دانشگاه پیام نور

کاربرد الگوریتم زنبورعسل در بهینه‌سازی مسائل ریاضی

مهندسی فناوری اطلاعات

ارائه شده به:

گروه علمی فناوری اطلاعات و ارتباطات

دانشکده‌ی فنی و مهندسی

دانشگاه پیام نور مرکز نجف آباد

توسط:

فریبرز شمس کیا

شهریور ۱۳۹۱


بسم الله الرحمن الرحیم

تقدیم‌نامه

به پاس تعبیر عظیم و انسانی‌شان از کلمه‌ی ایثار و از خودگذشتگی

به پاس عاطفه‌ی سرشار و گرمای امیدبخش وجودشان که در این سردترین روزگاران بهترین پشتیبان است

به پاس قلب‌های بزرگشان که فریادرس است و سرگردانی و ترس در پناهشان به شجاعت می‌گراید

و به پاس محبت‌های بی دریغشان که هرگز فروکش نمی‌کند

این مجموعه را به پدر و مادر عزیزم تقدیم می‌کنم

سپاس نامه

صدها فرشته بوسه بر آن دست می زند

کز کـــــــار خلق یک گره بسته وا کند

«سپاس، خداوند مهربانی را که به انسان توانایی و دانایی بخشید تا به بندگانش شفقت ورزد، مهربانی کند و در حل مشکلاتشان یاری‌شان نماید. »

به مصداق «من لم یشکر المخلوق لم یشکر الخالق» بسی شایسته است از استاد فرهیخته و فرزانه، سرکار خانم الهام حری

که سرزمین دل را روشنی بخشیدند و گلشنِ سرای علم و دانش را با راهنمایی های کارساز و سازنده بارور ساختند، تقدیر و تشکر نمایم.

و سپاس از خانواده‌ی عزیزم،

مهربان فرشتگانی که لحظات نابِ باورِ شدن، غرورِ دانستن، جسارتِ خواستن، عظمتِ رسیدن و تجربه‌های زیبای زندگیم، مدیون حضور سبز آن‌هاست.

همچنین سپاس از دوستان مهربانم که مرا در راه رسیدن به اهدافم یاری نمودند.

چکیده

انسان همیشه برای الهام گرفتن به جهان زنده‌ی پیرامون خود نگریسته است. یکی ازبهترین طرح‌های شناخته شده، طرح پرواز انسان است که ابتدا لئوناردو داوینچی(۱۵۱۹-۱۴۵۲) طرحی از یک ماشین پرنده را براساس ساختمان بدن خفاش رسم نمود. چهارصد سال بعد کلمان آدر ماشین پرنده ای ساخت که دارای موتور بود و به جای بال از ملخ استفاده می‌کرد.

در دهه های اخیر، روش های تکاملی و فراکاوشی به عنوان یک ابزار جستجو و بهینه سازی در حوزه‌های مختلفی مانند علوم تجاری و مهندسی مورد استفاده قرار گرفته است. وسعت دامنه‌ی کاربرد، سهولت استفاده و قابلیت دست یابی به جواب نزدیک و بهینه ی مطلق از جمله دلایل موفقیت این روش‌ها می‌باشد.

هوش دسته جمعی، زیر شاخه ای از هوش مصنوعی است که بر پایه ی رفتار جمعی سیستم های غیر متمرکز و خود سازمان‌ده بنا شده است. نمونه‌ای از هوش جمعی، کلونی زنبور عسل است.

یکی از کاربردهای این الگوریتم ، مسائل بهینه سازی چندتایی است برای همین برخی به آن الگوریتم بهینه‌سازی زنبورعسل می‌گویند. دراین مقاله، الگوریتم **کلونی زنبورعسل** مورد استفاده قرار می‌گیرد و نتایج تولید شده توسط الگوریتم مقایسه می‌شوند.

موضوع کلونی زنبور عسل خود به دو بخش جستجوی غذا و فرآیند جفت گیری زنبورها تقسیم می‌شود.

کلمات کلیدی: الگوریتم‌های تکاملی، هوش جمعی، کلونی زنبور عسل و بهینه‌سازی

فهرست مطالب

فهرست شکل‌ها و جدول‌ها

عنوان **صفحه**

مقدمه:

امروزه یکی از مهم‌ترین زمینه‌های تحقیق و پژوهش، توسعه بر مبنای اصول تکامل طبیعی می‌باشد.

حشرات اجتماعی (زنبورعسل ، زنبورمعمولی، مورچه‌ها و موریانه‌ها) برای میلیون‌ها سال بر روی کره ی زمین زندگی کرده‌اند. آشیانه‌های مختلف و بسیاری سازه‌های پیچیده‌تر ساخته‌اند و آذوقه‌شان را سازماندهی کرده‌اند. کلونی حشرات اجتماعی بسیار انعطاف پذیر محسوب می شود و به خوبی قابلیت همساز شدن با محیط جدید را دارند. این انعطاف پذیری این امکان را به کلونی می دهد تا بتواند حتی با مواجه شدن با شرایط سخت و مشکلات به زندگی خود ادامه دهد.

پویاگرایی جمعیت حشرات نتیجه ای از عملکردها و تعاملات بین حشرات با یکدیگر و با محیط اطراف است. تعاملات بین حشرات بر اساس یک سری عوامل فیزیکی و شیمیایی امکان پذیر شده است. محصول نهایی این تعاملات و عملکردها رفتار اجتماعی این گونه حشرات محسوب میشود.

مثالی برای چنین رفتارهایی ترشح فرومون(هورمون جنسی) در مورچه‌هاست که موجب راه گذاری برای مورچه‌های دیگر خواهد شد. مثال دیگری برای این حالت، رقص زنبورهای‌عسل در هنگام جمع‌آوری محصول است. این سیستم‌های ارتباطی بین حشرات مختلف موجب به وجود آمدن مقوله ای به نام "هوش اشتراکی" می‌شود. به این معنی که حشرات فوق به هنگام قرار گرفتن در کنار یکدیگر دارای فاکتوری هوشمند می‌شوند که در غیاب یکدیگر قادر به انجام چنین کاری نیستند.

سیستم سازمانی زنبورها بر اساس یک سری قوائد ساده‌ی خارجی حشرات بنا شده است. با اینکه نژادهای بسیاری از حشرات مختلف بر روی کره‌ی زمین موجود هستند و همین باعث تفاوت‌هایی در الگوی رفتاری آن‌ها می‌شود ولی با این حال این سری حشرات اجتماعی را می‌توان دارای قابلیت حل مسائل پیچیده دانست. همگی زنبورها در یک فرایند "تصمیم‌گیری" شرکت می کنند. هر زنبوری قابلیت درک و دریافت اطلاعات زنبورهای دیگر را براساس کیفیت دارد. به کمک این روش، زنبورها این قابلیت را دارند که با استفاده از اطلاعات دیگران، راه‌های بهتر حل مساله را پیدا کنند.

هم اکنون کار روی توسعه سیستم های هوشمند با الهام از طبیعت از زمینه های خیلی پرطرفدار هوش مصنوعی است. الگوریتم های ژنتیک که با استفاده از ایده‌ی تکاملی داروینی و انتخاب طبیعی مطرح شده، روش بسیار خوبی برای یافتن مسائل بهینه‌سازیست. ایده‌ی تکاملی داروینی بیانگر این مطلب است که هر نسل نسبت به نسل قبل خود دارای تکامل است و آنچه در طبیعت رخ می دهد حاصل میلیون‌ها سال تکامل نسل به نسل موجوداتی مثل مورچه است.

هوش جمعی

هوش دسته جمعی، زیر شاخه‌ای از هوش مصنوعی است که بر پایه ی رفتار جمعی سیستم های غیر متمرکز و خود سازمان‌ده بنا شده است. این اصطلاح برای اولین بار در سال ۱۹۸۹ در زمینه‌ی سیستم‌های روباتی سلولی به کار برده شد.

سیستم‌های Swarm Intelligence(SI) به طور نمونه از یک گروه از عوامل ساده ساخته شده که به طور محلی با یکدیگر و نیز با محیط پیرامونشان بر هم کنش دارند. بنابراین در آن ها ساختار کنترلی متمرکزی وجود ندارد که به هر عامل منفرد دستور دهد که چگونه رفتار کند. نمونه های طبیعی SI شامل کلونی مورچه‌ها، دسته‌های پرندگان، گله‌های حیوانات، رشد باکتریایی و دسته‌های ماهی‌ها می‌باشد.

فرض کنید شما و گروهی از دوستانتان به دنبال گنج می‌گردید. هر یک از اعضای گروه یک فلزیاب و یک بی‌سیم دارد که می‌تواند مکان و وضعیت کار خود را به همسایگان نزدیک خود اطلاع بدهد. بنابراین شما می‌دانید آیا همسایگانتان از شما به گنج نزدیکترند یا نه؟ پس اگر همسایه‌ای به گنج نزدیکتر بود شما می‌توانید به طرف او حرکت کنید. با چنین کاری شانس شما برای رسیدن به گنج بیشتر می‌شود و همچنین گنج زودتر از زمانی که شما تنها باشید، پیدا می‌شود.

این یک مثال ساده از رفتار جمعی است که افراد برای رسیدن به یک هدف نهایی همکاری می‌کنند. این روش مؤثرتر از زمانی است که افراد جداگانه عمل کنند. *Swarm* را می‌توان به صورت مجموعه‌ای سازمان یافته از موجوداتی تعریف کرد که با یکدیگر همکاری می‌کنند. در کاربردهای محاسباتی *Swarm intelligence* از موجوداتی مانند مورچه‌ها، زنبورها، موریانه‌ها، دسته‌ی ماهیان و دسته‌ی پرندگان الگو برداری می‌شود. در این نوع اجتماعات هر یک از موجودات ساختار نسبتاً ساده‌ای دارند ولی رفتارجمعی آن‌ها بی‌نهایت پیچیده است. برای مثال در کلونی مورچه‌ها، هر یک از مورچه‌ها یک کار ساده‌ی مخصوص را انجام می‌دهد ولی به طور جمعی عمل و رفتار مورچه‌ها، ساختن بهینه‌ی لایه ی محافظت از ملکه و نوزادان، تمیز کردن لانه، یافتن بهترین منابع غذایی و بهینه‌سازی استراتژی حمله را تضمین می‌کند. رفتار کلی یک *Swarm* به صورت "غیر خطی" از آمیزش رفتارهای تک‌تک اجتماع ب ه دست می‌آید یا به عبارتی یک رابطه‌ی بسیار پیچیده بین رفتار جمعی و رفتار فردی یک اجتماع وجود دارد. رفتار جمعی فقط وابسته به رفتار فردی افراد اجتماع نیست بلکه به چگونگی تعامل میان افراد نیز وابسته است. تعامل بین افراد، تجربه‌ی افراد درباره‌ی محیط را افزایش می‌دهد و موجب پیشرفت اجتماع می‌شود. ساختار اجتماعی *Swarm* بین افراد مجموعه کانال های ارتباطی ایجاد می‌کند که طی آن افراد می‌توانند به تبادل تجربه‌های شخصی بپردازند، مدل‌سازی محاسباتی *Swarm*ها، کاربردهای موفق و بسیاری را درپی داشته است مانند:

scheduling, structural ,optimal roots optimization, Finding Function analysis optimization, Image and data

کاربردهای زیادی از مطالعه‌ی *Swarm*های مختلف وجود دارد. از این دسته می‌توان به کلونی مورچه‌ها و دسته‌ی پرندگان و کندوی زنبورها اشاره نمود.

فصل اول

الگوریتم‌های تکاملی

۱-۱ هوش مصنوعی[1]

هنوز تعریف دقیقی که مورد قبول همه‌ی دانشمندان این علم باشد برای هوش مصنوعی ارائه نشده‌است و این امر به هیچ وجه ما یه‌ی تعجب نیست. چرا که مقوله‌ی مادر و اساسی‌تر از آن، یعنی خود هوش هم هنوز به طور همه‌جانبه و فراگیر تن به تعریف نداده است. درواقع، می‌توان نسل‌هایی از دانشمندان را سراغ گرفت که تمام دوران زندگی خود را صرف مطالعه و تلاش در راه یافتن جوابی به‌ این سؤال عمده نموده‌اند که: هوش مصنوعی چیست؟

اما اکثر تعریف‌هایی که در این زمینه ارایه شده‌اند بر پایه یکی از ۴ باور زیر قرار می‌گیرند:

1- Artificial intelligence

شکل 1-1: شماي گرافيكي
مغز انسان

- سیستم هایی که به طور منطقی فکر می‌کنند.
- سیستم هایی که به طور منطقی عمل می‌کنند.
- سیستم‌هایی که مانند انسان فکر می‌کنند.
- سیستم‌هایی که مانند انسان عمل می‌کنند.

