بسم الله الرحمن الرحیم

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 بهینه‌سازی فاخته

استادان راهنما

دکتر محمدعلی پرتانیان و دکتر امین رفیعی

استاد مشاور

دکتر محمود امین‌طوسی

پژوهشگر:

سمیه حکم آبادی

شهریور ۱۳۹۶

باسمه تعالی

فرم ارزشیابی و صورتجلسه دفاع از پایان‌نامه کارشناسی ارشد

فرم ۱۱۳-ت

جلسه دفاع از پایان‌نامه آقای /خانم **سمیه حکم آبادی** دانشجوی رشته ریاضی کاربردی گرایش تحقیق در عملیات به شماره دانشجویی ۹۴۱۳۱۳۳۱۰۶ با عنوان:

حل مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود با الگوریتم بهینه‌سازی فاخته

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 سمیه حکم آبادی به عنوان کارشناس ارشد در رشته مذکور شناخته می‌شود .

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

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

مشخصات هیات دوران

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

امضا

مدیر گروه

امضا

رئیس دانشکده

سوگند نامه دانش آموختگان دانشگاه حکیم سبزواری

به نام خداوند جان و خرد کزین برتر اندیشه بر نگذرد

اینک که به خواست آفریدگار پاک، کوشش خویش و بهره گیری از دانش استادان و سرمایه‌های مادی و معنوی این مرز و بوم، توشه‌ای از دانش و خرد گردآورده‌ام، در پیشگاه خداوند بزرگ سوگند یاد 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: سمیه حکم آبادی

تاریخ و امضا:

تأییدیه‌ی صحت و اصالت نتایج

باسمه تعالی

اینجانب سمیه حکم آبادی به شماره دانشجویی ۹۴۱۳۱۳۳۱۰۶ 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: سمیه حکم آبادی

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.

استادان راهنما: دکتر محمدعلی پرتانیان

دکتر امین رفیعی

تاریخ و امضا:

تقدیم به:

همسر و فرزندانم

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 اساتید راهنمای خود، جناب آقای دکتر پرتانیان و جناب آقای دکتر رفیعی، صمیمانه تشکر و قدردانی کنم که قطعاً بدون راهنمایی های ارزنده ایشان، این مجموعه به انجام نمی رسید.
از جناب آقای دکتر امین طوسی 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 پدید آورندگان بسته ی زی پرشین و گروه پارسی لاتک کمال قدردانی را داشته باشم.
در پایان، تشکر می‌کنم از خانواده عزیزم به پاس عاطفه سرشار و گرمای امیدبخش وجودشان، که بهترین پشتیبان من بودند.

سمیه حکم آبادی
شهریور ۱۳۹۶

فهرست مطالب

فهرست جداول

فهرست تصاویر

فرم چکیده ی پایان‌نامه ی دوره ی تحصیلات تکمیلی

مدیریت تحصیلات تکمیلی

دانشگاه حکیم سبزواری

نام خانوادگی دانشجو:حکم آبادی	نام:سمیه	ش. دانشجویی:۹۴۱۳۱۳۳۱۰۶
استادان راهنما: دکتر محمدعلی پرتانیان و دکتر امین رفیعی		
استاد مشاور: دکتر محمود امین‌طوسی		
دانشکده ریاضی و علوم کامپیوتر	رشته: ریاضی کاربردی	گرایش: تحقیق در عملیات
مقطع: کارشناسی ارشد	تاریخ دفاع: شهریور ۱۳۹۶	تعداد صفحات: ۷۸
عنوان پایان‌نامه: حل مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود با الگوریتم بهینه‌سازی فاخته		
کلید واژه‌ها:مکان‌یابی، تسهیلات، ظرفیت نامحدود، فراابتکاری، الگوریتم بهینه‌سازی فاخته		

چکیده: مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود شامل مکان‌یابی تعداد نامشخصی مراکز تسهیلات با ظرفیت نامحدود می‌باشد به گونه‌ای که مجموع هزینه‌های ثابت راه‌اندازی مراکز و هزینه‌های متغیر برآوردن تقاضای بازار از این تسهیلات حداقل شود.

مسأله مکان‌یابی تسهیلات در عین حال که ساختار ساده‌ای دارد، از نظر پیچیدگی محاسباتی جزء مسائل NP-hard است. با توجه به اینکه الگوریتم‌های فراابتکاری برای مسائل پیچیده و بزرگ، جوابی نزدیک به بهینه در زمانی مطلوب ارائه می‌دهند، از این الگوریتم‌ها برای حل UFLP استفاده می‌نماییم.

در این تحقیق یکی از الگوریتم‌های فراابتکاری به نام «الگوریتم بهینه‌سازی فاخته» را مورد بررسی قرار می‌دهیم. سپس این الگوریتم را برای حل مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود، به‌کار می‌بریم و در مقایسه با نتایج دو الگوریتم ژنتیک و بهینه‌سازی ازدحام ذرات برای حل UFLP ، کارایی الگوریتم بهینه‌سازی فاخته را ارزیابی می‌نماییم.

پیش‌گفتار

در سال های اخیر مطالعات مکان‌یابی به عنوان یکی از عناصر کلیدی در موفقیت و بقای مراکز صنعتی مطرح است. به عنوان مثال، تعیین محل کارخانه یکی از کلیدی‌ترین گام‌های تأسیس کارخانه است چرا که نتایج این تصمیم در دراز مدت ظاهر می‌شود. یکی از جنبه‌های تأثیر درون سازمانی مطالعات مکان‌یابی، تأثیر مستقیم آن در سوددهی کارخانه است و از بعد برون سازمانی، ساخت کارخانه‌های بزرگ در یک منطقه می‌تواند شرایط مختلف اقتصادی، اجتماعی، فرهنگی، محیط زیست و ... را تحت تأثیر خود قرار دهد. لذا انتخاب مکان مناسب برای تسهیلات یکی از تصمیمات استراتژیک در کسب و کار محسوب می‌شود و توجه ویژه به این‌گونه مسائل از اهمیت زیادی برخوردار است.

توجه به مسائل مکان‌یابی به سال ۱۸۶۹ برمی‌گردد، زمانی که کمیل جردن [۱] مسأله‌ای در مورد فرم‌های درجه دوم مطالعه می‌کرد و اتفاقاً اولین مسأله مکان‌یابی (روی یک شبکه درختی) را در نظر گرفت. ولی این آلفرد وبر [۲] بود که اولین مدل مکان‌یابی را در کتابی در سال ۱۹۰۹ معرفی کرد. با این حال پس از سال ۱۹۶۰ این موضوع به طور متمرکز مورد مطالعه قرار گرفته و پیشرفت قابل توجهی در این زمینه رخ داده است. کتابشناسی اختصاصی مکان‌یابی تسهیلات در سال ۱۹۸۵ توسط دومسچک [۳] و دریکسل [۴] منتشر شد که شامل بیش از ۱۵۰۰ منبع می‌باشد. نگاهی به مقالات پراکندگی زیادی از مسائل را نشان می‌دهد[۱].

مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود (UFLP) به شکل اولیه مسائل گسسته در ابتدا توسط بالینسکی [۵] ، کوهن [۶] و هامبورگر [۷] ، من [۸] و استول استایمر [۹] فرموله شد[۲].

ارلنکوتر [۱۰] در سال ۱۹۷۸ یک الگوریتم مبتنی بر دوگان برای حل UFLP ارائه کرده است که یکی از کارآمدترین تکنیک‌های حل این مسأله است. قبل از ارلنکوتر بهترین روش شناخته شده برای حل UFLP ، الگوریتم شاخه و کران که توسط افریسون [۱۱] و ری [۱۲] (۱۹۶۶) توسعه یافت و روش شمارش ضمنی اسپیلبرگ [۱۳] (۱۹۶۹) بود[۳]. درسال ۱۹۸۳ پروزان [۱۴] و کراروپ [۱۵] بررسی جامعی از مطالب نوشته شده درباره UFLP تهیه کردند و با نشان دادن روابط بین UFLP و مسائل پوشش و افراز مجموعه، NP –کامل بودن UFLP را به اثبات رساندند[۳].

تا سال ۱۹۹۷ هیچ الگوریتم تقریبی برای حل این مسائل شناخته نشده بود. تا آن زمان چند روش کاملاً متفاوت برای ثابت کردن یک کران بالا برای نسبت تقریب استفاده شده است. در سال ۲۰۰۰ کروپلو [۱۶] ، فلکستون [۱۷] و راجارامان [۱۸]

---

[۱] Camille Jordan [۲] Alfred Weber [۳] Domschke [۴] Drexl [۵] Balinski [۶] Kuehn

[۷] Hamburger [۸] Manne [۹] Stollsteimer [۱۰] Erlenkotter [۱۱] Efroymson [۱۲] Ray [۱۳] Spielberg

[۱۴] Pruzan [۱۵] Krarup [۱۶] Korupolu [۱۷] Plaxton [۱۸] Rajaraman

برای اولین بار روش جستجوی محلی را برای مسائل مکان‌یابی تسهیلات به کار بردند و به نتایج خوبی دست یافتند[۲]. الگوریتم‌های فراابتکاری نظیر الگوریتم ژنتیک، الگوریتم بهینه‌سازی ازدحام ذرات و الگوریتم کلونی زنبور مصنوعی برای حل UFLP در دهه گذشته ارائه شده که توانسته‌اند در زمان‌های معقول جواب‌هایی با کیفیت بالا تولید کنند.

مسأله مکان‌یابی از زمان تولدش در دهه ۶۰ میلادی تا کنون بسیار مورد توجه بوده و نظر به دامنه وسیع کاربردهای آن، الگوریتم‌های دقیق و تقریبی بسیاری توسط پژوهشگران حوزه‌های مختلف ریاضی، علوم کامپیوتر و مهندسی برای حل آن ارائه شده است. مسأله مکان‌یابی تسهیلات در عین حال که ساختار ساده‌ای دارد، از نظر پیچیدگی محاسباتی جزء مسائل NP-hard است و شاید همین امر موجب شده است که روش‌های نوظهور برای حل مسائل بهینه‌سازی ترکیبیاتی یا مسائل بهینه‌سازی خطی با اعداد صحیح، این مسأله را به عنوان یکی از نمونه‌های اولیه مورد آزمایش خود در نظر بگیرند.

این پایان‌نامه شامل ۴ فصل است.

در فصل اول مسائل مکان‌یابی و به طور متمرکز مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود مطالعه می‌شود.

در فصل دوم روش‌های حل مسائل بهینه‌سازی مورد بررسی قرار می‌گیرد و دو الگوریتم ژنتیک و بهینه‌سازی ازدحام ذرات معرفی می‌شوند.

در فصل سوم الگوریتم بهینه‌سازی فاخته که محور اصلی تحقیق است، با جزئیات کامل بیان می‌شود.

در فصل چهارم گسسته‌سازی‌های انجام شده بر روی الگوریتم بهینه‌سازی فاخته شرح داده می‌شود و بعد از پیاده‌سازی الگوریتم گسسته فاخته بر روی UFLP ، نتایج حاصل را با نتایجی که از الگوریتم ژنتیک و الگوریتم بهینه‌سازی ازدحام ذرات برای حل UFLP به‌دست آمده است، مقایسه و کارایی الگوریتم بهینه‌سازی فاخته را بررسی می‌کنیم.

قسمتی از مطالب این پایان‌نامه برگرفته شده از مقاله زیر می‌باشد:

Rajabioun, R. Cuckoo Optimization Algorithm. Applied Soft Computing Journal., 11:5508- 5518, 2011.

فصل ۱

مکان‌یابی

۱-۱ مقدمه

مسائل مکان‌یابی [۱] رده بسیار وسیعی را در حوزه مسائل تحقیق در عملیات و مسائل ترکیبیاتی شامل می‌شود. کاربرد وسیع مسائل مکان‌یابی باعث شده تا هم چنان جزء مسائل جذاب شمرده شده و هر سال تحقیقات متنوعی پیرامون آن صورت گیرد.

هدف این گونه مسائل تعیین مکانی مناسب برای ایجاد مراکز خدمات‌رسانی است به طوری که کارایی مراکز تا آنجا که امکان دارد بهینه گردد. تعیین مکانی مناسب برای ایستگاه‌های آتش‌نشانی، بیمارستان‌ها، ادارات پست، مراکز هسته‌ای، فرودگاه‌ها و ... از جمله کاربردهای آن به شمار می‌رود. قلمرو این گونه مسائل به شهرسازی خلاصه نمی‌شود. تعیین مکان برای سرورهای کامپیوتری، طراحی بردهای الکترونیکی، تعیین مکان برای انبار و مثال‌های متنوع دیگر، مبین کاربردهای متنوع این مسائل در علوم مختلف است.

همان طور که گفته شد مسأله مکان‌یابی به تصمیم‌گیری در مورد مکان راه‌اندازی تسهیلات [۲] جدید با توجه به تسهیلات موجود و مشتری‌ها، برای بهینه‌سازی برخی از معیارهای اقتصادی، مربوط می‌شود. کارخانه‌ها، انبارها، مدارس، بیمارستان‌ها، ساختمان‌های اداری و فروشگاه‌های بزرگ نمونه‌هایی از تسهیلات می‌باشند.

۱-۲ انواع مسائل مکان‌یابی

مسائل مکان‌یابی دارای تنوع زیادی هستند، از این رو برای سهولت در بیان، این مسائل را به طرق مختلفی دسته‌بندی کرده‌اند. یکی از این دسته‌بندی‌ها با توجه به فضای مکان‌یابی یا به عبارت دیگر دامنه مکان‌یابی می‌باشد که مسائل مکان‌یابی را در

---

[۱] Location Problem [۲] Facility

سه دسته قرار می‌دهد: اولین دسته شامل مسائلی است که فضای مکان‌یابی پیوسته است. قابل ملاحظه‌ترین مسأله از این دسته، مسأله‌ی فرما-وبر [۱] است که شامل پیدا کردن یک نقطه در فضای R^m به طوری که مجموع فاصله‌های وزن دار آن تا n نقطه‌ی داده شده کمینه شود، می‌باشد.

دومین دسته شامل مسائل مکان‌یابی روی شبکه‌هاست. در این مسائل تسهیلات باید بر روی یک شبکه طوری قرار گیرند تا تابع هدف که تابعی از فاصله رئوس است کمینه شود. فاصله بین دو رأس، طول کوتاه‌ترین مسیر موجود بین آن‌ها در شبکه است.

مسائل مکان‌یابی که یک مجموعه متناهی جواب‌های شدنی را دربردارند، مسائل مکان‌یابی گسسته نامیده می‌شوند و دسته سوم را تشکیل می‌دهند. بیشتر مسائل مکان‌یابی دنیای واقعی به دلیل در دسترس بودن زمینه، مقررات منطقه بندی، ملاحظات طراحی تاسیسات و... به صورت مسأله بهینه‌سازی گسسته طراحی می‌شوند. [۱]

دسته‌بندی دیگری نیز با توجه به نوع تابع هدف، برای مسائل مکان‌یابی وجود دارد:

۱- **مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود** [۲] : این مسائل در دسته مسائل حداقل مجموع قرار می‌گیرند. اما در این مسائل، هزینه، شامل هزینه ثابت نیز می‌شود و هزینه ثابت به مکانی بستگی دارد که تسهیلات در آن قرار می‌گیرد. در این مسائل ظرفیت هر مرکز نامحدود است.

۲- **مسأله مکان‌یابی تسهیلات با ظرفیت محدود** [۳] : این مسائل شبیه به مسائل مکان‌یابی تسهیلات با ظرفیت نامحدود هستند، اما ظرفیت هر کدام از مراکز محدود می‌باشد. ممکن است در این موارد جواب بهینه به گونه‌ای باشد که یک مشتری به بیش از یک منبع تأمین ارجاع داده شود.

۳- **مسأله p -میانه** [۴] : این قبیل مسائل به منظور مکان‌یابی تعداد p تسهیلات از بین نقاط ممکن برای تسهیلات، در p مکان انجام می‌شود به طوری که مجموع فواصل بین نقاط تسهیلات و نقاط مشتری‌ها حداقل شده و تمام تقاضاها برآورده شوند. این مسائل نیز جزء مسائل حداقل مجموع می‌باشد.

۴- **مسأله p -مرکز** [۵] : این مسائل برای تعیین مکان تعداد p تسهیلات به منظور حداقل کردن حداکثر فاصله هر مرکز تسهیلات تا نقطه تقاضایی که برای خدمت دادن به آن نقطه مورد تقاضا، تعیین شده است، استفاده می‌شوند.

۱-۲-۱ نمونه‌هایی از مسائل مکان‌یابی تسهیلات

یک شرکت پروژه مهمی برای کابل کشیدن در یک ناحیه وسیع را به عهده گرفته است[۴]. کابل‌ها به صورت قرقره آماده شده‌اند. این قرقره‌ها به تعداد کافی در انبارهایی ذخیره شده‌اند و به ۶ موقعیت کابل گذاری فرستاده می‌شوند.

۵ مکان برای انبارها وجود دارد که با A، B، C، D و E برچسب‌گذاری شده‌اند و شرکت می‌تواند آن‌ها را اجاره کند.

---

[۱] Fermat-Weber [۲] Uncapacitated Facility Location Problem [۳] Capacitated Facility Location Problem [۴] P-Median [۵] P-Center

هر کدام از این انبارها ظرفیت کافی برای ذخیره همه‌ی قرقره‌های مورد نیاز کل مدت پروژه را دارند. اجاره‌ی انبارها که به موقعیت مکانی وابسته است، در جدول ۱-۱ آمده است. اجاره‌هایی که در جدول آمده است، اجاره یک انبار به اندازه کافی بزرگ برای کل مدت پروژه است.

جدول ۱-۱: هزینه‌ی اجاره‌ی انبارها

انبار	A	B	C	D	E
اجاره($)	۱۲۵۰۰۰۰	۱۵۰۰۰۰۰	۱۰۰۰۰۰۰	۵۰۰۰۰۰	۷۵۰۰۰۰

مکان‌های کابل‌گذاری با اعداد ۱، ۲، ۳، ۴، ۵ و ۶ برچسب‌گذاری شده است. فاصله‌ی مکان‌های کابل‌گذاری تا مکان‌های بالقوه‌ی انبارها با شبکه‌ی جاده‌ای موجود، برحسب کیلومتر در جدول ۱-۲ آ مده است.

جدول ۱-۲: فاصله‌ی بین انبارها و مکان‌های کابل‌گذاری

	۱	۲	۳	۴	۵	۶
A	۲.۹	۲.۶	۴.۸	۴.۶	۳.۷	۲.۵
B	۲.۳	۴.۸	۴.۴	۴.۴	۳.۴	۳.۳
C	۲.۳	۲.۴	۳.۸	۲.۹	۳.۳	۴.۷
D	۳.۱	۲.۶	۴.۳	۳.۵	۴.۶	۲.۸
E	۳.۶	۲.۱	۴.۱	۴.۸	۲.۸	۲.۸

هزینه‌ی جابه‌جایی $ ۱۰۰ برای هر کیلومتر، در کل مدت پروژه فرض شده است. کابل مورد تقاضا در شش مکان ۱ تا ۶ به ترتیب ۷۰۰، ۲۳۰۰، ۵۰۰، ۳۰۰۰، ۲۰۰۰ و ۱۵۰۰ قرقره می‌باشد.

شرکت با این مسأله‌ی تصمیم رو به روست که کدام انبارها را اجاره کند و تقاضای هر مکان کابل‌گذاری از کدام انبار تأمین شود تا کل هزینه‌ها مینیمم شود؛ که هزینه‌های کلی در این حالت شامل هزینه‌ی اجاره‌ی انبارها و هزینه‌ی انتقال قرقره‌های کابل از انبارها به مکان‌های کابل‌گذاری می‌باشد.

برای مثال اگر شرکت تصمیم بگیرد که انبارهای B و C را اجاره کند، در این صورت انبار B باید تقاضای مکان‌های کابل‌گذاری ۱ و ۶ را تأمین کند و انبار C تقاضای مکان‌های کابل‌گذاری ۲، ۳، ۴ و ۵ را برآورد.

در این حالت شرکت باید $ ۲۵۰۰۰۰۰ برای اجاره و $ ۲۹۲۸۰۰۰بابت هزینه‌های انتقال بپردازد؛ به عبارت دیگر کل هزینه‌ها $ ۵۴۲۸۰۰۰ می‌باشد. مسأله‌ی شرکت برای کم کردن هزینه‌های کل با این شرایط و محدودیت‌ها، به مسأله‌ی مکان‌یابی تسهیلات بدون ظرفیت (ظرفیت نامحدود) برمی‌گردد.

مسأله‌ی کمی متفاوت‌تر زمانی رخ می‌دهد که انبارها در هر یک از مکان‌ها به اندازه‌ی کافی بزرگ نباشند. در این صورت ظرفیت انبارهای در دسترس در هر یک از مکان‌های باالقوه مشخص است. به عنوان مثال ظرفیت‌هایی که برای انبارها در مکان‌های مختلف در جدول ۱-۳ داده شده است را در نظر بگیرید.

جدول ۱-۳: ظرفیت انبارها

انبار	A	B	C	D	E
ظرفیت	۳۰۰۰	۵۵۰۰	۱۵۰۰	۲۵۰۰	۲۷۰۰

مسأله‌ی تصمیم‌گیری در مورد محل مستقر کردن انبارها و چگونگی تأمین مکان‌های کابل‌گذاری از انبارهای مستقر شده با محدودیت‌های ظرفیت انبارها را، مسأله‌ی مکان‌یابی تسهیلات با ظرفیت (ظرفیت محدود) می‌نامند.

این مسأله از مسأله‌ی مکان‌یابی تسهیلات بدون ظرفیت به دلایلی پیچیده‌تر است. توجه داشته باشید که هر مجموعه از انبارها جواب شدنی برای مسأله‌ی با ظرفیت محدود نیست. برای مثال، جوابی که برای حالت بدون ظرفیت برای اجاره‌ی انبارها فقط در B و C در نظر گرفته شد، برای مسأله با ظرفیت محدود شدنی نیست، زیرا مجموع ظرفیت این انبارها ۸۵۰۰ قرقره است، در حالی که کل تقاضای مکان‌های کابل‌گذاری ۱۰۰۰۰ قرقره می‌باشد.

به علاوه در مسأله‌ی مکان‌یابی تسهیلات با ظرفیت نامحدود، هر یک از انبارها که اجاره شده است مطمئناً همه‌ی تقاضای مکان‌های کابل‌گذاری که نزدیک‌ترین انبار به آن‌هاست را برآورده می‌کند.

با محدودیت ظرفیت، نزدیک‌ترین انبار شاید قادر به تأمین همه‌ی تقاضای مکان‌های کابل‌گذاری نباشد. به عبارت دیگر چند بخشی شدن تأمین در مسائل مکان‌یابی تسهیلات با ظرفیت محدود می‌تواند رخ دهد اما در مسائل مکان‌یابی با ظرفیت نامحدود خیر.

به عنوان مثال، فرض کنید شرکت حق انتخاب اجاره‌ی انبار در مکان‌های A و B و C را دارد. مجموع ظرفیت ۱۱۰۰۰ قرقره است و بنابراین آن‌ها می‌توانند تقاضای همه‌ی ۶ مکان کابل‌گذاری را برآورند. مجموع هزینه‌ای که شرکت باید برای اجاره‌ی انبارها بپردازد $ ۳۷۵۰۰۰۰ است. تقاضای ۶ مکان کابل‌گذاری باید مطابق آن‌چه در جدول ۱-۴ نشان داده شده است تقسیم‌بندی شود و کل هزینه‌ی انتقال $ ۳۳۴۵۰۰۰ است. مجموع هزینه‌های شرکت در این وضعیت، $ ۸۱۸۵۰۰۰ می‌باشد، که این یک جواب بهینه برای مسأله شرکت نیست.

جدول ۱-۴: تقسیم تقاضای ۶ مکان وقتی A، B و C انبار باشند

	۱	۲	۳	۴	۵	۶
A		۱۰۰۰	۵۰۰			۱۵۰۰
B	۷۰۰	۱۳۰۰		۱۵۰۰	۲۰۰۰	
C				۱۵۰۰		

در ادامه دو مسأله‌ی مکان‌یابی دیگر را توصیف می‌کنیم که دسته‌بندی ۴ گونه مسائل مکان‌یابی تسهیلات که مطرح شد را کامل می‌کند.

فرض کنید یک سازمان تحقیقات در ۷ ساختمان مختلف با برچسب‌های A، B، C، D، E، F و G در قسمت‌های مختلف شهر مستقر شده است. این ساختمان‌ها به ترتیب ۱۲، ۱۷، ۱۵، ۴۰، ۳۰، ۲۵ و ۱۸ پژوهشگر را در خود جای داده‌اند که همه‌ی آن‌ها احتیاج به دسترسی به امکانات کتابخانه‌ای را دارند.

