آ) یک نقطه برش

یک نقطه در طول کروموزوم به عنوان محل برش انتخاب می‌شود و کروموزوم به دو قسمت

[۲۲] Elistit selection
[۲۳] Modifacial generational replacement selection

تقسیم می شود، اکنون این قسمت‌های جدا شده از دو والد را با هم عوض می‌کنیم، اگر $P_۱$ و $P_۲$ والدین باشند، به صورت زیر،

$$P_۱ : ۱۱۱۱۱/۰۰۰$$

$$P_۲ : ۰۰۰۰۰/۱۱۱$$

آن‌گاه فرزندان به صورت زیر خواهند بود،

$$O_۱ : ۱۱۱۱۱/۱۱۱$$

$$O_۲ : ۰۰۰۰۰/۰۰۰$$

لازم به توضیح است که محل این نقطه برش، می‌تواند در ابتدا، یا به‌طور تصادفی در زمان اجرا انتخاب شود.

(ب) چند نقطه برش

این روش که بیشتر در کدگذاری دودویی مورد استفاده قرار می‌گیرد مانند حالت یک نقطه برش است، فقط چندین نقطه برش انتخاب می شوند. معمولاً اگر کروموزوم از کدگذاری چند متغیر به وجود آمده باشد بهتر است که نقطه برش، محل اتصال متغیرها انتخاب شود. تعداد و محل نقاط برش را می‌توان از ابتدا معلوم کرد یا در طول اجرا به‌طور تصادفی انتخاب شوند. اگر والدین به صورت زیر باشند،

$$P_۱ : X_۱/Y_۱/Z_۱/W_۱,$$

$$P_۲ : X_۲/Y_۲/Z_۲/W_۲,$$

فرزندان به صورت زیر خواهند بود،

$$O_۱ : X_۱/Y_۲/Z_۱/W_۲,$$

$$O_۲ : X_۲/Y_۱/Z_۲/W_۱.$$

(ج) یکنواخت

بر اساس این عملگر، یک ژن از هر دو والد به طور مستقل از سایر ژن‌ها، شانس برابر برای حضور در کروموزوم یک فرزند را دارند. در این حالت، بر اساس یک توزیع تصادفی باینری مشخص می‌گردد که یک ژن از کدام والد انتخاب گردد. مثلاً اگر توزیع صفر و یک عدد ۱ را نشان دهد، آن ژن از والد اول و اگر ۰ را نشان دهد از والد دوم انتخاب می‌شود. این عمل برای تمامی ژن‌های یک فرزند انجام می‌گردد. در نتیجه، فرزندان ترکیبی از ژن‌های والدین خواهند بود. در مثال زیر، در صورتی که عدد تصادفی ۱ باشد، ژن فرزند از والد اول و در صورتی که

صفر باشد، از والد دوم انتخاب می‌گردد. برای فرزند دوم، عکس فرزند اول در نظر گرفته شده است.

$P_۱$: ۱۰۱۱۰۰۱۱

$P_۲$: ۰۰۰۱۱۰۱۰

مقدار تصادفی : ۱۱۰۱۰۱۱۰

$O_۱$: ۱۰۰۱۱۰۱۰

$O_۲$: ۰۰۱۱۰۰۱۱

(د) بخش ـ نگاشته

این روش که به PMX معروف است، در حقیقت همان عملگر دو نقطه برش می باشد که برای کدگذاری ترتیبی ارائه شده است. در این روش دو عدد به صورت تصادفی به عنوان نقطه برش انتخاب می‌گردند، سپس قسمت مابین دو نقطه برش در دو کروموزوم تعویض می‌شوند و آن‌گاه قسمت‌های دو طرف نقاط برش طوری مقدارگذاری می‌شوند که در هیچ‌کدام از دو کروموزوم، تکراری صورت نگیرد. روند این‌چنین است که در والد اول از ابتدای کروموزوم شروع نموده و هر ژنی که در قسمت مابین کروموزوم جدید نباشد عیناً نوشته می‌شود و برای تکراری‌ها جای خالی قرار داده می‌شود، سپس در والد دوم از ابتدای کروموزوم شروع کرده و هر عددی که در نوزاد جدید نباشد به جای محل خالی گذاشته می‌شود تا کلیه مکان‌های خالی پر گردد، برای نوزاد دوم نیز به همین صورت عمل می‌شود. به عنوان مثال

$P_۱$: ۴۳/۶۵۲۸/۷۹۱

$P_۲$: ۶۵/۸۳۴۹/۲۱۷

(۱) $O_۱$: − − /۸۳۴۹/ − −−

$O_۲$: − − /۶۵۲۸/ − −−

(۲) $O_۱$: − − −/۸۳۴۹/۷ − ۱

(۳) $O_۱$: ۶۵/۸۳۴۹/۷۲۱

$O_۲$: ۴۳/۶۵۲۸/۹۱۷

(ه) ترتیب

در ادبیات الگوریتم ژنتیک به این عملگر OX گفته می‌شود و عموماً برای کدگذاری ترتیبی مورد استفاده قرار می‌گیرد. مانند PMX در اینجا هم دو نقطه تصادفی برای برش انتخاب می شوند، سپس قسمت مابین دو نقطه، در دو کروموزوم ثابت نگه داشته می‌شود. برای نوزاد اول در والد دوم از ابتدای کروموزوم شروع کرده و ژن‌هایی که در قسمت به ارث‌برده از والد ۱ نباشد به ترتیب در محل جاهای خالی قرار می‌گیرند، برای فرزند دوم هم فرایندی مشابه عمل می شود. به عنوان مثال

$$P_۱ : ۴۷۶/۳۵۹۸/۱۲$$

$$P_۲ : ۸۳۵/۷۶۴۱/۲۹$$

$$(۱) \quad O_۱ : - - -/۳۵۹۸/ - -$$

$$(۲) \quad O_۱ : ۷۶۴/۳۵۹۸/۱۲$$

$$O_۲ : ۳۵۹/۷۶۴۱/۸۲$$

(و) چرخه

این عملگر که به CX معروف است و معمولاً برای کدگذاری ترتیبی به کار می‌رود، ابتدا اولین ژن را عیناً از والد اول به نوزاد اول کپی و سپس چرخه بین ژن‌هایی که دو کروموزوم والد اول و والد دوم وجود دارد ایجاد می‌کند، نقطه شروع همان ژن فوق می‌باشد. برای ایجاد چرخه باید ابتدا مکان ژن را در کروموزوم دوم یافته، سپس ژن موجود در همان مکان از کروموزوم اول تثبیت گردد. این عمل تا جایی که چرخه کامل شود ادامه می‌یابد (تا جایی که به نقطه شروع برسد) در این لحظه برای تکمیل نوزاد اول باید ژن‌های باقی‌مانده از کروموزوم دوم را به همان ترتیب نوزاد اول جایگذاری نمود.

$$P_۱ : ۳۴۶۵۷۸۲۹۱$$

$$P_۲ : ۷۹۴۳۵۶۲۸۱$$

$$(۱) \quad O_۱ : ۳ - -۵۷ - - - -$$

$$(۲) \quad O_۱ : ۳۹۴۵۷۶۲۸۱$$

(ز) مکان-محور

این عملگر ابتدا تعدادی ژن والد اول را به صورت تصادفی انتخاب کرده و عیناً به نوزاد اول

انتقال می‌دهد، سپس ژن‌های انتخاب شده از والد اول در والد دوم حذف شده و جاهای خالی در نوزاد اول با ژن‌های به دست آمده از والد دوم پر می‌شود.به صورت زیر

$$P_۱ : ۴۶۵۳۱۷۲۸۹$$

$$P_۲ : ۸۳۵۱۷۹۴۲۶$$

$$(۱) \quad O_۱ : - - ۵ - ۱ - - ۸ -$$

$$(۲) \quad O_۱ : ۳۷۵۹۱۴۲۸۶$$

(ح) ترتیب-محور

در این روش ابتدا تعدادی عدد تصادفی به عنوان مکان ژن‌ها در کروموزوم‌های والد تولید شده، سپس برای تولید نوزاد اول به این صورت عمل می‌شود که در والد دوم ژن‌هایی که برای ژن‌های موجود در مکان‌های متناظر با اعداد تصادفی به دست آمده از والد اول هستند حذف شده و بقیه ژن‌ها عیناً به عنوان ژن‌های فرزند اول در نظر گرفته می‌شوند. حال برای تکمیل فرزند اول به ترتیب ژن‌های انتخاب شده تصادفی از والد اول قرار می‌گیرند

$$P_۱ : ۳۴۷۶۸۹۵۱۲$$

$$P_۲ : ۱۳۲۴۶۹۷۵۸$$

اگر سه عدد تصادفی ۱، ۵ و ۷ به دست آمده باشند، داریم

$$O_۱ : ۱۳۲۴۶۹۷۸۵$$

$$O_۲ : ۳۴۱۶۸۹۵۷۲$$

در پایان باید متذکر شد که علاوه بر عملگرهای معرفی شده در بالا عملگرهای بسیار دیگری وجود دارند که در مسائل فازی و غیره به کار می‌روند. در ضمن عملگرهای تقاطعی هم وجود دارند که بیشتر از دو والد را در ساختن فرزندان مورد استفاده قرار می‌دهند که می‌توان به UNDX و تقاطع سیمپلکس اشاره کرد.

۳. عملگر جهش

این عملگر، عمل جهش طبیعی را شبیه‌سازی می‌کند و باعث ورود اطلاعات جدید به جمعیت می‌شود. این عملگر، ابتدا چند ژن از یک کروموزوم را انتخاب می کند و مقادیر آن‌ها را تغییر می‌دهد. کروموزوم موردنظر غالباً، حاصل فرایند تقاطع است. احتمال وقوع جهش در یک کروموزوم جهش را نرخ جهش می‌گویند و با P_M نشان می‌دهند. مواردی که در این عملگر مهم هستند عبارتند از

- تعداد محل‌هایی که قرار است تغییر یابد
- نحوه انتخاب محل‌ها
- نحوه تعیین عملیات تغییر

انواع عملگرهای جهش عبارتند از

(آ) یکنواخت

در این عملگر، یک ژن از کروموزوم به صورت تصادفی انتخاب شده و مقدار آن به مقدار تصادفی دیگری تغییر می‌یابد، یعنی یک عدد تصادفی در بازه $[۱, L]$ که L طول کروموزوم می‌باشد، انتخاب شده و ژن موجود در آن مکان از کروموزوم تغییر می‌کند. به عنوان مثال فرض کنید

$$P : ۰۱۰۰۱۰۱۱۰$$

یک کروموزوم و عدد تصادفی تولید شده ۵ باشد ، بنابراین کروموزوم موجود در محل ۵ عوض می شود، یعنی یک تبدیل به صفر شده در نتیجه به صورت زیر در می‌آید

$$O : ۰۱۰۰۰۰۱۱۰$$

(ب) کرانی

این روش مانند روش یکنواخت بوده و به این صورت تعریف می‌شود که به جای ژن x_k به صورت تصادفی یکی از دو حد پایین یا بالای x_k جایگزین می‌گردد. این عملگر معمولاً در کدگذاری حقیقی به کار برده می شود.

(ج) جابه‌جایی

در این روش دو ژن به صورت تصادفی انتخاب شده و با یکدیگر تعویض می‌شوند. مثلاً اگر

$$P : ۳۴\underline{۷}۵۲\underline{۸}۱۶$$

و دو عدد تصادفی ۳ و ۶ باشند، آن‌گاه خواهیم داشت

$$O : ۳۴\underline{۸}۵۲\underline{۷}۱۶$$

(د) وارونگی

در این روش ابتدا دو عدد تصادفی را بین $[۱, L]$ که L طول کروموزوم می‌باشد انتخاب کرده و کروموزوم به سه بخش تقسیم می شود. حال قسمت وسط برعکس می‌گردد. مثلاً فرض کنید

$$P : ۴۳/۵۷۱/۶۲۸$$

و دو عدد تصادفی ۲ و ۵ باشند، به صورت زیر در می‌آید

$$O : ۴۳/۱۷۵/۶۲۸$$

(ه) جایگذاری

در این روش یک ژن به صورت تصادفی انتخاب شده و جای آن عوض می‌شود، مثلاً فرض کنید

$$P : ۳۴۲۶۵\underline{۷}۱۸$$

وعدد تصادفی به‌دست آمده برای مشخص کردن ژن جهت جابه‌جایی برابر ۶ بوده و همچنین برای محل جدید آن نیز مکان بین ۳ و ۴ انتخاب شده باشد، در این صورت O به صورت زیر در می‌آید

$$O : ۳۴۲\underline{۷}۶۵۱۸$$

(و) تغییر مکان

این عملگر حالت کلی‌تر جایگذاری بوده که به جای انتخاب یک ژن یک مجموعه از کروموزوم انتخاب شده و جای آن‌ها عوض می شود. مثلاً فرض کنید

$$P : ۳۷۶۲۵۴۸۱$$

و زیرمجموعه ۶-۲-۵ انتخاب شده و محل جدید نیز اول کروموزوم باشد، آن‌گاه P تبدیل به رشته زیر می‌شود

$$O : ۶۲۵۳۷۴۸۱$$

(ز) ابتکاری

این روش یک حالت ابتکاری بوده و روش کار به این صورت است که ابتدا چند ژن (تعداد از قبل معلوم است) به صورت تصادفی انتخاب شده و تمام جایگشت‌های ممکن آن به عنوان حالت جدید در نظر گرفته می‌شود، سپس از میان کروموزوم‌های تولید شده بهترین آن به عنوان نوزاد انتخاب می‌شود، مثلاً فرض کنید

$$P : ۶۷۳۵۱۸۲۴$$

و سه ژن به‌دست آمده ۲، ۱ و ۷ باشند، حالات مختلف عبارتند از

$$O_۱ : ۶۷۳۵۲۸۱۴$$

$$O_۲ : ۶۲۳۵۱۸۷۴$$

$$O_۳ : ۶۲۳۵۷۸۱۴$$

$$O_۴ : ۶۱۳۵۲۸۷۴$$

$$O_۵ : ۶۱۳۵۷۸۲۴$$

اکنون تابع برازش را برای هر کروموزوم به دست آورده و بهترین نتیجه انتخاب می‌شود.

باید توجه داشت که تعیین پارامترهای P_M، P_C و اندازه جمعیت، تاثیر مستقیم به عملکرد الگوریتم ژنتیک می‌گذارد. مثلاً اگر P_M زیاد باشد به خاطر جهش‌های فراوان، احتمال ورود اطلاعات جدید افزایش پیدا می‌کند ولی در عوض باعث منقرض شدن مواد ژنتیکی می‌شود. افزایش جمعیت باعث افزایش گوناگونی می‌شود ولی سرعت الگوریتم را پایین می‌آورد. سؤال مهمی که در اینجا مطرح می‌شود این است که مقدار بهینه این پارامترها چیست؟
در حالت کلی نمی‌توان به این سؤال به طور دقیق جواب داد، چون تحلیل این پارامترها به ساختار مساله‌ای که الگوریتم ژنتیک برای آن تعریف می‌شود بستگی پیدا می کند.

تابع برازندگی (برازش)

تابعی است که خوب بودن یک جواب را تعیین می‌کند و معمولاً بین صفر و یک نرمال می‌شود. تابع برازندگی[۲۴] را معمولاً تابع هدف یا تابعی یکنوا از تابع هدف می‌گیرند، ولی در مواردی هم با گذاشتن جریمه یا تشویق در تابع برازندگی جواب‌ها را به سمت خاصی از فضای جست‌وجو رهنمون می‌کنند.

شرط توقف

یکی دیگر از عوامل بهبود الگوریتم ژنتیک برای پیاده‌سازی روی یک مساله خاص شرط توقف است. عمده‌ترین شرطهای توقف را می‌توان به صورت زیر فهرست کرد؛

۱. رسیدن به جوابی که صلاحیت ۱ را داشته باشد (در حالت نرمال). این روش معمولاً خیلی کند است چون محاسبه تابع برازندگی در اکثر موارد تقریبی است مقدار دقیق ۱ مطلوب نیست.

۲. رسیدن به جوابی که دارای صلاحیت (برازندگی) بیشتر از مقدار تعیین شده باشد.

۳. تولید تعداد معلوم نسل.

[۲۴] Fitness

۴. رسیدن به جمعیتی که برازندگی آن‌ها تقریباً یکی باشد یا تغییرات محسوس در جمعیت صورت نگیرد و یا واریانس برازندگی جواب‌ها از مقدار مشخص کمتر نگردد.

برای درک بهتر مزیت‌های الگوریتم ژنتیک ابتدا مشکلاتی را که در روش‌های کلاسیک (روش‌هایی که در هر تکرار یک جواب را بهتر می‌کنند و از قوانین و تبدیلات غیراحتمالی استفاده می‌کنند) وجود دارند، مورد بررسی قرار می‌دهیم.
بیشتر الگوریتم‌های کلاسیک به صورت نقطه به نقطه و با استفاده از یک نقطه شروع و با اعمال و قوانین دقیق یک جهت را مشخص می‌کنند، و با حرکت در آن جهت به سمت نقطه بهینه حرکت می‌کنند. حال این نقطه را به عنوان یک نقطه شروع برمی‌گزینند. فرآیند بالا تا رسیدن به شرط توقف تکرار می‌گردد. روش‌های کلاسیک را با توجه به نوع انتخاب جهت به دو دسته زیر تقسیم می‌کنند،

- جست‌وجوی مستقیم
- روش‌های گرادیان

مثال ۲.۲.۲. حل مساله کوله پشتی با استفاده از الگوریتم ژنتیک

مساله کوله پشتی به صورت ساده، عبارت است از مساله‌ای که یک کوه‌نورد در هنگام انتخاب اقلام برای کوه‌نوردی با آن مواجه است. کوله پشتی این کوه‌نورد فضای محدودی داشته و کوه‌نورد مجبور است تعداد شیء محدودی را از بین n شیء که در اختیار دارد، انتخاب کند. به دلیل محدودیت فضا، امکان انتخاب همه اشیا وجود ندارد. کوه‌نورد به دنبال این است که اشیایی را انتخاب کند که بیشترین مطلوبیت را داشته باشند. در این مثال، فرض کنید ۱۰ شیء با مشخصات زیر در اختیار باشد و حداکثر وزن قابل قبول کوله برابر با ۲۰ کیلوگرم باشد.

گام صفر: تولید جمعیت اولیه

شماره شیء	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰
مطلوبیت	۶۵	۸۵	۳۲	۹۸	۲۴	۲۶	۵۷	۴۳	۶۴	۲۱
وزن	۵	۵/۵	۲/۵	۶	۱/۵	۱/۸	۳/۵	۳	۴	۲

جدول ۱.۲: مساله کوله پشتی با ۱۰ شیء

در این مرحله، یک جمعیت اولیه با ۴ کروموزوم به صورت تصادفی تولید می‌گردد. جدول (۲.۲) جمعیت اولیه تولید شده را نشان می‌دهد. همان‌طور که مشاهده می‌شود در این مساله از کدگذاری دودویی استفاده شده است. در این آرایه‌ها ۰ یا ۱ قرار گرفته‌اند که نشان‌دهنده حضور یا عدم حضور شیء مرتبط در کوله می‌باشند. نحوه کدبرگردانی کدهای داخل جدول به صورت زیر انجام می‌گیرد. برای مثال کروموزوم ۱ را

شماره کروموزوم	آرایه	وزن	امکان‌پذیری	F(x)	احتمال
۱	[۱۰۰۱۱۰۰۱۰۱]	۱۷/۵	Ok	۲۵۱	۰/۲۵
۲	[۱۱۰۱۰۰۱۰۰۰]	۲۰	Ok	۳۰۵	۰/۳۰
۳	[۰۰۱۰۱۱۱۰۱۱]	۱۵/۳	Ok	۲۲۴	۰/۲۲
۴	[۰۱۰۰۰۱۱۰۱۰]	۱۴/۸	Ok	۲۳۰	۰/۲۳

جدول ۲.۲: جمعیت اولیه تولید شده

در نظر بگیرید. مقدار ارزش این کد و وزن کوله به روش زیر محاسبه می‌گردد.

$$F([۱۰۰۱۱۰۰۱۰۱]) = ۶۵ + ۹۸ + ۲۴ + ۴۳ + ۲۱ = ۲۵۱.$$

$$W([۱۰۰۱۱۰۰۱۰۱]) = ۵ + ۶ + ۱/۵ + ۳ + ۲ = ۱۷/۵.$$

با توجه به این‌که وزن اشیا این جواب کمتر از ۲۰ می‌باشد، جواب امکان‌پذیر محسوب می‌گردد و نیاز به اصلاح ندارد. احتمال انتخاب هر یک از کروموزوم‌ها به صورت حاصل تقسیم مقدار برازندگی آن تقسیم بر مقدار برازندگی کل کروموزوم‌های جمعیت محاسبه شده است.

گام یک: تولید جمعیت اول

در این مرحله، با استفاده از رویه انتخاب، کروموزوم‌ها انتخاب شده و فرزندان تولید شده در جدول (۳.۲) نمایش داده شده است. در این گام، بر اساس چرخ رولت، کروموزوم‌های ۱ و ۲ و هم‌چنین کروموزوم های ۲ و ۴ برای تولید مثل انتخاب شده‌اند. از تقاطع یک نقطه برش استفاده شده و ضمناً نقطه تقاطع بر اساس یک قاعده تصادفی انتخاب شده است. دو کروموزوم اول در نقطه ۶ تقاطع یافته و دو کروموزوم دوم در نقطه ۴ تقاطع پیدا کرده‌اند. بر اساس این رویه تقاطع ۴ کروموزوم فرزند (شماره ۵ تا ۸) تولید گردید.

