بسم الله الرحمن الرحیم

دانشگاه حکیم سبزواری

دانشکده ریاضی و علوم کامپیوتر

پایان‌نامه برای دریافت درجه کارشناسی ارشد در رشته ریاضی کاربردی

گرایش تحقیق در عملیات

حل مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود با الگوریتم خفاش

استاد راهنما

دکتر محمود امین‌طوسی

استاد مشاور

دکتر مهدی زعفرانیه

پژوهشگر:

سکینه‌سادات قزی

آذر ۱۳۹۶

باسمه تعالی

فرم ارزشیابی و صورتجلسه دفاع از پایان‌نامه کارشناسی ارشد

فرم ۱۱۳-ت

جلسه دفاع از پایان‌نامه آقای /خانم **سکینه‌سادات قزی** دانشجوی رشته ریاضی کاربردی گرایش تحقیق در عملیات به شماره دانشجویی ۹۴۱۳۱۳۳۰۴۸ با عنوان:

حل مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود با الگوریتم خفاش

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

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

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

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

مشخصات هیات دوران

ردیف	نام و نام خانوادگی	سمت	مرتبه علمی	محل کار	امضـــــــــــا
۱	دکتر محمود امین‌طوسی	استاد راهنما	استادیار	دانشگاه حکیم سبزواری	
۲	دکتر مهدی زعفرانیه	استاد مشاور	استادیار	دانشگاه حکیم سبزواری	
۳	دکتر امید باغانی	استاد داور	استادیار	دانشگاه حکیم سبزواری	
۴	دکتر مهدی مهدی‌زاده	نماینده تحصیلات تکمیلی	استادیار	دانشگاه حکیم سبزواری	

امضا

مدیر گروه

امضا

رئیس دانشکده

سوگند نامه دانش آموختگان دانشگاه حکیم سبزواری

به نام خداوند جان و خرد کزین برتر اندیشه بر نگذرد

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

نام و نام خانوادگی: سکینه‌سادات قزی

تاریخ و امضا:

تأییدیه‌ی صحت و اصالت نتایج

باسمه تعالی

اینجانب سکینه‌سادات قزی به شماره دانشجویی ۹۴۱۳۱۳۳۰۴۸ دانشجوی رشته ریاضی کاربردی مقطع تحصیلی کارشناسی ارشد تأیید می‌نمایم که کلیه‌ی نتایج این پایان‌نامه حاصل کار اینجانب و بدون هرگونه دخل و تصرف است و موارد نسخه برداری شده از آثار دیگران را با ذکر کامل مشخصات منبع ذکر کرده‌ام. درصورت اثبات خلاف مندرجات فوق، به تشخیص دانشگاه مطابق با ضوابط و مقررات حاکم (قانون حمایت از حقوق مؤلفان و مصنفان و قانون ترجمه و تکثیر کتب و نشریات و آثار صوتی، ضوابط و مقررات آموزشی، پژوهشی و انضباطی ...) با اینجانب رفتار خواهد شد و حق هرگونه اعتراض درخصوص احقاق حقوق مکتسب و تشخیص و تعیین تخلف و مجازات را از خویش سلب می‌نمایم. در ضمن، مسؤولیت هرگونه پاسخگویی به اشخاص اعم از حقیقی و حقوقی و مراجع ذی صلاح (اعم از اداری و قضایی) به عهده ی اینجانب خواهد بود و دانشگاه هیچ گونه مسؤولیتی در این خصوص نخواهد داشت.

نام و نام خانوادگی: سکینه‌سادات قزی

تاریخ و امضا:

مجوز بهره برداری از پایان‌نامه

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

☐ بهره برداری از این پایان‌نامه برای همگان بلامانع است.

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

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

استاد راهنما: دکتر محمود امین‌طوسی

تاریخ و امضا:

تقدیم به:

همسر و فرزندانم

و

پدر و مادرم

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

در آغاز وظیفه خود می دانم از زحمات بی دریغ استاد راهنمای خود، جناب آقای دکتر محمود امین‌طوسی، صمیمانه تشکر و قدردانی کنم که قطعاً بدون راهنمایی های ارزنده ایشان، این مجموعه به انجام نمی رسید.

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

همچنین لازم می دانم از گروه پارسی‌لاتک در پاسخگویی به مشکلات کاربران کمال قدردانی را داشته باشم.

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

سکینه‌سادات قزی

آذر ۱۳۹۶

فهرست مطالب

فهرست جداول

فهرست تصاویر

فهرست الگوریتم‌ها

فرم چکیده ی پایان‌نامه ی دوره ی تحصیلات تکمیلی

مدیریت تحصیلات تکمیلی

دانشگاه حکیم سبزواری

نام خانوادگی دانشجو:قزی	نام:سکینه‌سادات	ش. دانشجویی:۹۴۱۳۱۳۳۰۴۸
استاد راهنما: دکتر محمود امین‌طوسی		
استاد مشاور: دکتر مهدی زعفرانیه		
دانشکده ریاضی و علوم کامپیوتر	رشته: ریاضی کاربردی	گرایش: تحقیق در عملیات
مقطع: کارشناسی ارشد	تاریخ دفاع:آذر ۱۳۹۶	تعداد صفحات: ۸۳
عنوان پایان‌نامه: حل مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود با الگوریتم خفاش		
کلید واژه‌ها:مکان‌یابی، تسهیلات، ظرفیت نامحدود، فراابتکاری، الگوریتم خفاش.		

چکیده: مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود یکی از مسائل اساسی و بنیادی بهینه‌سازی است که شامل انتخاب مکان‌هایی است که باید در آن تسهیلات عرضه‌کننده‌ی خدمات مشابه قرار داده شوند. منظور از کمینه‌کردن در این مسائل، باز کردن زیر مجموعه ای از تسهیلات از موقعیت های بالقوه و تخصیص هر مشتری به یک تسهیل باز است به طوری که مجموعه هزینه خدمات رسانی و هزینه ثابت به حداقل برسد. مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود یک مسأله از نوع صفر و یک است که در ادبیات موضوع با استفاده از روش‌های مختلف مورد بررسی قرار گرفته است. از آن جایی که نشان داده شده است که این‌گونه مسائل جزء مسائل NP–سخت هستند، به طور کلی تصور می‌شود که امیدی به پیدا کردن یک الگوریتم قابل حل در زمان چندجمله‌ای که همیشه جواب بهینه را بدست آورد نیست. در این پایان نامه سه روش حل مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود بر اساس سه نسخه صفر و یک از یک روش هوش جمعی جدید به نام الگوریتم خفاش نشان داده می‌شود. الگوریتم خفاش یکی از الگوریتم‌های فراابتکاری است که اخیراً بر اساس الهام گرفتن از رفتار پژواک‌یابی خفاش‌های کوچک در طبیعت مطرح و جهت حل مسائل بهینه‌سازی پیوسته به‌کار گرفته شده است. عملکرد برتر این الگوریتم در حل مسائل پیوسته در میان شناخته شده‌ترین الگوریتم‌ها مانند الگوریتم ژنتیک و بهینه‌سازی ازدحام ذرات به اثبات رسیده است. بنابراین در این پایان‌نامه مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود در نظر گرفته شده است. الگوریتم‌های صفر و یک خفاش بر روی ۹ مسأله‌ی استاندارد گرفته شده از OR-library، تست شده و عملکرد آن‌ها با الگوریتم‌های ژنتیک و بهینه‌سازی ازدحام ذرات مقایسه شده است.

با توجه به نتایج تجربی، الگوریتم‌های صفر و یک خفاش نتایج موفقیت آمیزی در حل مسائل مکان‌یابی تسهیلات با ظرفیت نامحدود از لحاظ کیفیت جواب‌ها و سرعت محاسبات بدست می‌آورند.

پیش‌گفتار

انتخاب مکان مناسب برای تسهیلات یکی از تصمیمات استراتژیک در کسب و کار محسوب می‌شود زیرا با مکان‌یابی و تأسیس تسهیلات جدید، هزینه بسیار بالایی به کسب و کار تحمیل می‌شود. علاوه بر این عدم استفاده از تسهیلات موجود و تلاش برای مکان‌یابی تسهیلات جدید ممکن است به دلیل هزینه بالا و یا پیچیدگی آن، امری نشدنی باشد؛ از این رو تصمیم‌گیری در مورد مکان تسهیلات فرایندی نیست که تنها از روش سعی و خطا قابل انجام باشد. لذا باید توجه ویژه‌ای به این‌گونه مسائل داشت. اغلب مسائل مکان‌یابی از نوع مسائل بهینه‌سازی ترکیبیاتی بوده و در کلاس مسائل NP-سخت قرار می‌گیرند. بنابراین استفاده از روش‌های حل دقیق، بخصوص در مسائلی با ابعاد بزرگ، چندان موثر نبوده و اغلب از الگوریتم‌های فراابتکاری و تقریبی به عنوان روش‌های حل اصلی این‌گونه مسائل استفاده می‌شود.

امروزه زمینه‌ی تحقیقاتی پیرامون الگوریتم‌های فراابتکاری هم در بحث توسعه و ایجاد روش‌های نوین و هم در مبحث کاربرد بسیار فعال و پویا بوده و هر روزه بر شمار کاربران این روش‌ها افزوده می‌شود.

در تحقیق پیش‌رو، یکی از مسائل پایه‌ای مکان‌یابی، یعنی مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود در نظر گرفته شده است که از نوع مسائل صفر و یک محض است. اهمیت یافتن روش‌های حل جدید برای بهینه‌سازی مسائل مکان‌یابی تسهیلات با ظرفیت نامحدود از این رو است که علاوه بر حل این‌گونه مسائل، با صرف اندکی هزینه می‌توان از آن‌ها برای بهینه‌سازی دیگر مسائل صفر و یک محض استفاده کرد.

بر اساس این خصیصه در این پایان‌نامه از روش‌های فراابتکاری صفر و یک مبتنی بر الگوریتم فراابتکاری خفاش برای حل مسائل مکان‌یابی تسهیلات با ظرفیت نامحدود بهره خواهیم برد. ۹ نمونه از مسائل مکان‌یابی تسهیلات با ظرفیت نامحدود موجود در سایت اینترنتی OR-Library جهت توابع تست برگزیده شده‌اند و نتایج بدست آمده از حل این نمونه‌ها توسط روش‌های معرفی شده با برخی روش‌های موجود در ادبیات موضوع مقایسه خواهد شد.

فصل ۱

علم مکان‌یابی

۱-۱ مقدمه و تاریخچه

بیشینه علم مکان‌یابی به قرن هفدهم میلادی، و به مسأله فرما برمی‌گردد، که در آن با فرض مشخص بودن مکان سه نقطه در صفحه، می‌بایست نقطه چهارم را به گونه ای یافت که مجموع فاصله‌های آن نقطه تا سه نقطه مفروض کمینه شود. این مسأله که بر اساس نام ریاضیدان فرانسوی پیر دو فرما [۲] (۱۶۰۱-۱۶۶۵) نام گذاری شده است در مقالات معتبری در هر دو نوع کمینه و بیشینه ارائه شده است. [۳]

در دهه‌های گذشته علم مکان‌یابی به حوزه پژوهشی بسیار فعالی تبدیل شده است که توجه بسیاری از محققان و پژوهشگران را به خود جلب نموده به طوری که در طول زمان و در راستای حل مسائل کاربردی مرتبط با سایر رشته‌ها این دانش تکامل یافته است. شرایط متفاوت موجود در مسائل مطرح شده در حوزه‌ی مکان‌یابی، انواع خاصی از مسائل را پدید آورده؛ مسائلی چون مکان‌یابی تدافعی[۳] ، مکان‌یابی تسهیلات[۴] ، مکان‌یابی رقابتی[۵] ، مکان‌یابی هاب[۶] و بسیاری دیگر که مجال بیان آن‌ها در اینجا نیست. مکان‌یابی تسهیلات در هسته مرکزی این رشته قرار دارد و شامل انتخاب "بهترین" محل جهت قرار دادن یک یا چند امکانات و یا تجهیزات به منظور سرویس‌دهی به مجموعه‌ای از مشتریانی است که در مناطق مختلف حضور دارند. معنای "بهترین" بستگی به ماهیت مسأله مورد مطالعه دارد یعنی به محدودیت‌ها و معیار بهینگی در نظر گرفته شده در مسأله، وابسته است. کاربرد مسائل مکان‌یابی تسهیلات در رشته‌های مختلف از جمله اقتصاد، جغرافیا، علوم منطقه‌ای و لجستیک[۷] (آماد) و ... می‌باشد.

در این‌گونه مسائل تعداد معینی مصرف‌کننده با تقاضای مشخص به همراه تعداد معینی مکان که به صورت بالقوه قابلیت سرویس‌دهی به مصرف‌کننده را دارند، حضور دارند. بنابراین دو نوع تصمیم باید گرفته شود؛ یک تصمیم مکان‌یابی که تعیین می‌کند کدام مکان جهت سرویس‌دهی فعال شود. دو، تصمیم تخصیص که مشخص‌کننده این موضوع است که نیاز هر

---

[۲] Fermat [۳] Defensive Location Problem [۴] Facility Location Problem

[۵] Competitive Location Problem [۶] Hub Location Problem [۷] Logistics

مصرف‌کننده توسط کدامین محل فعال برطرف شود. هر جواب شدنی برای این مسائل هزینه‌ی ثابت جهت فعال کردن مکان‌های سرویس دهنده و همچنین هزینه واگذاری متقاضیان به این مکان‌ها را به ما تحمیل می‌کند. در مسائل مکان‌یابی تسهیلات، هدف اصلی گرفتن بهینه‌ترین تصمیمات در ارتباط با هزینه‌های مطرح شده است.

توسعه علم مکان‌یابی از اوایل قرن بیستم در پی منتشر شدن کتاب آلفرد وبر[۱] در سال ۱۹۰۹ تحت عنوان "درباره مکان صنایع[۲]" [۴] آغاز شد. وبر در این کتاب نتایج تحقیقات خود در مورد صنایع کارخانه‌ای را ارائه کرد. در واقع می‌توان گفت که اولین تعریف مسأله مکان‌یابی به صورت کاربردی توسط وبر ارائه شد. نظریه وبر دارای سه اصل اساسی است. اول این‌که محل مواد اولیه معلوم است. دوم این‌که مکان و میزان مصرف مشتریان مشخص است. تعداد این مکان‌ها متناهی است و جدا از هم هستند. هر تولید کننده بازار نامحدودی در اختیار دارد و امتیازات انحصاری مکان وجود ندارد. سوم این‌که چندین مکان ثابت برای عرضه نیروی کار وجود دارد به طوری که این نیروی کار دارای قابلیت تحرک و جابجایی نبوده و عرضه آن نامحدود است.

در سال‌های ۱۹۶۳ و ۱۹۶۴ افرادی همچون استول استایمر[۳] [۵] ، کوهن و هامبورگ[۴] [۶] ، و من[۵] [۷] یک نوع خاص از این مسأله را با عنوان مسأله متریک مکان‌یابی تسهیلات با ظرفیت نامحدود[۶] معرفی نمودند. در سال ۱۹۸۳ نمهاوزر و ولسی[۷] [۸] نشان دادند که مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود یک مسأله NP- سخت است. نتایج آن‌ها سبب شد که این مسائل به خوبی گسترش پیدا کند. سلطان و فیزان[۸] [۹] در سال ۱۹۹۹ با بررسی مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود آن را با استفاده از روش فراابتکاری "جستجوی ممنوعه[۹] " حل نمودند. برخی از مدل‌های توسعه یافته این مسأله توسط هلمبرگ[۱۰] و همکاران [۱۰] با استفاده از روش آزادسازی لاگرانژ[۱۱] حل شد. همچنین روش‌های الگوریتم تجزیه برای حل این‌گونه مسائل توسعه یافته و روش‌های اولیه و اولیه _ دوگان برای حل مسائل مکان‌یابی با ظرفیت نامحدود مورد استفاده قرار گرفته است. در سال ۲۰۰۴ آریا[۱۲] و همکاران [۱۱] روش فراابتکاری جستجوی محلی[۱۳] را که توسط کاروپُلو، فلکستون و راجارامان[۱۴] [۱۲] در سال ۲۰۰۰ مورد استفاده قرار گرفته بود، توسعه دادند که در پی آن توانستند جواب بهینه را با دقت بیشتری بدست آوردند. برخی از مطالعات دیگر مانند استفاده از الگوریتم ژنتیک[۱۵] و الگوریتم بهینه سازی ازدحام ذرات[۱۶] برای حل مسائل مکان‌یابی با تسهیلات نامحدود در دهه گذشته ارائه شده است که توانسته‌اند جواب بهینه را در زمان قابل قبولی بدست آورند .

در ایران نیز روش‌های فراابتکاری همچون جستجوی ممنوعه، ازدحام ذرات و الگوریتم ژنتیک برای حل این مسائل مورد استفاده قرار گرفته است.

در ادامه این فصل به تعریف کامل‌تری از مکان‌یابی خواهیم پرداخت و با فرمول‌بندی مسائل مکان‌یابی و انواع آن‌ها آشنا خواهیم شد، قبل از بیان این موارد چند تعریف مورد نیاز را خواهیم دید. بیش‌تر این مفاهیم از مراجع [۱۳] و [۱۴] گردآوردی شده‌اند.

---

[۱] Alfred Weber [۲] Uber den standort der industrien [۳] Stollsteimer [۴] Kuehn and Hamburge
[۵] Manne [۶] Metric Uncapacitated Facility Location Problem [۷] Nemahauser and Wolsey
[۸] Sultan and Fazan [۹] Tabu search [۱۰] Holmberg [۱۱] Lagrangean relaxation [۱۲] Arya
[۱۳] Local search [۱۴] Korupolu,Plaxton and Rajaraman [۱۵] Genetic algorithm [۱۶] Particle swarm optimization

۱-۲ مفاهیم اولیه

تعریف ۱-۲-۱. مسأله بهینه‌سازی

به مسأله‌ای گفته می‌شود که به صورت زیر نمایش داده می‌شود

$$min\{f(x) \mid x \in X, X \subseteq S\}$$

به طوری که در آن S، X، x و f به ترتیب نشان دهنده‌ی فضای جواب، مجموعه شدنی، یک جواب شدنی و تابع هدف با مقادیر حقیقی می‌باشند. اگر S یک مجموعه گسسته یا قابل تبدیل به گسسته و متناهی اما با اندازه بزرگ باشد، به یک مسأله بهینه‌سازی ترکیبیاتی می‌رسیم. اگر $S = \mathbb{R}^n$ ، به بهینه سازی پیوسته خواهیم رسید. ما در این مسائل به دنبال یافتن جوابی با بهترین مقادیر ممکن که مطابق با شرایط مسأله باشد هستیم. برای مثال می‌توان مسأله فروشنده دوره‌گرد [۱] را نام برد که یک مسأله ترکیبیاتی از نوع جایگشتی است.

تعریف ۱-۲-۲. فضای جواب

به مجموعه جواب‌های ممکن مسأله، فضای جواب گفته می‌شود.

تعریف ۱-۲-۳. تابع هدف

یک مسأله‌ی بهینه‌سازی دارای یک فضای جواب یا مجموعه دامنه S و تابع مقدار به صورت $f : S \to R$ می‌باشد. $f(x) \in R$ مشخص کننده‌ی مقدار سود یا هزینه‌ای است که توسط $x \in S$ مشخص می‌شود و به آن تابع هدف، تابع هزینه یا تابع انرژی گفته می‌شود.

تعریف ۱-۲-۴. جواب شدنی [۲]

در مسأله بهینه‌سازی، هر عضو فضای جواب، یک جواب شدنی یا نقطه شدنی برای مسأله نامیده می‌شود؛ هرگاه در تمامی قیود مسأله صدق کند.

تعریف ۱-۲-۵. مجموعه شدنی

مجموعه تمامی جواب‌ها یا نقاط شدنی به ناحیه شدنی [۳] یا مجموعه شدنی موسوم می‌باشد.

تعریف ۱-۲-۶. مسأله تصمیم

در این گونه مسائل به دنبال این هستیم که مشخص کنیم یک دستور درست است یا نادرست. مسأله تصمیم در واقع دارای جواب بله یا خیر می‌باشد.

تعریف ۱-۲-۷. مرکز [۴]

نقطه‌ای روی شبکه که فاصله‌اش تا دورترین نقطه شبکه، کمینه باشد.

[۱] Traveling Salesman Problem [۲] Feasible solution [۳] Feasible Region [۴] Center

تعریف ۱-۲-۸. میانه[۱]

نقطه‌ای روی شبکه که مجموع فواصل آن با تمام نقاط شبکه، کمینه باشد.

تعریف ۱-۲-۹. نماد $\mathcal{O}$ بزرگ

در ریاضیات علامت $\mathcal{O}$ بزرگ رفتار حدی یک تابع را وقتی ورودی‌های آن به یک عدد خاص یا به بی‌نهایت میل می‌کند، توصیف می‌کند. علامت $\mathcal{O}$ بزرگ امکان ساده کردن تابع را فراهم می‌کند تا تمرکز بر روی نرخ رشد تابع افزایش یابد؛ بنابراین توابع مختلف با نرخ رشد یکسان می‌توانند دارای یک علامت $\mathcal{O}$ مشابه باشند.

مثال ۱-۲-۱۰. اگر $T(n)$ ، زمان لازم برای حل مسأله‌ای با n ورودی برابر باشد با:

$$T(n) = ۴n^۲ - ۵n + ۷$$

آن‌گاه هر زمان تعداد ورودی این مسأله به بی‌نهایت میل کند اندازه جمله $n^۲$ بسیار بزرگتر از دیگر جمله‌ها خواهد بود. در این صورت گفته می‌شود:

$$T(n) \in \mathcal{O}(n^۲)$$

$$T(n) = \mathcal{O}(n^۲)$$

به عبارتی اگر تعداد ورودی مسأله دو برابر شود زمان حل (با فرض زیاد بودن ورودی) چهار برابر خواهد شد.

تعریف ۱-۲-۱۱. پیچیدگی زمانی [۲]

پیچیدگی زمانی یک مسأله تعداد گام‌های مورد نیاز برای حل مسأله به عنوان تابعی از اندازه ورودی‌های الگوریتم می‌باشد. برای درک بهتر مفهوم پیچیدگی زمانی فرض کنید مسأله‌ای با n ورودی دارید که در $n^۲$ گام حل می‌شود. در این صورت می‌گوییم مسأله با پیچیدگی $n^۲$ است و با $\mathcal{O}(n^۲)$ نمایش داده می‌شود. تعداد دقیق گام‌های لازم برای حل مسأله به زبان برنامه نویسی و سیستم بستگی دارد؛ این موضوع محاسبه آن را دچار مشکل می‌نماید. برای حل این مشکل از نماد $\mathcal{O}$ استفاده می‌شود. در این صورت برای محاسبه‌ی پیچیدگی زمانی تنها عملیات کلیدی و پرتکرار را مورد توجه قرار می‌دهند و از مقادیر و ضرایب ثابت تابع صرف نظر می‌نمایند.

- کلاس‌های پیچیدگی

نظریه پیچیدگی زمانی به نوعی مشخص کننده میزان سختی در حل یک مسأله خواهد بود. برای سادگی کار مسائل به کلاس‌هایی تقسیم می‌شوند به طوری که مسائل یک کلاس از حیث زمان یا فضای مورد نیاز به هم مشابهت دارند. این کلاس‌ها در اصطلاح کلاس‌های پیچیدگی نامیده می‌شوند. معروف‌ترین کلاس‌های پیچیدگی، کلاس‌های P [۳] و NP [۴] هستند.

---

[۱] Median [۲] Time complexity [۳] Polynomial [۴] Non-deterministic polynomial

تعریف ۱-۲-۱۲. کلاس P

به طور شهودی می‌توان گفت، کلاس P شامل مسائلی است که الگوریتم‌های سریع برای پیدا کردن جواب قطعی آن‌ها وجود دارد به عبارت دیگر تمامی مسائل تصمیم که دارای جوابی از مرتبه چند جمله‌ای باشند در این کلاس قرار دارند. تعریف دیگری که از مسائل کلاس P ارائه می‌شود این است که هر گاه الگوریتمی وجود داشته باشد که قادر باشد به ازای هر ورودی به طول n حداکثر در cn^k مرحله (k و c اعداد ثابت مستقل از n) جواب درست بدهد آن‌گاه گوییم مسأله می‌تواند در زمان چندجمله‌ای حل شود و آن را در کلاس P قرار می‌دهیم و پیچیدگی زمانی آن را با $\mathcal{O}(n^k)$ نمایش می‌دهیم.

تعریف ۱-۲-۱۳. کلاس NP

کلاس NP شامل آن دسته از مسائلی است که در مورد آن‌ها نمی‌توان پاسخ قطعی داد که الگوریتم برای حل آن‌ها در زمان چندجمله‌ای وجود دارد یا خیر. اگر چه ممکن است پیدا کردن جواب درست قطعی برای آن ها نیاز به زمان زیادی داشته باشد اما بررسی کردن جواب به وسیله یک الگوریتم ممکن است. به عبارت دیگر اگر مسائلی باشند که بتوان با روشی برای آن‌ها جواب‌هایی حدس زد و در زمان نزدیک به زمان خطی و یا در زمان چند جمله‌ای صحت درستی جواب‌ها را بررسی کرد آن مسائل NP خواهند بود.

در حالت کلی تمامی مسائلی که جواب آن‌ها با الگوریتمی در زمان چند جمله‌ای قابل بررسی می‌باشد مسائل NP هستند. لذا هر مسأله P یک مسأله NP خواهد بود. هیچ کس نتوانسته است الگوریتمی با مرتبه زمانی چندجمله‌ای برای مسائل این گروه ارائه کند و این نکته که زمان اجرای چنین الگوریتم‌هایی، به سرعت با افزایش کم اندازه مسأله، زیاد می‌شود، بسیارحائز اهمیت است. البته ما نمی‌توانیم بگوییم که الگوریتم‌هایی با این پیچیدگی وجود ندارند، در عوض ما به دنبال آن است که نشان دهیم بسیاری از مسائلی که هیچ الگوریتم شناخته شده‌ای با مرتبه‌ی زمانی چندجمله‌ای ندارند، از نظر محاسباتی به هم مربوطند. در حقیقت مسائل NP به چندین دسته از جمله NP-کامل [۱] و NP-سخت [۲] تقسیم می‌شوند. یک مسأله NP-کامل دارای این خاصیت است که در زمانی چندجمله‌ای قابل حل خواهد بود اگر و فقط اگر تمام دیگر مسائل NP-کامل در زمانی چندجمله‌ای قابل حل باشند. اگر یک مسأله NP-سخت را بتوان در زمانی چندجمله‌ای حل نمود، آنگاه تمام مسائل NP-کامل در زمانی چندجمله‌ای قابل حل خواهند بود.

در نتیجه همه مسائل NP-کامل، NP-سخت هستند اما همه مسائل NP-سخت، NP-کامل نیستند [۱۵]. وقتی مسأله‌ای در کلاس NP-سخت معرفی می‌شود در واقع اینطور قلمداد می‌شود که هیچ الگوریتمی در زمان چند جمله‌ای برای حل آن‌ها وجود ندارد. به طور مثال پیدا کردن مسیر ساده‌ای (فاقد رأس و یال تکراری) بین دو رأس s و t به طول k از این نوع مسائل است. مسائل کلاس NP-کامل اگر چه راه حل‌های سختی دارند اما احتمال حضورشان در کلاس P از مسائل NP-سخت بیشتر است. مسأله‌ی فروشنده دوره‌گرد در این کلاس قرار می‌گیرد.

برای مشخص کردن این‌که آیا مسأله‌ای در این کلاس قرار دارد یا خیر روش اثباتی وجود ندارد و تنها با متناظر کردن مسأله‌ی جدید با یکی از مسائلی که قبلاً در این کلاس قرار گرفته است به گونه‌ای که راه حل یکی برای دیگری نیز کاربرد داشته باشد مسأله جدید نیز در این کلاس قرار خواهد گرفت.