شاید بتوان هوش مصنوعی را این گونه توصیف کرد: «هوش مصنوعی عبارت است از مطالعه ی اینکه چگونه کامپیوترها را می‌توان وادار به کارهایی کرد که در حال حاضر انسان‌ها آن ها را بهتر انجام می‌دهند» آزمون تورینگ آزمونی است که توسط آلن تورینگ در سال ۱۹۵۰ در نوشته‌ای به نام«محاسبات ماشینی و هوشمندی» مطرح شد. در این آزمون شرایطی فراهم می شود که شخصی با ماشین تعامل برقرار کند و پرسش‌های کافی برای بررسی هوشمندی او بپرسد. چنانچه در پایان آزمایش ،شخص نتواند تعیین کند که با انسان در تعامل بوده است یا با ماشین، تست تورینگ با موفقیت انجام شده است. تا کنون هیچ ماشینی از این آزمون با موفقیت بیرون نیامده است. کوشش این آزمون برای تشخیص درستی هوشمندی یک سیستم است که سعی در شبیه‌سازی انسان دارد.

۱-۲ الگوریتم چیست؟

آورده‌اند که واژه‌ی الگوریتم در حقیقت الخوارزمی نام دانشمند بزرگ ایرانی بوده که چند صدسال پیش در زمینه‌ی جبر و احتمال کارهای زیادی انجام داده است.

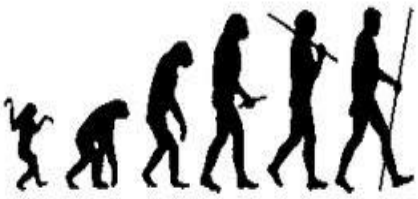
الگوریتم روش دقیق منطقی و مرحله به مرحله‌ی حل مساله است.

برای یادگیری الگوریتم‌ها نیازی به داشتن دانش ریاضی نیست اما تجربه ثابت کرده افرادی که دارای قدرت استدلال بالایی هستند در این زمینه موفق‌ترند.

برای هر الگوریتم وجود سه شرط لازم است

- هرالگوریتم باید معین وقطعی باشد.
- هرالگوریتم باید قابل اجرا باشد.
- هرالگوریتم باید پایان پذیر باشد.

۱-۳ الگوریتم‌های تکاملی



شکل 1-2: نمونه‌ای از تکامل

الگوریتم تکاملی[1]، الگوریتمی است که زیر مجموعه ای از محاسبات تکاملی است و در شاخه ی هوش مصنوعی قرار می گیرد. این الگوریتم‌ها از برخی مکانیزم‌های طبیعی الهام گرفته اند که عبارتند از: تولید، جهش، ترکیب و انتخاب. راه حل‌های منتخب برای مسائل بهینه‌سازی نقش اجزا را ایفا می‌کنند و تابع هزینه از میان این اجزا و افراد، تصمیم می گیرد که چه راه حل هایی باقی بمانند. پس می‌توان گفت: الگوریتم‌های تکاملی، رویه‌های جستجوی تصادفی با استفاده از سازوکار ژنتیکی و انتخاب طبیعی هستند. با مروری بر تاریخچه‌ی الگوریتم تکاملی مشخص می‌شود ایده‌ی اصلی تمام این نوع الگوریتم‌ها یکی است و به فرض جمعیتی از افراد برمی‌گردد که فشار محیطی آن‌ها را وادار به انتخاب طبیعی می‌کند و تابعی کیفی برای حداکثرکردن آن چیزی خواهند بود که به صورت تصادفی موجب پدید آمدن مجموعه‌ای از راه حل‌ها وعناصر می‌شود. به‌کارگیری تابع کیفی به عنوان معیاری برای هماهنگی در نظر گرفته می‌شود. بر اساس این هماهنگی، بعضی از کاندیداهای بهتر برای ایجاد نسل بعدی انتخاب می‌شوند و این فرایند تا رسیدن به جواب بهینه یا پایان یافتن زمان اجرا ادامه می‌یابد. الگوریتم‌های تکاملی کاربردهای مختلفی دارند.

معمولا از این الگوریتم‌ها در توابع بهینه‌سازی[2] استفاده می‌شود. زیرا الگوریتم‌های تکاملی مناسب بهینه‌سازی، تابع ترکیبی هستند واین نقش را با توجه به کروموزم‌هایشان در ارائه‌ی راه حل‌ها ایفا می‌کنند. هر فرد می‌تواند رشته‌ای ساده از صفر و یک و یا به پیچیدگی یک برنامه رایانه‌ای باشد. جمعیت اولیه به صورت تصادفی انتخاب می‌شود. سپس الگوریتم به منظور دست‌یابی به راه‌حل بهینه، به ارزیابی افراد می‌پردازد. معمولا هر راه حل منحصر به همان مسأله است و باید توسط کاربر حمایت و اداره شود. فرایند تکاملی باعث می‌شود تا جمعیت با محیط تطابق بهتری پیدا کند.

اجرای فرایند تکاملی در بسیاری از موارد اتفاقی است. هنگام انتخاب، افراد مناسب‌تر شانس بیشتری نسبت به سایرین دارند. اما معمولا حتی نامناسب‌ترین افراد هم شانسی برای والد وانتخاب شدن دارد. در ترکیب افراد، انتخاب اینکه کدام قسمت ترکیب شود نیز تصادفی است .

1- Evolutionary algorithm
2- Optimization functions

از بحث بالا می‌توان دید که الگوریتم‌های تکاملی به طور اساسی با دیگر روش‌های بهینه‌سازی و جستجوی مرسوم قدیمی تفاوت دارند. برخی از این تفاوت ها عبارتند از:

- الگوریتم‌های تکاملی تنها یک نقطه را جستجو نمی‌کنند بلکه جمعیتی از نقاط را به صورت موازی بررسی می‌نمایند.
- الگوریتم‌های تکاملی نیاز به اطلاعات ضمنی و دیگر دانش‌های مکمل ندارند وتنها تابع هدف وشایستگی‌های مربوطه در جهت‌های جستجو تاثیر گذارند.
- الگوریتم‌های تکاملی از قوانین در حال تغییر احتمالی بهره می‌برند ونه موارد مشخص و معین.
- استفاده از الگوریتم‌های تکاملی به طور کلی خیلی سرراست است زیرا هیچگونه محدودیتی برای تعریف تابع وجود ندارد.
- الگوریتم‌های تکاملی تعداد زیادی از پاسخ‌های قابل قبول را به دست می‌دهند و انتخاب پایانی بر عهده‌ی کاربر است.

۱-۳-۱ کاربردها

برخی از کاربردهای الگوریتم های تکاملی به شرح زیر است:

- بهینه‌سازی تابع
- بهینه‌سازی چندهدفه
- بهینه‌سازی ترکیبی
- مهندسی بهینه‌سازی ساختاری و طراحی
- مطلوب سازی محدودیت‌ها
- علم اقتصاد و دارایی
- بیولوژی
- استخراج داده و تحلیل داده
- مسائل ریاضی
- مهندسی برق و طراحی مدار
- شیمی و مهندسی شیمی
- برنامه‌ریزی
- رباتیک
- پردازش تصویر
- شبکه و ارتباطات
- پزشکی
- حداقل سازی منابع پشتیبانی و مراقبت‌های محیطی

- نظامی و دفاعی
- رفتارهای تکاملی

۱-۴ الگوریتم کلونی مورچه[1]

الگوریتم کلونی مورچه الهام گرفته شده از مطالعات و مشاهدات روی کلونی مورچه‌هاست. این مطالعات نشان داده که مورچه‌ها حشراتی اجتماعی هستند که در کلونی‌ها زندگی می‌کنند و رفتار آن ها بیشتر در جهت بقای کلونی است تا درجهت بقای یک جزء از آن. یکی از مهم ترین و جالب‌ترین رفتار مورچه‌ها، رفتار آن‌ها برای یافتن غذا است و به ویژه چگونگی پیدا کردن کوتاه ترین مسیر میان منابع غذایی و آشیانه. این نوع رفتار مورچه‌ها دارای نوعی هوشمندی توده ای است که اخیرا مورد توجه دانشمندان قرار گرفته است. ابتدا باید تفاوت هوشمندی توده‌ای(کلونی) و هوشمندی اجتماعی را روشن کنیم.

در هوشمندی اجتماعی عناصر میزانی از هوشمندی را دارا هستند. به عنوان مثال در فرآیند ساخت ساختمان توسط انسان، زمانی که به یک کارگر گفته می‌شود تا یک توده آجر را جابه‌جا کند، آنقدر هوشمند هست تا بداند برای اینکار باید از یک وسیله‌ی باربری(فرغون) استفاده کند نه مثلا بیل! نکته ی دیگر تفاوت سطح هوشمندی افراد این جامعه است. مثلا هوشمندی لازم برای فرد معمار با یک کارگر ساده متفاوت است.

در هوشمندی توده‌ای عناصر رفتاری تصادفی دارند و بین آن‌ها هیچ نوع ارتباط مستقیمی وجود ندارد و آن‌ها تنها به صورت غیرمستقیم و با استفاده از نشانه‌ها با یکدیگر در تماس هستند. مثالی در این مورد رفتار موریانه‌ها در لانه سازیست.

موریانه‌ها برای ساخت لانه سه فعالیت مشخص از خود بروز می‌دهند. در ابتدا صدها موریانه به صورت تصادفی به این طرف و آن طرف حرکت می‌کنند. هر موریانه به محض رسیدن به فضایی که کمی بالاتر از سطح زمین قرار دارد شروع به ترشح بزاق می‌کنند و خاک را به بزاق خود آغشته می‌کنند. به این ترتیب گلوله‌های کوچک خاکی با بزاق خود درست می‌کنند. علیرغم خصلت کاملا تصادفی این رفتار،نتیجه تا حدی منظم است. در پایان این مرحله ، در منطقه ای محدود، تپه‌های بسیار کوچک مینیاتوری از این گلوله‌های خاکی آغشته به بزاق شکل می‌گیرد. پس از این، همه ی تپه‌های مینیاتوری باعث می‌شود تا موریانه‌ها رفتار دیگری از خود بروز دهند. در واقع این تپه‌ها به صورت نوعی نشانه برای موریانه‌ها عمل می‌کنند. هر موریانه به محض رسیدن به این تپه‌ها با انرژی بسیار بالایی شروع به تولید گلوله‌های خاکی با بزاق خود می‌کند. این کار باعث تبدیل شدن تپه‌های مینیاتوری به نوعی ستون می‌شود. این رفتار ادامه می‌یابد تا زمانی که ارتفاع هر ستون به حد معینی برسد. در این صورت موریانه‌ها رفتار سومی از خود نشان می‌دهند. اگر در نزدیکی ستون فعلی ستون دیگری نباشد بلافاصله آن ستون را رها می‌کنند در غیر این صورت یعنی در حالتی که در

1- Ant Colony Optimization

نزدیکی این ستون تعداد قابل ملاحظه ای ستون دیگر باشد، موریانه‌ها شروع به وصل کردن ستون‌ها و ساختن لانه می‌کنند.

تفاوت‌های هوشمندی اجتماعی انسان با هوشمندی توده ای موریانه را در همین رفتار ساخت لانه می‌توان مشاهده کرد.

۱-۴-۱ بهینه‌سازی مسائل بروش کلونی مورچه(*ACO*) :

همان‌طور که می‌دانیم مساله‌ی یافتن کوتاه‌ترین مسیر، یک مساله بهینه سازیست که گاه حل آن بسیار دشوار است و گاه نیز بسیار زمان‌بر. به عنوان مثال، مساله‌ی فروشنده‌ی دوره گرد(*TSP*) را در نظر بگیرید. در این مساله، فروشنده‌ی دوره گرد باید از یک شهر شروع کرده، به شهرهای دیگر برود و سپس به شهر مبدا بازگردد به طوری که از هر شهر فقط یکبار عبور کند و کوتاه‌ترین مسیر را نیز طی کرده باشد. اگر تعداد این شهرها n باشد در حالت کلی این مساله از مرتبه $(n-1)!$ است که برای فقط ۲۱ شهر زمان واقعا زیادی می‌برد:

روز۱۰۱۳*۱/۷ = S ۱۰۱۶*۲/۴۳۳ = ms ۱۰*۱۰۱۸*۲/۴۳۳ = ۲۰!

با انجام یک الگوریتم برنامه سازی پویا برای این مساله، زمان از مرتبه نمایی به دست می‌آید که آن هم مناسب نیست. البته الگوریتم‌های دیگری نیز ارائه شده ولی هیچ کدام کارایی مناسبی ندارند. *ACO* الگوریتم کامل و مناسبی برای حل مساله‌ی *TSP* است.