شماره والد	آرایه والد	نقطه تقاطع	شماره فرزند	آرایه فرزند	وزن	امکان‌پذیری	F(x)
۱	[۱۰۰۱۱۰:۰۱۰۱]	۶	۵	[۱۰۰۱۱۰۱۰۰۰]	۱۶	Ok	۲۴۴
۲	[۱۱۰۱۰۰:۱۰۰۰]	۶	۶	[۱۱۰۱۰۰۰۱۰۱]	۲۱/۵	No	—
۳	[۱۱۰۱:۰۰۱۰۰۰]	۴	۷	[۱۱۰۱۰۱۱۰۱۰]	۲۵/۸	No	—
۴	[۰۱۰۰:۰۱۱۰۱۰]	۴	۸	[۰۱۰۰۰۰۱۰۰۰]	۹	Ok	۱۴۲
			۹	[۱۱۰۱۰۰۰۱۰۰]	۱۹/۵	Ok	۲۹۱
			۱۰	[۰۱۰۱۰۰۱۰۱۰]	۱۹	Ok	۳۰۴

جدول ۳.۲: فرزندان تولید شده در تکرار اول

با توجه به این‌که کروموزوم فرزند شماره ۶ و ۷ امکان‌پذیر نمی‌باشند، این کروموزوم‌ها به کروموزوم‌های

۹ و ۱۰ اصلاح گردیدند. در زیر نحوه اصلاح کروموزوم شماره ۶ نشان داده شده است.

$$\frac{\text{ارزش}}{\text{وزن}}(۱,۲,۴,۸,۱۰) = (\frac{۶۵}{۵}, \frac{۸۵}{۵/۵}, \frac{۹۸}{۶}, \frac{۴۳}{۳}, \frac{۲۱}{۲}) = (۱۳, ۱۵/۴۵, ۱۶/۳۳, ۱۴/۳۳, ۱۰/۵)$$

مطابق با محاسبات فوق رویه حذف اشیا از کروموزوم شماره ۶ به ترتیب ۱۰، ۱، ۸، ۲،۴ صورت می‌گیرد. بر این اساس ابتدا شیء شماره ۱۰ حذف می‌گردد و وزن کوله به ۱۹/۵ کیلوگرم کاهش می‌یابد. با توجه به این‌که جواب امکان‌پذیر شده است، شیء دیگری از کوله حذف نمی‌گردد و جواب اصلاح شده است. جدول (۴.۲) فرزندان تولید شده را بعد از عمل جهش نمایش می‌دهد. همان‌طور که در این جدول مشاهده می‌گردد، تنها فرزند اول و چهارم جهش پیدا کرده‌اند. در فرزند اول دو ژن و در فرزند چهارم یک ژن جهش یافته است. هر ژن به صورت تصادفی مستقل از سایر ژن‌ها شانس جهش خواهد داشت و این عمل بر اساس تولید عدد تصادفی انجام می‌گردد. برای هر ژن مستقل از سایر ژن‌ها، یک عدد تصادفی تولید و در صورتی که کمتر از عدد تصادفی جهش باشد، آن ژن جهش پیدا می‌کند.

از بین جمعیت فرزندان و جمعیت والدین، باید ۴ عضو انتخاب گردد. در این مثال، فرض می‌شود که بهترین و بدترین کروموزوم به صورت ثابت به جمعیت بعدی منتقل شده و ۲ کروموزوم دیگر به صورت تصادفی بر اساس مقدار برازندگی آن‌ها انتخاب می‌گردند. جمعیت جدید در جدول (۵.۲) نمایش داده شده است. رویه فوق تکرار می‌گردد تا در نهایت شرط توقف الگوریتم برآورده گردد.

شماره فرزند	آرایه فرزند	شماره ژن جهش یافته
۵	[۱۰۰۱۱۰۱۰۰۰]	۵ ، ۲
۸	[۰۱۰۰۰۰۱۰۰۰]	-
۹	[۱۱۰۱۰۰۰۱۰۰]	-
۱۰	[۰۱۰۱۰۰۱۰۱۰]	۳

شماره فرزند	آرایه فرزند	وزن	امکان‌پذیری	F(x)
۱۱	[۱۱۰۱۰۰۱۰۰۰]	۲۰	Ok	۳۰۵
۸	[۰۱۰۰۰۰۱۰۰۰]	۹	Ok	۱۴۲
۹	[۱۱۰۱۰۰۰۱۰۰]	۱۹/۵	Ok	۲۹۱
۱۲	[۰۱۱۱۰۰۱۰۱۰]	۲۱/۵	No	——
۱۳	[۰۱۰۱۰۰۱۰۱۰]	۱۹	Ok	۳۰۴

جدول ۴.۲: فرزندان تولید شده در تکرار اول بعد از عمل جهش

۶.۲.۲ معایب روش‌های کلاسیک

۱. وابستگی شدید این روش‌ها به نقطه شروع، که حتی انتخاب نامناسب آن در بعضی از حالات باعث شکست الگوریتم می‌شود.

شماره کروموزوم	آرایه	وزن	امکان‌پذیری	F(x)	احتمال
۲	[۱۱۰۱۰۰۱۰۰۰]	۲۰	Ok	۳۰۵	۰/۲۹۳
۸	[۰۱۰۰۰۰۱۰۰۰]	۹	Ok	۱۴۲	۰/۱۳۷
۱۳	[۰۱۱۱۰۰۱۰۱۰]	۱۹	Ok	۳۰۴	۰/۲۹۱
۹	[۱۱۰۱۰۰۰۱۰۰]	۱۹/۵	Ok	۲۹۱	۰/۲۷۹

جدول ۵.۲: جمعیت به دست آمده بعد از تکرار اول

۲. دچار بهینه‌های موضعی می‌شوند.

۳. معمولاً این الگوریتم‌ها در حل بعضی از مسائل کارا و در حل نوعی دیگر ناکارا عمل می‌کنند.

۴. این الگوریتم‌ها را معمولاً برای حل مسائل بهینه‌سازی گسسته نمی‌توان به کار برد.

۵. این الگوریتم‌ها را نمی‌توان با پیاده سازی روی ماشین‌های موازی، کاراتر کرد.

۷.۲.۲ مزایای الگوریتم ژنتیک

۱. الگوریتم ژنتیک در یک جمعیت از جواب‌ها و با مجموعه‌ای از آن‌ها شروع به جست‌وجو می‌کند. بدین ترتیب به جای یافتن نقطه مناسب، محدوده‌های مناسبی در فضای جست‌وجو شناسایی می گردد. در نتیجه هم‌زمان با ایجاد شانس بیشتر برای متغیرهای شایسته‌تر برای بقا و تولیدمثل، جست‌وجوی سایر مناطق فضای جست‌وجو با شایستگی کمتر نیز نادیده گرفته نمی شود. لذا احتمال یافتن نقطه بهینه سراسری افزایش می‌یابد. این ویژگی به خصوص برای توابع با تغییرات ناگهانی و دارای چندین نقطه بهینه موضعی مناسب می‌باشد. یکی از مهم‌ترین تفاوت‌های الگوریتم ژنتیک با الگوریتم‌های کلاسیک هم در این امتیاز است.

۲. در الگوریتم ژنتیک ما با تابع برازندگی سر و کار داریم که معمولاً از تابع هدف استفاده می‌کند بنابراین فقط اطلاعاتی درباره خود تابع هدف نیاز داریم نه مشتق وغیره درباره آن. به همین دلیل به راحتی می‌توان از این روش در بهینه‌سازی توابع گسسته و یا توابع مشتق‌ناپذیر و یا توابع با تغییرات ناگهانی و چندین جواب بهینه موضعی استفاده کرد.

۳. استفاده از قواعد احتمالی و ابتکاری به جای قواعد قطعی یکی دیگر از امتیازات الگوریتم ژنتیک است. این امتیاز سبب دوری از بهینه‌های موضعی می‌شود. البته استفاده الگوریتم ژنتیک از قواعد احتمالی به معنای یک جست وجو صرفاً تصادفی نیست، بلکه این الگوریتم از انتخاب تصادفی به عنوان ابزاری

برای هدایت عمل جست وجو استفاده می‌کند. توجه کنید که الگوریتم ژنتیک فقط به نسل قبل وابسته است، پس می توان آن را نوعی فرآیند مارکف دانست.

۴. الگوریتم ژنتیک دارای انعطاف بسیار بالایی است، به این معنا که با تعریف مناسب عملگرهای ژنتیکی این الگوریتم را روی هر مساله (مقید یا نامقید) و با هر فضای جواب (گسسته، پیوسته، مرکب، نامحدب) می توان پیاده‌سازی کرد.

۵. بدون توجه به دامنه یک مساله خاص، الگوریتم ژنتیک جست‌وجوی خود را به کمک عملیات فوق‌العاده ساده انجام داده، بسیار ساده و قابل درک است.

۶. در این روش محاسبات به طور دقیق انجام شده و هیچ‌گونه تقریبی نظیر خطی‌سازی مساله، گرد کردن نتایج و تغییر متغیرهای گسسته به پیوسته و بالعکس وجود ندارد.

۷. برتری دیگر الگوریتم ژنتیک، امکان پیاده‌سازی این الگوریتم روی ماشین‌های موازی است که باعث سرعت حل مسائل با پیشرفت علم کامپیوتر شده است.
توجه به این نکته ضروری است که روش‌های کلاسیک اگر چه در تمامی انواع مسائل قابل استفاده نیستند، اما در مورد مسائل خاص، به‌خوبی عمل می‌کنند، مثل عملکرد مطلوب روش گرادیان مزدوج در حل مسائل درجه دوم. نکته دیگر این است که با توجه به ماهیت ابتکاری الگوریتم ژنتیک نمی‌توان تضمینی برای به دست آوردن جواب بهینه داشت.

فصل ۳

الگوریتم جست‌وجوی ممنوع

۱.۳ مقدمه

جست‌وجوی ممنوع اولین بار توسط فرد گلوور[۱] در مقاله‌ای که در سال ۱۹۸۶ منتشر گردید[۱۹]، ارایه شد. همچنین بعداً دو مقاله توسط ایشان با نام ساده "جست‌وجوی ممنوع" در سال‌های ۱۹۸۹ و ۱۹۹۰ منتشر گردید که در آن‌ها، خیلی از کاربردهای جست‌وجوی ممنوع را معرفی کرد. البته شهرت این تکنیک به بعضی از محققین همچون دکتر ورا[۲] و همکارانش از موسسه فنی سوئیس[۳][۲۴] نیز که در معرفی این تکنیک نقش بالایی داشتند، برمی‌گردد. در طول این سال‌ها، این تکنیک جایگاه خاصی برای خود به وجود آورده است و هنوز هم به عنوان یک تکنیک قوی در حل خیلی از مسائل مورد استفاده قرار می‌گیرد. در همان سال‌ها، رقابتی بین جست‌وجوی ممنوع و روش گرم و سرد کردن شبیه‌سازی شده به وجود آمد. در بسیاری از موارد، الگوریتم‌های ابتکاری مبتنی بر جست‌وجوی ممنوع نتایج بهتری را به دست آوردند که این خود منجر به افزایش شهرت جست‌وجوی ممنوع گردید.

در طول سال‌های بعدی، توسعه‌های زیادی در جزئیات این تکنیک صورت پذیرفت و اولین کتاب به صورت یک فصل، توسط گلوور در سال ۱۹۹۳ منتشر گردید. این فصل کتاب تکنیک جست‌وجوی ممنوع را به صورت پیشرفته و آن چیزی که امروز ما می‌بینیم، شرح نداده بود، بلکه مفاهیم پایه و اولیه این تکنیک را به خوبی بیان کرده بود. شکل کامل شده این تکنیک را نیز مثل سایر تکنیک‌های فراابتکاری، نمی‌توان به شخص خاصی ارجاع داد و شکل توسعه یافته آن، نتیجه تلاش محققین مختلفی است.

اساس نام‌گذاری این روش، استفاده آن از لیستی به نام لیست ممنوع[۴] می‌باشد. این لیست برای جلوگیری

---

[۱] Fred glover
[۲] Werra
[۳] Swiss federal institute of technology
[۴] Tabu list

از گیر افتادن الگوریتم در بهینه محلی طراحی شده است. روش جست‌وجوی ممنوع مورد استفاده در حل مسائل بهینه‌سازی، به الگوریتم جست‌وجوی ممنوع نیز مشهور است.
این الگوریتم از یک نقطه یا جواب شروع کرده و در اطراف آن نقطه، به جست‌وجوی همسایگی می‌پردازد. در بین همسایه ها، بهترین را انتخاب و به آن نقطه حرکت می‌کند. این جست‌وجو را تا زمانی ادامه می‌دهد که یک معیار توقف برآورده گردد. در پایان جست‌وجو، نقطه بهینه گزارش می‌گردد. این الگوریتم فراابتکاری برای جلوگیری از افتادن در بهینه محلی، از یک حافظه کوتاه مدت استفاده می نماید. وظیفه این حافظه کوتاه مدت نگهداری از آخرین حرکت‌ها می‌باشد (با تعداد محدود و تعریف شده). این حرکت‌ها در یک لیست (لیست ممنوع) نگهداری می شوند. در هر حرکت، در صورتی که حرکت در لیست ممنوع قرار گرفته باشد، از آن انتقال جلوگیری می‌گردد مگر آن‌که حرکت ممنوع دارای شرایط خاصی باشد.

۲.۳ اصول کلی الگوریتم جست‌وجوی ممنوع

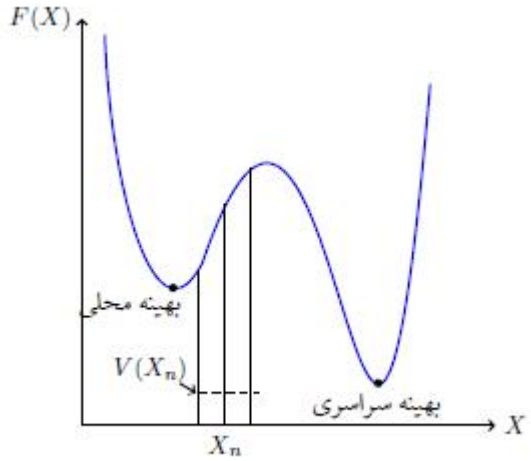
الگوریتم ابتکاری[۵]TS را می‌توان به عنوان یک استراتژی جست‌وجوی محلی در نظر گرفت. این جست‌وجو شامل حرکت از یک جواب (نقطه) به جواب دیگری در همسایگی او مطابق با بعضی از قواعد تعریف شده می‌باشد. برای مثال، مساله حداقل‌سازی یک تابع $F(x)$ روی یک مجموعه محدود از نقاط X را به عنوان یک حالت عمومی از یک مساله بهینه‌سازی ترکیبی در نظر بگیرید. روش جست‌وجوی ممنوع از یک جواب اختیاری $x_1 \in X$ شروع کرده و در هر مرحله n، به جست‌وجوی همسایگی می‌پردازد. در این جست‌وجوی همسایگی، به هر$x \in X$ یک زیرمجموعه $V(x) \subset X$ اختصاص داده می‌شود که همسایه x نامیده می‌شود. در مرحله n، یک جواب جدید x_{n+1} در همسایگی $V(x_n)$ از جواب جاری x_n را انتخاب می‌نماید. معمولاً این جواب بهترین جواب در بین همسایه‌ها بوده و دارای شرط زیر می‌باشد:

$$F(x_{n+1}) \leq F(x) \qquad \forall x \in V(x_n)$$

الگوریتم، بعد از انتخاب جواب x_{n+1}، از نقطه x_n به x_{n+1} حرکت می‌کند و آن‌گاه جواب x_{n+1} به عنوان جواب جاری بعدی شناخته می‌شود. این جست‌وجو تا زمانی ادامه پیدا می‌کند که یک شرط توقف تعریف شده برآورده شود.
مشکلی که ممکن است به هنگام حرکت از یک نقطه به نقطه دیگر ایجاد شود، مشکلی است که به نام افتادن در بهینه محلی معروف است. برای شرح این مشکل شکل (۱.۳) را در نظر بگیرید:

همان‌طور که در این شکل دیده می‌شود، فضای جواب مساله دارای دو نقطه می‌نیمم یا بهینه می‌باشد. نقطه می‌نیمم یا بهینه محلی، به نقطه‌ای گفته می‌شود که آن نقطه بهترین نقطه در بین تمامی نقاط اطراف خود

[۵]Tabu search

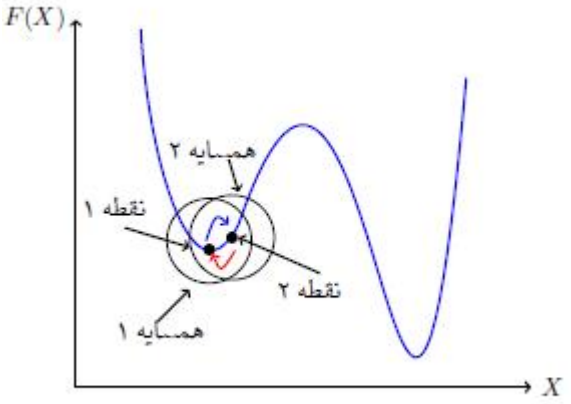


شکل ۱.۳: بهینه محلی و بهینه سراسری در فضای جواب مساله بهینه‌سازی

باشد.به عبارت دیگر، نقطه‌ای می‌نیمم محلی است که تمامی جواب‌های اطراف آن بدتر از خود آن باشند. اما نقطه بهینه سراسری یک ناحیه از تمام نقاط موجود در ناحیه بهتر است. در این شکل، یک نقطه بهینه محلی و یک نقطه بهینه سراسری نمایش داده شده است.

مشکل بهینگی، زمانی اتفاق می‌افتد که الگوریتم به نقطه بهینه محلی می‌رسد. در این نقطه، الگوریتم به جست‌وجوی همسایگی پرداخته و یک همسایه را انتخاب می‌نماید. لازم به توضیح است که الگوریتم در جست‌وجوی همسایگی، حتماً باید یک نقطه را از بین همسایگان و غیر از خود، انتخاب کند و به آن نقطه حرکت نماید، حتی اگر کیفیت آن نقطه پایین‌تر باشد. زمانی که الگوریتم در بهینه محلی قرار دارد، نقاط همسایه او دارای مقدار تابع (کیفیت) بدتر از خود نقطه بهینه محلی می‌باشند. از روی اجبار، الگوریتم به بهترین نقطه بین آن‌ها حرکت می‌کند. بعد از حرکت به آن نقطه، دوباره به جست‌وجوی همسایگی می‌پردازد و با توجه به این‌که نقطه بهینه محلی قبلی در همسایگی نقطه جدید قرار دارد، مجدداً به بهینه محلی برمی‌گردد. در این حالت الگوریتم در یک دور افتاده و از آن خارج نمی‌گردد.این وضعیت به افتادن در بهینه محلی معروف می‌باشد که در شکل (۲.۳) نیز نمایش داده شده است. در این شکل بهینه محلی با نقطه ۱ نمایش داده شده و همسایه انتخاب شده برای حرکت، نقطه ۲ می‌باشد. بعد از حرکت به نقطه ۲، بهترین نقطه در همسایگی آن، نقطه ۱ است و به آن برمی‌گردد. این روند همواره تکرار شده و الگوریتم از این دور خارج نمی‌گردد.

الگوریتم جست‌وجوی ممنوع برای جلوگیری از این مشکل از ابزار لیست ممنوع استفاده می‌کند. در هر مرحله، در هنگام حرکت از یک نقطه به نقطه دیگر، مشخصاتی از حرکت (مثلاً فاصله و جهت) را



شکل ۲.۳: مشکل افتادن در بهینه محلی

به حافظه سپرده و در نقطه جدید از انجام حرکتی که منجر به برگشت به عقب گردد، جلوگیری می‌کند. به منظور کارایی بیشتر، به جای یک حرکت قبلی، مجموعه‌ای از حرکت‌های قبلی را به حافظه سپرده و آن‌ها در لیستی به نام لیست ممنوع ذخیره می‌نماید. این لیست پویا بوده و در طول اجرای الگوریتم بهنگام می‌گردد. به عبارت دیگر، در هر حرکت، مشخصه حرکت جدید وارد لیست شده و مشخصه حرکت‌های قدیمی از لیست حذف می‌گردد. یک قاعده ساده این است که لیست طول محدودی داشته و زمان ورود یک حرکت به لیست، قدیمی‌ترین حرکت از آن خارج گردد.

الگوریتم در هر مرحله، بهترین نقطه را در بین نقاط همسایه جست‌وجو کرده و در صورتی به آن نقطه حرکت می‌کند که آن حرکت ممنوع نباشد. به عبارت دیگر، حرکت مد نظر را با لیست ممنوع مقایسه نموده و در صورتی که حرکت ممنوع باشد، از آن حرکت صرف‌نظر نموده و بهترین حرکت بعدی را در بین همسایگان انتخاب می‌نماید. نقطه بعدی انتخاب شده نیز با لیست ممنوع مقایسه می‌گردد.