---

[۱] NP-complete [۲] NP-hard

۱-۳ مکان‌یابی تسهیلات

در مسائل مکان‌یابی، یک مجموعه‌ی بالقوه از تسهیلات[۱] (مراکز خدماتی) و تعدادی مشتری[۲] وجود دارد که باید تقاضای [۳] آن‌ها توسط یک یا چند مرکز از بین تسهیلات فوق تامین شود. با توجه به سایر تسهیلات و محدودیت‌های موجود، انتخاب تسهیلات به گونه‌ای صورت می‌گیرد که هدف یا اهداف از پیش تعیین شده‌ای نیز برآورده شود. این هدف یا اهداف در یک مسأله‌ی مکان‌یابی می‌تواند هزینه‌ی احداث تسهیلات، فاصله‌ی مکانی بین تسهیلات یا فاصله‌ی زمانی بین آن‌ها و مواردی از این قبیل باشد. هدف‌ها در شناسایی و اولویت‌بندی معیارهای تصمیم‌گیری در یک مسأله مکان‌یابی، اهمیت و نقش مهمی دارند.

۱-۴ انواع مسائل مکان‌یابی تسهیلات

مسائل مکان‌یابی تسهیلات دارای تنوع زیادی هستند. از این رو برای سهولت در بیان، آن‌ها را با توجه به ویژگی‌ها، اهداف و قیود، به دسته‌های متفاوتی دسته‌بندی می‌کنند. برای روشن‌تر شدن موضوع، مسائل زیر را در نظر بگیرید:

- تعیین مکان تعدادی آمبولانس، به منظور کمینه کردن بیشترین زمان پاسخ به متقاضی؛
- تعیین مکان یک انبار در یک زنجیره‌ی تأمین [۴] ، برای کمینه کردن زمان حمل و نقل

از نظر تابع هدف، مسائل مکان‌یابی به سه دسته‌ی کمینه مجموع، کمینه بیشینه و پوشش تقسیم‌بندی می‌شوند. بنابراین از نظر تابع هدف، مساله‌ی مکان‌یابی آمبولانس یک مسأله‌ی کمینه بیشینه و مسأله‌ی تعیین مکان انبار یک مساله‌ی کمینه مجموع است. از طرف دیگر یک مسأله‌ی مکان‌یابی از نظر تعداد تسهیلات نیز قابل دسته‌بندی می‌باشد. از این منظر، این دو مسأله به ترتیب، چند تسهیلاتی و تک تسهیلاتی می‌باشند. دسته‌بندی را می‌توان از جهات دیگر هم در نظر گرفت، مثلاً با توجه به محدودیت ظرفیت (با ظرفیت محدود یا با ظرفیت نامحدود)، از نظر قطعیت (قطعیت کامل، قطعی یا همراه با ریسک و عدم قطعیت) و یا با توجه به ویژگی فضای تصمیم‌گیری، مسائل مکان‌یابی را می‌توان دسته‌بندی کرد. مسائل مکان‌یابی را با توجه به ویژگی فضای تصمیم‌گیری، به دو دسته‌ی مسائل در فضای حقیقی چند بعدی و مسائل در فضای شبکه تقسیم می‌شوند که هر یک از این دو دسته، خود به دو زیر دسته‌ی پیوسته و گسسته تقسیم می‌شوند.

۱-۴-۱ مسائل مکان‌یابی گسسته

در این نوع از مسائل مکان‌یابی، مجموعه‌ای از مکان‌های کاندید موجود است؛ که مراکز خدماتی تنها در این مکان‌ها می‌توانند احداث شوند. بنابراین، مطلوب انتخاب مکان برای یک یا چند مرکز با در نظر گرفتن سایر مراکز و محدودیت‌های موجود

---

[۱] Facilities [۲] Clinet [۳] Demand [۴] Supply chain

است به گونه‌ای که هدف ویژه‌ای بهینه شود. به عنوان مثال، هزینه‌ی کل کمینه شود یا فاصله‌ی این مرکز یا مراکز تا نقاط متقاضی بیشینه یا کمینه گردد. در این نوع مسائل، برای استقرار یا عدم استقرار تسهیلات در نقاط مورد نظر تصمیم‌گیری می‌شود. بنابراین با یک متغیر تصمیم‌گیری دودویی و در نتیجه با یک مدل برنامه‌ریزی خطی عدد صحیح مواجه هستیم.

۱-۴-۲ مسائل مکان‌یابی پیوسته

در مسائل مکان‌یابی پیوسته، فضای جواب پیوسته است؛ یعنی مراکز خدماتی در هر جایی از منطقه‌ی جواب، می‌توانند احداث شوند. مدل‌های مکان‌یابی در فضای پیوسته، مکان بهینه‌ی یک یا چند تسهیلات را در فضای دو بعدی تعیین می‌کنند. ضعف عمده‌ی این مدل آن است که ممکن است مکان بهینه‌ی به دست آمده موجه نباشد. به عنوان مثال، وسط رودخانه، دریا یا دریاچه باشد یا جایی که دولت مانع از ساخت چنین واحدی در آن محل می‌شود.

در مسائل مکان‌یابی پیوسته بر خلاف مسائل گسسته نمی توان لیست جامع و کاملی از نقاط در دسترس را ارائه داد. از این جهت مسائل پیوسته را مسائل ایجاد مکان و مسائل گسسته را مسائل انتخاب مکان نیز خوانده‌اند. یکی از ابزارهای مفید در ریاضیات در مواجه با مسائل بزرگ و پیوسته ایده گسسته سازی فضاست. یکی از دلایل استفاده از این ایده ساده‌تر شدن و کاهش فضای جواب به تعداد متناهی مجموعه و کاربردهای ویژه در بعضی حالات خاص از مسأله است. در مسائل گسسته، در حقیقت هر نقطه نماینده یک ناحیه از فضا است. قابل توجه است که در حالت پیوسته باید یک تابع برای تخمین فاصله بین نقاط انتخاب شود که نحوه انتخاب یک تابع مناسب، خود موضوع مقالات زیادی در زمینه مکان‌یابی پیوسته بوده است ولی در حالت گسسته، فاصله واقعی پیموده شده و یا از قبل محاسبه شده، مورد استفاده قرار می‌گیرد. بنابراین مسائل پیوسته به راحتی موقعیت‌های عملی و واقعی را مدل می‌کنند و به سرعت اجرا می‌شوند؛ زیرا به بانک‌های اطلاعاتی عظیم مسافت‌ها و همچنین محاسبه کوتاه‌ترین مسیر احتیاجی ندارند. این نکته قابل ذکر است که به خاطر استفاده از تابع تخمین، دقت کمتری نسبت به انواع گسسته دارند. البته انتخاب نوع فضا به نظر تصمیم گیرنده بستگی دارد. بعضی اوقات ممکن است مزایای حل مسائل در فضای پیوسته سبب شود که از دقت آن تا حدی صرف نظر شود[۱۶].

۱-۴-۳ مسائل مکان‌یابی حداقل فاصله

یکی از مسائلی که بر اساس هدف مکان‌یابی دسته‌بندی شده است مسأله حداقل فاصله یا مسأله وبر با هدف به حداقل رساندن هزینه‌های حمل و نقل می‌باشد. در این نوع مسائل موقعیت تسهیلات بایستی به گونه‌ای تعیین شود که مجموع فواصل وزنی (فاصله‌ای که در تعداد ساکنان ضرب می‌شود) طی شده، به حداقل برسد. محدودیت‌های (قیود) اعمال شده در این مسائل شامل موارد زیرند:

- فقط تعداد خاصی از تسهیلات، مکان‌یابی خواهد شد.
- هر گره تقاضا به نزدیک‌ترین مرکز خدماتی سفر خواهد کرد.

در این‌گونه مسائل محل احداث تسهیلات در جایی قرار خواهد گرفت که موقعیت میانه باشد؛ به عبارت دیگر مکان‌هایی که مرکزیت بیشترین نقاط تقاضا باشند. به علت این‌که در مسائل حداقل فاصله، تعداد تسهیلاتی که باید مکان‌یابی شوند؛ از پیش مشخص است با نام مسأله P-میانه نیز مطرح می‌شوند که در آن مقدار P، نشان دهنده تعداد تسهیلاتی است که مکان‌یابی خواهند شد.

۱-۴-۴ مسائل مکان‌یابی حداقل بیشینه فاصله

هدف مسائل حداقل بیشینه فاصله، به حداقل رساندن، بیشترین فاصله بین مشتریان و تسهیلات تخصیص یافته به آن‌ها می‌باشد. محدودیت‌های اعمال شده در این مسائل همانند مسائل حداقل فاصله شامل موارد زیرند:

- فقط تعداد خاصی از تسهیلات، مکان‌یابی خواهد شد.
- هر گره تقاضا به نزدیک‌ترین مرکز خدماتی سفر خواهد کرد.

این مسائل برای تعیین مکان تعداد مشخصی تسهیلات به منظور حداقل کردن حداکثر فاصله هر مرکز، تا نقطه تقاضایی که برای خدمت‌رسانی به آن تعیین شده است، استفاده می‌شوند؛ از این رو به آن‌ها مسائل P-مرکز نیز اطلاق می‌شود که P نشان دهنده تعداد تسهیلاتی است که باید مکان‌یابی شوند. در واقع این گونه مسائل برای استقرار خدمات اورژانس، مانند: آتش‌نشانی، خدمات آمبولانس و مراکز پلیس در جامعه مورد استفاده قرار می‌گیرند.

۱-۴-۵ مسائل مکان‌یابی تسهیلات با ظرفیت نامحدود

این مسائل در دسته مسائل حداقل مجموع قرار می‌گیرند اما در این مسائل هزینه کل، هزینه ثابت را نیز شامل می‌شود و هزینه ثابت به مکانی بستگی دارد که تسهیلات در آن قرار می‌گیرد. تعداد تسهیلاتی که باید استقرار یابند از پیش مشخص نیست اما به گونه‌ای معین می‌شوند که هزینه را کمینه کنند. در این گونه مسائل ظرفیت هر مرکز نامحدود در نظر گرفته می‌شود از این رو تخصیص یک نقطه تقاضا به بیش از یک تسهیل، هرگز سودبخش نخواهد بود.

۱-۴-۶ مسائل مکان‌یابی تسهیلات با ظرفیت محدود

این مسائل شبیه به مسائل مکان‌یابی تسهیلات با ظرفیت نامحدود هستند، فقط در این مسائل ظرفیت هر کدام از مراکز محدود است. ممکن است در این مورد جواب بهینه به گونه‌ای باشد که یک مشتری به بیش از یک تسهیل، ارجاع داده شود. در واقع ممکن است پس از تخصیص مشتری به یک مرکز، پس از برآوردن بخشی از تقاضای مشتری، ظرفیت مرکز به پایان برسد و برای برآوردن باقی‌مانده تقاضای مشتری مجبور به اختصاص آن به دیگر تسهیلات که هزینه بیشتری نیز در بر دارند، شویم.

۱-۵ مدل‌سازی مسائل مکان‌یابی تسهیلات

ساده‌ترین مسأله مکان‌یابی همان مسأله‌ای است که فرما مطرح کرد. تعمیمی از آن که به مسأله فرما–وبر معروف می‌باشد به این صورت است که فرض می‌کنیم m نقطه در صفحه موجودند؛ می‌خواهیم نقطه‌ای مانند x را به گونه‌ای بیابیم که مجموع فاصله x تا نقاط موجود مینیمم شود. یعنی اگر فاصله x تا هر نقطه را به ترتیب به صورت $d_۱, d_۲, ..., d_m$ نمایش دهیم آن‌گاه مسأله به صورت زیر خواهد بود:

$$\min \sum_{j=۱}^{m} d_j$$

مسأله فوق به مسأله تک تسهیلاتی با کمترین مجموع معروف است. همچنین اگر هدف پیدا کردن x به گونه‌ای باشد که فاصله x تا دورترین نقطه موجود کمینه شود، آن‌گاه مسأله را مسأله تک تسهیلاتی مینیماکس گویند که به صورت زیر می‌باشد:

$$\min \max_{j=۱,...,m} d_j$$

در این گونه مسائل و در فضای گسسته اگر به جای یک نقطه، هدف پیدا کردن چند نقطه باشد یعنی چند مکان برای تسهیلات جدید به گونه‌ای بیابیم که بتوان نزدیک‌ترین نقطه را به نزدیک‌ترین تسهیل نسبت داده با مسائلی از نوع P–میانه و P–مرکز روبه‌رو هستیم.

در این حالت اگر P تعداد تسهیلات جدیدی باشد که باید از میان $I = \{۱, ۲, ..., n\}$ مکان بالقوه مکان‌یابی شود و $J = \{۱, ۲, ..., m\}$ مجموعه نقاط تقاضا باشد و x_{ij} نشان دهنده تخصیص نقطه تقاضای jام به نزدیک‌ترین تسهیل (تسهیل iام) باشد به طوری که فاصله بین آن‌ها را با d_{ij} نمایش دهیم. آن‌گاه تابع هدف در مسائل P–میانه و P–مرکز به ترتیب می‌تواند به صورت زیر نوشته شود:

$$\min z = \sum_{j \in J} \sum_{i \in I} d_{ij} x_{ij}$$

و

$$\min z = \max_{j \in J} \sum_{i \in I} d_{ij} x_{ij}$$

واضح است که در حالت چند تسهیلاتی علاوه بر پیدا کردن مکان تسهیلات، مسأله تخصیص نقاط به تسهیلات جدید نیز مورد نظر است.

جهت تکمیل مدل مسأله محدودیت‌های زیر را با به تابع هدف‌های ارائه شده اضافه می‌کنیم:

$$\sum_{i\in I} x_{ij} = ۱ \qquad \forall j = ۱, ۲, ..., m \tag{۱-۱}$$

$$\sum_{i\in I} y_i = p \tag{۱-۲}$$

$$x_{ij} \le y_i \qquad \forall j = ۱, ۲, ..., m; i = ۱, ۲, ..., n \tag{۱-۳}$$

$$x_{ij} \ge ۰ \qquad \forall j = ۱, ۲, ..., m; i = ۱, ۲, ..., n \tag{۱-۴}$$

$$y_i = ۰ \vee ۱ \qquad \forall i = ۱, ۲, ..., n \tag{۱-۵}$$

محدودیت (۱-۱) تضمین می‌کند که تقاضای همه مشتریان تأمین گردد. همچنین به دلیل عدم وجود محدودیت عرضه در تسهیلات از تخصیص هر مشتری به بیش از یک تسهیل جلوگیری می‌کند. در محدودیت (۱-۲) متغیر y_i نشان دهنده مکان‌هایی است که تسهیلات می‌توانند در آن‌ها قرار گیرند. این محدودیت تضمین می‌کند دقیقا P تسهیل ایجاد شوند و این در حالی است که محدودیت (۱-۳) از تخصیص مشتری به مکان‌های فاقد تسهیلات جلوگیری می‌کند. محدودیت (۱-۵) نشان دهنده دودویی بودن متغیر y_i است به طوری که اگر در مکان iام تسهیلات ایجاد شود مقدار y_i برابر یک و در غیر این صورت برابر با صفر خواهد بود. محدودیت‌های بیان شده در هر دو مسأله P-میانه و P-مرکز یکسان است.

در مسائل مکان‌یابی تسهیلات با ظرفیت نامحدود که هزینه علاوه بر فاصله، شامل هزینه ثابت احداث تسهیلات نیز می‌شود نیازمند تعریف متغیر جدید و تغییر تابع هدف مسأله هستیم. فرض می‌کنیم f_i هزینه احداث تسهیل iام است. بنابراین تابع هدف برای مسائل مکان‌یابی تسهیلات با ظرفیت نامحدود می‌تواند به صورت زیر بیان شود:

$$\min z = \sum_{i\in I} f_i y_i + \sum_{j\in J}\sum_{i\in I} d_{ij} x_{ij}$$

همچنین با توجه به مشخص نبودن تعداد تسهیلات در این گونه مسائل تنها با حذف محدودیت (۱-۲) می‌توان سایر محدودیت‌ها را برای مسائل مکان‌یابی تسهیلات با ظرفیت نامحدود جهت تأمین شروط مسأله بکار برد.

وجود محدودیت خدمات رسانی در مسائل مکان‌یابی تسهیلات با ظرفیت محدود سبب می‌شود، محدودیت جدیدی به مدل مسأله اضافه شود. فرض می‌کنیم q_i نشان دهنده بالاترین ظرفیت خدمات رسانی تسهیل iام و s_j میزان تقاضای مصرف کننده jام می‌باشد؛ بنابراین محدودیت جدید را به صورت زیر بیان می‌کنیم :

$$\sum_{j\in J} s_j x_{ij} \le q_i y_i$$

این محدودیت دو نقش را ایفا می‌نماید؛ یک، مراقبت می‌کند که ظرفیت تسهیلات از حد خود تجاوز نکند. دو، از تخصیص یک مشتری به تسهیلات غیرفعال جلوگیری می‌کند؛ بنابراین به راحتی می‌توان محدودیت جدید را جایگزین محدودیت

(۱-۳) نمود. در مسائل مکان‌یابی تسهیلات با ظرفیت محدود به دلیل محدود بودن ظرفیت خدمات رسانی گاها ممکن است مطالبات یک مشتری از بیش از یک تسهیل تأمین شود. در این صورت با تغییر محدودیت (۱-۱) از حالت مساوی به بزرگتر مساوی علاوه بر تضمین تأمین نیاز تمام مشتریان ممنوعیت تخصیص یک مشتری به بیش از یک تسهیل را مرتفع می‌نمایم.

فصل ۲

مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود

در علم مکان‌یابی، مدلی که به طور وسیعی مورد مطالعه قرار گرفته است؛ مسأله مکان‌یابی گسسته می‌باشد که به اصطلاح "مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود" نامیده می‌شود و به عنوان کلاسیک‌ترین و مهم‌ترین نوع مکان‌یابی مورد ملاحظه است و به اختصار با UFLP نمایش داده می‌شود. همچنین مکان‌یابی تسهیلات با ظرفیت نامحدود با عناوین مکان‌یابی تسهیلات ساده[۲] (SFLP) ، مسأله مکان‌یابی کارخانه[۳] (PLP) و یا مسأله مکان‌یابی انبار[۴] (WLP) نیز شناخته می‌شود.

۲-۱ مسأله

یک مجموعه متشکل از مکان‌های تقاضا (مشتری)، یک مجموعه به عنوان مکان‌های بالقوه تسهیلات و اطلاعات مربوط به هزینه‌ها، با هدف یافتن حداقل هزینه از اجرای طرح بازگشایی تعدادی از تسهیلات و تخصیص هر مشتری به یک تسهیل باز، به ما ارائه شده است.

برای درک بهتر جزئیات، ورودی‌های مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود شامل موارد زیر است:

- مجموعه‌ی I مکان‌های بالقوه جهت احداث تسهیلات
- مجموعه‌ی J مشتریان که باید توسط تسهیلات سرویس‌دهی شوند.
- هزینه c_{ij} به ازای هر $i \in I$ و $k \in J$ ؛ این هزینه با تأمین نیاز مشتری jام توسط تسهیل باز در مکان iتحمیل می‌شود.

---

[۲] Simple Facility Location Problem [۳] Plant Location Problem [۴] Warehouse Location Problem

- هزینه ثابت نامنفی f_i به ازای هر $i \in I$ ؛ این هزینه یک بار و آن هم جهت بازگشایی تسهیل در مکان iام باید پرداخت شود.

مسأله انتخاب زیرمجموعه‌ی $Q(Q \subseteq I)$ از مکان‌ها، جهت بازگشایی تسهیلات در آن‌ها و تخصیص هر مشتری به تنها یک تسهیل به طوری که هزینه کل شامل هزینه تخصیص و هزینه بازگشایی تسهیلات، حداقل شود؛ می‌باشد. تعداد تسهیلاتی که باز خواهند شد $|Q|$، از قبل مشخص نیست بلکه جواب بهینه تعیین کننده تعداد تسهیلات باز خواهد بود. هزینه c_{ij} معمولا به چندین عامل بستگی دارد؛ مثلاً هزینه تولید هر واحد کالا یا خدمات در تسهیل iام و هزینه انتقال هر واحد کالا یا خدمات از تسهیل i به مشتری jام.

مثال ۲-۱-۱.

$$m = |J| = ۳, n = |I| = ۳$$

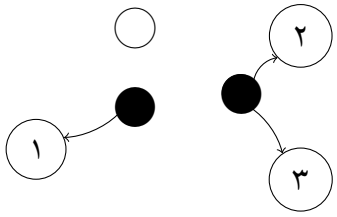
$$f_i = ۱ \qquad \forall (i = ۱, ..., n)$$

$$\mathbf{C} = \begin{bmatrix} ۳ & ۱ & ۲ \\ ۱ & ۴ & ۳ \\ ۲ & ۳ & ۴ \end{bmatrix}$$

جواب بهینه:

$$S = \{۱, ۲\}$$

با توجه به جواب بهینه تعداد ۲ تسهیل در مکان‌های یک و دو بازگشایی می‌شوند. از طرفی مشتری اول به تسهیل دو، مشتری دوم به تسهیل یک و مشتری سوم به تسهیل یک تخصیص داده خواهند شد. در پایان مقدار تابع هدف برابر با ۶ است.



شکل ۲-۱: نمایش جواب بهینه: دایره‌های مشکی تسهیلات فعال و کمان‌ها نحوه اتصال آن‌ها را به مشتریان نمایش می‌دهند

گزاره ۲-۱-۲. مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود، NP-سخت است.

اثبات: با توجه به نمونه‌ای از مسأله پوشش مجموعه که یک مسأله NP-کامل است [۱۷]؛ ما می‌توانیم نمونه‌ای از مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود بسازیم به طوری که جواب بهینه مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود، جواب بهینه‌ای برای مسأله پوشش مجموعه ارائه دهد. ابتدا یک گراف دو بخشی بر اساس نمونه پوشش مجموعه

ایجاد می‌کنیم. برای هر نقطه x_j یک گره x_j در سمت چپ گراف دو بخشی وجود دارد و برای هر مجموعه S_i، $S_i \subseteq \{x_1, ..., x_m\}$ یک گره S_i در سمت راست گراف دو بخشی موجود است. یال $x_j S_i$ وجود خواهد داشت اگر و تنها اگر نقطه x_j به مجموعه S_i تعلق داشته باشد. نمونه مسأله ULF به صورت زیر است:

مجموعه مشتری‌ها $\{x_1, ..., x_m\}$ ، مجموعه مکان‌های بالقوه $\{S_1, ..., S_n\}$ ، هزینه تخصیص c_{ij} که در صورت وجود یال $x_j S_i$ مقدار صفر می‌گیرد و در غیر این صورت مقدار ∞ ، و هزینه ثابت f_i برای هر $i = 1, ..., n$ برابر با ۱ خواهد بود. بنابراین مسأله شامل بازگشایی کمترین تعداد از تسهیلات به طوری که هر مشتری (نقطه) x_j، بتواند به یک تسهیل (مجموعه) S_i مجاور، که شامل x_j است تخصیص داده شود. به راحتی دیده می‌شود که یک جواب برای مسأله ULF (مجموعه تسهیلاتی که بازگشایی می‌شوند) بهینه است اگر و تنها اگر جواب متناظر از مسأله پوشش مجموعه، بهینه باشد[۸].

۲-۲ برنامه‌های کاربردی

مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود در طراحی برنامه‌های کاربردی فراوانی استفاده می‌شود. برخی از این برنامه‌ها عبارتند از تخصیص حساب بانکی[۱]، تجزیه و تحلیل خوشه‌ای[۲]، برنامه‌ریزی ماشین و مدیریت موجودی[۳]، محل صندوق پرداختی[۴]، محل اسکان سیستم‌های حفاری دریایی[۵] و طراحی شبکه‌های ارتباطی[۶]. در این‌جا ما تنها به شرح دو مورد اول اکتفا می‌کنیم.

مسأله تخصیص حساب بانکی از این حقیقت ناشی می‌شود که مدت زمان نقل و انتقال(یا تسویه) برای یک چک به شهر بانک i که در آن کشیده شده است و همچنین شهر بانک j که قرار است آن را پرداخت کند؛ بستگی دارد. برای یک شرکت که صورت حساب‌های خود را توسط چک به کارفرمایانی در شهرهای مختلف می‌پردازد؛ بسیار سودمند است که با برنامه‌ریزی دقیق چندین حساب بانکی در شهرهای مختلف ایجاد کند. در این صورت، صورت حساب کارفرمای ساکن در شهر j را از حساب بانکی در شهر i که دارای بیشترین مدت زمان تسویه حساب است؛ پرداخت می‌کند. در این‌جا J مجموعه کارفرمایانی است که شرکت با آن‌ها کار می‌کند و I مجموعه حساب‌های بانکی بالقوه هستند که می‌توانند در بانک‌های مختلف باز شوند، f_i هزینه نگهداری یا بازگشایی یک حساب در شهر i و c_{ij} ارزش پولی، مدت زمان تسویه حساب بین شهرهای i و j که نصیب شرکت می‌شود. واضح است که در این نوع مسأله هدف ماکزیمم کردن تابع هدف است.

در مسأله تجزیه و تحلیل خوشه‌ای، مجموعه J، شامل اهداف (اشیاء یا موضوع) به ما ارائه می‌شود و از ما خواسته می‌شود که آن‌ها را در چندین خوشه دسته‌بندی کنیم به طوری که اشیاء هر خوشه به هم شبیه باشند. در این‌جا I، زیرمجموعه‌ای از J و شامل نمایندگان خوشه‌های بالقوه است. میزان شباهت شیء j به i با c_{ij} محاسبه می‌شود. هزینه f_i هم ممکن است،

---

[۱] Bank account allocation [۲] Clustering analysis [۳] Machine scheduling and inventory management
[۴] Lock-box location [۵] Llocation of offshore drilling platforms [۶] Design of communication networks

صفر باشد؛ در این صورت خوشه‌بندی صرفاً بر اساس بیشترین شباهت بین اشیاء انجام می‌گیرد و هم ممکن است نامنفی باشد؛ در این صورت تعداد خوشه‌ها باید حداقل شود.

۲-۳ روش‌های حل مسأله

با توجه به این‌که ظاهراً برای مسائلی که ما معتقدیم در کلاس P قرار نمی‌گیرند، راه حل قطعی کارآمدی وجود ندارد بهتر است هدف ما از بررسی روش‌های حل این‌گونه مسائل چیزی جز جواب‌های قطعی و دقیق این مسائل باشد. سه روش استاندار عبارتند از:

- استفاده از ساختار خاصی از مسأله[۱] : شاید لازم نباشد که ما یک مورد کلی از یک مسأله را حل کنیم بلکه نیازمند حل یک نمونه خاص قابل انعطاف از آن باشیم.
- روش‌های تقریبی[۲] : روش‌هایی هستند که ثابت شده است، جواب‌هایی ارائه می‌دهند که مضربی از جواب بهینه هستند.
- روش‌های ابتکاری[۳] و فراابتکاری[۴] : روش‌هایی که برآورد و جواب‌های معقولی را ارائه می‌دهند اما تضمینی برای رسیدن به جواب مناسب ندارند.

۲-۳-۱ روش‌های تقریبی

این روش‌ها، الگوریتم‌های تقریبی، مسلماً از دید ریاضی رضایت بخش‌تر هستند. یک الگوریتم، یک الگوریتم تقریبی با ضریب α برای یک مسأله به شمار می‌آید اگر و تنها اگر بتواند برای هر نمونه از آن مسأله یک جواب α برابر جواب بهینه پیدا کند.

اگر مسأله در دست بررسی از نوع کمینه باشد آن‌گاه $\alpha > ۱$ است، به این معنی که جواب پیدا شده به وسیله الگوریتم حداکثر α برابر جواب بهینه است. اگر مسأله از نوع ماکزیمم سازی باشد، $\alpha < ۱$ خواهد بود و این بدین معنی است که تضمین شده جواب الگوریتم تقریبی، حداقل α برابر جواب بهینه باشد[۱۸].

نماد OPT(I) در تعریف ارزش یک جواب بهینه‌ی مسأله در دست بررسی با I ورودی مورد استفاده قرار می‌گیرد. برای مثال زمانی که ما مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود را بررسی می‌کنیم؛ $OPT(|C| + n)$ به معنای مجموعه تسهیلات باز است که کمترین هزینه بازگشایی و تخصیص به تمامی مشتریان را با توجه به ورودی‌های مسأله دارد. البته در مواردی که نیاز به توضیح بیشتر نباشد و ایجاد ابهام نکند می‌توان تنها از نماد OPT به جای OPT(I) استفاد کرد.