۱ـ۴ـ۲ مورچه‌ها چگونه می‌توانند کوتاه‌ترین مسیر را پیدا کنند؟

شکل 1-3: سختی در حمل غذا و لزوم یافتن کوتاه‌ترین مسیر

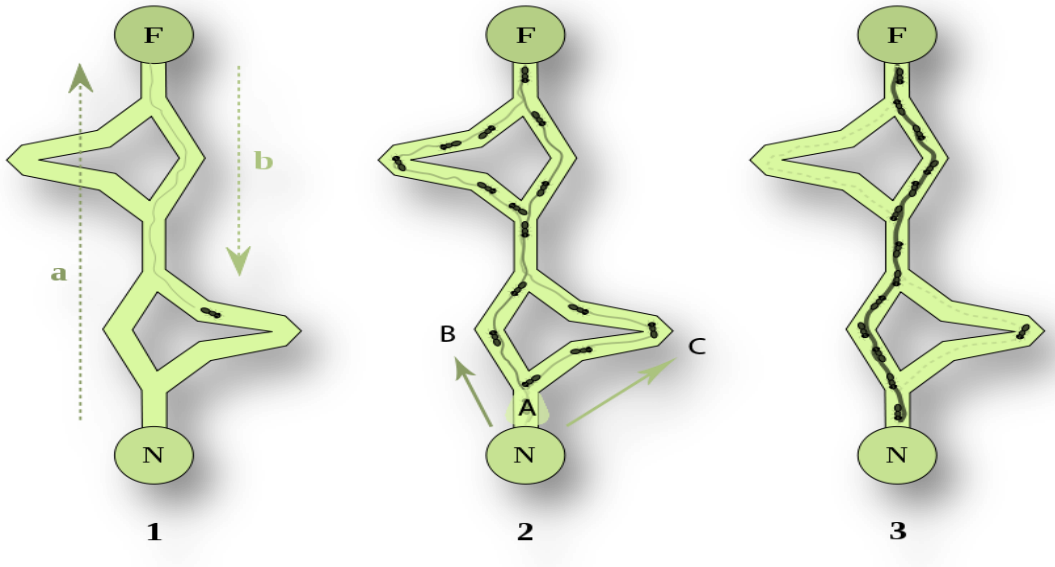
مورچه‌ها هنگام راه رفتن، از خود ردی از ماده شیمیایی فرومون [1] به جای می‌گذارند البته این ماده زود تبخیر می‌شود ولی در کوتاه مدت به عنوان رد مورچه بر سطح زمین باقی می‌ماند. یک رفتار پایه‌ای ساده در مورچه‌ها وجود دارد:

1- Pheromone

آن‌ها هنگام انتخاب بین دو مسیر ب ه صورت تصادفی(*Statistical*) مسیری را انتخاب می‌کنند که فرومون بیشتری داشته باشد یا به عبارت دیگر مورچه‌های بیشتری قبلا از آن عبور کرده باشند. حال دقت کنید که‌این کار چگونه منجر به پیدا کردن کوتاه‌ترین مسیر خواهد شد.

فرومون مسیر باعث پیشرفت و ترقی مورچه‌های کلونی در پیدا کردن کوتاه‌ترین راه می‌شود. بنابراین وقتی که تعداد مسیرهای زیادی بین منبع غذا به سمت کلونی در دسترس است، مورچه‌ها قادرند که کوتاه‌ترین مسیر از لانه به منبع غذا و بر عکس را پیدا کنند.

همان‌طور که در شکل ۱ می‌بینیم مورچه ای منبع غذایی را پیدا کرده و روی مسیر NF در حال بازگشت به لانه است زمانی که به لانه رسید به هم نوعان خود جهت منبع غذا را اطلاع داده و مورچه‌ها به ترتیب به سمت غذا حرکت می کنند.(شکل ۲) زمانی که مورچه‌ها به دوراهی A می رسند چون هیچ فرومونی نمی‌بینند به تصادف یکی از دو مسیر A یا B را انتخاب می‌کنند پس از گذشت مدت زمانی فرومون‌هایی که هر مورچه بر جای گذاشته شروع به تبخیر شدن می کند اما چون مسیر AC طولانی تر از AB است و حرکت مورچه‌ها را تصادفی فرض کردیم اثر فرومونی مسیر AC کمتر از AB می‌شود. لذا زمانی که مورچه‌های بعدی به دو راهی A می‌رسند مسیر AB که دارای اثر فرومونی بیشتری است را انتخاب می‌کنند.



شکل ۱-۴: فرومون و چگونگی یافتن کوتاه‌ترین مسیر

شکل 1-4: چگونگي يافتن كوتاه‌ترين مسير

برای تفهیم بیشتر مساله، به مثال زیر توجه کنید:

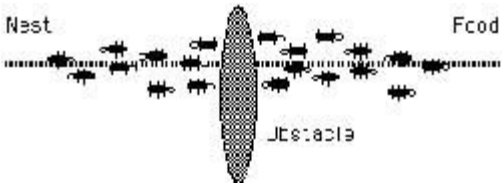
همان‌طور که در شکل ۲ می‌بینیم مورچه‌ها روی مسیر AB در حرکتند (در دو جهت مخالف) اگر در مسیر مورچه‌ها مانعی قرار دهیم(شکل ۲) مورچه‌ها دو راه برای انتخاب کردن دارند. اولین مورچه از A می‌آید و به C می‌رسد، در مسیر هیچ فرومونی نمی‌بیند بنابراین برای مسیر چپ و راست احتمال یکسان می‌دهد و

به طور تصادفی و احتمالاتی مسیر CED را انتخاب می‌کند. اولین مورچه ای که مورچه ی اول را دنبال می‌کند زودتر از مورچه‌ی اولی که از مسیر CFD رفته به مقصد می‌رسد. مورچه‌ها در حال برگشت و به مرور زمان یک اثر بیشتر فرومون را روی CED حس می‌کنند و آن را به طور احتمالی و تصادفی(نه حتما و قطعا) انتخاب می‌کنند. در نهایت مسیر CED بعنوان مسیر کوتاه تر برگزیده می‌شود. در حقیقت چون طول مسیر CED کوتاه‌تر است زمان رفت و برگشت از آن هم کمتر می‌شود و در نتیجه مورچه‌های بیشتری نسبت به مسیر دیگر آن را طی خواهند کرد چون فرومون بیشتری در آن وجود دارد.

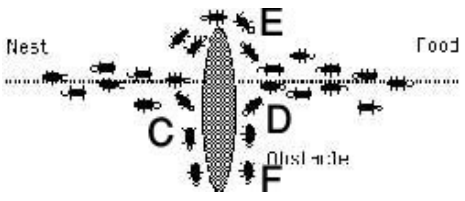
نکته‌ی بسیار با اهمیت این است که هر چند احتمال انتخاب مسیر پر فرومون تر توسط مورچه‌ها بیشتر است ولی این کماکان احتمال است و قطعیت نیست. یعنی اگر مسیر CED پرفرومون‌تر از CFD باشد به هیچ عنوان نمی‌شود نتیجه گرفت که همه مورچه‌ها از مسیر CED عبور خواهند کرد بلکه تنها می‌توان گفت که مثلا ۹۰٪ مورچه‌ها از مسیر کوتاه تر عبور خواهند کرد. اگر فرض کنیم که به جای این احتمال، قطعیت وجود می‌داشت، یعنی هر مورچه فقط و فقط مسیر پرفرومون تر را انتخاب می کرد آنگاه اساسا این روش ممکن نبود به جواب برسد. اگر تصادفا اولین مورچه مسیر CFD(مسیر دورتر) را انتخاب می‌کرد و ردی از فرومون بر جای می‌گذاشت آنگاه همه مورچه‌ها به دنبال او حرکت می‌کردند و هیچ وقت کوتاه ترین مسیر یافته نمی‌شد. بنابراین تصادف و احتمال نقش عمده ای در ACO بر عهده دارد.

نکته‌ی دیگر مساله تبخیر شدن فرومون بر جای گذاشته شده است. برفرض اگر مانع در مسیر AB برداشته شود و فرومون تبخیر نشود مورچه‌ها همان مسیر قبلی را طی خواهند کرد. ولی در حقیقت این طور نیست. تبخیر شدن فرومون و احتمال به مورچه‌ها امکان پیدا کردن مسیر کوتاه‌تر جدید را می‌ده‌د.

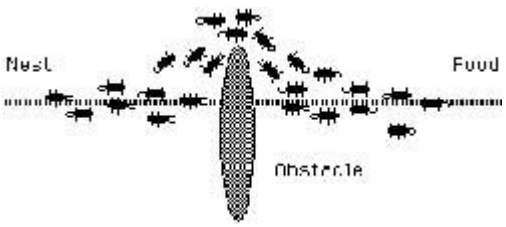
۱-۵



۲-۵



۳-۵



۴-۵

شکل 1-5: عدم تاثیر موانع در یافتن کوتاه‌ترین مسیر

۱-۴-۳ الگوریتم

۱. برای پیاده سازی کلونی مورچه از مورچه‌های مصنوعی به عنوان عناصر بهینه‌سازی استفاده می‌شود. البته‌این عناصر تفاوت اساسی با مورچه‌های واقعی دارند که عبارتند از:
۲. **حافظه:** برای مورچه‌های مصنوعی می‌توان یک حافظه در نظر گرفت که مسیر حرکت را در خود نگه می‌دارد.
۳. **موانع ساختگی:** تغیر دادن جزئیات مساله برای بررسی الگوریتم و رسیدن به جواب‌های متنوع.
۴. **حیات در محیط گسسته:** مورچه‌های واقعی نمی توانند جدا از کلونی به حیات خود ادامه دهند.

۱-۴-۳-۱ الگوریتم کلی حرکت:

الف) قانون کلی حرکت: همان‌طور که گفته شد مورچه‌ها بر اساس میزان فرومون ترشح شده، مسیر برتر را پیدا می‌کنند.

در پیاده سازی، ابتدا هر عنصر(مورچه‌ی مصنوعی) در یک حالت اولیه قرار داده می‌شود و میزان فرومون اولیه برای هر مورچه مشخص می‌شود. مورچه جهت حرکت به شهر S روی گره r قرار داده می‌شود. حرکت این مورچه بر اساس فرمول زیر انجام می‌شود :

(1_1)

$$S= \begin{cases} Arg\ max_{\ u\ \epsilon\ j_k(r)} \left\{ Ph(r,u).D(r,u)^{\beta} \right\} & ; \quad \text{اگر بهترین حالت انتخاب شود :} \\ S & ; \quad \text{اگر حالت بعدی انتخاب شود :} \end{cases}$$

که در رابطه‌ی قبل :

$j_k(r)$: مجموعه‌ی شهرهای باقی مانده در مسیر مورچه k که روی شهر r قرار دارد.

$Ph(r,u)$: میزان فرومون ترشح شده روی مسیر r تا u .

$D(r,u)$: معکوس مسافت r تا u . هر چه مسافت کمتر باشد D بیشتر می‌شود یعنی مسافت کوتاه‌تر است و احتمال انتخاب آن بالا می‌رود.

β: یک متغیر جهت تشخیص نسبت اهمیت فرومون در مقابل فرومون.

S: یک متغیر تصادفی که بر اساس رابطه‌ی زیر به دست می‌آید:

(2_1)

$$P_k(r,s) = \begin{cases} \dfrac{Ph(r,s).[D(r,s)]^{\beta}}{\sum_{u \in j_k(r)} Ph(r,u).[D(r,u)]^{\beta}} & ; \text{اگر } s \text{ جزء شهرهای باقی مانده باشد :} \\ 0 & ; \text{اگر } s \text{ جزء شهرهای باقی مانده نباشد :} \end{cases}$$

رابطه‌ی فوق، احتمال انتخاب مسیر برای مورچه k را از شهر s نشان می‌دهد.

۱-۴-۴ شبه کد و فلوچارت الگوریتم:

۱) مقداردهی اولیه‌ی مورچه:

در این مرحله کلونی مورچه تولید می‌شود. مورچه‌ها در حالت اولیه قرار می‌گیرند و فرومون اولیه به‌اندازه $Ph0$ مقداردهی می‌شود.

۲) ارزیابی شایستگی:

در این مرحله سازگاری کلیه مورچه‌ها بر پایه تابع هدف ارزیابی می‌شود. با ارزیابی صلاحیت نظیر به نظیر مورچه‌ها، فرومون به مسیر خاص شامل این مورچه‌ها اضافه می‌شود.

