بسم الله الرحمن الرحيم

دانشکده ریاضی و علوم کامپیوتر

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

حل مسائل مکان‌یابی با استفاده از روش‌های فراابتکاری

استاد راهنما:

آقای دکتر مهدی زعفرانیه

استاد مشاور:

آقای دکتر محمود امین‌طوسی

پژوهشگر:

سمیرا شاهی

شهریور ۱۳۹۲

باسمه تعالی

صورتجلسه‌ی دفاع از پایان‌نامه کارشناسی ارشد

با تلاوت آیاتی چند از کلام الله مجید جلسه دفاع از پایان‌نامه **خانم سمیرا شاهی** دانشجوی رشته ریاضی کاربردی با عنوان « **حل مسائل مکان‌یابی با استفاده از روش‌های فراابتکاری** » در ساعت ۱۲ مورخ ۹۲/۷/۱۴ در محل **دانشکده ریاضی و علوم کامپیوتر** تشکیل گردید.

پس از استماع گزارش ارائه شده توسط دانشجو و استاد راهنما، هیأت داوران و حاضران سؤالاتی را مطرح و **خانم سمیرا شاهی** دفاع از موضوع پرداخت و به سؤالات آن‌ها پاسخ گفت.

سپس پایان‌نامه توسط هیأت داوران مورد ارزشیابی قرار گرفت و نمره **نوزده** برابر درجه **عالی** برای آن تعیین گردید.

به این ترتیب ضمن تصویب پایان‌نامه مزبور از این تاریخ **خانم سمیرا شاهی** به عنوان کارشناس ارشد در رشته **ریاضی کاربردی** شناخته می‌شود.

ردیف	سمت	نام و نام خانوادگی	مرتبه دانشگاهی	دانشگاه یا مؤسسه	امضـــــــا
۱	استاد راهنما	دکتر مهدی زعفرانیه	استادیار	دانشگاه حکیم سبزواری	
۲	استاد مشاور	دکتر محمود امین‌طوسی	استادیار	دانشگاه حکیم سبزواری	
۳	استاد داور	دکتر امین رفیعی	استادیار	دانشگاه حکیم سبزواری	
۴	نماینده تحصیلات تکمیلی	خانم دکتر عباس‌نژاد	استادیار	دانشگاه حکیم سبزواری	

نام و نام خانوادگی و امضای مدیر گروه:

سوگند نامه دانش آموختگان دانشگاه حکیم سبزواری

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

نام و نام خانوادگی: سمیرا شاهی

تاریخ و امضا:

باسمه تعالی

تأییدیه‌ی صحت و اصالت نتایج

اینجانب سمیرا شاهی به شماره دانشجویی ۹۰۱۳۱۳۳۰۴۶ دانشجوی رشته ریاضی کاربردی مقطع تحصیلی کارشناسی ارشد تأیید می‌نمایم که کلیه‌ی نتایج این پایان‌نامه حاصل کار اینجانب و بدون هرگونه دخل و تصرف است و موارد نسخه‌برداری‌شده از آثار دیگران را با ذکر کامل مشخصات منبع ذکر کرده‌ام. درصورت اثبات خلاف مندرجات فوق، به تشخیص دانشگاه مطابق با ضوابط و مقررات حاکم (قانون حمایت از حقوق مؤلفان و مصنفان و قانون ترجمه و تکثیر کتب و نشریات و آثار صوتی، ضوابط و مقررات آموزشی، پژوهشی و انضباطی ...) با اینجانب رفتار خواهد شد و حق هرگونه اعتراض درخصوص احقاق حقوق مکتسب و تشخیص و تعیین تخلف و مجازات را از خویش سلب می‌نمایم. در ضمن، مسؤولیت هرگونه پاسخگویی به اشخاص اعم از حقیقی و حقوقی و مراجع ذی‌صلاح (اعم از اداری و قضایی) به عهده‌ی اینجانب خواهد بود و دانشگاه هیچ‌گونه مسؤولیتی در این خصوص نخواهد داشت.

نام و نام خانوادگی: سمیرا شاهی

تاریخ و امضا:

مجوز بهره‌برداری از پایان‌نامه

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

☐ بهره‌برداری از این پایان‌نامه برای همگان بلامانع است.

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

☐ بهره‌برداری از این پایان‌نامه تا تاریخ ..... ممنوع است.

استاد راهنما: آقای دکتر مهدی زعفرانیه

تاریخ:

امضا:

با عرض ادب به پیشگاه چهارده معصوم (ع)

ماحصل آموخته‌هایم را تقدیم می‌کنم به:
استوارترین تکیه‌گاهم

دستان پر مهر پدرم

و به سبزترین نگاه زندگیم

چشمان سبز مادرم

ره‌آوردی گران‌سنگ‌تر از این ارزان نداشتم تا به خاک پای‌تان نثار کنم، باشد که حاصل تلاشم نسیم‌گونه، غبار خستگی‌تان را بزداید.

بوسه بر دستان پر مهرتان

قدردانی

سپاس خداوندگار حکیم را که با لطف بی‌کران خود، آدمی را زیور عقل آراست.

در آغاز از زحمات استاد راهنما و استاد مشاور خود، جناب آقای دکتر مهدی زعفرانیه و جناب آقای دکتر محمود امین‌طوسی که زحمت مطالعه و مشاوره این رساله را تقبل فرمودند و در آماده سازی این رساله، به نحو احسن اینجانب را مورد راهنمایی قرار دادند، کمال امتنان را دارم.

همچنین لازم می‌دانم از پدید آورندگان بسته زی‌پرشین، مخصوصاً جناب آقای وفا خلیقی، که این پایان‌نامه با استفاده از این بسته، آماده شده است و همه دوستانمان در گروه پارسی‌لاتک کمال قدردانی را داشته باشم.

در پایان، بوسه می‌زنم بر دستان خداوندگاران مهر و مهربانی، پدر و مادر عزیزم و بعد از خدا، ستایش می‌کنم وجود مقدس‌شان را و تشکر می‌کنم از خانواده عزیزم به پاس عاطفه سرشار و گرمای امیدبخش وجودشان، که بهترین پشتیبان من بودند.

دانشگاه حکیم سبزواری

فرم چکیده‌ی پایان‌نامه‌ی دوره‌ی تحصیلات تکمیلی

مدیریت تحصیلات تکمیلی

نام خانوادگی دانشجو: شاهی	**نام:** سمیرا	**شماره دانشجویی:** ۹۰۱۳۱۳۳۰۴۶
استاد راهنما: دکتر مهدی زعفرانیه	**استاد مشاور:** دکتر محمود امین‌طوسی	
دانشکده: ریاضی و علوم کامپیوتر	**رشته:** ریاضی کاربردی	**گرایش:** تحقیق در عملیات
مقطع: کارشناسی ارشد	**تاریخ دفاع: ۱۳۹۲/۷/۱۴**	**تعداد صفحات:۵۵**
عنوان پایان‌نامه: حل مسائل مکان‌یابی با استفاده از روش‌های فراابتکاری		
کلید واژه‌ها: مکان‌یابی، جستجوی ممنوع، الگوریتم ژنتیک، الگوریتم ازدحام ذرات .		

چکیده:

مسأله مکان‌یابی مراکز با ظرفیت نامحدود، شامل مکان‌یابی یک تعداد از اماکن سرویس‌دهنده با ظرفیت نامحدود برای کمینه کردن مجموع هزینه‌های ثابت و هزینه‌های متغیر سرویس‌دهی از این اماکن به محل تقاضاها می‌باشد.

در این پایان‌نامه، مسأله مکان‌یابی مراکز با ظرفیت نامحدود که از مسائل NP-hard است را با سه الگوریتم فراابتکاری جستجوی ممنوع، ژنتیک و ازدحام ذرات حل کردیم.

با توجه به کارایی زیادی که الگوریتم‌های فراابتکاری برای حل مسائل پیچیده و بزرگ، در زمان‌های کوتاه با جواب‌هایی با کیفیت بالا دارند، از این الگوریتم‌ها برای حل این مسأله استفاده کردیم.

هدف اصلی پایان‌نامه، مقایسه میانگین زمان تا همگرایی، میانگین تعداد تکرار تا همگرایی، میانگین خطا و فرکانس همگرایی سه روش فوق در حل مسأله مکان‌یابی است، طبق نتایج بدست آمده الگوریتم ژنتیک کارایی بیشتری در حل این مسأله دارد.

فهرست مطالب

فهرست تصاویر

فهرست جداول

فهرست الگوریتم‌ها

پیش‌گفتار

در اکثر رشته‌ها و شاخه‌های علوم، از قبیل مهندسی، ریاضیات، فیزیک، شیمی و ... مسائلی دیده می‌شود که ماهیتی از نوع تصمیم‌گیری دارند. در هر مسأله تصمیم‌گیری، تصمیم‌گیرنده قصد انتخاب بهترین گزینه را با توجه به مجموعه‌ای از اهداف و بهینه‌سازی مسأله دارد.

برای یافتن بهترین راه‌حل یک مسأله، مجموعه‌ای از محاسبات در قالب معادلات ریاضی و الگوریتم‌های کامپیوتری باید انجام شود. محاسبات مورد نیاز به قدری زیاد است که حتی با کامپیوترهای پیشرفته امروزی نیز ممکن است زمان زیادی برای حل آن‌ها لازم باشد [۴]. در سال‌های اخیر شاهد حرکتی مستمر در زمینه پردازش اطلاعات، برای حل مسائلی که راه حل تحلیلی برای آن‌ها وجود ندارد، بوده‌ایم. انتخاب یک روش صحیح که سرعت خوبی نیز در ارائه جواب داشته باشد، می‌تواند تأثیر قابل ملاحظه‌ای در زندگی صنعتی و اجتماعی داشته باشد[۲].

مکان‌یابی یک مسأله مهم در علم مدیریت و تخصیص منابع می‌باشد و این مسائل در اندازه‌های کاربردی و عملی خود به قدری بزرگ هستند که نمی‌توان جواب بهینه آن‌ها را در مدت زمان قابل پذیرش بدست آورد، با این وجود این مسائل باید حل شوند. الگوریتم‌های فراابتکاری، جواب‌هایی با کیفیت بالا را در زمان کوتاه برای مسائل بهینه‌سازی پیچیده ارائه می‌دهند.

این پایان‌نامه شامل ۳ فصل است:

در فصل اول، به معرفی روش‌های فراابتکاری و جستجوی ممنوع می‌پردازیم.

در فصل دوم، الگوریتم ژنتیک و الگوریتم ازدحام ذرات را معرفی می‌نماییم.

در فصل سوم، مسأله مکان‌یابی بدون ظرفیت را تعریف می‌کنیم، سپس روش حل این مسائل با استفاده از روش های فراابتکاری معرفی شده در فصل اول و دوم بیان می‌شود و نتایج بدست آمده از سه روش را مقایسه می‌نماییم.

ایده‌ی اصلی پایان‌نامه از مقاله‌ی زیر گرفته شده است.

Michel L, Hentenryck PV. A Simple Tabu Search Warehouse Location. European Journal of

Operational Research, 2004;157: 576-591.

فصل ۱

معرفی روش‌های فراابتکاری و روش جستجوی ممنوع

۱-۱ مقدمه

محاسبه راه‌حل‌های بهینه برای اکثر مسائل بهینه‌سازی که در خیلی از زمینه‌های کاربردی و عملی مشاهده می‌گردند، کاری سخت و دشوار است. مثلاً مسیر کامیون‌های حمل ونقل باید تعیین شود، انبارها یا نقاط فروش محصولات باید جایابی شوند، شبکه‌های ارتباطی باید طراحی شوند [۲]. این مسائل در اندازه‌های کاربردی و عملی خود به قدری بزرگ هستند که نمی‌توان جواب بهینه آن‌ها را در مدت زمان قابل پذیرش بدست آورد.

۱-۱-۱ روش‌های بهینه‌سازی

دو دسته کلی از روش‌های حل مسائل بهینه‌سازی روش‌های دقیق[۱] و روش‌های تقریبی[۲] می‌باشد.

تعریف ۱-۱-۱. روش‌های دقیق؛ الگوریتم‌هایی هستند که قادر به یافتن جواب بهینه به صورت دقیق هستند یعنی جواب‌های بهینه را پیدا کرده و شرط بهینگی را تضمین می‌نمایند، بعضی از این الگوریتم‌ها عبارتند از برنامه‌ریزی پویا[۳]، الگوریتم شاخه و کران[۴].

تعریف ۱-۱-۲. روش‌های تقریبی؛ قادر به یافتن جواب‌های خوب (نزدیک بهینه) در زمان معقول برای مسائل بهینه‌سازی NP-hard[۵] هستند و به دو دسته روش‌های تقریب‌زنی و فراابتکاری[۶] تقسیم می‌شوند.

---

[۱] Exact [۲] Approximate [۳] Dynamic Programming [۴] Branch and Bound [۵] Nondeterministic Polynomial-time hard [۶] Metaheuristic

تعریف ۱-۱-۳. مسائل **NP-hard**؛ مسائل بهینه‌سازی هستند که الگوریتم مؤثر اثبات شده‌ای برای حل آن‌ها وجود ندارد و زمان حل بهینه آن‌ها، در رده زمان نمایی می‌باشد. در زیر دسته‌ای از این مسائل آورده شده است:

- مسائل زمان‌بندی
- مسائل تخصیص
- مسائل مسیریابی

۱-۱-۲ روش‌های تقریب‌زنی

در الگوریتم‌های تقریب‌زنی، تضمینی برای حدود جواب بهینه سراسری وجود دارد. یک الگوریتم تقریب‌زنی ϵ، یک راه‌حل تقریبی a را به نحوی تولید می‌نماید که به اندازه فاکتور ϵ بار از راه‌حل بهینه سراسری کمتر نباشد. از جمله‌ی این روش‌ها می‌توان جستجوی محلی را نام برد.

۱-۱-۳ روش‌های فراابتکاری

الگوریتم‌های فراابتکاری به دلیل استفاده از دو استراتژی تنوع و تشدید جواب‌هایی با کیفیت بالا در زمانی کوتاه برای مسائل بهینه‌سازی پیچیده ارائه می‌دهند. هر چند ضمانتی برای دست‌یابی به جواب بهینه با استفاده از این الگوریتم‌ها وجود ندارد، ولی توانایی بالای آن‌ها در دست‌یابی به جواب‌های نزدیک بهینه در زمان کوتاه برای این مسائل، موجب شهرت فراوان آن‌ها شده است.

دسته بندی الگوریتم‌های فراابتکاری

- الهام گرفته از طبیعت: خیلی از الگوریتم‌های فراابتکاری از فرایندهای طبیعی گرفته شده‌اند، از قبیل: کلونی مورچه‌ها[۷] و کلونی زنبوران[۸]، الگوریتم ژنتیک[۹] .

  تعریف ۱-۱-۴. حافظه؛ محلی برای ذخیره اطلاعات می‌باشد.

- با حافظه و بدون حافظه: برخی از این الگوریتم‌ها فاقد حافظه[۱۰] هستند، به این معنا که این نوع الگوریتم‌ها از اطلاعات بدست آمده در حین جستجو استفاده نمی‌کنند مانند شبیه‌سازی تبریدی[۱۱]. این در حالی است که در برخی نظیر جستجوی ممنوع[۱۲] از حافظه استفاده می‌کنند، این حافظه اطلاعات بدست آمده در حین جستجو را در خود ذخیره می‌کند.

---

[۷] Ant Colonies [۸] Bee Colonies [۹] Genetic Algorithm [۱۰] Memory [۱۱] Simulated Annealing [۱۲] Tabu Search

- مبتنی بر یک جواب و مبتنی بر جمعیت: الگوریتم‌های مبتنی بر یک جواب در حین فرایند جستجو، یک جواب را تغییر می‌دهند مانند جستجوی ممنوع. در الگوریتم‌های مبتنی بر جمعیت، در حین جستجو، یک جمعیت از جواب‌ها در نظر گرفته می‌شود و در طول فرایند جستجو آن‌ها را تغییر می‌دهند مانند الگوریتم ژنتیک.

انتخاب و به کارگیری الگوریتم‌های فراابتکاری، وابستگی شدیدی به شرایط مورد بحث، نوع متغیرها، گسترش و بزرگی مسأله، شرایط و محیط به کارگیری آن دارد. اشتباه در به کارگیری این ابزارها هزینه و زمان زیادی تلف خواهد کرد. بنابراین باید در انتخاب این ابزارها دقت زیادی به خرج داد، شرایط مسأله را به صورت دقیق بررسی کرد و در آخر از مشاوره‌ای قوی برای انتخاب ابزارها استفاده کرد [۲].

۲-۱ جستجوی ممنوع (TS)

جستجوی ممنوع یکی از روش‌های بهینه‌یابی فراابتکاری برای حل مسائل بهینه‌سازی ترکیبی[۱۳] است. ریشه بحث جستجوی ممنوع به اواخر دهه ۱۹۶۰ و اوایل دهه ۱۹۷۰ برمی‌گردد، ولی شکل اصلی این روش در سال ۱۹۸۶ توسط گلور[۱۴] [۱۴] ارائه شد و این نام از روی ایده‌ای که سال‌ها پیش وی برای طراحی رویکرد حل مسأله در هوش مصنوعی و بهینه‌سازی بدست آورده بود ریشه می‌گیرد.

کلمه تابو از زبان تونگا[۱۵] که یک زبان پلینیزی[۱۶] است گرفته شده است. تابو در این زبان برای اطلاق به مواردی به کار می‌رود که به دلیل مقدس بودن نباید به آن‌ها دست زد. همچنین بر اساس لغت‌نامه وبستر[۱۷] این کلمه به معنای چیز منع شده توسط عرف جامعه به عنوان معیار مصونیت و یا چیزی غدغن شده به خاطر در بر داشتن نوعی ریسک است [۲].

اساس نام‌گذاری این روش، استفاده از لیستی به نام لیست ممنوع[۱۸] است. این الگوریتم از یک جواب اولیه تصادفی شروع کرده و به جستجوی همسایگی می‌پردازد، در بین همسایه‌ها، بهترین را انتخاب می‌کند و به آن حرکت می‌نماید. در هنگام حرکت از یک نقطه به نقطه‌ی دیگر، مشخصاتی از حرکت در لیست ممنوع ثبت می‌شود. الگوریتم، این جستجو را تا زمانی ادامه می‌دهد که یک معیار توقف برآورده گردد [۴] .

استراتژی‌های استفاده شده توسط جستجوی ممنوع می‌توانند به دو صورت تمرکزدهی[۱۹] و تنوع‌بخشی[۲۰] طبقه بندی شوند. استراتژی تمرکزدهی به گونه‌ای است که به وسیله اصلاح قوانین انتخاب، حرکاتی که در طی فرایند خوب تشخیص داده شده‌اند را مورد توجه قرار می‌دهد. این استراتژی ممکن است روش حل را مجدداً به نواحی با جذابیت بیشتر هدایت کند تا جستجوی دقیق‌تری صورت پذیرد و در بسیاری از پیاده‌سازی‌های روش

[۱۳] Combinatorial Optimization Problems [۱۴] Glover [۱۵] Tongan [۱۶] Polynesia [۱۷] Webester
[۱۸] Tabu List [۱۹] Intensification [۲۰] Diversification

تابو استفاده می‌شود، اما همیشه ضروری نیست، زیرا حالت‌های بسیاری وجود دارد که در آن‌ها جستجوی معمولی کفایت می‌کند.

روش‌های جستجو جهت افزایش کارایی خود در جستجوی فضای جواب مسائل بهینه‌سازی ترکیباتی، معمولاً از استراتژی‌های تنوع بخشی استفاده می‌کنند. این استراتژی برای هدایت جستجو به مناطق جدید طراحی شده است (تصادفی بودن جواب یک نوع تنوع است) [۱۴].