---

[۱] Exploiting special problem structure [۲] Approximation algorithms [۳] Heuristics [۴] Meta-Heuristics

اهمیت استفاده از کران پایین : ممکن است باور این مسأله سخت باشد که بتوان اثبات کرد الگوریتمی، یک الگوریتم تقریبی با تقریب α است! به راستی چگونه می‌توان اثبات کرد که یک الگوریتم همیشه جواب‌هایی ارائه می‌دهد که α برابر OPT است آن هم در حالی که مقدار بهینه مسأله (OPT) شناخته نشده است؟ اگرچه در مسائلی قطعاً برای ما مقدار OPT ناشناخته است اما واضح است که ما می‌توانیم در اغلب این موارد از کران پایین (یا در مسائل از نوع ماکزیمم سازی از یک کران بالا) به عنوان OPT بهره بگیریم. به عبارتی اگر ما بتوانیم نشان دهیم که الگوریتم ما همیشه یک جواب با ارزش حداکثر α برابر کران پایین مسأله تولید می‌کند آن‌گاه جواب‌های الگوریتم مضرب α از OPT نیز هست. بنابراین پیدا کردن یک کران پایین مناسب برای OPT یک گام مهم در آنالیز یک الگوریتم تقریبی است. در واقع، جستجو برای کران پایین مناسب اغلب منجر به ایده‌هایی در مورد چگونگی طراحی یک الگوریتم تقریبی خوب می‌شود. تکنیک‌های بسیاری برای طراحی الگوریتم‌های تقریبی حل مسأله مکان‌یابی تسهیلات مورد استفاده قرار گرفته‌اند. برخی از آن‌ها شامل گردکردن برنامه خطی[۱]، رویکرد اولیه-دوگان[۲]، رویکرد حریصانه[۳] و جستجوی محلی می‌باشد. هر کدام از این تکنیک‌ها مزایا و معایب خاص خود را دارند. اولین الگوریتم تقریبی برای مسائل مکان‌یابی تسهیلات توسط شمایز [۴]، تردار[۵] و آردل[۶] با استفاده از رویکرد گردگردن برنامه خطی طراحی شد. این رویکرد شامل حل برنامه خطی است که از تضعیف فرمول برنامه‌ریزی خطی عدد صحیح مسأله بدست آمده است از این رو عموماً بسیار وقت‌گیر است. الگوریتم‌های برگرفته از رویکرد اولیه-دوگان نسبتاً سریع‌تر هستند. در مورد الگوریتم‌هایی که با رویکرد حریصانه و جستجوی محلی هستند، به طور کلی می‌توان گفت پیاده سازی آن‌ها ساده است اما آنالیز آن‌ها سخت است. در جدول ۲-۱ الگوریتم‌هایی که با استفاده از این رویکردها جهت حل مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود طراحی شده با توجه به ضریب تقریبی[۷] و پیچیدگی زمانی اجرای الگوریتم آورده شده است [۱۹].

جدول ۲-۱: الگوریتم‌های تقریبی مسأله UFL

ضریب تقریب	طراح	زمان اجرا-رویکرد
$\mathcal{O}(\ln n_c)$	Hochbum	$\mathcal{O}(n^3)-$ رویکرد حریصانه
۳.۱۶	Shmoys et al.	گردکردن برنامه خطی
۲.۴۱	Guha and Khuller	گردکردن برنامه خطی + رویکرد حریصانه بهبود یافته
۱.۷۳۶	Chudak and Shmoys	گردکردن برنامه خطی
$۵+\epsilon$	Korupolu et al.	$\mathcal{O}(n^6 \log(n/\epsilon))-$ جستجوی محلی
۳	Jain and Vaziran	$\mathcal{O}(n^2 \log n)-$ رویکرد اولیه-دوگان
۱.۸۵۳	Charikar and Guha	$\mathcal{O}(n^2)-$ رویکرد اولیه-دوگان + رویکرد حریصانه بهبودیافته
۱.۷۲۸	Charikar and Guha	گردکردن برنامه خطی+اولیه-دوگان+ حریصانه بهبودیافته
۱.۸۶۱	Mahdian et al.	$\mathcal{O}(n^2 \log n)-$ رویکرد حریصانه
۱.۶۱	Jain et al.	$\mathcal{O}(n^3)-$ رویکرد اولیه-دوگان
۱.۵۸۲	Sviridenko	گردکردن برنامه خطی
۱.۵۲	Mahdian et al.	$\tilde{\mathcal{O}}(n)-$ رویکرد حریصانه + صرفه جویی هزینه

[۱] LP-rounding [۲]Primal-dual approach [۳]Greedy approach [۴]Shmoys [۵]Tardos
[۶]Aardal [۷]Approximation factor

۲-۳-۱-۱ برنامه‌ریزی خطی

برای فرمول‌بندی مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود بر اساس برنامه‌ریزی خطی عدد صحیح برای هر مکان بالقوه‌ی $i \in I$ با هزینه ثابت بازگشایی f_i تعریف می‌کنیم

$$y_i = \begin{cases} ۱ & \text{اگر تسهیلات در مکان } i \text{ باز شود} \\ ۰ & \text{در غیر این صورت} \end{cases}$$

قطعاً مشتریان باید به نزدیک‌ترین تسهیل تخصیص داده شوند. برای هر $j \in J$ ، $\sigma(i)$ برابر با نزدیک‌ترین j است به طوری که $y_i = ۱$. تعریف می‌کنیم

$$x_{ij} = \begin{cases} ۱ & \text{اگر مشتری } j \text{ به وسیله تسهیل } i \text{ سرویس‌دهی شود} \\ ۰ & \text{در غیر این صورت} \end{cases}$$

بنابراین فرمول برنامه‌ریزی خطی عدد صحیح مسأله عبارتند از:

$$min \quad \sum_{i \in I} f_i y_i + \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \tag{۱-۲}$$

$$\text{به طوری که} \quad x_{ij} \leq y_i \qquad i \in I, j \in J$$

$$\sum_{i \in I} x_{ij} = ۱ \qquad j \in J$$

$$x_{ij} \in \{۰, ۱\} \qquad i \in I, j \in J$$

$$y_i \in \{۰, ۱\} \qquad i \in I$$

با تضعیف محدودیت‌های عدد صحیح فرمول (۲-۱)، مسأله به صورت برنامه‌ریزی خطی زیر تبدیل می‌شود:

$$min \quad \sum_{i \in I} f_i y_i + \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \tag{۲-۲}$$

$$\text{به طوری که} \quad x_{ij} \leq y_i \qquad i \in I, j \in J$$

$$\sum_{i \in I} x_{ij} = ۱ \qquad j \in J$$

$$x_{ij} \geq ۰ \qquad i \in I, j \in J$$

$$y_i \geq ۰ \qquad i \in I$$

فرمول برنامه خطی، اول بار توسط بالینسکی[1] در سال ۱۹۶۵ ارائه شد. دوگان این برنامه‌ریزی خطی به صورت زیر می‌باشد:

$$
\begin{aligned}
max \quad & \sum_{j\in J} v_j && && && (۳-۲)\\
& v_i - w_{ij} \le c_{ij} && i\in I, j\in J && \text{به طوری که}\\
& \sum_{j\in J} w_{ij} \le f_i && i\in I\\
& w_{ij} \ge ۰ && i\in I, j\in J\\
& v_j \quad \text{نامقید} && j\in J
\end{aligned}
$$

مثال ۲-۳-۱. مدل برنامه‌ریزی خطی برای مثال ۳-۲-۱ را می‌توان به صورت زیر بیان نمود:

$$minz = y_۱ + y_۲ + y_۳ + ۳x_{۱۱} + x_{۱۲} + ۲x_{۱۳} + x_{۲۱} + ۴x_{۲۲} + ۳x_{۲۳} + ۲x_{۳۱} + ۳x_{۳۲} + ۴x_{۳۳}$$

به طوری که:

$$
\begin{aligned}
-y_۱ + x_{۱۱} &\le ۰\\
-y_۱ + x_{۱۲} &\le ۰\\
-y_۱ + x_{۱۳} &\le ۰\\
-y_۲ + x_{۲۱} &\le ۰\\
-y_۲ + x_{۲۲} &\le ۰\\
-y_۲ + x_{۲۳} &\le ۰\\
-y_۳ + x_{۳۱} &\le ۰\\
-y_۳ + x_{۳۲} &\le ۰\\
-y_۳ + x_{۳۳} &\le ۰\\
x_{۱۱} + x_{۲۱} + x_{۳۱} &= ۱\\
x_{۱۲} + x_{۲۲} + x_{۳۲} &= ۱\\
x_{۱۳} + x_{۲۳} + x_{۳۳} &= ۱
\end{aligned}
$$

$$x_{ij} \ge ۰ \qquad i\in I, \qquad j\in J \qquad y_i \ge ۰ \qquad i\in I$$

[1] Balinski

که در آن w_{ij} متغیرهای دوگان برای محدودیت اول برنامه خطی و v_i متغیرهای دوگان برای محدودیت دوم آن است. الگوریتم گرد کردن برنامه‌ریزی خطی، برنامه خطی (۲-۲) را حل می‌کند و سپس با استفاده از یک روش مناسب گردکردن، جواب‌های کسری از مسأله خطی اولیه را به جواب‌های عدد صحیح برای برنامه خطی عدد صحیح (۲-۱) تبدیل می‌کند[۱].

الگوریتم ۲-۱ الگوریتم LP-rounding(شمایز، تردار و آردل)

ورودی: یک نمونه $(I, J, (f_i)_{i\in I}, (c_{ij})_{i\in I, j\in J})$ از مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود
خروجی: یک جواب $X \subseteq I$ و $\sigma : J \longrightarrow X$

۱: یک جواب بهینه (x^*, y^*) برای (۲-۲) و یک جواب بهینه برای (v^*, w^*) برای (۲-۳) بدست آورید.
۲: قرار دهید: $K := ۱$، $X := \emptyset$ و $U = J$
۳: $j_k \in U$ را به گونه ای انتخاب کنید که $v^*_{j_k}$ کمترین مقدار باشد.
$i_k \in I$ را با $x^*_{i_k j_k} > ۰$ و $\min f_{i_k}$ انتخاب کنید. قرار دهید: $X := X \cup \{i_k\}$
مجموعه $N_k := \{j \in U : \exists i \in I : x^*_{ij_k} > ۰, x^*_{ij} > ۰\}$ را ایجاد کنید.
به ازای هر $j \in N_k$ قرار دهید: $\sigma(j) := i_k$;
قرار دهید: $U := U \backslash N_k$
۴: قرار دهید: $K := K + ۱$
اگر $U \neq \emptyset$ آن‌گاه به گام ۳ بروید.

شمایز و همکاران [۲۰] در سال ۱۹۹۷ اولین ضریب ثابت تقریبی را برای این رویکرد ارائه و نشان دادند که در حل مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود الگوریتم ۲-۱، الگوریتمی با ضریب تقریب ۴ است. یک سال بعد چاداک[۱] و شمایز [۲۱] ضریب تقریبی الگوریتم را تا ۱.۷۳۶ بهبود دادند و در سال ۲۰۰۲، سوریدنکو[۲] [۲۲] ضریب تقریبی الگوریتم را تا مرز ۱.۵۸۲ کاهش داد. با این حال یک الگوریتم سریع‌تر و ساده‌تر که ضمانت اجرایی بهتری نیز دارد، وجود دارد که از الگوریتم برنامه‌ریزی خطی استفاده نمی‌کند؛ این الگوریتم از رویکرد اولیه-دوگان بهره می‌برد.

۲-۱-۳-۲ رویکرد اولیه-دوگان

جان[۳] و وزیرانی[۴] [۲۳] در سال ۲۰۰۱ یک الگوریتم تقریبی متفاوت پیشنهاد دادند؛ الگوریتمی که در شکل کلاسیک خود از محاسبه همزمان جواب‌های شدنی اولیه و دوگان برای برنامه‌های خطی (۲-۲) و (۲-۳) بهره می‌برد.

شروع کار با یک جواب شدنی دوگان و یک جواب نشدنی اولیه (در ابتدا تمام تسهیلات بسته هستند) همراه است. سپس در هر مرحله یک جواب شدنی اولیه به صورت عدد صحیح ساخته می‌شود با این رویکرد که در هر تکرار جواب اولیه بهبود یابد در نتیجه جواب نهایی به صورت عدد صحیح ارائه می‌شود.

در این الگوریتم مقادیر متغیرهای دوگان که در ابتدا صفر فرض می‌شوند، در هر تکرار تدریجاً افزایش می‌یابند تا زمانی که مشتری jام به صورت آزمایشی به یک تسهیل تخصیص داده شود در این هنگام مقدار v_j ثابت می‌شود. به دلیل این‌که

---

[۱] Chudak [۲] Sviridenko [۳] Jain [۴] Vazirani

متغیرهای w_{ij} در تابع هدف مسأله دوگان نیستند؛ می‌توانیم در هر مرحله $w_{ij} := \max\{۰, v_j - c_{ij}\}$ در نظر بگیریم.

در هر تکرار در صورت وجود شرایطی تسهیلاتی به طور آزمایشی باز و مشتریان به آن‌ها متصل می‌شوند. الگوریتم ارائه شده توسط جان و وزایرانی در حل مسائل مکان‌یابی تسهیلات با ظرفیت نامحدود دارای ضریب تقریبی ۳ و زمان اجرایی $\mathcal{O}(n \log n)$ در جایی که $n = |I||J|$ می‌باشد.

در سال ۲۰۰۲ الگوریتم اولیه-دوگان توسط جان، مهدیان[۱] و صابری[۲] [۲۴] بهبود یافت به طوری که در حل مسائل مکان‌یابی تسهیلات با ظرفیت نامحدود دارای ضریب تقریبی ۱.۶۱ و زمان اجرایی $\mathcal{O}(|I|^۲|J|)$ گردید. جهت آشنایی بیشتر با این الگوریتم و اثبات کارایی آن به منبع [۱] مراجعه نمایید.

الگوریتم ۲-۲ الگوریتم Primal-Dual[۱]

ورودی: یک نمونه $(I, J, (f_i)_{i\in I}, (c_{ij})_{i\in I, j\in J})$ از مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود
خروجی: یک جواب $X \subseteq I$ و $\sigma : J \longrightarrow X$

۱: قرار دهید: $X := \emptyset$ و $U := J$.
۲: قرار دهید: $t_۱ := \min\{c_{ij} : i \in X, J \in U\}$.
قرار دهید $t_۲$ را برابر با:
$\min\{\tau : \exists i \in I\backslash X : \sum_{j\in U} \max\{۰, \tau - c_{ij}\} + \sum_{j\in J\backslash U} \max\{۰, c_{\sigma(j)j} - c_{ij}\} = f_i\}$.
قرار دهید $t := \min\{t_۱, t_۲\}$
۳: برای هر $i \in I\backslash X$
با $\sum_{j\in U} \max\{۰, t - c_{ij}\} + \sum_{j\in J\backslash U} \max\{۰, c_{\sigma(j)j} - c_{ij}\} = f_i$ انجام دهید:
$X := X \cup \{i\}$.
برای هر $j \in J\backslash U$ با شرط $c_{ij} < c_{\sigma(j)j}$ قرار بدهید: $\sigma(j) := i$.
۴: برای هر $i \in X$ و $j \in U$ با شرط $c_{ij} \leq t$ تنظیمات زیر را انجام دهید:
$\sigma(j) := i$، $v_j := t$ و $U := U\backslash\{j\}$
۵: اگر $U \neq \emptyset$ آن‌گاه به گام ۲ بروید.

۲-۳-۱-۳ رویکردحریصانه

رویکرد حریصانه یکی از ساده‌ترین تکنیک‌هایی است که در طراحی الگوریتم‌های حل مسائل بهینه‌سازی مورد استفاده قرار گرفته است. رویکردی که روش حریصانه برای حل مسائل بهینه‌سازی دارد شامل تصمیم‌گیری‌های پشت سرهم است که برای هر تصمیم‌گیری تنها از اطلاعات بدست آمده تا آن مرحله استفاده می‌کند. بنابراین اصطلاحاً گفته می‌شود که تصمیم‌گیری بر اساس انتخاب‌هایی صورت می‌پذیرد که به صورت محلی بهینه هستند. در این رویکرد حل مسأله امیدواریم تا به جواب بهینه برسیم. اما این جواب بهینه در برخی موارد بدست نمی‌آید.

در این رویکرد برای هر الگوریتم پیشنهادی باید نشان داده شود که پاسخ همواره در تمامی موارد بهینه است. برای مسائل

[۱] Mahdian [۲] Saberi

زیادی مانند کمینه کردن درخت‌های پوشا[۱] ، کوتاه‌ترین مسیر[۲] و کدهای هافمن[۳] این تکنیک یک جواب بهینه سراسری را ارائه می‌کند. در حل مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود این روش چه به تنهایی و چه در ترکیب با دیگر روش‌ها نتایج مطلوبی ارائه داده است . با توجه به جدول (۲-۱) بهترین ضریب تقریبی که تا کنون برای حل این مسائل شناخته شده با استفاده از این رویکرد بدست آمده است.

در حل مسائل با شیوه حریصانه هر تکرار از سه بخش تشکیل شده است:

- روند انتخاب جواب
- بررسی شدنی بودن
- بررسی جواب

در این روش ابتدا یک تسهیل با کمترین هزینه بازگشایی f_i ، انتخاب و به مجموعه تهی X اضافه می‌شود سپس تمام تسهیلات از آن سرویس‌دهی می‌شوند و مقدار $C_s(X)$ که مجموع هزینه تخصیص است، محاسبه می‌گردد. روند انتخاب جواب در تکرارهای بعد به این صورت است که هر یک از تسهیلات باز نشده به ترتیب داشتن کمترین هزینه بازگشایی، انتخاب و به مجموعه X اضافه می‌شود و عمل تخصیص مشتریان دوباره صورت می‌گیرد به طوری که هر مشتری به نزدیکترین تسهیل باز تخصیص داده شود.

در بخش بررسی شدنی بودن، این سوال مطرح است که در صورت اضافه شدن تسهیل جدید به مجموعه X ، هزینه کل، $C(X \cup \{i : f_i = \min f_i, i \in I \backslash X\})$ ، کاهش می‌یابد یا خیر؟ در پاسخ به این سوال باید گفت، اگر روابط:

$$C_s(X \cup \{i\}) \leq C_s(X)$$

$$C_s(X) - C_s(X \cup \{i\}) \leq f_i$$

برقرار باشند نتیجه می‌شود که با وجود تحمیل هزینه بازگشایی از باز کردن تسهیل جدید، هزینه کل از بخش تخصیص کاهش می‌یابد. واضح است که در صورت برقرار نبودن هر یک از روابط فوق اضافه شدن تسهیل iام سود بخش نبوده یا به عبارتی باز کردن تسهیل iام شدنی نخواهد بود. این مراحل برای تمامی تسهیلات باز نشده انجام می‌پذیرد.

در بخش بررسی جواب با محاسبه $\max \frac{C_s(X)-C_s(X\cup\{i\})}{f_i}$ بهترین تسهیل جهت بازگشایی معرفی و به مجموعه X اضافه می‌شود. در پایان هر تکرار، مجموعه X به عنوان بهترین جواب حال حاضر (بهینه محلی) معرفی می‌شود. واضح است که در هر تکرار بهترین جواب در حال بهبود است و تکرارها تا زمانی که روند بهبود ادامه دارد؛ ادامه می‌یابد.

---

[۱] Spanning tree [۲] Shortest path [۳] Huffman codes

۲-۳-۱-۴ رویکرد جستجوی محلی

جستجوی محلی تکنیکی است که در اغلب موارد در عمل موفقیت‌آمیز بوده است اگر چه معمولاً نمی‌توان ضمانت تقریبی خوبی برای اینکه همیشه موفق خواهد بود نشان داد. بنابراین زمانی که این روش توانست مسأله مکان‌یابی را تقریباً خوب حل کند تعجب همگان را برانگیخت. اولین بار رویکرد جستجوی محلی توسط کاروپلو و همکاران [۱۲] در سال ۲۰۰۰ برای حل مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود مورد بررسی قرار گرفت و نتایج قوی و خوبی از خود ارائه داد.آن‌ها ضریب تقریب برابر با $۵ + \epsilon$ را گزارش کردند. در تکنیک جستجوی محلی با هر جواب شدنی X می‌توان کار را شروع کرد. در هر تکرار از این روش، دو قاعده مهم وجود دارد:

- در هر تکرار جواب بعدی چگونه تعریف شود یا به عبارتی همسایگی بهترین جواب حال حاضر کجاست؟
- در چه صورتی و با چه شرطی جواب یافته شده به عنوان یک جواب یا نتیجه انتخاب یا رد صلاحیت می شود؟ مثلاً اگر یک همسایه مثل Y در اطراف X وجود دارد که به طور قابل توجهی از X بهتر است قرار می‌دهیم: $X := Y$ و به تکرار بعد می‌رویم.

در سال ۲۰۰۴ آریا [1] و همکاران [۱۱] ضریب تقریبی روش جستجوی محلی برای حل مسائل مکان‌یابی تسهیلات با ظرفیت نامحدود را از $۵ + \epsilon$ به ۳ بهبود بخشیدند. الگوریتم (۲-۳) روش جستجوی محلی آریا را شرح می‌دهد.

الگوریتم ۲-۳ الگوریتم جستجوی محلی

ورودی: یک جواب شدنی، $X_۰$
خروجی: جواب بهبود یافته X

۱: قرارد دهید: $X := X_۰$
۲: تا زمانی که $cost(op(X)) \leq (۱ - \frac{\epsilon}{p(n,m)})cost(X)$ عملیات$op(X) := X$ را انجام بده.
۳: X را به عنوان جواب بهبود یافته معرفی کن.

در این‌جا $۰ < \epsilon$ ثابت است، $n = |I|$ تعداد تسهیلات و $m = |J|$ تعداد مشتریان می‌باشد و $p(m,n)$ یک چندجمله‌ای در n و m است. تابع هزینه با $cost(X)$ و تنها عملیات مجاز $op(X)$ برای مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود روش جانشینی یا مبادله‌ای است. عملیات مبادله با بستن یک تسهیل باز در X ، $x \in X$ ، و باز کردن تسهیل بسته، $x' \notin I \backslash X$ عمل می‌کند.

$$op(X) := X - x + x' \qquad x \in X, x' \in I \backslash X$$

گاهاً می‌توان مجموعه $A \subseteq X$ را با مجموعه $B \subseteq I \backslash X$ مبادله کرد که به این روش جستجوی محلی با مبادله چندگانه

[1] Arya

می‌گویند. اگر ما X^* را جواب بهینه و X_0 را جواب اولیه معرفی کنیم در این صورت تعداد op که الگوریتم انجام می‌دهد، حداکثر برابر با $\frac{\log(cost(X_0)/cost(X^*))}{\log \frac{1}{1-\epsilon/p(n,m)}}$ خواهد بود. یک نمونه از چندجمله‌ای $p(n,m)$ می‌تواند $\log(cost(x))$ باشد که در اندازه ورودی‌های مسأله است و هر عملیات op را در زمان چندجمله‌ای اجرایی می‌کند [۱۱] [۱۸].

با وجود این‌که روش جستجوی محلی در رده تکنیک‌های ساخت الگوریتم‌های تقریبی قرار گرفته است اما در مطالعات بسیاری از تکنیک جستجوی محلی در ساخت الگوریتم‌های ابتکاری و فراابتکاری استفاده شده است.

۲-۳-۲ روش‌های ابتکاری و فراابتکاری

روش‌های ابتکاری و خصوصاً فراابتکاری جواب‌هایی با کیفیت بالا در زمانی کوتاه برای مسائل بهینه‌سازی ارائه می‌دهند. هر چند ضمانتی برای دست‌یابی به جواب بهینه با استفاده از این روش‌ها وجود ندارد، ولی توانایی بالای آن‌ها در دست‌یابی به جواب‌های نزدیک به بهینه در زمان کوتاه برای این مسائل، موجب شهرت فراوان آن‌ها شده است. انتخاب و به کارگیری این روش‌ها، 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 برسی کرد و در آخر از مشاوره‌ای قوی برای انتخاب ابزار مناسب استفاده کرد.

۲-۳-۲-۱ روش‌های ابتکاری

ابتکاری به تکنیک‌هایی برای حل، یادگیری و کشف مسأله، در مسائلی که جستجوی کامل فضای جواب غیر عملی است، اشاره دارد. روش‌های ابتکاری جهت سرعت بخشیدن به امر پیدا کردن جواب رضایت بخش وارد عمل می‌شوند. در علوم کامپیوتر ، هوش مصنوعی و بهینه‌سازی ریاضی، ابتکاری یک تکنیک طراحی شده برای حل مسائل با سرعت بیشتر در زمانی که روش‌های کلاسیک خیلی آهسته عمل می‌کنند، است. یا زمانی که روش‌های کلاسیک در پیدا کردن جواب قطعی شکست می‌خورند. اما آن‌ها هیچ ضمانتی جهت یافتن بهترین جواب ارائه نمی‌دهند. بنابراین روش‌های ابتکاری به نوعی الگوریتم تقریبی به‌شمار می‌آیند و الگوریتم دقیقی محسوب نمی‌شوند. این الگوریتم‌ها معمولاً خیلی سریع و آسان یک جواب نزدیک به جواب بهینه پیدا می‌کنند.

تکنیک‌ها: استفاده از تکنیک‌های شاخه و کران[۱] و برنامه‌ریزی پویا[۲] در حل مسائل بسیار مؤثر هستند امابه دلیل بالا بودن پیچیدگی زمانی برای مسائل NP- کامل غیر قابل قبول هستند. تکنیک تپه-نوردی[۳] نیز روش مؤثری است اما یک مشکل اساسی دارد و آن هم‌همگرایی پیش از موعد است به عبارت دیگر به دلیل اینکه این روش حریصانه است همیشه نزدیکترین جواب بهینه محلی حتی با کیفییت پایین را پیدا می‌کند. هدف روش‌های ابتکاری جدید غلبه بر این معایب و مشکلات است.

---

[۱] Branch-and-bound [۲] Dynamic programming [۳] Hill-climbing

الگوریتم تبرید شبیه سازی شده[1] که در سال ۱۹۸۳ ایجاد شد، از رویکرد مشابه تپه-نوردی استفاده می‌کند با این تفاوت که گاهاً جواب‌های را که از جواب حاضر شرایط بدتری دارند نیز می‌پذیرد. البته در روند اجرا، احتمال پذیرش جواب‌های بدتر به مرور زمان کاهش می‌یابد.

جستجوی ممنوعه هدف اجتناب کردن از بهینه محلی را با استفاده از ساختار حافظه، عملیاتی می‌کند. مشکل روش تبرید شبیه‌سازی شده این است که ممکن است بعد از یک پرش یا خیز(قبول جواب بدتر) در گام بعد دوباره به سمت جواب بهینه محلی قبلی کشیده شود. در جستجوی ممنوعه حرکت به سمت مسیری که اخیراً از آن آمده‌ایم ممنوع است.

هوش گروهی[2] در سال ۱۹۸۹ ایجاد شده است و یک تکنیک هوش مصنوعی بر پایه‌ی مطالعه رفتار جمعی سازمان‌های غیر تشکیلاتی و سیستم‌های خودسازمان یافته، می‌باشد. دو مورد از موفق‌ترین نوع این رویکرد شامل بهینه‌سازی کلونی مورچه[3] (ACO) و بهینه‌سازی ازدحام ذرات (PSO) می‌باشد. در ACO مورچه‌های مصنوعی با حرکت بر روی گراف مسأله و تغییر مسیرهای آن جواب‌ها را طوری می‌سازند که مورچه‌های جدید بتوانند جواب‌های بهتری بسازنند. PSO با مسائلی رو به رو است که یک جواب بهتر با یک نقطه یا سطح از یک فضای n- بعدی قابل نمایش باشد. مزیت اصلی تکنیک هوش جمعی این است که آن‌ها نسبت به بهینه محلی مقاوم هستند.

شبکه عصبی[4] از سیستم نورون زیستی الهام گرفته است. آن‌ها شامل واحدهایی به نام نورون و ارتباطات بین نورون‌ها می‌باشند. بعد از فراگیری برخی اطلاعات، شبکه عصبی می‌تواند روی موضوعاتی که تعلیم داده نشده پیش‌بینی کند. در عمل شبکه عصبی همیشه خیلی خوب کار نمی‌کند. یکی از مشکلات آن همگرایی زود هنگام آن است [۲۵].