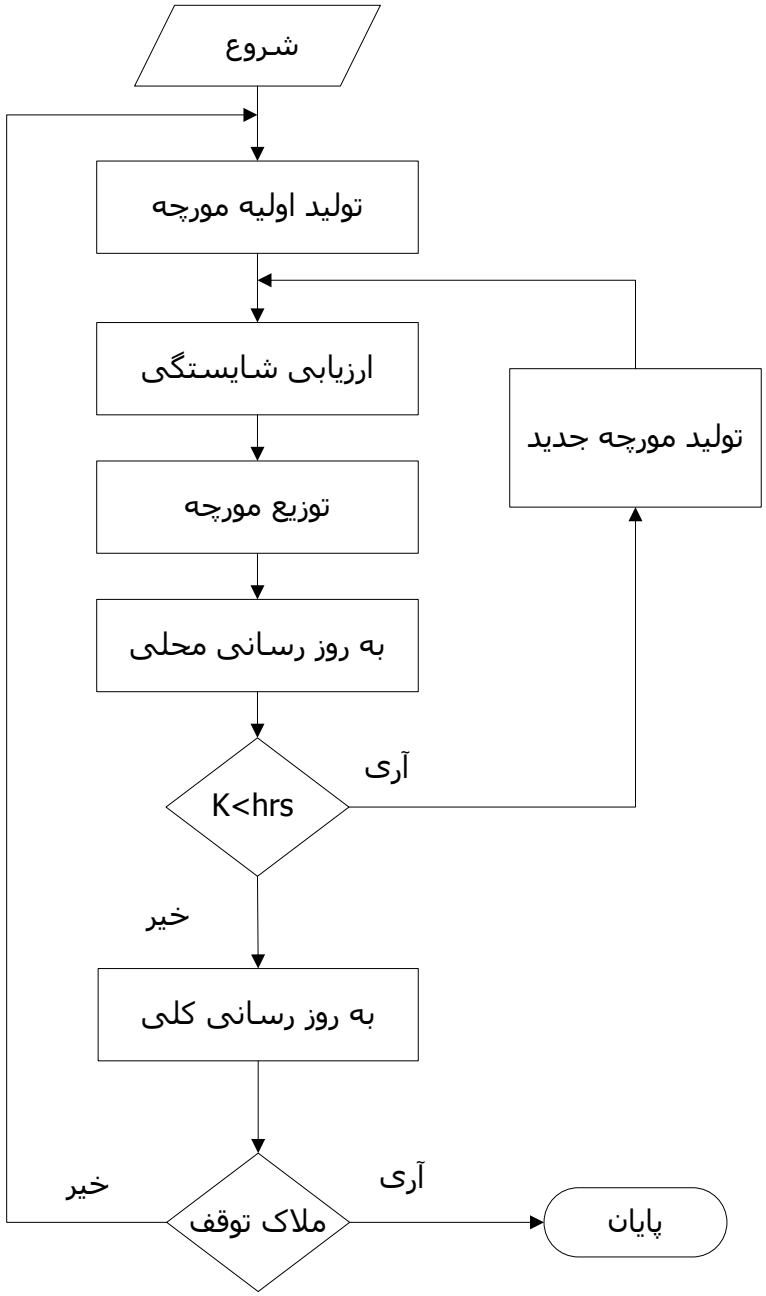
۳) توزیع مورچه:

در این مرحله مورچه‌ها بر اساس سطح فرومون و میزان مسافت توزیع می‌شوند.

۴) معیار اتمام تکرار :

فرایند تا رسیدن به حداکثر تعداد مورچه‌ها یا عدم بهبود جواب ادامه می‌یابد.

کلیه‌ی مسیرهای عبوری توسط مورچه‌ها بایستی در هر تکرار ارزشیابی شوند. در صورتی که یک مسیر بهتر در فرآیند پیدا شد آن مسیر ذخیره می‌گردد. بهترین مسیر انتخاب شده در میان کلیه‌ی تکرارها به عنوان جواب مساله برگزیده می‌شود.



شکل 1-6: فلوچارت الگوریتم مورچه

۱-۴-۵ مزیت‌ها:

همان‌طور که گ‌فته شد«تبخیر شدن فرومون» و«احتمال -تصادف» به مورچه‌ها امکان پیدا کردن کوتاه‌ترین مسیر را می‌ده د. این دو ویژگی باعث ایجاد انعطاف در حل هرگونه مساله ی بهینه‌سازی می‌شود. مثلا در گراف شهرهای مساله ی فروشنده‌ی دوره‌گرد، اگر یکی از یال ها(یا گره‌ها) حذف شود الگوریتم این توانایی را دارد تا به سرعت مسیر بهینه را با توجه به شرایط جدید پیدا کند. به این ترتیب که اگر یال(یا گره‌ی) حذف شود دیگر لازم نیست که الگوریتم از ابتدا مساله را حل کند بلکه از جایی که مساله حل شده

تا محل حذف یال(یا گره) هنوز بهترین مسیر را داریم، از این به بعد مورچه‌ها می‌توانند پس از مدت کوتاهی مسیر بهینه(کوتاه‌ترین مسیر) را بیابند.

۱-۴-۶ کاربردها :

از کاربردهای ACO می‌توان به بهینه کردن هر مساله ای که نیاز به یافتن کوتاه ترین مسیر دارد، اشاره نمود :

- مسیر یابی داخل شهری و بین شهری
- مسیر یابی بین پست‌های شبکه‌های توزیع برق ولتاژ بالا
- مسیر یابی شبکه‌های کامپیوتری

۱-۷ الگوریتم رقابت استعماری[1]

شکل 1-7 :
استعمار

بعضی از الگوریتم‌های متاهیوریستیک الهام گرفته شده از تکامل فرهنگی می‌باشند. آن‌چه که واضح است این است که تکامل فکری و فرهنگی بشر بسیار سریع‌تر از تکامل جسمی و ژنتیکی او صورت می‌پذیرد. بنابراین تکامل فرهنگی و دیدگاهی بشر نیز نادیده گرفته نشده و دسته‌ای از الگوریتم‌ها، موسوم به الگوریتم‌های فرهنگی معرفی شده‌اند. الگوریتم‌های فرهنگی در حقیقت یک دسته‌ی کاملا جدید از الگوریتم‌ها نیستند، بلکه‌ایده‌ی اصلی این است که این الگوریتم‌ها با افزودن قابلیت تکامل فرهنگی(با افزودن امکان تبادل اطلاعات میان اعضای جمعیت) به الگوریتم‌های موجود، سرعت همگرایی آن‌ها را مطابق انتظار افزایش می‌دهند. با توجه به‌اینکه اغلب روش‌های عمده و شناخته شده‌ی محاسبات تکاملی، شبیه‌سازی کامپیوتری فرایندهای طبیعی و زیستی هستند، در این قسمت یک الگوریتم جدید در زمینه‌ی محاسبات تکاملی معرفی می‌شود که بر مبنای تکامل اجتماعی و سیاسی انسان پایه‌گذاری شده است.

1- Imperialist Competitive Algorithm

الگوریتم رقابت استعماری با الهام‌گیری از یک فرایند اجتماعی- سیاسی، نسبت به روش‌های مطرح شده دارای توانایی بالایی بوده و تا حد بسیار زیادی نیز سریع می‌باشد.

الگوریتم رقابت استعماری در وهله‌ی اول با داشتن یک دیدگاه کاملا نو به مبحث بهینه‌سازی، پیوندی جدید میان علوم انسانی و اجتماعی از یک سو و علوم فنی و ریاضی از سوی دیگر، برقرار می‌کند. ارتباط میان این دو شاخه از علم به گونه‌ای می‌باشد که غالبا ریاضیات به عنوان ابزاری قوی و دقیق در خدمت علوم انسانی کلی‌نگر قرار گرفته و به درک و تحلیل نتایج آن کمک می‌کند. اما الگوریتم رقابت استعماری بر خلاف معمول، نقطه‌ی قوت علوم انسانی و اجتماعی، یعنی کلی‌نگری و وسعت دید آن را به خدمت ریاضیات درآورده و از آن به عنوان ابزاری برای درک بهتر ریاضیات و حل بهتر مسائل ریاضی استفاده می‌کند. بنابراین حتی بدون در نظر گرفتن قابلیت‌های ریاضی و عملی روش توسعه داده شده، پیوند ایجاد شده میان این دو شاخه به ظاهر جدا از هم، به عنوان یک پژوهش میان رشته‌ای، در نوع خود دارای ارزش بسیاری می‌باشد.

الگوریتم رقابت استعماری روشی در حوزه ی محاسبات تکاملی است که به یافتن پاسخ بهینه ی مسائل مختلف بهینه‌سازی می‌پردازد. این الگوریتم با مدلسازی ریاضی فرایند تکامل اجتماعی - سیاسی، الگوریتمی برای حل مسائل ریاضی بهینه‌سازی ارائه می‌دهد.

پایه‌های اصلی این الگوریتم را *سیاست همسان سازی*[1]، *رقابت استعماری*[2] و *انقلاب*[3] تشکیل می‌دهند. این الگوریتم با تقلید از روند تکامل اجتماعی، اقتصادی و سیاسی کشورها و با مدلسازی ریاضی بخشهایی از این فرایند، عملگرهایی را در قالب منظم به صورت الگوریتم ارائه می‌دهد که می‌توانند به حل مسائل پیچیده بهینه‌سازی کمک کنند. در واقع این الگوریتم جواب های مساله‌ی بهینه‌سازی را در قالب کشورها نگریسته و سعی می‌کند در طی فرایندی تکرار شونده این جواب‌ها را رفته رفته بهبود داده و در نهایت به جواب بهینه‌ی مساله برساند.

امپریالیسم، در لغت به سیاست توسعه قدرت و نفوذ یک کشور در حوزه خارج از قلمرو شناخته شده برای آن، اطلاق می‌شود. یک کشور می‌تواند کشور دیگر را به طور قانونگذاری مستقیم و یا از طریق روش‌های غیر مستقیم، مثل کنترل کالاها و مواد خام، کنترل کند. استعمار یک پدیده ی ذاتی در تاریخ بوده‌است. استعمار در مراحل ابتدایی، به صورت نفوذ سیاسی و نظامی در کشورها و به صورت صرف استفاده از منابع زمینی، انسانی و سیاسی بوده است. بعضی مواقع نیز استعمار، به صرف جلوگیری از نفوذ کشور استعمارگر رقیب انجام می‌شد. به هر حال کشورهای استعمارگر رقابت شدیدی را برای به استعمار کشیدن مستعمرات همدیگر نشان می‌دادند.

1- Assimilation
2- Imperialistic Competition
3- Revolution

همانند دیگر الگوریتم‌های تکاملی، این الگوریتم، نیز با تعدادی جمعیت اولیه ی تصادفی که هر کدام از آن‌ها یک «کشور» نامیده می‌شوند؛ شروع می‌شود. تعدادی از بهترین عناصر جمعیت به عنوان امپریالیست انتخاب می‌شوند. باقیمانده‌ی جمعیت نیز به عنوان مستعمره، در نظر گرفته می‌شوند. استعمارگران بسته به قدرتشان، این مستعمرات را با یک روند خاص که در ادامه می‌آید؛ به سمت خود می‌کشند. قدرت کل هر امپراطوری، به هر دو بخش تشکیل دهنده ی آن یعنی کشور امپریالیست (به عنوان هسته مرکزی) و مستعمرات آن، بستگی دارد. در حالت ریاضی، این وابستگی با تعریف قدرت امپراطوری به صورت مجوع قدرت کشور امپریالیست، به اضافه‌ی درصدی از میانگین قدرت مستعمرات آن، مدل شده‌است. با شکل‌گیری امپراطوری‌های اولیه، رقابت امپریالیستی میان آن‌ها شروع می‌شود. هر امپراطوری‌ای که نتواند در رقابت استعماری، موفق عمل کرده و بر قدرت خود بیفزاید(و یا حداقل از کاهش نفوذش جلوگیری کند)، از صحنه‌ی رقابت استعماری، حذف خواهد شد. بنابراین بقای یک امپراطوری، وابسته به قدرت آن در جذب مستعمرات امپراطوری‌های رقیب، و به سیطره در آوردن آن ها خواهد بود. در نتیجه، در جریان رقابت‌های امپریالیستی، به تدریج بر قدرت امپراطوری‌های بزرگ تر افزوده شده و امپراطوری‌های ضعیف‌تر، حذف خواهند شد. امپراطوری‌ها برای افزایش قدرت خود، مجبور خواهند شد تا مستعمرات خود را نیز پیشرفت دهند.