کاربردهای جستجوی ممنوع

این روش تاکنون در زمینه‌های مختلفی به کار گرفته شده است و این تنوع کاربرد به دلیل انعطاف‌پذیری بالای این روش می‌باشد. جدول ۱-۱ برخی از این کاربردها را نشان می‌دهد.

جدول ۱-۱: کاربردهای جستجوی ممنوع [۲]

ارتباطات	زمانبندی
مسیریابی خطوط تلفن شبکه‌های نوری طراحی شبکه برای سرویس‌دهی تخصیص مسیر برنامه‌ریزی تخفیف مشتری	زمان‌بندی کلاس درس زمان‌بندی جریان زمان‌بندی در کارگاه زمان‌بندی ماشین برنامه‌ریزی نیروی انسانی
بهینه‌سازی عمومی	طراحی
برنامه‌ریزی صفر و یک برنامه‌ریزی غیرخطی غیر محدب	طراحی شبکه حمل‌ونقل طراحی شبکه با هزینه ثابت
تولید، انبار و سرمایه‌گذاری	مکان‌یابی و واگذاری
سیستم‌های تولیدی انعطاف پذیر سرمایه‌گذاری مجموعه ثابت	استخراج نفت در دریا مکان‌یابی و واگذاری چند معیاره
بهینه‌سازی گراف	منطق و هوش مصنوعی
رنگ کردن گراف مسائل p-میانه افراز گراف	منطق احتمالی دسته‌بندی شبکه عصبی/ آموزش و طراحی
تکنولوژی	مسیریابی
لرزه‌نگاری	فروشنده دوره‌گرد

۱-۲-۱ فضای جستجو و ساختار همسایگی

دو رکن اساسی جستجوی ممنوع، فضای جستجو[۲۱] و ساختار همسایگی[۲۲] است.

تعریف ۱-۲-۱. فضایی از همه جواب‌های ممکن است که در طی جستجو می‌تواند در نظر گرفته شود، **فضای جستجو** نامیده می‌شود.

اکنون فرض کنید S فضای جستجو باشد، هر $s \in S$ به یک همسایگی وابسته شده است که با $N(s)$ نشان داده می‌شود:

تعریف ۱-۲-۲. مجموعه جواب‌هایی که با یک انتقال یا حرکت از جواب sبدست می‌آید، **همسایگی** نامیده می‌شود.

مثال ۱-۲-۳. اگر S مجموعه ای از بردارهای صفر و یک و $s \in S$ باشد، یک مجموعه همسایگی $N(s)$ برای s، مجموعه جواب‌های $s \in S$ است که با تغییر وضعیت تنها یکی از مؤلفه‌ها از صفر به یک یا بالعکس بدست آمده‌اند. مثلا اگر $s = (۰, ۰, ۱, ۰)$ باشد آنگاه:

$$N(s) = \{(۱, ۰, ۱, ۰)(۰, ۱, ۱, ۰)(۰, ۰, ۰, ۰)(۰, ۰, ۱, ۱)\}$$

۱-۲-۲ جستجوی محلی

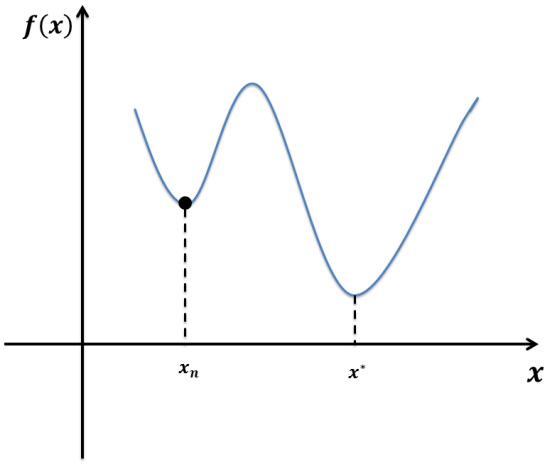
منظور از جستجوی محلی حرکت از جوابی به جواب دیگر در همسایگی آن در فضای جواب مسأله است. نقطه ضعف عمده روش‌های جستجوی محلی ناتوانی آن‌ها در گریز از جواب‌های بهینه محلی است. به مثال ۱-۲-۴ توجه کنید.

مثال ۱-۲-۴. مسأله حداقل‌سازی تابع $f(x)$روی یک مجموعه محدود از نقاط را در نظر بگیرید. مشکل گیر کردن در بهینه محلی، زمانی اتفاق می‌افتد که الگوریتم به نقطه بهینه محلی (x_n) برسد. در این نقطه الگوریتم به جستجوی همسایگی پرداخته و یک همسایه را انتخاب می‌نماید.

زمانی که الگوریتم در بهینه محلی قرار دارد، نقاط همسایه آن دارای مقدار تابع 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 جستجوی همسایگی می‌پردازد و با توجه به اینکه بهینه محلی قبلی در همسایگی نقطه جدید قرار دارد مجدداً به بهینه محلی برمی‌گردد. در این حالت الگوریتم در یک دور افتاده است و از آن خارج نمی‌شود، این وضعیت به گیر کردن در بهینه محلی معروف است. شکل ۱-۱ اسیر شدن الگوریتم در بهینه محلی را نشان می‌دهد [۴].

---

[۲۱] Search Space [۲۲] Neighborhood Structure



شکل ۱-۱: اسیر شدن الگوریتم در جواب بهینه محلی (x_n)

الگوریتم ۱-۱ شبه کد کلی روش جستجوی محلی.

ورودی: S و تابع f

خروجی: مینیمم تابع f

۱: برای شروع $s \in S$ انتخاب کنید.

۲: $s' \in N(s)$ پیدا کنید که $f(s') < f(s)$.

۳: اگر s' پیدا نشد، s بهینه محلی است و جستجو متوقف می‌شود.

۴: در غیر این صورت s' جایگزین s می‌شود سپس به گام دو بروید.

جستجوی ممنوع یک روش جستجوی هدایت شده است و با استفاده از ساختار حافظه و استراتژی‌های خاص سعی در اصلاح معایب روش فوق را دارد.

۱-۲-۳ حافظه کوتاه مدت

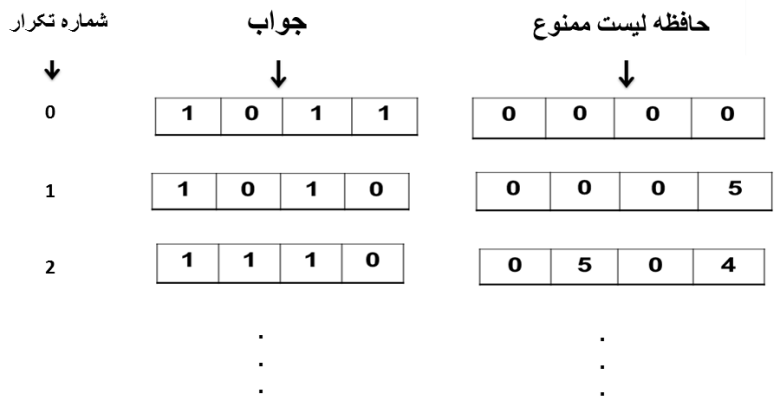
حافظه کوتاه مدت نوعی از جستجو را جهت یافتن بهترین جواب‌ها تشکیل می‌دهد و می‌توان این‌گونه بیان کرد که هسته‌ی اصلی جستجوی ممنوع در فرایند حافظه کوتاه مدت مجسم می‌شود. در این حافظه جوابی که قبلاً دیده شده است ممکن است دوباره با یک همسایگی متفاوت دیده شود [۱۴].

۱. **لیست ممنوعه:** برای جلوگیری از دوری شدن در روش جستجوی ممنوع، از لیست ممنوع استفاده می‌شود که حرکت یا خصوصیاتی از حرکت‌های ممنوع در آن ثبت شده است. طول این لیست می‌تواند ثابت یا متغیر باشد، لیست ممنوع با طول ثابت همیشه نمی‌تواند از دور جلوگیری کند. اندازه لیست ممنوع نیز بسیار مهم است، اگر مقدار آن کوچک باشد ممکن است از دور جلوگیری نشود و اگر اندازه‌ی آن بزرگ

باشد ممکن است فضای جستجو محدود شود. نحوه‌ی شکل‌گیری لیست ممنوع را در مثال ۱-۲-۵ بیان می‌کنیم.

مثال ۱-۲-۵. فرض کنید جواب مسأله‌ای فقط شامل صفر و یک است و لیست ممنوع نیز در ابتدا صفر باشد، مدت ممنوع بودن هر تغییر نیز تا ۵ تکرار است. حرکت جواب جاری به جواب همسایه با تغییر یکی از عنصرهای جواب جاری از صفر به یک یا برعکس تعریف می‌شود. بعد از تغییر، عنصر متناظر با آن مؤلفه در حافظه‌ی لیست ممنوع، عدد ۵ است و در هر تکرار یک واحد از آن کاهش می‌یابد.

همان‌طور که در شکل ۱-۲ می‌بینید عنصر چهارم جواب جاری در تکرار اول از یک به صفر تغییر یافته است، بنابراین عنصر چهارم از حافظه لیست ممنوع عدد ۵ است و در تکرار دوم یک واحد کاهش پیدا می‌کند.



شکل ۱-۲: نحوه‌ی شکل‌گیری لیست ممنوع

۲. **اعتبار تابو**[۲۳]: تعداد تکرارهایی که یک مشخصه به صورت فعال-ممنوع باقی می‌ماند توسط اعتبار تابو تعیین می‌گردد و می‌تواند ثابت یا متغیر باشد.

۳. **لیست کاندید**[۲۴]: به طور کلی در نظر گرفتن کل همسایگی‌ها، جواب‌های با کیفیت بالا تولید خواهد کرد ولی از طرف دیگر ممکن است از نظر زمان حل خیلی گران تمام شده و باعث کندی پیشرفت حل مسأله شود.

لیست کاندید ابزاری برای کاهش میزان همسایگی‌هاست به نحوی که بتوان با طراحی مؤثر زیر مجموعه‌ای از حرکات نوید بخش را در هر تکرار جستجوی ممنوع، آزمایش کرد. زمانی که تعداد

[۲۳] Tabu Tenure [۲۴] Candidate List

کل حرکات قابل انجام کوچک باشد و از لحاظ ارزیابی اصطلاحاً گران نباشد، لیست کاندید می‌تواند شامل تمام حرکات قابل انجام حاضر باشد، در غیر این صورت مجموعه‌ای از حرکات در لیست کاندید انتخاب می‌شود تا بتوان مطلوب‌ترین حرکات را انجام داد. جهت ایجاد یک لیست کاندید برای مسأله، نمونه‌برداری تصادفی از مجموعه‌ای از جواب‌ها را می‌توان نام برد.

۴. **معیار آرمانی**[۲۵]: بخش مهمی از انعطاف‌پذیری در الگوریتم تابو بوسیله معیار آرمانی تعریف می‌شود. ممنوع بودن یک جواب امری مطلق نیست و در صورتی که شرایط خاصی اتفاق افتد، می‌تواند نادیده گرفته شود یعنی جواب ممنوعی که از تمام جواب‌هایی که تاکنون یافت شده بهتر است، می‌تواند موجه در نظر گرفته شود [۲].

۱-۲-۴ معیارهای توقف الگوریتم

معیارهای توقف الگوریتم می‌تواند شامل موارد زیر باشد:

۱- توقف بعد از تکرار مشخص به عنوان مثال پس از ۵۰۰ یا ۱۰۰۰ تکرار و یا پس از کارکرد cpu به اندازه مدت زمان مشخص.

۲- پس از تعدادی تکرار بدون اینکه جواب تابع هدف تغییر کند.

۳- وقتی که جواب به مقداری از پیش تعیین شده برسد.

۱-۲-۵ الگوریتم جستجوی ممنوع

ساختار ساده جستجوی ممنوع، در هنگام استفاده برای حل مسائل بهینه‌سازی، نیاز به تنظیمات خاصی دارد. در صورتی که این تنظیمات به خوبی انجام گردد، الگوریتم دارای کیفیت بالایی خواهد بود. از جمله‌ی این تنظیمات می‌توان ساختار همسایگی، نوع و نحوه طراحی لیست ممنوع را نام برد.

همان‌طور که قبلاً اشاره شد، این الگوریتم با یک جواب اولیه شروع می‌شود. در اکثر موارد این جواب به صورت تصادفی انتخاب می‌شود. بعد از این مرحله، مقدار تابع هدف برای جواب اولیه محاسبه می‌شود. در شروع الگوریتم مقدار بدست آمده از این جواب، به‌عنوان مقدار بهینه در نظر گرفته می‌شود. اکنون الگوریتم به جستجوی همسایه‌های راه‌حل جاری می‌پردازد و کلیه همسایه‌ها را مورد بررسی قرار می‌دهد.

در بین همسایه‌ها، بهترین جواب انتخاب می‌شود. اگر این حرکت در لیست ممنوع نباشد، از جواب جاری به جواب جدید حرکت انجام می‌شود. بعد از هر حرکت، مشخصه حرکت جدید وارد لیست ممنوع خواهد شد

[۲۵] Aspiration Criteria

و این لیست بهنگام می‌شود. در طول الگوریتم همواره بهترین جواب ذخیره می‌گردد. در ابتدا اولین جواب، به‌عنوان بهترین جواب در نظر گرفته می‌شود و هر زمان که حرکتی انجام شود، جواب جدید با بهترین جوابی که تا آن لحظه بدست آمده است، مقایسه می‌گردد. در صورتی که جواب جدید بهتر باشد، به‌عنوان بهترین جواب ذخیره می‌گردد و در غیر این‌صورت، بهترین جواب تغییری نمی‌کند. در صورتی که شرط توقف برآورده شود، الگوریتم پایان می‌یابد و در غیر این‌صورت الگوریتم دوباره به جستجوی همسایگی جواب جاری می‌پردازد [۴].

الگوریتم ۱-۲ شبه کد روش جستجوی ممنوع.

ورودی: S و تابع f

خروجی: مینیمم تابع f

۱: یک جواب اولیه s در S انتخاب کنید و قرار دهید: $s^* = s$ ، $f^* = f(s)$، در این گام لیست ممنوع تهی است $(T = \varnothing)$.

۲: تا زمانیکه شرط خاتمه برقرار شود گام‌های زیر را انجام دهید:

۳: بهترین s' را در مجموعه $(N(s) - T)$ بیابید و قرار دهید $s = s'$.

۴: اگر $f(s') < f(s)$ سپس $s^* = s'$.

۵: لیست ممنوع بهنگام شود.

۱-۲-۶ حل دو مسأله با استفاده از روش جستجوی ممنوع

در این مرحله برای روشن‌تر شدن مراحل روش جستجوی ممنوع دو مثال را به این روش حل می‌کنیم.

مثال ۱-۲-۶. یک مسأله زمان‌بندی[۲۶] ساده را با یک ماشین و ۴ کار j_1، j_2، j_3، j_4 در نظر بگیرید [۲]. هدف این مسأله کمینه کردن مجموع تأخیرها است. داده‌های این مسأله به شرح زیر می‌باشند:

جدول ۱-۲: داده‌های مربوط به مسأله نمونه زمان‌بندی

کارها(j_j)	۱	۲	۳	۴
زمان انجام کار به دقیقه (p_i)	۱۰	۱۰	۱۳	۴
مهلت تحویل کار به دقیقه (d_i)	۴	۲	۱	۱۲
وزن کار (W_i)	۱۴	۱۲	۱	۱۲

هر جواب از این مسأله به شکل یک بردار چهارتایی از ارقام ۱، ۲، ۳، ۴ نشان داده می‌شود. مثلاً اگر

[۲۶] Scheduling

به ترتیب کارهای j_3، j_2، j_1 و در انتها نیز j_4 انجام شوند، جواب مربوط به شکل $(4, 1, 2, 3^{\rightarrow})$ نشان داده می‌شود.

اگر تأخیر نظیر j_j را T_i بنامیم، تابع هدف به شکل زیر محاسبه می‌شود:

$$F = \sum_i W_i T_i$$

برای طراح مسأله مهم است که تابع هدف مقداری کمتر از ۴۱۰ داشته باشد. لذا از همین موضوع می‌توان به عنوان شرط توقف استفاده کرد. طول فهرست ممنوع نیز برابر ۲ داده شده است.

تکرار ۱: جواب دلخواهی مثل $(3, 4, 1, 2)$ را به عنوان جواب اولیه x_1 انتخاب می‌کنیم:

$$T_2 = p_2 - d_2 = ۱۰ - ۲ = ۸ \Rightarrow W_2 T_2 = ۹۶$$

$$T_1 = ۲۰ - ۴ = ۱۶ \quad \Rightarrow W_1 T_1 = ۲۲۴$$

$$T_4 = ۲۴ - ۱۲ = ۱۲ \quad \Rightarrow W_4 T_4 = ۱۴۴$$

$$T_3 = ۳۷ - ۱ = ۳۶ \quad \Rightarrow W_3 T_3 = ۳۶$$

$$\Rightarrow \textstyle\sum_i W_i T_i = ۵۰۰$$

اکنون باید جواب‌های همسایه نظیر این جواب پیدا شوند. در این جا جواب‌های همسایه هر جواب را با جابه‌جایی ترتیب هر دو کار مجاور (پشت سر هم) در آن جواب بدست می‌آوریم. بنابراین جواب‌های مجاور x_1 عبارتند از:

$$(1, 2, 4, 3) \quad , \quad \textstyle\sum_i W_i T_i = ۴۸۰$$

$$(2, 4, 1, 3) \quad , \quad \textstyle\sum_i W_i T_i = ۴۳۶ \quad \leftarrow$$

$$(2, 1, 3, 4) \quad , \quad \textstyle\sum_i W_i T_i = ۶۵۲$$

همان‌طور که ملاحظه می‌شود در این تکرار، جواب $(3, 1, 4, 2)$ که تابع هدف کمتری از بین سایرین دارد انتخاب می‌شود و در مرحله بعد حرکت به آن صورت می‌گیرد. در این مثال، ما حرکت‌های صورت گرفته را در فهرست ممنوع قرار می‌دهیم. از آن‌جا که جابه‌جایی بین کارهای j_1، j_4 در x_1 باعث به‌وجود آمدن $x_2 = (3, 1, 4, 2)$ شده است، این حرکت را به شکل $(4, 1)$ نشان می‌دهیم و در تکرار بعدی آن را در فهرست

ممنوع قرار می‌دهیم.

تکرار ۲: لیست ممنوع برابر با $TabuList = \{(۱, ۴)\}$ است.

$$x_۲ = (۲, ۴, ۱, ۳) \quad , \quad \sum_i W_i T_i = ۴۳۶$$

همسایگی‌های $x_۲$ به صورت زیر هستند:

$$(۴, ۲, ۱, ۳) \quad , \quad \sum_i W_i T_i = ۴۶۰ \leftarrow$$

$$(۲, ۱, ۴, ۳) \quad , \quad \sum_i W_i T_i = ۵۰۰ \quad \leftarrow Tabu$$

$$(۲, ۴, ۳, ۱) \quad , \quad \sum_i W_i T_i = ۶۰۸$$

در این مرحله مقدار تابع هدف تمام جواب‌های همسایه از مقدار تابع هدف $x_۲$ بدتر است، ولی باز هم به بهترین جواب همسایه یعنی $(۴, ۲, ۱, ۳)$ می‌رویم، چون ممکن است در دام یک جواب بهینه محلی افتاده باشیم، از طرفی شرط توقف نیز برقرار نشده است. با توجه به جواب انتخاب شده، در مرحله بعد، $(۲, ۴)$ به انتهای فهرست ممنوع اضافه می‌شود.