۲-۳-۲-۲ روش‌های فراابتکاری

در علوم کامپیوتر، فراابتکاری، یک تکنیک محاسباتی را که می‌تواند یک جواب کاندید را تا میزان مشخص شده با روش‌های تکراری بهبود ببخشد،برای حل مسأله انتخاب می‌کند. فراابتکاری‌ها هیچ پیش‌فرضی در مورد مسأله بهینه‌سازی موجود ندارند و میتوانند فضای بزرگی از جواب‌های کاندید را برای یافتن بهینگی مسأله جستجو کنند، با این حال فراابتکاری‌ها هیچ تضمینی جهت یافتن جواب بهینه ارائه نمی‌دهند؛ از آن جهت که بسیاری از فرآیندهای بهینه‌سازی در فراابتکاری‌ها تصادفی و اتفاقی است. ویژگی‌های فراابتکاری‌ها عبارتند از:

- فراابتکاری‌ها، راهبردهای هدایت فرایندهای جستجو هستند و هدفشان کاوش مؤثر فضای جستجو به منظور یافتن جواب‌های نزدیک به بهینه است.
- از جستجوی محلی ساده تا فرآیندهای پیچیده یادگیری می‌توانند جزء تکنیک‌هایی باشند که در ساختار الگوریتم‌های فراابتکاری مورد استفاده قرار می‌گیرند.
- فراابتکاری‌ها الگوریتم‌های تقریبی و معمولاً غیر قطعی هستند.

---

[1] Simulated annealing [2] Swarm intelligence [3] Ant Colony Optimization [4] Neural Networks

- فراابتکاری‌ها مخصوص مسأله خاصی نیستند، به عبارتی از آن‌ها در حل تمامی مسائل می‌توان استفاده کرد.

یک نقل قول که اختلاف بین روش‌های ابتکاری و فراابتکاری را توضیح می‌دهد این است که "یک روش ابتکاری یک قاعده بسیار خوب است و یک فراابتکاری یک قاعده بسیار خوب برای یافتن قواعد بسیار خوب است." [۱]

تکنیک‌ها: به صورت رسمی یک روش فراابتکاری فرایند تولید تکراری تعریف می‌شود که یک تکنیک ابتکاری زیردست خود را با ترکیب هوشمندانه مفاهیم مختلف جهت پیگیری و پایش فضای جستجو به منظور یافتن جواب‌های نزدیک به بهینه هدایت می‌کند [۲۵].

طبقه‌بندی : فراابتکاری‌ها الگوریتم‌های بهینه‌سازی تقریبی هستند که دارای راهکارهای برون‌رفت از نقاط بهینه محلی‌اند و قابلیت کاربرد در طیف گسترده‌ای از مسائل را دارند. معیارهای مختلفی می‌تواند برای طبقه‌بندی الگوریتم‌های فراابتکاری استفاده شود:

- **مبتنی بر یک جواب و مبتنی بر جمعیت :** الگوریتم‌های مبتنی بر یک جواب در حین فرایند جستجو یک جواب را تغییر می‌دهند در حالی که در الگوریتم‌های مبتنی بر جمعیت در حین جستجو، یک جمعیت از جواب‌ها در نظر گرفته می‌شوند.
- **الهام گرفته شده از طبیعت و بدون الهام از طبیعت:** بسیاری از الگوریتم‌های فراابتکاری از طبیعت الهام گرفته شده‌اند؛ در این میان برخی از الگوریتم‌های فراابتکاری نیز از طبیعت الهام گرفته نشده‌اند.
- **با حافظه و بدون حافظه:** برخی از الگوریتم‌های فراابتکاری فاقد حافظه می‌باشند، به این معنا که، این نوع الگوریتم‌ها از اطلاعات بدست آمده در حین جستجو استفاده نمی‌کنند (به طور مثال تبرید شبیه‌سازی شده). این در حالی است که در برخی از الگوریتم‌های فراابتکاری نظیر جستجوی ممنوعه از حافظه استفاده می‌شود. این حافظه اطلاعات بدست آمده در حین جستجو را در خود ذخیره می‌کند.
- **قطعی و احتمالی:** یک الگوریتم فراابتکاری قطعی نظیر جستجوی ممنوعه، مسأله را با استفاده از تصمیمات قطعی حل می‌کند. اما در الگوریتم‌های فراابتکاری احتمالی نظیر تبرید شبیه‌سازی شده، یک سری قوانین احتمالی در حین جستجو مورد استفاده قرار می‌گیرد.

برخی از الگوریتم‌های شناخته شده فراابتکاری عبارتند از: الگوریتم ژنتیک، تبرید شبیه‌سازی شده و جستجوی ممنوعه. الگوریتم ژنتیک به تقلید از فرایند تکاملی در طبیعت ساخته شده است. در حالی که جستجوی ممنوعه از ساختار حافظه‌ای موجودات زنده بهره می‌گیرد. شبیه سازی تبریدی از فرآینده حرارت دادن زیاد جامدات متبلور و سپس سرد کردن تدریجی آن‌ها، تقلید می‌کند.

[۱] "A heuristic is a pretty good rule. A meta-heuristic is a pretty good rule for finding pretty good rules."

۲-۴ حل مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود با روش‌های فراابتکاری

در این بخش به شرح مختصری از برخی روش‌های فراابتکاری که نتایج قابل قبولی در مورد حل مسائل مکان‌یابی تسهیلات با ظرفیت نامحدود از خود نشان داده‌اند می‌پردازیم.

۲-۴-۱ الگوریتم ژنتیک

الگوریتم ژنتیک، (GA) که اولین بار در سال ۱۹۷۵ توسط جان هولند[۱] پیشنهاد شد یک راهبرد تکاملی است. مکانیسم اساسی در (GA) بر پایه نظریه تکامل داروین [۲] بنا نهاده شده است: «صفات خوب باقی می‌مانند و در فرزندآوری و تولید نسل جدید به کار می‌روند در حالی صفات بد از جمعیت حذف می‌شوند.»

الگوریتم ژنتیک یک الگوریتم بر پایه جمعیت است که با یک جمعیت اولیّه به عنوان نسل اول شروع به کار می‌کند. در اکثر مواقع انتخاب جمعیت اولیّه به صورت تصادفی انجام می‌پذیرد. هر جواب به تنهایی در جمعیت به عنوان کروموزم[۳] شناخته شده و هر عضوش یک ژن[۴] نامیده می‌شود. انواع مختلفی از رشته‌های کروموزم وجود دارد؛ مانند رشته‌های صفر و یک، اعداد حقیقی، جایگشتی از عناصر و غیره. اگر یک کروموزم نتیجه بهتری، برای تابع هدف مسأله مورد نظر ارائه دهد، گزینه‌ی مناسبی برای باقی ماندن در گروه (جمعیت) است. یک نسل جدید در هر تکرار با به‌روز رسانی جمعیت بازمانده ساخته می‌شود. به‌روز رسانی جمعیت با عملیات ژنتیکی است که تغییراتی در بازماندگان ایجاد می‌کند.

تابع برازندگی[۵]: برازندگی هر کروموزم با یک تابع محاسبه می‌شود و یک ارزش به هر کروموزم تخصیص داده می‌شود. تابع برازندگی تنها بخشی از الگوریتم است که به طور کلی مربوط به مسأله خاصی است که باید بهینه شود. اپراتورها نقش مهمی در الگوریتم ژنتیک دارند. مهم‌ترین آن‌ها عبارتند از:

- **انتخاب[۶]:** اعضای جمعیت جدید بر اساس برازندگی خود انتخاب می‌شوند. روش‌های گزینش بسیار زیادی با پیچیدگی‌های متنوع وجود دارند. یک روش سهل‌گیرانه این است که تعداد زیادی از جواب‌ها را انتخاب کنیم و یک روش سخت‌گیرانه تنها به چند یا یک جواب اجازه انتخاب شدن می‌دهد. با این حال تعادل باید رعایت شود تا مانع به دام افتادن الگوریتم در بهینه محلی شود. یک روش ساده انتخاب، روش چرخنده انتخاب است که در آن برای تمام افراد جامعه با توجه به میزان برازندگیشان، شانسی برای انتخاب شدن در نظر گرفته می‌شود.
- **ترکیب[۷]:** مهم‌ترین اپراتور تولید مثل ژنتیک، ترکیب است. در آخرین مرحله، فرزند از دو کروموزم انتخاب شده حاصل می‌شود. ویژگی‌های ژن‌های فرزند از انتخاب تعدادی ویژگی‌های پدر و مادر بدست می‌آید. انواع مختلفی از تقاطع (ترکیب) وجود دارد؛ مانند تقاطع یک نقطه‌ای که در آن، ابتدا یک محل بر روی کروموزم والدین انتخاب

---

[۱] John Holland [۲] Darvin [۳] Chromosome [۴] Gene [۵] Fitness function [۶] Selection [۷] Crossover

و سپس والدین را از آن قسمت به دو قسمت برش می‌زند و در نهایت فرزند اول از اتصال قسمت اول والد اول به قسمت دوم والد دوم ایجاد می‌شود و فرزند دوم از اتصال قسمت دوم والد اول به قسمت اول والد دوم ایجاد می‌گردد.

- **جهش**[۱]: شامل ایجاد تغییرات تصادفی بر روی ژن‌های یک کروموزم است که در طول اجرای الگوریتم با یک احتمال خاص و به صورت تصادفی اعمال می‌شود و هدف از اجرای آن حفظ پراکندگی در جمعیت است. یک روش ایجاد جهش جابه‌جایی مکان دو ژن بر روی یک کروموزم است و روش دیگر در مسائل صفر و یک تغییر مقدار ژن منتخب از ۰ به ۱ و برعکس می‌باشد.

نتیجه این عملیات‌ها (انتخاب، ترکیب،جهش) تولید یک نسل جدید هم اندازه نسل اولیّه است [۲].

الگوریتم ۲-۴ الگوریتم ژنتیک

ورودی: تابع برازندگی ، فضای جواب
خروجی: جواب بهبود یافته X

۱: با انتخاب یک جمعیت اولیّه به طور تصادفی شروع کنید.
۲: افرادی را برای تولید مثل انتخاب کنید.
۳: نسل جدید را با عملیات ژنتیکی (جهش-تقاطع) تولید کنید.
۴: فرزندان را در جمعیت قرار دهید و جمعیت را به‌روز رسانی کنید.
۵: اگر معیار توقف حاصل شده است الگوریتم را متوقف کنید، در غیر این صورت به مرحله ۲ بازگردید.

هیرواکی تاهی‌آما[۲] و همکارانش [۲۶] در سال ۲۰۱۱ الگوریتم ژنتیک را جهت حل مسأله مکان‌یابی تسهیلات بکار بردند و آن را با روش جستجوی ممنوعه، جستجوی محلی و الگوریتم حجم-محدود مقایسه کردند. نتایج حاصل از کار آما نشان می‌دهد که الگوریتم جستجوی محلی ساده در حل بیشتر مسائل ULF شکست خورده است و از میان دیگر الگوریتم‌ها، الگوریتم ژنتیک با نتایج بهتر نسبت به دو الگوریتم دیگر در خصوص کیفیت جواب‌ها، موفق‌تر عمل کرده است اما از نظر زمان از دو الگوریتم دیگر زمان‌برتر است. ولی در کل او الگویتم ژنتیک را الگوریتم کارا در حل مسائل مکان‌یابی تسهیلات با ظرفیت نامحدود معرفی می‌کند. شاهی [۲۷] نیز این روش را جهت حل مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود مورد استفاده قرار داد. او کارایی الگوریتم ژنتیک را بهتر از الگوریتم جستجوی ممنوعه و الگوریتم بهینه‌سازی ازدحام ذرات گزارش کرده است.

۲-۴-۲ الگوریتم جستجوی ممنوعه

جستجوی ممنوعه (TS) اولین بار توسط گلور[۳] [۲۸] در سال ۱۹۸۶ ابداع شد. ایده‌های اساسی آن توسط هانس[۴] [۲۹] در سال ۱۹۸۶ طراحی شده است و ورا و هرتز[۵] [۳۰] تلاش‌هایی برای فرمول‌بندی رسمی آن انجام داده‌اند. می‌توان مثال‌ها

---

[۱]Mutation [۲]Hiroaki tohyama [۳]Glover [۴]Hansen [۵]Werra and Hertz

و کاربردهای خوب فراوانی از جستجوی ممنوعه به همراه مجموعه‌ای از ارجاعات را در کتاب گلور و لانگونا[۱] [۳۱] پیدا کرد. همگرایی این الگوریتم توسط هنفی[۲] [۳۲] اثبات شده است. الگوریتم جستجوی ممنوعه یک جستجوی همسایگی (محلی) با روش تکراری است که در تکرار K ، از جواب حاضر X به نزدیک‌ترین جواب X_{new} می‌رود. X_{new} از میان مجموعه جواب‌های همسایه $N(X,K)$ از جواب حاضر X انتخاب می‌شود. حداقل سه بخش در جستجوی ممنوعه ساده وجود دارد که به شرح زیر است:

- **تابع ارزیابی**[۳]: تابع ارزیابی $f(X)$ که مشخصه هر مسأله است و ارزش هر جواب را محاسبه می‌کند.
- **لیست ممنوعه (تابو)**[۴]: الگوریتم جستجوی ممنوعه دارای دو نوع حافظه، کوتاه مدت و بلند مدت است. حافظه کوتاه مدت با ذخیره سازی آخرین حرکت انجام شده در لیست ممنوعه $T(X)$ و منع کردن حرکت به سوی مسیری که ممکن است به جواب‌های قبلی منجر شود؛ از تکراری بودن جواب‌ها در یک تعداد محدود از حرکات متوالی، جلوگیری می‌کند.
- **حافظه بلند مدت**[۵]: ارزش $f(X^*)$ از بهترین جواب X^* که در طول مدت اجرای برنامه بدست آمده است در حافظه بلند مدت ذخیره می‌شود. حافظه بلند مدت زمانی که یک جواب بهتر پیدا شود به‌روز رسانی می‌شود، بنابراین این حافظه در زمان نیاز با پرش ناگهانی به بهترین جواب مانع از به دام افتادن در بهینه محلی بدتر می‌شود. گاها برای این حافظه یک ظرفیت به‌روزرسانی (C) مشخص می‌شود که در تمام طول اجرای برنامه فقط C مرتبه اجازه به‌روز رسانی حافظه بلند مدت را می‌دهد.
- **دو مفهوم مهم:**

تشدید[۶]: گاهی اوقات فرایند جستجو در برخی مناطق فضای جستجو تشدید می‌شود چرا که در هر تکرار جواب‌ها در آن مناطق بهتر می‌شوند، بنابراین برای دادن اولویت بیشتر به جواب‌های نزدیک به جواب حال حاضر در آن مناطق ممکن است یک تعریف اضافی در تابع ارزیابی اعمال شود که جواب‌های دورتر از ناحیه مورد نظر را جریمه می‌کند.

تنوع[۷]: در زمان‌هایی لازم است با ترک منطقه حال حاضر و رفتن به مناطق دیگر فضای جواب، به جواب‌ها تنوع ببخشیم، چرا که ادامه جستجو در منطقه حال حاضر از نظر تابع ارزیابی رضایت‌بخش نیست؛ بنابراین جهت دادن اولویت بیشتر به جواب‌های دورتر از جواب حال حاضر، ممکن است یک تعریف اضافی در تابع ارزیابی اعمال شود که جواب‌های نزدیک به جواب حاضر را جریمه می‌کند. با استفاده از تشدید و تنوع، تابع ارزیابی به صورت زیر تغییر می‌کند.

$$f = f + \text{جریمه تشدید} + \text{جریمه تنوع}$$

- **معیار تنفس**[۸]: معیار تنفس $A(i)$ یک پیش‌بینی یا احتیاط اضافی است که مانع از دست دادن جواب‌های خوب می‌شود. بعضی اوقات یک جواب به رغم تعلق داشتن به لیست ممنوعه بهتر از هر جوابی است که تاکنون بدست

---

[۱] Languna [۲] Hanafi [۳] Evaluation function [۴] Tabu list [۵] Long term memory
[۶] Intensification [۷] Diversification [۸] Aspiration criteria

آمده است، بنابراین الگوریتم اجازه می‌دهد که ممنوعیت از روی این جواب برداشته و رفتن به آن جواب قابل قبول فرض شود.

الگوریتم ۲-۵ الگوریتم جستجوی ممنوعه [۲]

ورودی: تابع برازندگی ، فضای جواب S
خروجی: جواب بهبود یافته X^*

۱: جواب اولیّه X را از S انتخاب کنید و قرار دهید: $X^* := X$ و $K := ۰$
۲: قرار دهید: $K := K + ۱$ و مجموعه $N(X, K)$ را به وسیله $T(X)$ و $A(X)$ تولید کنید.
۳: بهترین X_{new} را از میان $N(X, K)$ انتخاب کنید.
۴: قرار دهید: $X := X_{new}$.
۵: اگر $f(X) < f(X^*)$ آن‌گاه قرار دهید: $X^* := X$.
۶: لیست ممنوعه را به روز رسانی کنید.
۷: اگر شرایط توقف فراهم شده متوقف شوید. در غیر این صورت به گام ۲ بروید.

برخی از شرایط توقف فوری برای الگوریتم (TS) توسط هرتز و همکاران [۳۳] در سال ۱۹۹۵ معرفی گردیده است. در سال ۲۰۰۶ سان[۱] [۳۴] روش جستجوی ممنوعه را با نتایج روش لاگرانژین و روش ابتکاری جستجوی همسایگی قوش[۲] و همچنین روش ترکیبی فراابتکاری[۳] مقایسه کرد، که جواب روش جستجوی ممنوعه در بعضی نمونه‌ها با جواب این الگوریتم‌ها یا برابری می‌کند و یا برتر است. میچل و هنتنریک[۴] [۳۵] در سال ۲۰۰۴ از یک روش جستجوی ممنوعه ساده برای حل مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود استفاده کردند و معتقدند که این روش در مجموع روش باارزشی برای حل این مسأله است.

۲-۴-۳ الگوریتم بهینه‌سازی ازدحام ذرات

الگوریتم بهینه‌سازی ازدحام ذرات (PSO) که یک الگوریتم الهام گرفته شده از طبیعت و مبتنی بر جمعیت است در سال ۱۹۹۵ توسط ابرهارت و کندی[۵] معرفی شد. ایده‌ی اصلی آن از حرکت گروهی پرندگان و ماهی‌ها گرفته شده است. الگوریتم بهینه‌سازی ازدحام ذرات به دلیل همگرایی سریع، محاسبات ساده و فهم آسان، کاربردهای فراوانی در مسائل بهینه‌سازی پیوسته دارد. همچنین قابلیت استفاده در مسائل گسسته را دارا می‌باشد. عملکرد این الگوریتم به پارامترهای آن بستگی دارد و ممکن است الگوریتم در بهینه محلی به دام افتد.

[۱]Sun [۲]Ghosh [۳]A hybrid multi start heuristic [۴]Michel and Hentenryck [۵]Eberhart and Kennedy

۲-۴-۳-۱ الگوریتم بهینه‌سازی ازدحام ذرات برای مسائل صفر و یک

در سال ۱۹۹۷ کندی و ابرهارت [۳۶] یک الگوریتم بهینه‌سازی ذرات را برای بهینه‌سازی تابع $f : B^n \longrightarrow \mathcal{R}$ که در آن $B := \{۰,۱\}$ پیشنهاد دادند. در رویکرد آن‌ها هر کدام از ذرات دارای یک موقعیت $\vec{X}_{i.t} \in \{۰.۱\}^n$ که یک بردار صفر و یک است و یک سرعت پیوسته $\vec{V}_{i,t} \in [-V_{max}, V_{max}]^n$ می‌باشد. علاوه بر این، هر ذره یک راهنمای خصوصی $\vec{P}_{i,t}$ که یک بردار صفر و یک است را در خود ذخیره می‌کند. این بردار شامل اطلاعات بهترین موقعیت آن ذره تا آن مرحله است. همچنین هر ذره با همسایگان خود در ارتباط است تا یک راهنمای محلی $\vec{L}_{i,t}$ را بدست آورد. راهنمای محلی یک بردار صفر و یک، حامل مشخصات بهترین ذره در کل جمعیت است. در الگوریتم بهینه‌سازی ازدحام ذرات جهت به‌روز رسانی سرعت در مسائل پیوسته ما از رابطه:

$$\vec{V}_{i,t} = \omega.\vec{V}_{i,t-۱} + C_۱.\vec{r}_{۱,i,t} \odot (\vec{P}_{i,t-۱} - \vec{X}_{i.t-۱}) + C_۲.\vec{r}_{۲,i,t} \odot (\vec{L}_{i,t-۱} - \vec{X}_{i.t-۱})$$

که در آن $r_۱$ و $r_۲$ اعداد تصادفی در بازه [۰،۱] هستند و $C_۱$ و $C_۲$ ثابت‌های مثبت می‌باشند؛ استفاده می‌کنیم. در حالت صفر و یک این الگوریتم، با حدف وزن ω از رابطه فوق از آن جهت به‌روز رسانی سرعت استفاده می‌شود. برای همه مولفه d از بردار سرعت، $d \in \{۱, ..., v\}$ با استفاده از تابع $g : [-V_{max}, V_{max}]^n \longrightarrow [۰, ۱]$ به بازه صفر و یک می‌رود. مقدار تابع $g(\vec{V}_{i,t,d})$ مشخص کننده احتمال یک بودن مولفه مربوط به مولفه d در بردار موقعیت است. بنابراین به‌روز رسانی موقعیت هر ذره بر اساس شرایط زیر صورت می‌پذیرد:

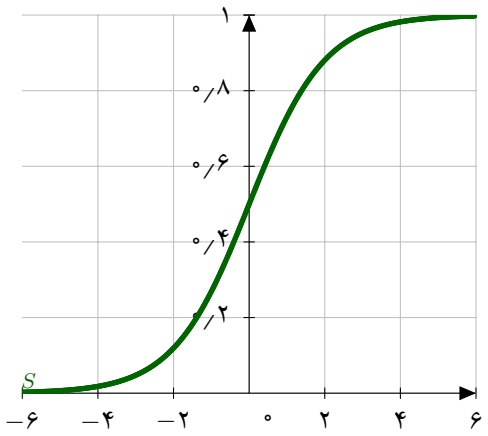
$$\vec{X}_{i.t,d} = \begin{cases} ۱ & R_{i,t,d} < g(\vec{V}_{i,t,d}) \text{ اگر} \\ ۰ & \text{در غیر این صورت} \end{cases}$$

که در آن $R_{i,t,d}$ یک عدد تصافی با توزیع یکنواخت از بازه $[۰, ۱]$ انتخاب شده است. یک تابع اختیاری که برای $g : [-V_{max}, V_{max}]^n \longrightarrow [۰, ۱]$ می‌تواند مورد استفاده قرار گیرد تابع افزایش یکنواخت سیگموئید می‌باشد.

$$S(v) = \frac{۱}{۱ + e^{-v}}$$

تعریف ۲-۴-۱. تابع سیگموئید: تابعی حقیقی، کراندار و مشتق‌پذیر که به ازای کلیه مقادیر حقیقی قابل تعریف بوده و دارای مشتق مثبت است. این تابع به لحاظ گرافیکی شکلی شبیه حرف S انگلیسی دارد.

زمانی که ما از تابع سیگموئید استفاده می‌کنیم ممکن است مثلاً V_{max} را برابر با ۶ و یا ۴ در نظر بگیریم در این صورت $S(-۶) \approx ۰/۹۹۷۵$ و $S(۶) \approx ۰/۰۰۴۵$ (و به ترتیب $S(-۴) \approx ۰/۹۸۲۰$ و $S(۴) \approx ۰/۰۱۸۰$) بنابراین برای هر مولفه d در بردار موقعیت احتمال بسیار کمی وجود دارد که با وجود بالا بودن مقدار $S(v)$ آن مولفه، مقدار صفر بگیرد و



شکل ۲-۲: رسم تابع سیگموئید $S(v) = \frac{۱}{۱+e^{-v}}$ در بازه [۶، ۶-]

برعکس با وجود پایین بودن $S(v)$ مولفه d ، مقدار آن مولفه در بردار موقعیت یک باشد. بنابراین در این‌جا عملی شبیه به جهش در الگوریتم تکاملی رخ می‌دهد. انتخاب V_{max} های کوچک‌تر از ۴ میزان احتمال این عمل را برای هر ذره افزایش می‌دهد [۳۷].

بر اساس تجزیه و تحلیل زمان اجرای برنامه الگوریتم PSO صفر و یک، توسط نیومن، شادهولت و ویت[۱] [۳۸] در سال ۲۰۰۸ پیشنهاد می‌شود مقدار V_{max} برای حل مسائل به صورت زیر محاسبه شود:

$$V_{max} = \ln(n - ۱)$$

که در آن n تعداد ورودی‌های مسأله است.

در سال ۲۰۰۸ برای اولین بار گانر و سوکلی [۳۹][۲] دو مدل از الگوریتم بهینه‌سازی ازدحام ذرات شامل الگوریتم پیوسته‌ی بهینه‌سازی ازدحام ذرات[۳] (CPSO) ابداعی خود و الگوریتم گسسته بهینه‌سازی ازدحام ذرات[۴] (DPSO) ابداعی پن[۵] و همکاران [۴۰] را جهت حل ۱۵ مسأله معروف مکان‌یابی تسهیلات با ظرفیت نامحدود شامل ۱۲ مسأله در سایز کوچک و ۳ مسأله نسبتا بزرگ، موجود در سایت OR-library بکار بردند. همچنین آن‌ها با استفاده از یک نوع خاص از جستجوی محلی ساده (LS) در ساختار این مدل‌ها کارایی آن‌ها را افزایش دادند. آن‌ها با مقایسه این دو روش با یکدیگر به این نتیجه رسیدند که نوع $(CPSO_{LS})$ در حل مسائل مکان‌یابی تسهیلات با ظرفیت نامحدود ناکارآمد بوده اما روش $(DPSO_{LS})$ کارایی خوبی از خود نشان می‌دهد. سرانجام با مقایسه الگوریتم $DPSO_{LS}$ با الگوریتم ژنتیک معرفی شده توسط جارامیلو[۶] و همکاران [۴۱] و الگوریتم تبرید شبیه‌سازی شده تکاملی[۷] (ESA) مطرح شده توسط آیدین و فوگنرتی[۸] [۴۲] در شرایط اجرایی یکسان با $DPSO_{LS}$ نشان دادند که عملکرد $DPSO_{LS}$ کمی بهتر از GA است خصوصا در مورد زمان اجرایی خیلی بهتر از آن به نظر می‌رسد. اما در مورد الگوریتم ESA و در خصوص شاخص انحراف از جواب

---

[۱] Neumann, Sudholt and Witt [۲] Guner and Sevkli [۳] Continuous PSO [۴] Discrete PSO
[۵] Pan [۶] Jaramillo [۷] Evoluitionary [۸] Aydin and Fogarty

بهینه بسیار شبیه به هم هستند البته باز در مورد زمان اجرایی $DPSO_{LS}$ خصوصاً در مسائلی با اندازه بزرگ، عملکرد بهتری دارد.

۲-۴-۴ الگوریتم خفاش

الگوریتم خفاش[۱] یک الگوریتم فراابتکاری جدید در مفهوم هوش جمعی است که در سال ۲۰۱۰ توسط شین-شی یانگ[۲] [۴۳] و با الهام گرفتن از قابلیت پژواک‌یابی[۳] خفاش‌های کوچک معرفی شد. پژواک‌یابی مهم‌ترین ابزار خفاش‌های کوچک در یافتن شکار حتی در تاریکی مطلق به‌شمار می‌آید. پژواک‌یابی شامل انتشار ضربه‌هایی از صوت با بلندی خاص و سپس گوش سپردن به صدای بازتاب آن‌ها، به منظور یافتن شکار، اجتناب از موانع و قرار گرفتن در مخفی‌گاه است. در پژواک‌یابی، بلندی صدا با نزدیک شدن خفاش به شکار کاهش می‌یابد. با توجه به مطالعات انجام شده بر روی رفتار خفاش‌های کوچک، یک خفاش کوچک با محاسبه اختلاف بین زمان انتشار صدای خود و زمان شنیدن بازتاب صدا توسط گوشش و همچنین محاسبه اختلاف فرکانس[۴] بین این دو، توانایی تصویرسازی ذهنی از محیط به صورت سه بعدی را دارد. با توجه به این رفتار خفاش‌های کوچک، الگوریتم خفاش جهت یافتن جواب بهینه در فضای جواب فرمول‌بندی شده است.