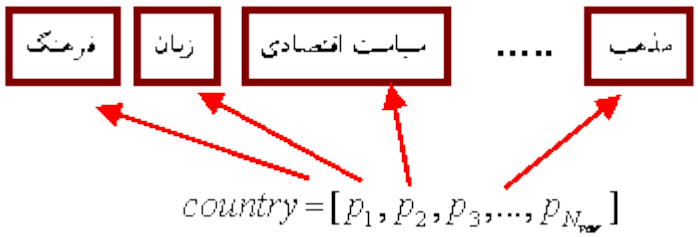
به طور خلاصه این الگوریتم به استعمار به عنوان جزئی لاینفک از سیر تکامل تاریخی انسان نگریسته و از چگونگی اثرگذاری آن بر کشورهای استعمارگر و مستعمره و نیز کل تاریخ، به عنوان منبع الهام یک الگوریتم کارا و نو در زمینه‌ی محاسبات تکاملی استفاده کرده است.

۱-۵-۱ شکل‌دهی امپراطوری‌های اولیه

در بهینه‌سازی، هدف یافتن یک جواب بهینه بر حسب متغیرهای مساله است. ما یک آرایه از متغیرهای مساله را که باید بهینه شوند، ایجاد می‌کنیم و آن را یک کشور می‌نامیم. در یک مساله ی بهینه‌سازی N_{var} بعدی، یک کشور، یک آرایه به طول N_{var} * ۱ است. این آرایه به صورت زیر تعریف می‌شود.

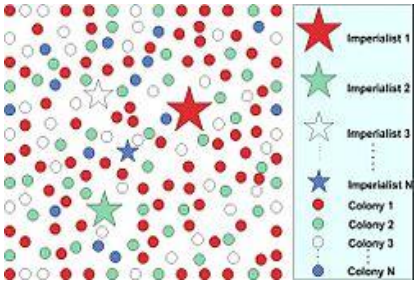
$$country = [p_1, p_2, ..., p_{Nvar}]$$

مقادیر متغیرها در یک کشور، به صورت اعداد اعشاری نمایش داده می‌شوند. از دیدگاه تاریخی فرهنگی، اجزای تشکیل دهنده ی یک کشور را می‌توان ویژگی‌های اجتماعی - سیاسی آن کشور، همچون فرهنگ، زبان، ساختار اقتصادی و سایر ویژگی‌ها در نظر گرفت. شکل زیر نحوه ی تناظر متغیرهای بهینه‌سازی مساله با ویژگی‌های اجتماعی- سیاسی را نشان می‌دهد.



شکل 1-8: شکل دهی امپراطوری‌های اولیه

برای شروع الگوریتم، تعداد $N_{country}$ کشور اولیه را ایجاد می‌کنیم. تا N_{imp} از بهترین اعضای این جمعیت (کشورهای دارای کمترین مقدار تابع هزینه) را به عنوان امپریالیست انتخاب می‌کنیم. باقیمانده N_{col} تا از کشورها، مستعمراتی را تشکیل می‌دهند که هرکدام به یک امپراطوری تعلق دارند. برای تقسیم مستعمرات اولیه بین امپریالست‌ها، به هر امپریالیست، تعدادی از مستعمرات را که این تعداد، متناسب با قدرت آن است، می‌دهیم. در شکل زیر نحوه تقسیم مستعمرات، میان کشورهای استعمارگر به صورت نمادین نشان داده شده است.



شکل 1-9: نحوه تقسیم مستعمرات، میان کشورهای استعمارگر

۱-۵-۲ سیاست جذب: حرکت مستعمره‌ها به سمت امپریالیست

سیاست همگون‌سازی(جذب) با هدف تحلیل فرهنگ و ساختار اجتماعی مستعمرات در فرهنگ حکومت مرکزی انجام می‌گرفت . کشورهای استعمارگر، برای افزایش نفوذ خود، شروع به ایجاد عمران (ایجاد زیرساخت‌های حمل و نقل، تاسیس دانشگاه و ...) کردند. به عنوان مثال کشورهایی نظیر انگلیس و فرانسه با تعقیب سیاست همگون‌سازی در مستعمرات خود در فکر ایجاد انگیس نو و فرانسه ی نو در مستعمرات خویش بودند. با در نظر گرفتن شیوه ی نمایش یک کشور در حل مس ائل بهینه‌سازی، در حقیقت این حکومت مرکزی با اعمال سیاست جذب ، سعی داشت تا کشور مستعمره را در راستای ابعاد مختلف اجتماعی- سیاسی به خود نزدیک کند. این بخش از فرایند استعمار در الگوریتم بهینه‌سازی، به صورت حرکت مستعمرات به سمت کشور امپریالیست، مدل شده‌است.

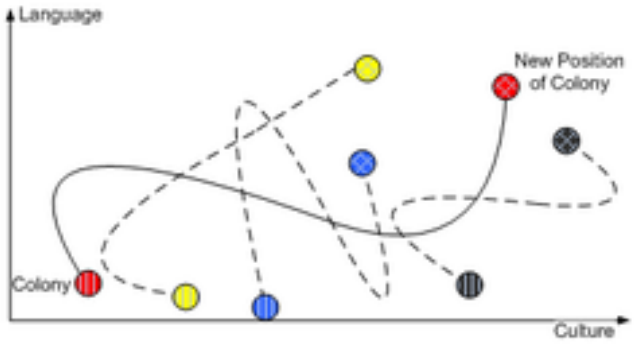
در راستای این سیاست، کشور مستعمره، به‌اندازه x واحد در جهت خط واصل مستعمره به استعمارگر، حرکت کرده و به موقعیت جدید کشانده می‌شود. x عددی تصادفی با توزیع یکنواخت می‌باشد. اگر فاصله ی میان استعمارگر و مستعمره با d نشان داده شود، معمولاً برای d داریم:

$$x \sim U(0,ß*d) \qquad (3\_1)$$

که در آن ß عددی بزرگتر از یک و نزدیک به ۲ می‌باشد. یک انتخاب مناسب می‌تواند ß=۲ باشد. وجود ضریب ß≥۱ باعث می‌شود تا کشور مستعمره در حین حرکت به سمت کشور استعمارگر، از جهت‌های مختلف به آن نزدیک شود. همچنین در کنار این حرکت، یک انحراف زاویه‌ای کوچک نیز با توزیع یکنواخت به مسیر حرکت افزوده می‌شود.

۱-۵-۳ انقلاب؛ تغییرات ناگهانی در موقعیت یک کشور

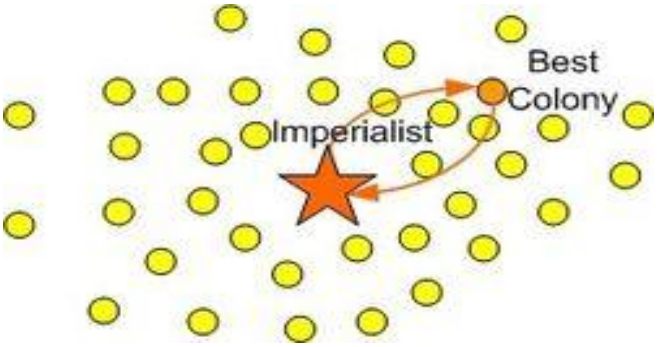
بروز انقلاب تغییرات ناگهانی را در ویژگی های اجتماعی سیاسی یک کشور ایجاد می کند. در الگوریتم رقابت استعماری، انقلاب با جابه‌جایی تصادفی یک کشور مستعمره به یک موقعیت تصادفی جدید مدلسازی می‌شود. انقلاب از دیدگاه الگوریتمی باعث می‌شود کلیت حرکت تکاملی از گیر کردن در دره های محلی بهینگی نجات یابد که در بعضی موارد باعث بهبود موقعیت یک کشور شده و آن را به یک محدوده‌ی بهینگی بهتری می‌برد.



شکل 1-10: تغییرات ناگهانی و موقعیت‌های انقلاب

۱-۵-۴ جابه‌جایی موقعیت مستعمره و امپریالیست

در حین حرکت مستعمرات به سمت کشور استعمارگر، ممکن است بعضی از این مستعمرات به موقعیتی بهتر از امپریالیست برسند(به نقاطی در تابع هزینه برسند که هزینه کمتری را نسبت به مقدار تابع هزینه در موقعیت امپریالیست، تولید می‌کنند.) در این حالت، کشور استعمارگر و کشور مستعمره، جای خود را با همدیگر عوض کرده و الگوریتم با کشور استعمارگر در موقعیت جدید ادامه یافته و این بار این کشور امپریالیست جدید است که شروع به اعمال سیاست همگون‌سازی بر مستعمرات خود می‌کند. نحوه ی جابه‌جایی موقعیت مستعمره و استعمارگر در شکل زیر نشان داده شده است.

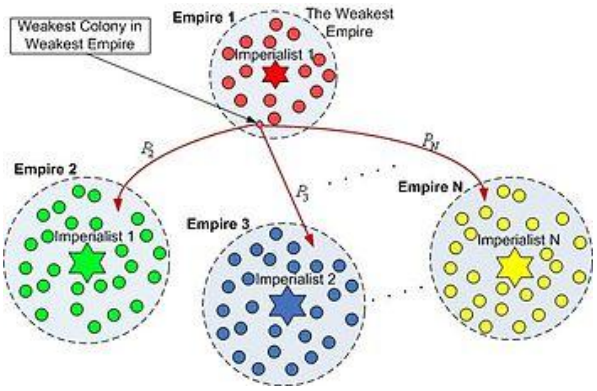


شکل 1-11: تعویض موقعیت مستعمره و استعمارگر

۱-۵-۵ رقابت استعماری

قدرت یک امپراطوری به صورت قدرت کشور استعمارگر، به اضافه درصدی از قدرت کل مستعمرات آن تعریف می‌شود.

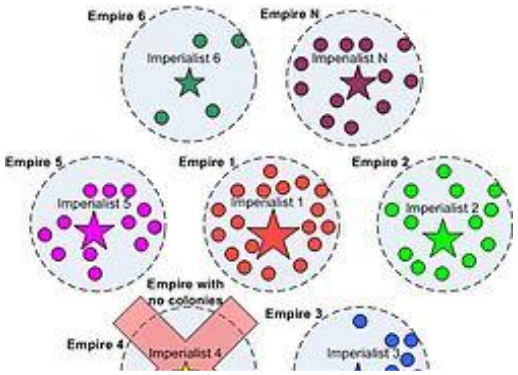
هر امپراطوری‌ای که نتواند بر قدرت خود بیفزاید و قدرت رقابت خود را از دست بدهد، در جریان رقابت‌های امپریالیستی، حذف خواهد شد. این حذف شدن، به صورت تدریجی صورت می‌پذیرد. بدین معنی که به مرور زمان، امپراطوری‌های ضعیف، مستعمرات خود را از دست داده و امپراطوری‌های قوی تر، این مستعمرات را تصاحب کرده و بر قدرت خویش می‌افزایند. در الگوریتم رقابت استعماری، امپراطوری در حال حذف، ضعیف‌ترین امپراطوری موجود است. بدین ترتیب، در تکرار الگوریتم، یکی یا چند مورد از ضعیف‌ترین مستعمرات ضعیف‌ترین امپراطوری را برداشته و برای تصاحب این مستعمرات، رقابتی را میان کلیه ی امپراطوری‌ها ایجاد می‌کنیم. مستعمرات مذکور، لزوماً توسط قوی ترین امپراطوری، تصاحب نخواهند شد، بلکه امپراطوری‌های قوی‌تر، احتمال تصاحب بیشتری دارند.



شکل 1-12: رقابت استعمارگران

۱-۵-۶ سقوط امپراطوری‌های ضعیف

در جریان رقابت‌های امپریالیستی، خواه ناخواه، امپراطوری های ضعیف به تدریج سقوط کرده و مستعمراتشان به دست امپراطوری‌های قوی‌تر می‌افتد. شروط متفاوتی را می‌توان برای سقوط یک امپراطوری در نظر گرفت. در الگوریتم پیشنهاد شده، یک امپراطوری زمانی حذف شده تلقی می‌شود که مستعمرات خود را از دست داده باشد.



شکل 1-13: سقوط یک امپراطوری

۱-۵-۷ شبه کد

مراحل ذکر شده در بالا را می‌توان به صورت شبه کد زیر خلاصه کرد.

۱. چند نقطه‌ی تصادفی روی تابع انتخاب کرده و امپراطوری‌های اولیه را تشکیل بده.
۲. مستعمرات را به سمت کشور امپریالیست حرکت بده (سیاست همسان‌سازی یا جذب).
۳. عملگر انقلاب را اعمال کن.
۴. اگر مستعمره‌ای در یک امپراطوری، وجود داشته باشد که هزینه‌ای کمتر از امپریالیست داشته باشد؛ جای مستعمره و امپریالیست را با هم عوض کن.
۵. هزینه‌ی کل یک امپراطوری را حساب کن(با در نظر گرفتن هزیره‌ی امپریالیست و مستعمراتشان).
۶. یک(یا چند) مستعمره از ضعیف‌ترین امپراطوری انتخاب کرده و آن را به امپراطوری‌ای که بیشترین احتمال تصاحب را دارد، بده.
۷. امپراطوری‌های ضعیف را حذف کن.
۸. اگر تنها یک امپراطوری باقی مانده توقف کن وگرنه به مرحله‌ی۲ بازگرد.