سازمان در نظر دارد امکانات کتابخانه‌ای را در دو ساختمان از آن ۷ ساختمان بسازد و همه‌ی پژوهشگران می‌خواهند که به یکی از این امکانات دسترسی داشته باشند. البته اجرای این تصمیم نیازمند دانستن فاصله‌ی بین هر جفت از ساختمان‌ها در شبکه‌ی جاده‌ای شهر می‌باشد. این فاصله‌ها در جدول ۱-۵ داده شده است.

جدول ۱-۵: فاصله‌ی بین هر جفت از ساختمان‌ها

G	F	E	D	C	B	A	
۴.۶	۸.۵	۳.۸	۴.۷	۵.۷	۵.۹	_	A
۱.۴	۳.۴	۲.۹	۲.۹	۴.۰	_		B
۵.۴	۵.۷	۶.۹	۱.۱	_			C
۴.۳	۶.۳	۵.۸	_				D
۴.۳	۶.۳	_					E
۴.۴	_						F

دو راه اصلی برای تصمیم‌گیری سازمان در مورد مکان مستقر کردن امکانات کتابخانه‌ای وجود دارد. یک راه برای توزیع منصفانه‌ی امکانات کتابخانه‌ای، مینیمم کردن مجموع مسافت‌هایی است که پژوهشگران برای دسترسی به امکانات طی می‌کنند.

اگر تعداد پژوهشگران در هر ساختمان در مفروضات داده نمی‌شد، آن‌گاه با مسأله‌ی ۲-میانه روبه‌رو بودیم. اما اگر فاصله‌ها وزن‌دار باشند آن‌گاه ما مسأله‌ی ۲-میانه‌ی وزن‌دار را داریم. مجموعه‌ی وزن‌های استدلالی در این حالت متناسب با تعداد پژوهشگرانی است که در هر ساختمان کار می‌کنند.

فرض کنید سازمان تصمیم بگیرد که امکانات کتابخانه‌ای را در ساختمان‌های B و C دائر کند. کتابخانه‌ای که در ساختمان B است باید پژوهشگران ساختمان‌های B، E، F و G را تأمین کند در حالی که کتابخانه‌ای که در ساختمان C است باید پژوهشگران ساختمان‌های A، C و D را تأمین کند.

در حالت بدون وزن، مجموع فاصله‌ها برای این جواب ۱۴/۵ کیلومتر خواهد بود ، در حالی که اگر فاصله‌ها با تعداد پژوهشگران هر ساختمان وزن‌دار شده باشد، مجموع فاصله‌ها ۳۰۹/۶ کیلومتر خواهد بود. هدف این سازمان در مسأله ۲-میانه، مکان‌یابی کتابخانه‌هاست به طوری که این مقادیر کمینه شود.

یک راه دیگر برای تصمیم‌گیری این است که مطمئن شویم ماکزیمم فاصله‌ای که پژوهشگران برای دسترسی به نزدیک‌ترین امکانات کتابخانه‌ای باید طی کنند، کمترین مقدار ممکن باشد. اگر تعداد پژوهشگران در هر ساختمان در مفروضات مسأله داده نشده باشد، آن‌گاه با مسأله‌ی ۲-مرکز روبه‌رو هستیم. اما اگر ما بخواهیم فاصله‌ای که پژوهشگران هر ساختمان باید طی کنند را وزن‌دار کنیم، احتمالاً با وزن‌هایی که مطابق تعداد پژوهشگران در هر ساختمان می‌باشد، آن‌گاه مسأله به عنوان یک مسأله‌ی ۲-مرکز وزن‌دار شناخته می‌شود.

اگر سازمان تصمیم بگیرد که امکانات کتابخانه‌ای را در ساختمان‌های B و C مستقر کند، آن‌گاه ماکزیمم فاصله‌ای که هر پژوهشگر برای استفاده از امکانات کتابخانه‌ای باید طی کند برابر است با ۵/۷ کیلومتر در حالت بدون وزن و ۸۷ کیلومتر

در حالتی که فاصله‌ها با تعداد پژوهشگران هر ساختمان وزن‌دار شده باشد. هدف سازمان در مسأله‌ی ۲-مرکز، مستقر کردن کتابخانه‌هاست به طوری که این مقادیر کمینه شوند.

البته مسائلی که در بالا شرح داده شد، می‌تواند به صورت مسائل P -میانه (وزن‌دار) و P -مرکز (وزن‌دار) تعمیم داده شوند، اگر با وضعیتی سر و کار داشته باشیم که بخواهیم P ($p \geq ۱$) تسهیلات را مکان‌یابی کنیم. جواب مسائل P -میانه و P -مرکز به ترتیب P -میانه و P -مرکز شبکه نامیده می‌شوند.

۱-۲-۲ خصوصیات مسائل مکان‌یابی

ما در این بخش ۴ دسته از مسائل مکان‌یابی را معرفی کردیم. جدول ۱-۶ خصوصیات هر کدام از این مسائل را قید کرده است.

جدول ۱-۶: خصوصیات مسائل مکان‌یابی

	فاصله‌ها	هزینه‌های راه‌اندازی	ظرفیت‌ها	هدف
مکان‌یابی بدون ظرفیت	دارد	دارد	ندارد	مینیمم کردن مجموع هزینه‌ها
مکان‌یابی با ظرفیت	دارد	دارد	دارد	مینیمم کردن مجموع هزینه‌ها
P-میانه	دارد	ندارد	ندارد	مینیمم کردن مجموع فاصله‌ها
P-مرکز	دارد	ندارد	ندارد	مینیمم کردن ماکزیمم فاصله

۱-۳ مسأله مکان‌یابی تسهیلات با ظرفیت نا محدود

تمرکز اصلی ما در این تحقیق پیرامون مسأله مکان‌یابی تسهیلات با ظرفیت نا محدود می‌باشد، لذا در این بخش به معرفی کلی این‌گونه مسائل می‌پردازیم.

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 مسائل مکان‌یابی مورد ملاحظه است و به اختصار با UFLP نمایش داده می‌شود.

UFLP هم‌چنین با عنوان مسأله مکان‌یابی تسهیلات ساده [۱] (SFLP) ومسأله مکان‌یابی کارخانه [۲] (PLP) ویا مسأله مکان‌یابی انبار [۳] (WLP) نیز شناخته می‌شود[۲].

همان‌طور که در دسته‌بندی گفته شد، مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود شامل مکان‌یابی کردن تعداد نامشخصی تسهیلات می‌باشد به گونه‌ای که مجموع هزینه‌های ثابت راه‌اندازی و هزینه‌های متغیر برآوردن تقاضای بازار

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

از این تسهیلات حداقل شود. نقاط تسهیلات و مناطق مشتری، نقاط مجزا روی صفحه در نظر گرفته می‌شوند. این طور فرض شده است که مکان‌های جایابی تسهیلات از پیش تعیین شده‌اند و تقاضا در هر منطقه مشتری در نقطه نمایندگی آن منطقه متمرکز شده است. تعداد مراکزی که باید استقرار یابند از پیش تعیین شده نیست، اما به گونه‌ای معین می‌شوند که هزینه را کمینه کنند. در این گونه مسائل ظرفیت هر مرکز تسهیلات نامحدود در نظر گرفته می‌شود، تخصیص یک تقاضا به بیش از یک نقطه تأمین، هرگز سود بخش نخواهد بود. [۳]

۴-۱ مدل‌سازی مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود

در UFLP یک مجموعه I از مشتری‌ها داریم که باید از مراکز تسهیلات که در آن‌ها نوعی خدمات عرضه می‌شود، تأمین شوند و یک مجموعه گسسته J که مکان‌های ممکن داده شده تسهیلات است.

هزینه‌ای ثابت برای باز کردن تسهیلات در مکان $j \in J$ در نظر گرفته شده که با f_j نشان داده می‌شود و هزینه تأمین همه تقاضای مشتری $i \in I$ از تسهیلات $j \in J$ ثابت در نظر گرفته شده و با c_{ij} نشان داده می‌شود(این هزینه می‌تواند شامل هزینه‌های تولید وهزینه‌های حمل‌ونقل باشد).

مسأله عبارت است از یافتن تعداد و مکان تسهیلات برای بهره‌برداری و هم‌چنین تخصیص مشتری‌ها به تسهیلات به منظور به حداقل رساندن هزینه‌های کل. برای مدل‌سازی مسأله به دو نوع متغیر تصمیم نیاز داریم، نوع اول شامل متغیرهای دودویی y_j برای هر $j \in J$ که اگر تسهیلات در مکان j احداث شود مقدار ۱ و در غیر این صورت مقدار ۰ می‌گیرد و همچنین متغیرهای پیوسته و نامنفی x_{ij} که مربوط به کسری از تقاضای مشتری i است که توسط تسهیلات j برآورده می‌شود. به دلیل نامحدود بودن ظرفیت مراکز تسهیلات، هر مشتری می‌تواند تمام تقاضای خود را از یک مرکز تسهیلات برآورد و لذا x_{ij} مقدار ۰ یا ۱ می‌گیرد. اگر $x_{ij} = ۱$ باشد یعنی، تمام تقاضای مشتری i از مرکز j تأمین می‌شود.

$$\min f(x,y) = \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} + \sum_{j\in J} f_j y_j$$

s.t.

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

$$۰ \leqslant x_{ij} \leqslant y_j \qquad i \in I \;\; j \in J$$

$$y_j \in \{۰, ۱\} \qquad j \in J$$

محدودیت اول این را بیان می‌کند که تقاضای هر مشتری باید کاملاً برآورده شود ومفهوم محدودیت دوم این است که تقاضاها فقط می‌توانند از تأسیساتی که باز هستند تأمین شوند. این فرمول‌بندی برای UFLP معمولاً فرمول‌بندی قوی نامیده

می‌شود. با جایگزین کردن محدودیت دوم با مجموعه محدودیت‌های فشرده زیر فرمول‌بندی دیگری برای UFLP به‌دست می‌آید:

$$\sum_{i \in I} x_{ij} \leqslant |I|.y_j \qquad j \in J$$

برای این که ببینیم چگونه این دو محدودیت را می‌توان با هم جایگزین کرد، توجه کنید که اگر $y_j = ۰$ باشد، هر دو فرمول‌بندی ایجاب می‌کنند که $x_{ij} = ۰$ برای همه $i \in I$ و اگر برای $i \in I$ و $j \in J$ ، $x_{ij} > ۰$ باشد، در نتیجه y_j باید مقدار ۱ را داشته باشد. [۱]

فصل ۲

روش‌های بهینه‌سازی

۲-۱ مقدمه

مسائل بهینه‌سازی اهمیت زیادی در اکثر شاخه‌های علوم دارند و هدف یافتن بهترین جواب ممکن برای مسأله‌ای مشخص است. بهینه‌سازی یک فعالیت مهم و تعیین‌کننده در طراحی ساختاری است. طراحان زمانی قادر خواهند بود طرح‌های بهتری تولید کنند، که بتوانند با روش‌های بهینه‌سازی در مصرف زمان و هزینه طراحی صرفه‌جویی نمایند. بسیاری از مسائل بهینه‌سازی در مهندسی، طبیعتاً پیچیده‌تر و مشکل‌تر از آن هستند که با روش‌های مرسوم بهینه‌سازی نظیر روش برنامه‌ریزی ریاضی و نظایر آن قابل حل باشند. روش‌های متفاوتی برای حل یک مسأله بهینه‌سازی وجود دارد. برخی از این روش‌ها از رویه‌های طبیعی الهام گرفته‌اند. این روش‌ها معمولاً با مجموعه اولیه‌ای از متغیرها آغاز می‌شوند و سپس جلو می‌روند تا زمانی که حداقل یا حداکثر مطلق تابع هدف به‌دست آید. به عبارت دیگر، بهینه‌سازی روند تنظیم ورودی‌ها، مشخصه‌های یک وسیله، فرآیند ریاضیاتی یا آزمایش برای یافتن کمینه یا بیشینه خروجی و نتیجه است. ورودی از یک سری متغیرها و تابع هدف یا تابع برازندگی تشکیل شده است؛ و خروجی نیز هزینه یا برازندگی است.

۲-۱-۱ تعاریف

تعریف ۲-۱-۱. یک مسأله‌ی بهینه‌سازی دارای یک مجموعه دامنه D و تابع مقدار به صورت$f : D \to R$ می‌باشد. $f(x) \in R$ مشخص کننده‌ی مقدار سود یا هزینه‌ای است که توسط $x \in D$ مشخص می‌شود و به آن **تابع هدف** ، تابع هزینه یا تابع انرژی گفته می‌شود.

تعریف ۲-۱-۲. هر مقداردهی از متغیرهای تصمیم در دامنه آن‌ها را یک **جواب** گویند.

تعریف ۲-۱-۳. هر جواب که در مجموعه قیود صدق کند را یک **جواب شدنی** گویند.

تعریف ۲-۱-۴. هر جواب‌شدنی راکه در بین همه جواب‌های شدنی مقدار تابع هدف را بهینه کند، یک **جواب بهینه** گویند.

تعریف ۲-۱-۵. مرتبه زمانی یک الگوریتم متناسب با تعداد دستورالعمل‌های اجرایی آن است و با نماد O(.) نمایش داده می‌شود.

تعریف ۲-۱-۶. مسائلی که هیچ الگوریتم شناخته شده‌ای با مرتبه زمانی چند جمله‌ای برای حل آن‌ها تا کنون ارائه نشده است، مسائل **NP-Hard** و یا NP –سخت نامیده می‌شوند.

مسائل زمان‌بندی، تخصیص، مسیریابی و مکان‌یابی نمونه‌هایی از مسائل NP –سخت می‌باشند.

۲-۱-۲ مسائل بهینه‌سازی ترکیبیاتی

در دهه‌های ۵۰ و ۶۰ بهینه‌سازی بیشتر به مدل‌های خطی و غیرخطی با متغیرهای پیوسته مربوط می‌شد. بهینه‌سازی ترکیبیاتی که مربوط به مسائل بهینه‌سازی با متغیرهای گسسته می‌شود، در دهه ۷۰ مورد توجه قرار گرفت. بهینه‌سازی ترکیبیاتی [۱] جستجو برای یافتن جواب بهینه برای توابع با متغیرهای گسسته می‌باشد. بهینه‌سازی گسسته مسائل عملی از محیط زیست، تجدیدپذیری انرژی، توزیع، طراحی زیرساخت، ارتباطات و بهره‌وری در بخش‌های تولیدی و خدماتی را پوشش می‌دهد. یک مسأله بهینه‌سازی گسسته، مسأله‌ای است که در آن متغیرهای مسأله در یک بازه معین تغییرات گسسته دارند. راه‌های زیادی برای توصیف یک مسأله بهینه‌سازی گسسته وجود دارد. در عمومی‌ترین شکل آن می‌توان به مجموعه نمونه مسائلی گفت که با جفت (S, f) و نگاشت $f : S \longrightarrow R^+$ مشخص می‌شوند که S مجموعه‌ای از کاندیدای محدود تعریف شده در فضای جستجو و f تابع هزینه یا هدف می‌باشد، که هدف مسأله در حالت مینیمم‌سازی، پیدا کردن $s^* \in S$ که $f(s^*) \leq f(s)$ برای همه $s \in S$ می‌باشد که s^* جواب بهینه سراسری برای مسأله داده شده است.

روش های بهینه سازی درفضای گسسته، خود نوعی روش جستجو در فضای جواب گسسته هستند. روش های متفاوتی برای حل یک مسأله بهینه‌سازی ترکیبیاتی وجود دارد. روش‌ها والگوریتم‌های بهینه‌سازی به دو دسته روش‌ها یا الگوریتم‌های دقیق [۲] و روش‌ها یا الگوریتم‌های تقریبی [۳] تقسیم‌بندی می‌شوند.

۲-۲ روش‌های دقیق

الگوریتم‌های دقیق قادر به یافتن جواب بهینه به صورت دقیق هستند، اما در مورد مسائل بهینه‌سازی سخت کارایی کافی ندارند و زمان اجرای آن‌ها متناسب با ابعاد مسائل به صورت نمایی افزایش می‌یابد.

---

[۱] combinational optimization [۲] exact [۳] approximate

یک روش ابتدایی برای حل مسائل بهینه‌سازی ترکیبی، شمارش کامل است که در آن برای تمامی ترکیب‌های ممکن مقدار تابع هدف محاسبه شده و در نهایت بهترین جواب موجه انتخاب می‌شود. هرچند این شیوه در نهایت به جواب دقیق مسأله می‌رسد، اما به دلیل زیاد بودن ترکیبات ممکن در مسائل واقعی، عملاً استفاده از آن غیرممکن است. با توجه به مشکلات مربوط به روش شمارش کامل، تلاش بسیاری برای توسعه روش‌های مؤثرتر و کاراتر صورت گرفته است. در همین راستا الگوریتم‌های مختلفی به وجود آمده است که مشهورترین نمونه‌های آن الگوریتم شاخه‌وکران و شمارش ضمنی است.

۲-۳ روش‌های تقریبی

الگوریتم‌های تقریبی قادر به یافتن جواب‌های مطلوب (نزدیک به بهینه) در زمان حل کوتاه برای مسائل بهینه‌سازی سخت هستند. الگوریتم‌های تقریبی به سه دسته الگوریتم‌های ابتکاری [۱] ، فراابتکاری [۲] و فوق‌ابتکاری [۳] بخش‌بندی می‌شوند. مشکل اصلی الگوریتم‌های ابتکاری، گیر افتادن آن‌ها در نقاط بهینه محلی و همگرایی زودرس به این نقاط است. برای بهبود این الگوریتم‌ها از اواسط دهه هفتاد، موج تازه‌ای از رویکردها آغاز گردید. این رویکردها شامل الگوریتم‌هایی است که صریحاً یا به صورت ضمنی تقابل بین ایجاد تنوع جستجو (وقتی علائمی وجود دارد که جستجو به سمت مناطق بد فضای جستجو می‌رود) و تشدید جستجو (با این هدف که بهترین جواب در منطقه مورد بررسی را پیدا کند) را مدیریت می‌کنند. این الگوریتم‌ها متاهیوریستیك یا فراابتکاری نامیده می‌شوند. در واقع الگوریتم‌های فراابتکاری، یکی از انواع الگوریتم‌های بهینه‌سازی تقریبی هستند که دارای راه‌کارهای برون رفت از نقاط بهینه محلی هستند و قابلیت کاربرد در طیف گسترده‌ای از مسائل را دارند. رده‌های گوناگونی از الگوریتم‌های فراابتکاری در دهه‌های اخیر به صورت زیر توسعه یافته است.

مبتنی بر یک جواب و مبتنی بر جمعیت: الگوریتم‌های مبتنی بر یک جواب، در حین فرآیند جستجو یک جواب را تغییر می‌دهند مانند الگوریتم جستجوی ممنوعه، در حالی که در الگوریتم‌های مبتنی بر جمعیت در حین جستجو، یک جمعیت از جواب‌ها در نظر گرفته می‌شوند مانند الگوریتم شبیه‌سازی تبریدی.

الهام گرفته شده از طبیعت و بدون الهام از طبیعت: بسیاری از الگوریتم‌های فراابتکاری همچون الگوریتم ژنتیک و الگوریتم اجتماع مورچگان از طبیعت الهام گرفته شده‌اند و برخی از الگوریتم‌های فراابتکاری مانند جستجوی ممنوعه الهام گرفته شده از طبیعت نیستند.

با حافظه و بدون حافظه: برخی از الگوریتم‌های فراابتکاری فاقد حافظه‌اند، به این معنا که، این نوع الگوریتم‌ها از اطلاعات به‌دست آمده در حین جستجو استفاده نمی‌کنند، مانند الگوریتم شبیه‌سازی تبریدی. این در حالی است که برخی از الگوریتم‌های فراابتکاری نظیر جستجوی ممنوعه از حافظه استفاده می‌کنند. این حافظه اطلاعات به‌دست 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 شبیه‌سازی تبریدی، یک سری قوانین احتمالی در حین جستجو

---

[۱] heuristic [۲] meta-heuristic [۳] hyper-heuristic

مورد استفاده قرار می‌گیرد.

۲-۳-۱ الگوریتم‌های فراابتکاری

روش‌های دقیق برای مسائل با ابعاد بزرگ کارایی خود را از دست می‌دهند و عملکردی بهتر از شمارش کامل نخواهند داشت. به این دلیل، در سال های اخیر، استفاده از الگوریتم‌های فراابتکاری برای حل مسائل گوناگون جستجو و بهینه‌سازی‌های پیوسته و گسسته، به طور شگفت‌آوری رشد داشته است.

الگوریتم‌های شبیه‌سازی تبریدی، ژنتیک، بهینه‌سازی ازدحام ذرات، جستجوی ممنوعه و کلونی زنبور مصنوعی نمونه‌هایی از این دسته الگوریتم‌ها هستند که برای حل مسأله مکان‌یابی نیز به کار برده شده‌اند. مهمترین عاملی که سبب شده است استفاده از این الگوریتم‌ها به این اندازه مورد توجه جوامع علمی مختلف باشد، سادگی اعمال شدن و کاربرد آن‌ها برای حل مسائل با ماهیت بسیار متفاوت است. در این روش‌ها با جستجو در بین جواب‌های ممکن، جوابی نزدیک به بهینه در زمانی مطلوب برای یک مسأله ارائه می‌شود. معمولاً هیچ تضمینی برای بهینه بودن جواب وجود ندارد و حتی نمی‌توان میزان نزدیکی جواب به دست آمده به جواب بهینه را نیز تعیین کرد. لیکن توانایی بی‌بدیل این الگوریتم‌ها در کشف پاسخ‌های نزدیك به بهینه در زمانی نسبتاً کوتاه، موجبات شهرت فراوان آنها را فراهم ساخته است. با رجوع به مقالات و پژوهش‌های جدید، مشاهده می‌شود که تعداد الگوریتم‌های فراابتکاری مبتنی بر رفتار زیست‌شناختی موجودات رشد چشم‌گیری داشته است. الگوریتم‌های مبتنی بر زندگی کرم‌های شب‌تاب، گربه‌سانان، مگس‌های میوه، فاخته‌ها، خفاش‌ها و میگوها تنها چند نمونه از فراابتکاری‌های جدید ارائه شده هستند. این الگوریتم‌ها، در پروژه‌های تحقیقاتی، پایان‌نامه‌های کارشناسی ارشد و رساله‌های دکتری خیلی از رشته‌ها از قبیل مهندسی کامپیوتر، برق، مکانیك، عمران، صنایع، ریاضیات، فیزیك، شیمی، کشاورزی، اقتصاد، مدیریت و غیره به طور گسترده‌ای مورد استفاده قرار گرفته‌اند. [۵]

۲-۳-۲ الگوریتم‌های تکاملی

الگوریتم‌های تکاملی، دسته‌ای از الگوریتم‌های فراابتکاری هستند که، مبتنی بر قانون بقاء اصلح نظریه تکاملی داروین می‌باشند. تکامل یک فرآیند بهینه‌سازی و تطبیق‌پذیری موجودات زنده با تغییرات محیط است. هدف از این فرآیند، بهبود توانایی یک ارگانیزم برای نجات در یک محیط متغیر و پویا می‌باشد. این الگوریتم‌ها مبتنی بر جمعیت، الهام گرفته شده از طبیعت، با حافظه و احتمالی هستند.

الگوریتم ژنتیک(GA) یکی از تکنیک‌های محبوب در زمینه پژوهش‌های محاسبات تکاملی بوده است. الگوریتم ژنتیک از عواملی بهره می‌برد که توسط تنوع ژنتیک طبیعی و گزینش طبیعی الهام بخشیده شده است. مثال دیگر، بهینه‌سازی ازدحام ذرات(PSO) که از رفتار جمعی دسته پرندگان، یا حرکت جمعی ماهی‌ها، الهام گرفته شده است. بهینه‌سازی کلونی مورچگان(ACO) یک الگوریتم تکاملی بهینه‌سازی دیگر است که از برجا گذاشتن اثر فرومون در کلونی مورچه‌ها، الهام

گرفته شده است. در کنار این روش‌های شناخته شده، بررسی‌ها روی الگوریتم‌های بهینه‌سازی الهام گرفته از طبیعت، هم‌چنان در حال انجام است و روش‌های جدید دائماً در حال گسترش‌اند تا نوعی از مسائل غیرخطی را حل کنند. [۱۲]

در ادامه به معرفی الگوریتم ژنتیک و الگوریتم بهینه‌سازی ازدحام ذرات می‌پردازیم که نمونه‌هایی از الگوریتم‌های فراابتکاری تکاملی هستند و قصد داریم نتایج این الگوریتم‌ها را با نتایج الگوریتم بهینه‌سازی فاخته در حل مسائل مکان‌یابی تسهیلات با ظرفیت نامحدود مقایسه نماییم.