با توجه به این‌که در این پایان‌نامه از این روش برای حل مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود استفاده خواهد شد، در فصل بعد این روش به صورت مفصل بیان می‌شود.

---

[۱] Bat Algorithm [۲] Xin-She Yang [۳] Echolocation [۴] Frequency

فصل ۳

الگوریتم خفاش

در این فصل ما الگوریتمی برگرفته شده از طبیعت به نام الگوریتم خفاش که الهام گرفته از رفتار انعکاس‌صدای خفاش‌هاست را توضیح می‌دهیم. پس از بیان جزئیات و توضیح الگوریتم، به کاربرد آن در مسائل پیوسته و گسسته با حل یک مثال می‌پردازیم. بیشتر مفاهیم و توضیحات از منابع [۴۳] و [۴۴] گردآوری شده است.

۳-۱ نگاهی بر زندگی خفاش‌ها

خفاش‌ها حیوانات شگفت‌انگیزی هستند. آن‌ها تنها پستانداران بالدارند که قابلیت پیشرفته‌ی انعکاس صدا را دارند. تخمین زده می‌شود که حدود ۱۰۰۰ گونه مختلف از خفاش‌ها وجود دارد که ۲۰ درصد از تمام گونه‌های پستانداران را تشکیل می‌دهند. خفاش جانوری اجتماعی است. آنها در دسته‌های بزرگ در غارها یا روی درختان، بیتوته می‌کنند. به صورت سنتی دو دسته‌ی کلی برای خفاش‌ها معرفی شده‌است که عبارتند از:

- خفاش میوه‌خوار (بزرگ‌خفاش[۲])
- کوچک‌خفاش[۳]

بزرگ‌خفاش‌ها از میوه، شهد یا گرده تغذیه می‌کنند و سحرگاه و نزدیک غروب به جستجوی غذا می‌روند. آن‌ها برای پیدا کردن میوه و شیره‌ی درختان مورد علاقه‌شان از حس بینایی خوب و پیشرفته و حس بویایی خود بهره می‌برند. روباه پرنده[۴] از نام‌هایی است که در برخی گونه‌های خفاش‌های میوه‌خوار استفاده می‌شود. سر این خفاش‌ها حالتی مانند سر سگ دارد. بزرگترین آن‌ها خفاش غول پیکر است، سنگینی این خفاش ۱.۶ کیلوگرم و پهنای آن هنگامی که بال‌هایش باز است به ۱.۷ متر می‌رسد.

---

[۲]Megabat [۳]Microbat [۴] Flying fox

شکل ۳-۱: بزرگ خفاش میوه‌خوار

خوراک بیشتر **کوچک‌خفاش‌ها** حشره است. برخی از آن‌ها هم ماهی، قورباغه، پستانداران کوچک، خون دیگر حیوانات و گاهی گرده و شهد می‌خورند. کوچک‌خفاش برای شکار و راهیابی تنها بر پژواک‌یابی خود تکیه می‌کند. کوچکترین خفاش حشره‌خوار است، که قد آن ۲۹ تا ۳۴ میلی متر و سنگینی آن ۲ تا ۲.۶ گرم است و اگر بال‌هایش باز باشد طول آن به ۱۵ سانتی متر می‌رسد.

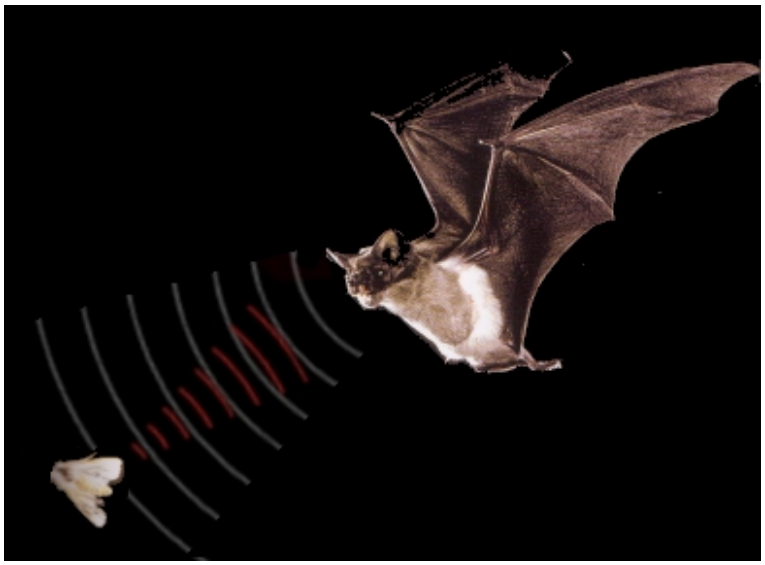
شکل ۳-۲: کوچک خفاش حشره‌خوار

۳-۱-۱ پژواک‌یابی

پژواک‌یابی در جانوران، که با نام زیست‌سونار[۱] نیز شناخته می‌شود، روشی مبتنی بر به کارگیری امواج صوتی توسط جانوران برای اهدافی چون موقعیت‌یابی، جهت‌یابی، و پیدا کردن طعمه است. چندین گونه جانور از پژواک‌یابی برای اهداف گوناگون بهره می‌گیرند. از جمله این جانوران می‌توان به خفاش‌ها و نهنگ‌های دندان‌دار[۲] اشاره کرد. پژواک‌یابی خفاش‌ها در عمل یک سامانه‌ی ادراکی است که در آن موج‌های فراصوتی برای بدست آوردن پژواک، تولید می‌شوند. مغز و دستگاه عصبی

[۱] Bio sonar [۲] Toothed whales

خفاش با مقایسه‌ی موج‌های فرستاده شده و موج‌های بازتاب شده، می‌تواند تصویری از فضای پیرامون و جزئیاتش برای خود بسازد. این توانایی به شب‌کورها اجازه می‌دهد تا در تاریکی مطلق محل شکارشان را شناسایی و حتی نوع شکار را تشخیص دهند. خفاش‌های کوچک استفاده زیادی از پژواک‌یابی می‌کنند به طوری که در برخی گونه‌ها خفاش‌های کوچک بینایی بسیار ضعیفی دارند؛ در حالی که خفاش‌های بزرگ این گونه نیستند؛ یا از پژواک‌یابی استفاده چندانی نمی‌کنند یا اصلاً بی‌بهره‌اند.



شکل ۳-۳: پژواک‌یابی: استفاده خفاش از صوت جهت یافتن شکار

امواج صوتی که توسط خفاش‌ها مورد استفاده می‌گیرد با توجه به نوع خفاش و نوع موجودی که شکار می‌کند با یکدیگر متفاوت است و ویژگی‌های خاص خود را دارد. بسیاری از خفاش‌ها از **فرکانس** با طول موج[۱] کوتاه استفاده می‌کنند، محدوده این فرکانس‌ها بین ۲۵ تا ۱۵۰ کیلو هرتز قرار دارد؛ هر چند بعضی از گونه‌ی خفاش‌ها می‌توانند فرکانس‌هایی بالاتر از ۱۵۰ کیلو هرتز نیز منتشر کنند.

با توجه به این‌که سرعت صوت در هوا به طور معمول ۳۴۰ متر بر ثانیه در نظر گرفته می‌شود. **طول موج** انفجارهای ساطع شده از خفاش‌ها با توجه به محدوده فرکانسی ۲۵ تا ۱۵۰ کیلو هرتزی و با استفاده از رابطه $\lambda = \frac{v}{f}$ در محدوده‌ی ۲ تا ۱۴ میلی‌متر قرار می‌گیرد. جالب است که اندازه این طول موج‌ها مطابق با اندازه شکار خفاش‌ها می‌باشد.

خفاش‌ها یک **پالس**[۲] یا انفجار با صدای بسیار بلند را منتشر می‌کنند و به بازتاب صدای آن گوش می‌دهند. هر انفجار کمتر از چند هزارم ثانیه (۸ تا ۱۰ میلی ثانیه) به طول می‌انجامد. برخی خفاش‌های کوچک می‌توانند ۱۰ تا ۲۰ انفجار را در هر ثانیه تولید کنند.

در هنگام شکار زمانی که خفاش به شکارش نزدیک می‌شود تعداد انفجارها افزایش می‌یابد به طور مثال در خفاش کوچک تا حدود ۲۰۰ انفجار در هر ثانیه می‌رسد. چنین سرعتی نشان‌دهنده توانایی فوق‌العاده‌ی خفاش‌ها در پردازش امواج صوتی است.اختلاف بلندی صدا بین امواج رفت و برگشت یک اکتاو[۳] است. به عبارتی یکی دو برابر دیگری است.

---

[۱] Wavelength [۲] Pulse [۳] Octave

تعریف ۳-۱-۱. در تئوری موسیقی، اُکتاو یا هنگام فاصله دو نت هم‌نام متوالی است. اگر فرکانس دو نت متوالی که با هم یک اکتاو فاصله دارند اندازه‌گیری شود، فرکانس یکی دو برابر دیگری خواهد بود (نسبت ۱:۲). مثلاً نت «لا» در یک پیانو فرکانسی برابر با ۴۴۰ هرتز دارد و نت لای بعدی بسامدش ۸۸۰ هرتز است و نت لای قبلی ۲۲۰ هرتز می‌باشد.

ویژگی‌های پژواک‌یابی خفاش‌های کوچک می‌تواند به گونه‌ای فرموله شود که با مرتبط کردن آن با تابع هدف امکان ایجاد یک الگورتیم بهینه‌سازی را فراهم کند.

۳-۲ الگوریتم خفاش

یانگ در سال ۲۰۱۰ الگوریتم خفاش را با توجه به ویژگی‌های پژواک‌یابی و بر اساس سه قانون اساسی زیر ایجاد نمود:

۱. همه خفاش‌ها از پژواک‌یابی به عنوان یک حس از راه دور استفاده می‌کنند. آن‌ها توانایی تشخیص تفاوت بین امواج بازگشتی از یک طعمه و موانع زمینه‌ای موجود بر سر راه را به طور شگفت‌انگیز و باور نکردنی در خود دارند.

۲. خفاش‌ها به صورت تصادفی با سرعت v_i در موقعیت x_i با فرکانس f_i یا طول موج λ_i و بلندی صدای A_0 جهت یافتن طعمه به پرواز در می‌آیند و توانایی تنظیم خودکار فرکانس (یا طول موج) و تعداد پالس‌های منتشر شده را با توجه به نزدیکی به هدف دارند.

۳. فرض می‌کنیم که بلندی صدا در محدودی A_0 با مقادیر مثبت تا مقدار ثابت حداقلی A_{min} در تغییر است

برای سادگی، می‌توانیم تنها از فرکانس جهت فرموله کردن الگوریتم خفاش استفاده کنیم چرا که فرکانس ذاتاً با طول موج ارتباط دارد. به عنوان مثال محدوده فرکانس بین ۲۰ تا ۵۰۰ کیلو هرتز مرتبط با محدوده‌ی طول موج از ۷ تا ۱۷ میلی‌متر در هوا است. انتخاب فرکانس یا طول موج به شرایط برنامه‌های کاربردی، سهولت اجرای برنامه و دیگر عوامل بستگی دارد.

۳-۲-۱ حرکت خفاش‌ها

هر خفاش با سرعت v_i^t و موقعیت x_i^t در تکرار tام در فضای جستجو یا جواب d بعدی برچسب‌گذاری می‌شود. در میان همه خفاش‌ها تنها یک جواب x^* به عنوان بهترین جواب حال حاضر وجود دارد. بنابراین می‌توان سه قانون اساسی فوق را در معادلات به‌روز رسانی موقعیت و سرعت هر خفاش استفاده کرد.

$$f_i = f_{min} + (f_{max} - f_{min})\beta \tag{۳-۱}$$

$$v_i^t = v_i^{t-1} + (x_i^{t-1} - x^*)f_i \tag{۳-۲}$$

$$x_i^t = x_i^{t-1} + v_i^t \tag{۳-۳}$$

که در آن $\beta \in [۰, ۱]$ یک بردار تصادفی با توزیع یکنواخت می‌باشد و x^* بهترین مکان فعلی است که در هر تکرار پس از بررسی موقعیت خفاش‌های مجازی انتخاب می‌شود. معمولاً فرکانس f را در محدوده $f_{min} = ۰$ و $f_{max} = \mathcal{O}(۱)$ در نظر می‌گیرند، محدوه فرکانس وابسته به دامنه‌ی مسأله مورد نظر است. به همین دلیل، می‌توان الگوریتم خفاش را به عنوان یک الگوریتم تنظیم کننده فرکانس، جهت اکتشاف فضای جواب و استخراج بهترین جواب، معرفی کرد. میزان بلندی صدا و تعداد انفجارهای منتشر شده با استفاده از مکانیسمی جهت دقیق شدن بر ناحیه‌ی که احتمال جواب بهتر در آن است، کنترل می‌شوند.

گاهی اوقات تشدید جستجو در اطراف بهترین جواب‌ها نتایج خوبی به دنبال دارد و موجب یافتن جواب‌های بهتر می‌شود؛ از این رو در هر تکرار با یک احتمالی، امکان دارد جستجو در اطراف خفاش iام متوقف شود و ادامه جستجوی شکار در اطراف یکی از بهترین خفاش‌ها صورت گیرد. در این حالت محل شکار (جواب جدید) با توجه به رابطه زیر بدست می‌آید:

$$x_{new} = x_{best} + \epsilon A^t \qquad (۳-۴)$$

که در آن $\epsilon \in [-۱, ۱]$ یک عدد تصادفی بوده و A^t میانگین بلندی صدای خفاش‌ها در تکرار t می‌باشد.

۳-۲-۲ تغییرات میزان بلندی صدا و تعداد انفجارها

به منظور ارائه یک مکانیسم موثر برای کنترل اکتشاف فضای جواب و استخراج بهترین جواب از آن و رفتن از مرحله اکتشاف به استخراج تغییراتی در بلندی صدا A_i و تعداد انفجارها[۱] r_i در هر تکرار ایجاد می‌شود. از آن جایی که در زمانی که خفاش شکار خود را می‌یابد بلندی صدا کاهش می‌یابد و برعکس آن تعداد انفجارهای منتشر شده افزایش می‌یابد؛ بلندی صدا می‌تواند مقداری بین A_{max} تا A_{min} باشد به طوری که $A_{min} = ۰$ فرض می‌شود به این معنی که در زمان رسیدن به شکار انتشار صوت توسط خفاش به طور موقت و برای مدتی متوقف می‌شود و خفاش هیچ صدای از خود تولید نمی‌کند. با این فرضیات ما داریم:

$$A_i^{t+۱} = \alpha A_i^t \qquad (۳-۵)$$

$$r_i^{t+۱} = r_i^۰[۱ - \exp(-\gamma t)] \qquad (۳-۶)$$

که در آن $r^۰$ بیشترین تعداد انفجارهایی است که یک خفاش می‌تواند از خود تولید کند و α و γ مقادیر ثابتی هستند. در اصل در این‌جا α مشابه یک عامل خنک کننده در الگوریتم تبرید شبیه‌سازی شده به کار رفته است.

[۱] Rates

برای هر $\alpha < ۱ > ۰$ و $\gamma > ۰$ ما داریم:

$$A_i^t \longrightarrow ۰ \qquad (۳-۷)$$

$$r_i^t \longrightarrow r_i^\circ \qquad (۳-۸)$$

که در آن $t \longrightarrow \infty$ میل می‌کند. در ساده‌ترین حالت می‌توان از $\alpha = \gamma$ استفاده کرد که پیشنهاد می‌شود برای حل مسائل مختلف با الگوریتم خفاش از $\alpha = \gamma = ۰/۹۸$تا$۰/۹$ استفاده شود. بلندی صدا و تعداد انتشار انفجارها تنها زمانی به‌روز رسانی می‌شوند که جواب‌های جدید، بهینه محلی را بهبود بخشیده باشند، به این معنی که خفاش‌ها به سوی جواب بهینه حرکت می‌کنند.

الگوریتم ۳-۱ الگوریتم خفاش

ورودی: تابع برازندگی ، فضای جواب S
خروجی: جواب بهبود یافته X^*

۱: جمعیت اولیّه خفاش‌ها x_i ، $i = ۱, ..., n$ را ایجاد کنید.
۲: به ازای $i = ۱, ..., n$ سرعت v_i ، فرکانس f_i، بلندی صدای A_i و تعداد انفجارهای r_i را ایجاد کنید.
۳: بهترین خفاش x^* را از میان x_iها انتخاب کنید.
۴: قرار دهید: $t := ۱$.
۵: با توجه به جمعیت خفاش‌ها گروهی از بهترین خفاش‌ها را انتخاب کنید.
۶: محل شکار هر خفاش را با توجه به معادلات (۳-۱) ، (۳-۲) و (۳-۳) مشخص کنید.
۷: به صورت تصادفی یک عدد در بازه صفر و یک انتخاب کنید. حال اگر این عدد از r_i بزرگ‌تر بود آن‌گاه یک جواب از میان گروه بهترین جواب‌ها با گام تصادفی انتخاب کرده و با استفاده از معادله (۳-۴) محل شکار را در اطراف بهترین جواب انتخاب شده، ایجاد کنید.
۸: با توجه به تابع برازندگی، ارزش محل شکار را محاسبه کنید.
۹: یک عدد تصادفی در بازه صفر و یک ایجاد کنید اگر این عدد از A_i کوچک‌تر بود و یا اگر ارزش محل شکار از محل کنونی خفاش بهتر بود خفاش به سمت شکار پرواز می‌کند.
۱۰: اگر خفاش محل بهتری نسبت به بهترین جواب حال حاضر پیدا کرد، x^* را به‌روز رسانی کنید. و طبق معادلات (۳-۵) و (۳-۶) ، r_i را افزایش و A_i را کاهش دهید.
۱۱: اگر t (حداکثر تعداد تکرارهای مجاز) به ماکزیمم مقدار خود رسیده است الگوریتم را متوقف کنید در غیر این صورت قرار دهید $t = t + ۱$ و به گام ۵ بروید.

الگوریتم خفاش مزایای زیادی دارد. یکی از مزایای کلیدی آن توانایی همگرایی سریع در مراحل ابتدایی به وسیله رفتن ناگهانی از حالت اکتشاف به استخراج است. این ویژگی باعث می‌شود این الگوریتم، یک الگوریتم تأثیر گذار برای کاربردهایی مثل دسته‌بندی و در کل زمانی که یک راه حل سریع نیاز است؛ باشد. هر چند که با اجازه دادن به الگوریتم برای رفتن سریع به مرحله استخراج، با تغییر سریع A و r ، در مراحل ابتدایی الگوریتم دچار رکود می‌شود. به منظور بهبود کارایی الگوریتم، روش‌ها و راهبردهای زیادی مطرح شده‌اند تا تنوع راه‌حل را افزایش دهند که منجر به تولید نسخه‌های مختلفی از الگوریتم خفاش شده است. برای مثال در سال ۲۰۱۲ زانگ و وانگ [۱] [۴۵] از جهش برای افزایش تنوع راه‌حل‌های مناسب استفاده کردند و این روش را در زمینه‌ی تطابق تصویر بکار گرفتند. در سال ۲۰۱۳ وانگ و جاوو[۲] [۴۶] الگوریتم خفاش را با الگوریتم هارمونی ترکیب کرده و گونه‌ای جدید با عنوان الگوریتم خفاش هیبریدی را به وجود آوردند. از طرف دیگر فیستر[۳] و همکاران [۴۷] در سال ۲۰۱۳با استفاده از الگوریتم تکامل تفاضلی[۴] به عنوان بخش جستجوی محلی الگوریتم خفاش، یک فوق الگوریتم خفاش[۵] را ایجاد نمودند.

۳-۲-۳ کاربردهای الگوریتم خفاش

الگوریتم استاندارد خفاش و تعدد مدل‌های آن نشان‌گر محدوده‌ی وسیع کاربردهای آن است. در واقع از سال ۲۰۱۰ که یانگ این الگوریتم را مطرح کرد، الگوریتم خفاش تقریباً در تمامی زمینه‌های زیر به کار گرفته شده است:

- بهینه‌سازی
- پردازش تصویر
- داده‌کاوی
- زمان‌بندی

در زمینه بهینه‌سازی الگوریتم خفاش در هر دو فضای پیوسته و گسسته کاربرد دارد. در ادامه این بخش به برخی از مطالعات انجام شده در این زمینه می‌پردازیم.

۳-۲-۳-۱ الگوریتم خفاش و بهینه‌سازی پیوسته

در میان اولین مطالعات انجام شده بر روی کاربردهای الگوریتم خفاش می‌توان از بهینه‌سازی پیوسته در زمینه بهینه‌سازی مهندسی طراحی[۶] نام برد. الگورتیم خفاش در این زمینه توانسته به طور مؤثر با مسائل بزرگ غیرخطی آن رو به رو شود و جواب بهینه را عیناً بیابد. برای مشاهده چگونگی اجرای الگوریتم خفاش، در حل این گونه مسائل، یانگ و گندمی[۷] [۴۸]

---

[۱] Zhang and Wang [۲] Guo [۳] Fister [۴] Differential evolution algorithm

[۵] Hybrid bat algorithm [۶] Engineering design optimisation [۷] Gandomi

در سال ۲۰۱۲ این الگوریتم را بر روی بعضی از مسائل شناخته شده و دشوار و در عین حال متنوع مهندسی طراحی، تست کرده‌اند. از میان هشت مورد مطالعاتی آن‌ها می‌توان به مورد زیر اشاره کرد:

مسأله هیمل‌بلاو[۱]: مسأله هیمل‌بلاو یک معیار معروف برای سنجش الگوریتم‌هاست. در اصل این مسأله توسط هیمل‌بلاو در سال ۱۹۷۲ معرفی شد و به عنوان یک تست معیار از مسأله بهینه‌سازی با محدودیت غیرخطی مورد استفاده قرار گرفت. در این مسأله، پنج متغییر تصمیم $X = [x_۱, x_۲, x_۳, x_۴, x_۵]$ ، شش محدودیت نامساوی غیرخطی و ده کران بالا یا پایین ساده وجود دارد.

مسأله هیمل‌بلاو را می‌توان به شرح زیر بیان کرد:

$$min : f(X) = ۵/۳۵۷۸۵۴۷x_۳^۲ + ۰/۸۳۵۶۸۹۱x_۱x_۵ + ۳۷/۲۹۳۲۳۹x_۱ - ۴۰۷۹۲/۱۴۱$$

به طوری که:

$$۰ \leq g_۱ \leq ۹۲, \quad ۹۰ \leq g_۲ \leq ۱۱۰, \quad ۲۰ \leq g_۳ \leq ۲۵$$

که در آن:

$$g_۱ = ۸۵/۳۳۴۴۰۷ + ۰/۰۰۵۶۸۵۸x_۱x_۵ + ۰/۰۰۰۶۲۶۲x_۱x_۴ - ۰/۰۰۲۲۰۵۳x_۳x_۵$$

$$g_۲ = ۸۰۵۱۲۴۹ + ۰/۰۰۷۱۳۱۷x_۲x_۵ + ۰/۰۰۲۹۹۵۵x_۱x_۲ - ۰/۰۰۱۹۰۸۵x_۳^۲$$

$$g_۳ = ۹/۳۰۰۹۶۱ + ۰/۰۰۴۷۰۲۶x_۳x_۵ + ۰/۰۰۱۲۵۴۷x_۱x_۳ - ۰/۰۰۱۹۰۸۵x_۳x_۴$$

با کران‌های ساده:

$$۷۸ \leq x_۱ \leq ۱۰۲, \quad ۳۳ \leq x_۲ \leq ۴۵, \quad ۲۷ \leq x_۳, x_۴, x_۵ \leq ۴۵$$

این مسأله اولین بار در سال ۱۹۷۲ توسط هیمل‌بلاو و با استفاده از روش گرادیان تعمیم‌یافته حل شده است. از آن پس، این مسأله با استفاده از چندین روش مختلف مانند الگوریتم ژنتیک توسط همایفر[۲] [۴۹] در سال ۱۹۹۴ و روش ازدحام ذرات توسط هی[۳] و همکاران [۵۰] در سال ۲۰۰۴ حل شده است. نتایج بدست آمده توسط آن‌ها و همچنین نتایج حاصل از اجرای الگوریتم خفاش توسط یانگ و گندمی در جدول ۳-۱ آورده شده است؛ که نشان‌دهنده عملکرد خوب الگوریتم خفاش در کنار دیگر الگوریتم‌های معروف است. در حل این مسأله جمعیت خفاش‌ها ۱۵ عدد در نظر گرفته شده است و میانگین مدت زمان اجرای الگوریتم خفاش در ۱۵۰۰۰ بار اجرای برنامه برابر با ۲.۷۶۲۲۴ ثانیه می‌باشد.

در سال ۲۰۱۱ تسایی[۴] و همکاران [۵۱] با پیشنهاد یک الگوی جدید محاسباتی برای الگوریتم خفاش آن را در حل مسائل عددی بکار گرفتن و نتایج آن را با الگوریتم خفاش استاندارد مقایسه نمودند. مدل پیشنهادی آن‌ها یعنی الگوریتم خفاش ارتقاء یافته[۵] (EBA)، بر اساس استفاده از ساختار کلی الگوریتم خفاش و تجدید نظر در متغیرهای بکار رفته در الگوریتم خفاش استاندارد، می‌باشد. در EBA نه تنها حرکت خفاش‌ها کاملاً متفاوت از BA استاندارد است، بلکه فرایند

---

[۱] Himmelblau's problem [۲] Homaifar [۳] He [۴] Tsai [۵] Evolved Bat Algorithm

جدول ۳-۱: نتایج آماری برای مسأله هیمل‌بلاو

متغیرهای تصمیم	PSO	BA	GA
$x_۱$	۷۸.۰۰۰۰	۷۸	۸۰.۶۱
$x_۲$	۳۳.۰۰۰۰	۳۳	۳۴.۲۱
$x_۳$	۲۹.۹۹۵۲	۲۹.۹۹۵۵	۳۱.۳۴
$x_۴$	۴۵.۰۰۰۰	۴۵	۴۲.۰۵
$x_۵$	۳۶.۷۷۵۸	۳۶.۷۷۵۲	۳۴.۸۵
$g_۱$	۹۲.۰۰۰۰	۹۱.۹۹۹۹۰	۹۰.۵۸۶
$g_۲$	۹۸.۸۴۰۵	۹۸.۸۴۰۳۹	صدق می‌کند
$g_۳$	۲۰.۰۰۰۰	۲۰.۰۰۰۰۱	۲۰.۱۲۲۳
$f(x)$	-۳۰۶۶۵.۵۳۹	-۳۰۶۶۵.۴۹۲۲	-۳۰۱۷۵.۸۰۴

پرواز تصادفی در اطراف بهینه محلی نیز به شیوه دیگر تعریف می‌شود.

الگوریتم خفاش ارتقاء یافته: در سیستم ردیابی صوتی، فاصله بین منبع موج صوتی و هدف، از رابطه زیر محاسبه می‌شود.

$$D = \frac{v.\Delta T}{۲} \tag{۹-۳}$$

که در آن D به عنوان فاصله در نظر گرفته می‌شود، v سرعت صوت و ΔT به معنای اختلاف زمان بین ارسال امواج صوتی و بازگشت بازتاب آن‌ها پس از برخورد به مانع است. در تعریف سرعت صوت با توجه به این‌که سرعت حرکت صوت در هوا ۳۴۰ متر بر ثانیه است؛ تسایی و همکاران از سرعت صوت در هوا برای مقدار v استفاده کرده و همچنین واحد v را از متر بر ثانیه به کیلومتر بر ثانیه تبدیل نموده‌اند. بنابراین معادله (۳-۹) را می‌توان به صورت زیر بیان نمود:

$$D = ۱۷۰.\Delta T(\text{متر بر ثانیه}) = ۰/۱۷.\Delta T(\text{کیلومتر بر ثانیه}) \tag{۱۰-۳}$$

آن‌ها از یک عدد تصادفی در بازه‌ی $[-۱, ۱]$ برای مقدار دهی ΔT بهره برده‌اند. زمانی که ΔT منفی انتخاب شود، جهت انتقال موج صدا مخالف محور مختصات است.چگونگی حرکت خفاش در EBA به صورت زیر تعریف می‌شود:

$$x_i^t = x_i^{t-۱} + D \tag{۱۱-۳}$$

از طرفی پرواز تصادفی یک خفاش در اطراف بهینه‌ی محلی به صورت زیر می‌باشد:

$$x_i^{tR} = \beta.(x_{best} - x_i^t) \tag{۱۲-۳}$$

که در آن β یک عدد تصادفی در بازه $[۰, ۱]$ و x^{tR} مکان جدید خفاش پس از فرایند پرواز تصادفی در اطراف بهینه محلی است.