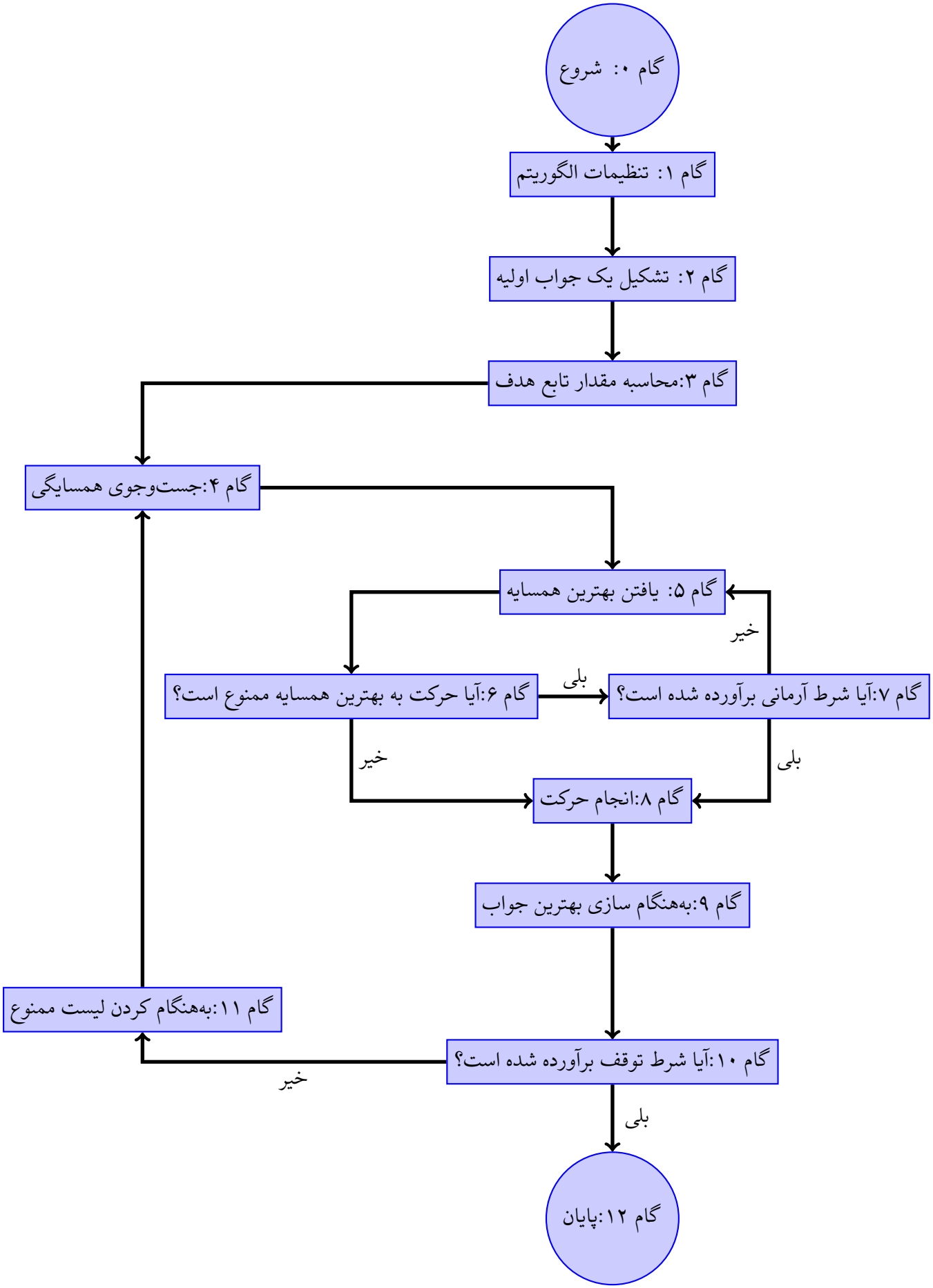
لیست ممنوع با وجود مفید بودن، محدودیت‌هایی را برای الگوریتم به وجود می‌آورد. در طول جست‌وجوی الگوریتم، ممکن است یک حرکت ممنوع باشد، ولی انجام حرکت با وجود ممنوع بودن تاثیر بالایی را در بهبود جهت حرکت الگوریتم داشته باشد. برای این منظور، الگوریتم از یک معیار به نام معیار آرمانی[۶] برای رهایی از این محدودیت در مواقع لزوم استفاده می‌نماید. یک معیار آرمانی ساده می‌تواند به این صورت تعریف گردد که ممنوعیت حرکتی که موجب رسیدن به نقطه‌ای گردد که از تمامی نقاطی که تا کنون به دست آمده است بهتر باشد، در نظر گرفته نمی شود.

[۶] Aspiration criterion

قاعده توقف پایان الگوریتم جست‌وجو را تعیین می‌کند. یک قاعده ساده می‌تواند به صورت محدودیتی برای کل تعداد حرکت‌ها باشد. به عبارت دیگر، وقتی تعداد حرکت‌ها به عدد خاصی رسید الگوریتم متوقف می‌گردد.

۳.۳ الگوریتم جست‌وجوی ممنوع

همان‌طور که قبلاً شرح داده شد، این الگوریتم از یک نقطه یا جواب شروع کرده و در اطراف آن نقطه به جست‌وجوی همسایگی می‌پردازد. در بین همسایه‌ها، بهترین را انتخاب کرده و به آن نقطه حرکت می‌کند. این جست‌وجو را تا زمانی ادامه می‌دهد که یک معیار توقف برآورده گردد. در پایان جست‌وجو، نقطه بهینه گزارش می‌گردد. الگوریتم برای جلوگیری از افتادن در بهینگی از یک حافظه کوتاه مدت استفاده می‌کند. وظیفه این حافظه کوتاه مدت، نگهداری از آخرین حرکت‌ها (با تعداد محدود و تعریف شده) می‌باشد. این حرکت‌ها در یک لیست (لیست ممنوع) نگهداری می‌شوند. در هر حرکت، در صورتی که حرکت در لیست ممنوع قرار گرفته باشد، از آن حرکت جلوگیری می‌گردد مگر آن‌که حرکت ممنوع دارای شرط آرمانی باشد. شکل زیر فلوچارت الگوریتم جست‌وجوی ممنوع را نشان می‌دهد. مراحل این فلوچارت در زیر شرح داده شده است.


گام ۰: شروع
گام ۱: تنظیمات الگوریتم
گام ۲: تشکیل یک جواب اولیه
گام ۳:محاسبه مقدار تابع هدف
گام ۴:جست‌وجوی همسایگی
گام ۵: یافتن بهترین همسایه
خیر
گام ۶:آیا حرکت به بهترین همسایه ممنوع است؟
بلی
گام ۷:آیا شرط آرمانی برآورده شده است؟
خیر
بلی
گام ۸:انجام حرکت
گام ۹:به‌هنگام سازی بهترین جواب
گام ۱۰:آیا شرط توقف برآورده شده است؟
خیر
گام ۱۱:به‌هنگام کردن لیست ممنوع
بلی
گام ۱۲:پایان

گام اول: تنظیمات الگوریتم

ساختار ساده الگوریتم جست‌وجوی ممنوع، در هنگام استفاده برای حل یک مساله بهینه‌سازی، نیاز به تنظیمات خاصی دارد. در صورتی که این تنظیمات به خوبی انجام گردد، الگوریتم دارای کیفیت بالایی خواهد بود و در صورتی که این تنظیمات به خوبی صورت نگیرد، کیفیت الگوریتم به شدت کاهش پیدا می‌کند. یکی از این تنظیمات، نحوه کد کردن جواب می‌باشد. کد، باید به نوعی طراحی گردد که قابل برنامه‌نویسی با کامپیوتر بوده و ضمناً در صورت استفاده در الگوریتم، کارایی خوبی هم داشته باشد. همچنین، یک خاصیت مهم کد، رابطه آن با فضای جواب می‌باشد. ساختار کد مناسب، باید تنها جواب‌های امکان‌پذیر را معرفی کرده و از طرفی رابطه یک به یک با فضای جواب داشته باشد. یکی دیگر از تنظیماتی که باید انجام گردد، تعریف ساختار همسایگی است. ساختار همسایگی، نحوه به دست آوردن همسایه‌ها و تعداد آن‌ها را برای هر جواب مشخص می‌نماید. ساختار همسایگی باید به نحوی طراحی گردد که فضای خیلی کوچکی را نسبت به کل فضا معرفی نموده و از طرفی مجموعه معرفی شده خصوصیات نزدیکی به نقطه‌ای که همسایگی برای آن به دست آمده است داشته باشد.

نوع و نحوه طراحی لیست ممنوع نیز یکی از پارامترهای مهم الگوریتم جست‌وجوی ممنوع است. در لیست ممنوع، مشخصه حرکت ذخیره می‌گردد. نحوه طراحی این مشخصه، دوره ممنوع بودن یا طول لیست ممنوع و نحوه به‌هنگام سازی لیست ممنوع نیز تاثیر بالایی در کیفیت الگوریتم خواهد داشت.

گام دوم: تشکیل یک جواب اولیه

الگوریتم جست‌وجوی ممنوع با یک جواب اولیه شروع می‌شود. نحوه ایجاد جواب اولیه باید از ابتدا مشخص باشد. در خیلی از مواقع، جواب اولیه به صورت تصادفی انتخاب می‌شود. در صورتی که ساختار کد جواب ساده باشد، این کار به سادگی انجام می‌گردد، اما در خیلی از مسائل به دلیل پیچیدگی فضای جواب و نحوه کدگذاری آن، ایجاد یک جواب اولیه امکان‌پذیر به صورت تصادفی کار ساده‌ای نیست. به همین دلیل، الگوریتم با مراحلی برای ایجاد جواب اولیه معرفی می‌گردد. در بعضی از مواقع نیز، به منظور افزایش کیفیت الگوریتم، الگوریتم با مراحلی طراحی می‌گردد تا منجر به ایجاد یک جواب اولیه با کیفیت نسبتاً خوبی گردد. این کار موجب کمک به الگوریتم اصلی در رسیدن به جواب‌های بهتر می‌گردد.

گام سوم: محاسبه مقدار تابع هدف

در این مرحله، مقدار تابع هدف مساله برای جواب اولیه در نظر گرفته شده، محاسبه می‌گردد. در بعضی از مواقع نیز، بعد از محاسبه تابع هدف، تابعی از تابع هدف به عنوان تابع برازندگی در نظر گرفته می‌شود. در مجموع، در این قدم، مقدار تابع برازندگی برای جواب اولیه محاسبه می‌گردد. در شروع الگوریتم مقدار به دست آمده از جواب اولیه به عنوان مقدار بهینه نیز در نظر گرفته می‌شود. این مقدار در طول الگوریتم به‌هنگام می‌گردد. همچنین، نقطه اولیه به عنوان جواب جاری نیز مد نظر قرار می‌گیرد.

گام چهارم: جست‌وجوی همسایگی

در هر مرحله الگوریتم، جست‌وجوی همسایگی به جست وجوی همسایه‌های جواب جاری می‌پردازد. در این مرحله، با توجه به ساختار معرفی شده برای همسایگی، کلیه همسایگی‌های جواب جاری مورد بررسی قرار می‌گیرند (مقدار تابع برازندگی آن‌ها محاسبه می‌گردد).

گام پنجم: یافتن بهترین همسایه

در جست وجوی همسایگی، کلیه همسایه‌ها بررسی می‌گردد. در بین همسایه‌ها (غیر از خود جواب جاری) بهترین همسایه انتخاب می‌گردد.

گام ششم: ارزیابی ممنوعیت حرکت

بعد از انتخاب بهترین همسایه، حرکت به آن جواب با لیست ممنوع مقایسه می‌گردد. در صورتی که این حرکت جزو لیست ممنوع باشد، به قدم هفتم رفته و در صورتی که حرکت ممنوع نباشد، الگوریتم به قدم هشتم خواهد رفت.

گام هفتم: معیار آرمانی

در این قدم، حرکت ممنوع شده با معیار آرمانی مقایسه می‌گردد. در صورتی که معیار آرمانی برآورده شده باشد، حرکت انجام شده و در صورتی که معیار آرمانی برآورده نشود، الگوریتم به قدم پنجم برمی‌گردد و بهترین جواب بعدی را انتخاب می‌کند.

گام هشتم: انجام حرکت

در این قدم، جواب جاری جدید به عنوان جواب جاری در نظر گرفته می شود. به عبارت دیگر، از جواب جاری به جواب جدید حرکت می‌شود و جواب جدید به عنوان جواب جاری شناخته می شود.

گام نهم: به‌هنگام سازی بهترین جواب

در طول الگوریتم، همواره بهترین جواب ذخیره می‌گردد. در ابتدا اولین نقطه، به عنوان بهترین جواب در نظر گرفته می‌شود و هر زمان که حرکتی انجام می‌گردد، جواب جدید با بهترین جوابی که تا آن لحظه به دست آمده است مقایسه می‌گردد، در صورتی که جواب جدید جواب بهتری باشد، جواب جدید به عنوان بهترین جواب ذخیره می‌گردد و در غیر این‌صورت، بهترین جواب تغییری نمی‌کند.

گام دهم: شرط توقف

بعد از هر حرکت، شرط توقف مورد ارزیابی قرار می‌گیرد. در صورتی که شرط توقف برآورده شود، الگوریتم پایان یافته و نتایج آن گزارش می‌شود و در غیر این‌صورت به قدم یازدهم منتقل خواهد شد.

گام یازدهم: به‌هنگام سازی لیست ممنوع

بعد از هر حرکت، مشخصه حرکت جدید وارد لیست ممنوع می‌گردد و سپس لیست ممنوع به‌هنگام می‌گردد. مدت زمان ماندن در لیست ممنوع و نحوه به‌هنگام سازی آن از ابتدا باید تعریف گردد. بر اساس این قاعده،

در زمان انتقال یک حرکت به لیست ممنوع، ممکن است یک یا چند حرکت قدیمی از لیست ممنوع حذف گردد. بعد از به‌هنگام سازی لیست ممنوع، الگوریتم به قدم چهارم رفته و به جست‌وجوی همسایگی در اطراف جواب جدید می‌پردازد.

به منظور تشریح بیشتر الگوریتم جست‌وجوی ممنوع، در زیر ساختار شبه کد این الگوریتم آورده شده است. قبل از بررسی الگوریتم، ابتدا اصطلاحات مورد استفاده در این الگوریتم را معرفی می‌کنیم. اصطلاحات مورد استفاده عبارتند از:

x_1	جواب اولیه
x_n	جواب جاری
x^*	جواب بهینه
$\bar{x}$	بهترین جواب در همسایگی x_n
$F(x_1)$	مقدار تابع هدف برای جواب اولیه
F^*	مقدار تابع هدف برای نقطه بهینه
$\bar{F}$	مقدار تابع هدف مربوط به $\bar{x}$

جدول ۱.۳: اصطلاحات مورد استفاده در الگوریتم جست‌وجوی ممنوع

بر پایه تعاریف فوق، الگوریتم ارائه شده در زیر ساختار کلی الگوریتم ابتکاری TS را نشان می‌دهد.

دستورالعمل جست‌وجوی ممنوع
قدم ۱
انتخاب یک جواب اولیه x_1 در X
قرار دهید: $F(x_1) \longrightarrow F^*$
$x_1 \longrightarrow x^*$
TL: (Tabu List) خالی است.
قدم ۲
تکرار $n = 1, 2, \ldots$ و x_n به عنوان جواب جاری در نظر گرفته شود.
$\bar{F}$ بهترین مقدار در دسترس از F از زیرمجموعه همسایگی $V(x_n)$ است.
در ابتدای هر مرحله، $\bar{F}$ را برابر ∞ قرار دهید.
برای تمامی xهای عضو $V(x_n)$:
اگر $F(x) < \bar{F}$ (اگر حرکت ($x \longrightarrow x_n$) ممنوع نباشد، یا اگر حرکت ممنوع باشد ولی سطح آرمانی را رعایت کند)
سپس $F(x) \longrightarrow \bar{F}$ و $x \longrightarrow \bar{x}$
قرار دهید: $\bar{x} \longrightarrow x_{n+1}$
اگر $\bar{F} < F^*$ سپس قرار دهید: $\bar{x} \longrightarrow x^*$ و $\bar{F} \longrightarrow F^*$
مشخصه مناسب از حرکت ($x_n \longrightarrow x_{n+1}$) وارد لیست ممنوع می‌گردد.
قبل از ورود یک مشخصه، کلیه مشخصه‌ها در لیست یک جایگاه به عقب برمی‌گردند و در صورتی که لیست پر باشد، آخرین مشخصه (قدیمی‌ترین مشخصه) از لیست خارج می‌گردد.
مشخصه ورودی در اولین جایگاه که خالی شده است، قرار می‌گیرد.
قدم ۳
اگر معیار توقف برآورده شد توقف کنید، در غیر این‌صورت مرحله ۲ را تکرار کنید.
تمام.

۴.۳ پارامترهای الگوریتم جست‌وجوی ممنوع

الگوریتم جست‌وجوی ممنوع مانند سایر الگوریتم‌های فراابتکاری، دارای پارامترهایی می‌باشد که طراحی مناسب آن‌ها، تاثیر بالایی روی کیفیت الگوریتم خواهد داشت. در زیر به بررسی بعضی از این پارامترها می‌پردازیم.

۱.۴.۳ کدگذاری

معمولاً، الگوریتم‌های فراابتکاری محاسبات زیادی را به همراه داشته و الزاماً بایستی با استفاده از کامپیوتر حل گردند. به منظور برنامه‌نویسی الگوریتم، بایستی مساله نیز به زبان کامپیوتری بیان گردد. کدگذاری موجب می‌گردد که یک جواب مساله که ممکن است به طرق مختلف بیان گردد، به صورت کدهای قابل فهم توسط کامپیوتر بیان گردد. در فصل دوم این پایان‌نامه، روش‌های مختلف کدگذاری و خصوصیات آن مورد بررسی قرار گرفته است و در این‌جا مانند آن‌چه که در فصل ۴ بیان خواهد شد از کدگذاری دودویی استفاده می‌شود.

۲.۴.۳ ساختار همسایگی

الگوریتم جست‌وجوی ممنوع بر پایه جست‌وجوی همسایگی استوار است. در این جست‌وجو، به هر $x \in X$ یک زیرمجموعه $V(x) \subset X$ اختصاص داده می‌شود که همسایه x نامیده می‌شود.

برای مثال، تخصیص درجه دو، یک جواب یا نقطه با یک ماتریس ترتیبی مشخص می‌گردد. همسایه این نقطه، می‌تواند از جابه‌جایی دو عضو این ماتریس به دست آید. سابقه جست‌وجوی محلی، شاید سابقه‌ای برابر عمر انسان داشته باشد. فرض کنید، شخصی با مشکلی مواجه است و جوابی برای آن پیدا نمی‌کند. در صورتی که افراد دیگری یک جواب به او پیشنهاد بدهند، معمولاً این شخص سعی می‌کند جواب دریافت شده را با کمی تغییر یا جست‌وجو بهبود بدهد. به عبارت دیگر، با دریافت یک جواب سعی می‌گردد با استفاده از تغییرات محلی نسبت به جواب اولیه، بررسی گردد که آیا امکان بهبود جواب وجود دارد یا خیر. در این بررسی، در صورت یافتن یک بهینه محلی، جست‌وجو پایان می‌یابد. شخص به هیچ وجه جواب به دست آمده را به عنوان بهترین جواب (بهینه سراسری) در نظر نمی‌گیرد و در صورت نیاز به بهبود مجدد جواب، سعی در تغییر حوزه بررسی (محله مورد جست‌وجو) می‌نماید. این رفتار بشر در حل مشکلات،چیزی شبیه به جست‌وجوی محلی و همسایگی می‌باشد.

الگوریتم‌های گرم و سرد کردن شبیه‌سازی شده و جست‌وجوی ممنوع جزء الگوریتم‌های جست‌وجوی محلی محسوب می‌گردند. الگوریتم گرم و سرد کردن شبیه‌سازی شده جست‌وجوی خود را بر اساس یک روند تقریباً تصادفی انجام می‌دهد، در حالی که الگوریتم جست‌وجوی ممنوع، بر اساس یک جست‌وجوی همسایگی عمل می‌نماید.

طراحی یک ساختار مناسب همسایگی، تاثیر بالایی بر کیفیت الگوریتم جست‌وجوی ممنوع دارد. ساختار همسایگی، نحوه به دست آوردن همسایه‌ها و تعداد آن‌ها را برای هر جواب مشخص می‌نماید. ساختار همسایگی باید به نحوی طراحی گردد که فضای کوچکی را نسبت به کل فضای جواب معرفی نموده و از طرفی، مجموعه معرفی شده خصوصیات نزدیکی به نقطه‌ای که قصد یافتن همسایه برای آن وجود دارد، داشته باشند. بعد از جست‌وجوی همسایگی، بهترین همسایه انتخاب و به آن نقطه حرکت می‌شود. ساختار همسایگی و حرکت باید دارای این خاصیت باشد که قابلیت جست وجو در کل فضا را داشته باشد. به عبارت دیگر، با استفاده از ساختار همسایگی و حرکت، هر نقطه‌ای از فضای جواب باید قابل دسترسی باشد و یا احتمال رسیدن به آن وجود داشته باشد. در صورتی که این شرط برقرار نباشد، ساختار همسایگی و حرکت، ناقص بوده و کیفیت الگوریتم را کاهش می‌دهد.