۲-۴ الگوریتم ژنتیک(GA)

الگوریتم ژنتیک [۱] یک الگوریتم جستجوی تصادفی است که بر اساس قوانین تکامل طبیعی عمل می‌کند. این الگوریتم اولین بار توسط جان هولند [۲] [۷] در سال ۱۹۷۵ ابداع شد و توسط گلدبرگ [۳] [۸] توسعه داده شد، که در سال‌های اخیر به طور گسترده‌ای برای مسائل بهینه‌سازی و جستجو به کار گرفته شده است.

در طبیعت، فرآیند تکامل هنگامی ایجاد می‌شود که شرایط زیر برقرار باشد:

۱- موجود توانایی تکثیر خود را داشته باشد.

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

۳- تنوع در بین این موجودات وجود داشته باشد.

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

اصول اولیه این الگوریتم از علم ژنتیک اقتباس شده است، علم ژنتیک، درباره چگونگی توارث و انتقال صفحات بیولوژیکی از نسلی به نسل بعد صحبت می‌کند. عامل اصلی انتقال صفحات بیولوژیکی در موجودات زنده کروموزم[۴] و ژن‌ها[۵] هستند. نحوه‌ی عملکرد آن‌ها به گونه‌ای است که در نهایت ژن‌ها و کروموزم‌های برتر و قوی‌تر باقی می‌مانند و ژن‌های ضعیف‌تر از بین می‌روند. به عبارت دیگر نتیجه عملیات متقابل ژن‌ها و کروموزم‌ها باقی ماندن موجودات اصلح و برتر است.

اساس این الگوریتم قانون تکامل داروین[۶](۱۸۵۹) است که می‌گوید موجودات ضعیف‌تر از بین می‌روند و موجودات قوی‌تر باقی می‌مانند. قانون انتخاب[۷] طبیعی بدین صورت است که تنها گونه‌هایی از یک جمعیت[۸] ادامه نسل[۹] می‌دهند که بهترین خصوصیات را داشته باشند و آن‌هایی که این خصوصیات را نداشته باشند به تدریج و در طی زمان از بین می‌روند.

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

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

---

[۱]Genetic Algorithm [۲]John Holland [۳]Goldberg [۴]Chromosome [۵]Genes [۶]Darvin
[۷]Selection [۸]Population [۹]Generation

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

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

جدول ۲-۱: رابطه بین الگوریتم ژنتیک و طبیعت

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

۲-۴-۱ تعاریف

تعریف ۲-۴-۱. اساس الگوریتم ژنتیک تبدیل هر مجموعه جواب به یک کدینگ[1] است. این کدینگ را اصطلاحاً **کروموزوم** نامند. به کروموزوم، فرد یا رشته نیز گفته می‌شود.

تعریف ۲-۴-۲. عناصر تشکیل دهنده یک کروموزوم که معمولاً اعداد هستند را **ژن** می‌گویند.

تعریف ۲-۴-۳. محل قرار گرفتن ژن در کروموزوم را یک **مکان** می‌گویند.

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

تعریف ۲-۴-۵. هر تکرار از الگوریتم را یک **نسل** می‌نامند.

۲-۴-۲ ساختار کلی الگوریتم ژنتیک

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

در هر تکرار هر یک از رشته‌های موجود در جمعیت، رمزگشایی شده و مقدار تابع هدف برای آن بدست می‌آید. بر اساس مقادیر بدست آمده تابع هدف در جمعیت رشته‌ها، به هر رشته یک عدد برازندگی نسبت داده می‌شود. این عدد برازندگی

[1] Encoding

احتمال انتخاب را برای هر رشته تعیین خواهد کرد. بر اساس این احتمال انتخاب، مجموعه‌ای از رشته‌ها به عنوان والدین انتخاب شده و با اعمال عمل ترکیب[1] بر روی آن‌ها چند جواب ممکن یا فرزند برای جمعیت جدید تولید می‌کند.

در حین اجرای الگوریتم جهش‌ها[2] و مهاجرت‌هایی به طور متناوب برای بهبود انجام می‌شود، علاوه بر این تعدادی از جواب‌های بد از جمعیت جاری خارج می‌شوند تا تعداد رشته‌ها در تکرارهای محاسباتی مختلف ثابت باشد. این فرآیند ادامه دارد تا اینکه شرط پایان برقرار شود.

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

۲-۴-۳ کاربردهای الگوریتم ژنتیک

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

جدول ۲-۲: کاربردهای الگوریتم ژنتیک

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

۲-۴-۴ عملگرهای الگوریتم ژنتیک

عمگرهای الگوریتم ژنتیک به صورت زیر است:

۲-۴-۴-۱ کدگذاری

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

[1] Crossover [2] Mutation

۲-۴-۴-۲ ارزیابی جواب‌های هر نسل

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

تعریف ۲-۴-۶. تابعی که بر اساس آن میزان سازگاری کروموزوم با محیط ارزیابی می‌شود را **تابع برازندگی**[۱] گوییم.

تابع ذکر شده در تعریف را می‌توان از اعمال تبدیل مناسب بر روی تابع هدف یعنی تابعی که قرار است بهینه شود، بدست می‌آورد. هر چه کیفیت رشته بالاتر باشد مقدار برازندگی جواب بیشتر است و احتمال مشارکت برای نسل بعدی نیز افزایش خواهد یافت.

۲-۴-۴-۳ روش‌های انتخاب

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

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

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

۲-۴-۴-۴ ترکیب

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

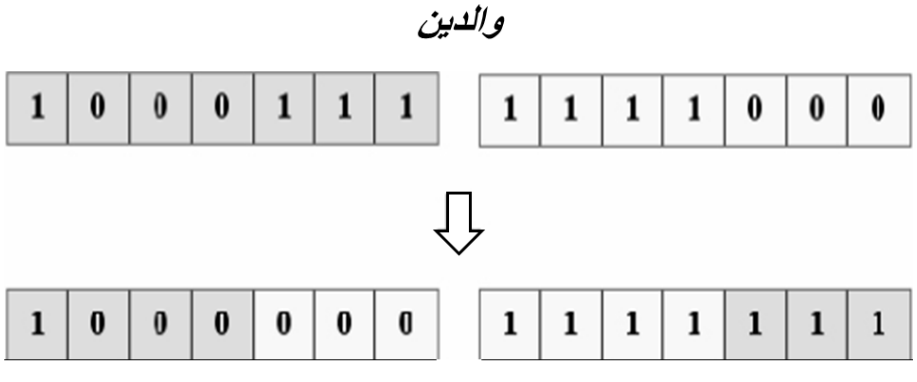
ترکیب یک نقطه‌ای: ابتدا عددی تصادفی در فاصله ۱تا طول کروموزوم تولید می‌شود، سپس دو کروموزوم از این نقطه تا انتهای دو عبارت شکسته شده و با هم ترکیب می‌شوند. شکل ۲-۱ ترکیب یک نقطه‌ای را نشان می‌دهد.

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

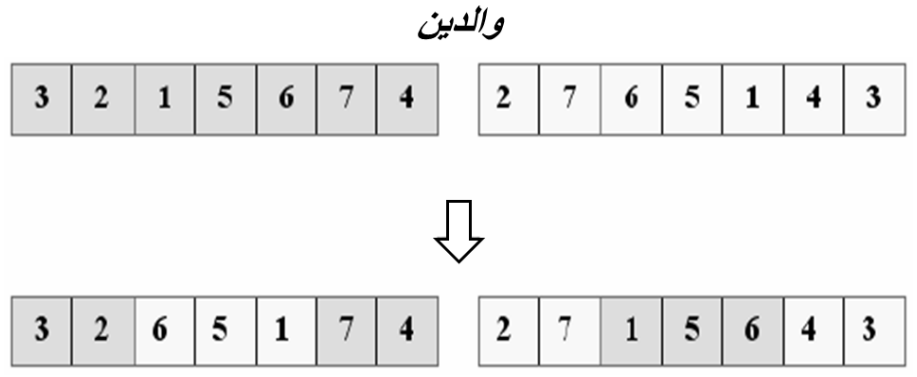
ترکیب یکنواخت: یک کروموزوم تصادفی هم طول با کروموزوم‌های موجود تولید می‌شود. بر اساس این عملگر، یک ژن از هر دو والد شانس برابر برای حضور در کروموزوم یک فرزند را دارند. کروموزوم تولید شده تعیین می‌کند که کدام ژن از

---

[۱] Fitness [۲] Roulette Wheel [۳] Ranking



شکل ۲-۱: ترکیب یک نقطه‌ای



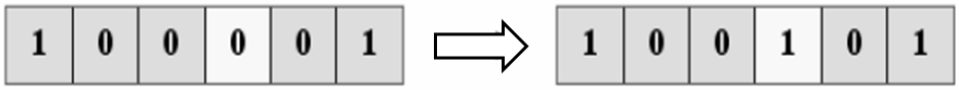
شکل ۲-۲: ترکیب دو نقطه‌ای

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

۲-۴-۴-۵ جهش

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

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



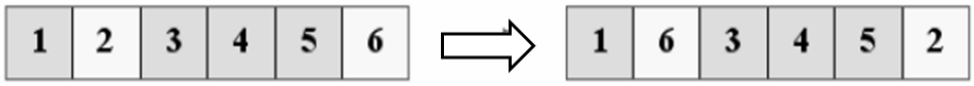
شکل ۲-۳: جهش ژنی

الگوریتم ۲-۱ شبه کد الگوریتم ژنتیک

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

۱: جمعیت n کروموزومی به طور تصادفی ایجاد کنید.
۲: برازندگی هر کروموزوم در جمعیت را ارزیابی کنید.
۳: مراحل زیر را تا زمانیکه شرط خاتمه برقرار شود انجام دهید:
۴: جمعیت جدیدی را تشکیل دهید:
۵: والدین را از میان جمعیت انتخاب کنید و با توجه به احتمال ترکیب، برای تشکیل فرزندان جدید آن‌ها را ترکیب کنید.
۶: با توجه به احتمال جهش فرزندان را مورد جهش قرار دهید.
۷: فرزندان جدید را در جمعیت بگنجانید.
۸: برازندگی هر کروموزوم را ارزیابی کنید.

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



شکل ۲-۴: جهش به روش جابه‌جایی

۲-۴-۴-۶ رمزگشایی (دی‌کدینگ)

رمزگشایی[1] عکس عمل کدگذاری است. در این مرحله بعد از اینکه الگوریتم بهترین جواب را برای مسأله ارائه کرد لازم است عکس عمل رمزگذاری روی جواب‌ها اعمال شود.

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

[1] Decoding

۲-۵ الگوریتم بهینه‌سازی ازدحام ذرات (PSO)

الگوریتم بهینه‌سازی ازدحام ذرات[۱] در سال ۱۹۹۵ توسط ابرهارت[۲] و کندی[۳] [۹] ابداع شد و ایده‌ی اصلی آن از حرکت پرندگان و ماهی‌ها گرفته شده است. به دلیل همگرایی سریع، محاسبات ساده و فهم آسان، PSO کاربرد وسیعی در مسائل بهینه سازی پیوسته مانند کنترل ولتاژ و توان دارد و اخیراً برای مسائل بهینه‌سازی گسسته مانند فروشنده دوره گرد، رنگ‌آمیزی گراف، نیز توسعه داده شده است. عملکرد این الگوریتم به پارامترهای آن بستگی دارد و ممکن است جواب در بهینه محلی به دام افتد، بنابراین محققان روش‌های زیادی را جهت حل این مشکل پیشنهاد داده‌اند.

تعریف ۲-۵-۱. هر جواب مسأله در الگوریتم بهینه‌سازی ازدحام ذرات را **ذره** نامیم .

الگوریتم بهینه‌سازی ازدحام ذرات با یک جمعیت اولیه از ذرات شروع می‌شود، هر ذره یک بردار موقعیت[۴] دارد که برازندگی بر اساس آن بدست می‌آید و یک بردار سرعت که جهت حرکت ذره را نشان می‌دهد. در هر تکرار مقدار برازندگی هر ذره محاسبه می‌شود و سرعت ذره بر اساس بهترین موقعیت ذره تاکنون $(pbest)$ و بهترین موقعیت در همسایگی ذره $(nbest)$ به‌هنگام می‌شود، زمانی‌که تمام جمعیت به عنوان همسایگی ذره در نظر گرفته شود بهترین همسایگی $gbest$ نامیده می شود. در واقع هر ذره در فضای جواب با تغییر جهت و سرعت بر اساس تجربه خود و همسایگانش حرکت می‌کند. بنابراین موقعیت ذرات دیگر روی چگونگی جستجوی یک ذره اثر می‌گذارد.

۲-۵-۱ معرفی پارامترها

پارامترهای لازم برای الگوریتم بهینه‌سازی ازدحام ذرات را در جدول ۲-۳ درج می‌نماییم.

جدول ۲-۳: معرفی پارامترها

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

۲-۵-۲ عملکرد الگوریتم بهینه‌سازی ازدحام ذرات

عملکرد الگوریتم بهینه‌سازی ازدحام ذرات برای حل مسائل مختلف به صورت زیر است:

---

[۱] Particle Swarm Optimization [۲] Eberhart [۳] Kennedy [۴] Position

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

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

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

به‌هنگام‌سازی سرعت و موقعیت به ترتیب با فرمول‌های زیر انجام می‌شود:

$$v_{id}^{t+1} = v_{id}^{t} + c_1 r_1 (b_{id}^{t} - x_{id}^{t}) + c_2 r_2 (b_{id}^{nt} - x_{id}^{t}) \qquad (۱-۲)$$

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

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

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

در معادله ۲-۱ ، c_1 و c_2 ثابت‌های مثبت هستند، c_1 مربوط به تجارب شخصی هر ذره و c_2 مربوط به تجارب جمع می‌باشد. این دو فاکتور تأثیر $pbest$ و $nbest$ را روی فرآیند جستجو کنترل می‌کنند.

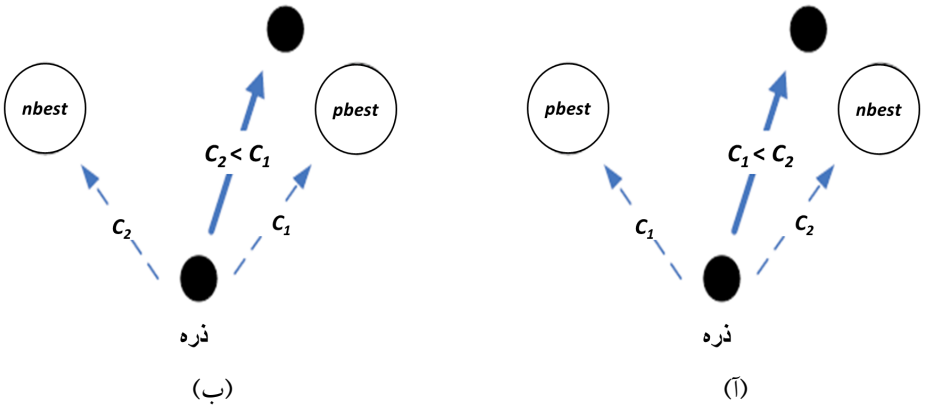
اگر $c_1 < c_2$ ذرات به سمت $nbest$ (بهترین موقعیت در همسایگی ذره) متمایل می‌شوند (شکل ۲-۵(آ)).

اگر $c_1 > c_2$ ذرات به سمت $pbest$ (بهترین موقعیت ذره) متمایل می‌شوند (شکل ۲-۵(ب)).

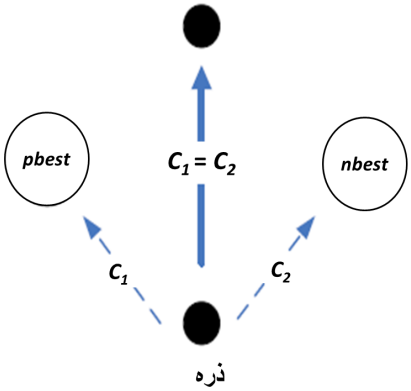
ابرهارت و کندی (۱۹۹۵) [۹] پیشنهاد کردند برای داشتن حالت میانگین باید $c_1 = c_2$ باشد و در این صورت تمایل رسیدن به $pbesti$ و $nbesti$ به یک میزان خواهد بود (شکل ۲-۶).

اگر c_1 و c_2 مقادیر بزرگی باشند، آنگاه سرعت رسیدن به بهترین جواب ممکن زیاد، ولی دقت محاسبه نقطه بهینه کم خواهد بود و احتمال بوجود آمدن نوسان در محاسبه نقاط بهینه زیاد می‌گردد. از طرفی اگر مقادیر c_1 و c_2 کوچک باشد دقت نقطه بدست آمده بالا است ولی سرعت رسیدن به نقطه بهینه کم می‌شود [۱۰].

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



شکل ۲-۵: حرکت ذره در حالتی که c_1 و c_2 برابر نیستند



شکل ۲-۶: حرکت ذره در حالتی که $c_1 = c_2$

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

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

الگوریتم ۲-۲ شبه کد الگوریتم بهینه‌سازی ازدحام ذرات.

ورودی: تعداد جمعیت اولیه، شرط خاتمه.
خروجی: جواب بهینه.

۱: مقداردهی اولیه به موقعیت و سرعت هر ذره.
۲: مقدار برازندگی هر ذره را محاسبه کنید.
۳: برای هر ذره $pbest$ و $nbest$ را محاسبه کنید.
۴: مراحل زیر را تا زمانیکه شرط خاتمه برقرار شود انجام دهید:
۵: سرعت هر ذره را با استفاده از معادله (۲-۲) به‌هنگام کنید.
۶: موقعیت هر ذره را با استفاده از معادله (۲-۳) به‌هنگام کنید.
۷: مقدار برازندگی هر ذره را محاسبه کنید.
۸: اگر مقدار برازندگی جاری ذره از مقدار برازندگی $pbest$ آن بهتر است، $pbest$ را به‌هنگام کنید.
۹: ذره ای با بهترین برازندگی بین همسایگی‌هایش انتخاب کنید و $nbest$ را به‌هنگام نمایید.

فصل ۳

الگوریتم بهینه‌سازی فاخته

۳-۱ مقدمه

در این فصل الگوریتم بهینه‌سازی فاخته [۱] معرفی می‌شود. این الگوریتم بهینه‌سازی از زندگی یک خانواده از پرندگان به نام فاخته‌ها، الهام گرفته شده است. زندگی خاص این نوع پرنده و ویژگی‌های آن در تخم‌گذاری و زاد و ولد، انگیزه اصلی پرورش این الگوریتم تکاملی بهینه‌سازی جدید بوده است. مانند سایر روش‌های تکاملی، الگوریتم بهینه‌سازی فاخته(COA) با یک جمعیت اولیه آغاز می‌شود. جمعیت فاخته‌ها، در جوامع مختلف آن‌ها، دو نوع است: فاخته‌های بالغ و تخم‌ها. تلاش فاخته‌ها برای زنده ماندن اساس الگوریتم بهینه‌سازی فاخته را تشکیل می‌دهد. در طول مسابقه برای زنده ماندن، برخی از فاخته‌ها یا تخم هایشان از بین می‌روند. جامعه فاخته‌های نجات یافته به یک محیط بهتر مهاجرت می‌کنند، و دوباره شروع به تکثیر و تخم‌گذاری می‌کنند. امید است تلاش فاخته‌ها برای زنده ماندن، به حالتی برسد که تنها یک جامعه از فاخته‌ها وجود داشته باشد که همه یک ارزش سود مشابه دارند.

۳-۲ فاخته‌ها و روش منحصر به فرد آن‌ها در تکثیر

همه ۹۰۰۰ گونه پرندگان شیوه یکسانی برای والد شدن دارند: روش تخم‌گذاری، هیچ پرنده‌ای نوزاد به دنیا نمی‌آورد. پرندگان به سرعت یک تخم درون یک پوسته محافظ تشکیل می‌دهند و روی آن می‌خوابند. اندازه بزرگ تخم‌ها، این را برای پرنده ماده دشوار می‌کند که در یک زمان، از بیش از یک تخم نگهداری کند. به همراه داشتن تخم‌ها، پرواز را سخت‌تر می‌کند و به انرژی بیشتری نیاز دارد و چون یک تخم، غنیمتی پر از پروتئین و مواد مغذی برای همه شکارچیان است، پرندگان باید جایی امن برای تخم‌گذاری بیابند. یافتن مکانی برای تخم‌گذاری به صورت ایمن و پرورش جوجه‌شان تا رسیدن به استقلال،

[۱] Cuckoo Optimization Algorithm

چالشی است که پرندگان آن را با راه‌های هوشمندانه‌ای حل کرده‌اند. آن‌ها از هنر، طراحی و مهندسی پیچیده استفاده می‌کنند. تنوع ساختاری لانه‌ها هیچ معادلی در دنیای جانوران ندارد. بسیاری از پرندگان، لانه‌های منفرد و ناپیدا، پنهان شده لابه‌لای پوشش گیاهی می‌سازند تا از شناسایی توسط شکارچیان دور بمانند. بعضی از آنان چنان در پنهان کردن لانه‌هایشان موفق‌اند که حتی چشم‌های جست‌وجوگر انسان هم به ندرت آن‌ها را دیده است.

پرندگانی هم هستند که تمام عرف‌های لانه‌سازی و والدی را رها می‌کنند و برای پروراندن خانواده‌شان به حیله‌گری متوسل می‌شوند. این‌ها «انگل جوجه» [1] هستند، پرندگانی که هرگز لانه خودشان را نمی‌سازند و درعوض در لانه گونه‌های دیگر تخم‌گذاری می‌کنند و نگه‌داری از جوجه‌شان را به والدین آن گونه می‌سپارند. فاخته، شناخته‌شده‌ترین انگل جوجه است و ماهر در فریب ظالمانه. رویکرد او شامل پنهان‌کاری، غافل‌گیری و سرعت است. فاخته مادر تخم پرنده گونه میزبان را برمی‌دارد، تخم خودش را به جای آن در لانه قرار می‌دهد و با تخم میزبان روی منقارش لانه را ترك می‌کند. کل این فرآیند کم‌تر از ۱۰ ثانیه طول می‌کشد. فاخته‌ها انگل لانه‌های انواع زیادی از گونه‌های پرندگان می‌شوند و با دقت رنگ و شکل تخم‌های خود را با تخم‌های گونه‌های میزبان تطبیق می‌دهند. هر فاخته ماده روی یک گونه میزبان خاص، تخصص دارد. این که چگونه فاخته قادر است طوری تخم‌گذاری کند که دقیقاً از تخم‌های میزبان تقلید کند، یکی از معماهای اصلی طبیعت است.

بسیاری از پرندگان یاد می‌گیرند که یک تخم فاخته قرار گرفته در لانه خودشان را شناسایی کنند و یا این تخم عجیب را به بیرون پرتاب کنند یا آن لانه را رها کنند و از نو لانه بسازند. بنابراین فاخته دائماً تلاش دارد در تقلید از تخم‌های میزبان پیشرفت کند، در حالی که پرندگان میزبان سعی در یافتن راه‌هایی برای شناسی تخم‌های انگل دارند. تقلای میان میزبان و انگل مشابه یک مسابقه تسلیحاتی است که در آن هر طرف سعی در غلبه بر دیگری در زنده ماندن دارد.

برای فاخته‌ها، محل زندگی مناسب جایی است که منبعی از غذا - به طور کلی حشرات و مخصوصاً کرم ابریشم - و جایی برای تخم‌گذاری را فراهم کند؛ برای جوجه‌های انگلی مکان مورد نیاز، همان محل زندگی مناسب گونه‌های میزبان است. فاخته‌ها در مکان‌های بسیار گوناگونی زندگی می‌کنند. اکثر این پرنده‌ها در جنگل‌ها و درختستان‌ها وجود دارند، بیشتر در جنگل‌های همیشه سبز استوا. علاوه بر جنگل‌ها بعضی از گونه‌های فاخته در فضاهای بازتر زندگی می‌کنند؛ این فضاها حتی می‌تواند شامل مناطق خشک، مثل کویر باشد. گونه‌های مهاجر به مناطق معتدل، همچون فاخته معمولی، در طیف وسیعی از زیست بوم‌ها سکونت دارند تا بتوانند حداکثر استفاده از میزبان‌های بالقوه جوجه را داشته باشند؛ از نیزارها گرفته تا صحراهای خالی از درخت.