الگوریتم ۳-۲ الگوریتم خفاش ارتقاء یافته

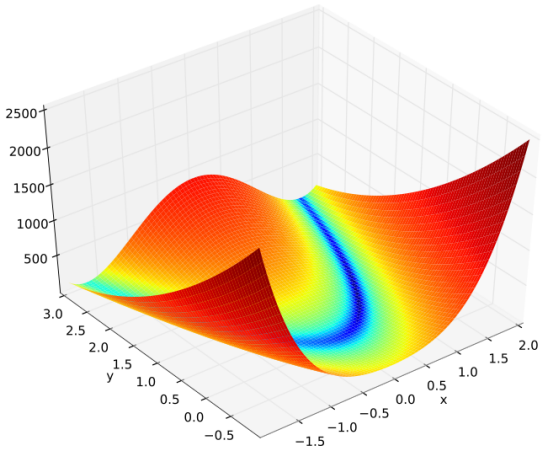
ورودی: تابع برازندگی ، فضای جواب S
خروجی: جواب بهبود یافته X^*

۱: مقدار دهی اولیه: به طور تصادفی خفاش‌ها را در فضای جواب به پرواز درآورید. بهترین خفاش را شناسایی کنید.
۲: خفاش‌ها را به وسیله معادلات (۳-۱۰) و (۳-۱۱) حرکت دهید. یک عدد تصادفی انتخاب کنید. اگر این عدد از تعداد انفجارهای مجاز خفاش بیش‌تر بود، خفاش را با استفاده از فرایند پرواز تصادفی که با معادله (۳-۱۲) تعریف شده است حرکت دهید.
۳: برازندگی خفاش‌ها را محاسبه کنید و بهترین جواب را به‌روز رسانی کنید.
۴: شرایط خاتمه الگوریتم را بررسی کنید در صورت عدم توفیق در رسیدن به آن‌ها به گام ۲ بروید.

تسایی و همکاران برای تجزیه و تحلیل دقت و سرعت محاسبات روش پیشنهادی خود، سه تابع آزمون، که در معادلات (۳-۱۳) تا (۳-۱۵) ذکر شده‌اند را در آزمایشات مورد استفاده قرار دادند. نتایج در مقایسه با BA استاندارد است و هدف بهینه‌سازی، در تمام توابع تست، بدست آوردن کمینه توابع می باشد.

- **تابع روزن‌بروک** [۱] :

$$f_۱(X) = \sum_{i=۱}^{M-۱} [۱۰۰.(x_{i+۱} - x_i^۲)^۲ + (x_i - ۱)^۲] \qquad (۱۳-۳)$$

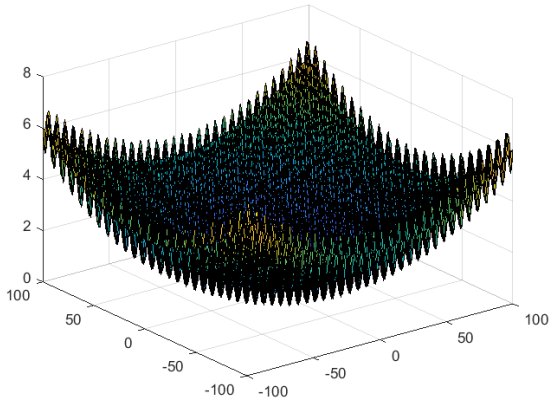


شکل ۳-۴: تابع روزن‌بروک با دو متغیر

[۱] Rosenbrock function

- **تابع گریونک [۱] :**

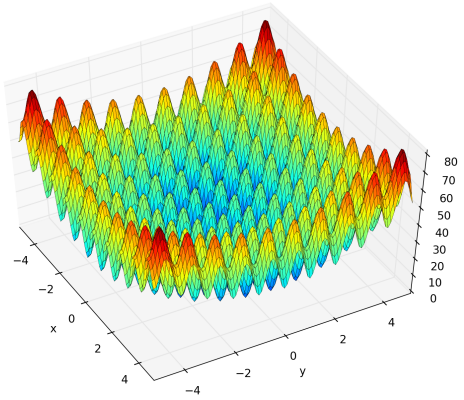
$$f_۲(X) = \frac{۱}{۴۰۰۰}\sum_{i=۱}^{M} x_i^۲ - \prod_{i=۱}^{M}\cos(\frac{x_i}{\sqrt{i}}) + ۱ \qquad (۱۴\text{-}۳)$$



شکل ۳-۵: تابع گریونک با دو متغیر

- **تابع راستریجن [۲] :**

$$f_۳(X) = \sum_{i=۱}^{M}[x_i^۲ - ۱۰.\cos(۲\pi x_i) + ۱۰] \qquad (۱۵\text{-}۳)$$



شکل ۳-۶: تابع راستریجن با دو متغیر

جمعیت خفاش‌ها در هر دو الگوریتم برابر با ۲۰ در نظر گرفته شده است و تعداد متغیرها برابر با ۳۰ به عبارتی $M = ۳۰$.

[۱] Griewank function [۲] Rastrigin function

هر الگوریتم برای هر تابع ۲۵ بار اجرا شده است. در جدول‌های ۳-۲، ۳-۳ و ۳-۴ مشخصات تابع‌های تست، جزئیات الگوریتم‌ها و نتایج حاصله از اجرای آن‌ها شامل زمان اجرای برنامه و میانگین مقدار تابع هدف در ۲۵ بار اجرای الگوریتم‌ها با عنوان میانگین خروجی آورده شده است.

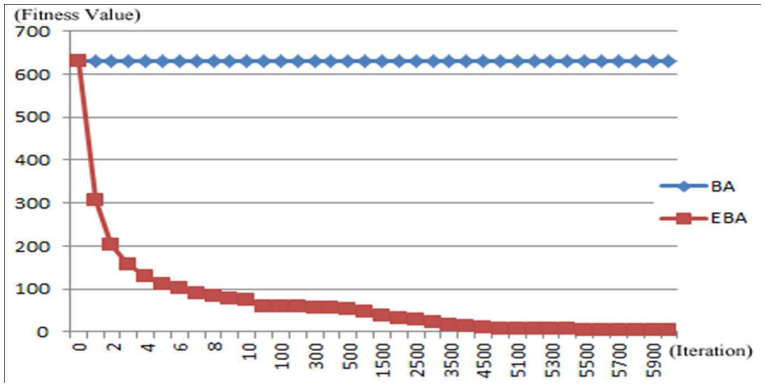
جدول ۳-۲: بازه اولیه و حداکثر تعداد تکرار برای هر تابع تست

تابع	بازه اولیه	حداکثر تکرار در هر بار اجرای آزمون
$f_۱$	$x \in [-۵۰, ۵۰]$	۵۰۰۰
$f_۲$	$x \in [-۶۰۰, ۶۰۰]$	۶۰۰۰
$f_۳$	$x \in [-۵/۱۲, ۵/۱۲]$	۵۰۰۰

جدول ۳-۳: متغیرهای الگوریتم خفاش و الگوریتم خفاش ارتقاء یافته

الگوریتم خفاش		الگوریتم خفاش ارتقاء یافته	
مقدار اولیه r_i	[۰،۱]	r ثابت	۰.۵
مقدار اولیه A_i	[۱،۲]	ΔT	[-۱ ،۱]
$[f_{min}, f_{max}]$	[۰،۱۰۰]		
ϵ	[-۱ ،۱]		
γ و α	۰.۹		

شکل ۳-۷: نتایج تجربی از اجرای الگوریتم‌ها بر روی تابع تست گریونگ



همان‌طور که در شکل ملاحظه می‌شود مقدار تابع گریونگ حاصل اجرای الگوریتم خفاش ارتقاء یافته تنها بعد از ۱۰ تکرار به مقدار قابل ملاحظه‌ای کاهش یافته است و بعد از ۵۵۰۰ تکرار به مقدار بهینه تابع دست یافته است و این در حالی است که الگوریتم خفاش استاندارد در یافتن جواب ناتوان است.

از نتایج بدست آمده مشخص است که در حل مسائل پیوسته الگوریتم خفاش ارتقاء یافته عملکردی بهتر و سریع‌تری نسبت به الگوریتم خفاش استاندارد دارد. به طوری که میانگین عملکرد الگوریتم خفاش ارتقاء یافته نسبت به الگوریتم خفاش استاندارد ۹۹.۴۲ درصد بهبود یافته و میانگین زمان اجرای الگوریتم ۶.۰۷ درصد کاهش یافته است.

جدول ۳-۴: میانگین خروجی و زمان اجرای برنامه

	الگوریتم خفاش ارتقاء یافته		الگوریتم خفاش	
	زمان	میانگین خروجی	زمان	میانگین خروجی
$f_1(x)$	۲.۵۸۹	$۲/۲۶۶ \times ۱۰^{۲}$	۲.۶۳۲	$۱/۸۶۴ \times ۱۰^{۹}$
$f_2(x)$	۳.۹۷۷	$۳/۳۶۴ \times ۱۰^{۰}$	۴.۳۹۲	$۱/۸۶۴ \times ۱۰^{۲}$
$f_3(x)$	۲.۷۶۴	$۲/۲۶۶ \times ۱۰^{۱}$	۲.۹۷۶	$۱/۸۶۴ \times ۱۰^{۲}$

۳-۲-۳-۲ الگوریتم خفاش و بهینه‌سازی گسسته

در سال ۲۰۱۶ انیکو اوسابا[۱] و همکارانش [۵۲] تغییراتی را در الگوریتم خفاش ایجاد و آن را سازگار با حل مسائلی از نوع جایگشتی نمودند سپس نتایج حاصل از آن را با الگوریتم‌های مختلف از جمله الگوریتم ژنتیک مقایسه کردند.

مسأله فروشنده دوره‌گرد: مسائل فروشنده دوره‌گرد متقارن[۲] و نامتقارن[۳] دو مورد از شناخته شده‌ترین و گسترده‌ترین مسائل علوم کامپیوتر و تحقیق در عملیات در طول تاریخ هستند. همانند خیلی از مسائل بهینه‌سازی ترکیبیاتی که در گروه مسائل NP- سخت قرار دارند، این مسائل نیز جزء مسائل NP- سخت به‌شمار می‌آیند. از این رو در بسیاری از کارهای تحقیقاتی بر روی مسائل گسسته به عنوان معیار سنجش مورد استفاده قرار می‌گیرند.

مسأله فروشنده دوره‌گرد را می‌توان به عنوان یک گراف کامل $G = (V, A)$ تعریف نمود، که در آن $V = \{v_1, v_2, ..., v_n\}$ مجموعه‌ای از رأس‌ها است که نشان دهنده‌ی شهرها و یا مکان‌هاست و $A = \{(v_i, v_j) : v_i, v_j \in V, i \neq j\}$ مجموعه‌ای از یال‌هایی است که نشان دهنده مسیرهای موجود بین شهرها و یا مکان‌ها می‌باشد. از طرف دیگر، به هر یال هزینه c_{ij} اختصاص یافته است. در مسأله متقارن فروشنده دوره‌گرد، هزینه رفت و برگشت بین دو نقطه در هر دو جهت یکسان است به عبارتی $c_{ij} = c_{ji}$. در حالی که در نوع نامتقارن این مسأله، هرچند که ممکن است در مکان‌هایی $c_{ij} = c_{ji}$ باشد اما به طور کلی $c_{ij} \neq c_{ji}$ است.

هدف از مسائل متقارن و نامتقارن فروشنده دوره‌گرد پیدا کردن یک تور با کمترین هزینه است، به طوری که از یک شهر یا نقطه شروع شود و پس از عبور از تمامی نقاط در نقطه شروع خاتمه یابد. البته این تور تنها مجاز به یک بار عبور از هر نقطه است. مثالی از فروشنده دورگرد با یک جواب شدنی $X = \{۱, ۵, ۲, ۷, ۰, ۴, ۹, ۸, ۶, ۳\}$ در شکل ۳-۸ آورده شده است.

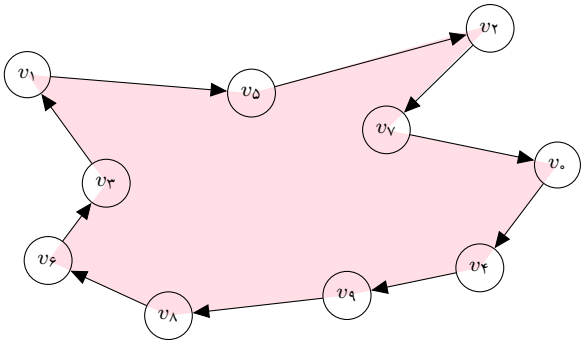
الگوریتم خفاش برای حل مسأله فروشنده دوره‌گرد: در مسئله‌ی فروشنده دوره گرد جواب‌ها به صورت جایگشتی از اعداد متوالی هستند که تعداد این اعداد برابر با تعداد شهرها می‌باشد.

مثال ۳-۲-۱. برای یک مسأله با ده شهر دو جواب زیر شدنی هستند:

$$X_1 = [۲, ۵, ۹, ۱, ۴, ۳, ۷, ۸, ۶, ۱۰]$$

$$X_2 = [۸, ۵, ۹, ۳, ۴, ۲, ۷, ۱, ۶, ۱۰]$$

[۱] Eneko Osaba [۲] Symmetric TSP [۳] Asymmetric TSP



شکل ۳-۸: یک مثال از مسأله فروشنده دوره‌گرد با ده گره به همراه یک جواب شدنی

در الگوریتم گسسته‌سازی شده خفاش سرعت خفاش‌ها با توجه به فرمول زیر به‌روز رسانی می‌شود:

$$v_i^t = random[۱, HammingDistanc(x_i^t, x^*)] \qquad (۱۶-۳)$$

که در آن Hamming Distanc (فاصله همینگ)، تعداد اختلاف‌های بین خفاش حال حاضر با بهترین خفاش است.

تعریف ۳-۲-۲. در تئوری اطلاعات فاصله‌ی همینگ برای دو رشته با طول مساوی، برابر تعداد مکان‌هایی است که نمادهای متناظر متفاوت هستند. به عبارت دیگر، کمترین تعداد جایگزینی‌هایی است که یک رشته به یک رشته دیگر تغییر پیدا کند، یا تعداد خطاهایی که یک رشته به رشته دیگر تبدیل گردد.

مثال ۳-۲-۳. در مثال ۳-۲-۱ فاصله همینگ بین دو جواب

$$X_۱ = [\underline{۲}, ۵, ۹, \underline{۱}, ۴, \underline{۳}, ۷, \underline{۸}, ۶, ۱۰]$$

و

$$X_۲ = [\underline{۸}, ۵, ۹, \underline{۳}, ۴, \underline{۲}, ۷, \underline{۱}, ۶, ۱۰]$$

برابر با ۴ است.

در الگوریتم گسسته‌سازی شده خفاش عملاً فرکانس هر خفاش کارایی خود را از دست می‌دهد، بنابراین از چرخه حذف می‌شود. در الگوریتم گسسته‌سازی شده خفاش از دو روش جایگزین زیر جهت حرکت دادن خفاش‌ها در فضای جواب استفاده می‌شود.

- **opt _ ۲:** این تابع در سال ۱۹۶۵ توسط لین[۱] تعریف شده است و کاربرد وسیعی در حل مسائل مسیریابی دارد. این تابع با حذف تصادفی دو یال از مسیر و ایجاد دو یال جدید در آن سعی در حذف زیر تورهای مسیر دارد.

[۱] Lin

- **opt _ ۳:** این عامل که نیز توسط لین پیشنهاد شده است، همانند $opt_$۲ عمل می‌کند با این تفاوت که در این مورد ۳ یال از مسیر حذف و ۳ یال جدید به مسیر اضافه می‌شود. بنابراین تغییرات بیشتری در مسیر ایجاد می‌شود.

با توجه به دو عامل حرکت خفاش‌ها جهت به‌روز رسانی موقعیت هر خفاش، با در نظر گرفتن سرعت خفاش یکی از دو روش زیر انتخاب می‌شود:

- جابه‌جایی دو آرایه از جواب با یکدیگر

$$x_i^t \longleftarrow ۲\_opt(x_i^{t-۱}, v_i^t)$$

- جابه‌جایی سه آرایه از جواب با یکدیگر

$$x_i^t \longleftarrow ۳\_opt(x_i^{t-۱}, v_i^t)$$

همانطور که گفته شد هر یک از این دو روش با توجه به سرعت خفاش انتخاب می‌شوند، اگر $v_i^t < \frac{\text{تعداد شهرها}}{۲}$ باشد از روش اول در غیر این صورت از روش دوم استفاده می‌شود.

مثال ۳-۲-۴. در مثال ۳-۲-۱ در صورتی که $X_۱$ موقعیت بهترین خفاش باشد و $X_۲$ موقعیت خفاش i در تکرار t-۱ باشد. سرعت خفاش در مرحله t با توجه به فاصله همینگ خفاش با بهترین خفاش، یک عدد صحیح بین ۱ تا ۴ است که در هر صورت کمتر از تعداد نیمی از شهرهاست. بنابراین جهت محاسبه موقعیت خفاش i در تکرار t از جابه جایی دو آرایه از جواب که به صورت تصادفی انتخاب می‌شوند؛ استفاده می‌کنیم.

$$v_i^t = random[۱, ۴]$$
$$v_i^t < \frac{۱۰}{۲}$$

از این رو

$$X_۲ = [\underline{۹}, ۵, \underline{۸}, ۳, ۴, ۲, ۷, ۱, ۶, ۱۰]$$

می‌تواند موقعیت جدید خفاش در تکرار t باشد.

سرانجام، در ارتباط با فرایند پرواز تصادفی در اطراف بهینه محلی در الگوریتم خفاش استاندارد، در الگوریتم گسسته‌سازی شده خفاش نیز اگر عدد تصادفی انتخاب شده بزرگتر از r_i باشد یک جواب در اطراف بهترین جواب حال حاضر انتخاب می‌شود. در این الگوریتم جهت انتخاب یک همسایه در اطراف بهینه محلی از حرکت‌های opt _ ۲ و opt _ ۳ استفاده می‌شود.

الگوریتم ۳-۳ الگوریتم گسسته‌سازی شده خفاش برای حل مسأله فروشنده دوره‌گرد

ورودی: تابع برازندگی ، فضای جواب S
خروجی: جواب بهبود یافته X^*

۱: جمعیت اولیّه خفاش‌ها x_i ، $i = ۱, ..., n$ را ایجاد کنید.
۲: به ازای $i = ۱, ..., n$ سرعت $v_i = ۰$ ، بلندی صدای A_i و تعداد انفجارهای r_i را ایجاد کنید.
۳: بهترین خفاش x^* رااز میان x_iها انتخاب کنید.
۴: قرار دهید: $t := ۱$.
۵: با توجه به جمعیت خفاش‌ها گروهی از بهترین خفاش‌ها را انتخاب کنید.
۶: با توجه به معادله (۳-۱۶) سرعت هر خفاش را به‌روز رسانی کنید.
اگر $v_i^t < \frac{\text{تعداد شهرها}}{۲}$
قرار دهید: $x_i^t \longleftarrow ۲\_opt(x_i^{t-۱}, v_i^t)$
در غیر این صورت
قرار دهید: $x_i^t \longleftarrow ۳\_opt(x_i^{t-۱}, v_i^t)$
۷: به صورت تصادفی یک عدد در بازه صفر و یک انتخاب کنید. حال اگر این عدد از r_i بزرگ‌تر بود آن‌گاه یک جواب در میان بهترین جواب‌ها با گام تصادفی انتخاب کرده و با استفاده از opt_۲ یا opt_۳ محل شکار را در اطراف بهترین جواب انتخاب شده، ایجاد کنید.
۸: با توجه به تابع برازندگی، ارزش محل شکار را محاسبه کنید.
۹: یک عدد تصادفی در بازه صفر و یک ایجاد کنید اگر این عدد از A_i کوچک‌تر بود و یا اگر ارزش محل شکار از محل کنونی خفاش بهتر بود خفاش به سمت شکار پرواز می‌کند.
۱۰: اگر خفاش محل بهتری نسبت به بهترین جواب حال حاضر پیدا کرد، x^* را به‌روز رسانی کنید. و طبق معادلات (۳-۵) و (۳-۶) ، r_i را افزایش و A_i را کاهش دهید.
۱۱: اگر t به ماکزیمم مقدار خود رسیده است الگوریتم را متوقف کنید در غیر این صورت قرار دهید $t = t + ۱$ و به گام ۵ بروید.

اوسابا و همکاران با انجام آزمایشاتی به این نتیجه رسیدن که استفاده هم زمان از دو عامل حرکت ذکر شده در الگوریتم خفاش نتایج بهتری نسبت به بکارگیری یک مورد دارد. همچنین آن‌ها با مقایسه الگوریتم پیشنهادی خود با الگوریتم ژنتیک، تبرید شبیه‌سازی شده تکاملی و چند روش دیگر نشان دادند که روش آن‌ها نتایج بهتری در حل آزمون‌های شناخته شده مسائل فروشنده دوره‌گرد ارائه می‌دهد. البته در مورد زمان اجرا در برخی موارد نسبت به الگوریتم ژنتیک کمی کندتر عمل می‌کند؛ هرچند که اختلاف زمان بسیار ناچیز است. البته با توجه به این‌که مقایسه آن‌ها تنها بر روی حل مسائلی از نوع فروشنده دوره‌گرد بوده است نمی‌توان نتایج کار آن‌ها و برتر بودن الگوریتم پیشنهادیشان را به دیگر مسائل گسسته تعمیم داد.

الگوریتم خفاش برای حل مسأله‌ی گسسته از نوع صفر و یک:

در سال ۲۰۱۲ ناکامورا[۱] و همکاران [۵۳] برای اولین بار الگوریتم خفاش را برای حل مسائل صفر و یک مورد استفاده قرار دادند. در مسائل صفر و یک هر خفاش دارای یک بردار سرعت و یک بردار موقعیت است. تعداد مولفه‌های این بردارها برابر با تعداد متغیرهای صفر ویک مسأله مورد نظر می‌باشد. در روش پیشنهادی ناکامورا ابتدا با استفاده از تابع سیگموئید، سرعت خفاش را به بازه [۰, ۱] برده شده و سپس با استفاده ازمعادله (۳-۱۸) موقعیت هر خفاش به‌روز رسانی می‌شود.

$$S(v_i^t) = \frac{۱}{۱ + e^{-v_i^t}} \tag{۳-۱۷}$$

$$x_i^t = \begin{cases} ۱ & S(v_i^t) > \sigma \text{ اگر} \\ ۰ & \text{در غیر این صورت} \end{cases} \tag{۳-۱۸}$$

که در آن σ یک عدد تصادفی در بازه [۰, ۱] است. بنابراین معادله (۳-۱۸) تنها قادر خواهد بود که برای هر خفاش، مقادیر صفر و یک تولید کند.

در سال ۲۰۱۳ میرجلیلی[۲] و همکاران [۵۴] جهت تولید موقعیت صفر و یک خفاش‌ها، استفاده از تابع انتقال v-شکل را پیشنهاد دادند. آنان در روش پیشنهادی خود با استفاده از معادلات (۳-۱۹) و (۳-۲۰) موقعیت هر خفاش را به‌روز رسانی می‌کنند.

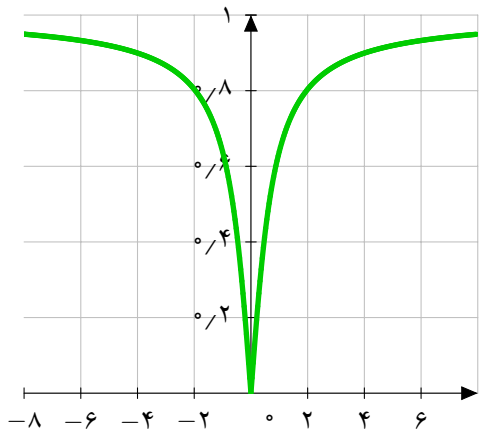
$$V(v_i^j(t)) = \left| \frac{۲}{\pi} \arctan(\frac{\pi}{۲} v_i^j(t)) \right| \tag{۳-۱۹}$$

$$x_i^j(t) = \begin{cases} (x_i^j(t-۱))^{-۱} & V(v_i^j(t)) > \sigma \text{ اگر} \\ x_i^j(t-۱) & \text{در غیر این صورت} \end{cases} \tag{۳-۲۰}$$

که در آن $x_i^j(t)$ و $v_i^j(t)$ به ترتیب نشان‌دهنده موقعیت و سرعت iامین خفاش در تکرار t و در jامین مولفه بردار موقعیت و یا سرعت است. همچنین $(x_i^j(t-۱))^{-۱}$ مکمل یک $x_i^j(t-۱)$ و σ یک عدد تصادفی در بازه [۰, ۱] می‌باشد.

در مورد پرواز تصادفی در اطراف بهترین خفاش روش پیشنهادی ناکامورا ابتدا موقعیت خفاش را با استفاده از معادله (۳-۴) به‌روز رسانی می‌کند سپس با استفاده از تابع سیگوئید مولفه‌های موقعیت خفاش را به بازه [۰, ۱] برده و با مقایسه آنان با به یک عدد تصادفی در بازه [۰, ۱] به مولفه‌های خفاش مقدار یک یا صفر را نسبت می‌دهد و اما در روش پیشنهادی میرجلیلی برای هر مولفه موقعیت یک عدد تصادفی در بازه [۰, ۱] انتخاب می‌شود در صورتی که این عدد از مقدار r_i بیشتر باشد مقدار جدید برای آن مولفه برابر با مقدار همان مولفه در بردار موقعیت بهترین خفاش است. در پیوست آ کد متلب

[۱] Nakamura [۲] Mirjalili



شکل ۳-۹: تابع انتقال v- شکل

آ-۳ برنامه صفر و یک میرجلیلی و کد متلب آ-۴ برنامه صفر و یک ناکامورا می‌باشد؛ که جهت درک بهتر این دو الگوریتم می‌توان به آن‌ها مراجعه کرد.

در فصل چهارم با استفاده از سه روش گسسته الگوریتم خفاش، مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود را حل خواهیم نمود.

فصل ۴

مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود و الگوریتم خفاش

الگوریتم خفاش از سال ۲۰۱۰ که اولین بار توسط یانگ مطرح شد تا کنون دستخوش تغییرات زیادی شده است. پژوهش‌گران زیادی از آن در حل مسائل گوناگون در فضای جواب پیوسته و گسسته بهره برده‌اند. ما نیز در نظر داریم از نوع صفر و یک این الگوریتم جهت حل مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود بهره ببریم و نتایج حاصله را با الگوریتم‌های مطرحی همچون الگوریتم ژنتیک و الگوریتم بهینه‌سازی ازدحام ذرات مقایسه کنیم. مسائل تست شامل ۹ مورد از مسائل استاندارد مکان‌یابی تسهیلات با ظرفیت نامحدود موجود در OR-library می‌باشند.

۴-۱ معرفی مسائل تست مکان‌یابی تسهیلات با ظرفیت نامحدود

در جدول ۴-۱ جزئیات ۹ مسأله آزمایشی آورده شده است. در این جدول اندازه مسأله نشان‌دهنده تعداد مکان‌های بالقوه جهت تأسیس تسهیلات و تعداد مشتریانی است که باید نیاز آن‌ها توسط تسهیلات باز تأمین شود که با توجه به اطلاعات جدول تعداد مشتریان در این ۹ نمونه یکسان و برابر با ۵۰ در نظر گرفته شده است. علاوه بر این هزینه جواب بهینه هر مسأله نیز در این جدول گنجانده شده است که نشان‌دهنده مقدار کمینه هر مسأله است. این هزینه شامل هزینه بازگشایی تسهیلات باضافه هزینه تخصیص مشتریان است.