برای یک مساله، ساختارهای همسایگی متعددی می‌توان طراحی کرد. این ساختارها دارای تاثیر متفاوتی روی کیفیت الگوریتم می‌باشند. در هنگام تنظیم الگوریتم، باید با یک سری آزمایشات اولیه، بهترین ساختار همسایگی را انتخاب کرد. در این پایان‌نامه، به منظور مقایسه دو روش ترکیبی الگوریتم جست‌وجوی موضعی

ژنتیک و الگوریتم جست‌وجوی ممنوع ژنتیک که در فصل ۴ تعریف خواهد شد، همسایگی تعریف شده در زیر را مورد استفاده قرار خواهیم داد، یعنی؛

تعریف ۱.۴.۳. S را به عنوان یک جواب شدنی در نظر بگیرید، به‌طوری‌که $S \subset I$ و $|S| = p$، آن‌گاه مجموعه‌ی همسایگی‌های S برابر است با

$$N(S) = \{T | T \subset I, |T| = p, |S - T| = ۱\}.$$

۳.۴.۳ ارزیابی همسایه‌ها

در هنگام ایجاد یک الگوریتم جست‌وجوی همسایگی، نحوه ارزیابی همسایه‌ها نیز باید در نظر گرفته شود. یک راه ساده محاسبه تابع هدف همسایه (نقطه جدید) و مقایسه آن با میزان تابع هدف نقطه جاری می‌باشد. فرض کنید نقطه جاری دارای تابع برازندگی $f(x)$ بوده و نقطه همسایه ناشی از حرکت دارای تابع $f(m)$ باشد. در این صورت، مقدار اختلاف به صورت

$$\triangle f(x, m) = f(m) - f(x)$$

محاسبه می‌گردد.

محاسبه تابع هدف برای خیلی از مسائل زمان قابل توجهی را خواهد گرفت و در نتیجه، بهتر است راهی پیدا کرد که از محاسبات متعدد تابع هدف جلوگیری شود. در مقابل محاسبه تابع هدف (برازندگی) همسایه، راه دوم محاسبه تابع برازندگی همسایه از روی نقطه جاری و نوع حرکت منجر به ایجاد همسایه می‌باشد. به عبارت دیگر بهتر است مقدار $\triangle f(x, m)$ به صورت مستقیم و بدون محاسبه مقدار تابع هدف نقطه جدید محاسبه گردد. البته، ممکن است محاسبه $\triangle f(x, m)$ به طور مستقیم برای خیلی از مسائل از جمله مساله مسیریابی وسایل حمل و نقل سخت و پیچیده باشد.در این پایان‌نامه از تابع هدف برای ارزیابی همسایه‌ها استفاده شده است. به این صورت که اگر تابع هدف نقطه همسایه از تابع هدف نقطه جاری بیشتر بود این نقطه در لیست بهترین همسایه‌ها قرار می‌گیرد.

۴.۴.۳ لیست کاندیدا

به طور کلی، در یک الگوریتم جست‌وجوی محلی، ضروری نیست که تمامی همسایه‌های اطراف یک نقطه$V(x_n)$، در یک تکرار جست‌وجو گردد، بلکه می‌توان زیرمجموعه‌ای از آن را جست‌وجو نمود. در حقیقت، الگوریتم گرم و سرد کردن شبیه‌سازی شده، تنها یک نقطه از بین همسایه‌ها را انتخاب نموده و در مقابل، الگوریتم جست‌وجوی ممنوع سعی دارد انتخاب هوشمندانه‌ای از همسایه‌های خود را داشته باشد. لیست کاندیدا[۷]، زیرمجموعه‌ای از همسایه‌ها را برای جست‌وجو در یک تکرار مشخص می‌نماید.

[۷] Candidate list

این استراتژی موجب می‌گردد که زمان اجرای الگوریتم کاهش یابد. در این استراتژی، سعی می‌گردد با یک طراحی مؤثر، زیرمجموعه‌ای از حرکت‌های نویدبخش را در هر تکرار جست‌وجوی ممنوع، مشخص و آزمایش نمود.

زمانی که تعداد کل همسایه‌ها کم باشد و ارزیابی آن‌ها خیلی زمان‌بر نباشد، می‌توان تمامی همسایه‌ها را در لیست کاندیدا در نظر گرفت. در غیر این‌صورت، مجموعه‌ای از همسایه‌ها در لیست کاندیدا انتخاب می‌شوند تا بتوان مطلوب‌ترین حرکت را انجام داد. در مواقعی که $V(x_n)$ بزرگ است یا این‌که ارزیابی عناصر آن مشکل است، استراتژی لیست کاندیدا می‌تواند در محدود کردن تعداد جواب‌های آزمایش شده در یک تکرار ضروری باشد. به دلیل اهمیت انتخاب مدبرانه عناصر، به کارگیری قوانین کارا در انتخاب کاندیداها در فرایند جست‌وجو بسیار مهم است. استراتژی لیست کاندیدا می‌تواند به طور قابل توجهی روی کیفیت و سرعت الگوریتم، مؤثر باشد. از طرفی، استراتژی لیست کاندیدا می‌تواند به هدف جهت‌دهی به روند جست‌وجو که یکی از اهداف الگوریتم جست‌وجوی ممنوع نیز می‌باشد، کمک نماید. این هدف موجب می‌گردد که جست‌وجو به سمت بهتری جهت‌دهی شده و به جای یک جست‌وجوی کاملاً تصادفی، از جهت‌دهی به سمت فضاهای بهتر نیز استفاده جوید.

روش‌ها و استراتژی‌های متعددی برای طراحی لیست کاندیدا وجود دارد. یکی از این استراتژی‌ها نمونه‌گیری تصادفی می‌باشد. در این روش، به تصادف زیرمجموعه‌ای از همسایه‌ها انتخاب و تنها این زیرمجموعه مورد ارزیابی قرار می‌گیرد. روش دیگر، تبدیل مجموعه همسایگی به چند زیرمجموعه می‌باشد. در این حالت، مجموعه همسایه به چند زیرمجموعه مشخص تقسیم شده و در هر تکرار یکی از این زیرمجموعه‌ها مورد ارزیابی قرار می‌گیرد. در این حالت، با استفاده از یک سیکل، همه زیرمجموعه‌ها مورد استفاده قرار می‌گیرد ولی در هر تکرار تنها یکی از آن‌ها مورد استفاده قرار می‌گیرد. روش ایده‌آل در این استراتژی این است که زیرمجموعه‌ای از همسایه‌ها انتخاب گردند که دارای کیفیت بالاتری باشند. برای مسائل مختلف، روش‌هایی وجود دارد که زیرمجموعه‌ای از فضای همسایگی را معرفی می‌کند که احتمال وجود جواب‌های بهتر در آن‌ها بیشتر باشد.

۵.۴.۳ حافظه کوتاه مدت

یکی از مشکلات روش‌های جست‌وجوی محلی، تکرار جست‌وجوهای قبلی می‌باشد. به عبارت دیگر، ممکن است یک الگوریتم جست‌وجوی محلی، یک نقطه جواب را بارها جست‌وجو نماید. در این حالت، الگوریتم در طی روند جست‌وجوی خود، چندین بار از یک نقطه عبور می‌نماید. محاسبه مجدد یک نقطه کاری بیهوده بوده و موجب می‌گردد که بخشی از جست‌وجوی الگوریتم بیهوده شده و کیفیت الگوریتم کاهش یابد. از

طرفی، همان‌طور که قبلاً نیز شرح داده شد، این مشکل می‌تواند حتی منجر به افتادن الگوریتم در بهینه محلی شود.

یک راه برای جلوگیری از جست‌وجوی تکراری، ثبت کلیه جواب‌های جست‌وجو شده و مقایسه جواب جدید با جست‌وجوهای قبلی می‌باشد. با ادامه روند جست‌وجو، این لیست خیلی طولانی شده و نیاز به حافظه بزرگی دارد. از طرفی، مقایسه یک جواب جدید با لیست کلیه جست‌وجوهای قبلی، نیاز به زمان بسیار زیادی دارد و در مجموع این روش، روش بیهوده و زمان‌بری است. در این حالت، حجم حافظه مورد نیاز و زمان مورد نیاز برای مقایسه با جواب‌های قبلی، با تعداد جواب‌های گذر شده ارتباط خطی داشته و با افزایش تعداد جواب‌ها، این حجم و زمان مورد نیاز محاسبه نیز به شدت افزایش خواهد یافت.

می‌توان به جای کلیه نقاط عبوری، مجموعه‌ای از نقاطی را که به تازگی از آن‌ها گذر شده است، در حافظه سپرد و هر جواب جدید را با آن مقایسه نمود. این روش نیز کارا نیست، زیرا سپردن تعداد کمی نقطه (جواب) ممنوعیت بسیار کمی را برای جست‌وجوی همسایگی به وجود می‌آورد و امکان برگشت مجدد به بهینه محلی و جست‌وجوی نقاط قبلی زیاد است. در صورتی که طول لیست کوتاه باشد، به دلیل ممنوعیت کم، خیلی مؤثر نبوده و در صورتی که طول لیست زیاد باشد، به دلیل نیاز به حافظه بزرگ و از طرفی زمان طولانی مورد نیاز برای قیاس جواب جدید با جواب‌های قبلی، روشی ناکارآمد می‌باشد.

لیست ممنوع

همان‌طور که در بالا شرح داده شد، به کارگیری حافظه کوتاه مدت[۸] برای ذخیره‌سازی نقاط عبوری توسط الگوریتم کاری مفید نیست. مشکل افتادن الگوریتم در بهینه محلی کماکان وجود دارد. الگوریتم جست‌وجوی ممنوع از روش مفیدی برای حل این مشکل استفاده می‌نماید. این روش به جای ممنوعیت برای رسیدن به یک نقطه که قبلاً بررسی شده است، روی ممنوعیت برای خود حرکت تکیه دارد. این روش مجموعه‌ای از حرکت‌هایی را که به تازگی گذر شده است، به عنوان حرکت ممنوع در نظر می‌گیرد. این مجموعه اندازه خیلی کوچکی نسبت به ابعاد مساله یا آن‌چه در بخش قبلی ارائه شد، دارد. این اندازه کوچک موجب می‌گردد مشکل استفاده از حافظه و هم‌چنین مشکل زمان لازم برای قیاس یک حرکت با لیست ممنوع برطرف گردد.

در الگوریتم جست‌وجوی ممنوع، خصوصیت اتصال[۹]، یک فرض پذیرفته شده است. این فرض، به این معنی است که از هر نقطه فضای جواب، با استفاده از مجموعه‌ای از حرکت‌ها می‌توان به جواب بهینه رسید. به منظور سادگی، فرض می‌شود که خصوصیت معکوس‌پذیری[۱۰] نیز برای حرکت‌ها در مجموعه جواب وجود دارد. طبق این خصوصیت، اگر فرض شود حرکت m موجب می‌گردد که از نقطه x به نقطه $(x \oplus m)$ حرکت

[۸] Short-term memory
[۹] Connectivity
[۱۰] Reversibility

شود، در آن صورت رابطه $(x \oplus m) \oplus m^{-1} = x$ برقرار خواهد بود. به عبارت دیگر، اگر روی یک نقطه x_n یک حرکت انجام شود و سپس روی نقطه به دست آمده (x_{n+1})، عکس حرکت قبلی انجام گردد، به نقطه اولیه (x_n) خواهد رسید. لذا می‌توان پذیرفت که به عنوان یک محدودیت، حرکت m^{-1} بایستی برای کلیه نقاط، ممنوع باشد. این ممنوعیت موجب می‌گردد در هر نقطه‌ای از فضای جواب و در هر تکرار، از برگشت به نقطه قبلی آن جلوگیری گردد. این ممنوعیت موجب می‌گردد که در اولین نقطه خروجی از بهینه محلی، از برگشت به بهینه محلی در حرکت بعدی جلوگیری شود.

وجود یک حرکت ممنوع برای خروج از بهینه محلی کافی نیست؛ زیرا ممکن است با استفاده از یک سری حرکت‌های ترکیبی، بعد از چند حرکت، مجدداً الگوریتم به بهینه محلی برگردد. برای جلوگیری از این مشکل، الگوریتم جست‌وجوی ممنوع مجموعه‌ای از حرکت‌های قبلی را ممنوع می‌نماید. به عبارت دیگر، حرکت m^{-1} نه تنها برای نقطه بعدی بلکه تا چندین نقطه و حرکت بعدی ممنوع باقی می‌ماند. حرکت‌های ممنوع برای دوره محدودی ممنوع می‌مانند. این عمل موجب می‌گردد که احتمال برگشت به جواب‌های بررسی شده خیلی کم گردد یا غیر محتمل باشد. حتی اگر این حالت اتفاق بیافتد (الگوریتم به نقاط بررسی شده برگردد)، با توجه به این‌که لیست ممنوع (یا حرکت‌های ممنوع) تغییرپیدا کرده است، الگوریتم مسیر دیگری را نسبت به مسیر قبلی خود در پیش می‌گیرد و دچار سیکل نمی‌شود. دوره ممنوعیت یک حرکت باید محدود و کم باشد و در مجموع باید کسر کوچکی از حرکت‌های ممکن در یک نقطه، ممنوع باشند. در نتیجه، مجموعه حرکت‌های ممنوع یا لیست ممنوع، یک حافظه کوتاه مدت را در الگوریتم ایجاد می‌نمایند.

در بالا، برای فهم ساده‌تر حرکت ممنوع، معکوس یک حرکت به عنوان حرکت ممنوع در نظر گرفته شد. اما ممکن است تعیین معکوس یک حرکت در خیلی از مسائل، مشکل یا مبهم باشد. برای تشریح بیشتر به مثال زیر توجه کنید.

فرض کنید، مساله عبارت از یافتن بهترین ترتیب از n جزء باشد. یک حرکت منطقی در این مساله، می‌تواند به صورت جابه‌جایی دو عضو i و j در این ترتیب تعریف شود $(1 \le i \le j \le n)$. هم‌چنین، فرض کنید حرکت جابه‌جایی دو عضو i و j را با نماد حرکت (i, j) معرفی شود. لازم به ذکر است که معکوس حرکت (i, j) نیز همان خود حرکت (i, j) خواهد بود. در این حالت، ممکن است مجموعه‌ای از حرکت‌های غیر ممنوع انجام شوند و با وجود این‌که ممنوع نیستند، یک دور را ایجاد نمایند. به عبارت دیگر، با وجود حرکت‌های ممنوع، ممکن است، از پدیده دور جلوگیری نشود. برای مثال، مجموعه حرکت‌های زیر را در نظر بگیرید:

$$(i, j) \to (k, p) \to (i, p) \to (k, j) \to (k, i) \to (j, p)$$

که با وجود عدم ممنوعیت، یک دور را به وجود می‌آورند.

یک راه برای جلوگیری از ایجاد سیکل، جلوگیری از بازگشت هم‌زمان دو نقطه جابه‌جا شده به جاهای قبلی

خود می‌باشد. در این حالت اگر شی j در جایگاه p_j و شی i در جایگاه p_i و جابه‌جایی اشیاء در محل‌های (i,j) صورت پذیرد، قاعده ممنوعیت می‌تواند به صورت جلوگیری از بازگشت شی j به جایگاه p_j و شی i به جایگاه p_i به طور هم‌زمان (هم‌زمان در جایگاه‌های قبلی خود نباشند) تا پایان دوره ممنوعیت، تعریف گردد. این تعریف، کارایی بیشتری در جلوگیری از سیکل دارد. به عبارت دیگر، این تعریف موجب می‌گردد که از هر گونه سیکلی که طول آن کوتاه‌تر از دوره ممنوعیت باشد، جلوگیری شود.

دوره شرط ممنوع

شرط ممنوع برای جلوگیری از بازگشت به عقب الگوریتم و گیر افتادن در بهینه‌های محلی به کار گرفته می‌شود. دوره شرط ممنوع به دوره‌ای گفته می‌شود که یک حرکت یا مشخصه‌ای از آن حرکت ممنوع می‌ماند. در صورتی که همه حرکت‌های ممنوع دارای دوره ممنوعیت مساوی باشند، معمولاً از واژه طول لیست ممنوع نیز برای معرفی دوره شرط استفاده می‌گردد.

در صورتی که دوره شروط ممنوع یا عمر آن‌ها کوتاه باشد، احتمال افتادن در بهینه محلی افزایش می‌یابد. هر چه‌قدر این دوره کوتاه‌تر باشد، به همان نسبت وضعیت الگوریتم به حالت بدون لیست ممنوع نزدیک‌تر می‌گردد. به عبارت دیگر، اگر دوره شروط ممنوع خیلی کوتاه باشد، الگوریتم نقاط کمی را جست‌وجو کرده و در اولین بهینه محلی گیر خواهد کرد. اگر این دوره بزرگتر گردد، احتمال مشاهده نقاط بیشتر توسط الگوریتم افزایش می‌یابد و احتمال پیدا کردن نقاط با کیفیت بیشتر افزایش می‌یابد. با این وجود، دوره شروط ممنوع نباید خیلی بزرگ باشد. در صورتی که دوره شروط ممنوع خیلی بزرگ باشد، احتمال افتادن در بهینه محلی، به علت محدودیت زیاد در حرکت و حرکت‌های کم قابل انجام، زیاد می‌شود. در این حالت، الگوریتم محدوده زیادی را جست‌وجو نخواهد کرد.

یک ایده در ارتباط با طول لیست ممنوع، استفاده از یک ساختار تصادفی برای تعیین طول دوره ممنوع می‌باشد. این ایده موجب می‌گردد که هم از مزایای کوچک بودن طول لیست ممنوع که منجر به جست‌وجوی محلی عمیق‌تر و یافتن جواب‌های بهتر در یک حوزه محلی شده استفاده گردد و هم از مزایای بزرگ بودن طول لیست ممنوع که منجر به جلوگیری از افتادن در بهینه‌های محلی شود، استفاده گردد. بدین منظور می‌توان در طول روند جست‌وجو، مدت زمان ماندن در لیست ممنوع را برای یک حرکت، به طور تصادفی تعیین کرد. هم‌چنین، می‌توان برای این طول دوره ممنوع تصادفی، یک حد پایین و حد بالا نیز انتخاب کرد. این حدود را می‌توان به سادگی تعیین کرد و هم‌چنین می‌توان اندازه‌های آن را بر اساس مشخصه‌هایی از جواب‌ها تغییر داد. نشان داده شده است که این استراتژی نتایج بهتری نسبت به استراتژی طول ثابت لیست ممنوع دارد.

معیار آرمانی

در بعضی مواقع، اجرای بعضی از شروط ممنوع بیهوده وحتی زیان‌آور به نظر می‌رسند. برای مثال، ممکن است یک حرکت که ممنوع باشد، منجر به رسیدن به جوابی شود که از بهترین جوابی که در طول جست‌وجو تا آن نقطه پیدا شده است، بهتر باشد. هیچ دلیل منطقی را نمی‌توان برای جلوگیری از این حرکت پیدا کرد. برای این‌که چنین حرکت‌هایی از دست داده نشوند، بایستی ساختاری برای نادیده گرفتن بعضی از ممنوعیت‌ها در شرایط خاص در نظر گرفته شود. به همین دلیل، در الگوریتم جست‌وجوی ممنوع شرایطی تعیین می‌گردد که اگر یک حرکت ممنوع آن شرایط را برآورده کرد، قابل انجام باشد. در الگوریتم جست‌وجوی ممنوع، این شرایط به نام معیارهای آرمانی شناخته می‌شوند.

در بسیاری از موارد، شرط بهبود بهترین جوابی که پیدا شده است، به عنوان تنها معیار آرمانی در نظر گرفته می‌شود؛ ولی با این وجود، معیارهای دیگری نیز توسط محققین علاوه بر معیار فوق ارائه شده است.

۶.۴.۳ همگرایی الگوریتم جست‌وجوی ممنوع

اصولاً همگرایی الگوریتم جست‌وجوی ممنوع پذیرفته شده است. با در نظر گرفتن یک جست‌وجو که تمامی جواب‌های مرور شده و جواب‌های انتخاب را به حافظه می‌سپرد، می‌توان نشان داد که قابلیت جست‌وجوی تمامی نقاط وجود دارد. البته، این خصوصیت در حالتی درست است که مجموعه جواب‌ها محدود بوده، تمامی همسایه‌ها معکوس‌پذیر باشند (برای هر همسایه x، خود x نیز جزء همسایه آن باشد) و قویاً دارای خصوصیت اتصال (از هر نقطه فضای جواب بتوان به سایر نقاط با استفاده از مجموعه‌ای حرکت رسید) باشد. همچنین، نشان داده شد که در حالت وجود لیست ممنوع تصادفی، رفتار الگوریتم شبیه به الگوریتم گرم و سرد کردن شبیه‌سازی شده بوده و دارای همگرایی است.

در مواردی نشان داده شده است که شکل ساده الگوریتم جست‌وجوی ممنوع، با وجود این‌که می‌تواند در عمل کارا باشد، ممکن است در شرایط خیلی بد نتواند به بهینه سراسری دست یابد. در چنین مواقعی بایستی، پارامترهای الگوریتم جست‌وجوی ممنوع به خوبی طراحی گردند که منجر به بهبود کیفیت الگوریتم گردد.