برخی گونه‌های فاخته غیر مهاجر هستند، ولی انواع بسیاری از این پرندگان مهاجرت‌های فصلی منظم دارند، و بسیاری دیگر مهاجرت‌های جزئی در قسمتی از محدوده خودشان دارند. این مهاجرت‌ها ممکن است روزانه باشد، مثل فاخته منقار بلند، یا شبانه باشد، مثل فاخته نوك زرد. برای گونه‌هایی که در عرض جغرافیایی بالاتر تخم‌گذاری می‌کنند، دسترسی به غذا، آن‌ها را به مهاجرت به نواحی گرم‌تر در طول زمستان مجبور می‌کند. فاخته دم‌دراز که در نیوزیلند زاد و ولد می‌کند، به قشلاق‌هایش در پلی‌نزی، میکرونزی و ملانزی می‌رود؛ شاهکاری که شاید بتوان آن را چشم‌گیرترین مهاجرت روی آب میان پرندگان خشکی‌زی توصیف کرد. فاخته نوك زرد و فاخته نوك سیاه در آمریکای شمالی زاد و ولد می‌کنند و از این سو

[1] Brood Parasite

شکل ۳-۱: مراحل رشد تخم فاخته در لانه میزبان

به آن سوی دریای کارائیب پرواز می‌کنند، یک پرواز بدون توقف ۴۰۰۰ کیلومتری. مهاجرت‌های طولانی دیگر، شامل فاخته صغیر است که از هند تا کنیا، روی اقیانوس هند (به اندازه ۳۰۰۰ کیلومتر) پرواز می‌کند؛ و فاخته‌های معمولی اروپایی که در سفرشان به آفریقای جنوبی، روی دریای مدیترانه و صحرای بزرگ آفریقا بدون توقف پرواز می‌کنند. درون آفریقا ۱۰ گونه از فاخته‌ها مهاجرت‌های درون قاره‌ای منظمی دارند که قطبی نامیده می‌شود؛ به این صورت که آن‌ها فصل بدون زاد و ولد را در مرکز استوایی این قاره می‌گذارنند و به منظور زاد و ولد در زمین‌های خشک‌تر و بازتر و صحراها، به شمال و جنوب می‌روند.

جوجه فاخته زودتر از جوجه میزبان سر از تخم بیرون می‌آورد و زودتر رشد می‌کند. در بیشتر موارد، جوجه فاخته حتی تخم‌ها و جوجه‌های گونه میزبان را از لانه بیرون می‌اندازد. جوجه فرصتی برای آموختن این کار نداشته است، پس این باید یک غریزه منتقل شده با ژنتیک باشد. جوجه فاخته، میزبان را با صدای تند درخواستش و دهان بازش که یک علامت محرك است، به همراه شدن با سرعت بالای رشدش تشویق می‌کند. انگل‌های فاخته ماده در قرار دادن تخم‌هایی که شباهت زیادی به تخم‌های میزبان منتخب‌شان دارند، تخصص دارند. این تخصص، یک گزینش طبیعی است؛ همان‌طور که بعضی پرندگان می‌توانند تخم‌های فاخته را از تخم‌های خودشان تمییز دهند؛ تخم‌هایی که کم‌ترین شباهت را به تخم‌هایی که از لانه بیرون انداخته شده‌اند، دارند. گونه‌های میزبان ممکن است در دفعه اول فعالیت‌های مستقیم‌تری برای جلوگیری از تخم‌گذاری فاخته‌ها در لانه‌هایشان انجام دهند. در نظر فاخته‌ها، پرندگانی که لانه‌شان بیشتر در معرض خطر اشغال توسط فاخته‌هاست، آن‌ها را از منطقه بیرون خواهند کرد. فاخته‌های انگل به گروه‌هایی تقسیم می‌شوند که هر گروه در زمینه یک گونه میزبان خاص تخصص دارد. شواهدی وجود دارد مبنی بر این که، این گروه‌ها از نظر ژنتیکی با یک‌دیگر متفاوت‌اند.

تخصص در زمینه گونه‌های میزبان با نیاز به تقلید از تخم‌های آن‌ها افزایش می‌یابد. [۱۲]

دو الگوریتم فراابتکاری با عنوان فاخته با الهام از زندگی این پرنده، ارائه شده است، الگوریتم جستجوی فاخته و الگوریتم بهینه‌سازی فاخته. این پایان‌نامه مبتنی بر الگوریتم بهینه‌سازی فاخته می‌باشد. قبل از این که الگوریتم مورد نظر بررسی شود، الگوریتم جستجوی فاخته را معرفی می‌کنیم.

۳-۳ الگوریتم جستجوی فاخته

الگوریتم جستجوی فاخته [۱]، یکی از الگوریتم‌های فراابتکاری الهام گرفته از طبیعت است که در سال ۲۰۰۹ توسط شین یانگ [۲] و دب ساوش [۳] [۱۳] با استفاده از پرواز لوی [۴]توسعه یافته است.

فرایند تصادفی : یک متغیر تصادفی را می‌توان عبارتی در نظر گرفت که مقدار آن نتیجه وقایعی در ارتباط با یک فرایند تصادفی است. به بیان ریاضی، یک متغیر تصادفی تابعی است که یک رویداد در عالم واقعی را به اعددی حقیقی نسبت می‌دهد.برای هر متغیر تصادفی یک تابع توزیع احتمال وجود دارد، که این توزیع اغلب با تابع چگالی احتمال نشان داده می‌شود.

قدم زدن تصادفی : قدم زدن تصادفی یک فرایند تصادفی است که عبارت است از طی یک سری گام‌های متوالی و تصادفی. فرض کنید S_N مجموع گام‌های متوالی X_i باشد، پس S_N یک قدم زدن تصادفی است به طوری که :

$$S_N = \sum_{i=1}^{N} X_i = X_1 + ... + X_N$$

و X_i یک گام تصادفی از یک توزیع احتمال می‌باشد.این رابطه را می‌توان به شکل بازگشتی زیر نشان داد:

$$S_N = S_{N-1} + X_N$$

که نشان می‌دهد گام بعدی S_N تنها به وضعیت موجود S_{N-1} جاری بستگی دارد و حاصل حرکت یا گذر X_N از وضعیت جاری خواهد بود. این فرایند در واقع خاصیت زنجیره مارکوف است.

اگر هر گام یا جهش در فضایی چند بعدی انجام گیرد، قدم زدن تصادفی چند بعدی خواهد بود. به علاوه هیچ دلیلی ندارد که طول گام‌ها همیشه ثابت باشد. در واقع، همانند جهت هر گام، اندازه هر گام نیز می‌تواند متغیری از یک توزیع شناخته شده باشد. اگر طول هر گام از توزیع گاوسی (نرمال) پیروی کند، قدم زدن تصادفی همان حرکت براونی [۵] خواهد بود.

ممکن است حرکت در هر گام از توزیع دیگری غیر از توزیع نرمال پیروی کند، از این رو باید قدم زدن تصادفی جامع‌تری

---

[۱]Cuckoo Search(CS) [۲]Yang, X.S [۳]Deb, S [۴]Levy Flight [۵]Brownian motion

را مورد بررسی قرار دهیم. یک مورد خیلی خاص زمانی است که طول هر گام از توزیع لوی پیروی کند، که در این صورت قدم زدن تصادفی را پرواز لوی می‌نامند.

توزیع لوی : توزیع لوی که توسط ریاضیدان فرانسوی، پاول پیر لوی [1] ارائه شد، یک توزیع پیوسته برای متغیرهای تصادفی نامنفی است که دارای واریانس نامحدود و میانگین بی‌نهایت می‌باشد و یکی از معدود توزیع‌هایی است که پایدار است. یک متغیر تصادفی U و توزیع احتمال آن را زمانی پایدار می‌گویند که هر ترکیب خطی از دو متغیر دلخواه آن مانند U_1 و U_2 از همان توزیع پیروی کنند. دو تابع دیگری که چنین خاصیتی دارند توزیع نرمال و توزیع کوشی هستند.

تابع چگالی احتمال توزیع لوی بر روی دامنه $X \geq 0$ به صورت زیر است:

$$f(x;c) = \sqrt{\frac{c}{2\pi}} \quad \frac{e^{\frac{-c}{2x}}}{x^{3/2}}$$

که c پارامتر مقیاس است.

پرواز لوی : پرواز لوی یك سیر تصادفی با گام های تصادفی است که طول این گام‌ها از توزیع لوی پیروی می‌کند. نتایج بسیاری از پژوهش‌هایی که پرواز پرندگان و حشرات را مورد مطالعه قرار داده‌اند، نشان داده است که الگوی پرواز بسیاری از پرندگان و حشرات از پرواز لوی یا سیر لوی تبعیت می‌کند.

$$levy \sim u = t^{-\lambda} \qquad (1 < \lambda \leq 3)$$

شکل ۳-۲ پرواز لوی را نشان می‌دهد.

شکل ۳-۲: نمایش پرواز لوی

گام‌های اساسی الگوریتم جستجوی فاخته، به شرح ذیل است:

[1] Paul Peierre Levy

- هر فاخته در هر تکرار فقط یک تخم می‌گذارد و تخم خود را در لانه‌ای که به صورت تصادفی انتخاب شده است، قرار می‌دهد.
- بهترین لانه با تخم‌های با کیفیت بالا به نسل بعد منتقل خواهد شد.
- تعداد لانه‌های میزبان در دسترس ثابت است و تخم‌های گذاشته شده توسط فاخته در این لانه‌ها با احتمالاتی توسط میزبان کشف می‌شود. در این مورد، پرنده میزبان یا می‌تواند از شر تخم خلاص شود و یا به سادگی لانه را رها کرده و یک لانه جدید بسازد.

هر تخم در یک لانه، نشان دهنده یک راه حل است و هر فاخته می‌تواند تنها یک تخم بگذارد، هدف استفاده از راه‌حل‌های جدید و به طور بالقوه بهتر به جای راه حل نه چندان خوب است.

تولید جواب جدید $x^{(t+1)}$ برای فاخته i بدین صورت است که، ابتدا گام تصادفی را از توزیع لوی تولید کرده، سپس در فضای جواب‌های شدنی، جوابی را با استفاده از آن گام تصادفی تولید می‌نماید:

$$x_i{}^{(t+1)} = x_i{}^t + \alpha \oplus levy(\lambda)$$

که $\alpha > 0$ و طول گام است که متناسب با مقیاس مسأله و اندازه فضای جواب انتخاب می‌شود. رابطه فوق، در اصل یک رابطه تصادفی برای حرکت یا پرواز تصادفی است. در واقع یک زنجیره مارکوف است که در آن، مکان بعدی تنها به مکان فعلی (عبارت اول در معادله) و احتمال انتقال (عبارت دوم) بستگی دارد و $\oplus$ به معنی ضرب مؤلفه به مؤلفه می‌باشد.

این احتمال انتقال با توزیع لوی به صورت زیر تنظیم می‌شود:

$$levy(\lambda) \sim t^{-\lambda} \qquad (1 < \lambda \leq 3)$$

که دارای واریانس و میانگین بی‌نهایت است.

به منظور پیاده‌سازی و استفاده از این توزیع، تولید اعداد تصادفی با پرواز لوی از دو مرحله تشکیل می‌شود: انتخاب جهت تصادفی و تولید اندازه هر گام که از توزیع لوی پیروی می‌کند. تولید جهت می‌تواند با توزیع یکنواخت باشد.برای تولید طول گام از توزیع لوی چند راه وجود دارد، یکی از راه‌های کارآمد و آسان استفاده از الگوریتم مانتگنا [1] برای توزیع پایدار و متقارن لوی است. البته از متقارن این است که طول گام‌ها می‌تواند مثبت یا منفی باشد. در روش مانتگنا طول گام s را با فرمول زیر می‌تان محاسبه کرد:

$$s = \frac{u}{|v|^{1\backslash\beta}}$$

که u و v دارای توزیع نرمال هستند. با این روش s دارای توزیع مورد نظر لوی خواهد بود.

[1] Mantegna

برخی از راه حل‌های جدید با پرواز لوی در اطراف بهترین جوابی که در هر مرحله به‌دست آمده تولید می‌شوند، این باعث سرعت بخشیدن به جستجوی محلی می‌شود. هم‌چنین بخشی از جواب‌های جدید را باید در میدانی دورتر از جواب‌های جاری به تصادف انتخاب کرد تا فرآیند جستجو در بهینه محلی گیر نیفتد [۱۴].

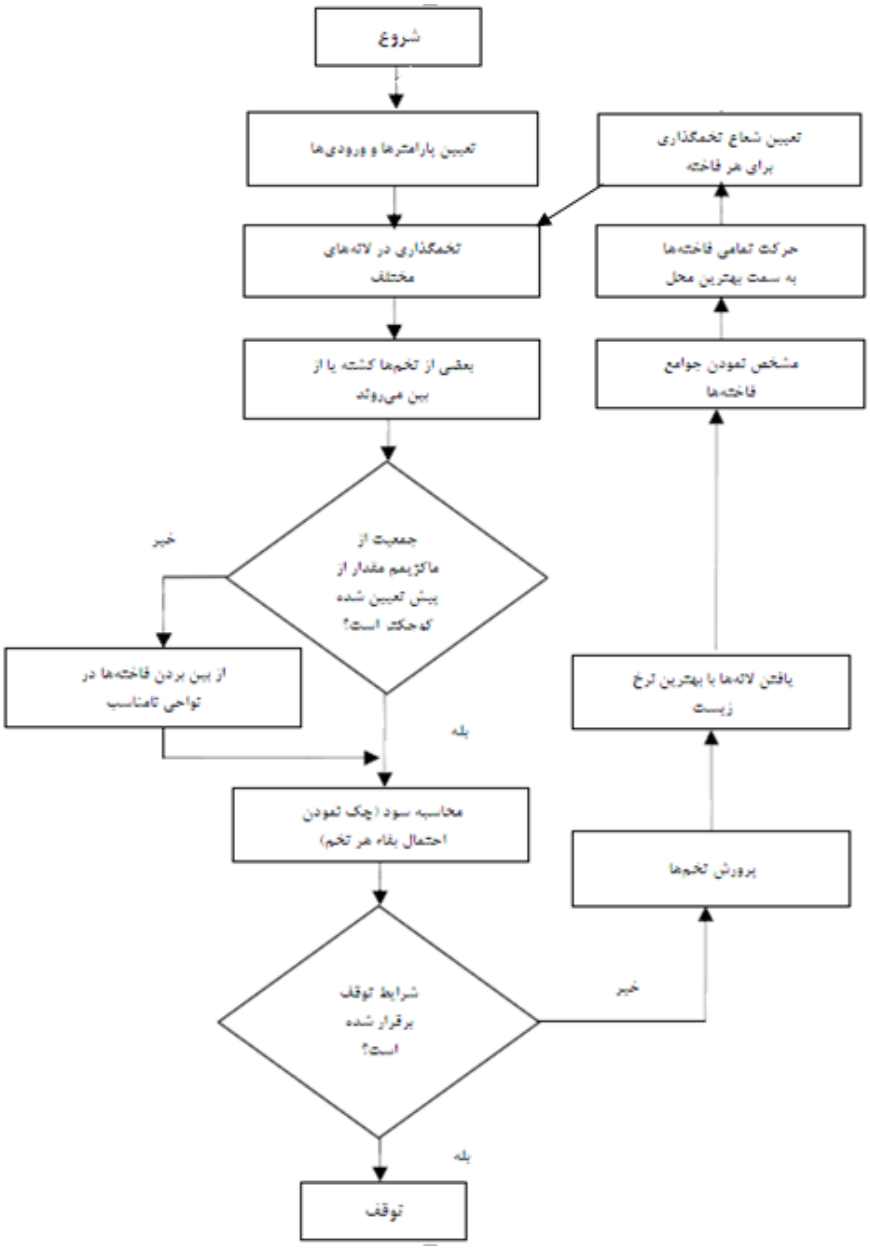
۳-۴ الگوریتم بهینه‌سازی فاخته

الگوریتم بهینه‌سازی فاخته با الهام از زندگی پرندگانی به نام فاخته، در سال ۲۰۱۱ توسط رامین رجبیون [۱۲] ارائه گردید. این الگوریتم با توجه به رده‌بندی ذکر شده در فصل ۲ برای الگوریتم‌های فراابتکاری، یک الگوریتم مبتنی بر جمعیت، الهام گرفته شده از طبیعت، با حافظه و احتمالی است، که در ادامه جزئیات آن بیان می‌شود.

همانند سایر الگوریتم‌های تکاملی، COA نیز با یک جمعیت اولیه از فاخته‌ها شروع می‌شود. این فاخته‌های اولیه تخم‌هایی برای قرار دادن در لانه‌های پرندگان میزبان دارند. برخی از این تخم‌ها که شباهت بیشتری به تخم‌های پرنده میزبان دارند، فرصت رشد کردن و تبدیل شدن به فاخته بالغ را دارند. تخم‌های دیگر توسط پرنده‌های میزبان شناسایی و از بین برده می‌شوند. تخم‌های رشد یافته میزان مناسب بودن لانه‌های آن منطقه را نشان می‌دهند. در یک منطقه، هر چه تخم‌های بیشتری زنده بمانند، سود بیشتری در آن منطقه به‌دست می‌آید. پس COA در پی یافتن موقعیتی است که تخم‌های بیشتری در آن زنده بمانند. فاخته‌ها به دنبال مناسب‌ترین منطقه برای تخم‌گذاری هستند تا نرخ زنده ماندن تخم‌هایشان را بیشینه سازند. پس از آن‌که تخم‌های باقی مانده رشد کردند و به فاخته بالغ تبدیل شدند، جوامعی را تشکیل می‌دهند. هر جامعه زیست بوم خودش را دارد. بهترین محل زندگی یافت شده فاخته‌ها، مقصد فاخته‌ها در جوامع دیگر خواهد بود. سپس آن‌ها به این بهترین محل زندگی مهاجرت می‌کنند. آن‌ها در جایی نزدیک بهترین زیست بوم ساکن می‌شوند. با توجه به تعداد تخم‌های هر فاخته، و همچنین فاصله فاخته از نقطه مقصد (بهترین زیست بوم) یک شعاع تخم‌گذاری به آن فاخته اختصاص داده می‌شود. سپس، فاخته شروع به تخم‌گذاری در لانه‌های تصادفی درون شعاع تخم‌گذاری‌اش می‌کند. این فرآیند ادامه می‌یابد تا بهترین موقعیت با بالاترین ارزش سود به‌دست آید و اکثر جمعیت فاخته‌ها در این مکان جمع شوند. فلوچارت الگوریتم بهینه‌سازی فاخته در شکل ۳-۳ آورده شده است.

۳-۴-۱ پارامترهای الگوریتم

پارامترهای لازم برای الگوریتم بهینه‌سازی فاخته را در جدول ۳-۱ درج می‌نماییم.



شکل ۳-۳: فلوچارت الگوریتم بهینه‌سازی فاخته

۳-۴-۲ عملکرد الگوریتم بهینه‌سازی فاخته

عملکرد الگوریتم بهینه‌سازی فاخته برای حل مسائل مختلف به صورت زیر است:

۳-۴-۲-۱ ایجاد محل زندگی اولیه

برای حل یک مسأله بهینه‌سازی، لازم است مقادیر متغیرهای مسأله به شکل یک ماتریس درآیند. در اصطلاحات مربوط به الگوریتم ژنتیک و الگوریتم بهینه‌سازی ازدحام ذرات، این ماتریس به ترتیب کروموزوم و موقعیت ذره نامیده می‌شود. اما

جدول ۳-۱: معرفی پارامترها

نماد	توضیح
N_{var}	تعداد متغیرهای مسأله
var_{low}	حد پایین متغیرها
var_{hi}	حد بالای متغیرها
N_{pop}	تعداد فاخته‌های ابتدایی جهت حل مسأله
N_{minE}	حداقل تعداد تخم‌های فاخته در هر تکرار
N_{maxE}	حداکثر تعداد تخم‌های فاخته در هر تکرار
$MaxIter$	ماکزیمم تعداد تکرارها
N_k	تعداد خوشه‌های k-means
F	ضریب حرکتی به سمت جواب بهینه
$Accuracy$	مقدار بهینه تابع هدف در صورت وجود
N_{maxCu}	حداکثر تعداد فاخته‌ها در هر تکرار
α	ضریب شعاع تخم‌گذاری

در الگوریتم بهینه‌سازی فاخته، این ماتریس، محل زندگی [۱] نامیده می‌شود. در یک مسأله بهینه‌سازی با N_{var} متغیر، محل زندگی، یک ماتریس $۱ \times N_{var}$ است که موقعیت فعلی فاخته را نشان می‌دهد. این ماتریس به شکل زیر تعریف می‌شود:

$$habitat = [x_۱, x_۲, ..., x_{N_{var}}]$$

سود محل زندگی توسط تابع برازندگی f_p به صورت زیر محاسبه می‌شود:

$$\text{سود} = f_p(habitat) = f_p(x_۱, x_۲, , x_{N_{var}})$$

برای شروع الگوریتم بهینه‌سازی، ماتریس محل زندگی $N_{pop} \times N_{var}$ برای جمعیت اولیه ایجاد می‌شود. تعداد تخم‌های هر فاخته در هر تکرار با استفاده از رابطه زیر مشخص می‌گردد:

$$round((N_{maxE} - N_{minE}) \times rand + N_{minE})$$

خصلت دیگر فاخته‌ها این است که در دامنه‌ای مشخص نسبت به محل زندگی‌شان تخم‌گذاری می‌کنند. در الگوریتم این فاصله بیشینه شعاع تخم‌گذاری [۲] (ELR) نامیده می‌شود. هر فاخته در هر تکرار، یک بیشینه شعاع تخم‌گذاری دارد که با تعداد کل تخم‌ها و تعداد تخم‌های آن فاخته در تکرار فعلی و هم‌چنین حدود بالا و پایین متغیرها متناسب است. بنابراین

[۱] habitat [۲] Egg Laying Radius

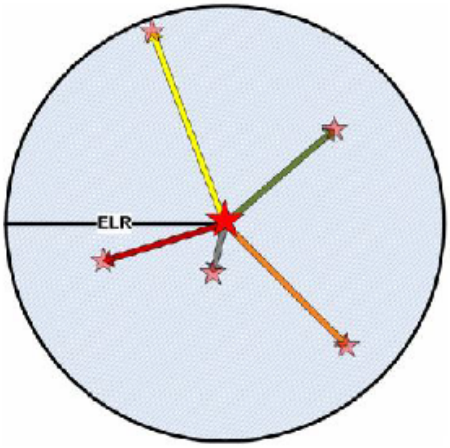
ELR به صورت زیر محاسبه می‌شود:

$$ELR = \alpha \times \frac{N_{cuckoo'sE}}{N_{totalE}} \times (var_{hi} - var_{low})$$

که α عدد صحیحی است که برای به دست آوردن مقدار ماکزیمم ELR به کار می‌رود و ضریب شعاع تخم‌گذاری نامیده می‌شود و $N_{cuckoo'sE}$ و N_{totalE} به ترتیب تعداد تخم‌های فاخته در این تکرار و تعداد کل تخم‌ها در تکرار فعلی می‌باشند.

۳-۴-۲-۲ روش تخم‌گذاری فاخته‌ها

هر فاخته در ELR خود، شروع به تخم‌گذاری به صورت تصادفی در بعضی از لانه‌های پرنده‌های میزبان می‌کند. شکل ۳-۴



شکل ۳-۴: شعاع تخم‌گذاری

نمای واضحی از این مفهوم را نمایش می‌دهد. ستاره بزرگ نشان‌دهنده فاخته مادر وستاره‌های کوچک نشان‌دهنده تخم‌ها می‌باشند.