جدول ۴-۱: جزئیات مسائل تست مکان‌یابی تسهیلات با ظرفیت نامحدود

شماره مسأله	نام مسأله	اندازه مسأله	هزینه جواب بهینه
۱	Cap71	۱۶ × ۵۰	۹۳۲،۶۱۵.۷۵
۲	Cap72	۱۶ × ۵۰	۹۷۷،۷۹۹.۴۰
۳	Cap73	۱۶ × ۵۰	۱،۰۱۰،۶۴۱.۹۸
۴	Cap101	۲۵ × ۵۰	۷۹۶،۶۴۸.۴۴
۵	Cap102	۲۵ × ۵۰	۸۵۴،۷۰۴.۲۰
۶	Cap103	۲۵ × ۵۰	۸۹۳،۷۸۲.۱۱
۷	Cap131	۵۰ × ۵۰	۷۹۳،۴۳۹.۵۶
۸	Cap132	۵۰ × ۵۰	۸۵۱،۴۹۵.۳۳
۹	Cap133	۵۰ × ۵۰	۸۹۳،۰۷۶.۷۱

۴-۲ الگوریتم خفاش برای حل مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود

از آن جایی که مسأله مکان‌یابی که ما با آن سر و کار داریم از نوع صفر و یک می‌باشد ما در این فصل سه روش گسسته خفاش که در فصل سه بیان نمودیم را بر روی مسائل تست مکان‌یابی تسهیلات با ظرفیت نامحدود، آزمایش و با هم مقایسه می‌کنیم. این سه روش عبارتند از:

۱. **روش صفر و یک برگرفته از روش جایگشتی اوسابا (BBAO)** [۵۲]

۲. **روش صفر و یک ناکامورا (BBAN)** [۵۳]

۳. **روش صفر و یک میرجلیلی (BBAM)** [۵۴]

جدول ۴-۲: متغیرهای الگوریتم خفاش در سه روش صفر و یک اوباسا، ناکامورا و میرجلیلی

متغیرها	BBAO	BBAN	BBAM
n	۵۰	۵۰	۵۰
r_i	۰.۶	۰.۶	۰.۶
A_i	۰.۹	۰/۹	۰.۹
$[f_{min}, f_{max}]$	-	[-۰.۰۸ ،۰.۰۸]	[۰ ،۲]
ϵ	-	[-۱ ،۱]	-
$\alpha\gamma$	۰.۹	۰.۹	۰.۹
حداکثر تعداد تکرارها	۱۰۰	۱۰۰	۱۰۰

در جدول ۴-۲ جزئیات هر کدام از روش‌ها آورده شده است. که در آن n تعداد خفاش‌ها، A_i مقدار اولیه بلندی صدای هر خفاش، r_i مقدار اولیه تعداد انفجار هر خفاش، v_i سرعت اولیه هر خفاش و f فرکانس هر خفاش می‌باشد.

در پیوست آ کد برنامه‌های هر کدام از روش‌های بالا آورده شده است.

جهت انجام آزمایشات بر روی روش‌های مورد مطالعه، با استفاده از رایانه شخصی با پردازنده Core-i3-3110M، 2.4 GHz CPU، 3GB RAM ، ویندوز ۷ (64 bits) و متلب ۲۰۱۵ آن‌ها را ۱۰۰ بار بر روی مسائل تست اجرا نموده و نتایج را از نظرات مختلف با هم مقایسه کرده‌ایم. شرایط خاتمه برای تمامی روش‌ها یکسان و شامل موارد زیر می‌باشد:

۱. **حداکثر تعداد تکرار:** در این شرط تمامی الگوریتم‌ها تنها مجاز به انجام ۱۰۰ تکرار در هر بار اجرا می‌باشند و بعد از رسیدن به حداکثر تعداد تکرار بدون در نظر گرفتن مقدار بهینگی بدست آمده باید از حلقه اجرایی خود خارج شوند. البته اگر الگوریتمی بتواند در کمتر از این تعداد تکرارها به جواب بهینه سراسری برسد؛ می‌تواند قبل از رسیدن به حداکثر تعداد تکرار از برنامه خارج شود که این مسأله یک مزیت برای الگوریتم به حساب خواهد آمد.

۲. **رسیدن به بهینگی نسبی:** در این شرط رسیدن به بهینه‌سراسری ملاک نخواهد بود بلکه تنها رسیدن به یک بهینه محلی نزدیک به بهینه‌سراسری می‌تواند مجوز خروج الگوریتم از برنامه اجرایی باشد. در این جا مقدار انحراف مجاز تنها یک ده‌هزارم مقدار بهینه‌سراسری است. الگوریتمی که بتواند در زمان کوتاه‌تر و همچنین تعداد تکرارهای کمتر به این بهینه محلی برسد از نظر عملکرد برتر شناخته خواهد شد.

۳. **محدودیت زمانی:** در این شرط با گذاشتن زمان‌سنج در ابتدا و انتهای برنامه‌ی اجرایی الگوریتم و محاسبه زمان اجرای برنامه، الگوریتم را وادار می‌کنیم پس از طی زمان مشخصی از برنامه خارج شود. زمان مشخص شده بر اساس اندازه مسأله تعریف می‌شود. این مقدار برابر با یک ده‌هزارم اندازه مسأله است.

۴-۲-۱ نتایج و نتیجه گیری

با توجه به هر کدام از شروط خاتمه، عملکرد الگوریتم‌های خفاش در حل مسائل مکان‌یابی تسهیلات با ظرفیت نامحدود از نظر میانگین مقدار تابع هدف، میانگین زمان اجرایی الگوریتم و درصد رسیدن به بهینگی(تعداد دفعاتی که الگوریتم موفق شده است مقدار بهینه سراسری را بدست آورد) بررسی و نتایج تجربی حاصل از عملکرد این سه روش به همراه عملکرد الگوریتم ژنتیک (GA) بکار گرفته شده توسط شاهی [۲۷] و الگوریتم بهینه‌سازی صفر و یک ازدحام ذرات (BPSO) معرفی شده توسط شادهولت و وایت [۳۷] در جداول ۴-۳ تا ۴-۸ درج و با یکدیگر مقایسه شده‌اند.

جدول ۴-۳: جدول میانگین مقدار تابع هدف- شرط خاتمه حداکثر تعداد تکرار

شماره مسأله	روش				
	GA	BBAM	BBAN	BBAO	BPSO
۱	۹۳۲۷۸۹.۹۲۳	**۹۳۲۶۱۵.۷۵۰**	۹۳۴۴۶۲.۶۲۵	۹۳۲۸۴۱.۶۵۹	۹۳۲۷۱۰.۷۵۳
۲	۹۷۸۵۶۶.۱۴۸	**۹۷۷۸۴۸.۵۲۲**	۹۸۰۴۹۶.۸۷۳	۱۰۱۱۰۸۶.۵۳۱	۱۰۱۱۱۹۲.۱۰۸
۴	۷۹۸۰۲۸.۹۵۵	**۷۹۷۱۱۰.۵۱۷**	۸۰۴۰۷۴.۴۲۷	۷۹۸۳۶۸.۰۹۱	۷۹۷۳۸۵.۶۶۳
۵	۸۵۵۸۸۴.۰۴۱	**۸۵۴۹۶۰.۴۲۵**	۸۶۴۷۷۴.۷۹۶	۸۵۶۸۸۳.۷۵۳	۸۵۵۳۵۳.۴۸۲
۶	۸۹۴۹۱۹.۴۲۶	**۸۹۴۳۸۵.۷۹۵**	۹۰۶۲۴۱.۳۳۴	۸۹۵۲۸۲.۷۳۳	۸۹۴۴۷۰.۳۵۲
۷	۷۹۸۹۲۵.۸۷۸	۷۹۹۱۴۷.۹۹۴	۸۳۴۱۵۲.۹۸۲	۸۳۳۴۲۴.۴۶۱	**۷۹۶۵۰۸.۸۴۲**
۸	۸۵۵۴۳۲.۷۳۳	۸۵۷۲۷۲.۹۱۸	۹۰۸۵۷۸.۴۸۸	۹۱۱۴۴۰.۶۵۹	**۸۵۳۳۲۹.۴۳۶**
۹	۸۹۶۹۳۸.۲۶۸	۸۹۸۰۰۰.۷۰۴	۹۶۵۸۵۱.۶۴۴	۹۷۷۸۵۴.۸۳۳	**۸۹۴۶۶۶.۶۵۷**
میانگین	۸۹۱۴۴۹.۴۰۲	۸۹۱۳۴۷.۸۸۰	۹۱۲۳۳۳.۷۲۲	۹۱۰۵۸۹.۸۲۶	**۸۹۰۴۰۷.۶۵۹**

جدول ۴-۴: جدول میانگین زمان اجرای الگوریتم- شرط خاتمه حداکثر تعداد تکرار

شماره مسأله	روش				
	GA	BBAM	BBAN	BBAO	BPSO
۱	۰.۲۷۲	۰.۱۸۰	۰.۲۸۷	۰.۳۷۷	**۰.۱۶۸**
۲	۰.۲۷۰	۰.۱۸۷	۰.۲۹۱	۰.۳۷۸	**۰.۱۶۷**
۳	۰.۲۶۴	۰.۱۸۵	۰.۲۹۰	۰.۳۷۷	**۰.۱۶۴**
۴	۰.۳۱۵	۰.۲۰۳	۰.۳۱۲	۰.۴۴۲	**۰.۱۸۵**
۵	۰.۳۵۲	۰.۲۰۰	۰.۳۱۱	۰.۴۴۰	**۰.۱۸۳**
۶	۰.۳۰۱	۰.۲۰۰	۰.۳۱۱	۰.۴۳۸	**۰.۱۷۹**
۷	۰.۴۲۷	۰.۲۳۱	۰.۳۶۰	۰.۵۷۸	**۰.۲۲۷**
۸	۰.۵۰۴	۰.۲۲۸	۰.۳۵۶	۰.۵۷۴	**۰.۲۲۴**
۹	۰.۴۳۴	۰.۲۲۵	۰.۳۵۸	۰.۵۷۸	**۰.۲۲۲**
میانگین	۰.۳۴۹	۰.۲۰۴	۰.۳۱۹	۰.۴۶۵	**۰.۱۹۱**

با توجه به جدول‌های ۴-۳ و ۴-۴ ملاحظه می‌شود که پس از اعمال شرط خاتمه اول مقدار بهینه‌ی بدست آمده توسط الگوریتم صفر و یک خفاش میرجلیلی و الگوریتم بهینه‌سازی ازدحام ذرات در مقایسه با سایر الگوریتم‌ها بهتر است. همچنین این الگوریتم‌ها پس از صرف زمان کمتری به این نتایج مطلوب رسیده‌اند که این خود یک برتری دیگر برای آنها نسبت به سایر الگوریتم‌ها می‌باشد.

جدول ۴-۵: جدول میانگین مقدار تابع هدف- شرط خاتمه رسیدن به بهینگی نسبی

شماره مسأله	روش				
	GA	BBAM	BBAN	BBAO	BPSO
۱	۹۳۴۵۴۷.۲۹۱	**۹۳۲۶۴۷.۴۱۸**	۹۳۴۴۰۶.۴۸۶	۹۳۲۸۴۹.۲۵۰	**۹۳۲۶۴۷.۴۱۸**
۲	۹۸۰۹۴۹.۹۷۲	**۹۷۷۷۹۹.۴۰۰**	۹۸۰۵۸۷.۰۹۸	۹۷۸۲۵۱.۳۳۱	۹۷۷۹۵۳.۳۹۸
۳	۱۰۱۲۷۸۱.۰۷۳	**۱۰۱۰۷۸۸.۲۹۲**	۱۰۱۲۷۲۳.۸۶۲	۱۰۱۱۲۴۳.۶۰۷	۱۰۱۱۴۲۹.۵۴۵
۴	۸۰۳۵۹۴.۶۳۶	**۷۹۷۱۶۶.۸۷۹**	۸۰۴۷۳۱.۷۴۳	۷۹۸۳۵۷.۳۵۶	۷۹۷۲۵۱.۷۵۷
۵	۸۶۳۱۴۱.۵۷۶	**۸۵۴۸۹۷.۴۹۱**	۸۶۴۰۸۵.۱۰۶	۸۵۶۷۳۹.۰۴۵	۸۵۵۳۲۲.۴۶۴
۶	۹۰۳۹۸۳.۳۸۷	**۸۹۴۳۰۱.۶۳۴**	۹۰۴۹۷۰.۲۵۱	۸۹۵۳۷۳.۸۵۵	۸۹۴۷۰۳.۶۵۴
۷	۸۲۸۷۸۶.۰۵۰	۷۹۸۶۸۸.۳۱۶	۸۳۲۷۰۵.۴۵۶	۸۳۱۴۹۶.۱۶۵	**۷۹۷۳۹۶.۰۹۷**
۸	۹۰۴۷۳۳.۴۵۶	۸۵۷۶۳۴.۶۷۶	۹۱۰۳۶۲.۵۲۵	۹۱۰۴۹۸.۱۳۹	**۸۵۳۳۱۴.۵۴۴**
۹	۹۶۵۸۸۳.۵۷۴	۸۹۹۱۱۵.۲۰۳	۹۶۹۰۲۴.۴۳۶	۹۷۶۷۱۳.۳۱۴	**۸۹۴۸۱۲.۲۲۵**
میانگین	۹۱۰۹۳۳.۴۴۶	۸۹۱۴۴۸.۸۱۲	۹۱۲۶۲۱.۸۸۵	۹۱۰۱۶۹.۱۱۸	**۸۹۰۵۳۶.۷۸۹**

جدول ۴-۶: جدول درصد رسیدن به بهینگی - شرط خاتمه رسیدن به بهینگی نسبی

شماره مسأله	روش				
	GA	BBAM	BBAN	BBAO	BPSO
۱	۳۲.۰۰۰	**۹۸.۰۰۰**	۲۸.۰۰۰	۸۸.۰۰۰	**۹۸.۰۰۰**
۲	۱۹.۰۰۰	**۱۰۰.۰۰۰**	۲۱.۰۰۰	۸۶.۰۰۰	۹۶.۰۰۰
۳	۱۵.۰۰۰	**۹۲.۰۰۰**	۱۸.۰۰۰	۶۶.۰۰۰	۵۹.۰۰۰
۴	۱.۰۰۰	۴۹.۰۰۰	۰.۰۰۰	۲۴.۰۰۰	**۵۱.۰۰۰**
۵	۰.۰۰۰	**۸۵.۰۰۰**	۰.۰۰۰	۱۴.۰۰۰	۶۷.۰۰۰
۶	۰.۰۰۰	**۳۰.۰۰۰**	۱.۰۰۰	۱۷.۰۰۰	۱۲.۰۰۰
۷	۰.۰۰۰	۲.۰۰۰	۰.۰۰۰	۰.۰۰۰	**۶.۰۰۰**
۸	۰.۰۰۰	۱.۰۰۰	۰.۰۰۰	۰.۰۰۰	**۲۲.۰۰۰**
۹	۰.۰۰۰	۱.۰۰۰	۰.۰۰۰	۰.۰۰۰	**۵.۰۰۰**
میانگین	۷.۴۴۴	**۵۰.۸۸۹**	۷.۵۵۶	۳۲.۷۷۸	۴۶.۲۲۲

نتایج جدول‌های ۴-۵ و ۴-۶ که بر اساس شرط خاتمه رسیدن به بهینگی نسبی بدست آمده است نشان دهنده این مطلب است که همچنان الگوریتم صفر و یک خفاش میرجلیلی و بهینه‌سازی ازدحام ذرات نتایج بهتری چه از نظر تابع هدف و چه از نظر درصد رسیدن به نتایج بهینه و مطلوب از دیگر الگوریتم‌ها برتر هستند.

جدول ۴-۷: جدول میانگین مقدار تابع هدف- شرط خاتمه محدودیت زمانی

شماره مسأله	روش				
	GA	BBAM	BBAN	BBAO	BPSO
۱	۹۳۳۰۸۲.۰۷۵	**۹۳۲۶۶۳.۲۵۲**	۹۳۷۸۱۰.۸۷۴	۹۳۶۶۸۱.۹۶۰	۹۳۲۷۱۰.۷۵۳
۲	۹۷۸۷۹۵.۱۲۵	**۹۷۷۷۹۹.۴۰۰**	۹۸۲۱۲۱.۸۹۲	۹۸۲۳۴۲.۶۰۲	۹۷۷۹۱۴.۸۹۹
۳	۱۰۱۱۵۶۵.۱۱۷	**۱۰۱۰۹۱۸.۴۴۶**	۱۰۱۴۸۰۵.۹۵۵	۱۰۱۴۳۲۸.۰۱۶	۱۰۱۱۲۳۰.۶۰۷
۴	۷۹۸۶۰۲.۳۴۵	**۷۹۷۱۷۷.۷۲۹**	۸۰۶۱۹۲.۹۸۸	۸۰۸۶۸۹.۹۲۷	۷۹۷۳۲۰.۹۳۶
۵	۸۵۶۶۷۸.۹۰۸	**۸۵۵۰۰۱.۶۲۵**	۸۶۷۲۵۱.۴۸۲	۸۷۰۳۶۷.۵۸۳	۸۵۵۴۸۴.۵۷۲
۶	۸۹۵۲۴۲.۹۸۴	**۸۹۴۳۷۴.۱۰۳**	۹۱۱۴۲۰.۷۳۵	۹۱۶۳۶۱.۰۰۰	۸۹۴۶۳۹.۲۹۶
۷	۷۹۹۲۶۰.۳۵۹	۷۹۸۶۶۰.۱۶۴	۸۳۵۷۸۵.۲۴۷	۸۴۹۹۸۴.۳۸۴	**۷۹۶۶۸۹.۷۰۲**
۸	۸۵۶۶۴۷.۶۵۲	۸۵۷۴۶۸.۴۳۳	۹۱۳۷۳۴.۰۰۸	۹۴۴۱۱۱.۰۹۹	**۸۵۳۳۳۶.۷۷۴**
۹	۸۹۸۱۵۵.۳۰۲	۸۹۸۳۷۹.۰۳۲	۹۷۶۱۸۲.۴۷۸	۱۰۲۸۵۲۷.۹۰۵	**۸۹۴۷۹۱.۸۴۸**
میانگین	۸۹۲۰۰۳.۳۱۹	۸۹۱۳۸۲.۴۶۵	۹۱۶۱۴۵.۰۷۳	۹۲۷۹۳۲.۷۲۰	**۸۹۰۴۵۷.۷۱۰**

جدول ۴-۸: جدول درصد رسیدن به بهینگی - شرط خاتمه محدودیت زمانی

شماره مسأله	روش				
	GA	BBAM	BBAN	BBAO	BPSO
۱	۷۵.۰۰۰	**۹۷.۰۰۰**	۴.۰۰۰	۸.۰۰۰	۹۴.۰۰۰
۲	۷۲.۰۰۰	**۱۰۰.۰۰۰**	۵.۰۰۰	۵.۰۰۰	۹۷.۰۰۰
۳	۵۲.۰۰۰	**۸۴.۰۰۰**	۶.۰۰۰	۵.۰۰۰	۶۹.۰۰۰
۴	۱۹.۰۰۰	**۴۶.۰۰۰**	۰.۰۰۰	۰.۰۰۰	**۴۶.۰۰۰**
۵	۲۴.۰۰۰	**۷۹.۰۰۰**	۰.۰۰۰	۰.۰۰۰	۵۰.۰۰۰
۶	۱۷.۰۰۰	**۳۲.۰۰۰**	۰.۰۰۰	۰.۰۰۰	۱۳.۰۰۰
۷	۱.۰۰۰	۳.۰۰۰	۰.۰۰۰	۰.۰۰۰	**۹.۰۰۰**
۸	۲.۰۰	۰.۰۰۰	۰.۰۰۰	۰.۰۰۰	**۱۴.۰۰۰**
۹	۲.۰۰۰	۰.۰۰۰	۰.۰۰۰	۰.۰۰۰	**۵.۰۰۰**
میانگین	۲۹.۳۳۳	**۴۹.۰۰۰**	۱.۶۶۷	۲.۰۰۰	۴۴.۱۱۱

نتایج جدول‌های ۴-۷ و ۴-۸ که بر اساس شرط خاتمه محدودیت زمانی است نشان می‌دهد در زمان مشخص نیز الگوریتم صفر و یک خفاش میرجلیلی و بهینه‌سازی ازدحام ذرات نتایج بهتری هم از نظر تابع هدف و هم از نظر درصد رسیدن به نتایج بهینه و مطلوب از خود نشان می‌دهند.

در مجموع از کل نتایج بدست آمده می‌توان نتیجه گرفت که هر سه الگوریتم صفر و یک خفاش در حل مسائل مکان‌یابی تسهیلات با ظرفیت نامحدود کارا هستند و توانایی رقابت با الگوریتم‌های شناخته شده را دارند و ازمیان آن‌ها الگوریتم صفر و یک خفاش میرجلیلی کارآمدتر است.

۴-۲-۲ پیشنهاد

با توجه به اینکه کارایی الگوریتم صفر و یک خفاش میرجلیلی در حل مسائل مکان‌یابی تسهیلات با ظرفیت نامحدود با افزایش اندازه مسأله کمی کاهش می‌یابد؛ پیشنهاد می‌شود این الگوریتم بر روی دیگر مسائل تست مکان‌یابی تسهیلات با ظرفیت نامحدود که از اندازه بزرگتری برخودار هستند آزمایش شود و در صورت اثبات روند افت کارایی الگوریتم با افزایش اندازه مسأله، سعی شود با ترکیب الگوریتم با دیگر روش‌های ابتکاری و تجهیز آن به تکنیک‌های شناخته شده کارایی الگوریتم بهبود یابد.

فهرست منابع

[1] Bernhard, Korte and Vygen, J. Combinatorial optimization: Theory and algorithms. *Springer, Third Edition, 2005.*, 2008.

[2] Farahani, RZ and Hekmatfar, M. Facility location: concepts, models, algorithms and case studies. 2009. *Physica-Verlag, Heidelberg*.

[3] Laporte, Gilbert, Nickel, Stefan, and da Gama, Francisco Saldanha. *Introduction to Location Science*, pp. 1–18. Springer International Publishing, Cham, 2015.

[4] Weber, Alfred. *Ueber den standort der industrien*, vol. 2. Рипол Классик, 1909.

[5] Stollsteimer, John F. A working model for plant numbers and locations. *Journal of Farm Economics*, 45(3):631–645, 1963.

[6] Kuehn, Alfred A and Hamburger, Michael J. A heuristic program for locating warehouses. *Management science*, 9(4):643–666, 1963.

[7] Manne, Alan S. Plant location under economies-of-scale—decentralization and computation. *Management Science*, 11(2):213–235, 1964.

[8] Cornuéjols, Gérard, Nemhauser, George L, and Wolsey, Lairemce A. The uncapacitated facility location problem. tech. rep., Carnegie-mellon univ pittsburgh pa management sciences research group, 1983.

[9] Al-Sultan, Khaled S and Al-Fawzan, Mohammad Abdulrahman. A tabu search approach to the uncapacitated facility location problem. *Annals of Operations Research*, 86:91–103, 1999.

[10] Holmberg, Kaj. Exact solution methods for uncapacitated location problems with convex transportation costs. *European Journal of Operational Research*, 114(1):127–140, 1999.

[11] Arya, Vijay, Garg, Naveen, Khandekar, Rohit, Meyerson, Adam, Munagala, Kamesh, and Pandit, Vinayaka. Local search heuristics for k-median and facility location problems. *SIAM Journal on computing*, 33(3):544–562, 2004.

[12] Korupolu, Madhukar R, Plaxton, C Greg, and Rajaraman, Rajmohan. Analysis of a local search heuristic for facility location problems. *Journal of algorithms*, 37(1):146–188, 2000.

[13] Cormen, Thomas H. *Introduction to algorithms*. MIT press, 3rd ed. , 2009.

[۱۴] سادین، فاطمه. بررسی مساله مکانیابی هاب. پایان‌نامه کارشناسی‌ارشد، وزارت علوم، تحقیقات و فناوری - دانشگاه صنعتی شاهرود، ۱۳۹۰.

[۱۵] امین‌طوسی، محمود. مروری بر مسائل NP-Hard و NP-Complete. در مجله صفر و یک، صفحات ۲۵-۳۳. گروه کامپیوتر، دانشگاه فردوسی مشهد، مشهد، ایران، بهار ۱۳۷۹. شماره سوم.

[۱۶] محمدی، سعداله. مسئله مکانیابی معکوس روی گراف‌ها. پایان‌نامه کارشناسی‌ارشد، وزارت علوم، تحقیقات و فناوری - دانشگاه صنعتی شاهرود - دانشکده ریاضی، ۱۳۹۲.

[17] Karp, Richard M. Reducibility among combinatorial problems. in *Complexity of computer computations*, pp. 85–103. Springer, 1972.

[18] Bansal, Manisha. Approximation algorithms for facility location problems. Master's thesis, Department of Computer Science, University of Delhi, India, October 2013.

[19] Mahdian, Mohammad, Ye, Yinyu, and Zhang, Jiawei. Approximation algorithms for metric facility location problems. *SIAM Journal on Computing*, 36(2):411–432, 2006.

[20] Shmoys, David B, Tardos, Éva, and Aardal, Karen. Approximation algorithms for facility location problems. in *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pp. 265–274. ACM, 1997.

[21] Chudak, Fabián A and Shmoys, David B. Improved approximation algorithms for the uncapacitated facility location problem. *SIAM Journal on Computing*, 33(1):1–25, 2003.

[22] Sviridenko, Maxim. An improved approximation algorithm for the metric uncapacitated facility location problem. in *International Conference on Integer Programming and Combinatorial Optimization*, pp. 240–257. Springer, 2002.

[23] Jain, Kamal and Vazirani, Vijay V. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *Journal of the ACM (JACM)*, 48(2):274–296, 2001.

[24] Jain, Kamal, Mahdian, Mohammad, and Saberi, Amin. A new greedy approach for facility location problems. in *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pp. 731–740. ACM, 2002.

[25] Desale, Sachin, Rasool, Akhtar, Andhale, Sushil, and Rane, Priti. Heuristic and meta-heuristic algorithms and their relevance to the real world: a survey. *Int. J. Comp. Eng. Res. Trends*, 2(5):296–304, 2015.

[26] Tohyama, Hiroaki, Ida, Kenichi, and Matsueda, Jun. A genetic algorithm for the uncapacitated facility location problem. *Electronics and Communications in Japan*, 94(5):47–54, 2011.

[۲۷] شاهی، سمیرا. حل مسائل مکان‌یابی با استفاده از روش‌های فراابتکاری. پایان‌نامه کارشناسی‌ارشد، دانشگاه حکیم- سبزوار - دانشکده ریاضی و کامپیوتر، ۱۳۹۲.

[28] Glover, Fred. Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5):533–549, 1986.

[29] Hansen, Pierre. The steepest ascent mildest descent heuristic for combinatorial programming. in *Congress on numerical methods in combinatorial optimization, Capri, Italy*, pp. 70–145, 1986.

[30] de Werra, Dominique and Hertz, Alain. Tabu search techniques. *Operations-Research-Spektrum*, 11(3):131–141, 1989.

[31] Reeves, Colin R. *Modern heuristic techniques for combinatorial problems*. John Wiley & Sons, Inc., 1993.

[32] Hanafi, Saïd. On the convergence of tabu search. *Journal of Heuristics*, 7(1):47–58, 2001.

[33] Hertz, Alain, Taillard, Eric, and De Werra, Dominique. A tutorial on tabu search. in *Proc. of Giornate di Lavoro AIRO*, vol. 95, pp. 13–24, 1995.

[34] Sun, Minghe. Solving the uncapacitated facility location problem using tabu search. *Computers & Operations Research*, 33(9):2563–2589, 2006.

[35] Michel, Laurent and Van Hentenryck, Pascal. A simple tabu search for warehouse location. *European Journal of Operational Research*, 157(3):576–591, 2004.