۷.۴.۳ حافظه بلند مدت

در مواردی که همسایه‌ها بر اساس یک مجموعه ثابت از حرکت‌ها، به عبارت دیگر نوع حرکت به جواب جاری وابسته نباشد، تعیین می‌شوند، آمار حرکت‌های انتخابی در طول جست‌وجو می‌تواند جالب توجه باشد. اگر بعضی از حرکت‌ها خیلی بیشتر از سایر حرکت‌ها تکرار شوند، ضروری است فرض کنیم که جست‌وجو با مشکلاتی در ارتباط با ایجاد گوناگونی در فضای جست‌وجو مواجه شده است و ممکن است جست‌وجو در یک دره پهناور گیر بیافتد. در صورتی که فضای جواب را دو بعدی فرض کنیم و مقدار تابع هدف را به

صورت ارتفاع آن فرض کنیم، ممکن است مجموعه جواب شبیه به مجموعه‌ای از دره‌ها و کوه‌ها دیده شود. در عمل، مسائل خیلی زیادی وجود دارند که شامل دره‌های متعدد و پهناوری هستند. گیر افتادن در دره، زمانی اتفاق می‌افتد که دامنه حرکت‌ها نسبت به خود مجموعه جواب یا تابع هدف، بسیار کوچک باشد. در چنین مواقعی، استفاده از لیست ممنوع شامل تعداد کمی از معکوس حرکت‌هایی که اخیراً انجام شده است، به عنوان تنها ابزار برای جهت‌دهی به جست‌وجو، در اغلب موارد، برای جلوگیری از محدود شدن در بعضی از دره‌ها کافی نیست. این در حالی است که اگر هم تعدادحرکت‌های ممنوع را افزایش دهیم، ممکن است که جست‌وجو در دامنه‌ها انجام گردد و به دلیل حرکت‌های ممنوع فراوان عمق دره‌ها را پیدا نکند. به عبارت دیگر، با افزایش طول لیست ممنوع ممکن است که الگوریتم، از یک دره خارج گردد و به دره دیگری وارد شود ولی در مجموع، امکان یافتن عمق دره‌ها کم است. در نتیجه، ضروری است که از مکانیزم‌های دیگری برای جهت‌دهی به جست‌وجو در بلند مدت استفاده نماییم.

به منظور اطمینان از وجود گوناگونی[۱۱] در طول جست‌وجو، بدون استفاده از تعداد زیادی از حرکت‌های ممنوع، یکی از تکنیک‌ها، استفاده از جریمه برای حرکت‌هایی است که خیلی بیشتر از بقیه اتفاق افتاده‌اند. روش‌های مختلفی را می‌توان برای تخصیص جریمه معرفی کرد. به طور مثال، ممنوعیت برای حرکت‌هایی که بیشتر از حد آستانه اتفاق افتاده‌اند یا افزایش جریمه‌ای متناسب با فرکانس هر حرکت، در زمان انتخاب حرکت‌ها، می‌تواند روش‌هایی از تخصیص جریمه باشد. در چنین مواقعی، جست‌وجو تمایل پیدا می‌کند که حرکت‌هایی را که کمتر استفاده شده‌اند با احتمال بیشتری مورد استفاده قرار دهد. این روش معمولاً به روش استفاده از حافظه مبتنی بر فرکانس[۱۲] مشهور است.

روش دیگری که ممکن است برای ایجاد حافظه بلند مدت استفاده گردد، اجبار برای انجام بعضی از حرکت‌ها[۱۳] می‌باشد. در این حالت، حرکت‌هایی که در تعداد زیادی از تکرارها، استفاده نشده‌اند، شناسایی شده و انجام آن‌ها، بدون توجه به کیفیت حرکت، اجباری می‌گردد. این روش به خروج از دره‌ها و ایجاد گوناگونی در جست‌وجو کمک می‌نماید.

مثال ۲.۴.۳. حل مساله تخصیص درجه دو با الگوریتم جست‌وجوی ممنوع

مساله تخصیص درجه دو عبارت است از تخصیص n شیء به n محل مشخص. بین این اشیا جریان f_{ij} وجود دارد. این جریان نشان‌دهنده میزان جریان یا برای مثال میزان حمل و نقل بین شیء iام و شیء jام می‌باشد.بین محل‌ها، فاصله‌ای برابر با d_{rs} وجود دارد که نشان‌دهنده میزان فاصله بین محل rام تا محل sام می‌باشد. هدف، حداقل سازی مجموع میزان جریان انتقالی بین اشیا ضرب در فاصله بین آن‌ها می‌باشد. یک مساله تخصیص درجه دو را با ۵ عضو در نظر بگیرید. ماتریس جریان آن با F و ماتریس فواصل آن با D به

[۱۱] Diversity
[۱۲] Frequency-based memory
[۱۳] Obligation to carry out move

صورت زیر تعریف می‌شوند؛

$$F = \begin{Bmatrix} ۰ & ۵ & ۲ & ۴ & ۱ \\ ۵ & ۰ & ۳ & ۰ & ۲ \\ ۲ & ۳ & ۰ & ۰ & ۰ \\ ۴ & ۰ & ۰ & ۰ & ۵ \\ ۱ & ۲ & ۰ & ۵ & ۰ \end{Bmatrix},$$

$$D = \begin{Bmatrix} ۰ & ۱ & ۱ & ۲ & ۳ \\ ۱ & ۰ & ۲ & ۱ & ۲ \\ ۱ & ۲ & ۰ & ۱ & ۲ \\ ۲ & ۱ & ۱ & ۰ & ۱ \\ ۳ & ۲ & ۲ & ۱ & ۰ \end{Bmatrix}.$$

تکرار صفر: جواب اولیه به صورت $p_۰ = (۲, ۴, ۱, ۵, ۳)$ فرض می‌شود. به این معنی که شیء دوم در محل اول، شیء چهارم در محل دوم، شیء اول در محل سوم، شیء پنجم در محل چهارم و شیء سوم در محل پنجم قرار دارند. متغیر x یک متغیر صفر و یک است که مشخص می‌کند چه شیای به چه محلی اختصاص یابد. با توجه به جواب اولیه داریم:

$$x_{۲۱} = x_{۴۲} = x_{۱۳} = x_{۵۴} = x_{۳۵} = ۱$$

مقدار تابع هدف از رابطه زیر محاسبه می‌شود:

$$\sum_i \sum_j \sum_r \sum_s f_{ij} d_{rs} x_{ir} x_{js}$$

بنابراین هزینه جواب اولیه برابر است با ۷۲. ساختار همسایگی، به صورت جابه‌جایی محل دو عضو از این ماتریس فرض می‌گردد. لذا تعداد همسایه‌های هر جواب برابر است با انتخاب ۲ از ۵ یعنی ۱۰. در ابتدای الگوریتم ماتریس ممنوع (لیست ممنوع) خالی است ($\tau = ۰$).

تکرار اول: برای ماتریس اولیه حرکت‌های ممکن تعیین می‌گردد. در زیر، همسایه‌های جواب جاری همراه با مقدار هزینه (تابع برازندگی) آن‌ها نشان داده شده است. مقدار تابع برازندگی به صورت اختلاف بین تابع هدف جدید و تابع هدف جاری است. همان‌طور که در جدول (۲.۳) دیده می‌شود، سه حرکت $(۱, ۴), (۱, ۳), (۱, ۲)$

move	(۱،۲)	(۱،۳)	(۱،۴)	(۱،۵)	(۲،۳)	((۲،۴	(۲،۵)	(۳،۴)	(۳،۵)	(۴،۵)
cost	۲	-۱۲	-۱۲	۲	۰	-۱۰	-۱۲	۴	۸	۶

جدول ۲.۳: همسایه‌های جواب اولیه و تابع برازندگی آن‌ها

و $(۲, ۵)$ سودی برابر با ۱۲ (کاهش میزان تابع هدف به اندازه ۱۲) را که بیشترین سود نیز هست، دارند. در زمانی که بهترین حرکت چندین حرکت باشد، بایستی قانونی برای انتخاب یکی از آن‌ها تعریف گردد. در این‌جا فرض می‌شود که در چنین شرایطی اولین حرکت انتخاب گردد. لذا حرکت $(۱, ۳)$ به عنوان بهترین

حرکت انتخاب می‌گردد. مقدار دوره شرط ممنوع (t) به طور تصادفی برابر با $t = ۹$ به دست آمده است. در نتیجه، عکس حرکت فوق تا ۹ تکرار دیگر (تکرار شماره ۱۰) ممنوع خواهد بود. مقدار ۱۰ برای عکس حرکت فوق در ماتریس ممنوع قرار می‌گیرد. در نتیجه، شیء شماره ۲ در محل ۱ و شیء شماره ۱ در محل ۳ تا تکرار شماره ۱۰ نمی‌توانند به طور هم‌زمان قرار بگیرند. ماتریس ممنوع در زیر نشان داده شده است؛

$$\tau = \begin{Bmatrix} ۰ & ۰ & ۱۰ & ۰ & ۰ \\ ۱۰ & ۰ & ۰ & ۰ & ۰ \\ ۰ & ۰ & ۰ & ۰ & ۰ \\ ۰ & ۰ & ۰ & ۰ & ۰ \\ ۰ & ۰ & ۰ & ۰ & ۰ \end{Bmatrix}$$

تکرار دوم: نقطه جاری به صورت $p_۱ = (۱, ۴, ۲, ۵, ۳)$ با هزینه ۶۰ به دست می‌آید. برای این نقطه، حرکت‌های ممکن مشخص و در زیر نمایش داده شده است. حرکت $(۱, ۳)$ به دلیل این‌که موجب برگشت اشیا ۱ و۲ به محل‌های قبلی خود می‌شود، ممنوع است. در بین سایر حرکت‌ها، حرکت $(۱, ۴)$ بیشترین سود (کاهش ۸ واحد از تابع هدف) را داشته و به عنوان بهترین حرکت انتخاب می‌گردد. مقدار t به طور تصادفی برابر با ۶ به دست آمده است.ماتریس ممنوع به‌هنگام شده و در زیر نمایش داده شده است. در نتیجه عکس حرکت فوق تا ۶ تکرار دیگر (تکرار شماره ۸) ممنوع خواهد بود.

move	(۱،۲)	(۱،۳)	(۱،۴)	(۱،۵)	(۲،۳)	((۲،۴	(۲،۵)	(۳،۴)	(۳،۵)	(۴،۵)
cost	۱۴	۱۲	-۸	۱۰	۰	۱۰	۸	۱۲	۱۲	۶
tabu		yes								

جدول ۳.۳: حرکت‌های ممکن برای $p_۱$

$$\tau = \begin{Bmatrix} ۸ & ۰ & ۱۰ & ۰ & ۰ \\ ۱۰ & ۰ & ۰ & ۰ & ۰ \\ ۰ & ۰ & ۰ & ۰ & ۰ \\ ۰ & ۰ & ۰ & ۰ & ۰ \\ ۰ & ۰ & ۰ & ۸ & ۰ \end{Bmatrix}$$

تکرار سوم: نقطه جاری به صورت $p_۲ = (۵, ۴, ۲, ۱, ۳)$ با هزینه ۵۲ به دست می‌آید. برای این نقطه، حرکت‌های ممکن مشخص شده و در زیر نشان داده شده است. حرکت $(۱, ۳)$ که در تکرار قبلی ممنوع بود، در این تکرار ممنوع نیست زیرا انجام این حرکت در این تکرار منجر به برگشت هم‌زمان هیچ دو عضوی که قبلاً جابه‌جا شده‌اند، نمی‌گردد. حرکت $(۱, ۴)$ ممنوع است. در بین حرکت‌های ممکن، هیچ حرکتی نمی‌تواند تابع هدف را بهبود دهد. با این وجود، طبق الگوریتم جست‌وجوی ممنوع، بهترین حرکت اجباراً باید انجام گردد. در بین حرکت‌های ممکن، حرکت $(۲, ۳)$ از بقیه حرکت‌ها بهتر است و به عنوان حرکت بعدی انتخاب می‌گردد. مقدار t به طور تصادفی برابر با ۸ به دست می‌آید. ماتریس ممنوع به‌هنگام شده و در زیر نمایش داده شده است. در نتیجه، عکس حرکت فوق تا ۸ تکرار دیگر (تکرار شماره ۱۱) ممنوع خواهد بود.

move	(۱،۲)	(۱،۳)	(۱،۴)	(۱،۵)	(۲،۳)	((۲،۴	(۲،۵)	(۳،۴)	(۳،۵)	(۴،۵)
cost	۱۰	۲۴	۸	۱۰	۰	۲۲	۲۰	۸	۸	۱۴
tabu			yes							

جدول ۴.۳: حرکت‌های ممکن برای $p_۲$

$$\tau = \begin{Bmatrix} ۸ & ۰ & ۱۰ & ۰ & ۰ \\ ۱۰ & ۰ & ۱۱ & ۰ & ۰ \\ ۰ & ۰ & ۰ & ۰ & ۰ \\ ۰ & ۱۱ & ۰ & ۰ & ۰ \\ ۰ & ۰ & ۰ & ۸ & ۰ \end{Bmatrix}$$

تکرار چهارم: نقطه جاری به صورت $p_۳ = (۵, ۲, ۴, ۱, ۳)$ با هزینه ۵۲ به دست می‌آید. برای این نقطه حرکت‌های ممکن مشخص شده و در زیر نشان داده شده است. حرکت‌های $(۲, ۳)$ و $(۱, ۴)$ ممنوع می‌باشند. در بین حرکت‌های ممکن، هیچ حرکتی نمی‌تواند تابع هدف را بهبود دهد، با این وجود، طبق الگوریتم جست‌وجوی ممنوع، بهترین حرکت اجباراً باید انجام گردد. در بین حرکت‌های ممکن، حرکت $(۲, ۳)$ از بقیه حرکت‌ها هزینه کمتری را داشته ولی چون ممنوع است، انجام نمی‌گردد. بعد از این حرکت، حرکت‌های $(۲, ۴), (۱, ۴)$ و $(۲, ۵)$ بهترین حرکت‌ها هستند. حرکت $(۱, ۴)$ ممنوع است و در نتیجه حرکت $(۲, ۴)$ به عنوان حرکت بعدی انتخاب می‌گردد. مقدار $t = ۵$ به طور تصادفی به دست آمده است. ماتریس ممنوع به‌هنگام شده و در زیر نمایش داده شده است. در نتیجه، عکس حرکت فوق تا ۵ تکرار دیگر (تکرار شماره ۹) ممنوع خواهد بود.

move	(۱،۲)	(۱،۳)	(۱،۴)	(۱،۵)	(۲،۳)	((۲،۴	(۲،۵)	(۳،۴)	(۳،۵)	(۴،۵)
cost	۲۴	۱۰	۸	۱۰	۰	۸	۸	۲۲	۲۰	۱۴
tabu			yes		yes					

جدول ۵.۳: حرکت‌های ممکن برای $p_۳$

$$\tau = \begin{Bmatrix} ۸ & ۰ & ۱۰ & ۹ & ۰ \\ ۱۰ & ۹ & ۱۱ & ۰ & ۰ \\ ۰ & ۰ & ۰ & ۰ & ۰ \\ ۰ & ۱۱ & ۰ & ۰ & ۰ \\ ۰ & ۰ & ۰ & ۸ & ۰ \end{Bmatrix}$$

تکرار پنجم: نقطه جاری به صورت $p_۴ = (۵, ۱, ۴, ۲, ۳)$ با هزینه ۶۰ به دست می‌آید. همان‌طور که مشاهده می‌گردد، حرکت قبلی منجر به بدتر شدن مقدار تابع هدف شد. این حرکت با وجود این‌که حرکت بدی بوده ولی منجر به خروج از بهینه محلی می‌گردد. برای این نقطه حرکت‌های ممکن مشخص شده و در زیر نشان

داده شده است. حرکت‌های $(۱, ۴)$,$(۲, ۳)$ و $(۲, ۴)$ ممنوع می‌باشند. در بین حرکت‌های ممکن، حرکت $(۱, ۳)$ بهترین حرکت بوده و سودی برابر ۱۰ واحد به همراه دارد. این حرکت منجر به بهبود تابع هدف به مقدار ۵۰ می‌شود و بهترین نقطه به دست آمده تا کنون را نیز بهبود می‌دهد. ادامه الگوریتم تاثیری در بهترین

move	(۱،۲)	(۱،۳)	(۱،۴)	(۱،۵)	(۲،۳)	((۲،۴	(۲،۵)	(۳،۴)	(۳،۵)	(۴،۵)
cost	۱۲	-۱۰	۱۲	۱۰	۰	-۸	۴	۱۴	۲۰	۱۰
tabu			yes		yes	yes				

جدول ۶.۳: حرکت‌های ممکن برای $p_۴$

جواب به دست آمده ندارد، زیرا جواب $p_۴ = (۴, ۱, ۵, ۲, ۳)$ با هزینه ۵۰، جواب بهینه مساله می‌باشد که در تکرار پنجم به دست آمد.

فصل ۴

حل مساله مکان‌یابی پایانه‌های اتوبوس با استفاده از الگوریتم‌های تکاملی و ابتکاری

۱.۴ مقدمه

در این فصل ابتدا به بررسی روش GLS[۱] (جست‌وجوی موضعی ژنتیک) برای حل مسائل مکان‌یابی پایانه اتوبوس می‌پردازیم، که توسط قنبری و مهدوی امیری [۱۶] ارائه شده است. سپس الگوریتمی ترکیبی با استفاده از الگوریتم ژنتیک و الگوریتم جست‌وجوی ممنوع ارائه می‌دهیم.

همان‌طور که در فصل اول بیان شد مساله مکان‌یابی پایانه اتوبوس یا BTLP به تعیین مکان پایانه اتوبوس با هدف ماکزیمم کردن سرویس حمل و نقل عمومی می‌پردازد.

ایده ترکیب کردن یک الگوریتم ژنتیک و ابتکاری برای حل مسائل بهینه‌سازی ترکیبی بیش از دو دهه است که مورد توجه برخی از پژوهشگران قرار گرفته است. جست‌وجوی موضعی ژنتیک که به الگوریتم Memetic معروف است، یک روش ابتکاری ترکیبی است که مزیت‌های الگوریتم ژنتیک و جست‌وجوی موضعی یا جست‌وجوی محلی (LS[۲]) را ترکیب می‌کند. در این الگوریتم جست‌وجوی موضعی بهترین جواب‌ها (نه بهینه کلی) را پیدا کرده و سپس الگوریتم ژنتیک جواب‌ها را بهبود می‌بخشد. در برخی حالات بحث‌های ابتکاری موضعی یا بسط جست‌وجوی موضعی به جای LS مورد استفاده قرار می‌گیرند.

الگوریتم GLS به طور وسیع در مسائل بهینه‌سازی ترکیبی مورد استفاده قرار گرفته است[۳۱]، [۳۷]، [۳۶]، [۲۷]. تعاریف مناسب از عملگرهای ژنتیک از قبیل: عملگر انتخاب، عملگر ترکیب و عملگر جهش اثر بسزایی در کارایی الگوریتم جست‌وجوی موضعی ژنتیک برای مسائل بهینه‌سازی ترکیبی خواهند داشت. از این رو برای یک مساله بهینه سازی، عملگرها و پارامترهای الگوریتم تکاملی ژنتیک باید با دقت تعیین شوند.

[۱] Genetic local search
[۲] Local search

قنبری و مهدوی امیری برای حل مساله مکان‌یابی پایانه اتوبوس توسط الگوریتم GLS عملگرهای ترکیب و جهش جدیدی را مورد بررسی قرار داده‌اند و همچنین برای هر گره یک تابع هدف پتانسیل که به اختصار با POF[۳] نشان می دهیم را تعریف کرده‌اند، ما در این فصل به بیان و بررسی چگونگی عملکرد آن‌ها می‌پردازیم.

۲.۴ مدل ریاضی

فرض کنید J مجموعه رئوسی از یک شهر است که تعداد مسافران ورودی و خروجی در این رئوس معلوم است که همان پتانسیل رئوس نامیده می‌شود. I را هم مجموعه‌ای از نقاط داوطلب برای پایانه‌های اتوبوس در نظر بگیرید، جایی که هر پایانه اگر تاسیس شود می‌تواند نقطه سرویس‌دهی در همسایگی خودش باشد. فرض شده است که هزینه‌های تاسیس پایانه‌های اتوبوس مساوی هستند، از این رو بدون از دست دادن کلیت این هزینه‌ها صفر در نظر گرفته می شوند.
هدف انتخاب p پایانه است به‌طوری‌که تابع سرویس‌دهی که در زیر تعریف می‌کنیم ماکزیمم شود.

تعریف ۱.۲.۴. برای هر $i \in I$، همسایگی راس i که با J_i^* نشان می‌دهیم به صورت زیر تعریف می شود

$$J_i^* = \{j \in J : C_{ij} \leq r\}$$

که r یک عدد ثابت و C_{ij} فاصله‌ی بین راس j و پایانه i می‌باشد. همچنین قرار دهید $J^* = \cup_{i \in S} J_i^*$ که $S \subseteq I$ می باشد.