در الگوریتم برای جستجوی محلی بهتر، محل قرار گرفتن تخم‌ها در اطراف فاخته مادر به روش زیر تعیین می‌شود. ابتدا به تعداد تخم‌هایی که برای فاخته در نظر گرفته شده است، زیر شعاع‌های R_i کمتر یا مساوی بیشینه شعاع تخم‌گذاری‌اش به صورت زیر محاسبه می‌شود:

$$R_i = ELR \times rand \qquad i = ۱, ..., N_{cuckoo'sE}$$

$rand$ عددی تصادفی بین ۰ و ۱ می‌باشد. سپس زاویه‌های θ_i محاسبه می‌شود:

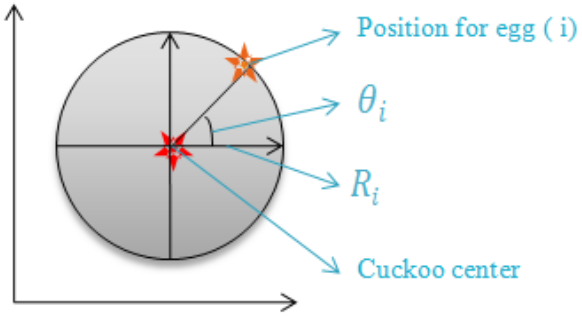
$$\theta_i = i \times \frac{۲\pi}{N_{cuckoo'sE}} \qquad i = ۱, ..., N_{cuckoo'sE}$$

اگر مختصات فاخته مادر را با X و مختصات تخم را با x نشان دهیم، مختص jام تخم iام به صورت زیر به‌دست می‌آید:

$$x_{ij} = X_j + (-۱)^r [R_i \times cos(\theta_i) + R_i \times sin(\theta_i)]$$

$$i = ۱, ..., N_{cuckoo's E} \qquad j = ۱, ..., N_{var}$$

r انتخاب تصادفی عدد ۱ یا ۲ می‌باشد.



شکل ۳-۵: تعیین موقعیت تخم‌ها با توجه به موقعیت فاخته مادر

پس از آن که همه تخم‌های فاخته‌ها در لانه‌های پرنده‌های میزبان قرار گرفتند، بعضی از آن‌ها که کمتر شبیه تخم‌های پرنده میزبان هستند، توسط میزبان شناسایی و بیرون انداخته می‌شوند. بنابراین بعد از فرآیند تخم‌گذاری، تعدادی از تخم‌ها با مقادیر سود کم‌تر نابود می‌شوند. این تخم‌ها شانسی برای رشد کردن ندارند. مابقی تخم‌ها در لانه‌های میزبان رشد می‌یابند، جوجه‌ها سر از تخم بیرون می‌آورند و پرندگان میزبان آن‌ها را تغذیه می‌کنند. نکته جالب توجه دیگر درباره تخم‌های فاخته، این است که تنها یک تخم در هر لانه فرصت رشد می‌یابد. به این خاطر که وقتی تخم فاخته می‌شکند و جوجه فاخته سر از تخم بیرون می‌آورد، تخم‌های پرنده میزبان را از لانه بیرون می‌اندازد. در صورتی که تخم‌های پرنده میزبان زودتر از تخم فاخته بشکنند، جوجه فاخته بیشتر غذایی که پرنده میزبان به لانه می‌آورد را می‌خورد (چون جثه‌اش سه برابر بزرگ‌تر است، باقی جوجه‌ها را کنار می‌زند و بیشتر می‌خورد). بعد از چند روزی جوجه‌های پرنده میزبان می‌میرند و فقط جوجه فاخته در لانه می‌ماند.

۳-۲-۴-۳ حذف فاخته‌ها در بدترین محل‌های زندگی

با توجه به این واقعیت که همیشه در جمعیت پرنده‌ها، تعادل برقرار است، بنابراین عدد N_{maxCu} تعداد بیشینه فاخته‌های زنده در محیط را کنترل و محدود می‌کند. این تعادل به علت محدودیت‌های غذایی، کشته شدن توسط شکارچیان و همچنین عدم توانایی در یافتن لانه مناسب برای تخم‌ها به وجود می‌آید. در الگوریتم، بعد از فرآیند تخم‌گذاری سود هر محل زندگی محاسبه می‌شود و تنها تعداد N_{maxCu} از فاخته‌ها که مقادیر سود بالاتری دارند، زنده می‌مانند، بقیه از بین می‌روند.

۳-۴-۲-۴ مهاجرت فاخته‌ها

هنگامی که جوجه‌های فاخته رشد می‌کنند و بالغ می‌شوند، برای مدتی در منطقه و جامعه خودشان می‌مانند. اما وقتی به زمان تخم‌گذاری نزدیک می‌شوند، به محل زندگی جدید و بهتر مهاجرت می‌کنند.

در این محل‌ها، تخم‌های فاخته شباهت ببیشتری به تخم پرندگان میزبان دارند و غذای بیشتری نیز برای جوجه‌های جدید وجود دارد. پس از شکل گرفتن گروه‌های فاخته در مناطق مختلف، جامعه با بهترین مقدار سود به عنوان نقطه هدف برای سایر فاخته‌ها برگزیده می‌شود تا به آن مهاجرت کنند. وقتی فاخته‌های بالغ در محیطی زندگی می‌کنند، مشکل می‌توان تشخیص داد که کدام فاخته به کدام گروه تعلق دارد. برای حل این مشکل، در COA گروه‌بندی فاخته‌ها به روش خوشه‌بندی k-means انجام می‌شود.

خوشه‌بندی به روش k-means

الگوریتم زیر، الگوریتم پایه برای این روش محسوب می‌شود:

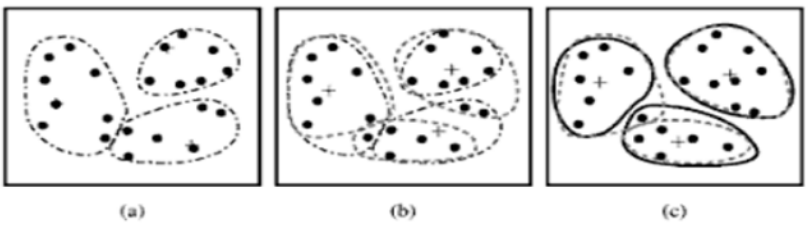
۱. در ابتدا k نقطه به عنوان نقاط مراکز خوشه‌ها انتخاب می‌شوند.

۲. هر نمونه داده به خوشه‌ای که مرکز آن خوشه، کمترین فاصله تا آن داده را داراست، نسبت داده می‌شود.

۳. پس از تعلق تمام داده‌ها به یکی از خوشه‌ها، برای هر خوشه یک نقطه جدید به عنوان مرکز محاسبه می‌شود (میانگین نقاط متعلق به هر خوشه).

۴. مراحل ۲و ۳ تکرار می‌شوند تا زمانی که دیگر هیچ تغییری در مراکز خوشه‌ها حاصل نشود.

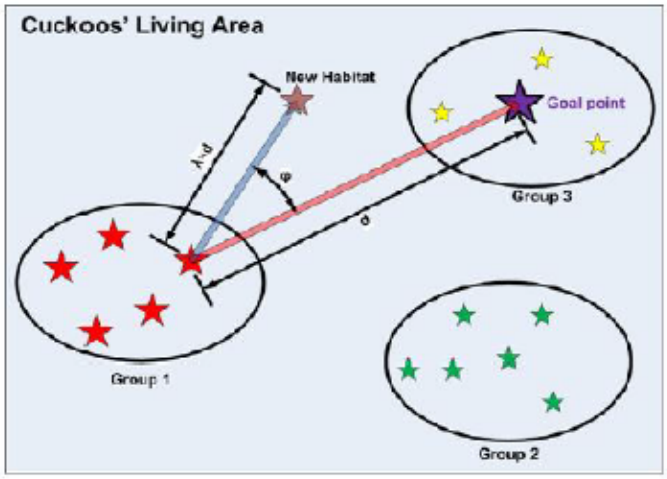
در شکل ۳-۶مراحل انجام خوشه‌بندی نشان داده شده است.



شکل ۳-۶: مراحل خوشه‌بندی به روش k-means

اکنون که گروه‌های فاخته تشکیل شده‌اند، سود میانگین هر گروه محاسبه می‌شود. مقدار بیشینه این میانگین‌ها گروه هدف را تعیین می‌کند. سرانجام، بهترین محل زندگی در این گروه، مقصد جدید فاخته‌ها برای مهاجرت است.

در هنگام حرکت به سمت نقطه هدف، فاخته‌ها تمام طول راه به سمت مقصد را پرواز نمی‌کنند. آن‌ها فقط بخشی از راه را پرواز می‌کنند و همچنین یک انحراف از مسیر دارند. این شیوه حرکت در حالت دوبعدی به وضوح در شکل ۳-۷ نشان داده شده است. همان طور که در شکل دیده می شود، هر فاخته فقط λ برابر کل فاصله تا مقصدش را پرواز می‌کند و



شکل ۳-۷: مهاجرت یک فاخته به محل زندگی هدف

همچنین انحرافی به اندازه ϕ رادیان دارد. این دو پارامتر λ و ϕ به فاخته‌ها کمک می‌کند تا موقعیت‌های بیشتری را در محیط جستجو کنند. در یک مسأله دو متغیره، برای هر فاخته λ یک عدد تصادفی با توزیع یکنواخت بین ۰ و ۱ است و ϕ پارامتری است که میزان انحراف از مقصد را محدود می‌کند و عددی بین $\frac{\pi}{۶}rad$ و $\frac{-\pi}{۶}rad$ برای همگرایی مناسب جمعیت فاخته‌ها، برای سود کلی بیشینه، کافی به نظر می‌رسد. البته در فضاهای بیشتر از سه بعد، تعیین زاویه انحراف مفهومی ندارد. لذا در الگوریتم بهینه‌سازی فاخته در قسمت عملگر مهاجرت، مکان جدید فاخته بعد از مهاجرت به روش زیر تعیین می‌شود:

$$(X_{Next})_j = (X_{Current})_j + F \times rand \times ((X_{Goal})_j - (X_{Current})_j)$$

$$j = ۱, ..., N_{var}$$

F ضریب حرکتی به سمت نقطه هدف نامیده می‌شود و $X_{Current}$ و X_{Next} و X_{Goal} به ترتیب موقعیت فعلی فاخته، موقعیت فاخته بعد از مهاجرت و موقعیت نقطه هدف مهاجرت می‌باشند.

۳-۴-۲-۵ همگرایی الگوریتم

وقتی همه فاخته‌ها به سمت نقطه هدف مهاجرت کردند و محل زندگی جدید آن‌ها مشخص گردید، به هر فاخته بالغ چند تخم داده می‌شود. سپس ELR هر فاخته محاسبه و فرآیند تخم‌گذاری از نو آغاز می‌شود.

بعد از چند تکرار، همه جمعیت فاخته‌ها به بهترین محل زندگی با بیشترین شباهت به تخم‌های پرندگان میزبان و نیز بیشترین منابع غذایی مهاجرت می‌کنند. این محل زندگی بیشترین سود ممکن را ایجاب می‌کند. کمترین تلفات تخم‌ها در این محل زندگی رخ می‌دهد.

الگوریتم ۳-۱ شبه کد الگوریتم بهینه‌سازی فاخته.

ورودی: پارامترهای الگوریتم.
خروجی: جواب بهینه.
۱: مکان‌های اولیه سکونت فاخته‌ها را به صورت تصادفی مشخص نمایید.
۲: تعدادی تخم به هر فاخته اختصاص دهید.
۳: شعاع تخم‌گذاری هر فاخته را تعیین نمایید.
۴: فاخته‌ها در لانه‌های میزبانانی که در شعاع تخم‌گذاری‌شان قرار دارند، تخم‌گذاری می‌کنند.
۵: محل سکونت فاخته‌ها را ارزیابی نمایید.
۶: با توجه به حداکثر تعداد فاخته‌هایی که در هر تکرار می‌توانند زنده بمانند، آن‌هایی را که در مکان‌های نامناسب هستند از بین ببرید.
۷: فاخته‌ها را با استفاده از روش k-means خوشه‌بندی و بهترین گروه فاخته را به عنوان مکان سکونت هدف مشخص نمایید.
۸: جمعیت جدید فاخته‌ها به سمت مکان هدف مهاجرت می‌کنند.
۹: اگر شرط توقف برقرار گردیده توقف کنید، در غیر این صورت به گام ۲ بروید.

مثال ۳-۴-۱. مسأله بهینه‌سازی غیرخطی زیر را در نظر بگیرید:

$$Max \;\; z = \sum_{i=۱}^{۲} x_i^{۲} \qquad ۰ \leq x_i \leq ۴$$

با توجه به نوع تابع هدف و ضریب مثبت هر دو متغیر، بهترین جواب به ازای حد بالای متغیرها به‌دست می‌آید:

$$Max(z) = ۴^{۲} + ۴^{۲} = ۳۲$$

حال جواب بهینه را با استفاده از الگوریتم بهینه‌سازی فاخته به‌دست می‌آوریم.

راه حل: ابتدا پارامترهای مسأله را مقداردهی می‌کنیم:

1. $N_{var} = 2$
2. $Var_{low} = 0$
3. $Var_{hi} = 4$
4. $NumCuckoos = 3$
5. $MinNumberofEggs = 2$
6. $MaxNumberofEggs = 4$
7. $MaxIter = 2$

8. $K - ClusterNum = 2$
9. $MotionCoeff(F) = 0.5$
10. $Accuracy = 32$
11. $MaxNumberofCuckoos = 6$
12. $RadiusCoeff(\alpha) = 5$

هر کدام از پارامترها به شرح زیر است:

۱. مسأله بهینه‌سازی از نوع دو متغیره است.

۲. حد پایین متغیرها ۰ می‌باشد.

۳. حد بالای متغیرها ۴ است.

(چنان‌چه حد پایین و بالا متغیرها مشخص نباشد، سعی کنید بازه را به اندازه کافی بزرگ انتخاب نمایید تا الگوریتم در جواب بهینه محلی متوقف نشود.)

۴. تعداد ۳ فاخته اولیه برای حل مسأله در نظر گرفته شده است.

۵. حداقل تخم‌هایی که هر فاخته می‌گذارد ۲ در نظر گرفته شده است.

۶. حداکثر تخم‌هایی که هر فاخته می‌گذارد ۴ در نظر گرفته شده است.

۷. الگوریتم تا ۲ تکرار انجام می‌شود.

(البته ماکزیمم تکرارهای بهینه‌سازی با توجه به ابعاد مسأله می‌تواند تا چندهزار متفاوت باشد.)

۸. جمعیت فاخته‌ها در خوشه‌بندی به روش k-means به دو گروه خوشه‌بندی می‌شوند.

۹. ضریب حرکتی به سمت جواب بهینه ۰/۵ در نظر گرفته شده است.

(این مقدار هم با توجه به ابعاد مسأله و حدود متغیرها تعیین می‌شود. به عنوان مثال چنان‌چه هر دو متغیر در بازه ۰ تا ۱ باشند، ضریب حرکتی بزرگی مانند ۷ باعث می‌شود تا جستجو روی محیط فضای جواب باشد و زمان رسیدن به جواب را طولانی خواهد کرد.)

۱۰. مقدار بهینه تابع هدف در این مسأله ۳۲ می‌باشد.

(اگر مقدار بهینه تابع هدف مشخص باشد از این پارامتر برای شرط توقف الگوریتم استفاده می‌شود. یعنی زمانی که الگوریتم به این مقدار یا نزدیکی آن رسید، متوقف می‌شود. اما اگر این مقدار مشخص نباشد، برای مسائل ماکزیمم‌سازی از مثبت بی‌نهایت استفاده می‌شود.)

۱۱. حداکثر ۶ فاخته در هر تکرار می‌توانند زنده بمانند.

۱۲. ضریب شعاع تخم‌گذاری، که حداکثر شعاع تخم‌گذاری با آن تنظیم می‌شود، ۵ در نظر گرفته شده است.

(این ضریب در این مثال ثابت در نظر گرفته شده است، اما می‌توان برای افزایش سرعت همگرایی الگوریتم، در هر تکرار با نزدیک شدن به جواب بهینه این مقدار را کوچکتر کرد.)

الگوریتم با جمعیت اولیه ۳ فاخته شروع می‌شود. در این مرحله متغیرها به صورت تصادفی در بازه [۰, ۱] مقداردهی می‌شوند:

$$\begin{bmatrix} Cuckoo۱ \\ Cuckoo۲ \\ Cuckoo۳ \end{bmatrix} : \begin{bmatrix} ۰ & ۱ \\ ۲ & ۱/۵ \\ ۱/۲۵ & ۰/۳ \end{bmatrix} \rightarrow \begin{bmatrix} ۱ \\ ۶/۲۵ \\ ۱/۶۵۲۵ \end{bmatrix} : \begin{bmatrix} f_۱(x) \\ f_۲(x) \\ f_۳(x) \end{bmatrix}$$

تکرار صفر:

Goal Point = (0, 0)

Max Profit = 0

Global Best Cuckoo = Goal Point = (0, 0)

Global Max Profit = Max Profit = 0

Profit Vector = [1, 6.25, 1.6525]

شرط توقف:

If Iteration < Max Itr and Max Profit < accuracy, Then : Iteration + 1

0 < 2 and 0 < 32, Then : Iteration = 1

تکرار اول:

۱-۱ : تعداد تصادفی تخم به هر فاخته با توجه به فرمول زیر، اختصاص می‌یابد:

$$round((N_{maxE} - N_{minE}) \times rand + N_{minE})$$

$$\begin{bmatrix} NumberofEggs, Cuckoo۱ \\ NumberofEggs, Cuckoo۲ \\ NumberofEggs, Cuckoo۳ \end{bmatrix} : \begin{bmatrix} ۳ \\ ۳ \\ ۴ \end{bmatrix}$$

۱-۲ : بیشینه شعاع تخم‌گذاری هر فاخته نیز طبق فرمول زیر محاسبه می‌شود:

$$ELR = \alpha \times \frac{N_{cuckoo'sE}}{N_{totalE}} \times (var_{hi} - var_{low})$$

$$ELR_{Cuckoo۱} = ۵ \times \frac{۳}{۱۰}(۴ - ۰) = ۶$$

$$ELR_{Cuckoo۲} = ۵ \times \frac{۳}{۱۰}(۴ - ۰) = ۶$$

$$ELR_{Cuckoo۳} = ۵ \times \frac{۴}{۱۰}(۴ - ۰) = ۸$$

۳-۱ : حال هر فاخته به طور تصادفی در محدوده شعاع تخم‌گذاری خود و با توجه به حدود متغیرهای تصمیم شروع به تخم‌گذاری می‌نماید.

جدول ۳-۲: مختصات تخم‌ها در تکرار اول

فاخته‌ها	مختصات تخم‌ها	مقدار تابع هدف در محل تخم‌گذاری
فاخته اول	Egg۱ : (۰, ۳)	$f(X) = ۰^۲ + ۳^۲ = ۹$
	Egg۲ : (۲/۹, ۳/۸)	$f(X) = ۲/۹^۲ + ۳/۸^۲ = ۲۲/۸۵$
	Egg۳ : (۳, ۱)	$f(X) = ۳^۲ + ۱^۲ = ۱۰$
فاخته دوم	Egg۱ : (۳, ۰)	$f(X) = ۳^۲ + ۰^۲ = ۹$
	Egg۲ : (۴, ۲)	$f(X) = ۴^۲ + ۲^۲ = ۲۰$
	Egg۳ : (۰, ۰)	$f(X) = ۰^۲ + ۰^۲ = ۰$
فاخته سوم	Egg۱ : (۲, ۰)	$f(X) = ۲^۲ + ۰^۲ = ۴$
	Egg۲ : (۱, ۰/۵)	$f(X) = ۱^۲ + ۰/۵^۲ = ۱/۲۵$
	Egg۳ : (۳/۲, ۲)	$f(X) = ۳/۲^۲ + ۲^۲ = ۱۴/۲۴$
	Egg۴ : (۴, ۱/۵)	$f(X) = ۴^۲ + ۱/۵^۲ = ۱۸/۲۵$

بنابراین بردار جواب بصورت زیر حاصل می گردد:

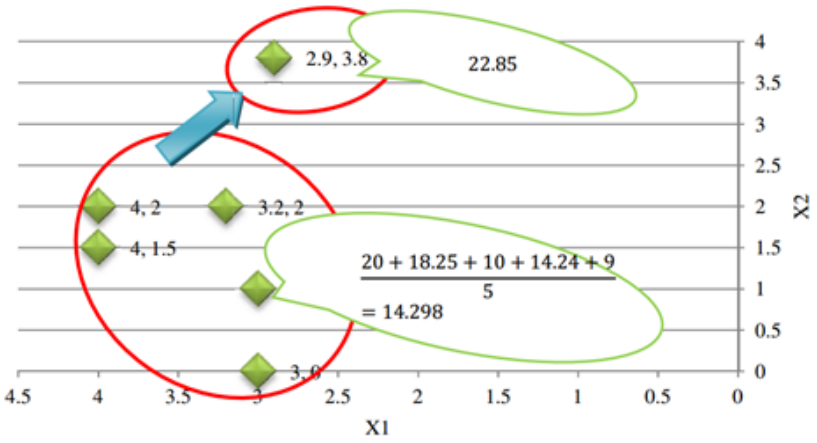
$$ProfitVector = [۱, ۹, ۲۲/۸۵, ۱۰, ۶/۲۵, ۹, ۲۰, ۰, ۱/۶۵۲۵, ۴, ۱/۲۵, ۱۴/۲۴, ۱۸/۲۵]$$

۴-۱ : با توجه به اینکه در هر مرحله حداکثر ۶ فاخته می‌توانند زنده باشند، بنابراین از این ۱۳ جواب ۷ جواب از بدترین ها حذف می‌شوند:

$$ProfitVector = [۲۲/۸۵, ۱۰, ۹, ۲۰, ۱۴/۲۴, ۱۸/۲۵]$$

۵-۱ : فاخته ها را با استفاده از روش k-means خوشه بندی و بهترین گروه فاخته را به عنوان محل سکونت هدف مشخص می‌نماییم. برای این منظور با محاسبه میانگین سود گروه، بهینگی نسبی هر گروه به‌دست می‌آید. سپس در گروهی که دارای بیشترین مقدار متوسط سود (بهینگی) می‌باشد، محلی که بیشترین مقدار سود را دارد به عنوان نقطه هدف انتخاب و گروه های دیگر به سمت آن مهاجرت می‌کنند. همان‌گونه که در شکل ۳-۸ مشاهده می‌شود، از گروه با میانگین ۱۴/۲۹۸ به

سمت فاخته با سود ۲۲/۸۵ حرکت می‌کنیم. با استفاده از فرمول عملگر مهاجرت مختصات نقاط جدید محاسبه می‌گردد.



شکل ۳-۸: خوشه‌بندی جواب‌ها در تکرار اول و تعیین نقطه هدف مهاجرت

با استفاده از فرمول عملگر مهاجرت، موقعیت جدید فاخته‌ها تولید می‌شود:

$$(X_{Next})_j = (X_{Current})_j + F \times rand \times ((X_{Goal})_j - (X_{Current})_j)$$

$$j = ۱, ۲$$

۶-۱ : مهاجرت فاخته ها به محل سکونت بهینه (محاسبه مختصات نقاط جدید پس از مهاجرت):

$$\begin{cases} X۱_{new} = (۲/۹, ۳/۸) + ۰/۵[(۲/۹, ۳/۸) - (۲/۹, ۳/۸)] = (۲/۹, ۳/۸) \\ X۲_{new} = (۳, ۱) + ۰/۵[(۲/۹, ۳/۸) - (۳, ۱)] = (۲/۹۵, ۲/۴) \\ X۳_{new} = (۳, ۰) + ۰/۵[(۲/۹, ۳/۸) - (۳, ۰)] = (۲/۹۵, ۱/۹) \\ X۴_{new} = (۴, ۲) + ۰/۵[(۲/۹, ۳/۸) - (۴, ۲)] = (۳/۴۵, ۲/۹) \\ X۵_{new} = (۳/۲, ۲) + ۰/۵[(۲/۹, ۳/۸) - (۳/۲, ۲)] = (۳/۰۵, ۲/۹) \\ X۶_{new} = (۴, ۱/۵) + ۰/۵[(۲/۹, ۳/۸) - (۴, ۱/۵)] = (۳/۴۵, ۲/۶۵) \end{cases} \rightarrow \begin{cases} f(X۱) = ۲۲/۸۵ \\ f(X۲) = ۱۴/۴۶ \\ f(X۳) = ۱۲/۳۱ \\ f(X۴) = ۲۰/۳۱ \\ f(X۵) = ۱۷/۷۱ \\ f(X۶) = ۱۸/۹۲ \end{cases}$$

همان گونه که مشاهده می‌شود تمامی نقاط در حدود بازه ۰ تا ۴ قرار دارند. اگر نقطه‌ای خارج از این محدوده بود (خارج از var_{hi} و ar_{low})، مختصاتش را با حدود بازه تنظیم می‌کنیم؛ یعنی اگر کمتر از حد پایین باشد، برابر حد پایین قرار می‌دهیم و اگر بیشتر از حد بالا باشد، برابر حد بالا قرار می‌دهیم. یکی از دلایل خارج از حدود قرار گرفتن متغیرها، می‌تواند انتخاب نامناسب F باشد. برای این مسأله F های بزرگتر از ۱ باعث خروج متغیرها از بازه می شود.