تکرار ۳: لیست ممنوع برابر با $TabuList = \{(۱, ۴), (۲, ۴)\}$ است.

$$x_۳ = (۴, ۲, ۱, ۳) \quad , \quad \sum_i = W_i T_i = ۴۶۰$$

همسایگی‌های $x_۳$ به صورت زیر می‌باشد:

$$(۲, ۴, ۱, ۳) \quad , \quad \sum_i W_i T_i = ۴۳۶ \leftarrow Tabu$$

$$(۴, ۱, ۲, ۳) \quad , \quad \sum_i W_i T_i = ۴۴۰ \quad \leftarrow$$

$$(۴, ۲, ۳, ۱) \quad , \quad \sum_i W_i T_i = ۴۳۶$$

در این مرحله جوابی که بهترین تابع هدف را داشت، انتخاب نشد. چون رسیدن به آن ملزم به انجام حرکتی بود که در فهرست ممنوع قرار داشت. بنابراین دومین جواب خوب همسایه انتخاب شد.

با انتخاب $(۴, ۱, ۲, ۳)$ باید حرکت $(۱, ۲)$ به فهرست ممنوع اضافه شود. از آنجا که طول فهرست ممنوع برابر با ۲ قرار داده شده است، حتماً باید یک حرکت از آن خارج شود که حرکت $(۱, ۴)$ از فهرست ممنوع

خارج می‌شود.

تکرار ۴: لیست ممنوع برابر با $TabuList = \{(۲, ۴), (۱, ۲)\}$ است.

$$x_۴ = (۴, ۱, ۲, ۳) \quad , \quad \sum_i = W_i T_i = ۴۴۰$$

همسایگی‌های $x_۴$ به صورت زیر می‌باشد:

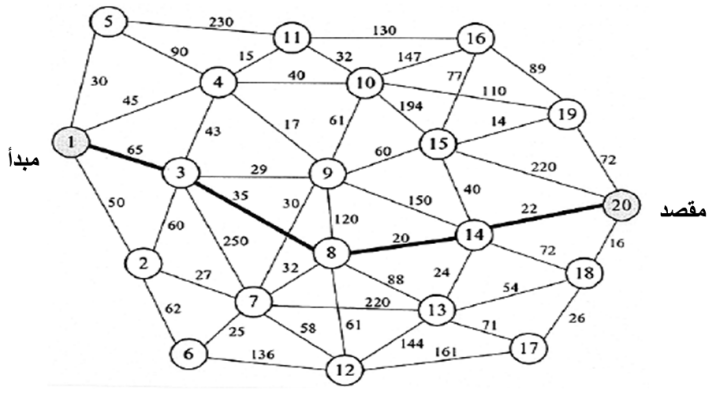
$$(۱, ۴, ۲, ۳) \quad , \quad \sum_i W_i T_i = ۴۰۸ \leftarrow$$

$$(۴, ۲, ۱, ۳) \quad , \quad \sum_i W_i T_i = ۴۶۰$$

$$(۴, ۱, ۳, ۲) \quad , \quad \sum_i W_i T_i = ۵۸۶$$

همان‌طور که ملاحظه می‌شود در این مرحله شرط توقف برقرار می‌شود یعنی با انتخاب $(۱, ۴, ۲, ۳)$ مقدار تابع هدف برابر با ۴۰۸ می‌شود و بنابراین متوقف می‌شویم.

مثال ۱-۲-۷. هدف پیدا کردن کوتاه‌ترین مسیر بین دو گره ۱ و ۲۰ در شبکه شکل ۱-۳ با استفاده از روش جستجوی ممنوع می‌باشد.



شکل ۱-۳: شبکه‌ای با ۲۰ گره

در این مثال برای یافتن کوتاه‌ترین مسیر از الگوریتم مرجع [۲۷] استفاده کردیم. اعتبار تابو، در ابتدا برابر با ۱۰ و از نوع متغیر است.

تکرار صفر: چون به تعداد ۲۰ گره داریم، بنابراین تعداد مؤلفه‌های هر جواب و مؤلفه‌های لیست ممنوع را برابر با ۲۰ در نظر می‌گیریم. ابتدا جواب اولیه‌ای از صفر و یک به طور تصادفی ساخته می‌شود، با توجه به اینکه گره شماره یک گره مبدأ است لذا باید عنصر متناظر با آن برابر یک باشد و لیست ممنوع، صفر است.

فرض کنید جواب اولیه به صورت زیر باشد:

جدول ۱-۳: جواب اولیه مثال ۱-۲-۷

گره	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰	۱۱	۱۲	۱۳	۱۴	۱۵	۱۶	۱۷	۱۸	۱۹	۲۰
وضعیت	۱	۱	۰	۱	۱	۰	۰	۱	۱	۱	۰	۱	۱	۰	۱	۰	۰	۱	۱	۱

سطر اول جدول ۱-۳ شماره‌ی گره‌ها و سطر دوم، وضعیت آن گره را نشان می‌دهد. اگر وضعیت گره‌ای برابر یک باشد به این معنی است که در طول مسیر می‌توان به آن گره وارد شویم و از آن عبور کنیم یا به اصطلاح، گره باز است. اگر وضعیت گره‌ای صفر باشد، آن گره بسته است. اکنون باید مقدار تابع هدف را برای این جواب، محاسبه کنیم که از رابطه‌ی زیر بدست می‌آید:

$$f = (\sum_{j=۱}^{N-۱} C_{yz})^{-۱}$$

N = تعداد گره‌های مسیر C_{yz}= هزینه اتصال گره y به گره z

برای محاسبه‌ی مقدار تابع هدف از بین گره‌های متصل به گره شماره یک که باز هستند، گره‌ای با کمترین هزینه را انتخاب کرده و به آن حرکت می‌کنیم. با توجه به شکل ۱-۳ ، گره شماره ۵ کمترین هزینه را دارد، بنابراین به مسیر اضافه می‌شود. دوباره برای گره شماره ۵، گره‌ای باز با کمترین هزینه را انتخاب می‌کنیم و سپس به مسیر اضافه می‌شود، در ادامه با تکرار، به گره مقصد(گره شماره ۲۰) می‌رسیم. مسیر و تابع هدف برای این جواب به صورت زیر است:

$$P: ۱ \rightarrow ۵ \rightarrow ۴ \rightarrow ۹ \rightarrow ۱۵ \rightarrow ۱۹ \rightarrow ۲۰ \quad f = ۲۸۳$$

البته ممکن است در بعضی از جواب‌ها، به گره مقصد نرسیم، در این صورت مسیری نامعتبر بدست آمده است.

تکرار ۱: همان‌طور که در مثال ۱-۲-۳ نحوه‌ی بدست آوردن همسایه‌های جوابی با عنصرهای صفر و یک بیان شد، بهترین همسایه را برای جواب اولیه می‌یابیم. در این مثال جوابی بهترین همسایه است که: با باز یا بسته شدن یکی از گره‌ها ($bestFlips$) کمترین مقدار تابع هدف را دارد و عنصر متناظر با آن گره در لیست ممنوع صفر باشد، همچنین اختلاف تابع هدف جواب جاری و بهترین همسایه ($bestGain$) بزرگ‌تر یا مساوی صفر باشد. در صورتی که بهترین همسایه پیدا نشد، الگوریتم مکانی باز و دلخواه را انتخاب می‌کند و آن مکان را می‌بندد. بهترین همسایه برای جواب اولیه با بسته شدن گره شماره ۵ بدست می‌آید.

لیست ممنوع نیز بهنگام می‌شود، یعنی به درایه‌ی پنجم لیست ممنوع مقدار ۱۰ اختصاص داده می‌شود.

جدول ۱-۴: جواب در تکرار یک

گره	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰	۱۱	۱۲	۱۳	۱۴	۱۵	۱۶	۱۷	۱۸	۱۹	۲۰
وضعیت	۱	۱	۰	۱	۰	۰	۰	۱	۱	۱	۰	۱	۱	۰	۱	۰	۰	۱	۱	۱

مسیر به صورت زیر تغییر می‌یابد:

$$P: ۱ \to ۴ \to ۹ \to ۱۵ \to ۱۹ \to ۲۰ \quad f = ۲۰۸$$

$$bestGain = ۲۸۳ - ۲۰۸ = ۷۵ \geqslant ۰ \qquad bestFlips = ۵$$

تکرارهای دیگر الگوریتم به همین نحو انجام می‌شود.

در جدول ۱-۵ جواب‌های الگوریتم برای ۵۰ تکرار نشان داده شده است و بهترین مسیر در کمترین زمان (۰/۴۸) به صورت زیر است:

$$P: ۱ \to ۳ \to ۸ \to ۱۴ \to ۲۰$$

جدول ۱-۵: مسیرهای بدست آمده در هر تکرار از روش جستجوی ممنوع

شماره تکرار	مسیر	تابع هدف	زمان(ثانیه)
۱	مسیر نامعتبر	∞	۰/۱۴
۲	$۱ \to ۴ \to ۹ \to ۷ \to ۸ \to ۱۴ \to ۲۰$	۱۶۶	۰/۷۶
۳	$۱ \to ۳ \to ۸ \to ۱۴ \to ۲۰$	۱۴۲	۰/۵۳
۴	نامعتبر	∞	۰/۱۳
۵	$۱ \to ۴ \to ۱۰ \to ۱۹ \to ۲۰$	۲۶۷	۰/۵۰
۶	$۱ \to ۴ \to ۱۰ \to ۱۹ \to ۲۰$	۲۶۷	۰/۶۸
۷	$۱ \to ۴ \to ۱۰ \to ۱۹ \to ۲۰$	۲۶۷	۰/۵۵
۸	مسیر نامعتبر	∞	۰/۱۴
۹	مسیر نامعتبر	∞	۰/۱۴
⋮	⋮	⋮	⋮
۴۷	مسیر نامعتبر	∞	۰/۱۵
۴۸	مسیر نامعتبر	∞	۰/۱۴
۴۹	$۱ \to ۳ \to ۸ \to ۱۴ \to ۲۰$	۱۴۲	۰/۴۸
۵۰	$۱ \to ۳ \to ۸ \to ۱۴ \to ۲۰$	۱۴۲	۰/۶۹

فصل ۲

الگوریتم ژنتیک و الگوریتم ازدحام ذرات

۲-۱ الگوریتم ژنتیک (GA)

یک الگوریتم جستجوی تصادفی است که بر اساس قوانین تکامل طبیعی عمل می‌کند. این الگوریتم اولین بار توسط جان هولند[۱] [۲۰] در سال ۱۹۷۵ ابداع شد و توسط گلدبرگ[۲] [۱۶] توسعه داده شد، که در سال‌های اخیر به طور گسترده‌ای برای مسائل بهینه‌سازی و جستجو به کار گرفته شده است.

در طبیعت، فرایند تکامل هنگامی ایجاد می‌شود که شرایط زیر برقرار باشد [۱]:

۱- موجود توانایی تکثیر خود را داشته باشد.

۲- یک جمعیت از چنین موجوداتی که قابلیت تولید خود را دارند، وجود داشته باشند.

۳- تنوع در بین این موجودات وجود داشته باشد.

۴- انواع این موجودات توسط بعضی از قابلیت‌ها، در زندگی محیط اطرافشان از هم مجزا شوند.

اصول اولیه این الگوریتم از علم ژنتیک اقتباس شده است، علم ژنتیک، درباره چگونگی توارث و انتقال صفحات بیولوژیکی از نسلی به نسل بعد صحبت می‌کند. عامل اصلی انتقال صفحات بیولوژیکی در موجودات زنده کروموزم[۳] و ژن‌ها[۴] هستند. نحوه‌ی عملکرد آن‌ها به گونه‌ای است که در نهایت ژن‌ها و کروموزم‌های برتر و قوی‌تر باقی می‌مانند و ژن‌های ضعیف‌تر از بین می‌روند. به عبارت دیگر نتیجه عملیات متقابل ژ نها و کروموزم‌ها باقی ماندن موجودات اصلح و برتر است.

---

[۱] John Holland [۲] Goldberg [۳] Chromosome [۴] Genes

اساس این الگوریتم قانون تکامل داروین[۵](۱۸۵۹) است که می‌گوید موجودات ضعیف‌تر از بین می‌روند و موجودات قوی‌تر باقی می‌مانند. قانون انتخاب[۶] طبیعی بدین صورت است که تنها گونه‌هایی از یک جمعیت[۷] ادامه نسل[۸] می‌دهند که بهترین خصوصیات را داشته باشند و آن‌هایی که این خصوصیات را نداشته باشند به تدریج و در طی زمان از بین می‌روند.

الگوریتم ژنتیک به دلیل تقلید از طبیعت دارای چند اختلاف اساسی با روش‌های جستجوی مرسوم است:

۱) الگوریتم ژنتیک با رشته‌های بیتی کار می‌کند که هر کدام از این رشته‌ها کل مجموعه متغیرها را نشان می‌دهد، حال آنکه بیشتر روش‌ها به طور مستقل با متغیرهای ویژه برخورد می‌کنند.

۲) این الگوریتم برای راهنمایی جهت جستجو، انتخاب تصادفی انجام می‌دهد.

۳) الگوریتم ژنتیک فقط نیاز به اطلاعاتی در مورد کیفیت راه‌حل‌های ایجاد شده به وسیله هر مجموعه دارد، در صورتی که بعضی از روش‌های بهینه‌سازی نیاز به اطلاعات یا حتی نیاز به شناخت کامل از ساختمان مسأله و متغیرها دارند. چون این الگوریتم نیاز به چنین اطلاعات مشخصی از مسأله ندارد بنابراین قابل انعطاف‌تر از بیشتر روش‌های جستجو است [۳].

جدول ۲-۱: رابطه بین الگوریتم ژنتیک و طبیعت [۲]

الگوریتم ژنتیک	طبیعت
مسأله بهینه‌سازی	محیط پیرامون
جواب‌های موجه	افرادی که در محیط زندگی می‌کنند
کیفیت جواب‌ها (تابع برازندگی)	درجه انطباق افراد با محیط پیرامون
مجموعه‌ای از جواب‌های موجه	جمعیتی از موجودات زنده
اپراتورهای ژنتیکی	انتخاب، ترکیب مجدد و تغییر در فرایند تکامل
به کارگیری مکرر اپراتورهای ژنتیکی	تکامل جمعیت در جهت انطباق با محیط

۲-۱-۱ واژگان الگوریتم ژنتیک

تعریف ۲-۱-۱. اساس الگوریتم ژنتیک تبدیل هر مجموعه جواب به یک کدینگ[۹] است. این کدینگ را اصطلاحاً **کروموزوم** نامند. به کروموزوم، فرد، رشته نیز گفته می‌شود.

تعریف ۲-۱-۲. عناصر تشکیل دهنده یک کروموزوم که معمولاً اعداد هستند را **ژن** می‌گویند.

[۵]Darvin [۶]Selection [۷]Population [۸]Generation [۹]Encoding

تعریف ۲-۱-۳. محل قرار گرفتن ژن در کروموزوم را یک **مکان** می‌گویند.

تعریف ۲-۱-۴. مجموعه‌ای از کروموزوم‌ها را یک **جمعیت** می‌گویند.

تعریف ۲-۱-۵. هر تکرار از الگوریتم را یک **نسل** می‌نامند [۲].

۲-۱-۲ ساختار کلی الگوریتم ژنتیک

این الگوریتم با در نظر گرفتن مجموعه‌ای از نقاط فضای جواب که اغلب به طور تصادفی ایجاد می‌گردد، در هر تکرار محاسباتی به نحو مؤثری نواحی مختلف فضای جواب را جستجو می‌کند.

در هر تکرار هر یک از رشته‌های موجود در جمعیت، رمزگشایی شده و مقدار تابع هدف برای آن بدست می‌آید. بر اساس مقادیر بدست آمده تابع هدف در جمعیت رشته‌ها، به هر رشته یک عدد برازندگی نسبت داده می‌شود. این عدد برازندگی احتمال انتخاب را برای هر رشته تعیین خواهد کرد. بر اساس این احتمال انتخاب، مجموعه‌ای از رشته‌ها به عنوان والدین انتخاب شده و با اعمال عمل ترکیب[۱۰] بر روی آن‌ها چند جواب ممکن یا فرزند برای جمعیت جدید تولید می‌کند.

در حین اجرای الگوریتم جهش‌ها[۱۱] و مهاجرت‌هایی به طور متناوب برای بهبود انجام می‌شود، علاوه بر این تعدادی از جواب‌های بد از جمعیت جاری خارج می‌شوند تا تعداد رشته‌ها در تکرارهای محاسباتی مختلف ثابت باشد. این فرایند ادامه دارد تا اینکه شرط پایان برقرار شود.

روش‌های تصادفی که روی انتخاب و حذف رشته‌ها عمل می‌کنند به گونه‌ای هستند که رشته‌هایی که عدد برازندگی بیشتری دارند، احتمال بیشتری برای ترکیب و تولید رشته‌های جدید داشته و در مرحله جایگزینی نسبت به دیگر رشته‌ها مقاوم‌تر هستند [۳].

۲-۱-۳ کاربردهای الگوریتم ژنتیک

الگوریتم ژنتیك در فرایند حل مسأله احتیاجی به اطلاعات خاص درباره مسأله ندارد و فقط مقادیر عددي تابع هدف را لازم دارد، بنابراین زمینه استفاده و اعمال آن محدودیتي ندارد. جدول ۲-۲ تعدادی از کابردهای این روش را نشان می‌دهد.

عملگرهای الگوریتم ژنتیک به صورت زیر است:

---

[۱۰] Crossover [۱۱] Mutation

جدول ۲-۲: کاربردهای الگوریتم ژنتیک

فروشنده دوره‌گرد	مسأله مکان‌یابی پویا
رنگ آمیزی گراف	بهینه‌سازی توابع
برنامه‌ریزی درسی کلاس‌ها	برنامه‌ریزی خطوط تولید
حل مسأله تخصیص درجه ۲	حل مسأله p-میانه
آموزش شبکه‌های عصبی	یادگیری قواعد فازی
برنامه‌ریزی شبکه ارتباطات	طراحی مسیر و حل معادلات سیستماتیک معکوس برای ربات‌ها
مهندسی برق	مهندسی نرم‌افزار
مهندسی مواد	واگذاری

۲-۱-۴ کدگذاری

ابتدا پیش از هر چیز باید روشی برای تبدیل جواب‌های مسأله به یک کروموزوم تعریف کرد، این مرحله کدگذاری نام دارد و شاید مشکل‌ترین مرحله در روند کلی الگوریتم ژنتیک باشد. در واقع این الگوریتم به جای اینکه بر روی پارامترهای مسأله کار کند، با شکل کد شده آن‌ها سرو کار دارد. یکی از روش‌های معمول کدگذاری استفاده از رشته‌های صفر و یک است.

۲-۱-۵ ارزیابی جواب‌های هر نسل

در این مرحله با معرفی یک معیار یا اندازه به بهتر بودن یا نبودن هر جواب ممکن از جمعیت پی می‌بریم.

تعریف ۲-۱-۶. تابعی که بر اساس آن میزان سازگاری کروموزوم با محیط ارزیابی می‌شود، **تابع برازندگی**[۱۲] نام دارد و از اعمال تبدیل مناسب بر روی تابع هدف یعنی تابعی که قرار است بهینه شود، بدست می‌آید.

هر چه کیفیت رشته بالاتر باشد مقدار برازندگی جواب بیشتر است و احتمال مشارکت برای نسل بعدی نیز افزایش خواهد یافت.