تعریف ۲.۲.۴. فرض کنید پتانسیل رأس j توسط d_j و فاصله‌ی بین راس j و پایانه i توسط C_{ij} نشان داده شود، همچنین f را به عنوان یک تابع کاهشی در نظر بگیرید، که در اینجا از تابع کاهشی $f(x) = e^{-x}$ استفاده شده است.
برای $S \subseteq I$، تابع سرویس‌دهی به صورت زیر تعریف می شود

$$F(S) = \sum_{j \in J^*} d_j \times f(\min_{i \in S}\{C_{ij}\})$$

تعریف ۳.۲.۴. تابع هدف پتانسیل یا POF برای هر $i \epsilon I$ به صورت زیر تعریف می شود

$$POF(i) = \sum_{j \in J_i^*} d_j \times f(C_{ij})$$

اکنون می‌توانیم به بیان فرمول ترکیبی BTLP بپردازیم،
داده‌های ورودی برای مساله مکان یابی پایانه اتوبوس عبارتند از

- $I = \{i_1, i_2, \ldots, i_m\}$: مجموعه‌ای از رئوس داوطلب برای تاسیس پایانه‌ها.

[۳] Potential objective function

- $J = \{j_۱, j_۲, \ldots, j_n\}$: مجموعه‌ای از رئوس در شهر که تعداد مسافران در این رئوس مشخص می‌باشد.
- $C = [C_{ij}]$: ماتریس فاصله.
- f: تابع کاهشی.
- p: تعداد ایستگاه‌های مورد نیاز.
- $D = \{d_۱, d_۲, \ldots, d_n\}$: مجموعه پتانسیل رئوس متناظر با J.

در این صورت، هدف از مساله، ماکزیمم کردن $F(S)$ است با شرایط مطرح شده، یعنی

$$\text{Max} \qquad F(S) = \sum_{j \in J^*} d_j \times f(\min_{i \in S}\{C_{ij}\})$$

subject to:

$$S \subseteq I,$$

$$|S| = p.$$

برای بیان و توضیح روش GLS قبل از هر چیز باید مفاهیمی از قبیل: کدگذاری جواب ها، تولید جمعیت اولیه، عملگرهای ژنتیک و الگوریتم جست‌وجوی موضعی به طور واضح مشخص شوند. اکنون به توضیح تک تک این مفاهیم می‌پردازیم.

۱.۲.۴ کدگذاری

همان‌طور که در فصل ۲ شرح داده شد، الگوریتم ژنتیک به جای کار بر روی متغیرهای مساله، با جواب‌ها (کروموزوم‌ها) یا شکل کد شده آن‌ها کار می‌کند. در روش GLS به کار رفته برای حل مساله فوق، کدگذاری خاصی مورد استفاده قرار گرفته است، که با ذکر مثالی به توضیح آن می‌پردازیم.
فرض کنید $m = ۵$ تعداد نقاط کاندیدا برای تاسیس پایانه و $p = ۲$ تعداد پایانه‌هایی باشد که تاسیس می‌شوند. همچنین فرض کنید که نقاط ۱ و ۴ نقاطی هستند که مکان‌یابی شده‌اند، در این‌صورت جواب یا کروموزوم این مساله به صورت زیر کدگذاری و نمایش داده می‌شود

۱	۴

که به منظور استفاده‌ی آسان‌تر از این جواب‌ها ما آن‌ها را به شکل دودویی زیر تبدیل کرده‌ایم.

۱	۰	۰	۱	۰

۲.۲.۴ جمعیت اولیه

برای تولید جمعیت اولیه (GIP[۴]) معمولاً یک جمعیت تصادفی مد نظر قرار می‌گیرد، از آن‌جا که کارایی الگوریتم تکاملی ژنتیک و GLS به تولید جمعیت اولیه بستگی دارد، یعنی اگر کیفیت جمعیت اولیه بهتر از میانگین جمعیت تصادفی باشد کارایی الگوریتم نیز بیشتر می‌شود. برای رسیدن به جواب‌های بهتر در جمعیت اولیه از تابع هدف پتانسیل برای هر $i \in I$ استفاده شده است، به این صورت که این جواب‌ها توسط انتخاب p گره متفاوت از I با استفاده از چرخ رولت در تابع هدف پتانسیل از هر گره به دست می‌آیند. روش رولتی که مورد استفاده قرار گرفته است، به شرح زیر می‌باشد.

در گام اول به هر یک از کروموزوم‌های جمعیت فعلی احتمالی را نسبت می‌دهیم که شانس انتخاب آن کروموزوم برای انتقال به حوضچه ژنتیک را نشان می‌دهد. تابع توزیع احتمال نسبی بر اساس میزان شایستگی هر کروموزوم و با توجه به مجموع مقدار تابع هدف همه کروموزوم‌ها، احتمالی را به هر یک از آن‌ها نسبت می‌دهد. یعنی به کروموزوم iام احتمال p_i را به صورت زیر نسبت می‌دهیم

$$p_i = \frac{F_i}{\sum_{j=1}^{N} F_j}$$

که در آن N اندازه جمعیت، F_i مقدار تابع شایستگی کروموزوم iام (که در این پایان نامه، تابع هدف به عنوان تابع شایستگی در نظر گرفته شده است) و $\sum_{j=1}^{N} F_j$ مجموع مقادیر تابع شایستگی کل کروموزوم‌هاست.

در گام دوم یک عدد تصادفی بین ۰ و ۱ تولید می‌شود و توزیع تجمعی احتمال هر یک از کروموزوم‌ها محاسبه می‌شود، به این صورت که مقدار احتمال هر کروموزوم با مقدار احتمال کروموزوم‌های قبل از آن جمع می‌شود، سپس کروموزوم‌ها از اول بررسی شده و اولین کروموزوم که توزیع تجمعی احتمال آن بیشتر یا مساوی عدد تولید شده باشد برای انتقال به حوضچه انتخاب می‌گردد.

در تولید جمعیت اولیه همان‌گونه که توضیح داده شد، برای رسیدن به جواب‌های بهتر به جای تابع هدف هر کروموزوم، از تابع هدف پتانسیل برای هر گره استفاده شده است.

از طرف دیگر در الگوریتم‌های تکاملی ترجیح داده می‌شود که جمعیت اولیه تقسیم شود، از این رو نیمی از جمعیت اولیه به طور تصادفی و نیمی دیگر با استفاده از چرخ رولت انتخاب می‌شود. با توجه به این نکات جمعیت اولیه با دستورالعمل زیر تولید می‌شود.

[۴] Generating the initial population

دستورالعمل GIP
پارامتر: n (اندازه جمعیت)
$IP := \varnothing$ (جمعیت اولیه)
مرحله تصادفی:
۱)تا زمانی که $|IP| < \lfloor \frac{n}{۲} \rfloor$ مراحل زیر را تکرار کن
a)جواب تصادفی S_i را که $S_i \notin IP$ تولید کن
b)قرار بده $IP := IP \cup \{S_i\}$
مرحله ابتکاری:
۲)تا زمانی که $|IP| < n$ مراحل زیر را تکرار کن
a) جواب S_i را که $S_i \notin IP$ با انتخاب p گره از I با استفاده از چرخ رولت در POF هر گره تولید کن
b)قرار بده $IP := IP \cup \{S_i\}$
پایان

مثال ۴.۲.۴. فرض کنید از بین $I = \{۱, ۲, ۳, ۴\}$ پایانه می‌خواهیم $p = ۳$ پایانه را انتخاب کنیم، بنابراین هر زیر مجموعه ۳ عضوی از I که عضو تکراری نداشته باشد به عنوان یک جواب در نظر گرفته می‌شود. فرض کنید اندازه جمعیت برابر است با $n = ۷$، اکنون جمعیت اولیه به صورت زیر حاصل می‌شود؛

قرار دهید $|IP| = \emptyset$

مرحله تصادفی:

۱) تا زمانی که $|IP| < ۳$ مراحل زیر تکرار می‌شود

a) فرض کنید جواب تصادفی $S_۱ = \{۱, ۲, ۴\}$ تولید شده است.

b) $IP = IP \cup S_۱ = \{\{۱, ۲, ۴\}\}$ چون $|IP| = ۱$ است به a برمی‌گردیم

a) دوباره جواب تصادفی تولید می‌شود، اگر این جواب تولید شده در IP موجود باشد، a تا زمانی تکرار می‌شود که جواب تصادفی تولید شده در IP موجود نباشد. فرض کنید $S_۲ = \{۲, ۳, ۵\}$ تولید می‌شود که در IP نبوده، بنابراین به b می‌رویم

b) $IP = IP \cup S_۲ = \{\{۱, ۲, ۴\}, \{۲, ۳, ۵\}\}$، $|IP| = ۲$ است بنابراین به مرحله ابتکاری می‌رویم.

مرحله ابتکاری:

۲) تا زمانی که $|IP| < ۷$ باشد جواب S_i را با انتخاب ۳ گره از I با استفاده از چرخ رولت در POF هر گره تولید می‌کنیم. برای این کار ابتدا ماتریس فاصله و بردار پتانسیل به صورت تصادفی توسط نرم‌افزار متلب تولید شده‌اند (درایه‌ها اعداد تصادفی تولید شده در بازه $(۱, ۱۰)$ می‌باشند).

$$C = \begin{Bmatrix} ۹ & ۱۰ & ۲ \\ ۱۰ & ۷ & ۱ \\ ۳ & ۶ & ۱۰ \\ ۱۰ & ۲ & ۱۰ \\ ۱۰ & ۵ & ۹ \end{Bmatrix}$$

$$d = \{۱ \quad ۹ \quad ۱۰\}$$

ستون‌های ماتریس C نشان‌دهنده رئوسی است که تعداد مسافران در آن‌ها مشخص است، و سطرهای آن

نشان‌دهنده رئوس داوطلب برای تاسیس پایانه هستند. اکنون به محاسبه تابع هدف پتانسیل برای هر گره توسط فرمول زیر می‌پردازیم که در آن f تابع کاهشی e^{-x} می‌باشد.

$$POF(i) = \sum_{j \in J_i^*} d_j \times f(C_{ij})$$

$$POF(۱) = ۱/۳۵۳ \ , \ POF(۲) = ۳/۶۷۸ \ , \ POF(۳) = ۰/۰۴۹$$

$$POF(۴) = ۱/۲۱۸ \ , \ POF(۵) = ۰/۰۶۰$$

اکنون به هر گره احتمالی به صورت زیر نسبت داده می‌شود؛

$$p(i) = \frac{POF(i)}{\sum_{j=۱}^{۵} POF(j)}$$

خواهیم داشت؛

$$p(۱) = ۰/۲۱۲ \ , \ p(۲) = ۰/۵۷۸ \ , \ p(۳) = ۰/۰۰۷$$

$$p(۴) = ۰/۱۹۱ \ , \ p(۵) = ۰/۰۱۲$$

و توزیع تجمعی احتمال هر گره به صورت زیر خواهد بود؛

$$F(۱) = ۰/۲۱۲ \ , \ F(۲) = ۰/۷۹ \ , \ F(۳) = ۰/۷۹۸$$

$$F(۴) = ۰/۹۸۸ \ , \ F(۵) = ۱$$

حال فرض کنید عدد تصادفی $r = ۰/۵۰$ تولید شده است، اولین توزیع تجمعی بیشتر یا مساوی r برابر است با $F(۲) = ۰/۷۹$ پس گره ۲ انتخاب می‌شود، باز عدد تصادفی r تولید شده و این مرحله تا $p = ۳$ تکرار می‌شود. مجموعه ۳ عضوی به دست آمده به عنوان یک جواب انتخاب می‌شود با این شرط که این جواب در IP موجود نباشد، در غیر این‌صورت جواب جدید تولید می‌شود. به همین ترتیب تا رسیدن به اندازه جمعیت جواب‌ها تولید شده و جمعیت اولیه ساخته می‌شود.

۳.۲.۴ عملگرهای ژنتیک

۱. عملگر تقاطع (DPX[۵]) همان‌گونه که در فصل ۲ توضیح داده شد، مشخصات خوب والدین، که خود توسط چرخ رولت بر اساس رتبه‌بندی خطی انتخاب شده‌اند را برای تولید فرزندان بهتر ترکیب می‌کند. به عبارتی، عملگر تقاطع ناحیه‌ای از فضای جست‌وجو، که در آن بهترین جواب‌ها به عنوان خروجی مدنظر قرار گرفته می‌شوند را تعیین می‌کند. در این‌جا یک عملگر تقاطع جدید مورد استفاده

[۵] Distinctive preserve recombination

قرار گرفته است که دستورالعمل آن به شرح زیر می‌باشد

دستورالعمل DPX
پارامترها: a و b (والدین)
۱)$C := a \cap b$
۲)$t := p - |C|$
۳)$a \triangle b := (a - b) \cup (b - a)$
۴)زیرمجموعه‌ای با t عضو از $S := a \triangle b$ را به طور تصادفی انتخاب کن
۵)$children := C \cup S$
۶)پایان

مثال ۵.۲.۴. فرض کنید از بین $m = ۵$ پایانه می‌خواهیم $p = ۳$ پایانه را انتخاب کنیم، هم‌چنین فرض کنید والدین $a = [۱, ۳, ۵]$ و $b = [۲, ۴, ۵]$ داده شده‌اند، در این‌صورت دستورالعمل تقاطع به صورت زیر اعمال خواهد شد،

۱) $C = [۵]$,$|C| = ۱$

۲)$t = ۳ - ۱ = ۲$

۳)$a \triangle b = [۱, ۲, ۳, ۴]$

در این مرحله یک زیرمجموعه ۲ عضوی از $a \triangle b$ به طور تصادفی انتخاب می‌شود، فرض کنید زیر مجموعه زیر تولید شده است،

۴)$S = [۲, ۳]$

اکنون جواب به صورت زیر خواهد بود،

۵)فرزندان $= [۲, ۳, ۵]$

۲. **عملگر جهش:**

مرز و فریسلبن [۳۶] به این نتیجه رسیدند که عملگر جهش سعی بر این دارد که مرکز جست‌وجو را در انتخاب تصادفی فضایی قرار دهد که الگوریتم در صدد پیدا کردن آن دسته از جواب‌هایی است که توسط عملگر تقاطع (DPX) به سختی قابل دسترسی هستند.

از سوی دیگر هر چه سازگاری بین عملگر تقاطع و عملگر جهش بیشتر باشد کارایی الگوریتم ژنتیک افزایش داده می‌شود. در ادامه عملگر جهشی مورد استفاده قرار گرفته است که انتظار می‌رود منجر به دست‌یابی به جوابی بهتر شود. دستورالعمل زیر به شرح این عملگر می‌پردازد.

دستورالعمل جهش
پارامتر: S (یک جواب شدنی)
۱) $M := \{۱, \ldots, m\}$
۲) $T := M - S$
۳) $i \in S$ را به طور تصادفی انتخاب کن
۴) $j^* = arg(\max_{j \in T}(POF(j)))$
۵) $S := S - \{i\} \cup \{j^*\}$
۶) پایان

مثال ۶.۲.۴. قرار دهید $m = ۵$ و $p = ۳$ و فرض کنید $S = [۲, ۴, ۵]$ یک جواب شدنی است که می‌خواهیم عمل جهش را روی آن انجام دهیم، خواهیم داشت

۱) $M = \{۱, ۲, ۳, ۴, ۵\}$

۲) $T = M - S = \{۱, ۳\}$

فرض کنید $i = ۴$ به طور تصادفی انتخاب شده است. ماتریس‌های فاصله و پتانسیل به طور تصادفی توسط نرم‌افزار متلب تولید شده و مقدار تابع هدف پتانسیل مانند مثال (۴.۲.۴) برای گره‌های ۱ و ۳ به صورت زیر به دست آمده‌اند؛

$$POF(۱) = ۴/۰۶, \qquad POF(۳) = ۱۲/۵۳,$$

بنابراین $j^* = ۳$ می‌باشد، و جهش به صورت زیر به دست می‌آید

$$S = [۲, ۳, ۵].$$

۴.۲.۴ جست‌وجوی موضعی

بعضی از روش‌های جست‌وجوی موضعی در خصوص مسائل مکان‌یابی تسهیلات (FLP[۶]) در مقالات مختلفی ارائه شده‌اند. بر طبق نتایج خوبی که توسط گوش [۱۷] ارائه شده است، جست‌وجوی موضعی 2-swap انتخاب شده است. بنابراین همسایگی در مساله مکان‌یابی پایانه اتوبوس برای 2-swap به صورت زیر تعریف می شود.

تعریف ۷.۲.۴. (همسایگی $swap - ۲$)
S را به عنوان یک جواب شدنی در نظر بگیرید، به‌طوری‌که $S \subset I$ و $|S| = p$. آن‌گاه، مجموعه همسایگی‌های S به صورت زیر تعریف می‌شود،

$$N(S) = \{T | T \subset I, |T| = p, |S - T| = ۱\}.$$

یعنی یک همسایگی از S، جوابی است که با S تنها در یک رأس اختلاف دارد.

[۶] Facility location problem

معمولاً برای استفاده از جست‌وجوی موضعی دو رویکرد مورد توجه قرار می‌گیرند:

- با توجه به همسایگی تعریف شده، در حین جست‌وجو اگر همسایگی‌ای پیدا شود که تابع هدف آن نسبت به تابع هدف نقطه‌ی شروع بهتر باشد، آن همسایگی انتخاب می شود.
- در این رویکرد تمام همسایگی‌های نقطه شروع لیست شده، مقدار تابع هدف آن‌ها محاسبه شده و از بین آن‌ها همسایگی با تابع هدف بهتر انتخاب می‌شود.

آزمایش‌های عددی انجام شده نشان می‌دهد که رویکرد دوم، یعنی الگوریتم بهترین همسایگی‌ها نسبت به رویکرد اول لزوماً جواب‌های بهتری را فراهم نمی‌کند، ضمن این‌که زمان بیشتری نیز برای رسیدن به جواب بهتر نیاز دارد. از این رو برای استفاده از روش GLS در حل BTLP، رویکرد اول، یعنی رویکرد اولین پیشرفت حاصل شده در جست‌وجوی موضعی برای 2-swap مورد استفاده قرار گرفته است.
با توجه به آزمایش‌های عددی انجام شده، در روش GLS بیشترین زمان صرف الگوریتم جست‌وجوی موضعی می‌شد، از این رو در این قسمت، یک زمان مشخص برای حل LS در نظر گرفته شده، و در نهایت معیارهای توقف الگوریتم LS به صورت زیر در نظر گرفته شده است.

- تعداد تکرار معلوم
- تعداد پیشرفت معلوم
- بیشترین زمان اجرا
- درصد پیشرفت مشخص

از طرفی، چون الگوریتم GLS ترکیبی از الگوریتم ژنتیک و الگوریتم LS است، به محض این‌که جواب اولیه در LS بهبود پیدا کرد، روند LS متوقف شده و الگوریتم ژنتیک را روی این جواب‌های خوب به دست آمده از LS پیاده‌سازی می‌کنیم.
در دستورالعمل LS برای راحتی از عملگر $+_c$ استفاده شده است که در زیر تعریف شده است،

تعریف ۸.۲.۴. فرض کنید $a, b \in N \cup \{۰\}$ و $c \in N - \{۱\}$، آن‌گاه عملگر $+_c$ عبارت است از

$$a +_c b = \begin{cases} (a+b) - \lfloor \frac{a+b}{c} \rfloor \times c, & a+b > c \\ a+b, & o.w. \end{cases}$$

روش LSبه صورت زیر می‌باشد.

الگوریتم جست‌وجوی موضعی (LS)
پارامترها: $openT$ (یک جواب اولیه)، T_{max} (ماکزیمم تکرار)، I_{max} (ماکزیمم پیشرفت)، $Time_{max}$ (ماکزیمم زمان اجرا)، $pimprove$ (درصد بهبود).
۱) $closeT := \{۱, \ldots, m\} - openT$
۲) عددی تصادفی مانند i از $\{۱, \ldots, p\}$انتخاب کن
۳) عددی تصادفی مانند j از $\{۱, \ldots, m-p\}$انتخاب کن
۴) قرار بده $iter := ۰$ (تعداد تکرار)، $impro - iter := ۰$ (تعداد پیشرفت)
۵) $jj := ۰$
۶) $i := i +_p ۱$
۷) $t_i := openT[i]$
۸) $j := j +_{m-p} ۱$, $iter := iter + ۱$
۹) $Temp := openT \cup \{closeT$ازامین عضو $-j\} - \{t_i\}$
۱۰) اگر $F(Temp) - F(openT) > ۰$ آن‌گاه:
۱ + $impro - iter := impro - iter$ و $improved := \frac{F(Temp)-F(openT)}{F(openT)}$
و $closeT[j]$ و $openT[i]$ را با هم جابه‌جا کنید.
۱۱) اگر $improved > pimprove$ یا $impro - iter > I_{max}$ آن‌گاه آخرین $openT$به‌دست آمده را انتخاب کن و توقف کن.
۱۲) $pimprove := \frac{pimprove-improved}{۱+improved}$
۱۳) اگر $Time_{max} <$ زمان اجرا یا $iter > T_{max}$ آن‌گاه آخرین $openT$ به‌دست آمده را انتخاب کن و توقف کن.
۱۴) $jj := jj + ۱$
۱۵) اگر $jj < m - p$ به گام ۶ برو.
۱۶) پایان.

گام اول: در این گام $closeT$ محاسبه می‌شود که همه‌ی رئوس داوطلب برای تاسیس پایانه است به جز رئوسی که در جواب اولیه وجود دارند.

گام دوم: عدد تصادفی i از مجموعه $\{۱, \ldots, p\}$ انتخاب می‌شود، که p تعداد رئوس مورد نیاز بوده و مشخص است.