۷-۱ : مقدار بهینه تابع هدف و نقطه بهینه سراسری به‌روز رسانی می‌شود:

Max Profit= 22.85

Goal Point= (2.9, 3.8)

Global Best Cuckoo = (2.9, 3.8)

Global Max Profit = 22.85

۸-۱ : بررسی شرط توقف :

If Iteration < Max Itr and Max Profit < accuracy, Then : Iteration + 1

1< 2 and 22.85 < 32 , Then : Iteration =2

تکرار دوم:

۱-۲ : تعداد تصادفی تخم به هر فاخته با توجه به فرمول اختصاص می‌یابد:

$$\begin{bmatrix} Number of Eggs, Cuckoo۱ \\ Number of Eggs, Cuckoo۲ \\ Number of Eggs, Cuckoo۳ \\ Number of Eggs, Cuckoo۴ \\ Number of Eggs, Cuckoo۵ \\ Number of Eggs, Cuckoo۶ \end{bmatrix} : \begin{bmatrix} ۲ \\ ۳ \\ ۲ \\ ۳ \\ ۴ \\ ۵ \end{bmatrix}$$

۲-۲ : بیشینه شعاع تخم‌گذاری هر فاخته را با توجه به فرمول به‌دست می‌آید:

$$ELR_{Cuckoo۱} = ۵ \times \frac{۲}{۱۶}(۴ - ۰) = ۲/۵$$

$$ELR_{Cuckoo۲} = ۵ \times \frac{۳}{۱۶}(۴ - ۰) = ۳/۷۵$$

$$ELR_{Cuckoo۳} = ۵ \times \frac{۲}{۱۶}(۴ - ۰) = ۲/۵$$

$$ELR_{Cuckoo۳} = ۵ \times \frac{۳}{۱۶}(۴ - ۰) = ۳/۷۵$$

$$ELR_{Cuckoo۳} = ۵ \times \frac{۴}{۱۶}(۴ - ۰) = ۵$$

$$ELR_{Cuckoo۳} = ۵ \times \frac{۲}{۱۶}(۴ - ۰) = ۲/۵$$

۳-۲ : حال هر فاخته به طور تصادفی در محدوده شعاع خود با توجه به حدود بازه متغیرهای مسأله، شروع به تخم‌گذاری می‌نماید. علاوه بر آن، برازندگی هر تخم نیز ارزیابی می‌گردد.

جدول ۳-۳: مختصات تخم‌ها در تکرار دوم

فاخته‌ها	مختصات تخم‌ها	مقدار تابع هدف در محل تخم‌گذاری
فاخته اول	Egg۱ : (۳٫۳, ۳) Egg۲ : (۲٫۸, ۳٫۱)	$f(X) = ۱۹٫۸۹$ $f(X) = ۱۷٫۴۵$
فاخته دوم	Egg۱ : (۰, ۴) Egg۲ : (۲, ۳٫۷) Egg۳ : (۲٫۵, ۳)	$f(X) = ۱۶$ $f(X) = ۱۷٫۶۹$ $f(X) = ۱۵٫۲۵$
فاخته سوم	Egg۱ : (۱, ۴) Egg۲ : (۱٫۸, ۳٫۷)	$f(X) = ۱۷$ $f(X) = ۱۶٫۹۳$
فاخته چهارم	Egg۱ : (۳٫۲۵, ۲) Egg۲ : (۱, ۱) Egg۳ : (۱٫۵, ۰٫۲)	$f(X) = ۱۴٫۵۶$ $f(X) = ۲$ $f(X) = ۲٫۲۹$
فاخته پنجم	Egg۱ : (۰٫۵, ۰٫۵) Egg۲ : (۲, ۲) Egg۳ : (۳, ۴) Egg۴ : (۳٫۵, ۴)	$f(X) = ۰٫۵$ $f(X) = ۸$ $f(X) = ۲۵$ $f(X) = ۲۸٫۲۵$
فاخته ششم	Egg۱ : (۱, ۰٫۲) Egg۲ : (۳, ۴)	$f(X) = ۱٫۰۴$ $f(X) = ۲۵$

بنابراین بردار جواب بصورت زیر حاصل می گردد:

$$ProfitVector = [۲۲٫۸۵, ۱۹٫۸۹, ۱۷٫۴۵, ۱۴٫۶۴, ۱۶, ۱۷٫۶۹, ۱۵٫۲۵, ۱۲٫۳۱, ۱۷, ۱۶٫۹۳, ...$$

$$۲۰٫۳۱, ۱۴٫۵۶, ۲, ۲٫۲۹, ۱۷٫۷۱, ۰٫۵, ۸, ۲۵, ۲۸٫۲۵, ۱۸٫۹۲, ۱٫۰۴, ۲۵]$$

۲-۴ : با توجه به این‌که در هر مرحله حداکثر ۶ فاخته می‌توانند زنده باشند، بنابراین از این ۲۲ جواب ۱۶ جواب از بدترین‌ها حذف می‌شوند:

$$ProfitVector = [۲۲٫۸۵, ۱۹٫۸۹, ۲۰٫۳۱, ۲۵, ۲۸٫۲۵, ۲۵]$$

۲-۵ : مقدار بهینه تابع هدف و نقطه بهینه سراسری به‌روز رسانی می‌شوند:

Max Profit = 28.25

Goal Point = (3.5 , 4)

Global Best Cuckoo = Goal Point = (3.5 , 4)

Global Max Profit = Max Profit = 28.25

۲-۶ : بررسی شرط توقف:

If Iteration < Max Iter and Max Profit < accuracy, Then : Iteration + 1

2 = 2 and 28.25 < 32 , Then : stop

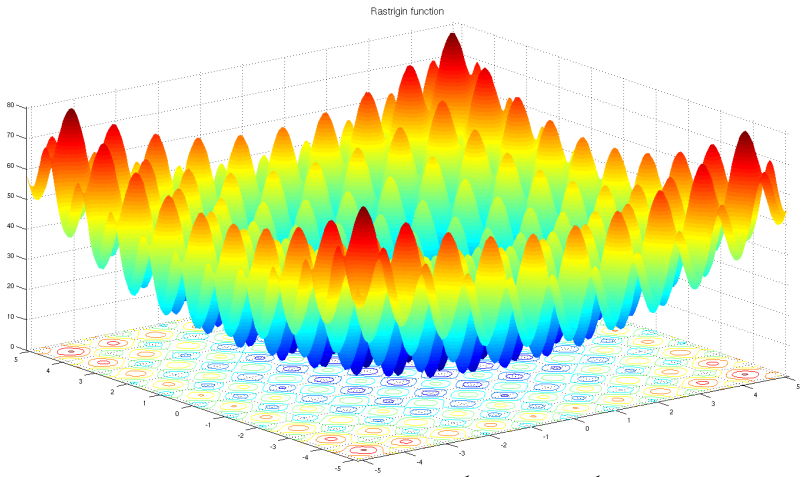
با برقراری شرط توقف، به جواب بهینه ۲۸٫۲۵ با مختصات (۳٫۵, ۴) می‌رسیم.

۳-۵ مقایسه دو الگوریتم فاخته

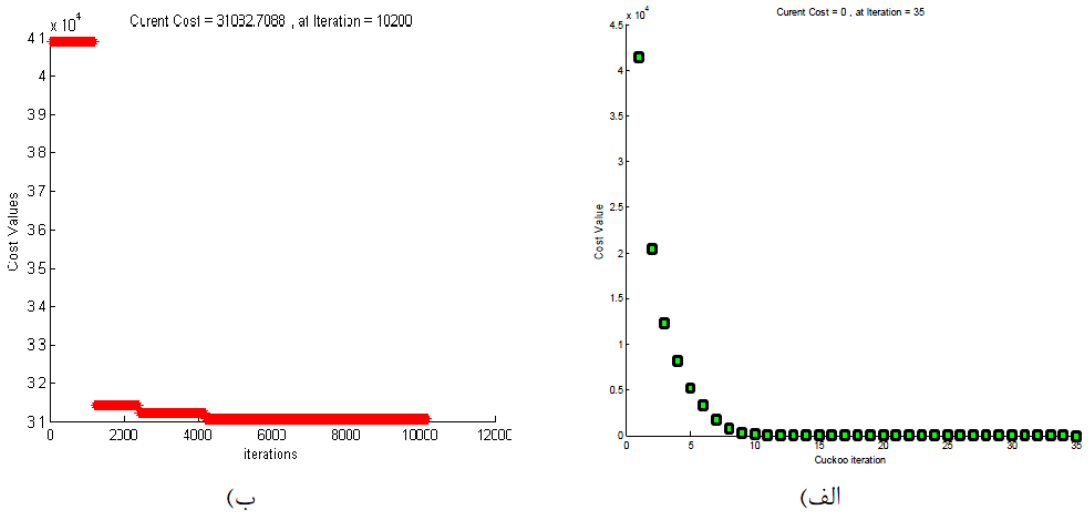
پس از بررسی الگوریتم جستجوی فاخته و الگوریتم بهینه‌سازی فاخته، در این بخش مقایسه‌ای که بین دو الگوریتم انجام شده است [۱۶]، را می‌آوریم.

به منظور روشن شدن تفاوت COA و CS نتایج کمینه‌سازی برای تابع رستریجین [۱] با ضابطه زیر، در ۱۰۰۰ بعد گزارش می‌شود.

$$f = ۱۰n + \sum_{i=۱}^{n} x_i^۲ - ۱۰cos(۲\pi x_i) \qquad -۵ \le x_i \le ۵$$



شکل ۳-۹: شکل تابع رستریجین با دو متغیر



ب) الف)

شکل ۳-۱۰: نمودار نتایج بهینه‌سازی تابع رستریجین با COA (الف) و CS (ب)

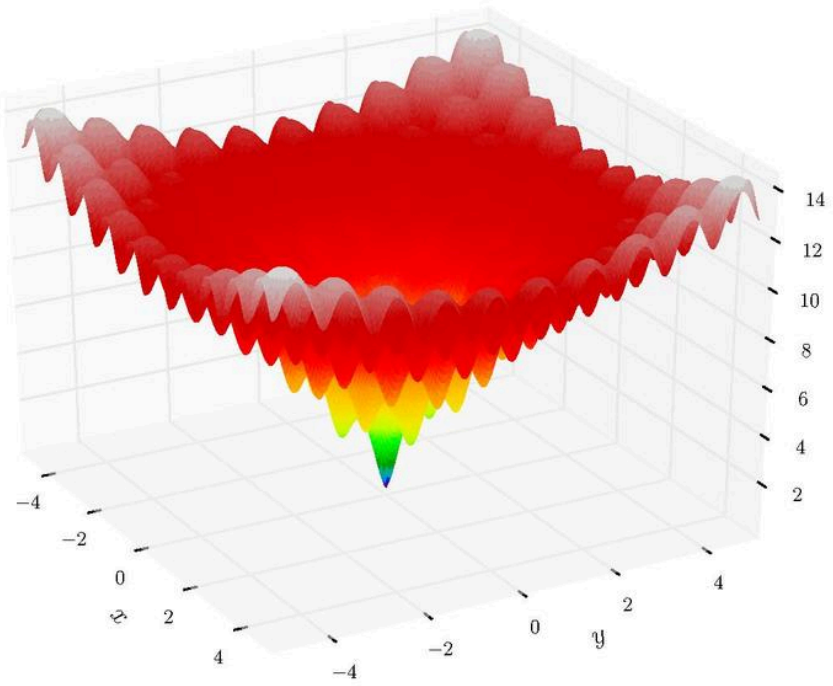
همان طور که در شکل ۳-۱۰ دیده می‌شود الگوریتم بهینه‌سازی فاخته (COA) در ۳۵ تکرار مقدار بهینه تابع رستریجین

[۱] Rastrigin

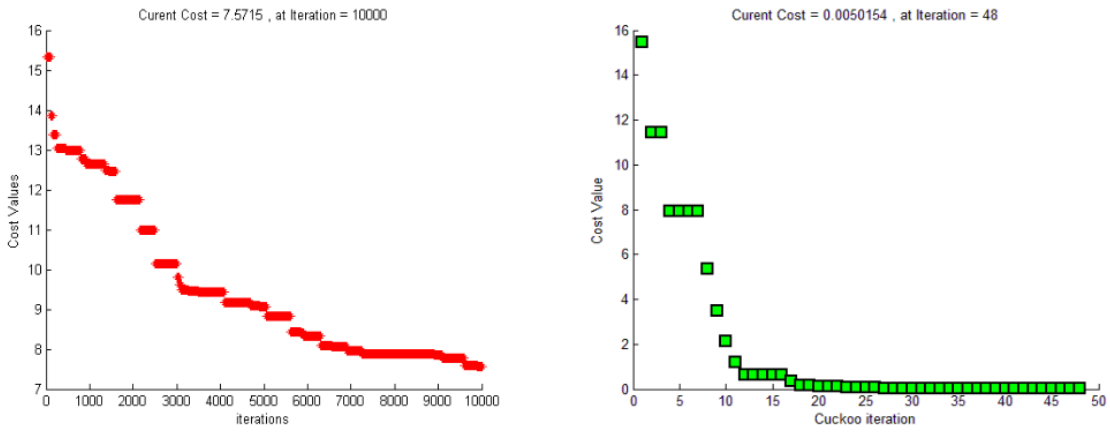
در ۱۰۰۰ بعد را نتیجه می‌دهد، در حالی که الگوریتم جستجوی فاخته (CS) در تکرار ۱۰۲۰۰ هنوز به نزدیکی مقدار بهینه هم نرسیده است.

در ادامه مقایسه دیگری از نتایج حاصل از کمینه‌سازی تابع اکلی [۱] با ضابطه زیر، در ۵۰۰۰ بعد را خواهیم داشت.

$$f = -۲۰exp(-۰/۲\sqrt{(۱/n\sum_{i=۱}^{n} x_i^۲)}) - exp(۱/n\sum_{i=۱}^{n} cos۲\pi x_i) + ۲۰ + e \quad -۵ \le x_i \le ۵$$



شکل ۳-۱۱: شکل تابع اکلی با دو متغیر



ب) الف)

شکل ۳-۱۲: نمودار نتایج بهینه‌سازی تابع اکلی با COA (الف) و CS (ب)

[۱] Ackley

همان طور که در شکل ۳-۱۲ نیز مشاهده می‌شود، الگوریتم بهینه‌سازی فاخته در ۴۸ تکرار مقدار بهینه را گزارش می‌کند و الگوریتم جستجوی فاخته در ۱۰۰۰۰ تکرار، هنوز به جواب بهینه نرسیده است.

۳-۶ محاسن الگوریتم بهینه‌سازی فاخته نسبت به الگوریتم جستجوی فاخته

الگوریتم بهینه‌سازی فاخته قادر است با ترکیب چندین عملگر که کمک شایانی به جستجوی محلی در حین جستجوی سراسری می‌کنند، به جواب‌های دقیق‌تر و قابل اعتمادتری دست یابد. فرآیند تخم‌گذاری در الگوریتم بهینه‌سازی فاخته، یک فرآیند جستجوی محلی است، که با توزیع تخم‌ها در اطراف نقطه بهینه فعلی به COA کمک می‌کند تا در بهینه محلی متوقف نشود. خوشه‌بندی در COA به فاخته‌ها کمک می‌کند تا محیط را به سرعت به چندین بخش تقسیم کرده و بهترین ناحیه را به صورت تخمینی مشخص نمایند. این ناحیه به احتمال زیاد شامل نقطه بهینه سراسری می‌باشد. سپس تمام فاخته‌ها به سمت این ناحیه مهاجرت می‌کنند و آن ناحیه را به صورت بهتری جستجو می‌نمایند، این امر موجب همگرایی سریع‌تر الگوریتم بهینه‌سازی فاخته می‌شود.

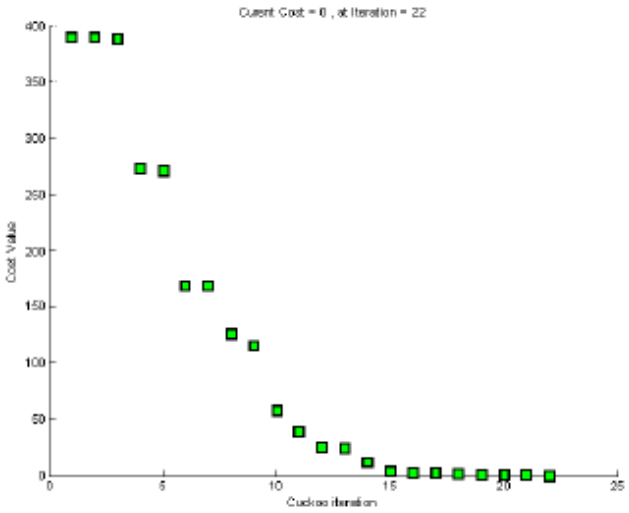
عملگرهای الگوریتم جستجوی فاخته، از الگوریتم کرم شب‌تاب [۱] گرفته شده است و شباهت کمی به روش زندگی فاخته‌ها دارد. این الگوریتم در جستجوی محلی نسبت به جستجوی سراسری قوی‌تر است. در مقالات مختلف از ترکیب جستجوی فاخته با الگوریتم‌های دیگری که توانایی جستجوی سراسری بهتری دارند، استفاده شده است تا نتایج خوبی به‌دست آید. نمونه‌ای از این مقالات مرجع[۱۵] است که در آن از الگوریتم تکامل تفاضلی (DE) برای جستجوی سراسری و از الگوریتم جستجوی فاخته برای جستجوی محلی استفاده شده است. با توجه به گزارش‌های آورده شده، ما در این پایان‌نامه تمرکز خود را بر روی الگوریتم بهینه‌سازی فاخته معطوف می‌نماییم.

۳-۶-۱ توانایی همگرایی COA

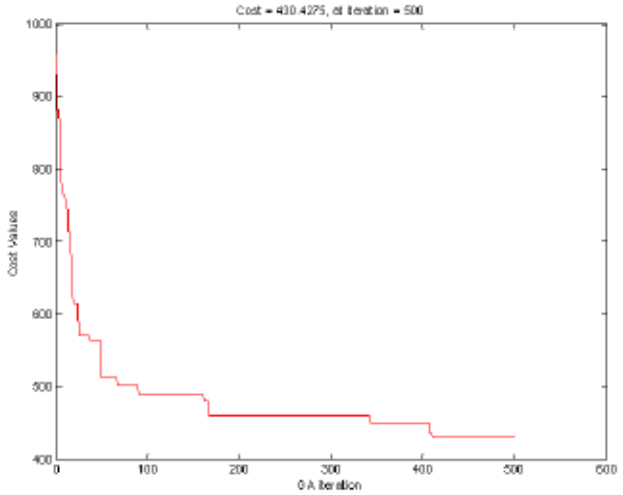
در این بخش هدف بررسی توانایی همگرایی COA در حل مسائل بهینه‌سازی با ابعاد بزرگ می‌باشد. مسأله مورد سنجش، پیدا کردن نقطه کمینه برای تابع رستریجین که در بخش ۳-۵ معرفی شد، با ۱۰۰ متغیر می‌باشد. همان‌طور که گفته شد COA یک الگوریتم ماکزیمم‌سازی است، لذا برای استفاده از آن در این مسأله، یک علامت منفی در تابع هزینه ضرب می‌کنیم و با تعداد اولیه ۵ فاخته الگوریتم را شروع می‌کنیم. به منظور بررسی توانایی همگرایی COA ، مسأله را با دو الگوریتم GA و PSO نیز حل می‌نماییم، برای الگوریتم‌های GA و PSO جمعیت اولیه ۵۰ را در نظر می‌گیریم.

---

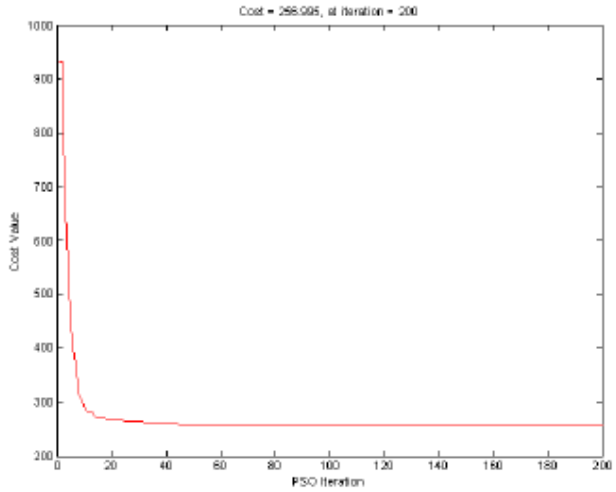
[۱] firefly



شکل ۳-۱۳: بهینه‌سازی با الگوریتم بهینه‌سازی فاخته



شکل ۳-۱۴: بهینه‌سازی با الگوریتم ژنتیک



شکل ۳-۱۵: بهینه‌سازی با الگوریتم ازدحام ذرات

با توجه به شکل‌های ۳-۱۳ و ۳-۱۴ و ۳-۱۵ ملاحظه می‌شود که COA در تکرار ۲۲ به جواب بهینه قطعی رسیده است ولی GA با گذشت ۵۰۰ تکرار و PSO با طی کردن ۲۰۰ تکرار هنوز به نزدیکی جواب بهینه هم نرسیده‌اند، که این نشان‌دهنده همگرایی COA در تعداد تکرارهای کمتر نسبت به دو روش مذکور در این مسأله بهینه‌سازی می‌باشد [۱۶].

۳-۷ کاربردهای الگوریتم بهینه‌سازی فاخته

از COA برای حل مسائل زیادی در زمینه‌های مختلف استفاده شده است که می‌توان به نمونه‌های زیر اشاره کرد [۱۷] :

- مسائل طراحی چیدمان
- مسائل زمان‌بندی و توالی عملیات
- طراحی شبکه‌های هوشمند
- طراحی کنترلرهای SISO و MIMO
- مهندسی عمران
- صنعت
- نیروگاه‌ها
- تشخیص بیماری‌ها

و . . .

۳-۸ الگوریتم بهینه‌سازی فاخته اصلاح‌شده

این الگوریتم در سال ۲۰۱۲ توسط هومر کهرمانلی [۱] [۱۸] ارائه شد. در COA اصلاح شده روشی جهت کاهش تدریجی شعاع تخم‌گذاری فاخته‌ها ارائه شده است. کاهش مرحله‌ای شعاع تخم‌گذاری می‌تواند منجر به بهبود جستجو شده و دقت جواب‌ها را بالا ببرد. در الگوریتم استاندارد COA این تغییر را می‌توان با کاهش تدریجی ضریب آلفا انجام داد.

[۱] Humar Kahramanli

در الگوریتم اصلاح‌شده ضریب شعاع تخم‌گذاری که در این‌جا با e نشان داده شده است، به صورت زیر کاهش می‌یابد:

$$t = \frac{maxiter}{C}$$

که C یک مقدار ثابت از بازه [۰, ۲۰) و $maxiter$ ماکزیمم تعداد تکرارهاست. از این رو:

$$a = [\frac{iteration}{t}] + ۱$$

بنابراین e در هر تکرار به صورت زیر اصلاح می‌شود:

$$e_{new} = \frac{e_{old}}{a}$$

فصل ۴

حل مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود

۴-۱ مقدمه

با توجه به کارایی زیادی که الگوریتم‌های فراابتکاری برای حل مسائل پیچیده و بزرگ دارند و جواب‌هایی با کیفیت بالا را در زمانی کوتاه برای این مسائل ارائه می‌دهند، از این الگوریتم‌ها برای حل مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود که از مسائل NP-hard است، نیز استفاده شده است.

هدف اصلی تحقیق، سنجش کارایی الگوریتم بهینه‌سازی فاخته برای حل UFLP می‌باشد. برای این منظور نتایج الگوریتم بهینه‌سازی فاخته را با نتایج دو الگوریتم ژنتیک و بهینه‌سازی ازدحام ذرات برای حل UFLP ، مقایسه می‌نماییم. ابتدا با اعمال تغییراتی بر روی نسخه اصلی الگوریتم بهینه‌سازی فاخته که جهت حل مسائل بهینه‌سازی پیوسته ارائه شده است، گسسته‌سازی لازم برای حل مسائل بهینه‌سازی گسسته را انجام می‌دهیم و سپس مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود را با استفاده از الگوریتم بهینه‌سازی فاخته و دو الگوریتم فراابتکاری مذکور حل می‌نماییم. نتایج به‌دست آمده از چند نقطه نظر با هم مقایسه می‌شوند.