۲-۱-۶ روش‌های انتخاب

۱-۳-۶-۱ انتخاب تصادفی: در این روش، والدین بر اساس یک روش کاملاً تصادفی از جمعیت انتخاب می‌گردند. به عنوان مثال می‌توان دو عدد تصادفی را در بازه یک و تعداد کروموزوم‌ها تولید و کروموزوم‌های مربوط به آن دو عدد را به عنوان والد انتخاب کرد.

[۱۲] Fitness

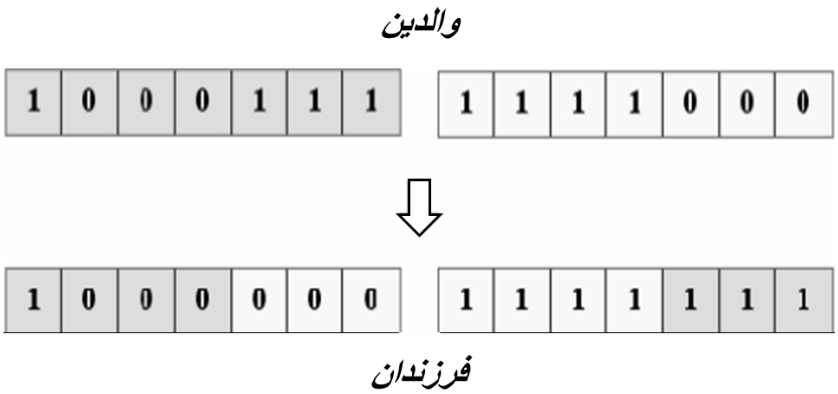
۱-۳-۶-۲ انتخاب بر اساس چرخ رولت[۱۳]: گلدبرگ روش چرخ رولت را برای تکثیر ارائه داد. در این روش یک دیسک که سطح آن متناسب با برازندگی هر رشته تقسیم‌بندی شده است، در مقابل یک نشانه می‌چرخد تا بالاخره متوقف شود و یکی از سطوح در مقابل نشانه قرار گیرد، به این ترتیب یک رشته انتخاب می‌شود. بنابراین با تکرار به تعداد دفعات مناسب می‌توان رشته‌های لازم برای اعمال عملگرهای ژنتیک را انتخاب نمود.

۱-۳-۶-۳ رتبه‌بندی[۱۴]: کروموزوم‌ها بر حسب برازندگی مرتب می‌شوند و برنامه به ترتیب انتخاب را انجام می‌دهد.

۲-۱-۷ ترکیب

مهم‌ترین عملگر در این الگوریتم، عملگر ترکیب است. فرایندی است که در آن نسل قدیمی کروموزوم‌ها با یکدیگر ترکیب می‌شوند تا نسل تازه‌ای را بوجود آورند. معمولاً این عملگر را به درصدی از جمعیت اعمال می‌کنند و درصد ترکیب را با p_c نشان می‌دهند. اگر درصد ترکیب را $p_c = ۰/۵۲$ فرض کنیم و ۴ کروموزوم داشته باشیم، در این صورت عملگر ترکیب بر روی دو کروموزوم اعمال می‌شود ($۴ \times ۰/۵۲ = ۲/۰۸ \simeq ۲$).

۱-۳-۷-۱ ترکیب یک نقطه‌ای: ابتدا عددی تصادفی در فاصله ۱تا طول کروموزوم تولید می‌شود، سپس دو کروموزوم از این نقطه تا انتهای دو عبارت شکسته شده و با هم ترکیب می‌شوند. شکل ۲-۱ ترکیب یک نقطه‌ای را نشان می‌دهد.

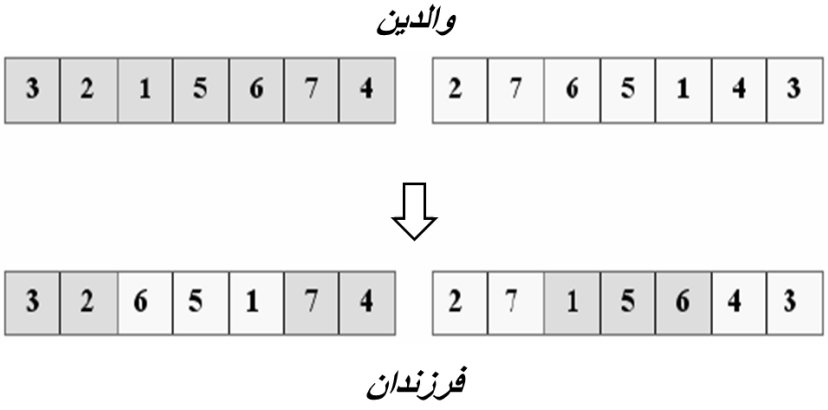


شکل ۲-۱: ترکیب یک نقطه‌ای

۱-۳-۷-۲ ترکیب k نقطه‌ای: این روش مانند قبلی است با این تفاوت که نقاط شکست بیشتر از یک نقطه می‌باشد. در شکل ۲-۲ ترکیب دو نقطه‌ای نشان داده شده است.

۱-۳-۷-۳ ترکیب یکنواخت: یک کروموزوم تصادفی هم طول با کروموزوم‌های موجود تولید می‌شود. بر اساس این عملگر، یک ژن از هر دو والد شانس برابر برای حضور در کروموزوم یک فرزند را دارند. کروموزوم تولید شده تعیین می‌کند که کدام ژن از والد اول و کدام ژن از والد دوم به فرزند منتقل می‌شود. در هر ژن از

[۱۳] Roulette Wheel [۱۴] Ranking



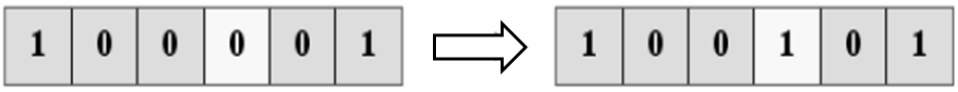
شکل ۲-۲: ترکیب دو نقطه‌ای

این کروموزوم در صورتی که عدد تصادفی ۱ باشد، ژن فرزند از والد اول و در صورتی که صفر باشد، از والد دوم انتخاب می‌گردد.

۲-۱-۸ جهش

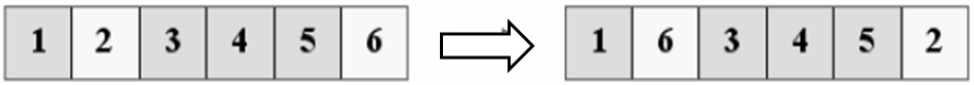
عملگر دیگری است که بعد از اینکه یک عضو در جمعیت بوجود آمد، یک یا چند ژن آن تغییر می‌یابد و وابسته به نوع کدگذاری، روش‌های متفاوت جهش استفاده می‌شود. از این عملگر باید کم استفاده شود و وظیفه آن حفظ پراکندگی در جمعیت است. مثلاً ممکن است در جمعیت تصادفی اولیه، همه رشته‌ها در یک ژن بخصوص مقدار «۱» داشته باشند ولی جواب بهینه در همان ژن مقدار «۰» داشته باشد. به این ترتیب هرگز به جواب بهینه نمی‌رسیم. بنابراین افزایش و تنوع ژنی لازم و ضروری است، این عملگر روی درصدی از جمعیت و گاهی به صورت متناوب بر روی درصدی از جمعیت اعمال می‌شود، این درصد را با p_m نمایش می‌دهیم. اگر درصد جهش $p_m = ۰/۰۱$ فرض شود، در این صورت روی جمعیتی با ۴ کروموزوم هیچ جهشی صورت نمی‌گیرد ($۰/۰۱ \times ۴ = ۰/۰۴ \simeq ۰$) [۱].

۱-۳-۸-۱ جهش ژنی: در ژن‌های صفر و یک این تغییر به معنای تغییر یک ژن از ۰ به ۱ یا برعکس است.



شکل ۲-۳: جهش ژنی

۱-۳-۸-۲ جابه جایی ژن‌ها: دو ژن از یک کروموزوم را انتخاب و جای آن دو را با هم عوض می کند. برای وضوح این نوع جهش، از کروموزوم‌هایی با کدگذاری صحیح استفاده شده است [۲].



شکل ۲-۴: جهش به روش جابه‌جایی

الگوریتم ۲-۱ شبه کد الگوریتم ژنتیک

ورودی: تعداد جمعیت اولیه (n)، درصد ترکیب (p_c)، درصد جهش (p_m)، شرط خاتمه، روش‌های انتخاب.

خروجی: جواب بهینه.

۱: جمعیت n کروموزومی به طور تصادفی ایجاد کنید.

۲: برازندگی هر کروموزوم در جمعیت را ارزیابی کنید.

۳: مراحل زیر را تا زمانیکه شرط خاتمه برقرار شود انجام دهید:

۴: جمعیت جدیدی را تشکیل دهید:

۵: والدین را از میان جمعیت انتخاب کنید و با توجه به احتمال ترکیب، برای تشکیل فرزندان جدید آن‌ها را ترکیب کنید.

۶: با توجه به احتمال جهش فرزندان را مورد جهش قرار دهید.

۷: فرزندان جدید را در جمعیت بگنجانید.

۸: برازندگی هر کروموزوم را ارزیابی کنید.

۲-۱-۹ رمزگشایی (دی‌کدینگ)

رمزگشایی[۱۵] عکس عمل کدگذاری است. در این مرحله بعد از اینکه الگوریتم بهترین جواب را برای مسأله ارائه کرد لازم است عکس عمل رمزگذاری روی جواب‌ها اعمال شود.

اکنون کروموزم‌های متولد شده جزء نسل جدید به حساب می‌آیند و به جمعیت اولیه افزوده می‌شوند و جواب‌هایی با برازندگی پایین حذف می‌شوند و الگوریتم دوباره به کار خود ادامه می‌دهد [۳].

۲-۱-۱۰ حل دو مسأله با استفاده از الگوریتم ژنتیک

در این مرحله برای روشن‌تر شدن مراحل الگوریتم ژنتیک دو مثال را به این روش حل می‌کنیم.

مثال ۲-۱-۷. مسأله کوله‌پشتی، مسأله‌ای است که یک کوهنورد در هنگام انتخاب وسایل لازم برای کوهنوردی با آن مواجه است [۴]. کوله‌پشتی این کوهنورد فضای محدودی دارد و کوهنورد مجبور است تعداد شئ محدودی را از بین n شئ که در اختیار دارد، انتخاب کند. به دلیل محدودیت فضا، امکان انتخاب همه اشیاء وجود ندارد. کوهنورد می‌خواهد اشیائی را انتخاب کند که بیشترین مطلوبیت را داشته باشد. این مسأله را

[۱۵] Decoding

می‌توان به صورت زیر فرمول‌بندی کرد:

$$\max Z = \sum_{i=۱}^{n} x_i v_i \qquad (۱-۲)$$

$$s.t.$$

$$\sum_{i=۱}^{n} x_i w_i \leqslant W$$

$$x_i \in \{۰, ۱\} \quad i = ۱, ..., n$$

که در آن x_i یک متغیر صفر یا یک است که نشان‌دهنده حضور یا عدم حضور شئ iام در کوله است، v_i نشان‌دهنده ارزش (مطلوبیت) حضور شئ iام و w_i نشان‌دهنده حجم یا وزن شئ iام می‌باشد. محدودیت حداکثر حجم یا وزن کوله با W نمایش داده شده است. این مدل به دنبال حداکثر کردن ارزش کوله با رعایت محدودیت حجم یا وزن آن می‌باشد.

در این مثال فرض کنید ۱۰ شئ با مشخصات زیر در اختیار باشد و حداکثر وزن قابل قبول کوله برابر با ۲۰ کیلوگرم باشد.

جدول ۲-۳: مسأله کوله‌پشتی با ۱۰ شی

شماره شئ	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

برای نشان دادن راه‌حل‌های این مسأله از کدگذاری صفر و یک استفاده می‌کنیم. برای مثال، از یک آرایه یک بعدی دارای ۱۰ ارزش به صورت $[x_۱, x_۲, ..., x_{۱۰}]$ استفاده می‌شود. در این آرایه مقادیر صفر یا یک قرار می‌گیرند که حضور یا عدم حضور شئ مرتبط در کوله را نشان می‌دهند. تعداد راه‌حل‌ها برابر $۲^{۱۰}$ خواهد بود که تعدادی از آن‌ها امکان‌پذیر نمی‌باشند.

با توجه به اینکه تعدادی از جواب‌های تولیدی امکان‌پذیر نمی‌باشند از روشی برای تبدیل جواب امکان‌ناپذیر به جواب امکان‌پذیر استفاده می‌شود. در این روش اشیاء داخل کوله بر اساس نسبت ارزش آن‌ها به وزن آن‌ها به صورت صعودی مرتب می‌گردند ($\frac{v_i}{w_i}$). سپس شروع به حذف اشیاء از جواب می‌کنیم تا یک جواب امکان‌پذیر بدست آید.

تولید جمعیت اولیه. در این مرحله، جمعیت اولیه با ۴ کروموزوم به صورت تصادفی تولید می‌گردد.

جدول ۲-۴ جمعیت اولیه تولید شده را نشان می‌دهد.

جدول ۲-۴: جمعیت اولیه تولید شده

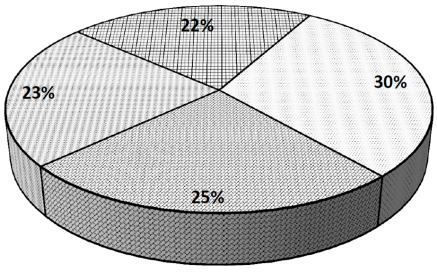
شماره کروموزوم	آرایه	وزن	امکان‌پذیری	$f(x)$	احتمال
۱	[۱۰۰۱۱۰۰۱۰۱]	۱۷/۵	بله	۲۵۱	۰/۲۵
۲	[۱۱۰۱۰۰۱۰۰۰]	۲۰	بله	۳۰۵	۰/۳۰
۳	[۰۰۱۰۱۱۱۰۱۱]	۱۵/۳	بله	۲۲۴	۰/۲۲
۴	[۰۱۰۰۰۱۱۰۱۰]	۱۴/۸	بله	۲۳۰	۰/۲۳
جمع				۱۰۱۰	۱

کروموزوم ۱ را 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 در جمعیت اولیه وزن اشیاء کمتر از ۲۰ می‌باشد، جواب‌ها امکان‌پذیر هستند. احتمال انتخاب هر یک از کروموزوم‌ها به صورت حاصل تقسیم مقدار برازندگی آن کروموزوم بر مقدار برازندگی کل کروموزوم‌ها محاسبه شده است و به صورت درصد در شکل ۲-۵ نمایش داده شده است.



شکل ۲-۵: احتمال انتخاب کروموزم‌ها با استفاده از چرخ رولت

تکرار ۱: تولید جمعیت جدید. در این مرحله، بر اساس چرخ رولت کروموزوم‌های ۱ و ۲ و همچنین کروموزوم‌های ۲ و ۴ برای تولیدمثل انتخاب شده‌اند. از ترکیب تک نقطه‌ای استفاده می‌شود و نقطه تقاطع بر اساس یک قاعده تصادفی انتخاب شده است. دو کروموزوم اول در نقطه ۶ و دو کروموزوم دوم در نقطه ۴ ترکیب شده‌اند. با این روش ترکیب، ۴ کروموزوم فرزند تولید می‌شود.

با توجه به اینکه وزن کروموزوم فرزند شماره ۶ و ۷ بیشتر از ۲۰ است لذا این دو کروموزوم قابل قبول نمی‌باشند، این کروموزوم‌ها به شماره‌های ۹ و ۱۰ اصلاح می‌شوند. نحوه‌ی اصلاح کروموزوم شماره ۶ به

جدول ۲-۵: فرزندان تولید شده در تکرار ۱

شماره والد	آرایه والد	شماره فرزند	آرایه فرزند	وزن	$f(x)$	توضیح
۱	[۱۰۰۱۱۰:۰۱۰۱]	۵	[۱۰۰۱۱۰۱۰۰۰]	۱۶	۲۴۴	
۲	[۱۱۰۱۰۰:۱۰۰۰]	۶	[۱۱۰۱۰۰۰۱۰۱]	۲۱/۵	—	اصلاح به کروموزوم ۹
۳	[۱۱۰۱:۰۰۱۰۰۰]	۷	[۱۱۰۱۰۱۱۰۱۰]	۲۵/۸	—	اصلاح به کروموزوم ۱۰
۴	[۰۱۰۰:۰۱۱۰۱۰]	۸	[۰۱۰۰۰۰۱۰۰۰]	۹	۱۴۲	
		۹	[۱۱۰۱۰۰۰۱۰۰]	۱۹/۵	۲۹۱	
		۱۰	[۰۱۰۱۰۰۱۰۱۰]	۱۹	۳۰۴	

صورت زیر است:

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

حذف اشیاء از کروموزوم ۶ به ترتیب ۱۰، ۱، ۸، ۲، ۴ می‌باشد، بر این اساس ابتدا شئ شماره ۱۰ حذف می‌شود و وزن کوله به ۱۹/۵ کیلوگرم کاهش می‌یابد. اکنون جواب شئ دیگری از کوله حذف نمی‌گردد و جواب اصلاح شده است.

جدول ۲-۶ فرزندان تولید شده را بعد از عمل جهش نشان می‌دهد. فرض کنید 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 شود. فرض می‌شود که بهترین و بدترین کروموزوم به صورت ثابت به جمعیت بعدی منتقل شوند و ۲ کروموزوم دیگر به صورت تصادفی بر اساس مقدار برازندگی آن‌ها انتخاب گردند. جمعیت جدید در جدول ۲-۷ نمایش داده شده است. مراحل فوق تکرار می‌شود تا شرط توقف برآورده گردد.

مثال ۲-۱-۸. در جدول ۲-۸ نتایج بدست آمده توسط الگوریتم ژنتیک در جعبه ابزار نرم‌افزار متلب برای شبکه‌ای با ۲۰ گره (شکل ۱-۳) برای ۵۰ تکرار نشان داده شده است. کروموزوم‌ها از نوع صفر و یک و جمعیت اولیه برابر با ۲۰ کروموزوم در نظر گرفته شده است.

جدول ۲-۶: فرزندان تولید شده در تکرار ۱ بعد از عمل جهش

شماره فرزند	آرایه فرزند	شماره ژن جهش یافته	شماره فرزند	آرایه فرزند	وزن	$f(x)$	توضیح
۵	[۱۰۰۱۱۰۱۰۰۰]	۲،۵	۱۱	[۱۱۰۱۰۰۱۰۰۰]	۲۰	۳۰۵	
۸	[۰۱۰۰۰۰۱۰۰۰]	-	۸	[۰۱۰۰۰۰۱۰۰۰]	۹	۱۴۲	
۹	[۱۱۰۱۰۰۰۱۰۰]	-	۹	[۱۱۰۱۰۰۰۱۰۰]	۱۹/۵	۲۹۱	
۱۰	[۰۱۰۱۰۰۱۰۱۰]	۳	۱۲	[۰۱۱۱۰۰۱۰۱۰]	۲۱/۵	—	اصلاح به کروموزوم ۱۳
			۱۳	[۰۱۰۱۰۰۱۰۱۰]	۱۹	۳۰۴	

جدول ۲-۷: جمعیت تولید شده بعد از تکرار ۱

شماره کروموزوم	آرایه	وزن	امکان‌پذیری	$f(x)$	احتمال
۲	[۱۱۰۱۰۰۱۰۰۰]	۲۰	بله	۳۰۵	۰/۲۹۳
۸	[۰۱۰۰۰۰۱۰۰۰]	۹	بله	۱۴۲	۰/۱۳۷
۱۳	[۰۱۱۱۰۰۱۰۱۰]	۱۹	بله	۳۰۴	۰/۲۹۱
۹	[۱۱۰۱۰۰۰۱۰۰]	۱۹/۵	بله	۲۹۱	۰/۲۷۹
جمع				۱۰۴۲	۱

۲-۲ الگوریتم ازدحام ذرات (PSO)

الگوریتم ازدحام ذرات[۱۶] در سال ۱۹۹۵ توسط ابرهارت[۱۷] و کندی[۱۸] [۲۱] ابداع شد و ایده‌ی اصلی آن از حرکت پرندگان و ماهی‌ها گرفته شده است. به دلیل همگرایی سریع، محاسبات ساده و فهم آسان، PSO کاربرد وسیعی در مسائل بهینه سازی پیوسته مانند کنترل ولتاژ و توان دارد و اخیراً برای مسائل بهینه‌سازی گسسته مانند فروشنده دوره گرد، رنگ‌آمیزی گراف، نیز توسعه داده شده است. عملکرد این الگوریتم به پارامترهای آن بستگی دارد و ممکن است جواب در بهینه محلی به دام افتد، بنابراین محققان روش‌های زیادی را جهت حل این مشکل پیشنهاد داده‌اند.