گام سوم: در این گام عدد تصادفی j از مجموعه $\{۱, \ldots, m-p\}$ یعنی تمام اعضای m به جز تعداد ایستگاه‌های مورد نیاز انتخاب می‌شود.

گام چهارم: معیارهای توقف تعداد تکرار و تعداد پیشرفت مساوی صفر قرار داده می‌شود.

گام پنجم: شمارنده $jj = ۰$ قرار داده می‌شود.

گام ششم: در این گام با استفاده از عملگر $+_c$، $i +_p ۱$ محاسبه می‌شود.

گام هفتم: iامین عضو از جواب اولیه به عنوان t_i در نظر گرفته می‌شود.

گام هشتم: در این گام یک واحد به مقدار تکرارها افزوده می‌شود و j با استفاده از عملگر $+_{m-p}$ محاسبه می‌شود.

گام نهم: در این گام جواب جدید با جابه‌جایی iامین عضو از جواب اولیه و jامین عضو از $closeT$ در جواب جاری حاصل می‌شود.

گام دهم: مقدار تابع هدف به ازای جواب جدید به دست آمده و با مقدار تابع هدف جواب جاری مقایسه می شود. چون مساله ماکزیمم سازی است اگر مقدار تابع هدف به ازای جواب جدید بیشتر از مقدار تابع هدف به ازای جواب جاری باشد یک واحد به تکرار پیشرفت‌ها اضافه کرده و $improved$ را از تفاضل این دو مقدار تقسیم بر تابع هدف جواب جاری به دست می‌آوریم و جواب جاری را به عنوان جواب جدید در نظر می‌گیریم.

گام یازدهم: دو معیار توقف درصد بهبود و ماکزیمم پیشرفت بررسی می‌شوند، اگر معیارهای توقف برآورده شده باشند آخرین جوابی را که به دست آمده به عنوان بهترین جواب در نظر گرفته و توقف می‌کنیم، در غیر این‌صورت به گام دوازدهم می‌رویم.

گام دوازدهم: در این گام $pimprove$ را با رابطه داده شده محاسبه می‌کنیم.

گام سیزدهم: در این گام دو معیار توقف ماکزیمم زمان اجرا و ماکزیمم تکرار بررسی می‌شوند، در صورت برقراری شرایط ذکر شده آخرین جواب به عنوان بهترین جواب در نظر گرفته شده و توقف می‌کنیم، در غیر این‌صورت به گام چهاردهم می‌رویم.

گام چهاردهم: یک واحد به شمارنده jj اضافه می‌کنیم.

گام پانزدهم: در گام پانزدهم اگر شمارنده jj از $(m-p)$ کمتر بوده به گام ششم رفته و i جدید را به دست می‌آوریم و مراحل تا برقراری اولین شرط توقف ادامه می‌یابد، در غیر این‌صورت توقف می‌کنیم و الگوریتم پایان می‌یابد.

برای درک بهتر الگوریتم جست‌وجوی موضعی به مثال زیر توجه کنید،

مثال ۹.۲.۴. $m = ۴$ و $p = ۲$ را در نظر بگیرید ، هم‌چنین $OpenT = \{۲, ۴\}$ را به عنوان یک جواب اولیه در نظر بگیرید. سعی داریم با الگوریتم جست‌وجوی موضعی از این جواب شروع کرده و آن را بهبود بخشیم، پارامترهای زیر نیز داده شده‌اند:

$$T_{max} = ۴, I_{max} = ۳, pimprove = ۱۰.$$

خواهیم داشت:

۱) $closeT = \{۱, ۲, ۳, ۴\} - \{۲, ۴\} = \{۱, ۳\}$;

۲) فرض کنید $i = ۲$ به‌طور تصادفی از بین $\{۱, ۲\}$ انتخاب شده است؛

۳) فرض کنید $j = ۲$ به‌طور تصادفی از بین $\{۱, ۲\}$ انتخاب شده است؛

۴) $impro - iter = ۰$, $iter = ۰$;

۵) $jj = ۰$;

۶) $i = ۲ +_{۲} ۱ = ۱$;

۷) $t_{۱} = ۲$;

۸) $iter = ۱$, $j = ۲ +_۲ ۱ = ۱$;

۹) $Temp = \{۱, ۴\}$;

۱۰) $F(Temp) - F(OpenT) = ۴/۰۶ - ۶/۷۶ = -۲/۷۰ < ۰$;

چون شرط برقرار نشده است به گام ۱۳ می‌رویم، داریم،

۱۳) $iter = ۱ < T_{max} = ۴$;

چون شرط توقف برقرار نشده است به گام ۱۴ می‌رویم،

۱۴) $jj = ۰ + ۱ = ۱$;

۱۵) $jj = ۱ < m - p = ۲$;

بنابراین به گام ۶ می‌رویم،

۶) $i = ۱ +_۲ ۱ = ۲$;

۷) $t_۲ = ۴$;

۸) $iter = iter + ۱ = ۲$, $j = ۱ +_۲ ۱ = ۲$;

۹) $Temp = \{۲, ۳\}$;

۱۰) $F(Temp) - F(OpenT) = ۴/۶۵ > ۰$;

شرط برقرار شده است، خواهیم داشت،

$impro - iter = impro - iter + ۱ = ۰ + ۱ = ۱, improved = ۰/۶۸۷, OpenT = \{۲, ۳\}$;

چون شروط گام ۱۱ برقرار نشده‌اند به گام ۱۲ می‌رویم،

۱۲) $pimprove = \frac{۱۰ - ۰/۶۸۷}{۱ + ۰/۶۸۷} = ۵/۵۴۹$;

هم‌چنان شرط توقف گام ۱۳ برقرار نشده است و به گام ۱۴ می‌رویم،

۱۴) $jj = ۱ + ۱ = ۲$;

۱۵) $jj = ۲ = m - p$;

۱۶) پایان.

مشاهده می‌کنید که $\{۲, ۳\}$ بهبود یافته $\{۲, ۴\}$ می‌باشد.

۳.۴ جست‌وجوی موضعی ژنتیک

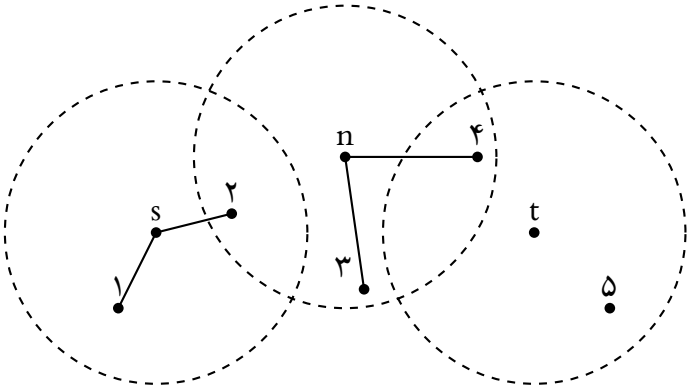
اکنون به بیان روش ترکیبی جست‌وجوی موضعی ژنتیک می‌پردازیم:

الگوریتم جست‌وجوی موضعی ژنتیک (GLS)
پارامترها:معیار توقف، np (اندازه جمعیت)، p_M (نرخ جهش)، p_C (نرخ تقاطع).
۱)جمعیت اولیه P را با دستورالعمل GIP تولید کن.
۲)برای تمام $p \in P$ قرار دهید، $p = LS(p)$
۳)مراحل زیر را تکرار کن
ساختن جمعیت جدیدTP
الف)$TP = \emptyset$
ب)برای np : $i = ۱$ مراحل زیر را انجام بده
a) دو والد $a, b \in P$ را با یک عملگر انتخاب، انتخاب کن
b) عدد تصادفی $r \in [۰, ۱]$ را تولید کن
c) اگر $r < p_C$ است، قرار بده $c = DPX(a, b)$، در غیر این‌صورت با احتمال تساوی قرار بده $c = a$ یا $c = b$
d) با احتمال p_M دستورالعمل جهش را روی c پیاده‌سازی کنید
e) قرار دهید، $c = LS(c)$
f) $TP = TP \cup \{c\}$
۴)با یک قانون انتخاب P جدید را از P و TP بساز
۵)قرار دهید، $P = newP$
۶)تا زمان برقراری معیار توقف روند را ادامه دهید
۷) پایان.

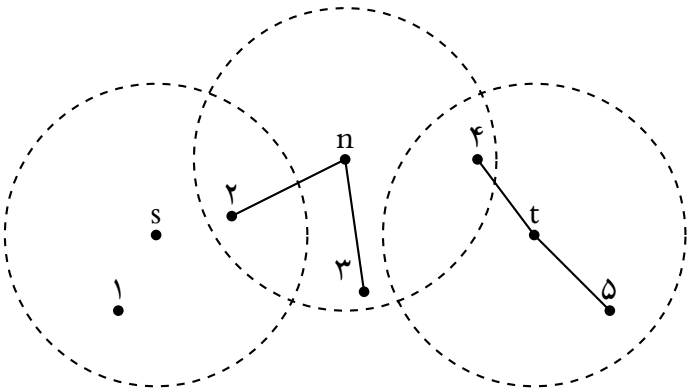
توجه کنید که معیار توقف در الگوریتم GLS برابر با تولید تعداد معلوم نسل قرار داده شده است، که ما در این‌جا تولید ۵ نسل را به عنوان معیار توقف در نظر گرفته‌ایم. هم‌چنین در انتخاب والدین از جمعیت اولیه، از عملگر انتخاب چرخ رولت استفاده شده است، که توضیح آن به طور کامل در فصل ۲ آورده شده است. لازم است توجه داشته باشید که محاسبه‌ی تابع هدف کار سخت و زمان‌بری است. بنابراین در روش جست‌وجوی موضعی تغییرات تابع هدف توسط POF هر ایستگاه به صورت زیر تخمین زده می‌شود.
فرض کنیدI، $S, T \subset I$، $|S| = |T| = p$، $|S - T| = ۱$، به عبارتی S و T دو جواب شدنی یا دو کروموزوم از BTLP هستند که تنها در یک عضو یا ژن با هم تفاوت دارند، که این همان تعریف همسایگی مورد نظر می‌باشد، پس وجود دارد $s \in S$ و $t \in T$ به‌طوری‌که $s \notin T$ و $t \notin S$ آن‌گاه $\Delta F(S, T) = F(T) - F(S)$ به صورت زیر تخمین زده می‌شود:

$$\widetilde{\Delta F} = \widetilde{\Delta F}(S, T) = POF(t) - POF(s)$$

توجه کنید که s و t در تعریف فوق منحصر به فرد هستند.
در محاسبه‌ی $\Delta F(S, T) = F(T) - F(S)$ که $s \in S$ بسته (به این معنی که در این نقطه پایانه‌ای وجود ندارد) و $t \in T$ باز (به این معنی که در این نقطه پایانه تاسیس شده است) می‌باشد، برای رئوس J چهار حالت پیش خواهد آمد. به عنوان مثال شکل (۱.۴) را ببینید. که در شکل (الف) گره‌های s و n باز هستند و t بسته است و در شکل (ب) گره n باز است و گره t به جای s باز است و s بسته است.
(الف)



(ب)



شکل ۱.۴: حالت‌های ممکن حاصل از همسایگی جدید

چهار حالت به صورت زیر داریم:

۱. رئوسی که با تغییر پایانه تغییر نکرده‌اند و هم‌چنان از پایانه قبل سرویس‌دهی می‌شوند (نظیر راس ۳ در شکل ۱.۴)

۲. رئوسی که قبلاً سرویس‌دهی می‌شدند اما در حال حاضر نمی‌شوند (نظیر راس ۱ در شکل ۱.۴)

۳. رئوسی که قبلاً سرویس‌دهی نمی‌شدند اما با تغییر پایانه امکان سرویس‌دهی به این رئوس فراهم شده است (نظیر راس ۵ در شکل ۱.۴)

۴. رئوسی که با تغییر پایانه، مراکز سرویس‌دهی آن‌ها تغییر کرده است (نظیر رئوس ۲ و ۴ در شکل ۱.۴)

به وضوح $\widetilde{\Delta F}$ یک تخمین از $\Delta F(S,T)$ می‌باشد، زیرا در محاسبه‌ی $\widetilde{\Delta F}$ ممکن است طراحی مناسبی از رئوس و سرویس‌دهی آن‌ها نداشته باشیم (به خصوص در حالت‌های ۱ و ۴).
در مثال زیر نتیجه حاصل از روش GLS را در دو حالت زیر برای یک مساله خاص بیان می‌کنیم.

۱. در حالتی که تابع برازندگی، همان تابع هدف است (GLS۱) .

۲. در حالتی که تابع برازندگی، تخمین تابع هدف یا به عبارتی تابع هدف پتانسیل می‌باشد (GLS۲) .

مثال ۱.۳.۴. فرض کنید می‌خواهیم از بین $m = ۴$ مکان پیشنهادی برای تاسیس $p = ۲$ پایانه را انتخاب کنیم، همچنین داده‌های ورودی زیر را در نظر بگیرید:

مجموعه رئوس داوطلب	I=$\{۱،۲،۳،۴\}$
مجموعه رئوس در شهر	J=$\{۱،۲،۳\}$
معیار توقف الگوریتم ژنتیک	تولید ۵ نسل
np: اندازه جمعیت	۴
T_{max}	۲۰
I_{max}	۱۰
pipmrove	۳۰
p_M	۰.۵۰
p_C	۰.۷۵
r:عدد ثابت	۳

جدول ۱.۴: اطلاعات مربوط به مثال (۱.۳.۴)

همچنین ماتریس فاصله و پتانسیل رئوس را به صورت زیر در نظر بگیرید:

$$C = \begin{bmatrix} ۲ & ۴ & ۶ \\ ۲ & ۲ & ۵ \\ ۳ & ۳ & ۱ \\ ۲ & ۴ & ۲ \end{bmatrix},$$

$$d = \{۳۰ \quad ۲۰ \quad ۲۰\}.$$

این برنامه توسط نرم‌افزار متلب نوشته و اجرا شد، در هر دو حالت فوق بهترین مقدار تابع هدفی که حاصل شد برابر ۱۸/۷۷۵۲ متناظر با جواب $[۲,۳]$ بود، و مدت زمان اجرای برنامه‌ها در حالت اول و دوم به ترتیب عبارت بود از ۰/۰۷۸۷ و ۰/۰۷۶۳. مشاهده می‌شود که با توجه به کوچک بودن داده‌های مساله فوق جواب به دست آمده به جواب بهینه نزدیک بوده و به همین دلیل اختلاف زیادی در دو حالت نام برده به وجود نیامده است. در بخش‌های بعد مسائل با اندازه‌های بزرگتر مورد بررسی و مقایسه قرار می‌گیرند.

۴.۴ الگوریتم ترکیبی جست‌وجوی ممنوع و ژنتیک

در ادامه این پایان‌نامه، به جای الگوریتم جست‌وجوی موضعی در الگوریتم ترکیبی GLS، از الگوریتم جست‌وجوی ممنوع استفاده شده است و نتایج قابل توجهی به دست آمده است. برای این منظور، بعد از تولید جمعیت اولیه، تک تک کروموزوم‌ها را با استفاده از الگوریتم جست‌وجوی ممنوع بهبود بخشیده و آن‌گاه الگوریتم ژنتیک را روی جواب‌های خوب به دست آمده از این روش اعمال می‌کنیم. از آن‌جا که در جست‌وجوی ممنوع به یافتن تمام همسایگی‌های جواب موردنظر پرداخته می‌شود، برای مثال‌های با ابعاد بزرگ زمان قابل توجهی صرف می‌شود.

روش جست‌وجوی ممنوع ژنتیک (GTS[۷]) به صورت زیر می‌باشد،

الگوریتم جست‌وجوی ممنوع ژنتیک (GTS)
پارامترها:معیار توقف، np (اندازه جمعیت)، p_M (نرخ جهش)، p_C (نرخ تقاطع).
۱)جمعیت اولیه P را با دستورالعمل GIP تولید کن.
۲)برای تمام $p \in P$ قرار دهید، $p = TS(p)$
۳)مراحل زیر را تکرار کن
ساختن جمعیت جدیدTP
الف)$TP = \emptyset$
ب)برای np : $i = ۱$ مراحل زیر را انجام بده
a) دو والد $a, b \in P$ رابا یک عملگر انتخاب، انتخاب کن
b) عدد تصادفی $r \in [۰, ۱]$ را تولید کن
c) اگر $r < p_C$ است، قرار بده $c = DPX(a, b)$، در غیر این‌صورت با احتمال تساوی قرار بده $c = a$ یا $c = b$
d) با احتمال p_M دستورالعمل جهش را روی c پیاده‌سازی کنید
e) قرار دهید، $c = TS(c)$
f) $TP = TP \cup \{c\}$
۴)با یک قانون انتخاب P جدید را از P و TP بساز
۵)قرار دهید، $P = newP$
۶)تا زمان برقراری معیار توقف روند را ادامه دهید
۷) پایان.

که در آن از همان معیار توقف و عملگر انتخاب به کار رفته در الگوریتم ترکیبی جست‌وجوی موضعی ژنتیک استفاده شده است. در قسمت بعد، برای مشاهده بهتر عملکرد روش‌های ترکیبی و ابتکاری، مسائل آزمون با الگوریتم ژنتیک نیز حل و نتایج عددی گزارش داده شده‌اند.

الگوریتم ژنتیک مورد استفاده در این پایان‌نامه به صورت زیر است:

[۷] Genetic tabu search

الگوریتم ژنتیک (GA)
پارامترها:معیار توقف، np (اندازه جمعیت)، p_M (نرخ جهش)، p_C (نرخ تقاطع).
۱)جمعیت اولیه P را با دستورالعمل GIP تولید کن.
۲)مراحل زیر را تکرار کن
ساختن جمعیت جدیدTP
الف)$TP = \emptyset$
ب)برای np : $i = ۱$ مراحل زیر را انجام بده
a) دو والد $a, b \in P$ رابا یک عملگر انتخاب، انتخاب کن
b) عدد تصادفی $r \in [۰, ۱]$ را تولید کن
c) اگر $r < p_C$ است، قرار بده $c = DPX(a, b)$، در غیر این‌صورت با احتمال تساوی قرار بده $c = a$ یا $c = b$
d) با احتمال p_M دستورالعمل جهش را روی c پیاده‌سازی کنید
e) $TP = TP \cup \{c\}$
۳)با یک قانون انتخاب P جدید را از P و TP بساز
۴)قرار دهید، $P = newP$
۵)تا زمان برقراری معیار توقف روند را ادامه دهید
۶) پایان.

۵.۴ مقایسه و نتیجه‌گیری

اکنون زمان آن فرا رسیده است که از بین روش‌های ذکر شده برای حل مساله مکان‌یابی پایانه اتوبوس، روش بهینه را شناسایی و معرفی کنیم. برای این کار، ۱۰ مساله آزمون مورد بررسی و مقایسه قرار گرفت، که برای هر یک از آن‌ها ماتریس‌های فاصله و پتانسیل به طور تصادفی تولید شدند، به این صورت که درایه‌های آن‌ها اعداد تصادفی تولید شده بین ۰ و ۱۰۰ بودند. هم‌چنین در این مسائل، $r = ۳۰$، $p_M = ۰/۵۰$، $p_C = ۰/۷۵$ و $np = ۲۰$ در نظر گرفته شد.

معیارهایی که برای کارایی و مقایسه الگوریتم‌ها مد نظر قرار گرفته‌اند، عبارتند از:

- زمان CPU
- مقدار تابع هدف

نتایج عددی به دست آمده به صورت جدول (۲.۴) و (۳.۴) می‌باشند. در جدول (۲.۴) مقدار تابع هدف مسائل با استفاده از روش‌های مختلف ارائه شده است و جدول (۳.۴) نشان‌دهنده زمان اجرای آن‌ها می‌باشد.