۱-۵-۸ مزیت‌ها

مزایای الگوریتم اجتماعی پیشنهادی را می‌توان به صورت زیر خلاصه کرد:

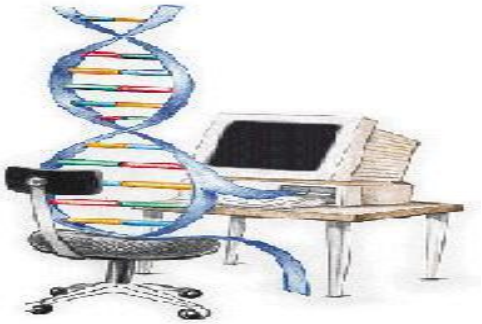
۱. نو بودن ایده‌ی پایه‌ای الگوریتم: به عنوان اولین الگوریتم بهینه‌سازی مبتنی بر یک فرایند اجتماعی -سیاسی

۲. توانایی بهینه‌سازی هم‌تراز و حتی بالاتر در مقایسه با الگوریتم‌های مختلف بهینه‌سازی، در مواجهه با انواع مسائل بهینه‌سازی
۳. سرعت مناسب یافتن جواب بهینه

۱-۵-۹ کلوبردها

- طراحی بهینه‌ی کنترل کننده برای ورودی - چند خروجی تبرید و صنعتی ستون تقطیر
- طراحی بهینه‌ی سیستم‌های پیشنهاددهی هوشمند
- طراحی بهینه‌ی آنتن‌های آرایه‌ای
- حل مسائل برنامه‌ریزی تولید در حوزه‌ی مهندسی صنایع و مدیریت
- یادگیری و تحلیل شبکه‌های عصبی مصنوعی
- طراحی بهینه‌ی موتورهای القای خطی
- طراحی استراتژی بهینه در نظریه‌ی بازی‌ها برای رسیدن به نقطه‌ی تعادل نش
- طراحی بهینه‌ی کنترل کننده‌های فازی
- انتخاب و بهینه‌سازی سبد سهام

۱-۶ الگوریتم ژنتیک[1]



شکل 1-14: نمای گرافیکی ژن

الگوریتم ژنتیک، روش بهینه‌سازی الهام گرفته از طبیعت جاندار(موجودات زنده) است که می‌توان در طبقه‌بندی‌ها، از آن به عنوان یک روش عددی جستجوی مستقیم و تصادفی یاد کرد. این الگوریتم، الگوریتمی مبتنی بر تکرار است و اصول اولیه‌ی آن از علم ژنتیک[2] اقتباس گردیده است و با تقلید از تعدادی از فرآیندهای مشاهده شده در تکامل طبیعی اختراع شده است و به طور موثری از معرفت قدیمی موجود در یک جمعیت استفاده می‌کند، تا حل‌های جدید و بهبود یافته را ایجاد کند. این الگوریتم در مسائل متنوعی

1- Genetic algorithm
2- Genetic science

نظیر بهینه‌سازی، شناسایی و کنترل سیستم، پردازش تصویر و مسائل ترکیبی، تعی ین توپولوژی و آموزش شبکه‌های عصبی مصنوعی و سیستم‌های مبتنی بر تصمیم و قاعده به کار می‌رود.

علم ژنتیک، علمی است که دربار ه‌ی چگونگی توارث و انتقال صفحات بیولوژیکی از نسلی به نسل بعد صحبت می‌کند. عامل اصلی انتقال صفحات بیولوژیکی در موجودات زنده ، کروموزوم‌ها و ژن‌ها می‌باشند و نحوه‌ی عملکرد آن‌ها به گونه‌ای است که در نهایت ژن‌ها و کروموزوم‌های برتر، قوی مانده و ژن‌های ضعیف‌تر از بین می‌روند. به عبارت دیگر نتیج ه‌ی عملیات متقابل ژن‌ها و کروموزوم‌ها ، باقی ماندن موجودات اَصلح و برتر می‌باشد.

اساس این الگوریتم ، قانون تکامل داروین (بقای بهترین) است که می‌گوید: موجودات ضعیف‌تر از بین می‌روند و موجودات قوی‌تر باقی می‌مانند. در واقع تکامل فر ایندی است که روی رشته‌ها صورت می‌گیرد، نه روی موجودات زنده‌ای که معرف موجودات رشته است. در واقع، قانون انتخاب طبیعی [1] برای بقا می‌گوید که هر چه امکان تطبیق موجود بیشتر باشد بقای موجود امکان‌پذیرتر است و احتمال تولید مثل بیشتری، برایش وجود دارد. این قانون بر اساس پیوند بین رشته‌ها و عملکرد ساختمان‌های رمزگشایی شده‌ی آن‌ها می‌باشد.

الگوریتم ژنتیک دارای چند اختلاف اساسی با روش‌های جستجوی مرسوم می‌باشد که در زیر به تعدادی از آنها اشاره می‌کنیم.

- الگوریتم ژنتیک با رشته‌های بیتی کار می‌کند که هر کدام از این رشته‌ها کل مجموعه‌ی متغیرها را نشان می‌دهد حال آنکه بیشتر روش‌ها به طور مستقل با متغیرهای ویژه برخورد می‌کنند.
- در الگوریتم ژنتیک، روش‌های جستجو بر اساس مکانیزم انتخاب و ژنتیک طبیعی عمل می‌نمایند.

این الگوریتم‌ها مناسب‌ترین رشته‌ها را از میان اطلاعات تصادفی سازماندهی شده انتخاب می‌کنند. در هر نسل یک گروه جدید رشته‌ها با استفاده از بهترین قسمت‌های دنباله‌های قبلی و بخش جدید اتفاقی ، برای رسیدن به یک جواب مناسب به وجود می‌آیند. هرچند این الگوریتم‌ها تصادفی هستند ولی به طور کارآمدی به اکتشاف اطلاعات گذشته در فضای جستجو می‌پردازند تا در نقطه ی جستجوی جدید با پاسخ‌های بهتر به سمت بهترین جواب پیش روند. سپس هر نقطه را به صورت انفرادی امتحان می‌کنند و با ترکیب محتویات آنها یک جمعیت جدید را که شامل نقاط بهبود یافته است تشکیل می‌ده ند. الگوریتم ژنتیک فقط نیاز به اطلاعاتی در مورد کیفیت حل‌های ایجاد شده به وسیله‌ی هر مجموعه از متغیرها دارد، در صورتی که بعضی از روش‌های بهینه‌سازی نیاز به اطلاعات یا حتی نیاز به شناخت کامل از ساختمان مسأله و متغیرها دارند. چون الگوریتم ژنتیک نیاز به چنین اطلاعات مشخصی از مسأله ندارد بنابراین قابل انعطاف‌تر از بیشتر روش‌های جستجو است. همچنین الگوریتم ژنتیک از روش‌های جستجوی نوعی که برای راهنمایی جهت روش‌های جستجویشان از انتخاب تصادفی استفاده می‌کنند متفاوت است زیرا اگر چه برای تعریف

3- The law of Natural Selection

روش‌های تصمیم‌گیری از تصادف و شانس استفاده می‌کند ولی در فضای جستجو به صورت تصادفی قدم نمی‌زند.

- الگوریتم ژنتیک از قوانین احتمالی پیروی می‌کند و نه از قوانین قطعی.

۱-۶-۱ مکانیزم الگوریتم ژنتیک

الگوریتم ژنتیک به عنوان یک الگوریتم محاسباتی بهینه‌سازی با در نظر گرفتن مجموعه‌ای از نقاط فضای جواب در هر تکرار محاسباتی به نحو مؤثری نواحی مختلف فضای جواب را جستجو می‌کند. در مکانیزم جستجو، گرچه مقدار تابع هدف تمام فضای جواب محاسبه نمی‌شود ولی مقدار محاسبه شده ی تابع هدف برای هر نقطه، در متوسط‌گیری آماری تابع هدف در کلیه ی زیر فضاهایی که آن نقطه به آن ها وابسته بوده دخالت داده می‌شود و این زیر فضاها به طور موازی از نظر تابع هدف متوسط‌گیری آماری می‌شوند. این روند باعث می‌شود که جستجوی فضا به نواحی ای از آن که متوسط آماری تابع هدف در آن ها زیاد بوده و امکان وجود نقطه‌ی بهینه مطلق در آن‌ها بیشتر است سوق پیدا کند.

امتیاز دیگر این الگوریتم آن است که هیچ محدودیتی برای تابع بهینه شونده، مثل مشتق‌پذیری یا پیوستگی لازم ندارد و در روند جستجو ی خود تنها به تعیین مقدار تابع هدف در نقاط مختلف نیاز دارد و هیچ اطلاعات کمکی دیگری، مثل مشتق تابع را استفاده نمی‌کند. لذا می‌توان در مسائل مختلف اعم از خطی، پیوسته یا گسسته از آن استفاده کرد و به سهولت با مسائل مختلف قابل تطبیق است.

در هر تکرار هر یک از رشته‌های موجود در جمعیت رشته‌ها، رمزگشایی شده و مقدار تابع هدف برای آن به دست می‌آید. بر اساس مقادیر به دست آمده ی تابع هدف در جمعیت رشته‌ها، به هررشته یک عدد برازندگی نسبت داده می‌شود. این عدد برازندگی ، احتمال انتخاب را برای هر رشته تعیین خواهد کرد. بر اساس این احتمال انتخاب، مجموعه‌ای از رشته‌ها انتخاب شده و با اعمال عملکردهای ژنتیکی روی آن ها، رشته‌های جدید جایگزین رشته‌هایی از جمعیت اولیه می‌شوند تا تعداد جمعیت رشته‌ها در تکرارهای محاسباتی مختلف ثابت باشد.

مکانیزم‌های 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 مرحله‌ی جایگزینی نسبت به دیگر رشته‌ها مقاوم‌تر هستند. بدین سبب جمعیت دنباله‌ها در یک رقابت بر اساس تابع هدف در طی نسل‌های مختلف، کامل شده و متوسط مقدار تابع هدف در جمعیت رشته‌ها افزایش می‌یابد. به طور کلی در این الگوریتم ضمن آنکه در هر تکرار محاسباتی، توسط عملگرهای ژنتیکی نقاطی جدید از فضای جواب مورد جستجو قرار می‌گیرند توسط مکانیزم انتخاب، روند جستجوی نواحی از فضا را که متوسط آماری تابع هدف در آن‌ها بیشتر است، کنکاش می‌کند. بر اساس سیکل اجرایی فوق، در هر تکرار محاسباتی توسط عملگرهای ژنتیکی، نقاط جدیدی از فضای جواب مورد جستجو قرار می‌گیرند که بر این اساس، در هر

تکرار محاسباتی، سه عملگر اصلی روی رشته‌ها عمل می‌کند ؛ این سه عملگر عبارتند از: دو عملگر ژنتیکی و عملگر انتخابی تصادفی.

بدن همه‌ی موجودات زنده از سلول‌ها تشکیل شده است و در هر سلولی دسته ای کروموزوم‌های یکسان وجود دارد. کروموزوم‌ها [1] رشته‌هایی از DNA هستند که در واقع الگویی برای تمام بدن هستند. هر کروموزومی محتوی دسته‌هایی DNA است که ژن [2] نامیده می‌شوند و هر ژنی پروتئین خاصی را رمزگذاری می‌کند. اساسا می‌توان گفت که هرژن، ویژگی خاصی(مثلا رنگ چشم) را رمزگذاری می‌کند.

در تولید مثل، ابتدا ترکیب(یا تغییر) اتفاق می‌افتد. ژن‌های والدین برای ایجاد کروموزوم‌های جدید ترکیب می‌شوند. سپس جنین تشکیل شده دچار تغییر می‌شود. جهش به این معناست که عناصر DNA، کمی تغییر پیدا می‌کنند و این تغییرات اغلب نتیجه نسخه‌برداری غلط از ژن‌های والدین است. میزان شایستگی موجود زنده(جنین) به واسطه‌ی بقای آن اندازه‌گیری می‌شود.

در الگوریتم ژنتیک، مجموعه ای از متغیرهای طراحی را توسط رشته‌هایی با طول ثابت یا متغیر کدگذاری می‌کنند که در سیستم‌های بیولوژیکی آن ها را کروموزوم می‌نامند. هر رشته یا کروموزوم یک نقطه‌ی پاسخ در فضای جستجو را نشان می‌دهد. الگوریتم‌های وراثتی ، فرآیندهای تکراری هستند که هر مرحله‌ی تکراری را نسل و مجموعه‌هایی از پاسخ‌ها در هر نسل را جمعیت نامیده‌اند.