تعریف ۲-۲-۱. در این روش هر جواب مسأله به عنوان یک پرنده در نظر گرفته شده است و **ذره** نامیده می‌شود.

الگوریتم ازدحام ذرات با یک جمعیت اولیه از ذرات شروع می‌شود، هر ذره یک بردار موقعیت[۱۹] دارد که برازندگی بر اساس آن بدست می‌آید و یک بردار سرعت که جهت حرکت ذره را نشان می‌دهد. در هر تکرار

[۱۶] Particle swarm optimization [۱۷] Eberhart [۱۸] Kennedy [۱۹] Position

جدول ۲-۸: مسیرهای بدست آمده در هر تکرار از الگوریتم ژنتیک

زمان(ثانیه)	تابع هدف	مسیر	شماره تکرار
۰/۳۰	۱۴۲	۱ → ۳ → ۸ → ۱۴ → ۲۰	۱
۰/۳۰	۱۴۲	۱ → ۳ → ۸ → ۱۴ → ۲۰	۲
۰/۳۳	۱۴۲	۱ → ۳ → ۸ → ۱۴ → ۲۰	۳
۰/۳۶	۱۴۲	۱ → ۳ → ۸ → ۱۴ → ۲۰	۴
۰/۴۰	۱۴۲	۱ → ۳ → ۸ → ۱۴ → ۲۰	۵
۰/۳۱	۱۴۲	۱ → ۳ → ۸ → ۱۴ → ۲۰	۶
۰/۳۲	۱۴۲	۱ → ۳ → ۸ → ۱۴ → ۲۰	۷
۰/۳۲	۱۴۲	۱ → ۳ → ۸ → ۱۴ → ۲۰	۸
۰/۴۲	۱۴۲	۱ → ۳ → ۸ → ۱۴ → ۲۰	۹
⋮	⋮	⋮	⋮
۰/۳۲	۱۴۲	۱ → ۳ → ۸ → ۱۴ → ۲۰	۴۷
۰/۳۲	۱۴۲	۱ → ۳ → ۸ → ۱۴ → ۲۰	۴۸
۰/۳۶	۱۴۲	۱ → ۳ → ۸ → ۱۴ → ۲۰	۴۹
۰/۳۰	۱۴۲	۱ → ۳ → ۸ → ۱۴ → ۲۰	۵۰

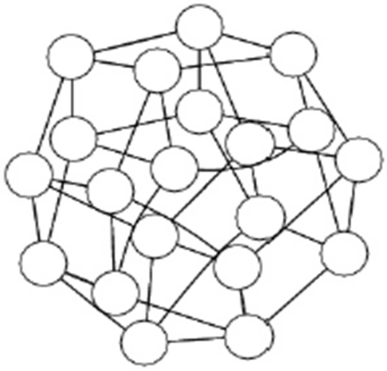
مقدار برازندگی هر ذره محاسبه می‌شود و سرعت ذره بر اساس بهترین موقعیت ذره تاکنون $(pbest)$ و بهترین موقعیت در همسایگی ذره $(nbest)$ بهنگام می‌شود، زمانی‌که تمام جمعیت به عنوان همسایگی ذره در نظر گرفته شود بهترین همسایگی $gbest$ نامیده می شود. در واقع هر ذره در فضای جواب با تغییر جهت و سرعت بر اساس تجربه خود و همسایگانش حرکت می‌کند. بنابراین موقعیت ذرات دیگر روی چگونگی جستجوی یک ذره اثر می‌گذارد.

۲-۲-۱ انواع توپولوژی همسایگی

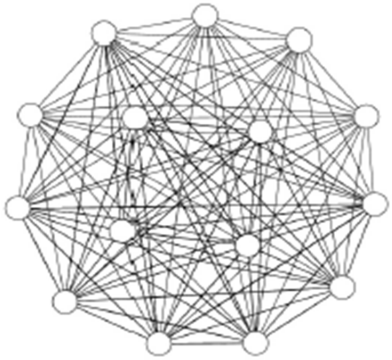
با توجه به تأثیر سرعت و موقعیت ذره‌های همسایه بر سرعت وموقعیت هر ذره، بنابراین تعریف همسایگی با توجه به مسأله مورد نظر مهم است. انواع توپولوژی همسایگی به صورت زیر است:

- **تصادفی**: همسایه‌ها به طور تصادفی انتخاب می‌شوند (شکل ۲-۶).
- **ستاره**: زمانی که تمام جمعیت به عنوان همسایگی ذره در نظر گرفته می‌شود (شکل ۲-۷).

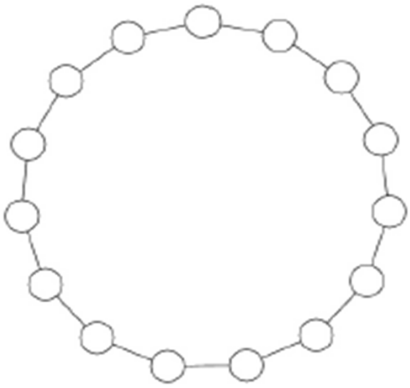
- **حلقه:** در این توپولوژی هر ذره به دو همسایه‌اش متصل شده است (شکل ۲-۸).



شکل ۲-۶: همسایگی تصادفی



شکل ۲-۷: همسایگی ستاره



شکل ۲-۸: همسایگی حلقه

۲-۲-۲ معرفی پارامترها

اکنون پارامترهای لازم را در جدول ۲-۹ خلاصه می‌کنیم.

فرایند به‌کارگیری PSO برای حل مسائل مختلف به شکل زیر خلاصه می‌شود:

۱- **مقداردهی اولیه ذرات:** جمعیت اولیه از ذرات در فضای جستجو به طور تصادفی تولید می‌شود، یعنی ابتدا سرعت و موقعیت هر ذره در جمعیت اولیه مقداردهی تصادفی می‌شوند.

۲- **ارزیابی مقدار برازندگی و محاسبه $pbest$ و $nbest$ برای هر ذره:** کیفیت هر ذره (جواب) توسط مقدار برازندگی آن محاسبه می‌شود، در تمام جمعیت اولیه ذره‌ای که بهترین موقعیت را دارد به عنوان $nbest$ انتخاب می‌شود. بهترین موقعیت هر ذره تاکنون، $pbest$ ذره می‌باشد.

جدول ۲-۹: معرفی پارامترها

نماد	توضیح
D	بعد فضای جستجو
N_s	تعداد ذرات در گروه
$x_i = [x_{i1}, ..., x_{iD}]$	موقعیت iامین ذره
$v_i = [v_{i1}, ..., v_{iD}]$	سرعت iامین ذره
$b_i = [b_{i1}, ..., b_{iD}]$	بهترین موقعیت ذره i $(pbesti)$
$b_i^n = [b_{i1}^n, ..., b_{iD}^n]$	بهترین همسایگی ذره i $(nbesti)$

۳- **بهنگام‌سازی بردار سرعت و بردار موقعیت:** در هر تکرار، مقدار سرعت جدید برای هر ذره بر اساس سرعت جاری و بهترین موقعیت قبلی و بهترین موقعیت کلی محاسبه می‌شود. سپس سرعت جدید برای محاسبه موقعیت بعدی مورد استفاده قرار می‌گیرد.

بهنگام‌سازی سرعت و موقعیت به ترتیب با فرمول‌های زیر محاسبه می‌شود:

$$v_{id}^{t+1} = v_{id}^t + c_1 r_1 (b_{id}^t - x_{id}^t) + c_2 r_2 (b_{id}^{nt} - x_{id}^t) \qquad (۲-۲)$$

$$x_{id}^{t+1} = x_{id}^t + v_{id}^{t+1} \quad i = 1, ..., N_s \quad d = 1, ..., D \qquad (۲-۳)$$

به منظور کنترل سرعت و جلوگیری از خروج ذره از فضای جستجو از v_d^{max} و v_d^{min}نیز به صورت زیر استفاده می‌شود:

اگر $v_i > v_d^{max}$ باشد آنگاه $v_i = v_d^{max}$ و اگر $v_i < v_d^{min}$ آنگاه $v_i = v_d^{min}$ در نظر گرفته می‌شود.

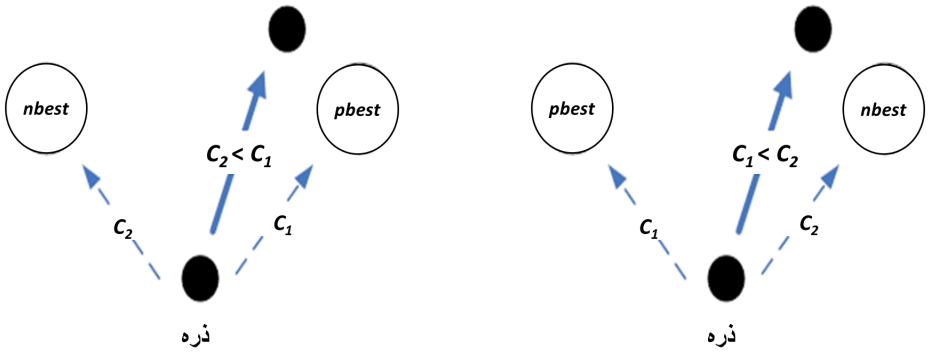
در معادله ۲-۲ c_1 و c_2 ثابت‌های مثبت هستند، c_1 مربوط به تجارب شخصی هر ذره و c_2 مربوط به تجارب جمع می‌باشد. این دو فاکتور تأثیر $pbest$ و $nbest$ را روی فرایند جستجو کنترل می‌کنند.

اگر $c_2 < c_1$ ذرات به سمت $nbest$ (بهترین موقعیت در همسایگی ذره) متمایل می‌شوند (شکل ۲-۹(آ)).

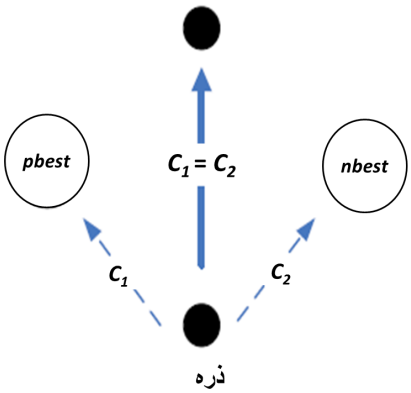
اگر $c_2 > c_1$ ذرات به سمت $pbest$ (بهترین موقعیت ذره) متمایل می‌شوند (شکل ۲-۹(ب)).

ابرهارت و کندی (۱۹۹۵) [۲۱] پیشنهاد کردند برای داشتن حالت میانگین باید $c_1 = c_2$ باشد و در این صورت تمایل رسیدن به $pbesti$ و $nbesti$ به یک میزان خواهد بود (شکل ۲-۱۰).

اگر c_1 و c_2 مقادیر بزرگی باشند، آنگاه سرعت رسیدن به بهترین جواب ممکن زیاد، ولی دقت محاسبه



شکل ۲-۹: تأثیر ضرایب نامساوی c_1 و c_2 در حرکت ذره



شکل ۲-۱۰: تأثیر $c_1 = c_2$ در حرکت ذره

نقطه بهینه کم خواهد بود و احتمال بوجود آمدن نوسان در محاسبه نقاط بهینه زیاد می‌گردد. از طرفی اگر مقادیر c_1 و c_2 کوچک باشد دقت نقطه بدست آمده بالا است ولی سرعت رسیدن به نقطه بهینه کم می‌شود [۳۱].

r_1 و r_2 اعداد تصادفی در بازه $[0, 1]$ هستند که نوعی گوناگونی در جواب‌ها بوجود می‌آورند و جستجوی کاملی روی فضا انجام می‌گیرد.

۴- **ارزیابی و بهنگام‌سازی بهترین موقعیت:** دوباره مقدار برازندگی هر ذره محاسبه می‌شود، اگر مقادیر بهتری برای b_i و b_i^n بدست آمده باشد، بهنگام می‌شوند.

۵- **شرط خاتمه الگوریتم:** گام‌های ۲ و ۳ تا زمانیکه شرط خاتمه برآورده شود تکرار می‌شود.

این روش به خاطر داشتن قواعد پیاده‌سازی راحت و همچنین عدم استفاده از اطلاعات گرادیان تابع هدف و سرعت در محاسبات، محبوبیت فراوانی کسب کرده است. در جدول ۲-۱۰ تعدادی از کاربردهای آن بیان

شده است.

جدول ۲-۱۰: کاربردهای الگوریتم ازدحام ذرات

مسائل بهینه‌سازی	تولید نیرو و سیستم‌های قدرت
گرافیک کامپیوتری	پردازش تصویر
برنامه‌ریزی و زمان‌بندی	اقتصاد و بازرگانی
الکترونیک و الکترومغناطیس	متالوژی
طراحی و بهینه‌سازی شبکه‌های ارتباطی	رباتیک
شبیه‌سازی تابع تقاضای انرژی	کنترل

شی[۲۰] و ابرهارت (۱۹۹۸) [۲۸][۲۱] با معرفی پارامتر وزنی w معادله ۲-۲ را به صورت زیر اصلاح کردند:

$$v_{id}^{t+1} = wv_{id}^{t} + c_1 r_1 (b_{id}^{t} - x_{id}^{t}) + c_2 r_2 (b_{id}^{nt} - x_{id}^{t}) \quad (۴-۲)$$

$$w = w_{max} - (w_{max} - w_{min}) \times \frac{Iter_{max}}{Iter} \quad (۵-۲)$$

- w_{max} و w_{min} =وزن‌های اولیه و پایانی (معمولاً از بازه‌ی $[0, 1]$ انتخاب می‌شوند)
- $Iter_{max}$=حداکثر تعداد تکرار
- $Iter$=تعداد تکرار جاری

موریس[۲۲] [۲۶] در سال ۱۹۹۹ برای جلوگیری از افزایش نامحدود سرعت ذره روش مؤثر زیر را پیشنهاد کرد:

$$v_{id}^{t+1} = w\left(v_{id}^{t} + c_1 r_1 (b_{id}^{t} - x_{id}^{t}) + c_2 r_2 (b_{id}^{nt} - x_{id}^{t})\right) \quad (۶-۲)$$

$$w = \frac{2}{|\ 2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\ |} \quad \varphi = c_1 + c_2 > 4 \quad (۷-۲)$$

ضریب w بر روی همگرایی الگوریتم تأثیر مستقیم دارد، در واقع می‌توان به وسیله این ضرایب تأثیر سرعت‌های گذشته را بر سرعت‌های زمان حال کنترل و از رشد بیش از حد سرعت جلوگیری کرد. مقدار زیاد برای w باعث می‌شود ذرات موجود در الگوریتم به جستجوی مناطق جدید روی بیاورند و جستجوی سراسری انجام دهند. در مقابل برای w با مقدار کم ذرات در منطقه می‌مانند و جستجوی محلی انجام می‌شود.

---

[۲۰] Shi [۲۱] Eberhart [۲۲] Maurice

الگوریتم ۲-۲ شبه کد الگوریتم ازدحام ذرات.

ورودی: تعداد جمعیت اولیه، شرط خاتمه.

خروجی: جواب بهینه.

۱: مقداردهی اولیه به موقعیت و سرعت هر ذره.

۲: مقدار برازندگی هر ذره را محاسبه کنید.

۳: برای هر ذره $pbest$ و $nbest$ را محاسبه کنید.

۴: مراحل زیر را تا زمانیکه شرط خاتمه برقرار شود انجام دهید:

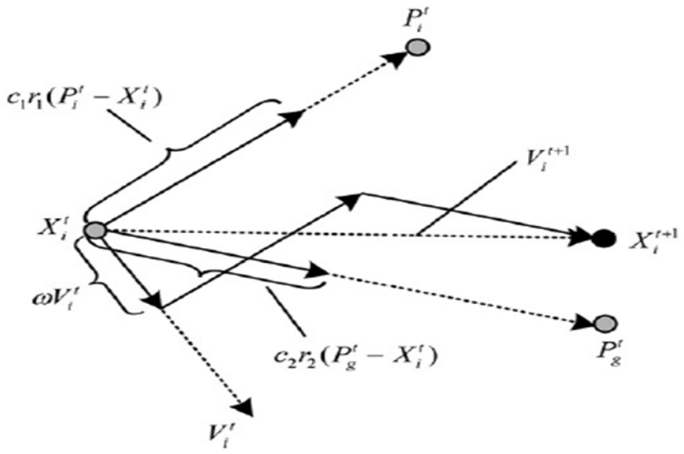
۵: سرعت هر ذره را با استفاده از معادله (۲-۴) بهنگام کنید.

۶: موقعیت هر ذره را با استفاده از معادله (۱-۳) بهنگام کنید.

۷: مقدار برازندگی هر ذره را محاسبه کنید.

۸: اگر مقدار برازندگی جاری ذره از مقدار برازندگی $pbest$ آن بهتر است، $pbest$ را بهنگام کنید.

۹: ذره ای با بهترین برازندگی بین همسایگی‌هایش انتخاب کنید و $nbest$ را بهنگام نمایید.



شکل ۲-۱۱: نحوه‌ی بهنگام شدن موقعیت

مثال ۲-۲-۲. شبکه‌ای با ۲۰ گره را در نظر بگیرید (شکل۱-۳)، می‌خواهیم کوتاه‌ترین مسیر از گره ۱ به گره ۲۰ را به روش pso بیابیم [۱۲].

آ) اندازه‌ی جمعیت : فرض کنید اندازه جمعیت ۳۰ ذره باشد.

ب) فرض کنید $c_1 = c_2 = ۲/۰۵$.

پ) حداکثر تعداد تکرار هم ۵۰ باشد.

الگوریتم ۲-۳ شبه کد برای الگوریتم کدگذاری مسیر.

ورودی: تعداد جمعیت اولیه، شرط خاتمه.

خروجی: جواب بهینه.

۱: فرض کنید $k = ۰$ و $v_P^K = ۱$ و $x = x^k$ و $t^k = ۱$ و $x^k(t^k) = -N_\infty$

۲: اگر $t^k = N_{max}$ یا $k > N_{max}$ به گام ۴ برو ، در غیر این صورت $k = k + ۱$ و به گام ۳ برو.

۳: از بین گره‌هایی که مستقیماً به گره $t^{k-۱}$ متصل هستند، گره t^k که بیشترین اولویت را دارد و $(t^k - t^{k-۱}) > -M$ است را به عنوان گره بعدی انتخاب کنید و قرار دهید $v_p^k = \{v_p^{k-۱}, t^k\}$ و $x^k(t^k) = -N_\infty$.