۴-۲ حل مسائل گسسته با الگوریتم‌های بهینه‌سازی

مسائل بهینه‌سازی از دیدگاه‌های متفاوتی طبقه‌بندی می‌شوند، که مهمترین آن از دیدگاه پیوسته یا گسسته بودن مسائل می‌باشد. طبق این دیدگاه از آن‌جا که جواب مسائل بهینه‌سازی در فضاهای پیوسته یا گسسته مطرح می‌شود، الگوریتم‌های تکاملی که مطرح می‌شوند بایستی قادر به حل مسائل در هر دو فضا باشند. از جمله الگوریتم‌هایی که این توانایی را دارند می‌توان به الگوریتم ژنتیک و الگوریتم ازدحام ذرات اشاره نمود. در واقع بیشتر الگوریتم‌های تکاملی در اوایل پیدایش، به صورت پیوسته مطرح می‌شوند. ولی بسیاری از مسائل وجود دارند که ماهیت گسسته دارند، اگر چه مسائل گسسته‌ای وجود دارند

که می‌توانند در فضای پیوسته حل شوند اما در مقابل مسائلی هم هستند که این قابلیت را ندارند، پس در چنین شرایطی نیاز به الگوریتم‌های گسسته احساس می‌شود.

در این بخش سه دسته از روش‌های حل مسائل گسسته با الگوریتم‌های بهینه‌سازی، با استفاده از گسسته‌سازی بر روی الگوریتم‌های با ماهیت پیوسته، شرح داده می‌شود. گسسته‌سازی‌های صورت گرفته، در واقع مطابقت الگوریتم با فضای گسسته است. محور اصلی پایان‌نامه با توجه به ساختار و فضای مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود، تغییر عملگرهای الگوریتم جهت تولید اعداد گسسته از نوع صفر و یک می‌باشد. در ادامه، سه مدل گسسته‌سازی صورت گرفته بر روی الگوریتم بهینه‌سازی فاخته، بررسی و از الگوریتم گسسته فاخته برای حل مسأله مکان‌یابی تسهیلات با ظرفیت استفاده می‌شود.

۴-۲-۱ حل مسائل گسسته با ماهیت پیوسته الگوریتم

این روش ساده‌ترین نوع حل مسائل گسسته با الگوریتم‌های بهینه‌سازی است، زیرا عملاً مسأله گسسته با الگوریتم پیوسته حل می‌شود. به عبارتی در این دسته از مسائل نگاشت فضا صورت می‌گیرد. یعنی در ابتدا فضای جمعیت به صورت گسسته تولید می‌شود و الگوریتم بدون تغییر با همان ماهیت پیوسته روی این فضای گسسته اعمال می‌گردد و خروجی الگوریتم مطابق فضای گسسته مسأله گزارش می‌شود.

۴-۲-۲ حل مسائل گسسته با نمایش بردار عدد صحیح در الگوریتم

روش دیگر حل مسائل گسسته که در آن، هر راه حل از فضای جستجو به وسیله برداری از اعداد صحیح نمایش داده می‌شود، استفاده از الگوریتم‌هایی با ماهیت گسسته است. در این روش عملگرهای الگوریتم طوری تعریف شده‌اند که با اعمال بر روی بردارهای صحیح ، بردارهای صحیح دیگری از فضای جستجو را می‌سازند. در صورتی که الگوریتم پیوسته باشد، باید عملگرهای الگوریتم را به گونه‌ای تغییر داد که قادر به تولید اعداد صحیح باشد. مسائل قابل حل در این دسته خود به دو بخش تقسیم می‌شوند: مسائل جایگشتی و مسائل غیرجایگشتی.

در مسائل جایگشتی، هر راه‌حل در فضای جستجو متناظر است با یک جایگشت. در این حالت به جواب‌های مسأله اعدادی اختصاص داده می‌شود که فقط یک کد برای شناختن آن‌ها هستند. سپس ترتیبی از این کدهاجواب مسأله را تشکیل می‌دهد. مسأله فروشنده دوره‌گرد و مسائل زمان‌بندی، مثال‌های واضح این دسته می‌باشند.

در کنار مسائل جایگشتی، مسائلی وجود دارند که غیرجایگشتی می‌باشند. در این نوع نمایش، هر راه‌حل در فضای جستجو متناظر است با یک بردار صحیح که ممکن است مقادیر تکراری هم در این بردارها وجود داشته باشد. مسأله رنگ‌آمیزی گراف نمونه بارز این نوع نمایش است.

۳-۲-۴ حل مسائل گسسته با نمایش دودویی در الگوریتم

برای مسائل گسسته از نوع دودویی، که حالت خاصی از مسائل گسسته غیرجایگشتی هستند، الگوریتم به گونه‌ای تغییر می‌یابد که هر راه حل از فضای جستجو به وسیله‌ی رشته‌های ۰ و۱ کد شوند. نمایش دودویی در بهینه‌سازی گسسته خیلی رایج است به این علت که بسیاری از مسائل، قابلیت فرموله شدن با متغیرهای دودویی را دارند. مسأله کوله‌پشتی و دوبخشی کردن گراف و هم‌چنین مسأله مکان‌یابی تسهیلات با ظرفیت نامحدود از نمونه مسائل این دسته‌اند.

۳-۴ گسسته‌سازی‌های انجام شده بر روی الگوریتم بهینه‌سازی فاخته

نسخه اصلی الگوریتم بهینه‌سازی فاخته جهت حل مسائل بهینه‌سازی پیوسته ارائه شده است، اما این الگوریتم با اعمال تغییراتی قابلیت حل مسائل بهینه‌سازی گسسته را نیز دارد. یکی از مسائل شناخته شده و مشهور که با این الگوریتم حل شده، مسأله رنگ‌آمیزی گراف است که یکی از حالت‌های خاص مسأله برچسب‌گذاری گراف می‌باشد.[۱۶]

برای حل مسائل بهینه‌سازی گسسته باCOA لازم است تا موقعیت‌های سکونت فاخته‌ها از حالت پیوسته به حالت گسسته تبدیل شود. ساده‌ترین روش انجام چنین کاری گردکردن مختصات سکونت فاخته‌ها به سمت نزدیک‌ترین عدد صحیح می‌باشد. در این روش به نوعی نگاشت فضا صورت می‌گیرد و الگوریتم بدون تغییر در عملگرها روی این فضای گسسته اعمال می‌گردد.

در این بخش ایده‌های مربوط به گسسته‌سازی الگوریتم بهینه‌سازی فاخته با توجه به طبقه‌بندی ذکر شده، را می‌آوریم.

۳-۴-۱ گسسته‌سازی با استفاده از نگاشت فضا

در این روش مقداردهی اولیه موقعیت‌ها در COA باز تعریف می‌شود و نگاشت فضا صورت می‌گیرد و تغییر ندادن دیگر بخش‌های الگوریتم منجر به اعمال الگوریتم پیوسته فاخته در فضای گسسته مسأله می‌شود. این ایده از عملیات ریاضی معکوس گرفته شده است که تعدادی عدد تصادفی بین حداقل و حداکثر مقدار تولید می‌کند. عملیات معکوس به شرح زیر است:

فرض کنید X_{norm} عددی تصادفی در بازه [۰, ۱] باشد، برای تولید یک عدد در محدوده $[var_{low}, var_{hi}]$ داریم:

$$x = (var_{hi} - var_{low}) \times X_{norm} + var_{low}$$

این رابطه اعداد مابین [۰, ۱] را به محدوده $[var_{low}, var_{hi}]$ نگاشت می‌دهد. اما اعداد تولید شده، ممکن است

اعشاری باشند. برای داشتن اعداد صحیح متناظر باید قسمت اعشاری به شرح زیر حذف شود:

$$x = floor((var_{hi} - var_{low}) \times X_{norm} + var_{low})$$

هم‌چنین برای تضمین تولید عدد var_{low} فرمول به شکل زیر کامل‌تر می‌شود:

$$x = max\{floor((var_{hi} - var_{low}) \times X_{norm} + var_{low}), var_{low}\}$$

در نهایت با رابطه اخیر هر عددی بین [۰, ۱] به عدد صحیحی در محدوده $[var_{low}, var_{hi}]$ نگاشت پیدا می‌کند. حال با این نگاشت می‌توان COA را برای مسائل با فضای گسسته در هر بازه‌ای اعمال کرد.

۴-۳-۲ گسسته‌سازی با استفاده از تغییر عملگر

COA الگوریتمی است که ذاتاً برای جستجوی فضای پیوسته مطرح شده است و مسائل بهینه‌سازی ترکیبیاتی از جمله مسائلی می‌باشند که در فضای گسسته مطرح می‌شوند، از این رو برای اعمال الگوریتم COA به فضای جستجوی گسسته برای حل این مسائل، عملگرهای محاسباتی استاندارد COA نیازمند دوباره تعریف شدن بر روی فضای گسسته می‌باشند. به طور کلی در این روش تغییر لازم و ضروری در الگوریتم پایه COA جهت بهینه‌سازی مسائل ترکیبیاتی، تغییر رابطه مهاجرت است. اما با توجه به متفاوت بودن ماهیت مسائل جایگشتی و غیرجایگشتی، یک تغییر دیگر در مسائل جایگشتی لازم است و آن تغییر در روال تخم‌گذاری است.

۴-۳-۳ گسسته‌سازی دودویی

در این بخش جهت استفاده از COA پیوسته در فضای دودویی، عملگر مهاجرت به صورت زیر باز تعریف می‌شود. X_{Goal} و $X_{Current}$ به ترتیب نقطه هدف جاری و موقعیت جاری یک فاخته در جمعیت می‌باشند. موقعیت بعدی فاخته (X_{Next}) به شکل زیر محاسبه می‌شود:

$$(X_{Next})_j = (X_{Current})_j + rand \times ((X_{Goal})_j - (X_{Current})_j)$$

$$j = ۱, ..., N_{var}$$

برای این‌که موقعیت جدید برای فضای دودویی مناسب باشد، با استفاده از تابع سیگموید رابطه (۴-۱) ، X_{Next} به محدوده [۰, ۱] نگاشت داده می‌شود و سپس بر اساس رابطه (۴-۲) ، مقدار موقعیت به مقدار دودویی ۰ و ۱ تغییر می‌یابد.

$$S = \frac{۱}{۱ + e^{-(X_{Next})_j}} \tag{۱-۴}$$

$$\begin{cases} if \quad S \leq rand \quad Then \quad (X_{Next})_j = ۰ \\ if \quad S > rand \quad Then \quad (X_{Next})_j = ۱ \end{cases} \tag{۲-۴}$$

۴-۴ حل UFLP با الگوریتم‌های فراابتکاری

در این بخش، الگوریتم ژنتیک، الگوریتم بهینه‌سازی ازدحام ذرات و الگوریتم بهینه‌سازی فاخته را برای حل UFLP به کار می‌بریم و نتایج را از چند منظر مقایسه می‌نماییم.

۴-۴-۱ الگوریتم ژنتیک برای حل UFLP

الگوریتم ژنتیک روش خوبی برای حل مسأله مکان‌یابی با ظرفیت نامحدود است. برای این منظور از جعبه‌ابزار الگوریتم ژنتیک در نرم‌افزار MATLAB استفاده شده است.

برای ورود پارامترها به جعبه ابزار الگوریتم ژنتیک از تابع $gaoptimset$ استفاده می‌شود. برای مثال، برای تغییر اندازه جمعیت[۱]، در پنجره دستورات MATLAB عبارت زیر را وارد کنید و به جای عبارت $value$، مقدار دلخواه را قرار دهید.

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

برای این مسأله اندازه جمعیت برابر با ۵۰ کروموزوم و نوع کدگذاری[۲]، صفر و یک[۳] در نظر گرفته شده است. مقدار یک در هر ژن از کروموزوم، تأسیس یک سرویس‌دهنده را در آن مکان نشان می‌دهد و مقدار صفر در هر ژن نیز نشان‌دهنده‌ی این است که سرویس‌دهنده‌ای در آن مکان تأسیس نشده است.

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

[۱] Population Size [۲] Population Type [۳] Bit String

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

در پیاده‌سازی الگوریتم ژنتیک، برازندگی هر کروموزوم، با استفاده از تابع اصلی که قصد مینیمم کردن آن را داریم، محاسبه می‌شود.

۴-۴-۲ الگوریتم بهینه‌سازی ازدحام ذرات برای حل UFLP

در مرجع [۱۹] الگوریتم ازدحام ذرات برای حل UFLP پیشنهاد شده است. الگوریتم PSO برای هر ذره سه بردار موقعیت (X_i)، سرعت (V_i) و وضعیت مکان (Y_i) را در نظر می‌گیرد. برای مسأله‌ای با n سرویس‌دهنده، هر ذره شامل n بعد است. در بردار $Y_i = [y_{i1}, ..., y_{in}]$، y_{ik} باز یا بسته بودن kامین مکان از iامین ذره را نشان می‌دهد. بردارهای موقعیت و سرعت از روابط زیر بدست می‌آیند:

$$x_{ij} = x_{min} + (x_{max} - x_{min}) \times r_1 \quad i = 1, ..., N_s \ , \ j = 1, ..., D$$
$$v_{ij} = v_{min} + (v_{max} - v_{min}) \times r_2$$

که $x_{min} = -10$، $x_{max} = 10$، $v_{min} = -4$، $v_{max} = 4$ و r_1 و r_2 به طور تصادفی از بازه‌ی $[0, 1]$ برای هر بعد ذره انتخاب می‌شوند. جمعیت اولیه الگوریتم ازدحام ذرات نیز برای هر مثال برابر با تعداد اماکن سرویس‌دهنده، ضرایب C_1 و C_2 برابر ۲/۰۵ در نظر گرفته شده است. برای ساختن جواب اولیه، بردار موقعیت به یک بردار صفر و یک تبدیل می‌شود $(X_i \longrightarrow Y_i)$، برای این منظور از رابطه زیر استفاده می‌شود. در این رابطه قدرمطلق بردار موقعیت را بر عدد ۲ تقسیم و باقی‌مانده به سمت نزدیکترین عدد صحیح گرد می‌شود.

$$Y_i = floor(|X_i| mod 2)$$

۴-۴-۳ الگوریتم بهینه‌سازی فاخته برای حل UFLP

هر سه روش گسسته‌سازی ذکر شده در بخش قبل، بر روی الگوریتم بهینه‌سازی فاخته اعمال شد و بعد از پیاده‌سازی در نرم‌افزار MATLAB و حل چند مثال از UFLP با هر سه روش، ملاحظه شد که گسسته‌سازی با استفاده از تغییر عملگر جواب‌های دقیق‌تری ارائه می‌دهد، لذا برای حل UFLP با استفاده از الگوریتم بهینه‌سازی فاخته، این روش گسسته‌سازی را مورد استفاده قرار می‌دهیم. پارامترهای الگوریتم به صورت زیر مقداردهی می‌شوند:

1. $Var_{low} = 0$
2. $Var_{hi} = 1$

3. $NumCuckoos = 5$
4. $MinNumberofEggs = 10$
5. $MaxNumberofEggs = 15$
6. $MaxIter = 200$
7. $K - ClusterNum = 1$
8. $MotionCoeff(F) = 1$
9. $MaxNumberofCuckoos = 100$
10. $RadiusCoeff(\alpha) = 1$

الگوریتم بهینه‌سازی فاخته به منظور حل UFLP ، برای هر فاخته یک ماتریس دودویی محل سکونت در نظر می‌گیرد، که برای مسأله‌ای با n تسهیلات، این ماتریس $n \times ۱$ می‌باشد و هر مؤلفه از آن نشان‌دهنده باز یا بسته بودن تسهیلات در آن مکان است. مقداردهی محل سکونت اولیه فاخته‌ها به صورت زیر انجام می‌شود:

$$X = randi([۰, ۱], ۱, n)$$

عبارت $randi$ یکی از دستورات زبان برنامه‌نویسی MATLAB است که در این‌جا یک آرایه $n \times ۱$ از اعداد صحیح بازه $[۰, ۱]$ تولید می‌کند. میزان مناسب بودن موقعیت هر فاخته، با استفاده از تابع اصلی که قصد مینیمم کردن آن را داریم، محاسبه می‌شود. ولی همان‌طور که قبلاً ذکر شد COA یک الگوریتم ماکزیمم‌سازی است، لذا در روند الگوریتم برای استفاده از تابع هزینه، تابع اصلی در یک منفی ضرب می‌شود.

بعد از فرآیند تخم‌گذاری و ارزیابی مقدار تابع هدف در هر موقعیت، خوشه‌بندی و مهاجرت انجام می‌شود. در عملگر مهاجرت برای تولید X_{Next}، ابتدا از میان درایه‌های نابرابر $X_{Current}$ نسبت به X_{Goal} ، تعدادی تصادفی را برای تغییر انتخاب نموده سپس به طور تصادفی تغییر می‌دهیم.

۴-۴-۴ نتایج

سه الگوریتم مذکور در نرم‌افزار MATLAB در یک محیط یکسان، با مقدار پارامترهایی که تعیین شد، پیاده‌سازی گردیدند و نتایج آن‌ها برای حل UFLP از منظرهای مختلف مورد بررسی قرار گرفت. هر روش روی ۹ مثال از OR-Library [۲۰] که مشخصات آن‌ها در جدول ۴-۱ درج شده است، ۱۰۰ بار اجرا شدند و نتایج در ادامه آورده شده است.

جدول ۴-۱: مشخصات مثال‌های شماره ۱ تا ۹

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

لازم به ذکر است که در اندازه مثال‌ها عدد اول تعداد تسهیلات یا انبارهاست و عدد دوم تعداد مشتری‌ها را نشان می‌دهد. برای مثال در مسأله cap۷۲ ، ۱۶ انبار و ۵۰ مشتری داریم.

نتایج بر اساس شرط توقف حداکثر تعداد تکرار در نظر گرفته شده است. پس از ۱۰۰ بار اجرای هر روش روی ۹ مثال، جدول ۴-۲ میانگین مقدار تابع هدف، جدول ۴-۳ میانگین خطا نسبت به جواب بهینه، جدول ۴-۴ میانگین زمان اجرا (بر حسب ثانیه) و جدول ۴-۵ میانگین تعداد تکرار را گزارش می‌دهند و ستون سمت راست در هر جدول شماره مثال‌ها را نشان می‌دهد.

Method			
PSO	GA	COA	No.
۹۴۵۷۰۸٫۴۱۲	۹۳۲۶۱۵٫۷۵۰	۹۳۲۶۱۵٫۷۵۰	۱
۹۹۴۰۷۳٫۸۶۲	۹۸۱۶۴۹٫۳۵۰	۹۷۷۹۹٫۴۰۰	۲
۱۰۴۷۸۶۷٫۵۲۵	۱۰۱۴۴۹۱٫۴۰۰	۱۰۱۰۶۴۱٫۴۵۰	۳
۸۲۲۰۸۶٫۱۵۰	۷۹۶۶۴۸٫۴۳۸	۷۹۶۶۴۸٫۴۳۸	۴
۹۱۲۴۹۰٫۰۲۵	۸۵۴۷۰۴٫۲۰۰	۸۵۴۷۰۴٫۲۰۰	۵
۹۱۵۱۲۳٫۱۳۸	۸۹۵۰۲۷٫۱۸۸	۸۹۴۰۰۸٫۱۳۷	۶
۸۶۲۶۵۲٫۲۶۲	۷۹۶۵۳۹٫۵۰۰	۷۹۳۴۳۹٫۵۶۳	۷
۹۳۰۷۲۱٫۱۲۵	۸۵۲۷۶۲٫۸۷۵	۸۵۱۶۷۰٫۱۲۵	۸
۹۷۶۷۲۲٫۳۵۰	۹۰۰۵۱۴٫۸۸۸	۸۹۶۳۴۷٫۹۱۲	۹
۹۳۴۱۶۰٫۵۳۹	۸۹۱۶۶۱٫۵۱۰	۸۸۹۷۶۳٫۸۸۶	Avg

جدول ۴_۲: MaxIter:Cost

Method			
PSO	GA	COA	No.
۱۳۰۹۲٫۶۶۲	۰٫۰۰۰	۰٫۰۰۰	۱
۱۶۲۷۴٫۴۶۲	۴٫۸۱۲	۰٫۰۰۰	۲
۳۷۲۲۶٫۰۷۵	۴٫۸۱۲	۰٫۰۰۰	۳
۲۵۴۳۷٫۷۱۳	۰٫۰۰۰	۰٫۰۰۰	۴
۵۷۷۸۵٫۸۲۵	۰٫۰۰۰	۰٫۰۰۰	۵
۲۱۳۴۱٫۰۲۶	۰٫۹۹۶	۰٫۱۸۱	۶
۶۹۲۱۲٫۷۰۰	۱٫۲۴۰	۰٫۰۰۰	۷
۷۹۲۲۵٫۸۰۰	۰٫۵۰۷	۰٫۰۷۰	۸
۸۳۶۴۵٫۶۳۸	۲٫۹۷۵	۱٫۳۰۸	۹
۴۴۸۰۴٫۶۵۶	۱٫۷۰۵	۰٫۱۷۳	Avg

جدول ۴_۳: MaxIter:Error

Method			
PSO	GA	COA	No.
۰٫۸۰۱	۰٫۷۸۷	۴٫۵۸۸	۱
۰٫۵۵۶	۰٫۵۵۲	۴٫۵۲۷	۲
۰٫۵۴۰	۰٫۵۳۶	۶٫۳۳۲	۳
۰٫۵۶۴	۰٫۵۵۸	۱۱٫۳۱۹	۴
۰٫۵۵۳	۰٫۵۴۸	۱۱٫۱۳۴	۵
۰٫۵۵۲	۰٫۵۴۶	۱۱٫۲۳۲	۶
۰٫۶۰۳	۰٫۵۸۱	۴۴٫۷۱۱	۷
۰٫۶۰۹	۰٫۵۷۸	۳۹٫۹۶۴	۸
۰٫۷۰۳	۰٫۵۸۰	۳۹٫۴۱۵	۹
۰٫۶۰۹	۰٫۵۸۵	۱۹٫۲۴۷	Avg

جدول ۴ـ۴: MaxIter:RunTime

Method			
PSO	GA	COA	No.
۱٫۰۰۰	۲۰۰٫۰۰۰	۷٫۰۰۰	۱
۱٫۰۰۰	۲۰۰٫۰۰۰	۷٫۰۰۰	۲
۱٫۰۰۰	۲۰۰٫۰۰۰	۹٫۰۰۰	۳
۱٫۰۰۰	۲۰۰٫۰۰۰	۱۱٫۰۰۰	۴
۱٫۰۰۰	۲۰۰٫۰۰۰	۱۱٫۰۰۰	۵
۱٫۰۰۰	۲۰۰٫۰۰۰	۱۱٫۰۰۰	۶
۲٫۰۰۰	۲۰۰٫۰۰۰	۲۳٫۰۰۰	۷
۳٫۰۰۰	۲۰۰٫۰۰۰	۲۱٫۰۰۰	۸
۱۵٫۰۰۰	۲۰۰٫۰۰۰	۲۱٫۰۰۰	۹
۲٫۸۸۹	۲۰۰٫۰۰۰	۱۳٫۴۴۴	Avg

جدول ۴ـ۵: MaxIter:NumIter

با توجه به جدول ۴-۲ ملاحظه می‌شود که میانگین مقدار تابع هدف گزارش شده، توسط COA و GA نسبت به PSO بهتر است و نتایج COA نسبت به دم الگوریتم دیگر دقیق‌تر می‌باشد.

در جدول ۴-۳ میانگین خطای گزارش شده برای COA در مقایسه با GA و PSO کمتر است و PSO خطای زیادی دارد که با افزایش ابعاد مسأله نیز بیشتر می‌شود.

همان‌طور که در جدول ۴-۴ دیده می‌شود، الگوریتم ژنتیک درزمان کوتاه‌تری نسبت به COA و حتی PSO متوقف شده و به شرط توقف یعنی حداکثر تکرار رسیده است.

میانگین تعداد تکرارهای اجرای هر الگوریتم برای ۱۰۰ بار اجرا روی هر مثال، در جدول ۴-۵ آورده شده است. ملاحظه می‌شود که PSO با کمترین تعداد تکرار متوقف شده است و GA تا رسیدن به حداکثر تکرار ادامه یافته است، البته با کمترین زمان اجرا. COA نیز با تعداد تکرارهای کمی متوقف شده است.

با توجه به نتایج ملاحظه می‌شود که الگوریتم بهینه‌سازی فاخته در مقایسه با الگوریتم ژنتیک و الگوریتم بهینه‌سازی ازدحام ذرات، جواب‌هایی با کیفیت بالا در تعداد تکرارهای کم، تولید می‌کند؛ و الگوریتم ژنتیک جواب‌های مطلوبی در

زمانی کوتاه ارائه می‌دهد.