الگوریتم‌های ژنتیک، جستجوی اصلی را در فضای پاسخ به اجرا می‌گذارند. این الگوریتم‌ها با تولید نسل آغاز می‌شوند که وظیفه‌ی ایجاد مجموعه نقاط جستجوی اولیه به نام «جمعیت اولیه» را بر عهده دارند و به طور انتخابی یا تصادفی تعیین می‌شوند. از آنجایی که الگوریتم‌های ژنتیک برای هدایت عملیات جستجو به طرف نقطه‌ی بهینه از روش‌های آماری استفاده می‌کنند، در فرآیندی که به انتخاب طبیعی وابسته است، جمعیت موجود به تناسب برازندگی افراد آن برای نسل بعد انتخاب می‌شود. سپس عملگرهای ژنتیکی شامل انتخاب، پیوند(ترکیب)، جهش و دیگر عملگرهای احتمالی اعمال شده و جمعیت جدید به وجود می‌آید. پس از آن جمعیت جدیدی جایگزین جمعیت پیشین می‌شود و این چرخه ادامه می‌یابد.

معمولاً جمعیت جدید برازندگی بیشتری دارد و این بدان معناست که از نسلی به نسل دیگر جمعیت بهبود می‌آید.

۱-۶-۲ عملگرهای الگوریتم ژنتیک

به طور خلاصه الگوریتم ژنتیک از عملگرهای زیر تشکیل شده است:

۱-۶-۲-۱ کدگذاری [3]

1- Chromosome
1- Gene
2- Coding

این مرحله شاید مشکل ترین مرحله‌ی حل مسأله در این الگوریتم باشد. الگوریتم ژنتیک به جای اینکه بر روی پارامترها یا متغیرهای مسأله کار کند، با شکل کد شد ه‌ی آن‌ها سروکار دارد. یکی از روش های کد کردن، کد کردن دودویی می‌باشد که در آن هدف، تبدیل جواب مسأله به رشته‌ای از اعداد باینری (در مبنای ۲) است.

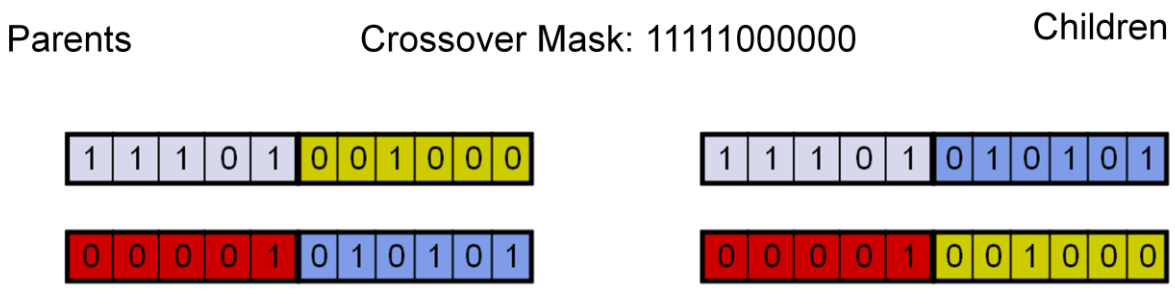
۱-۶-۲-۲ ارزیابی[1]

تابع برازندگی را از اعمال تبدیل مناسب بر روی تابع هدف ، یعنی تابعی که قرار است بهینه شود به دست می‌آورند. این تابع هر رشته را با یک مقدار عددی ارزیابی می‌کند که کیفیت آن را مشخص می‌نماید. هر چه کیفیت رشته ی جواب بالاتر باشد، مقدار برازندگی جواب بیشتر است و احتمال مشارکت برای تولید نسل بعدی نیز افزایش خواهد یافت.

۱-۶-۲-۳ ترکیب[2]

مهم‌ترین عملگر در الگوریتم ژنتیک، عملگر ترکیب است. ترکیب فرآیندی است که در آن نسل قدیمی کروموزوم‌ها با یکدیگر مخلوط و ترکیب می‌شوند تا نسل تازه‌ای از کروموزوم‌ها بوجود بیایند.

جفت‌هایی که در قسمت انتخاب به عنوان والد در نظر گرفته شدند در این قسمت ژن‌هایشان را با هم مبادله می‌کنند و اعضای جدید به وجود می‌آورند. ترکیب در الگوریتم ژنتیک باعث از بین رفتن پراکندگی یا تنوع ژنتیکی جمعیت می‌شود زیرا اجازه می‌دهد ژن‌های خوب یکدیگر را بیابند.



شکل 1-15: ترکیب در الگوریتم ژنتیک

1- Evaluation
2- Crossover

۱-۶-۲-۴ جهش[1]

جهش نیز عملگر دیگری هست که جواب‌های ممکن دیگری را متولد می‌کند. در الگوریتم ژنتیک بعد از اینکه یک عضو در جمعیت جدید به وجود آمد هر ژن آن با احتمال جهش، جهش می‌یابد. در جهش ممکن است ژنی از مجموعه ژن‌های جمعیت حذف شود یا ژنی که تا به حال در جمعیت وجود نداشته است به آن اضافه شود. جهش یک ژن به معنای تغییر آن ژن است و وابسته به نوع کدگذاری ، روش‌های متفاوت جهش استفاده می‌شود.

۱-۶-۲-۵ رمزگشایی[2]

رمزگشایی، عکس عمل رمزگذاری است. در این مرحله بعد از اینکه الگوریتم بهترین جواب را برای مسأله ارائه کرد لازم است عکس عمل رمزگذاری روی جواب‌ها یا همان عمل رمزگشایی اعمال شود تا بتوانیم نسخه‌ی واقعی جواب را به وضوح دردست داشته باشیم.

۱-۶-۳ شبه کد

در حالت کلی وقتی یک الگوریتم ژنتیکی اعمال می‌شود چرخه زیر را طی می‌کند:

ابتدا یک جمعیت اولیه از افراد به طور اتفاقی و بدون در نظرگرفتن معیار خاصی انتخاب می‌شود. برای تمامی کروموزوم‌های(افراد) نسل صفر مقدار برازش با توجه به تابع پردازش که ممکن است بسیار ساده یا پیچیده باشد تعیین می‌شود. سپس با مکانیزم‌های مختلف تعریف شده برای عملگر انتخاب زیرمجموعه‌ای از جمعیت اولیه انتخاب خواهد شد. سپس روی این افراد انتخاب شده عملیات برش و جهش در صورت لزوم با توجه به صورت مسأله اعمال خواهد شد.

حال باید این افراد که مکانیزم الگوریتم ژنتیک در موردشان اعمال شده است با افراد جمعیت اولیه(نسل صفر) از لحاظ مقدار برازش مقایسه شوند .(قطع توقع داریم که افراد نسل اول با توجه به یکبار اعمال الگوریتم‌های ژنتیک روی آنان از شایستگی بیشتری برخوردار باشند، اما الزاماً چنین نخواهد بود.) به هرحال افرادی باقی خواهند ماند که بیشترین مقدار برازش را داشته باشند. چنین افرادی در مقام یک مجموعه به عنوان جمعیت اولیه برای مرحله بعدی الگوریتم عمل خواهد کرد.

هر مرحله تکرار الگوریتم یک نسل جدید را ایجاد می‌کند که با توجه به اصلاحاتی که در آن صورت پذیرفته است رو به سوی تکامل خواهد داشت. تذکر این نکته خالی از لطف نیست که هر چند الگوریتم‌های ژنتیک دارای پایه ریاضی متقن و مشخصی نیستند اما به عنوان یک مدل اجرایی و مطمئن که به خوبی نیز پیاده‌سازی می‌شود کارایی خود را نشان داده‌اند.

1- Mutation
2- Decoding

طرح کلی یک الگوریتم به شرح زیر می باشد:

۱ - آغاز: جمعیت n کروموزومی به صورت تصادفی ایجاد کنید(راه‌حل‌های مناسب مسأله).
۲ - ارزش گذاری: برازندگی f(x) هر کروموزوم x در جمعیت را ارزیابی کنید.
۳ - جمعیت جدید: جمعیت جدیدی را تشکیل دهید. مراحل زیر را تکرار کنید تا جمعیت جدید کامل شود:

انتخاب: دو کروموزوم(والدین) را با توجه به برازندگی آن ها از میان جمعیت انتخاب کنید(هر چه برازندگی بیشتر باشد شانس انتخاب بیشتر است.)

ترکیب: با توجه به احتمال ترکیب شدن[1]، والدین را برای تشکیل فرزندان جدید[2] با هم ترکیب کنید.

جهش: با توجه به احتمال جهش ، فرزندان را در هر لوکاس[3](موقعیت در کروموزوم) مورد جهش قرار دهید.

پذیرفتن: فرزندان جدید را در جمعیت جدید بگنجانید.

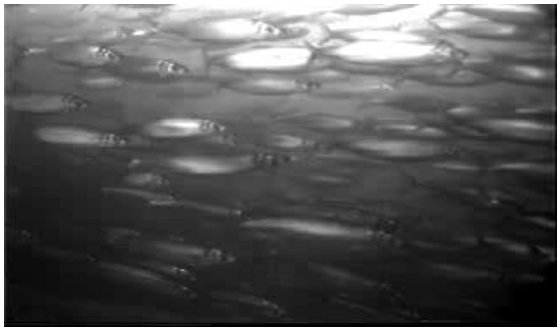
جایگزینی: جمعیت جدید ایجاد شده را برای روند الگوریتم بکار ببرید.

۱-۶-۴ کاربردها

- طراحی خودرو
- طراحی مهندسی پردازش تصویر
- رباتیک
- سخت افزارهای قابل تکامل
- مسیریابی مخابرات تلفنی بهینه شده (مسیریابی مخابرات تلفنی از مسافت)

1- Probability Crossover
2- Offspring
3- Lockus

۱-۷ الگوریتم ازدحام ذرات[1]



شکل 1-16: الگوریتم اجتماع ذرات

در این بخش به طور خلاصه به معرفی الگوریتم اجتماع ذرات که اختصارا PSO نامیده شده و تحت نام‌های مختلفی همچون **الگوریتم انبوه ذرات** ، **الگوریتم ازدحام ذرات** و **الگوریتم پرندگان** در ایران شناخته شده است، می‌پردازیم.

PSO یک الگوریتم کامپیوتری مبتنی بر جمعیت و کتره‌ای برای حل مساله است. PSO یک نوع هوش مبتنی بر اصول روانشناسی، اجتماعی و بینشی در رفتار اجتماعی و کمک کردن به کاربردهای مهندسی است.

الگوریتمPSO برای اولین بار در سال ۱۹۷۵ توسط جیمز کندی و ابرهارت توصیف شد. این تکنیک‌ها بسیار رشد کرده‌اند و نسخه‌ی اصلی این الگوریتم به طور واضحی در نسخه‌های امروزی قابل شناخت است این الگوریتم یکی از الگوریتم‌های قدرتمند و پر طرفدار برای بهینه سازی است که بیشتر به خاطر سرعت همگرایی نسبتا بالایی که دارد، مورد استفاده قرار می گیرد. این الگوریتم با وجود عمر کمی که دارد، اما توانسته است در حوزه‌های کاربردی بسیاری، از الگوریتم‌های قدیمی‌تر، مانند الگوریتم ژنتیک پیشی بگیرد.

عبارت Swarm در زبان انگلیسی به اجتماع دسته‌ی انبوهی از جانوران و حشرات اشاره می‌کند.

در زیر یک swarm از زنبورها را می‌بینید.

1- Particle Swarm Optimization

شکل 1-17: swarm زنبورها

همان‌طور که پیش‌تر گفته شد ایده‌ی Particle Swarm Optimization، برای اولین بار توسط کندی و ابرهارت در سال ۱۹۹۵ مطرح شد .PSO ، یک الگوریتم محاسبه‌ای تکاملی الهام گرفته از طبیعت و براساس تکرار می‌باشد. منبع الهام این الگوریتم، رفتار اجتماعی حیوانات، همانند حرکت دسته جمعی پرندگان و ماهی‌ها بود. از این جهت که PSO نیز با یک ماتریس جمعیت تصادفی اولیه، شروع می‌شود، شبیه بسیاری دیگر از الگوریتم‌های تکاملی همچون الگوریتم ژنتیک و الگوریتم رقابت استعماری است.
برخلاف الگوریتم ژنتیک، PSO هیچ عملگر تکاملی همانند جهش و تزویج ندارد. از این جهت می شود گفت که الگوریتم رقابت استعماری شباهت بیشتری به PSO دارد تا به GA. همچنین الگوریتم PSO همانند الگوریتم رقابت استعماری ابتدا برای حل مسائل پیوسته معرفی شد(نسخه‌های گسسته هر دو الگوریتم بعد از معرفی نسخه پیوسته معرفی شدند).