۴: در صورتی که گره پایانی گره مقصد است، مسیر معتبر v_p^k را برگردان و اگر گره پایانی گره مقصد نیست، مسیر نامعتبر v_p^k را برگردان.

ت) $v_d^{max} = ۳۰۰۰$ و $v_d^{min} = -۳۰۰۰$.

ث) جمعیت اولیه : برای تولید جمعیت اولیه، ابتدا باید هر ذره کدگذاری شود و چون به تعداد ۲۰ گره داریم، لذا بعد مسأله ۲۰ است، یعنی هر ذره دارای برداری با ۲۰ مؤلفه برای سرعت و موقعیت است. در کدگذاری بردار موقعیت از مقداری به نام اولویت استفاده می‌شود، به این صورت که هر مؤلفه‌ی بردار موقعیت به طور تصادفی در بازه‌ی معرفی شده برای اولویت انتخاب می‌شود و برای هر مؤلفه‌ی بردار سرعت ذره نیز، مقداری تصادفی در بازه‌ی معرفی شده برای سرعت انتخاب می‌شود. در این مثال بازه‌ی $[-۱۰۰, ۱۰۰]$ برای بردار اولویت و بازه‌ی $[-۱۰, ۱۰]$ برای بردار سرعت در نظر گرفته شده است.

برای مثال به ترتیب در سطر اول شماره‌ی مؤلفه‌های هر بردار، سطر دوم بردار موقعیت و سطر سوم بردار سرعت یک ذره را نشان می‌دهد.

۱	۲	۳	۴	۵	۶	۷	۸	۹	...	۱۴	۱۵	۱۶	۱۷	۱۸	۱۹	۲۰
۱	۶۵	۷۰	۱۲	۱۷	۶۷	۶۱	۵۶	۱۳	...	۵۵	۱۸	۲۹	۲۳	۲۵	۲۷	۵۱
۵	۳	-۱۰	۷	۱	۰	۴	۰	۱۰	...	-۶	۱۰	۳	۱	۰	۲	۴

ج) مسیر معتبر برای هر ذره با استفاده از الگوریتم ۲-۳ به صورت زیر محاسبه می‌شود:

تکرار صفر: در ابتدا چون گره شماره ۱، گره مبدأ است بنابراین $v_p^۰ = \{۱\}$ است یعنی مسیر در تکرار صفر شامل گره شماره ۱ است و با توجه به گام ۲ از الگوریتم ۲-۳، به مؤلفه اول از بردار موقعیت

مقدار $-N_\infty$ اختصاص داده می‌شود. t^k برابر با شماره گره‌ای با بیشترین اولویت می‌باشد و در این تکرار $t^\circ = ۱$ است، مقدار $M = ۴, q = c_۱ + c_۲, w = ۰/۷۲۹$ در نظر گرفته شده است.

۱	۲	۳	۴	۵	۶	۷	۸	۹	...	۱۴	۱۵	۱۶	۱۷	۱۸	۱۹	۲۰
$-N_\infty$	۶۵	۷۰	۱۲	۱۷	۶۷	۶۱	۵۶	۱۳	...	۵۵	۱۸	۲۹	۲۳	۲۵	۲۷	۵۱

تکرار ۱: با توجه به گام‌های ۲ و ۳ از الگوریتم ۲-۳، از بین گره‌های متصل به گره شماره ۱، گره شماره ۳ انتخاب می‌شود. در این صورت $v_p^۱ = \{۱, ۳\}$ و $t^۱ = ۳$ است و بردار موقعیت بعد از بهنگام شدن به صورت زیر است.

۱	۲	۳	۴	۵	۶	۷	۸	۹	...	۱۴	۱۵	۱۶	۱۷	۱۸	۱۹	۲۰
$-N_\infty$	۶۵	$-N_\infty$	۱۲	۱۷	۶۷	۶۱	۵۶	۱۳	...	۵۵	۱۸	۲۹	۲۳	۲۵	۲۷	۵۱

تکرار ۲: در این تکرار گره ۸ به مسیر اضافه می‌شود و داریم $v_p^۲ = \{۱, ۳, ۸\}$ و $t^۲ = ۸$.

۱	۲	۳	۴	۵	۶	۷	۸	۹	...	۱۴	۱۵	۱۶	۱۷	۱۸	۱۹	۲۰
$-N_\infty$	۶۵	$-N_\infty$	۱۲	۱۷	۶۷	۶۱	$-N_\infty$	۱۳	...	۵۵	۱۸	۲۹	۲۳	۲۵	۲۷	۵۱

اگر همین روند را تا ۸ تکرار ادامه دهیم، مسیر $v_p^۸ = \{۱, ۳, ۸, ۷, ۶, ۱۲, ۱۷, ۱۸, ۲۰\}$ بدست می‌آید که مسیری معتبر است چون به گره مقصد یعنی گره شماره ۲۰ رسیده‌ایم.

چ) برازندگی ذرات از معادله زیر بدست می‌آید:

$$f_i = (\sum_{j=۱}^{N_i - ۱} C_{yz})^{-۱} \qquad (۸-۲)$$

N_i = تعداد گره‌های مسیر ذره i C_{yz} = هزینه اتصال گره y به گره z

شماره ذره	مسیر معتبر ذره در تکرار k (v_p^k)	تعداد گره‌های مسیر ذره	مقدار برازندگی(f_i)
۱	۱, ۳, ۸, ۷, ۶, ۱۲, ۱۷, ۱۸, ۲۰	۹	۰/۰۰۲۰

ح) مقدار $pbest$ و $nbest$ با توجه به مقدار برازندگی محاسبه می شود.

بعد از محاسبه مسیر و مقدار برازندگی برای هر ذره، بردارهای موقعیت و سرعت با استفاده از $pbest$ هر ذره و $nbest$ بهنگام می‌شود. سه ذره زیر را در نظر بگیرید.

برای ذره ۱ با بردار موقعیت زیر، مسیر بدست آمده برابر $v_p = \{۱, ۳, ۸, ۷, ۶, ۱۲, ۱۷, ۱۸, ۲۰\}$ و مقدار برازندگی برابر $f_۱ = ۰/۰۰۲۰$ است.

جدول ۲-۱۱: بردار موقعیت و سرعت ذره ۱

موقعیت	۱	۶۵	۷۰	۱۲	۱۷	۶۷	۶۱	۵۶	...	۱۸	۲۹	۲۳	۲۵	۲۷	۵۱
سرعت	۵	۳	−۱۰	۷	۱	۰	۴	۰	...	۱۰	۳	۱	۰	۲	۴

برای ذره ۲ با بردار موقعیت زیر، مسیر بدست آمده برابر $v_p = \{۱, ۳, ۹, ۱۵, ۱۶, ۱۹, ۲۰\}$ و مقدار برازندگی برابر $f_۲ = ۰/۰۰۲۵$ است.

جدول ۲-۱۲: بردار موقعیت و سرعت ذره ۲

موقعیت	۱	۴۱	۵۹	۳۰	۱۸	۲۵	۱۳	۱۲	...	۴۱	۳۰	۷۰	۲۹	۶	۱۰
سرعت	۱	۶	−۲	۰	۴	۳	۱	۱۰	...	۳	۱	−۹	۷	۶	۳

برای ذره ۳ با بردار موقعیت زیر، مسیر بدست آمده برابر $v_p = \{۱, ۲, ۷, ۶, ۱۲, ۱۳, ۱۸, ۲۰\}$ و مقدار برازندگی برابر $f_۲ = ۰/۰۰۲۲$ است.

جدول ۲-۱۳: بردار موقعیت و سرعت ذره ۳

موقعیت	۱	۶۰	۵۰	۱۰	−۱۰	۷۰	۸۰	...	−۳۵	۹۵	۱۰	۴۵	۸۰	۱۵
سرعت	۲	۳	۱	۰	−۱	۷	−۳	...	۲	−۵	۳	۰	۴	۶

بین این سه ذره، ذره شماره ۲ مقدار برازندگی بیشتری دارد و بنابراین بردار موقعیت آن به عنوان $nbest$ انتخاب می‌شود. در تکرار اول $pbest$ هر ذره برابر بردار موقعیت همان ذره است. با استفاده از معادله ۲-۶ مؤلفه‌های بردار سرعت هر ذره و سپس بردار موقعیت آن‌ها بهنگام می‌شود.

بهترین مسیر برای این روش به صورت زیر است و جدول ۲-۱۴ مسیرهای بدست آمده در هر تکرار را نشان می‌دهد.

$$p : ۱ \to ۳ \to ۸ \to ۱۴ \to ۲۰$$

جدول ۲-۱۴: مسیرهای بدست آمده در هر تکرار از الگوریتم ازدحام ذرات

زمان(ثانیه)	تابع هدف (برازندگی)	مسیر	شماره تکرار
۰/۳۶	۲۳۴	۱ → ۴ → ۹ → ۱۴ → ۲۰	۱
۰/۳۶	۲۳۴	۱ → ۴ → ۹ → ۱۴ → ۲۰	۲
۰/۳۴	۱۴۲	۱ → ۳ → ۸ → ۱۴ → ۲۰	۳
۰/۳۴	۱۴۲	۱ → ۴ → ۹ → ۱۴ → ۲۰	۴
۰/۳۴	۲۴۰	۱ → ۳ → ۹ → ۱۵ → ۱۹ → ۲۰	۵
۰/۳۵	۱۴۲	۱ → ۳ → ۸ → ۱۴ → ۲۰	۶
۰/۳۳	۲۰۸	۱ → ۴ → ۹ → ۱۵ → ۱۹ → ۲۰	۷
۰/۳۵	۱۴۲	۱ → ۳ → ۸ → ۱۴ → ۲۰	۸
۰/۳۶	۲۱۷	۱ → ۲ → ۷ → ۸ → ۱۴ → ۱۸ → ۲۰	۹
⋮	⋮	⋮	⋮
۰/۳۶	۱۴۲	۱ → ۳ → ۸ → ۱۴ → ۲۰	۴۷
۰/۳۱	۱۴۲	۱ → ۳ → ۸ → ۱۴ → ۲۰	۴۸
۰/۳۵	۱۴۲	۱ → ۳ → ۸ → ۱۴ → ۲۰	۴۹
۰/۳۵	۱۴۲	۱ → ۳ → ۸ → ۱۴ → ۲۰	۵۰

فصل ۳

حل مسأله مکان‌یابی با روش‌های فراابتکاری

۳-۱ مقدمه

مطالعات در علم مکان‌یابی توسط آلفرد وبر[۱] [۳۲] آغاز شد، وی اولین کسی بود که مسأله مکان‌یابی[۲] را فرمول‌بندی کرد، یعنی مسأله پیدا کردن مکان یک سرویس‌دهنده را که مجموع فاصله آن تا چند مشتری کمترین مقدار ممکن است را معرفی کرده و مورد بررسی قرار داد. بعد از این مطالعه ابتدایی تعدادی از محققان سعی در بسط و توسعه مفهوم عرضه شده توسط وبر کردند. به عنوان مثال هاتلینگ[۳] مسأله فروشنده بستنی در ساحل را مطرح کرد که هدف آن جذب بیشترین تعداد مشتری در یک بازار ساحلی مشترک بین دو یا چند فروشنده است.

مکان‌یابی در سال ۱۹۶۴ تولد مجددی را توسط حکیمی[۴] [۱۹] تجربه کرد. حکیمی مسأله مکان‌یابی بر روی شبکه را برای پیدا کردن مکان گشت‌های پلیس در بزرگراه‌ها و مناطق شهری مورد استفاده قرار داد. وی مسأله مکان‌یابی را در یک شکل کلی‌تر مطرح کرد، او فرض کرد که تعداد سرویس‌دهنده‌ها (گشت‌های پلیس) بیشتر از یکی باشد بنابراین مشتری‌ها از بین سرویس‌دهنده‌ها کسی را انتخاب می‌کنند که کمترین فاصله را با او داشته باشند. بدین ترتیب از اواسط دهه‌ی ۶۰ مکان‌یابی با پیشرفت شگرفی مواجه شد.

در سال‌های اخیر مطالعات مکان‌یابی به عنوان یکی از عناصر کلیدی در موفقیت و بقای مراکز صنعتی مطرح است. به دلیل پیچیدگی‌های فراوان در اکثر مسائل دنیای واقعی، محاسبات خیلی زیادی برای حل مسائل مرتبط لازم است و نمی‌توان جواب بهینه آن‌ها را در مدت زمان قابل پذیرش بدست آورد. در این میان شناخت هدف‌ها و روش‌هایی برای حل این مسائل، از اهمیت بسیار زیادی برخوردار است. مسأله مکان‌یابی شامل مکان‌یابی یک تعداد از اماکن برای کمینه کردن مجموع هزینه‌های ثابت و هزینه‌های متغیر سرویس‌دهی

---

[۱] Alfered Weber [۲] Facility Location [۳] Hotelling [۴] Hakimi

از این اماکن به محل تقاضاها می‌باشد. حالت‌های مختلف این مسأله عبارتند از:

۱- مسأله مکان‌یابی مراکز با ظرفیت نامحدود(UFLP)

۲- مسأله p-میانه(P-median)

با توجه به نتایج خوب سه روش جستجوی ممنوع، الگوریتم ژنتیک و الگوریتم پرندگان در مقالات متعدد و برای مسائل مختلف، از این سه روش برای حل مسأله UFLP، استفاده خواهیم کرد. در هر بخش، هر الگوریتم را برای مثال‌هایی از سایت OR-Library [۹] که در جدول ۳-۶ نشان داده شده است، آزمایش کرده‌ایم. سپس جواب هر روش را با جواب‌های بهینه‌ی موجود در OR-Library مقایسه می‌کنیم.

نتایج بدست آمده از هر روش را در جدولی در انتهای هر بخش قرار داده‌ایم. هر جدول شامل بهترین جواب بدست آمده، میانگین جواب‌ها، بدترین جواب، کمترین زمان اجرا و درصد خطا پس از ۱۰۰ بار اجرای الگوریتم برای هر نمونه می‌باشد. مقدار خطا از رابطه زیر محاسبه شده است.

$$\frac{f - f_{opt}}{|f_{opt}|} \times ۱۰۰ \qquad (۱-۳)$$

۳-۲ مکان‌یابی مراکز با ظرفیت نامحدود

اگر در مسأله مکان‌یابی ظرفیت اماکن سرویس‌دهنده نامحدود در نظر گرفته شود، مسأله مکان‌یابی نامحدود [۵] نامیده می‌شود. معرفی این مسأله به دهه‌ی ۶۰ برمی‌گردد (کوهن [۶] و هامبورگر [۷] (۱۹۶۳) [۲۵] و بالینسکی [۸] (۱۹۶۵) [۷]) و این مسأله با عنوان UWLP [۹] و SPLP [۱۰] نیز شناخته شده است.

UFLP شامل m مشتری و n مکان به عنوان سرویس‌دهنده می‌تواند توسط یک شبکه با $m+n$ گره (رأس) و mn مسیر (یال) معرفی شود. در این‌گونه مسائل به دلیل نامحدود بودن ظرفیت مراکز، تخصیص یک مشتری به بیش از یک مرکز عرضه سودبخش نخواهد بود. در مدل آن، $f_j > ۰$ هزینه باز بودن مکان سرویس‌دهنده j و $c_{ij} \geqslant ۰$ برای معرفی هزینه سرویس‌دهی مشتری i از مکان j یا اختصاص مشتری i به مکان j استفاده می‌شود. متغیر y_j برای تعیین وضعیت مکان سرویس‌دهنده در این مدل استفاده می‌شود، اگر $y_j = ۱$ باشد یعنی مکان j باز خواهد بود و اگر $x_{ij} = ۱$ باشد، آنگاه مشتری i توسط مکان j سرویس داده

[۵] Uncapacitated Facility Location Problem [۶] Kuhen [۷] Hamburger [۸] Balinski [۹] Uncapacitated Warehouse Location Problem [۱۰] Simple Plant Location Problem

می‌شود. مدل این مسأله به شکل زیر است [۲۹]:

$$\min f(y,x) = \sum_{i=۱}^{m}\sum_{j=۱}^{n} c_{ij}x_{ij} + \sum_{j=۱}^{n} f_j y_j \qquad (۲-۳)$$

s.t.

$$\sum_{j=۱}^{n} x_{ij} = ۱ \quad i = ۱,...,m$$

$$۰ \leqslant x_{ij} \leqslant y_j \quad i = ۱,...,m \quad j = ۱,...,n$$

$$y_j \in \{۰,۱\} \quad j = ۱,...,n$$

الگوریتم‌های دقیق متنوعی برای حل این مسأله استفاده شده است مانند: الگوریتم‌های لاگرانژین[۱۱] (بارسلو[۱۲] و همکارانش (۱۹۹۰)) [۸]، روش دوگان (ارلن‌کاتر[۱۳] (۱۹۷۸)) [۱۰]، روش اولیه- دوگان[۱۴] (کرکل[۱۵] (۱۹۸۹)) [۲۲].

چون UFLP مسأله بهینه‌سازی ترکیبی Np-hard است (کراروپ[۱۶] و پرازان[۱۷] (۱۹۸۳) [۲۳])، الگوریتم‌های دقیق ممکن نیست بتوانند مسائل کاربردی بزرگ را به طور مؤثر حل کنند، اما الگوریتم‌های فراابتکاری برای حل مسائل بزرگ بسیار کارا هستند و راه‌حل‌های خوبی در زمان معقول ارائه می‌نمایند. مطالعات زیادی برای حل این مسأله با روش‌های فراابتکاری وجود دارد که می‌توان به موارد زیر اشاره کرد:

فگارتی[۱۸] و آیدین[۱۹] (۲۰۰۴) [۶] نوع جدیدی از الگوریتم‌های شبیه‌سازی تبریدی را معرفی کردند که جواب‌های خوبی در زمان کوتاه بدست می‌آورد. جن[۲۰] و همکارانش (۱۹۹۶) [۱۱]، راه‌حل شبکه عصبی را به کار بردند. الگوریتم جستجوی ممنوع توسط فوزان[۲۱] و سلطان[۲۲] (۱۹۹۹) [۵] به کار برده شده است که جواب‌های خوبی تولید می‌کند، اما زمان محاسبه قابل توجهی احتیاج دارد. قوش[۲۳] (۲۰۰۳) [۱۳] از روش جستجوی همسایگی استفاده کرد. در سال ۲۰۰۶ سون[۲۴] [۲۹]روش جستجوی ممنوع را با نتایج روش لاگرانژین و روش‌های ابتکاری که توسط قوش گزارش شده بود و همچنین با روش ترکیبی فراابتکاری [۲۵] مقایسه کرد، که جواب روش جستجوی ممنوع در بعضی نمونه‌ها با جواب این الگوریتم‌ها یا برابری می‌کند و یا برتر است.

میچل[۲۶] و هنتزیک[۲۷] (۲۰۰۴) [۲۷] از یک روش جستجوی ممنوع ساده برای حل مسأله UFLP استفاده کردند و معتقدند که این روش در مجموع روش با ارزشی برای حل این مسأله است. کراتیکا[۲۸] از الگوریتم

[۱۱]Relaxation Lagrangian [۱۲]Barcelo [۱۳]Erlenkotter [۱۴]Primal-Dual [۱۵]Koerkel [۱۶]Krarup
[۱۷]Pruzan [۱۸]Fogarty [۱۹]Aydin [۲۰]Gen [۲۱]Al-Fawzan [۲۲]Al-Sultan [۲۳]Ghosh [۲۴]Sun
[۲۵]A hybrid multi start heuristic [۲۶]Michel [۲۷]Hentenric [۲۸]Kractica

ژنتیک استفاده کرد (۱۹۹۶، ۱۹۹۸، ۲۰۰۱) که روش بسیار موفقی بوده است و جواب‌های بهینه را با فراوانی بالا تولید می‌کند [۲۴]. هیرواکی تاهی آما[۲۹] و همکارانش (۲۰۱۱) [۳۰] نیز الگوریتم ژنتیک را به کار بردند و با چند روش مقایسه نمودند. در سال ۲۰۰۶ و ۲۰۰۸ گنر[۳۰] و سوکلی[۳۱] [۱۷، ۱۸] الگوریتم ازدحام ذرات را برای مسأله مکان‌یابی مراکز بدون ظرفیت به کار بردند.