۵-۴-۴ پیشنهاد

کدهای الگوریتم بهینه‌سازی فاخته، در مقایسه با کدهایی مانند کد الگوریتم ژنتیک، کدهای ایده‌آلی نیستند و هیچ نوع بهینه‌سازی از لحاظ کدنویسی بر روی آن اعمال نشده و کدها صرفاً برای کار کردن برنامه و جواب‌دهی، در ساده‌ترین شکل ممکن نوشته شده است. در اکثر بخش‌های برنامه از آرایه‌های داینامیک استفاده شده است که مقداردهی اولیه نشده‌اند و فقط با یک عبارت به صورت $A = [\ ]$ در حافظه شکل گرفته‌اند و در هر تکرار در حلقه for مقداری به آن اضافه می‌شود. از این‌رو می‌توان با کدنویسی حرفه‌ای و رفع این نواقص زمان اجرای الگوریتم را نیز کمتر کرد.

فهرست منابع

[1] Labbe, M, Laport, G, Tanczos, K and Toint, P. Operation Reserch and Decision Aid Methodologies in Traffic and Transportation Management. Springer Science and Business Media., Volume 166, 2013.

[2] Korte, B and Vyggen, J. Combinatorial Optimization. Springer., 2012.

[3] Siselt, H and Marianov, V. Foundations of Location Analysis. Springer Science and Business Media ., Volume 155, 2011.

[4] Sierksma, G and Ghosh, D. Networks in Action: Text and Computer Exercisees in Network Optimization. Springer Science and Business Media., 2010.

[۵] صنیعی آباده، محمد. الگوریتم های تکاملی و محاسبات زیستی. انتشارات نیاز دانش، ۱۳۹۴.

[۶] شاهی، سمیرا. حل مسائل مکان‌یابی با استفاده از الگوریتم‌های فراابتکاری. پایان‌نامه دوره کارشناسی ارشد در رشته ریاضی کاربردی گرایش تحقیق در عملیات، سبزوار، دانشگاه حکیم سبزواری، ۱۳۹۲.

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

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

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

[10] Tsai, CY and kao, IW. Particle Swarm Optimization with Selective Particle Regeneration for Data Clustering. Expert Systems With Application., 38:6565-6576, 2011.

[11] Kennedy, J and Eberhart, RC. A Discrete Binary Version of the Particle Swarm Algorithm. in: Proceedings of the International Conference on Systems, Man,and Cybernetics, IEEE Service Center., 4104-4108, 1997.

[12] Rajabioun, R. Cuckoo Optimization Algorithm. Applied Soft Computing Journal., 11:5508-5518, 2011.

[13] Yang, X SH and Deb, S. Engineering Optimization by Cuckoo Search. Int J Mathematical Modelling and Numerical Optimization., 4:330-343, 2010.

[14] Yang, X SH. Nature-Insoired Metaheuristic Algorithms. Luniver Press., 2010.

[15] Wang, G. A Hybrid Meta-heuristic DE/CS Algorithm for UCAV Three-Dimension Path Planning. The Scientific World Journal., 2012.

[۱۶] محمودی، شادی. گسسته‌سازی الگوریتم بهینه‌سازی فاخته–مطالعه موردی مسئله رنگ‌آمیزی گراف. پایان‌نامه دوره کارشناسی ارشد در رشته مهندسی نرم‌افزار کامپیوتر گرایش هوش مصنوعی، تبریز، موسسه آموزش عالی نبی‌اکرم (ص)، ۱۳۹۱.

[۱۷] ابوذری خویی، نازیلا، و حاتملو، عبدالرضا. کاربرد الگوریتم بهینه‌سازی فاخته در حل مسائل مختلف بهینه‌سازی. همایش ملی مهندسی رایانه و مدیریت فناوری اطلاعات، ۱۳۹۳.

[18] Kahramanli, H. A Modified Cuckoo Optimization Algorithm for Engineering Optimization. International Jurnal of Future Computer and Communication., 199-201, 2012.

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

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

پیوست آ

کد الگوریتم بهینه سازی فاخته

در این پیوست، قسمت اصلی کد الگوریتم بهینه سازی فاخته که با نرم‌افزار MATLAB نوشته شده است، آورده می‌شود.

```
%% initialize population:

cuckooPop = cell(numCuckooS,1);
% initialize egg laying center for each cuckoo
for cuckooNumber = 1:numCuckooS
    cuckooPop{cuckooNumber}.center = ( varHi-varLo )*rand(1,npar) + varLo;
end


iteration = 0;
goalPoint = (varHi - varLo)*rand(1,npar) + varLo; % a random goalpoint to start COA
maxProfit = -feval(costFunction,goalPoint);
globalBestCuckoo = goalPoint;
globalMaxProfit = maxProfit;
profitVector = [];
while ( (iteration <= maxIter) && (-maxProfit > accuracy) )

    iteration = iteration + 1

    % initialize number of eggs for each cuckoo
    for cuckooNumber = 1:numCuckooS
        cuckooPop{cuckooNumber}.numberOfEggs = floor( (
            maxNumberOfEggs - minNumberOfEggs) * rand +
            minNumberOfEggs );
    end

    % get total number of available eggs between all cuckooS
    summ = 0;
```

```
27      for cuckooNumber = 1:numCuckooS
28          summ = summ + cuckooPop{cuckooNumber}.numberOfEggs;
29      end
30
31      % calculate egg laying radius for each Cuckoo, considering
           problem
32      % limitations and ratio of each cuckoo's eggs
33      for cuckooNumber = 1:numCuckooS
34          cuckooPop{cuckooNumber}.eggLayingRadius = cuckooPop{
               cuckooNumber}.numberOfEggs/summ * ( radiusCoeff * (
               varHi-varLo) );
35      end
36
37      % To lay eggs, we produced some radius values less than egg
           laying
38      % radius of each cuckoo
39      for cuckooNumber = 1:numCuckooS
40          cuckooPop{cuckooNumber}.eggLayingRadiuses = cuckooPop{
               cuckooNumber}.eggLayingRadius * rand(cuckooPop{
               cuckooNumber}.numberOfEggs,1);
41      end
42
43      for cuckooNumber = 1:numCuckooS
44          params = cuckooPop{cuckooNumber}.center;        % get
               center values
45          tmpRadiuses = cuckooPop{cuckooNumber}.eggLayingRadiuses;
46          numRadiuses = numel(tmpRadiuses);
47          % divide a (hyper)circle to 'numRadiuses' segments
48          % This is to search all over the current habitat
49          angles = linspace(0,2*pi,numRadiuses);    % in Radians
50          newParams = [];
51          for cnt = 1:numRadiuses
52              addingValue = zeros(1,npar);
53              for iii = 1:npar
54                  randNum = floor(2*rand)+1;
55                  addingValue(iii) = (-1)^randNum * tmpRadiuses(
                       cnt)*cos(angles(cnt)) + tmpRadiuses(cnt)*sin(
                       angles(cnt));
56              end
57              newParams = [newParams; params +  addingValue ];
58          end
59
60
61          % check for variable limits
62          newParams(find(newParams>varHi)) = varHi;
63          newParams(find(newParams<varLo)) = varLo;
64
65          cuckooPop{cuckooNumber}.newPosition4Egg = newParams;
66      end
```

```
67
68      % Now egg laying is done
69
70
71      % Now that egg positions are found, they are laid, and so
            its time to
72      % remove the eggs on the same positions (because each egg
            only can go to one nest)
73      for cuckooNumber = 1:numCuckooS
74          tmpPopulation = cuckooPop{cuckooNumber}.newPosition4Egg;
75          tmpPopulation = floor(tmpPopulation * 1e20)/1e20;
76          ii = 2;
77          cntt = 1;
78          while ii <= size(tmpPopulation,1) || cntt <= size(
                tmpPopulation,1)
79              if all((tmpPopulation(cntt,:) == tmpPopulation(ii,:)
                    ))
80                  tmpPopulation(ii,:) = [];
81              end
82              ii = ii + 1;
83              if ii > size(tmpPopulation,1) && cntt <= size(
                    tmpPopulation,1)
84                  cntt = cntt + 1;
85                  ii = cntt + 1;
86                  if ii > size(tmpPopulation,1)
87                      break
88                  end
89              end
90          end
91          cuckooPop{cuckooNumber}.newPosition4Egg = tmpPopulation;
92      end
93
94
95      % Now we evalute egg positions
96      for cuckooNumber = 1:numCuckooS
97          cuckooPop{cuckooNumber}.profitValues = -feval(
                costFunction,[cuckooPop{cuckooNumber}.center ;
                cuckooPop{cuckooNumber}.newPosition4Egg]);
98      end
99
100     % Now we check to see if cuckoo population is more than
            maxNumOfCuckoos
101     % this case we should keep 1st maxNumOfCuckoos cuckoos and
            kill the others
102     allPositions = [];
103     whichCuckooPopTheEggBelongs = [];
104     tmpProfits = [];
105     if numCuckooS > maxNumOfCuckoos
106         for cuckooNumber = 1:numCuckooS
```

```
107             tmpProfits = [tmpProfits; cuckooPop{cuckooNumber}.
                    profitValues];
108             allPositions = [allPositions; [cuckooPop{
                    cuckooNumber}.center; cuckooPop{cuckooNumber}.
                    newPosition4Egg(:,1:npar)]];
109             whichCuckooPopTheEggBelongs = [
                    whichCuckooPopTheEggBelongs; cuckooNumber*ones(
                    size(cuckooPop{cuckooNumber}.newPosition4Egg(:,1:
                    npar),1),1) ];
110         end
111         % now we sort cuckoo profits
112         [sortedProfits, sortingIndex] = sort(tmpProfits,'descend
                ');
113         % Get best cuckoo to be copied to next generation
114         bestCuckooCenter = allPositions(sortingIndex(1),1:npar);
115
116         sortedProfits = sortedProfits(1:maxNumOfCuckoos);
117         allPositions = allPositions(sortingIndex(1:
                maxNumOfCuckoos),:);
118         clear cuckooPop
119         for ii = 1:maxNumOfCuckoos
120             cuckooPop{ii}.newPosition4Egg = allPositions(ii,:);
121             cuckooPop{ii}.center = allPositions(ii,:);
122             cuckooPop{ii}.profitValues = sortedProfits(ii);
123         end
124         numCuckooS = maxNumOfCuckoos;
125     else
126         for cuckooNumber = 1:numCuckooS
127             tmpProfits = [tmpProfits; cuckooPop{cuckooNumber}.
                    profitValues];
128             allPositions = [allPositions; [cuckooPop{
                    cuckooNumber}.center; cuckooPop{cuckooNumber}.
                    newPosition4Egg(:,1:npar)] ];
129             whichCuckooPopTheEggBelongs = [
                    whichCuckooPopTheEggBelongs; cuckooNumber*ones(
                    size(cuckooPop{cuckooNumber}.newPosition4Egg(:,1:
                    npar),1),1) ];
130         end
131         [sortedProfits, sortingIndex] = sort(tmpProfits,'descend
                ');
132         % Get best cuckoo to be copied to next generation
133         bestCuckooCenter = allPositions(sortingIndex(1),1:npar);
134     end
135
136     maxProfit  = sortedProfits(1);
137     currentBestCuckoo = bestCuckooCenter;
138     currentMaxProfit = -feval(costFunction,currentBestCuckoo);
139     if currentMaxProfit > globalMaxProfit
140         globalBestCuckoo = currentBestCuckoo;
```

```
141         globalMaxProfit = currentMaxProfit;
142     end
143
144     % ======== now we have some eggs that are safe and will grow
            up ==========
145     %========= mating: =============
146
147     % first we put all egg positions under each other
148     allPositions = [];
149     whichCuckooPopTheEggBelongs = [];
150     for cuckooNumber = 1:numCuckooS
151         allPositions = [allPositions; cuckooPop{cuckooNumber}.
                newPosition4Egg(:,1:npar)];
152         whichCuckooPopTheEggBelongs = [
                whichCuckooPopTheEggBelongs; cuckooNumber*ones(size(
                cuckooPop{cuckooNumber}.newPosition4Egg(:,1:npar),1)
                ,1) ];
153     end
154
155     if sum(var(allPositions)) < cuckooPopVariance
156         break
157     else
158         [clusterNumbers, clusterCenters] = kmeans(allPositions,
                knnClusterNum,'emptyaction','singleton');
159     end
160     % make newly made clusters
161     cluster = cell(knnClusterNum,1);
162     for ii = 1:knnClusterNum
163         cluster{ii}.positions = [];
164         cluster{ii}.profits = [];
165     end
166
167     pointer = zeros(numel(cuckooPop),1);
168     for tmpCounter = 1:numel(cuckooPop)
169         nEggs = size(cuckooPop{tmpCounter}.newPosition4Egg,1);
170         pointer(tmpCounter) = nEggs;
171     end
172     for cnt = 1:length(clusterNumbers)
173         cluster{clusterNumbers(cnt)}.positions = [cluster{
                clusterNumbers(cnt)}.positions; cuckooPop{
                whichCuckooPopTheEggBelongs(cnt)}.newPosition4Egg(end
                -pointer(whichCuckooPopTheEggBelongs(cnt))+1,1:npar)
                ];
174         cluster{clusterNumbers(cnt)}.profits   = [cluster{
                clusterNumbers(cnt)}.profits; cuckooPop{
                whichCuckooPopTheEggBelongs(cnt)}.profitValues(end-
                pointer(whichCuckooPopTheEggBelongs(cnt))+1)];
175         pointer(whichCuckooPopTheEggBelongs(cnt)) = pointer(
                whichCuckooPopTheEggBelongs(cnt)) - 1;
```

```
176     end
177
178     % Determine the best cluster
179     f_mean = zeros(knnClusterNum,1);
180     for cnt = 1:knnClusterNum
181         f_mean(cnt) = mean(cluster{cnt}.profits);
182     end
183
184     [sorted_f_mean, sortingIndex_f_mean] = sort(f_mean,'descend'
            );
185     maxFmean = sorted_f_mean(1);   indexOfBestCluster =
            sortingIndex_f_mean(1);
186
187     % now that we know group with number 'indexOfBestCluster' is
            the best we
188     % should select their best point az Goal Point of other
            groups
189     [maxProfitInBestCluster, indexOfBestEggPosition] = max(
            cluster{indexOfBestCluster}.profits);
190     goalPoint  = cluster{indexOfBestCluster}.positions(
            indexOfBestEggPosition,1:npar);
191
192     % now all other mature Cuckoos must go toward this goal
            point for laying
193     % their eggs
194     numNewCuckooS = 0;
195     for cntClstr = 1:size(cluster,1)
196         tmpCluster = cluster{cntClstr};
197         tmpPositions = tmpCluster.positions;
198         for cntPosition = 1:size(tmpPositions,1)
199             tmpPositions(cntPosition,:) = tmpPositions(
                    cntPosition,:) + ...
200                                           motionCoeff * rand(1,
                                              npar) .*  (
                                              goalPoint  -
                                              tmpPositions(
                                              cntPosition,:));
201         end
202         % check if variables are in range
203         tmpPositions(find( tmpPositions>varHi )) = varHi;
204         tmpPositions(find( tmpPositions<varLo )) = varLo;
205
206         % update cluster positions
207         cluster{cntClstr}.positions = tmpPositions;
208         cluster{cntClstr}.center = mean(tmpPositions);
209         % update number of cuckoos: numCuckooS
210         numNewCuckooS = numNewCuckooS + size(cluster{cntClstr}.
                positions,1);
211     end
```

```
212
213     numCuckooS = numNewCuckooS;
214     % update cuckooPop
215     clear cuckooPop
216     cuckooPop = cell(numCuckooS,1);
217     cntNumCuckooS = 1;
218     for cnt = 1:size(cluster,1)
219         tmpCluster = cluster{cnt};
220         tmpPositions = tmpCluster.positions;
221         for cntPosition = 1:size(tmpPositions,1)
222             cuckooPop{cntNumCuckooS}.center = cluster{cnt}.
                    positions(cntPosition,1:npar);
223             cntNumCuckooS = cntNumCuckooS + 1;
224         end
225     end
226     % Copy the Best cuckoo and its randomized form of this
            population to go
227     % to the next generation
228     currentBestCuckoo = bestCuckooCenter;
229     currentMaxProfit = -feval(costFunction,currentBestCuckoo);
230     if currentMaxProfit > globalMaxProfit
231         globalBestCuckoo = currentBestCuckoo;
232         globalMaxProfit = currentMaxProfit;
233     end
234     cuckooPop{end}.center = globalBestCuckoo; % This is because
            the best cuckoo will live longer and won't die right
            after egg laying
235     cuckooPop{end}.profitValues = -feval(costFunction,cuckooPop{
            end}.center);
236
237     tmp = rand(1,npar).*globalBestCuckoo;
238     tmp(find( tmp>varHi )) = varHi;
239     tmp(find( tmp<varLo )) = varLo;
240     cuckooPop{end-1}.center = tmp;
241     cuckooPop{end-1}.profitValues = -feval(costFunction,
            cuckooPop{end-1}.center);
242
243 end      % end of while
244
245 %% Now Algorithm has stopped
246
247 disp('Best Params = ')
248 disp(globalBestCuckoo')
249
250 fprintf('\n')
251
252 disp('Cost = ')
253 disp(-globalMaxProfit)
```

واژه‌نامه فارسی به انگلیسی

ابتکاری . . . . . hiuristic
افتادن در دور . . . . . Cycling
الگوریتم بهینه سازی فاخته . . . . . cuckoo optimization algorithm(COA)
الگوریتم ژنتیک . . . . . Genetic Algorithm(GA)
الگوریتم‌های تقریبی . . . . . approximation algorithms
الگوریتم تکامل تفاضلی . . . . . differential evolution algorithm(DE)
الگوریتم تکاملی . . . . . evolutionary algorithm
الگوریتم جستجوی فاخته . . . . . cuckoo search algorithm(CS)
الگوریتم‌های دقیق . . . . . exact algorithms
الگوریتم کرم شب‌تاب . . . . . firefly algorithm
انبار . . . . . warehouse
انتخاب تصادفی . . . . . Random Selection
انتخاب چرخ رولت . . . . . Roulette Wheel Selection
اندازه جمعیت . . . . . Population Size
بهینه‌سازی . . . . . optimization
بهینه‌سازی ازدحام ذرات . . . . . Particle Swarm Optimization(PSO)
بهینه‌سازی ترکیبیاتی . . . . . combinational optimization
بهینه سراسری . . . . . global optimum
بهینه‌سازی کلونی مورچگان . . . . . Ant Colonies Optimization(ACO)
بهینه محلی . . . . . Local Optimal
تابع برازندگی . . . . . Fitness Function
تابع هزینه . . . . . cost function
تخم‌گذاری . . . . . egg laying

ترکیب . . . . . Crossover

ترکیب K نقطه‌ای . . . . . K-Point Crossover

ترکیب یک نقطه‌ای . . . . . One Point Crossover

تسهیلات . . . . . facility

تکرار . . . . . iteration

جستجوی محلی . . . . . exploitation search

جمعیت . . . . . Population

جواب . . . . . solution

جواب بهینه . . . . . optimum solution

جهش . . . . . Mutation

حافظه . . . . . Memory

خوشه‌بندی . . . . . clustering

ذره (پرنده) . . . . . Particle

رتبه‌بندی . . . . . Ranking

رشته . . . . . String

رمزگشایی . . . . . Decoding

زمان‌بندی . . . . . Scheduling

زمان چندجمله‌ای . . . . . Polynomial-time

شعاع . . . . . radius

شعاع تخم‌گذاری . . . . . Egg Laying Radius(ELR)

مکانیابی . . . . . location

ظرفیت محدود . . . . . capacitated

ظرفیت نامحدود . . . . . uncapacitated

غیر چندجمله‌ای-سخت . . . . . NP-hard

فراابتکاری . . . . . Meta-heuristic

فضای جواب . . . . . solution space

کارخانه . . . . . plant

کدگذاری . . . . . Encoding

کروموزوم . . . . . Chromosome

محل سکونت . . . . . habitat

مرکز . . . . . center

مشتری . . . . . client

موقعیت . . . . . position

مهاجرت . . . . . immigrate

میانه . . . . . median

نامعین . . . . . Nondeterministic

همگرایی . . . . . convergence

واژه‌نامه انگلیسی به فارسی

بهینه‌سازی کلونی مورچگان . . . . . Ant Colonies Optimization(ACO)

الگوریتم‌های تقریبی . . . . . approximation algorithms

ظرفیت محدود . . . . . capacitated

مرکز . . . . . center

کروموزوم . . . . . Chromosome

مشتری . . . . . client

خوشه‌بندی . . . . . clustering

بهینه‌سازی ترکیبیاتی . . . . . combinational optimization

همگرایی . . . . . convergence

تابع هزینه . . . . . cost function

ترکیب . . . . . Crossover

الگوریتم بهینه سازی فاخته . . . . . cuckoo optimization algorithm(COA)

الگوریتم جستجوی فاخته . . . . . cuckoo search algorithm(CS)

افتادن در دور . . . . . Cycling

رمزگشایی . . . . . Decoding

الگوریتم تکامل تفاضلی . . . . . differential evolution algorithm(DE)

تخم‌گذاری . . . . . egg laying

شعاع تخم‌گذاری . . . . . Egg Laying Radius(ELR)

کدگذاری . . . . . Encoding

الگوریتم تکاملی . . . . . evolutionary algorithm

الگوریتم‌های دقیق . . . . . exact algorithms

جستجوی محلی . . . . . exploitation search

تسهیلات . . . . . facility

الگوریتم کرم شب‌تاب . . . . . firefly algorithm

تابع برازندگی . . . . . Fitness Function

الگوریتم ژنتیک . . . . . Genetic Algorithm(GA)

بهینه سراسری . . . . . global optimum

محل سکونت . . . . . habitat

ابتکاری . . . . . hiuristic

مهاجرت . . . . . immigrate

تکرار . . . . . iteration

مسئله کوله‌پشتی . . . . . knapsack problem

ترکیب K نقطه‌ای . . . . . K-Point Crossover

پرواز لوی . . . . . levy flight

بهینه محلی . . . . . Local Optimal

جستجوی محلی . . . . . Local Search

مکانیابی . . . . . location

میانه . . . . . median

فرا ابتکاری . . . . . meta heuristic

حافظه . . . . . Memory

جهش . . . . . Mutation

نامعین . . . . . Nondeterministic

غیر چندجمله‌ای-سخت . . . . . NP-hard

ترکیب یک نقطه‌ای . . . . . One Point Crossover

جواب بهینه . . . . . optimal solution

بهینه‌سازی . . . . . optimization

ذره (پرنده) . . . . . Particle

بهینه‌سازی ازدحام ذرات . . . . . Particle Swarm Optimization(PSO)

کارخانه . . . . . plant

زمان چندجمله‌ای . . . . . Polynomial-time

جمعیت . . . . . Population

اندازه جمعیت . . . . . Population Size

نوع جمعیت . . . . . Population Type

موقعیت . . . . . position

شعاع . . . . . radius

رتبه‌بندی . . . . . Ranking

انتخاب تصادفی . . . . . Random Selection

انتخاب چرخ رولت . . . . . Roulette Wheel Selection

جواب . . . . . solution

فضای جواب . . . . . solution space

ظرفیت نامحدود . . . . . uncapacitated

انبار . . . . . warehouse

نمایه

حرکت

پرندگان، ۲۲

Hakim Sabzevari University

An Outline of MSc. Thesis

Surname:Hokmabadi	Name:Somayye	Student No.:9413133106

Supervisors: Dr. MohammadAli Partanian and Dr. Amin Rafiei

Advisor: Dr. Mahmood Amintoosi

Faculty of Mathematics and Computer Science

Program: Applied Mathematics Field:Operational Research

Title of thesis: Solving Uncapacitated Facility Location Problems Using Cuckoo Optimization Algorithm

Keywords:Location,Facility,Uncapacitated,Metaheuristic,Cuckoo Optimization Algorithm.

Abstract: The uncapacitated facility location problem (UFLP) involves locating an undetermined number of facilities to minimize the sum of the fixed setup costs and the variable costs of serving the market demand from these facilities.

UFLP is an NP-Hard problem and metaheuristic algorithms with efficiency and robustness, solve the UFLP in reasonable cpu time.

In this dissertation, we analyze one of metaheuristic methods called ”Cuckoo Optimization Algorithm”, then use it for solving the UFLP, and compare its efficiency with two of metaheuristic methods that using for this problem, Genetic Algorithm and Particle Swarm Optimization Algorithm.

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 Uncapacitated Facility Location Problems Using Cuckoo Optimization Algorithm

Supervisors:
Dr. MohammadAli Partanian and Dr. Amin Rafiei

Advisor:
Dr. Mahmood Amintoosi

By:
Somayye Hokmabadi

September 2017