[36] Kennedy, James and Eberhart, Russell C. A discrete binary version of the particle swarm algorithm. in *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, vol. 5, pp. 4104–4108. IEEE, 1997.

[37] Sudholt, Dirk and Witt, Carsten. Runtime analysis of a binary particle swarm optimizer. *Theoretical Computer Science*, 411(21):2084–2100, 2010.

[38] Neumann, Frank, Sudholt, Dirk, and Witt, Carsten. Rigorous analyses for the combination of ant colony optimization and local search. *Ant Colony Optimization and Swarm Intelligence*, pp. 132–143, 2008.

[39] Guner, Ali R and Sevkli, Mehmet. A discrete particle swarm optimization algorithm for uncapacitated facility location problem. *Journal of Artificial Evolution and Applications*, 2008, 2008.

[40] Pan, Quan-Ke, Tasgetiren, M Fatih, and Liang, Yun-Chia. A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. *Computers & Operations Research*, 35(9):2807–2839, 2008.

[41] Jaramillo, Jorge H, Bhadury, Joy, and Batta, Rajan. On the use of genetic algorithms to solve location problems. *Computers & Operations Research*, 29(6):761–779, 2002.

[42] Aydin, M Emin and Fogarty, Terence C. A distributed evolutionary simulated annealing algorithm for combinatorial optimisation problems. *Journal of Heuristics*, 10(3):269–292, 2004.

[43] Yang, Xin-She. A new metaheuristic bat-inspired algorithm. *Nature inspired cooperative strategies for optimization (NICSO 2010)*, pp. 65–74, 2010.

[44] Fister, Iztok, Yang, Xin-She, Fong, Simon, and Zhuang, Yan. Bat algorithm: recent advances. in *computational intelligence and informatics (CINTI), 2014 IEEE 15th international symposium on*, pp. 163–167. IEEE, 2014.

[45] Zhang, Jia Wei and Wang, Gai Ge. Image matching using a bat algorithm with mutation. in *Applied Mechanics and Materials*, vol. 203, pp. 88–93. Trans Tech Publ, 2012.

[46] Wang, Gaige and Guo, Lihong. A novel hybrid bat algorithm with harmony search for global numerical optimization. *Journal of Applied Mathematics*, 2013, 2013.

[47] Fister Jr, Iztok, Fister, Dušan, and Yang, Xin-She. A hybrid bat algorithm. *arXiv preprint arXiv:1303.6310*, 2013.

[48] Yang, Xin-She and Hossein Gandomi, Amir. Bat algorithm: a novel approach for global engineering optimization. *Engineering Computations*, 29(5):464–483, 2012.

[49] Homaifar, Abdollah, Qi, Charlene X, and Lai, Steven H. Constrained optimization via genetic algorithms. *Simulation*, 62(4):242–253, 1994.

[50] He, S, Prempain, E, and Wu, QH. An improved particle swarm optimizer for mechanical design optimization problems. *Engineering Optimization*, 36(5):585–605, 2004.

[51] Tsai, Pei Wei, Pan, Jeng Shyang, Liao, Bin Yih, Tsai, Ming Jer, and Istanda, Vaci. Bat algorithm inspired algorithm for solving numerical optimization problems. in *Applied Mechanics and Materials*, vol. 148, pp. 134–137. Trans Tech Publ, 2012.

[52] Osaba, Eneko, Yang, Xin-She, Diaz, Fernando, Lopez-Garcia, Pedro, and Carballedo, Roberto. An improved discrete bat algorithm for symmetric and asymmetric traveling salesman problems. *Engineering Applications of Artificial Intelligence*, 48:59–71, 2016.

[53] Nakamura, Rodrigo YM, Pereira, Luis AM, Costa, KA, Rodrigues, Douglas, Papa, João P, and Yang, X-S. Bba: a binary bat algorithm for feature selection. in *Graphics, Patterns and Images (SIBGRAPI), 2012 25th SIBGRAPI Conference on*, pp. 291–297. IEEE, 2012.

[54] Mirjalili, Seyedali, Mirjalili, Seyed Mohammad, and Yang, Xin-She. Binary bat algorithm. *Neural Computing and Applications*, 25(3-4):663–681, 2014.

پیوست آ

کد برنامه‌های متلب

برنامه آ-۱: کد برنامه الگوریتم خفاش ارائه شده توسط یانگ

```
                                                                    ۱
% ========================================================== %      ۲
% Files of the Matlab programs included in the book:        %      ۳
% Xin-She Yang, Nature-Inspired Metaheuristic Algorithms,   %      ۴
% Second Edition, Luniver Press, (2010).   www.luniver.com  %      ۵
% ========================================================== %      ۶
                                                                    ۷
% ---------------------------------------------------------- %      ۸
% Bat-inspired algorithm for continuous optimization (demo)%       ۹
% Programmed by Xin-She Yang @Cambridge University 2010     %      ۱۰
% ---------------------------------------------------------- %      ۱۱
% Usage: bat_algorithm([20 0.25 0.5]);                      %      ۱۲
                                                                    ۱۳
function [best,fmin,N_iter]=bat_algorithm(para)                     ۱۴
% Display help                                                      ۱۵
 help bat_algorithm.m                                               ۱۶
                                                                    ۱۷
% Default parameters                                                ۱۸
if nargin<1,  para=[10 0.25 0.5];  end                              ۱۹
n=para(1);      % Population size, typically 10 to 25              ۲۰
A=para(2);      % Loudness  (constant or decreasing)               ۲۱
```

```
r=para(3);      % Pulse rate (constant or decreasing)
% This frequency range determines the scalings
Qmin=0;         % Frequency minimum
Qmax=2;         % Frequency maximum
% Iteration parameters
tol=10^(-5);    % Stop tolerance
N_iter=0;       % Total number of function evaluations
% Dimension of the search variables
d=2;
% Initial arrays
Q=zeros(n,1);   % Frequency
v=zeros(n,d);   % Velocities
% Initialize the population/solutions
for i=1:n,
  Sol(i,:)=rand(1,d)-.5;
  Fitness(i)=Fun(Sol(i,:));
end
% Find the current best
[fmin,I]=min(Fitness);
best=Sol(I,:);

% ======================================================  %
% Note: As this is a demo, here we did not implement the  %
% reduction of loudness and increase of emission rates.   %
% Interested readers can do some parametric studies       %
% and also implementation various changes of A and r etc  %
% ======================================================  %

% Start the iterations -- Bat Algorithm
while (fmin>tol)
        % Loop over all bats/solutions
        for i=1:n,
          Q(i)=Qmin+(Qmax-Qmin)*rand;
          v(i,:)=v(i,:)+(Sol(i,:)-best)*Q(i);
```

```
          S(i,:)=Sol(i,:)+v(i,:);
          % Pulse rate
          if rand>r
              S(i,:)=best+0.01*randn(1,d);
          end

     % Evaluate new solutions
           Fnew=Fun(S(i,:));
     % If the solution improves or not too loudness
           if (Fnew<=Fitness(i)) & (rand<A) ,
                Sol(i,:)=S(i,:);
                Fitness(i)=Fnew;
           end

          % Update the current best
          if Fnew<=fmin,
                best=S(i,:);
                fmin=Fnew;
          end
        end
N_iter=N_iter+n;
end
% Output/display
disp(['Number of evaluations: ',num2str(N_iter)]);
disp(['Best =',num2str(best),' fmin=',num2str(fmin)]);
% Objective function -- Rosenbrock's 3D function
function z=Fun(u)
z=(u(1)+u(2)).^2;
%%%% ============ end ====================================
```

برنامه آ-۲: کد برنامه الگوریتم صفر و یک خفاش (برگرفته از روش جایگشتی اوسابا)

```
function[x,fval,iter] =BBA_Osaba(f)
Fun=@f;
n=50;      % Population size
A=zeros(1,n)+0.9;      % Loudness
r=zeros(1,n)+0.6;      % Pulse rate
a=0.98;
Y=a;
% This frequency range determines the scalings
% the frequency is'nt useed in this algorithm
% Iteration parameters
N_iter=0;       % Total number of function evaluations
% Dimension of the search variables
d=numel(f);
% Initial arrays
F=zeros(n,1);   % Frequency
v=zeros(n,d);   % Velocities
V=zeros(1,n);
% Initialize the population/solutions
for i=1:n,
    Sol(i,:)=randi([0 1],1,d);
    Fitness(i)=Fun(Sol(i,:));
end
% Find the current best
[fmin,I]=min(Fitness);
[gfmin,gbest]=sort(Fitness);
gfmin=gfmin(1:5);gbest=gbest(1:5);
best=Sol(I,:);
while N_iter< 100
% Loop over all bats/solutions
    for i=1:n,
          S(i,:)=Sol(i,:);
        for j=1:d
            v(i,j)=(abs(Sol(i,j)-best(j)));
```

```
        end                                                    ٣٤
            if sum(V)~=0;                                      ٣٥
            V(i)=randperm(1,sum(v));                           ٣٦
                if V(i)<= n/2;                                 ٣٧
                   j=randperm(sum(V),2);                       ٣٨
                   m= Sol(i,:);                                ٣٩
                   S(i,j(1))= m(1,j(2));                       ٤٠
                   S(i,j(2))= m(1,j(1));                       ٤١
            elseif sum(v)>3                                    ٤٢
                   j=randperm(sum(V),3);                       ٤٣
                  m= Sol(i,:);                                 ٤٤
                  S(i,j(1))=m(1,j(2));                         ٤٥
                  S(i,j(2))=m(1,j(3));                         ٤٦
                  S(i,j(3))=m(1,j(1));                         ٤٧
             end                                               ٤٨
          end                                                  ٤٩
                                                               ٥٠
% Pulse rate                                                  ٥١
           if rand>r(i)                                        ٥٢
             mm=randperm(5,1);                                 ٥٣
             bests= Sol(gbest(mm),:);                          ٥٤
             j=randperm(d,3);                                  ٥٥
             S(i,:)=bests;                                     ٥٦
             S(i,j(1))=bests(1,j(2));                          ٥٧
             S(i,j(2))=bests(1,j(3));                          ٥٨
             S(i,j(3))=bests(1,j(1));                          ٥٩
           end                                                 ٦٠
 % Evaluate new solutions                                      ٦١
        Fnew=Fun(S(i,:));                                      ٦٢
% If the solution improves or not too loudness                ٦٣
         if (Fnew<=Fitness(i)) || (rand<A(i)) ,                ٦٤
            Sol(i,:)=S(i,:);                                   ٦٥
            Fitness(i)=Fnew;                                   ٦٦
         end                                                   ٦٧
```

```
                                                                    ۶۸
% Update the current best                                           ۶۹
          if Fnew<=fmin,                                           ۷۰
             best=S(i,:);                                          ۷۱
             fmin=Fnew;                                            ۷۲
             A(i)=a*A(i);                                          ۷۳
            r(i)=1*(1-exp(-Y*N_iter));                             ۷۴
         end                                                       ۷۵
     end                                                           ۷۶
N_iter=N_iter+1;                                                   ۷۷
iter= N_iter;                                                      ۷۸
fval=fmin;                                                         ۷۹
x=best;                                                            ۸۰
```

برنامه آ-۳: کد برنامه الگوریتم صفر و یک خفاش (میرجلیلی)

```
function[x,fval,iter] = BBA_Mirjalili(f)
Fun=f;
n=50;      % Population size
A=zeros(1,n)+0.95;      % Loudness
r=zeros(1,n)+0.6;      % Pulse rate
a=0.98;
Y=a;
% This frequency range determines the scalings
Fmin=0;          % Frequency minimum
Fmax=2;          % Frequency maximum
% Iteration parameters
N_iter=0;        % Total number of function evaluations
% Dimension of the search variables
d=numel(f);
% Initial arrays
F=zeros(n,1);    % Frequency
v=zeros(n,d);    % Velocities
% Initialize the population/solutions
Sol=zeros(n,d);
Fitness=zeros(n);
Vv=zeros(n,d);
S=zeros(1,d);
for i=1:n,
    Sol(i,:)=randi([0 1],1,d);
    Fitness(i)=(Sol(i,:))*Fun';
end
% Find the current best
[fmin,I]=min(Fitness);
best=Sol(I,:);
while N_iter< 500 && fmin>0
    % Loop over all bats/solutions and  Pulse rate
for i=1:n,
    for j=1:d
```

```
            if rand>r(i)                                                    ۳۴
            S(1,j)=best(1,j);                                               ۳۵
            else                                                            ۳۶
            F(i)=Fmin+(Fmax-Fmin)*rand;                                     ۳۷
            v(i,j)=v(i,j)+(Sol(i,j)-best(j))*F(i);                          ۳۸
            Vv(i,j)=abs((2/pi)*atan((pi/2)*v(i,j)));                        ۳۹
                if Vv(i,j) <= rand;                                         ۴۰
                    S(1,j)=~Sol(i,j);                                       ۴۱
                else                                                        ۴۲
                    S(1,j)= Sol(i,j);                                       ۴۳
                end                                                         ۴۴
             end                                                            ۴۵
    end                                                                     ۴۶
% Evaluate new solutions                                                   ۴۷
            Fnew=(S(i,:))*Fun';                                             ۴۸
% If the solution improves or not too loudness                            ۴۹
        if (Fnew<=Fitness(i)) || (rand<A(i)) ,                              ۵۰
            Sol(i,:)=S(1,:);                                                ۵۱
            Fitness(i)=Fnew;                                                ۵۲
        end                                                                 ۵۳
% Update the current best                                                  ۵۴
        if Fnew<=fmin,                                                      ۵۵
            best=S(i,:);                                                    ۵۶
            fmin=Fnew;                                                      ۵۷
            A(i)=a*A(i);                                                    ۵۸
           r(i)=1*(1-exp(-Y*N_iter));                                       ۵۹
        end                                                                 ۶۰
 end                                                                        ۶۱
  N_iter=N_iter+1;                                                          ۶۲
end                                                                         ۶۳
iter= N_iter;                                                               ۶۴
fval=fmin;                                                                  ۶۵
x=best;                                                                     ۶۶
```

برنامه آ-۴: کد برنامه الگوریتم صفر و یک خفاش (ناکامورا)

```
function[x,fval,iter] = BBA_Nakamura(f)
Fun= f
n=50;      % Population size
A=zeros(1,n)+0.95;      % Loudness
r=zeros(1,n)+0.6;      % Pulse rate
a=0.98;
Y=a;
% This frequency range determines the scalings
Fmin=-0.0008;          % Frequency minimum
Fmax=0.0008;          % Frequency maximum
% Iteration parameters
N_iter=0;       % Total number of function evaluations
% Dimension of the search variables
d=numel(f);
% Initial arrays
F=zeros(n,1);   % Frequency
v=zeros(n,d);   % Velocities
% Initialize the population/solutions
for i=1:n,
    Sol(i,:)=randi([0 1],1,d);
    Fitness(i)=(Sol(i,:))*Fun';
end
% Find the current best and the bests Bat
[fmin,I]=min(Fitness);
[gfmin,gbest]=sort(Fitness);
gfmin=gfmin(1:5);gbest=gbest(1:5);
best=Sol(I,:);
while N_iter< 100
    % Loop over all bats/solutions
    for i=1:n,
        for j=1:d
            F(i)=Fmin+(Fmax-Fmin)*rand;
            v(i,j)=v(i,j)+(Sol(i,j)-best(j))*F(i);
```

```
            Sv(i,j)=1/(1+exp(-v(i,j)));
             if Sv(i,j) <= rand;
                S(i,j)=0;
            else
                S(i,j)= 1;
            end

       end
% Pulse rate
        if rand>r(i)
            mm=randperm(5,1);
            bests= Sol(gbest(mm),:);
            E=-1+(1-(-1))*rand;
            At=sum(A);
            S(i,:)=best+E*At;
     for i=1:n,
        for j=1:d
             SS(i,j)=1/(1+exp(-v(i,j)));
              if SS(i,j) <= rand;
                S(i,j)=0;
            else
                S(i,j)= 1;
             end

        end
     end
        end
% Evaluate new solutions
        Fnew=(S(i,:))*Fun';
% If the solution improves or not too loudness
        if (Fnew<=Fitness(i)) || (rand<A(i)) ,
               Sol(i,:)=S(i,:);
               Fitness(i)=Fnew;
        end
```

```
                                                                    ۶۸
% Update the current best                                           ۶۹
        if Fnew<=fmin,                                             ۷۰
                best=S(i,:);                                       ۷۱
                fmin=Fnew;                                         ۷۲
               A(i)=a*A(i);                                        ۷۳
                r(i)=1*(1-exp(-Y*N_iter));                         ۷۴
        end                                                        ۷۵
    end                                                            ۷۶
 N_iter=N_iter+1;                                                  ۷۷
end                                                                ۷۸
iter= N_iter;                                                      ۷۹
fval=fmin;                                                         ۸۰
x=best;                                                            ۸۱
```

واژه‌نامه فارسی به انگلیسی

ابتکاری . . . . . Heuristics

تخصیص حساب بانکی . . . . . Bank account allocation

ترکیبی فراابتکاری . . . . . A hybrid multi start heuristic

زنجیره‌ی تأمین . . . . . Supply chain

فرکانس . . . . . Frequency

هنگام ،اکتاو . . . . . Octave

الگوریتم ازدحام ذرات . . . . . Particle swarm optimization

الگوریتم تبرید شبیه سازی شده . . . . . Simulated annealing

الگوریتم تکامل تفاضلی . . . . . Differential evolution algorithm

الگوریتم خفاش ارتقاء یافته . . . . . Evolved Bat Algorithm

الگوریتم خفاش . . . . . Bat Algorithm

الگوریتم ژنتیک . . . . . Genetic algorithm

انتخاب . . . . . Selection

آزادسازی لاگرانژ . . . . . Lagrangean relaxation

برنامه‌ریزی پویا . . . . . Dynamic programming

برنامه‌ریزی ماشین و مدیریت موجودی . . . . . Machine scheduling and inventory management

بزرگ‌خفاش . . . . . Megabat

بهینه‌سازی کلونی مورچه . . . . . Ant Colony Optimization

بهینه‌سازی مهندسی طراحی . . . . . Engineering design optimisation

پالس . . . . . Pulse

پژواک‌یابی . . . . . Echolocation

پیچیدگی زمانی . . . . . Time complexity

پیوسته . . . . . Continuous

تابع ارزیابی . . . . . Evaluation function

تابع برازندگی . . . . . Fitness function

تپه-نوردی . . . . . Hill-climbing

تجزیه و تحلیل خوشه‌ای . . . . . Clustering analysis

ترکیب . . . . . Crossover

تشدید . . . . . Intensification

تعداد انفجارها . . . . . Rates

تقاضا . . . . . Demand

تکاملی . . . . . Evoluitionary

تنوع . . . . . Diversification

جستجوی تابو . . . . . Tabu search

جستجوی محلی . . . . . Local search

جواب شدنی . . . . . Feasible solution

جهش . . . . . Mutation

چندجمله‌ای . . . . . Polynomial

حافظه بلند مدت . . . . . Long term memory

درخت‌های پوشا . . . . . Spanning tree

روباه پرنده . . . . . Flying fox

روش‌های تقریبی . . . . . Approximation algorithms

رویکرد اولیه-دوگان . . . . . Primal-dual approach

رویکرد حریصانه . . . . . Greedy approach

زیست‌سونار . . . . . Bio sonar

ژن . . . . . Gene

ساختار خاصی از مسأله . . . . . Exploiting special problem structure

شاخه و کران . . . . . Branch-and-bound

شبکه عصبی . . . . . Neural Networks

ضریب تقریبی . . . . . Approximation factor

طراحی شبکه‌های ارتباطی . . . . . Design of communication networks

طول موج . . . . . Wavelength

علوم منطقه‌ای، آماد . . . . . Logistics

غیر چند جمله‌ای-سخت . . . . . NP-hard

غیر چند جمله‌ای-کامل . . . . . NP-complete

غیر چند جمله‌ای . . . . . Non-deterministic polynomial

فاصله همینگ . . . . . Hamming Distanc

فراابتکاری . . . . . Meta-Heuristics

فوق الگوریتم خفاش . . . . . Hybrid bat algorithm

کدهای هافمن . . . . . Huffman codes

کروموزم . . . . . Chromosome

کوتاه‌ترین مسیر . . . . . Shortest path

کوچک‌خفاش . . . . . Microbat

گردکردن برنامه خطی . . . . . LP-rounding

گسسته . . . . . Discrete

لیست ممنوعه . . . . . Tabu list

متریک . . . . . Metric

متقارن . . . . . Symmetric

محل اسکان سیستم‌های حفاری دریایی . . . . . Location of offshore drilling platforms

محل صندوق پرداختی . . . . . Lock-box location

مرکز . . . . . Center

مسأله فروشنده دوره‌گرد . . . . . Traveling Salesman Problem

مسأله مکان‌یابی انبار . . . . . Warehouse Location Problem

مسأله مکان‌یابی کارخانه . . . . . Plant Location Problem

مشتری . . . . . Clinet

معیار تنفس . . . . . Aspiration criteria

مکان‌یابی تدافعی . . . . . Defensive Location Problem

مکان‌یابی تسهیلات ساده . . . . . Simple Facility Location Problem

مکان‌یابی تسهیلات . . . . . Facility Location Problem

مکان‌یابی رقابتی . . . . . Competitive Location Problem

مکان‌یابی هاب . . . . . Hub Location Problem

میانه . . . . . Median

ناحیه شدنی . . . . . Feasible Region

نامتقارن . . . . . Asymmetric

نهنگ‌های دندان‌دار . . . . . Toothed whales

هوش گروهی . . . . . Swarm intelligence

واژه‌نامه انگلیسی به فارسی

ترکیبی فراابتکاری . . . . . A hybrid multi start heuristic

بهینه‌سازی کلونی مورچه . . . . . Ant Colony Optimization

روش‌های تقریبی . . . . . Approximation algorithms

ضریب تقریبی . . . . . Approximation factor

معیار تنفس . . . . . Aspiration criteria

نامتقارن . . . . . Asymmetric

تخصیص حساب بانکی . . . . . Bank account allocation

الگوریتم خفاش . . . . . Bat Algorithm

زیست‌سونار . . . . . Bio sonar

شاخه و کران . . . . . Branch-and-bound

مرکز . . . . . Center

کروموزم . . . . . Chromosome

مشتری . . . . . Clinet

تجزیه و تحلیل خوشه‌ای . . . . . Clustering analysis

مکان‌یابی رقابتی . . . . . Competitive Location Problem

پیوسته . . . . . Continuous

ترکیب . . . . . Crossover

مکان‌یابی تدافعی . . . . . Defensive Location Problem

تقاضا . . . . . Demand

طراحی شبکه‌های ارتباطی . . . . . Design of communication networks

الگوریتم تکامل تفاضلی . . . . . Differential evolution algorithm

گسسته . . . . . Discrete

تنوع . . . . . Diversification

برنامه‌ریزی پویا . . . . . Dynamic programming

پژواک‌یابی . . . . . Echolocation

بهینه‌سازی مهندسی طراحی . . . . . Engineering design optimisation

تابع ارزیابی . . . . . Evaluation function

تکاملی . . . . . Evoluitionary

الگوریتم خفاش ارتقاء یافته . . . . . Evolved Bat Algorithm

ساختار خاصی از مسأله . . . . . Exploiting special problem structure

مکان‌یابی تسهیلات . . . . . Facility Location Problem

ناحیه شدنی . . . . . Feasible Region

جواب شدنی . . . . . Feasible solution

تابع برازندگی . . . . . Fitness function

روباه پرنده . . . . . Flying fox

فرکانس . . . . . Frequency

ژن . . . . . Gene

الگوریتم ژنتیک . . . . . Genetic algorithm

رویکرد حریصانه . . . . . Greedy approach

فاصله همینگ . . . . . Hamming Distanc

ابتکاری . . . . . Heuristics

تپه-نوردی . . . . . Hill-climbing

مکان‌یابی هاب . . . . . Hub Location Problem

کدهای هافمن . . . . . Huffman codes

فوق الگوریتم خفاش . . . . . Hybrid bat algorithm

تشدید . . . . . Intensification

آزادسازی لاگرانژ . . . . . Lagrangean relaxation

جستجوی محلی . . . . . Local search

محل اسکان سیستم‌های حفاری دریایی . . . . . Location of offshore drilling platforms

محل صندوق پرداختی . . . . . Lock-box location

علوم منطقه‌ای، آماد . . . . . Logistics

حافظه بلند مدت . . . . . Long term memory

گردکردن برنامه خطی . . . . . LP-rounding

برنامه‌ریزی ماشین و مدیریت موجودی . . . . . Machine scheduling and inventory management

میانه . . . . . Median

بزرگ‌خفاش . . . . . Megabat

فراابتکاری . . . . . Meta-Heuristics

متریک . . . . . Metric

کوچک‌خفاش . . . . . Microbat

جهش . . . . . Mutation

شبکه عصبی . . . . . Neural Networks

غیر چند جمله‌ای . . . . . Non-deterministic polynomial

غیر چند جمله‌ای-کامل . . . . . NP-complete

غیر چند جمله‌ای-سخت . . . . . NP-hard

هنگام ،اکتاو . . . . . Octave

الگوریتم ازدحام ذرات . . . . . Particle swarm optimization

مسأله مکان‌یابی کارخانه . . . . . Plant Location Problem

چندجمله‌ای . . . . . Polynomial

رویکرد اولیه-دوگان . . . . . Primal-dual approach

پالس . . . . . Pulse

تعداد انفجارها . . . . . Rates

انتخاب . . . . . Selection

کوتاه‌ترین مسیر . . . . . Shortest path

مکان‌یابی تسهیلات ساده . . . . . Simple Facility Location Problem

الگوریتم تبرید شبیه سازی شده . . . . . Simulated annealing

درخت‌های پوشا . . . . . Spanning tree

زنجیره‌ی تأمین . . . . . Supply chain

هوش گروهی . . . . . Swarm intelligence

متقارن . . . . . Symmetric

لیست ممنوعه . . . . . Tabu list

جستجوی تابو . . . . . Tabu search

پیچیدگی زمانی . . . . . Time complexity

نهنگ‌های دندان‌دار . . . . . Toothed whales

مسأله فروشنده دوره‌گرد . . . . . Traveling Salesman Problem

مسأله مکان‌یابی انبار . . . . . Warehouse Location Problem

طول موج . . . . . Wavelength

Hakim Sabzevari University

An Outline of MSc. Thesis

Surname:Ghezi	Name:Sakine Sadat	Student No.:9413133048

Supervisor: Dr. Mahmood Amintoosi

Advisor: Dr. Mehdi Zaferaniehi

Faculty of Mathematics and Computer Science

Program: Applied Mathematics Field:Operational Research

Title of thesis: Solving the Uncapacitated Facility Location Problems using Bat Algorithm

Keywords:Location, Facility, Uncapacitated, Metaheuristic, Bat Algorithm

Abstract: The Uncapacitated Facility Location Problem (UFLP) is a fundamental optimization problem involving the selection of locations at which facilities supplying the same service are to be placed. The ”minimize” UFLP is to open facilities at a subset of potential sites and to assign each client to an open facility so as to minimize the sum of the ”service costs” and the ”fixed costs”. The UFLP is a binary optimization problem investigated by using various methods in the literature. Since it has been shown that the UFLP is NP-hard, it has generally been thought that there is no hope of finding a polynomial time algorithm by which an optimal solution is always obtained. In this dissertation demonstrates three solution methodologies for UFLP by three binary versions of a novel swarm intelligence method namely bat algorithm (BA). BA is one of the recently proposed heuristic algorithms employed for solving continuous optimization problems in the literature, suggested by inspiring the echolocation behavior of micro bats in nature. The superior performance of this algorithm for continuous problems has been proven among the other most well-known algorithms such as genetic algorithm (GA) and particle swarm optimization (PSO). Therefore in this dissertation, the UFLP is considered; the binary Bat algorithms are tested on 9 standard test problems taken from OR-library and their performance is compared with the well-known algorithms such as GA and PSO. According to the experimental results, binary Bat algorithms acquires successful results for solving UFLP in terms of solution quality and computation speed.

Hakim Sabzevari University
Faculty of Mathematics and Computer Science

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

Solving the Uncapacitated Facility Location Problems using Bat Algorithm

Supervisor:
Dr. Mahmood Amintoosi

Advisor:
Dr. Mehdi Zaferaniehi

By:
Sakine Sadat Ghezi

December 2017