۳-۲-۱ جستجوی ممنوع برای حل مسأله UFLP

در مرجع [۲۷] الگوریتم جستجوی ممنوع به صورت زیر توصیف شده است:

همان‌طور که می‌دانیم این الگوریتم از یک جواب اولیه که اغلب تصادفی است شروع می‌کند، برای مسأله UFLP جواب به صورت برداری از صفر و یک نشان داده می‌شود. بردار $y = (y_1, y_2, ..., y_n)$ را در نظر بگیرید که:

$$y_w = \begin{cases} ۱ & \text{اگر w باز باشد} \\ ۰ & \text{در غیر این صورت} \end{cases}$$

برای معرفی مکان‌هایی که در y باز هستند از نماد زیر استفاده می‌کنیم:

$$Open(y) = \{w \in N | \, y_w = ۱\} \qquad (۳-۳)$$

بعد از تولید جواب اولیه مقدار تابع هدف مسأله برای این جواب محاسبه می‌شود. مجموعه‌ای از n مکان w و m مشتری s را در نظر می‌گیریم. برای مکان w هزینه ثابت f_w و هزینه حمل و نقل از مکان w تا مشتری s، C_{ws} را بکار می‌بریم. مقدار تابع هدف از فرمول زیر بدست می‌آید:

$$obj(Open) = \sum_{w \in Open} f_w + \sum_{s \in Stores} \min_{a \in Open} C_{as} \qquad (۴-۳)$$

در هر مرحله، الگوریتم به جستجوی همسایگی جواب جاری می‌پردازد. همسایگی از تغییر وضعیت یک مکان بدست می‌آید، همسایگی بردار y برای مکان w را با $N(y)$ نشان می‌دهیم، $|N(y)| = n$ و به صورت زیر

[۲۹] Hiroaki tohyama [۳۰] Guner [۳۱] Sevkli

تعریف شده است:

$$\begin{cases} N(y) = \{flip(y,w) |\ w \in W\} \\ flip((y_1,...,y_n),w) = (y_1,...,y_{w-1},!y_w,y_{w+1},...,y_n) \end{cases} \quad (۵-۳)$$

بعد از بررسی کلیه همسایگی‌ها الگوریتم بهترین همسایگی را انتخاب می‌کند و در صورتی که این حرکت جزو لیست ممنوع T نباشد، به آن حرکت می‌کند. مجموعه‌ای از همسایگی‌ها که ممنوع نیستند به صورت زیر انتخاب می‌شوند:

$$N^T(y) = \{flip(y,w) |\ \ w \in W \setminus T\} \quad (۶-۳)$$

بهترین مقدار تابع هدف از میان همسایگی‌ها با استفاده از رابطه زیر بدست می‌آید:

$$bestObj(y) = \max_{e \in N^T(y)} obj(Open(e)) \quad (۷-۳)$$

مجموعه ای از همسایگی‌ها که توسط جستجوی ممنوع در نظر گرفته شده است:

$$N^*(y) = \{e \in N^T(y) |\ obj(Open(e)) = bestObj(y)\} \quad (۸-۳)$$

اگر همسایگی‌ها در $N^*(y)$ جواب جاری را بهبود دهند، الگوریتم به طور تصادفی به یکی از این همسایگی‌ها، به عنوان مثال مکان w حرکت می‌کند. مکان w برای تعدادی تکرار در لیست ممنوع قرار می‌گیرد و طول لیست ممنوع بهنگام می‌شود. در غیر این صورت اگر همسایگی‌ها در $N^*(y)$ جواب جاری را بهبود ندهند، به طور تصادفی مکانی باز انتخاب و بسته می‌شود.

الگوریتم همچنین از عمل بهترین تغییر ($bestFlips$) که شامل باز یا بسته کردن مکان‌هایی است که با تغییر وضعیت آن‌ها جواب بهبود می‌یابد و عمل بهترین نتیجه ($bestGain$) را که تفاوت مقدار تابع هدف جاری و مقدار تابع هدف حاصل از بهترین همسایگی است، استفاده می‌کند:

$$bestFlips(y) = \{w | flip(y,w) \in N^*(y)\} \quad (۹-۳)$$

$$bestGain(y) = obj(y) - bestObj(y) \quad (۱۰-۳)$$

در هر تکرار اگر مقدار $bestGain$ غیر منفی باشد، الگوریتم به طور تصادفی مکانی را در $bestFlips(y)$ انتخاب می‌کند و طول لیست ممنوع بهنگام می‌شود. اگر مقدار آن منفی باشد الگوریتم به طور تصادفی مکانی باز انتخاب می‌کند و آن را می‌بندد، تا یک جواب جدید از میان همسایه‌ها انتخاب شود.

یک مقدار ثابت برای اعتبار تابو نمی‌تواند جستجوی قدرتمندی را ایجاد کند. برای غلبه بر این مشکل در مرجع [۲۷] از اعتبار تابوی متغیر استفاده شده است. لیست تابو برای هر مکان w، به شمارنده $t[w]$ وابسته شده است به طوری که $t[w] = it + tlen$ و it شمارنده تکرار و $tlen = ۱۰$ مدت ممنوعیت حرکت را نشان می‌دهد که با تغییر $tlen$ در هر تکرار اعتبار تابو تغییر می‌کند. اگر $t[w] \geqslant it$ باشد مکان w در لیست ممنوع قرار دارد.

در جدول ۳-۱ مقادیر حاصل از روش جستجوی ممنوع برای مثال‌هایی از سایت OR-Library نشان داده شده است.

جدول ۳-۱: نتایج حاصل از روش جستجوی ممنوع

نمونه	اندازه	بهترین	میانگین	بدترین	زمان(ثانیه)	خطا
$cap۷۱$	۱۶ × ۵۰	۹۳۲۶۱۵/۷۵	۹۳۳۰۹۰/۷۷	۹۳۴۱۹۹/۱۴	۰/۳۵	۰/۰۰
$cap۷۲$	۱۶ × ۵۰	۹۷۷۷۹۹/۴۰	۹۷۹۵۴۱/۶۰	۹۸۳۷۱۳/۹۰	۰/۳۸	۰/۰۰
$cap۷۳$	۱۶ × ۵۰	۱۰۱۰۶۴۱/۴۵	۱۰۱۲۱۷۷/۶۴	۱۰۱۴۴۹۱/۴۰	۰/۳۶	۰/۰۰
$cap۷۴$	۱۶ × ۵۰	۱۰۳۴۹۷۶/۹۸	۱۰۳۴۹۷۶/۹۸	۱۰۳۷۷۱۷/۰۸	۰/۳۶	۰/۰۰
$cap۱۰۱$	۲۵ × ۵۰	۷۹۶۶۴۸/۴۳	۷۹۸۲۵۳/۲۰	۸۰۲۱۹۱/۲۸	۰/۵۹	۰/۰۰
$cap۱۰۲$	۲۵ × ۵۰	۸۵۴۷۰۴/۲۰	۸۵۶۲۴۷/۶۵	۸۶۲۲۹۳/۰۰	۰/۶۰	۰/۰۰
$cap۱۰۳$	۲۵ × ۵۰	۸۹۳۷۸۲/۱۱	۸۹۵۸۹۶/۸۶	۹۰۳۳۳۳/۷۲	۰/۶۱	۰/۰۰
$cap۱۰۴$	۲۵ × ۵۰	۹۲۸۹۴۱/۷۵	۹۳۳۱۵۷/۷۵	۹۴۹۶۵۳/۷۵	۰/۶۰	۰/۰۰
$cap۱۳۱$	۵۰ × ۵۰	۷۹۳۴۳۹/۵۶	۷۹۸۵۹۲/۶۶	۸۰۶۳۱۲/۶۹	۱/۲۲	۰/۰۰
$cap۱۳۲$	۵۰ × ۵۰	۸۵۱۴۹۵/۳۲	۸۵۵۴۳۸/۱۴	۸۶۴۱۶۱/۹۴	۱/۱۶	۰/۰۰
$cap۱۳۳$	۵۰ × ۵۰	۸۹۳۰۷۶/۷۱	۸۹۶۳۰۱/۷۷	۹۰۸۸۵۳/۰۶	۱/۹۱	۰/۰۰
$cap۱۳۴$	۵۰ × ۵۰	۹۲۸۹۴۱/۷۵	۹۳۴۲۳۷/۳۴	۹۴۵۴۳۸/۰۷	۱/۱۴	۰/۰۰

۳-۲-۲ الگوریتم ژنتیک برای حل مسأله UFLP

الگوریتم ژنتیک نیز روش بسیار خوبی برای حل مسأله مکان‌یابی بدون ظرفیت است. در این پایان‌نامه از جعبه‌ابزار الگوریتم ژنتیک در نرم‌افزار MATLAB، برای حل مسأله UFLP استفاده کردیم.

برای ورود پارامترها به جعبه ابزار الگوریتم ژنتیک از تابع $gaoptimset$ استفاده می‌شود. برای مثال، برای تغییر اندازه جمعیت[۳۲]، در پنجره دستورات MATLAB عبارت زیر را وارد کنید و به جای عبارت $value$، مقدار دلخواه را قرار دهید.

$$options = gaoptimset('PopulationSize', value)$$

برای این مسأله اندازه جمعیت برابر با ۵۰ کروموزوم و نوع کدگذاری[۳۳]، صفر و یک[۳۴] در نظر گرفته شده است. مقدار یک در هر ژن از کروموزوم، تأسیس یک سرویس‌دهنده را در آن مکان نشان می‌دهد و مقدار صفر در هر ژن نیز نشان‌دهنده‌ی این است که سرویس‌دهنده‌ای در آن مکان تأسیس نشده است.

$$options = gaoptimset('PopulationSize', ۵۰)$$

$$options = gaoptimset('PopulationType','bitstring')$$

در پیاده‌سازی الگوریتم ژنتیک، برازندگی هر کروموزوم، با استفاده از تابع اصلی که قصد مینیمم کردن آن را داریم، محاسبه می‌شود. عملگر ترکیب از نوع یکنواخت و احتمال ترکیب $p_c = ۰/۸$ در نظر گرفته شده است. در صورتی که مقادیر ورودی برای پارامترهای الگوریتم ژنتیک در این نرم‌افزار تعریف نشود، نرم‌افزار از مقادیر پیش‌فرض برای حل مسأله استفاده می‌کند.

نتایج حاصل از این الگوریتم در جدول ۳-۲ نشان داده شده است.

۳-۲-۳ الگوریتم ازدحام ذرات برای حل مسأله UFLP

در مرجع [۱۷] الگوریتم ازدحام ذرات [۳۵] برای حل مسأله UFLP پیشنهاد شده است. الگوریتم PSO برای هر ذره سه بردار موقعیت (X_i)، سرعت (V_i) و وضعیت مکان Y_i را در نظر می‌گیرد. برای مسأله‌ای با n سرویس‌دهنده، هر ذره شامل n بعد است. در بردار $Y_i = [y_{i۱}, ..., y_{in}]$، y_{ik} باز یا بسته بودن kامین مکان از

[۳۲] Population Size [۳۳] Population Type [۳۴] Bit String [۳۵] Particle Swarm Optimization

جدول ۳-۲: نتایج حاصل از الگوریتم ژنتیک

نمونه	اندازه	بهترین	میانگین	بدترین	زمان(ثانیه)	خطا
cap۷۱	۱۶ × ۵۰	۹۳۲۶۱۵/۷۵	۹۳۲۷۸۹/۹۲	۹۳۴۱۹۹/۱۴	۰/۱۴	۰/۰۰
cap۷۲	۱۶ × ۵۰	۹۷۷۷۹۹/۴۰	۹۷۸۶۵۶/۵۶	۹۸۳۷۱۳/۸۱	۰/۱۴	۰/۰۰
cap۷۳	۱۶ × ۵۰	۱۰۱۰۶۴۱/۴۵	۱۰۱۱۳۱۹/۹۰	۱۰۱۴۲۵۳/۴۴	۰/۱۴	۰/۰۰
cap۷۴	۱۶ × ۵۰	۱۰۳۴۹۷۶/۹۸	۱۰۳۵۷۹۹/۰۱	۱۰۳۷۷۱۷۰/۷۵	۰/۱۴	۰/۰۰
cap۱۰۱	۲۵ × ۵۰	۷۹۶۶۴۸/۴۳	۷۹۷۹۹۵/۵۴	۸۰۰۰۰۴/۹۹۱	۰/۱۵	۰/۰۰
cap۱۰۲	۲۵ × ۵۰	۸۵۴۷۰۴/۲۰	۸۵۵۹۹۲/۵۰	۸۶۳۴۵۵/۹۴	۰/۱۴	۰/۰۰
cap۱۰۳	۲۵ × ۵۰	۸۹۳۷۸۲/۱۱	۸۹۵۱۳۱/۸۴	۸۹۹۹۰۸/۱۳	۰/۱۴	۰/۰۰
cap۱۰۴	۲۵ × ۵۰	۹۲۸۹۴۱/۷۵	۹۳۱۴۹۹/۲۷	۹۴۹۶۵۳/۷۵	۰/۱۵	۰/۰۰
cap۱۳۱	۵۰ × ۵۰	۷۹۳۴۳۹/۵۶	۷۹۸۹۶۴/۴۹	۸۰۶۰۰۶/۴۶	۰/۱۶	۰/۰۰
cap۱۳۲	۵۰ × ۵۰	۸۵۱۴۹۵/۳۲	۸۵۷۱۶۲/۵۳	۸۶۴۹۲۸/۹۹	۰/۱۶	۰/۰۰
cap۱۳۳	۵۰ × ۵۰	۸۹۳۰۷۶/۷۰	۸۹۸۳۵۴/۸۰	۹۱۴۵۳۲/۶۹	۰/۱۷	۰/۰۰
cap۱۳۴	۵۰ × ۵۰	۹۲۸۹۴۱/۷۵	۹۳۴۹۲۶/۶۰	۹۶۲۹۵۱/۰۹	۰/۱۶	۰/۰۰

iامین ذره را نشان می‌دهد. بردارهای موقعیت و سرعت از روابط زیر بدست می‌آیند:

$$x_{ij} = x_{min} + (x_{max} - x_{min}) \times r_۱ \quad i = ۱, ..., N_s \ , \ j = ۱, ..., D \tag{۱۱-۳}$$

$$v_{ij} = v_{min} + (v_{max} - v_{min}) \times r_۲ \tag{۱۲-۳}$$

که $x_{min} = -۱۰$، $x_{max} = ۱۰$، $v_{min} = -۴$، $v_{max} = ۴$ و $r_۱$ و $r_۲$ به طور تصادفی از بازه‌ی $[۰, ۱]$ برای هر بعد ذره انتخاب می‌شوند. برای ساختن جواب اولیه، بردار موقعیت به یک بردار صفر و یک تبدیل می‌شود $(Y_i \longleftarrow X_i)$، برای این منظور از رابطه زیر استفاده می‌شود. در این رابطه قدرمطلق بردار موقعیت را بر عدد ۲ تقسیم می‌کنیم، سپس باقی‌مانده به سمت نزدیکترین عدد صحیح گرد می‌شود.

$$Y_i = \lfloor |X_i| (mod ۲) \rfloor \tag{۱۳-۳}$$

مثال ۳-۲-۱. فرض کنید مسأله‌ای با ۵ سرویس‌دهنده داریم، با توجه به جدول ۳-۳، برای مؤلفه پنجم از $Y_۵$

داریم:

$$\lfloor |-۵/۴۵|(mod۲) \rfloor = \lfloor ۵/۴۵(mod۲) \rfloor = \lfloor ۱/۴۵ \rfloor = ۱$$

سایر مؤلفه‌های بردار موقعیت به همین طریق محاسبه می‌شوند.

جدول ۳-۳: مثالی برای توضیح ساخت بردار موقعیت

بردارهای ذره iام	بعد ذره				
	۱	۲	۳	۴	۵
بردار موقعیت (X_i)	۱/۸	۳/۰۱	-۰/۹۹	۰/۷۲	-۵/۴۵
بردار سرعت (V_i)	-۰/۵۲	۲/۰۶	۳/۵۶	۲/۴۵	-۱/۴۴
بردار وضعیت (Y_i)	۱	۱	۰	۰	۱

اکنون برازندگی iامین ذره توسط بردار Y_i محاسبه می‌شود. جدول ۳-۴ هزینه‌های ثابت و هزینه سرویس‌دهی برای ۵ سرویس‌دهنده و ۶ مشتری را نشان می‌دهد.

جدول ۳-۴: سرویس‌دهی از ۵ سرویس‌دهنده به ۶ مشتری

اماکن سرویس‌دهنده		۱	۲	۳	۴	۵
هزینه ثابت سرویس‌دهنده (f_w)		۱۲	۵	۳	۷	۹
هزینه عرضه به مشتری (C_{ws})	۱	۲	۳	۶	۷	۱
	۲	۰	۵	۸	۴	۱۲
	۳	۱۱	۶	۱۴	۵	۸
	۴	۱۹	۱۸	۲۱	۱۶	۱۳
	۵	۳	۹	۸	۷	۱۰
	۶	۴	۷	۹	۶	۰

فرض کنید سرویس‌دهنده‌های ۱ و ۲ و ۵ باز باشند $(Open = \{۱, ۲, ۵\})$، هزینه کلی برای این سرویس‌دهی از رابطه زیر بدست می‌آید.

$$totalcost = \sum_{w \in Open} f_w + \sum_{s \in Customers} \min_{w \in Open} C_{ws}$$

$$totalcost = \{(۱۲ + ۵ + ۹) + \min(۲, ۳, ۱) + \min(۰, ۵, ۱۲) + \min(۱۱, ۶, ۸)$$
$$+ \min(۱۹, ۱۸, ۱۳) + \min(۳, ۹, ۱۰) + \min(۴, ۷, ۰)\}$$
$$= \{۲۶ + (۱ + ۰ + ۶ + ۱۳ + ۳ + ۰)\} = \{۲۶ + ۲۳\} = ۴۹$$

نتایج بدست آمده از الگوریتم ازدحام ذرات در جدول ۳-۵ نشان داده شده است.