در جدول (۲.۴) و (۳.۴)، GA نشان‌دهنده الگوریتم ژنتیک، GLS1 نشان‌دهنده جست‌وجوی موضعی ژنتیک در حالتی است که تابع هدف به عنوان تابع برازندگی در نظر گرفته شده است، GLS2 نشان‌دهنده جست‌وجوی موضعی ژنتیک برای حالتی است که تابع هدف پتانسیل به عنوان تابع برازندگی در نظر گرفته شده است و GTS نشان‌دهنده جست‌وجوی ممنوع ژنتیک می‌باشد. ستون آخر جدول (۲.۴) مقدار بهینه را از بین ۴ روش

test	m	n	p	GA	GLS۱	GLS۲	GTS	best
۱	۲۰	۱۰	۸	۴۸/۰۸۳۸	۴۸/۰۸۳۸	۴۸/۰۸۳۸	۴۸/۰۸۳۸	۴۸/۰۸۳۸
۲	۳۰	۱۰	۱۰	۵۱/۱۰۶۹	۵۱/۱۰۶۹	۵۱/۱۰۶۹	۵۱/۱۰۶۹	۵۱/۱۰۶۹
۳	۵۰	۳۰	۱۰	۱۴۸/۱۷۵۲	۱۵۰/۰۹	۱۴۴/۱۹۱۴	۱۵۲/۵۶۷	۱۵۲/۵۶۷
۴	۵۰	۳۰	۲۰	۱۷۹/۵۰۷۹	۱۸۳/۰۵۸	۱۸۲/۵۳۸	۱۸۵/۳۴۰	۱۸۵/۳۴۰
۵	۷۰	۴۰	۲۰	۴۱۶/۶۰۵۲	۴۱۸/۳۹۲۱	۴۰۷/۶۶۰۳	۴۴۴/۲۰۶۸	۴۴۴/۲۰۶۸
۶	۷۰	۴۰	۴۰	۴۸۲/۰۷۷	۴۸۳/۰۲۳	۴۸۲/۵۵۸۱	۴۸۴/۰۲۵	۴۸۴/۰۲۵
۷	۱۰۰	۴۰	۳۰	۴۶۷/۰۰۶۶	۴۶۷/۳۷۶	۴۵۷/۸۳۸۷	۴۷۷/۶۹۵۶	۴۷۷/۶۹۵۶
۸	۱۰۰	۴۰	۵۰	۵۲۴/۱۹۴۴	۵۳۲/۰۰۱۴	۵۲۳/۷۷۰	۵۳۲/۴۳۳	۵۳۲/۴۳۳
۹	۲۰۰	۷۰	۵۰	۷۷۶/۹۵۲۳	۸۲۶/۳۷۲۸	۸۰۸/۳۹۶۲	۸۴۲/۱۹۸۳	۸۴۲/۱۹۸۳
۱۰	۲۰۰	۷۰	۸۰	۸۸۰/۶۸۱۸	۸۹۶/۷۷۵۸	۸۸۱/۵۰۲۶	۹۰۰/۲۰۲۲	۹۰۰/۲۰۲۲

جدول ۲.۴: نتایج به دست آمده بر اساس مقدار تابع هدف

test	m	n	p	GA	GLS۱	GLS۲	GTS
۱	۲۰	۱۰	۸	۱/۰۳۵۳	۲/۳۵۲۱	۲/۰۹۲۵	۹/۱۱۷۷
۲	۳۰	۱۰	۱۰	۱/۹۳۶۴	۲/۱۰۷۶	۲/۲۲۴۶	۱۸/۹۳۳۲
۳	۵۰	۳۰	۱۰	۳/۹۱۳۷	۴/۴۳۴۱	۴/۲۱۷۸	۷۶/۹۴۹۷
۴	۵۰	۳۰	۲۰	۳/۷۰۷۷	۴/۲۳۶۵	۴/۰۲	۱۱۰/۴۸۱۰
۵	۷۰	۴۰	۲۰	۵/۶۶۸۴	۶/۳۷۶۷	۵/۸۷۷۹	۲۸۷/۷۹۷۵
۶	۷۰	۴۰	۴۰	۵/۲۸۰۷	۶/۰۱۴۳	۵/۴۳۶۲	۳۱۹/۲۳۲۹
۷	۱۰۰	۴۰	۳۰	۷/۶۷۰	۸/۶۸۱۴	۷/۹۹۹۳	۷۷۷/۹۷۴۰
۸	۱۰۰	۴۰	۵۰	۷/۱۴۷	۸/۰۰۷۹	۷/۴۰۶۴	۸۸۲/۹۷۸۰
۹	۲۰۰	۷۰	۵۰	۱۵/۲۶۶۰	۱۷/۳۲۶۴	۱۵/۴۸۷۵	۵۴۳۴/۰۵۸۶
۱۰	۲۰۰	۷۰	۸۰	۱۴/۴۱۲۰	۱۵/۹۷۵۱	۱۴/۹۲۷۷	۷۵۵۹/۷۵۱۶

جدول ۳.۴: زمان اجرای هر یک از مسائل با روش‌های به کار رفته

ارائه شده نشان می‌دهد، از آن‌جایی که مساله به صورت یک مساله ماکزیمم سازی فرمول‌بندی شده است پس بهترین مقدار تابع هدف مربوط به روشی است که تابع هدف آن ماکزیمم باشد.

۱۰ مساله آزمون فوق توسط چهار روش ارائه شده، حل شدند، که برای نوشتن برنامه‌های کامپیوتری از نرم‌افزار متلب استفاده کرده‌ایم. همان‌طور که مشاهده می‌شود، در دو مساله اول چون داده‌های مساله، داده‌های کوچکی هستند، در هر چهار روش مقدار تابع هدف بهینه حاصل شده است، با این تفاوت که زمان اجرای برنامه‌ها برای هر روش متفاوت از دیگری است، همان‌گونه که انتظار می‌رفت الگوریتم ترکیبی جست‌وجوی ممنوع زمان زیادی را برای رسیدن به جواب بهینه صرف کرده است، و این امر به این دلیل است که در این روش برای هر جواب تمام همسایه‌ها را پیدا کرده، مقدار تابع هدف آن‌ها را محاسبه می‌کنیم و سپس از بین آن‌ها بهترین را انتخاب می‌کنیم. اگر چه زمان بیشتری در این روش نسبت به بقیه روش‌ها استفاده می شود، اما همان‌گونه که در مسائل آزمون سوم تا دهم مشاهده می‌شود، جواب‌های بهتری در این

روش نسبت به سه روش دیگر به دست آمده است. در GLS2 که از تابع هدف پتانسیل به جای تابع هدف در الگوریتم تکاملی جست‌وجوی موضعی استفاده شده است، در طی زمان کمتری نسبت به GLS1 جواب بهینه حاصل شده است، در صورتی که جواب‌های به دست آمده ازGLS1 به جواب بهینه نزدیک‌تر هستند. به وضوح در سه روش GTS، GLS1 و GLS2 جواب‌های به دست آمده نسبت به الگوریتم ژنتیک جواب‌های بهتری هستند و زمان اجرای برنامه‌ها نیز زمان بیشتری می‌باشد.

پیوست آ
کدهای $MATLAB$

آ.۱ کد $MATLAB$ برای روش جست‌وجوی موضعی ژنتیک

در اینجا کد $MATLAB$ روش ترکیبی جست‌وجوی موضعی و الگوریتم ژنتیک (GLS) ، برای بدست آوردن نتایج عددی بالا قرار داده شده است.

```
IP=GIP(p,m,n,C);
for A=1:5
R=zeros(1,n);
for i=1:n
e=FUN(IP(:,i),C);
R(i)=e;
end
L=max(R);
for i=1:n
if  R(i)~=L
R(i)=0;
else
R(i)=1;
end
end
```

```
ind=Find(R,1,'first');
best1=IP(:,ind);
IP1=zeros(m,n);
for  i=1:n
Y=IP(:,i);
IP1(:,i)=LS(Y,p,m,C);
end
TP=zeros(m,n);
for  i=1:n
FU=zeros(1,n);
for  g=1:n
FU(:,g)=FUN(IP1(:,g),C);
end
S1=sum(FU,2);
for  j=1:n
Y(j)=FUN(IP1(:,j),C)/S1;
end
FT(1)=Y(1);
for  k=2:n
FT(k)=FT(k-1)+Y(k);
end
ab=zeros(1,2);
r=rand;
ab(1)=Find(FT>=r,1,'first');
ab(2)=ab(1);
while ab(2)==ab(1)
r=rand;
ab(2)=Find(FT>=r,1,'first');
end
```

```
b=IP1(:,ab(2));
a=IP1(:,ab(1));
r1=rand;
if  r1<p_{C}
c1=DPX(a,b,p,m);
else
c1=a;
end
r2=rand;
if  r2<p_{M}
c1=Mutation(c1,p,m,C);
end
c1=LS(c1,p,m,C);
TP(:,i)=c1;
end
TP;
R1=zeros(1,m);
for i=1:n
e=FUN(TP(:,i),C);
R1(i)=e;
end
L1=max(R1);
for  i=1:m
if  R1(i)~=L1
R1(i)=0;
else R1(i)=1;
end
end
ind=Find(R1,1,'first');
```

```
best2=TP(:,ind);
pnew=zeros(m,n);
sw=0;
for  i=1:n
for  j=1:n
if  TP(:,i)==IP(:,j)
sw=1;
end
end
if  sw==0
su=0;
for  k=1:n  &  k~=i
if  pnew(:,i)==pnew(:,k)
su=1;
end
end
if  su==0
pnew(:,i)=TP(:,i);
end
if  su==1
pnew(:,i)=IP(:,i);
end
end
if  sw==1
pnew(:,i)=IP(:,i);
end
end
R2=zeros(1,m);
for i=1:n
```

```
e=FUN(pnew(:,i),C);
R2(i)=e;
end
L2=max(R2);
for  i=1:m
if  R2(i)~=L2
R2(i)=0;
else
R2(i)=1;
end
end
ind=Find(R2,1,'first');
best3=pnew(:,ind);
e1=FUN(best1,C);
e2=FUN(best2,C);
e3=FUN(best3,C);
e=[e1 e2 e3];
best4=[best1 best2 best3];
for  i=1:3
if  e(i)==max(e)
best=best4(:,i);
end
end
end
IP=pnew
end
```

مراجع

[1] Balinski, M.L., (1966), *On finding integer solutions to linear programs, Proceedings of IBM Scientific Symposium on Combinatorial Problems*, IBM, White Plains, NY, 225–248.

[2] Bander, J.L., White, C.C., (2002), *A heuristic search approach for a nonstationary stochastic shortest path problem with terminal cost*, Transportation Science, 36(2):218-230.

[3] Bilde, O., Krarup, J., (1977), *Sharp lower bounds and efficient algorithms for the simple plant location problem*, Annals of Discrete Mathematics 1:79–97.

[4] Bouzaiene-Ayari, B., Gendreau, M., Nguyen, S., (2001), *Modeling bus stops in transit networks: A survey and new formulations*, Transportation Science, 35(3):304-321.

[5] Cerny, V., (1985), *Thermodynamical approach to travelling "salesman problem: an efficient simulation algorithm"*, Journal Optimization Theary Applications, 45:41-45.

[6] Cheriyan, J., Ravi, R., (1998), *Aproximation for networks problems*.

[7] Corra, E.S., Steiner, M.T.A., Freitas, A.A., Cornieri, C., (2001), *A genetic algorithm for the p-median problem*, Available in Address: Citeseer.nj.ncg.com/458099.html.

[8] Croes, G.A., (1958), *A method for solving travelling salesman problems*, Operations Research, 6:791-796.

[9] Daskin, M.S., (1995), *Network and discrete location: models, algorithms and applications*, John Wiley, New York.

[10] Dibble, C., Densham, P.J., (1993), *Generating interesting alternatives in GIS and SDSS using genetic algorithms*, GIS/LIS Symposium, University of Nebraska, Lincoln.

[11] Efroymson, M.A., Ray, T.L., (1966), *A branch and bound algorithm for plant location*, Operations Research 14:361–368.

[12] Erkut, E., Bozkaya, B., Zhang, J., (2001), *An effective genetic algorithm for the p-median problem*.

[13] Erlenkotter, D., (1978), *A dual-based procedure for uncapacitated facility location*, Operations Research 14:361–368.

[14] Francis, R., McGinnis, Jr.L.F., White, J.A., (1992), *Facility layout and location: An analytical approach*, Prentice Hall.

[15] Galvao, R.D., Raggi, L.A., (1989), *A method for solving to optimality uncapacitated location problems*, Annals of Operations Research 18:225–244.

[16] Ghanbari, R., Mahdavi-Amiri, N., (2011), *Solving bus terminal location problems using evolutionary algorithm*, Applied Soft Computing 11:991-999.

[17] Ghosh, D., (2003), *Neighborhood search heuristics for uncapacitated facility location problem*, European Journal of Operational Research (150):150-162.

[18] Ginkel, A., Schobel, A., (2007), *To wait or Not to wait? The bicriteria delay management problem in public transportation*, Transportation Science, 41(4):527-538.

[19] Glover, F., (1986), *Future paths for integer programming and links to artificial intelligence*, Computers and Operation Research, 13(5):533-549.

[20] Goldberg, D.E., Deb, K., (1991), *A comparison of selection schemes used in genetic algorithms*, In Rawlins, G., editor, Foundations of Genetic Algorithms, 34:69-93.

[21] Guignard, M., Spielberg, K., (1977), *Algorithms for exploiting the structure of the simple plant location problem*, Annals of Discrete Mathematics 1:247–271.

[22] Handler, G.Y., Mirchandani, P.B., (1979), *Location on networks: theory and algorithms*, MIT Press, Cambridge.

[23] Hansen, P., Meladenovic, N., (1997), *Variable neighborhood search for the p-median*, Location Science, 5:207-226.

[24] Hertz, A., Werra, D., (1991), *The tabu search metaheuristic: How we used it*, Annals of Mathematics and Artificial Intelligence, 1:111-121.

[25] Holland, J.H., (1975), *Adoptation in natural and artificial systems*, Ann Arbor MI: The University of Michigan Press.

[26] Hosage, M., Goodchild, M.F., (1986), *Discrete space location-allocation solutions from genetic algorithms*, Annals of Operational Research, 6:35-46.

[27] Jaszkiewicz, A., Kominek, P., (2003), *Genetic local search with distance preserving recombination operator for a vehicle routing problem*, European Journal of operational Research 151:352-364.

[28] Kariv, O., Hakimi, S.L., (1979), *An algorithmic approach to network location problems. Part II: p-medians*, SIAM Journal of Applications Math, 37:539-560.

[29] Khumawala, B.M., (1972), *An efficient branch and bound algorithm for the warehouse location problem*, Management Science 18:718–731.

[30] Kirkpatrick, S., Gelaltt, C.D., Vecchi, M.P., (1983), *Optimization by simulated annealing*, Science, 220(4598):671-680.

[31] Kolen, A., pesch, E., (1994), *Genetic local search in combinatorial optimization*, discrete Applied Mathematics and combinatorial operations Research and computer science 48:273-284.

[32] Koza, b., John, R., (1991), *Concept formation and decision tree induction using the genetic programming paradigm*, In Schwefel, Hans-Paul, and Maenner, Reinhard (editors). Parallel Problem Solving from Nature. Berlin: Springer-Verlag. 45:124-128.

[33] Krarup, J., Pruzan, P.M., (1983), *The simple plant location problem: Survey and synthesis*, European Journal of Operational Research 12:36–81.

[34] Lapierre, S.D., Ruiz, A.B., (1974), *Facility layout and location: an analytical approach*, Prentice-Hall.

[35] Maranzana, F.E., (1964), *On the location of supply points to minimize transport costs*, Operational Research Quarterly, 15:261-270.

[36] Merz, P., Freisleben, B., (2000), *Fitness location landscape analysis and memetic algorithms for quadratic assignment problem*, IEEE Transaction on Evolutionary computation 4:337-352.

[37] Merz, P., Freisleben, B., (1997), *Genetic local search for the TSP: New results*, in: proceedings of the 1997 IEEE International conference on Evolutionary Computation, IEEE press, New York , 76:159-164.

[38] Metrapolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., (1953), *Equation of state calculation by fast computing Machines*, The Journal of chemical physics, 21(6).

[39] Moreno-Perez, J.A., Moreno-Vega, J.M., Meladenovic, N., (1994), *Tabu search and simulated annealing in p-median problems*, Talk at the Canadian Operational Research Society Conference, Montral.

[40] Reeves, C.R., (1993), *Modern heuristic techniques for combinatorial problems*, Blackwell Scientific Publications, Oxford.

[41] Rolland, E., Schilling, D.A., Current J.R., (1997), *An efficient tabu search heuristic for the p-median problem*, EJOR, 96:329-342.

[42] Tamir, A., (1996), *An $O(pn^{۲})$ algorithm for the p-median and related problems on tree graphs*, Operations Research Letters, 19:59-64.

[43] Teitz, M.B., Bart, P., (1968), *Heuristic methods for estimating generalized vertex median of a weighted graph*, Operations Research, 16:955-961.

واژه‌نامه فارسی به انگلیسی

الگوریتم تکاملی ..... Evolutionary Algorithm

الگوریتم حریصانه ..... Greedy Algorithm

الگوریتم شاخه و کرانه ..... Branch and Bound Algorithm

ایستگاه اتوبوس ..... Bus Station

برازندگی ..... Fitness

بهینه‌سازی ترکیبی ..... Combinatorial Optimization

بهینه‌سازی خطی ..... Linear Optimization

بهینه سراسری ..... Global Optimal

بهینه موضعی ..... Local Optimal

پتانسیل راس ..... Potential of Node

تابع هدف ..... Object Function

تابع برازش ..... Fitness Function

تابع کاهشی ..... Decreasing Function

تابع هدف پتانسیل ..... Potential of Object Function

تخصیص ..... Assignment

تسهیلات ..... Facility

تسهیلات بدون ظرفیت ..... Uncapacitated Facility

تقاطع ..... Crossover

تولید مثل ..... Reproduction

جست‌وجوی ممنوع ..... Tabu Search

جست‌وجوی موضعی ..... Local Search

جست‌وجوی همسایگی ..... Neighborhood Search

جمعیت اولیه ..... Initial Population

جهش ..... Mutation

چرخ رولت ..... Roulette Wheel

حمل و نقل ..... Transport

رتبه‌بندی خطی ..... Linear Ranking

روش‌های ابتکاری ..... Heuristic Methods

روش‌های تقریبی ..... Approximate Methods

روش‌های دقیق ..... Exact Methods

رئوس داوطلب ..... Candidate of Nodes

ژن ..... Gene

ژنتیک ..... Genetic

شرط توقف ..... Stopping Criterion

عملگر انتخاب ..... Selection Operator

کارایی ..... Efficiency

کدگذاری ..... Coding

کروموزوم ..... Chromosome

گام ..... Step

لیست ممنوع ..... Tabu List

ماتریس فاصله ..... Distance Matrix

مرکز سرویس‌دهی ..... Serving Center

مسیریابی ..... Routing

معیار آرمانی ..... Aspiration Criterion

مکان ..... Site

مکان‌یابی ..... Location

مکانیزم ..... Mechanism

نسل ..... Foster

نقاط تقاضا ..... Demand of Nodes

نقاط همسایه ..... Neighborhood of Nodes

نمونه‌گیری ..... Sampling

والدین ..... Parents

هزینه ..... Cost

واژه‌نامه انگلیسی به فارسی

روش‌های تقریبی ..... Approximate Methods

معیار آرمانی ..... Aspiration Criterion

تخصیص ..... Assignment

الگوریتم شاخه و کرانه ..... Branch and Bound Algorithm

ایستگاه اتوبوس ..... Bus Station

رئوس داوطلب ..... Candidate Nodes

کروموزوم ..... Chromosome

کدگذاری ..... Coding

بهینه‌سازی ترکیبی ..... Combinatorial Optimization

هزینه ..... Cost

تقاطع ..... Crossover

تابع کاهشی ..... Decreasing Function

نقاط تقاضا ..... Demand of Node

ماتریس فاصله ..... Distance Matrix

کارایی ..... Efficiency

الگوریتم تکاملی ..... Evolutionary Algorithm

روش‌های دقیق ..... Exact Methods

تسهیلات ..... Facility

برازندگی ..... Fitness

تابع برازش ..... Fitness Function

نسل ..... Foster

ژن ..... Gene

ژنتیک ..... Genetic

بهینه سراسری ..... Global Optimal

الگوریتم حریصانه ..... Greedy Algorithm

روش‌های ابتکاری ..... Heuristic Methods

جمعیت اولیه ..... Initial Population

رتبه‌بندی خطی ..... Linear Ranking

بهینه‌سازی خطی ..... Linear Optimization

بهینه موضعی ..... Local Optimal

جست‌وجوی موضعی ..... Local Search

مکان‌یابی ..... Location

مکانیزم ..... Mechanism

جهش ..... Mutation

نقاط همسایه ..... Neighborhood of Node

جست‌وجوی همسایگی ..... Neighborhood Search

تابع هدف ..... Object Function

والدین ..... Parents

پتانسیل راس ..... Potential of Node

تابع هدف پتانسیل ..... Potential Object Function

تولیدمثل ..... Reproduction

چرخ رولت ..... Roulette Wheel

مسیریابی ..... Routing

نمونه‌گیری ..... Sampling

عملگر انتخاب ..... Selection Operator

مرکز سرویس‌دهی ..... Serving Center

مکان ..... Site

گام ..... Step

شرط توقف ..... Stopping Criterion

لیست ممنوع ..... Tabu List

جست‌وجوی ممنوع ..... Tabu Search

حمل و نقل ..... Transport

تسهیلات بدون ظرفیت ..... Uncapacitated Facility

Surname: Khorasani Name: Fatemeh

Title: Bus terminal location problem

Supervisor: Dr.Jafar fathali

Advisor: Dr.Barat Allah Ghaznavi ghosoni

Degree: Master of Science Subject: Applied Mathematics

Field: Operations Research

Shahrood University of Technology Faculty Of Mathematical Sciences

Date: September 2013 Number of pages: 90

Keywords: Bus Terminal Location, Combinatorial Optimization, Genetic Algorithm, Local Search, Tabu Search

Abstract

In this thesis, a combinatorial and heuristic method is stated for solving bus terminal location problem. In this method, advantages of Genetic algorithm and local search evolutionary algorithm are combined. Then, a new method is presented for solving bus station location problem. In this method, tabu search evolutionary algorithm is used instead of local search algorithm. Finally, numerical results achieved of these two methods are compared and the best method is introduced.

Shahrood University of Technology
Faculty Of Mathematical Sciences

Dissertation Submitted in Partial
Fulfillment of The Requirements For The
Degree of Master of Science in

Applied Mathematics

Bus terminal location problem

Supervisor

Dr.Jafar fathali

Advisor

Dr.Barat Allah Ghaznavi ghosoni

by

Fatemeh Khorasani

September 2013