جدول ۳-۵: نتایج حاصل از الگوریتم ازدحام ذرات

نمونه	اندازه	بهترین	میانگین	بدترین	زمان(ثانیه)	خطا
cap۷۱	۱۶ × ۵۰	۹۳۲۶۱۵/۷۵	۹۳۵۰۹۴/۵۱	۹۴۱۴۳۹/۷۷	۰/۷۸	۰/۰۰
cap۷۲	۱۶ × ۵۰	۹۷۷۷۹۹/۴۰	۹۸۰۵۵۵/۵۲	۹۸۹۶۰۸/۶۰	۰/۷۸	۰/۰۰
cap۷۳	۱۶ × ۵۰	۱۰۱۰۶۴۱/۴۵	۱۰۱۲۲۶۹/۱۸	۱۰۱۸۶۱۷/۳۹	۰/۷۹	۰/۰۰
cap۷۴	۱۶ × ۵۰	۱۰۳۴۹۷۶/۹۸	۱۰۳۸۹۳۶/۲۷	۱۰۵۶۲۳۴/۳۶	۰/۷۹	۰/۰۰
cap۱۰۱	۲۵ × ۵۰	۷۹۶۶۴۸/۴۳	۸۰۵۹۴۷/۰۸	۸۱۹۲۲۸/۰۰	۱/۴۵	۰/۰۰
cap۱۰۲	۲۵ × ۵۰	۸۵۴۷۰۴/۲۰	۸۶۳۱۰۱/۶۱	۸۷۸۵۱۸/۷۰	۱/۵۰	۰/۰۰
cap۱۰۳	۲۵ × ۵۰	۸۹۴۵۷۳/۷۱	۹۰۳۹۰۶/۲۱	۹۱۴۳۱۳/۵۶	۱/۵۰	۰/۰۹
cap۱۰۴	۲۵ × ۵۰	۹۲۸۹۴۱/۷۵	۹۴۶۶۶۳/۶۸	۹۷۷۰۰۷/۳۰	۱/۵۰	۰/۰۰
cap۱۳۱	۵۰ × ۵۰	۸۰۵۶۵۹/۱۷	۸۲۵۹۹۳/۹۰	۸۴۶۳۶۳/۳۱	۴/۴۲	۱/۵۴
cap۱۳۲	۵۰ × ۵۰	۸۶۱۸۴۵/۸۵	۸۹۵۱۲۱/۴۶	۹۲۱۱۴۰/۵۴	۴/۴۰	۱/۲۱
cap۱۳۳	۵۰ × ۵۰	۹۰۶۸۴۷/۳۶	۹۴۴۵۸۳/۸۹	۹۸۷۳۷۵/۰۹	۴/۴۰	۱/۵۴
cap۱۳۴	۵۰ × ۵۰	۹۲۹۴۷۷/۵۶	۱۰۰۱۲۱۳/۳۵	۱۰۸۰۹۴۱/۷۹	۴/۴۰	۰/۰۶

۳-۳ نتایج

سه روش فراابتکاری فوق‌الذکر در یک محیط یکسان در نرم افزار MATLAB پیاده‌سازی گردیدند و نتایج آن‌ها از منظرهای مختلف مورد مقایسه واقع گردید. پارامترهای استفاده شده در هر یک از سه روش به شرح زیر است:

در روش جستجوی ممنوع مدت ممنوع بودن هر حرکت تا ۱۰ تکرار است و این مقدار برای هر جواب تغییر می‌یابد. جمعیت اولیه در الگوریتم ژنتیک ۵۰ کروموزوم می‌باشد. جمعیت اولیه الگوریتم ازدحام ذرات نیز برای هر مثال برابر با تعداد اماکن سرویس‌دهنده، ضرایب $C_۱$ و $C_۲$ برابر ۲/۰۵ در نظر گرفته شده است.

در مقالات عموماً به منظور مقایسه، زمان اجرا و درصد رسیدن به جواب بهینه گزارش می‌شود. در این پایان‌نامه بر اساس معیار پیشنهادی **همگرایی**، موارد قبلی به صورت دیگری تعریف و مقایسات انجام شده‌اند. معیار پیشنهادی یک مقایسه ساده میزان خطای خروجی روش است، لیکن تا آنجا که بررسی شده است، مقایسات قبلی به این ترتیب انجام نشده است و به نظر مؤلفان، این شیوه‌ی مقایسه دید روشن‌تری از کارایی الگوریتم‌ها را ارائه می‌کند.

تعریف ۳-۳-۱. معیار همگرایی؛ گوییم یک الگوریتم همگرا شده است، هرگاه: $err \leq \epsilon$

که در آن err میزان خطای الگوریتم (اختلاف جواب تولید شده با جواب بهینه) و ϵ یک حد آستانه مناسب است. با استفاده از تعریف فوق، معیارهای زیر را تعریف می‌کنیم:

تعریف ۳-۳-۲. زمان همگرایی؛ زمان سپری شده از ابتدای اجرای الگوریتم تا همگرا شدن الگوریتم.

تعریف ۳-۳-۳. فرکانس همگرایی؛ درصد اجراهایی که الگوریتم همگرا شده است.

تعریف ۳-۳-۴. تعداد تکرار تا همگرایی؛ تعداد دفعات تکرار حلقه‌ی اصلی الگوریتم تا همگرایی.

هر روش روی ۱۲ مثال از OR-Library که مشخصات آن‌ها در جدول ۳-۶ آورده شده است، ۱۰۰ بار اجرا شد، نتایج در شکل‌های ۳-۱ تا ۳-۴ نشان داده شده است. محور افقی در هر شکل شماره مثال‌ها را نشان می‌دهد. در کادر راهنما، میانگین کلی هر روش نیز نشان داده شده است. هر اجرای هر الگوریتم وقتی یکی از

جدول ۳-۶: مشخصات مثال‌های شماره ۱ تا ۱۲

شماره	مثال	اندازه	شماره	مثال	اندازه	شماره	مثال	اندازه
۱	cap۷۱	۱۶×۵۰	۵	cap۱۰۱	۲۵×۵۰	۹	cap۱۳۱	۵۰×۵۰
۲	cap۷۲	۱۶×۵۰	۶	cap۱۰۲	۲۵×۵۰	۱۰	cap۱۳۲	۵۰×۵۰
۳	cap۷۳	۱۶×۵۰	۷	cap۱۰۳	۲۵×۵۰	۱۱	cap۱۳۳	۵۰×۵۰
۴	cap۷۴	۱۶×۵۰	۸	cap۱۰۴	۲۵×۵۰	۱۲	cap۱۳۴	۵۰×۵۰

سه شرط: (الف) همگرا شدن، (ب) رسیدن تعداد تکرارها به مقدار از پیش تعیین شده یا (ج) حداکثر زمان از قبل مشخص شده، صادق شوند خاتمه می‌یابد. در هر آزمایش ϵ، حد آستانه‌ی همگرایی، حداکثر زمان اجرا و حداکثر تعداد تکرار هر سه الگوریتم یکسان و مناسب آزمایش مدنظر درنظر گرفته شده است.

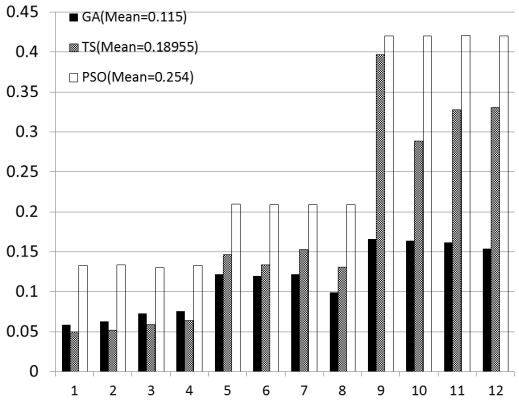
شکل ۳-۱ میانگین زمان تا همگرا شدن هر الگوریتم را نشان می‌دهد (بر حسب ثانیه). همان‌گونه که مشاهده می‌شود، GA کمترین و PSO بیشترین زمان محاسباتی را دارند. زمان همگرایی PSO با افزایش اندازه

مسأله، به سرعت افزایش پیدا کرده است. در این آزمایش، ϵ، حد آستانه‌ی همگرایی، حداکثر زمان اجرا و حداکثر تعداد تکرار به اندازه‌ای بزرگ درنظر گرفته شده‌اند که حتماً هر سه الگوریتم همگرا شوند.

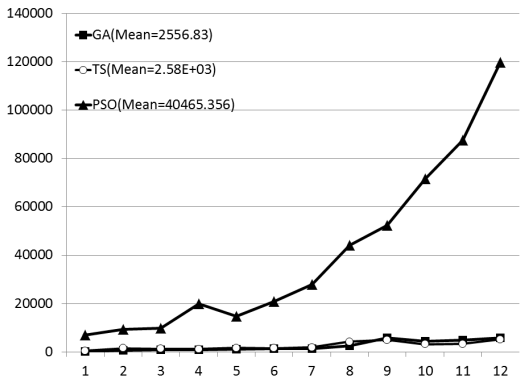
شکل ۳-۲ میانگین خطای سه روش را نشان می‌دهد. در این آزمایش برای مقادیر مختلف حد آستانه‌ی همگرایی، حداکثر زمان اجرا و حداکثر تعداد تکرار، GA کمترین و PSO بیشترین خطا را داشته است.

شکل ۳-۳ فرکانس همگرایی را نشان می‌دهد. در این آزمایش، حد آستانه‌ی همگرایی، حداکثر زمان اجرا و حداکثر تعداد تکرار به اندازه‌ای درنظر گرفته شده‌اند که الگوریتم‌ها همیشه همگرا نشوند و بتوان آن‌ها را بهتر مقایسه کرد. همان‌گونه که مشاهده می‌شود الگوریتم GA بیشترین نرخ همگرایی و الگوریتم PSO کمترین نرخ همگرایی را دارد.

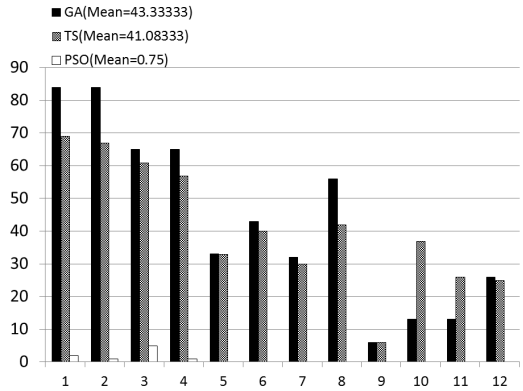
شکل ۳-۴ میانگین تعداد تکرار تا همگرایی را نشان می‌دهد. مطابق این نمودار، GA کمترین و PSO بیشترین تعداد تکرار را داشته اشت. به علاوه الگوریتم GA کمترین نرخ افزایش در تعداد تکرار مورد نیاز تا همگرا شدن، با افزایش اندازه مسأله را داشته است.



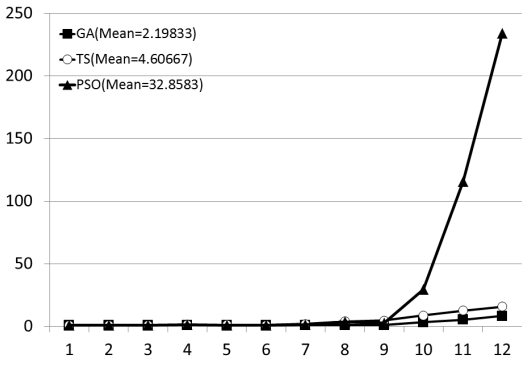
شکل ۳-۱: میانگین زمان تا همگرایی



شکل ۳-۲: میانگین خطا



شکل ۳-۳: فرکانس همگرایی



شکل ۳-۴: میانگین تعداد تکرار تا همگرایی

۳-۳-۱ نتیجه‌گیری

نظر به نتایج فوق‌الذکر، حداقل با فرم پیاده‌سازی شده توسط مؤلفین، الگوریتم ژنتیک کارایی بیشتری در حل مسأله‌ی UFLP دارد. فرم ساده‌ی الگوریتم‌های فوق مورد مقایسه قرار گرفتند و بی‌شک تغییراتی در هر یک از آن‌ها می‌تواند نتایج را تحت تاثیر قرار دهد.

مراجع

[۱] طاهری حسن. حل مسأله تخصیص درجه ۲ با روش‌های فراابتکاری. [پایان‌نامه]. مشهد: دانشگاه فردوسی؛ ۱۳۸۱.

[۲] عالم تبریز اکبر، زندیه مصطفی، محمد رحیمی علی‌رضا. الگوریتم‌های فراابتکاری در بهینه‌سازی ترکیبی. تهران: انتشارات صفار، ۱۳۸۷.

[۳] عباسی‌کیا مصطفی. الگوریتم‌های فرااکتشافی جستجو. ناشر: www.irpdf.com.

[۴] فتاحی پرویز. الگوریتم‌های فراابتکاری. همدان: انتشارات دانشگاه بوعلی سینا، ۱۳۸۸.

[5] Al-Fawzan, Al-Sultan. Tabu Search Approach to the Uncapacitated Facility Location Problem. Annals of Operations Research 1999;86:91–103.

[6] Aydin ME, Fogarty TC. A Distributed Evolutionary Simulated Annealing Algorithm for Combinatorial Optimisation Problems. Journal of Heuristics 2004;10:269–292.

[7] Balinski ML. Integer programming: methods, uses, computation. Management Science 1965;12:253-313.

[8] Barcelo J, Hallefjord A, Fernandez E, Jo¨rnsten K. Lagrangean Relaxation and Constraint Generation Procedures for Capacitated Plant Location Problems with Single Sourcing. OR Spektrum 1990;12:79–88.

[9] Beasley JE. OR-Library, http://people.brunel.ac.uk/ mastjjb/job/info.html.

[10] Erlenkotter D. A Dual-based Procedure for Uncapacitated Facility Location: General solution procedures and computational experience. Operations Research 1978;26:992–1009.

[11] Gen M, Tsujimura Y, Ishizaki S. Optimal Design of A Star-LAN using Neural Networks. Computers And Industrial Engineering 1996;31:855-859.

[12] Geok TK, Mohemmed AW, Sahoo NC. Solving Shortest Path Problem using Particle Swarm Optimization. Applied Soft Computing 2008;8:1643-1653.

[13] Ghosh D. Neighborhood Search Heuristics for the Uncapacitated Facility Location Problem. European Journal of Operational Research 2003;150:150-162.

[14] Glover F, Laguna M. Tabu search. Kluwer Academic Publishers, Boston/Dordrecht/London 1997.

[15] Glover F. Future paths for integer programming and links to artificial intelligence. Computers Operations Research 1986;533-549.

[16] Goldberg DE. Genetic algorithms in search, Optimization and Machine Learning. Addison-Wesley Menlo Park CA 1989.

[17] Guner A, Sevkli M. A Discrete Swarm Optimization Algorithm for Uncapacitated Facility Location Problem. Journal of Artificial Evolution and Applications 2008;1-9.

[18] Guner A, Sevkli M. A Continuous Particle Swarm Optimization Algorithm for Uncapacitated Facility Location Problem. Computer Science Springer Berlin Germany 2006;4150:316–323.

[19] Hakimi SL. Optimum Locations Of Switching Centers and the Absolute Centers and Medians of a Graph. Operation Research 1964;12:450-459.

[20] Holland J. Adaptation in natural and artificial systems. University of Michigan Press Ann Arbor 1975.

[21] Kennedy J, Eberhart RC. Particle Swarm Optimization. in: Proceedings of the IEEE International Conference on Neural Networks 1995;1942–1948.

[22] Koerkel M. On the Exact Solution of Large-scale Simple Plant Location Problems. European Journal of Operations Research 1989;39 :157–173.

[23] Krarup J, Pruzan PM. The simple plant location problem: survey and synthesis. European Journal of Operational Research 1983;12:36-81.

[24] Kratica J, Filipovic V, Tosic D. Solving the Uncapacitated Warehouse Location Problem By SGA with ADD-Heuristic. Proc XV ECPU International Conf on Material Handling and Warehousing 1998;2.28-2.32.

[25] Kuehn AA, Hamburger MJ. A Heuristic Program for Locating Warehouses. Management Science 1963;9:643–666.

[26] Maurice C. The swarm and queen: Towards a deterministic and adaptive particle swarm optimization. in: Proceedings of the IEEE Congress on Evolutionary Computation 1999;1951–1957.

[27] Michel L, Hentenryck PV. A Simple Tabu Search Warehouse Location. European Journal of Operational Research 2004;157: 576-591.

[28] Shi Y, Eberhart RC. A Modified Particle Swarm Optimizer. In: Proceedings of IEEE International Conference on Evolutionary Computation, Anchorage, Alaska 1998;66–73.

[29] Sun M. Solving the Uncapacitated Facility Location Problem using Tabu Search. Computers and Operations Research 2006;33:2563-2589.

[30] Tohyama H, Ida K, Matsueda J. A Genetic Algorithm for the Uncapacitated Facility Location Problem. Electronics and Communications in Japan 2011;49.

[31] Tsai C-Y, kao I-W. Particle Swarm Optimization with Selective Particle Regeneration for Data Clustering. Expert Systems With Application 2011;38:6565-6576.

[32] Weber A. Uber den standort der industrient. University of Chicago 1929.

پیوست آ

کد الگوریتم ازدحام ذرات

در این پیوست، قسمت اصلی کد مربوط به حل مسأله UFLP با استفاده از الگوریتم ازدحام ذرات که با نرم‌افزار MATLAB نوشته شده است، آورده می‌شود.

```
function [x,fval] = UWLP_PSO(problem)
%Initialize the position and velocity of each particle
for j=1:nP
    r1=rand;  r2=rand;
       p(j,1:nW)=x_min+((x_max-x_min)*r1);
       v(j,1:nW)=v_min+((v_max-v_min)*r2);
end
%Evaluate fitness value of each particle
for j=1:nP
   F_pri(j,1)=uwlp_objfun_pso(p(j,:));
end
[min_Fit,k]=min(F_pri);   pbest=p;   nbest=p(k,:);
while(count<MaxIt)
     for j=1:nP
        for i=1:nW
            % Update Velocity
            v(j,i)=w*v(j,i)+c1*rand*(pbest(j,i)-p(j,i))+c2*
               rand*(nbest(1,i)-p(j,i));
           % Apply Velocity Limits
             v(j,i)=max(v(j,i),v_min);
             v(j,i)=min(v(j,i),v_max);
            % Update Position
             x(j,i)=v(j,i)+p(j,i);
            % Velocity Mirror Effect
             if (x(j,i)<x_min | x(j,i)>x_max)
              v(j,i)=-v(j,i);
             end
        end
      end
   % update nbest
    for j=1:nP
       F(j)=uwlp_objfun_pso(x(j,:));
    end
    [min_F,id]=min(F);
      if F(id)< uwlp_objfun_pso(nbest)
         nbest=x(id,:);
      end
   % update pbest
    for j=1:nP
      if F(j)< uwlp_objfun_pso(p(j,:))
          p(j,:)=x(j,:);    pbest(j,:)=p(j,:);
      end
    end
 count=count+1;
end
 fval=uwlp_objfun_pso(nbest);  x=nbest;
```

شکل آ-۱: کد الگوریتم ازدحام ذرات

Hakim Sabzevari University

Information form for M.Sc. Thesis

Surname: Shahi	**Name:** Samira	**Student no:** 9013133046
Supervisor: Dr. Mehdi Zaferanieh	**Advisor:** Dr. Mahmood Amintoosi	
Faculty: Mathematics and Computer Science		
Major: Applied Mathematics	**Field of study:** Operation Research	
Title of thesis: Solving the faciliy location problems with metaheuristic methods		

Key words:

Location, Tabu search, Genetic algorithm, Particle swarm optimization.

Abstract:

The uncapacitated facility location problem plays a major role in location theory. The UFLP problem is to find the location of some service provider in which the summation of fixed cost and the varying costs between costumer and service provider is minimized.

In this dissertation, the UFLP problem that is an NP-hard problem is solved by three metaheuristic methods including the genetic algorithm, tabu search method and particle swarm optimization. Noting the efficiency and robustness of metaheuristic algorithms we solved the UFLP problem in reasonable cpu time.

The main aim in this dissertation, is comparing the efficiency of the three foregoing metaheuristic methods by comparing their convergent frequency, mean error and cpu time to solve the UFLP problem. Noting to the results the genetic algorithm is most efficient.

Hakim Sabzevari University
Faculty of Mathematics and Computer Science

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

Solving location problems using meta-heuristic methods

Supervisor:

Dr. Mehdi Zaferanieh

Advisor:

Dr. Mahmood Amintoosi

By:

Samira Shahi

September 2013