در این الگوریتم هر عنصر جمعیت، یک ذره نامیده می‌شود (که همان معادل کروموزوم در GA و یا کشور در الگوریتم رقابت استعماری) است. در واقع الگوریتم PSO از تعداد مشخصی از ذرات تشکیل می‌شود که به طور تصادفی، مقدار اولیه می‌گیرند. برای هر ذره دو مقدار وضعیت و سرعت، تعریف می‌شود که به ترتیب با یک بردار مکان و یک بردار سرعت، مدل می‌شوند. این ذرات، به صورت تکرارشونده‌ای در فضای n بعدی مساله حرکت می‌کنند تا با محاسبه‌ی مقدار بهینگی به عنوان یک ملاک سنجش، گزینه‌های ممکن جدید را جستجو کنند. بُعد فضای مساله، برابر تعداد پارامترهای موجود در تابع مورد نظر برای بهین ه‌سازی می‌باشد.

این ذرات حین حرکت در این ابر فضا دو توانایی ضروری نیز از خود نشان می‌دهند:

۱ حافظه‌ای برای ذخیره سازی بهترین موقعیت خود

۲ حافظه‌ای برای ذخیره‌ی بهترین موقعیت پیش آمده در میان همه‌ی ذرات

با تجربه‌ی حاصل از این حافظه‌ها هر ذره برای اعمال تغییری مناسب اطلاعات زیر را دارا می‌باشد:
"بهترین عمومی" که برای همه شناخته شده است وهنگامی که هر ذره بهترین مکان جدیدی را شناسایی کند، فورا برای بقیه‌ی ذرات، اطلاعات مربوطه را به روز رسانی می‌کند.
"بهترین همسایگی" که ذره، از طریق ارتباط با زیر مجموعه‌های گروه آن را به دست می‌آورد.
"بهترین محلی" که بهترین راه حلی است که ذره تاکنون تجربه کرده است.
با استفاده از اطلاعات به دست آمده، ذرات تصمیم می‌گیرند که در نوبت بعدی، چگونه حرکت کنند. در هر بار تکرار، هم‌ه‌ی ذرات در فضای n بعدی مساله حرکت می‌کنند تا بالاخره نقط ه‌ی بهین‌ه‌ی عام، پیدا شود. ذرات، سرعت‌هایشان و موقعیت‌شان را بر حسب بهترین جواب‌های مطلق و محلی به‌روز می‌کنند. یعنی

(4_1)

$$p_{m,n}^{new} = p_{m,n}^{old} + v_{m,n}^{new}$$

که در آن

- $v_{m,n}$، سرعت ذره
- $p_{m,n}$، متغیرهای ذره
- r_1, r_2، اعداد تصادفی مستقل با توزیع یکنواخت
- Γ_1, Γ_2، فاکتورهای یادگیری
- $p_{m,n}^{localbest}$، بهترین جواب محلی
- $p_{m,n}^{globalbest}$، بهترین جواب مطلق

می‌باشند. الگوریتمPSO ، بردار سرعت هر ذره را به‌روز کرده و سپس مقدار سرعت جدید را به موقعیت و یا مقدار ذره می‌افزاید. به‌روز کردن‌های سرعت، تحت تأثیر هر دو مقدار بهترین جواب محلی و بهترین جواب مطلق قرار می‌گیرند. بهترین جواب محلی و بهترین جواب مطلق، بهترین جواب هایی هستند که تا لحظه‌ی جاری اجرای الگوریتم، به ترتیب توسط یک ذره و در کل جمعیت به دست آمده‌اند. ثابت‌های Γ_1 و Γ_2 به ترتیب، پارامتر ادراکی و پارامتر اجتماعی نامیده می‌شوند. مزیت اصلی PSO این است که پیاده‌سازی این الگوریتم ساده بوده و نیاز به تعیین پارامترهای کمی‌دارد. همچنین PSO قادر به بهینه‌سازی توابع هزینه‌ی پیچیده با تعداد زیاد مینیمم محلی است.

۱ـ۷ـ۱ کاربردها

- طراحی بهینه‌ی شبکه‌های مانیتورینگ تغییر شکل
- پیشگویی مقادیر مفقود شده با استفاده از بهینه‌سازی گروه ذرات مشارکتی
- بهره برداری بهینه از ایستگاه‌های پمپاژ متوالی
- طراحی بهینه‌ی حجم مخازن سدها
- بهینه‌سازی سازه‌ها و کاربردهای این روش در مهندسی عمران
- بهینه‌سازی چند هدفه و بهره برداری از مخازن سدها
- شبیه سازی تابع تقاضای انرژی
- طبقه بندی معنایی تصاویر
- تعیین ذخیره‌ی توان راکتیو شبکه‌های برق
- الگوریتمی جهت یافتن بهینه‌ی سراسری در مسائل پیچیده

۱ـ۸ کدام الگوریتم بهتر است؟

شکل 1-18: کدام الگوریتم؟

اینکه ادعا کنیم یک الگوریتم بهینه سازی بهترین روش حل برای مسائل بهینه سازی است چندان صحیح نیست. چرا که هر ساله الگوریتم های جدیدتری با پوشاندن نقاط ضعف روش های قبلی پا به عرصه ی ظهور می گذارند.

زمانی GA و بعد ها PSO در نوع خود بهترین الگوریتم بهینه سازی محسوب می شدند و هنوز هم شاهد استفاده از آن ها در تعداد زیادی از مقالات هستیم. این روش ها چنان جای خود را در مسائل بهینه‌سازی باز کرده‌اند که به عنوان معیارهایی برای مقایسه‌ی عملکرد در روش‌های جدید می‌باشند.

بعضی از اساتید معتقدند که هیچ دلیلی وجود ندارد که یک روش بهینه سازی تکاملی بهتر از دیگری باشد! چون هر روش به دلیل وجود تکامل(Evolution) باید قادر به یافتن جواب بهینه باشد. نکته ای که در این بین وجود دارد این است که برخی از روش ها همانند مدل های واقعیشان در طبیعت به کندی دچار تکامل می شوند و معادل‌های برنامه‌نویسی شده‌ی آن‌ها هم برای رسیدن به جواب بهینه باید تعداد تکرارهای این الگوریتم ها را هم زیاد در نظر گرفت. مثلا در تکامل ژنتیکی که الهام از ژن های واقعی است تغییر و تکامل به کندی و در طول سال های زیاد صورت می گیرد. شاید این دلیلی باشد که GA نیاز به تعداد تکرارهای بیشتری برای پیدا کردن جواب بهینه داشته باشد. در مقابل الگوریتم PSO به دلیل الهام از گروه پرندگان که برای تغذیه‌ی خود بسیار سریع‌تر عمل می‌کنند، نسبت به GA در حل مسائل نیز سریع تر عمل می‌کند.

به هرحال دلیل واقعی هرچه باشد، آنچه در عمل دیده می شود این است که تعدادی از الگوریتم ها ذاتا سریع‌تر از بقیه عمل می کنند و این امر باعث تمایل بیشتر کاربران به استفاده از الگوریتم های بهینه‌سازی تکاملی سریع‌تر و دقیق‌تر می شود.

این مساله در مورد الگوریتم COA هم صادق است و در کاربردهای تست شده تاکنون این الگوریتم بسیار بهتر از بقیه الگوریتم ها عمل کرده است. این امر می تواند انگیزه ی خوبی برای استفاده از COA در حل بسیاری از مسائل با ابعاد بالا و بسیار پیچیده باشد.

فصل دوم

الگوریتم زنبور عسل(BeeAlgorithm)

شکل2-1: هديه اي از جانب خدا

" و پروردگارت به زنبورعسل وحی کرد: «در کوه ها و درخت ها و از داربست های انگور برای خودت لانه بساز. پس از تمام میوه‌های خداداده تناول کن و راه های پروردگارت_که هموار پدیدار فرموده _ بپیما. . . » به واقع در این پدیده برای کسانی که به تفکر قیام کرده اند، نشانه‌هایی است. "

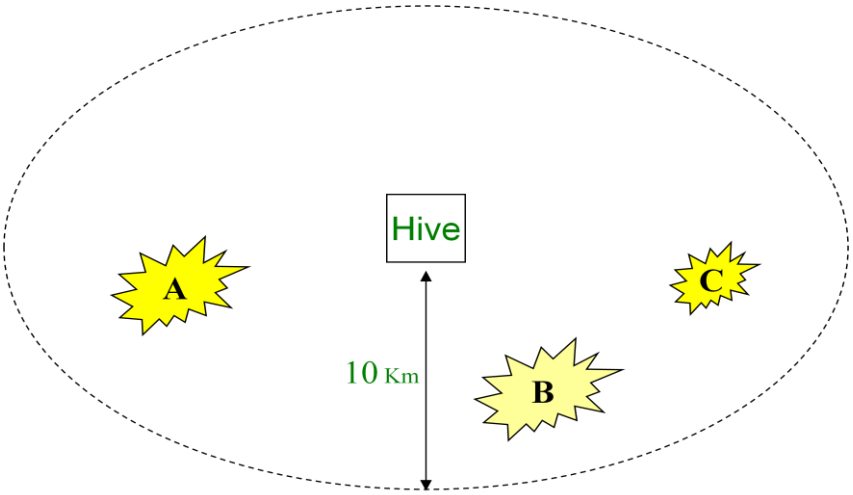
سوره نحل آیات ۶۸و۶۹

۲-۱ تعریف

الگوریتم زنبور شامل گروهی مبتنی بر الگوریتم جستجو است که اولین بار در سال ۲۰۰۵ توسعه یافت؛ این الگوریتم شبیه سازی رفتار جستجوی غذای گروه های زنبور عسل است. در نسخه‌ی ابتدایی این الگوریتم، الگوریتم نوعی از جستجوی محلی انجام می‌دهد که با جستجوی تصادفی {Random} ترکیب شده و می‌تواند برای بهینه سازی ترکیبی {زمانی که بخواهیم چند متغیر را همزمان بهینه کنیم.} یا بهینه‌سازی تابعی به کار رود.

۲-۲ کلونی زنبورها

یک **کلونی** زنبور عسل می‌تواند در مسافت زیادی و نیز در جهت‌های گوناگون پخش شود تا از منابع غذایی بهره برداری کند:



شکل2-2: تلاش برای یافتن قطعات گلدار

قطعات گلدار با مقادیر زیادی نکتار و گرده که با تلاشی کم قابل جمع آوری است، به وسیله‌ی تعداد زیادی زنبور بازدید می‌شود؛ به طوری که قطعاتی از زمین که گرده یا نکتار کمتری دارد، تعداد کمتری زنبور را جلب می‌کند.

کلونی زنبورها در طبیعت شامل سه بخش منابع غذایی، زنبورهای کارگر و زنبورهای غیرکارگر می‌باشد. زنبورهای کارگر صرفا با منبع غذایی که در حال حاضر مشغول استخراج شهد از آن می‌باشند، در ارتباط هستند. به علاوه، این زنبورها اطلاعاتی نظیر فاصله، جهت و میزان سودبخشی منبع را با خود حمل می‌کنند و در کندو این اطلاعات را با دیگران به اشتراک می‌گذارند. در حالی که زنبورهای غیرکارگر همواره به دنبال منابع غذایی جهت استخراج شهد آنان می‌باشند.

زنبورهای غیرکارگر به دو گروه عمده‌ی پیش‌آهنگان [1] و تماشاگران [2] تقسیم می‌شوند. پیش‌آهنگان محیط پیرامون را برای یافتن منابع غذایی جدید کاوش می‌کنند و تماشاگران در کندو منتظر می‌مانند و اطلاعات را از زنبورهای کارگر دریافت می‌کنند.

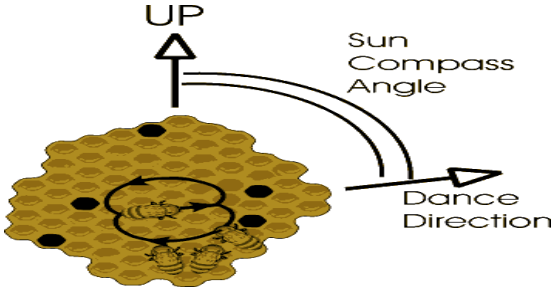
زنبورهای عسل از یک سیستم ارتباطی پیچیده استفاده می‌کنند. این سیستم آن‌ها را قادر می‌سازد اطلاعاتی در مورد محل و کیفیت منابع غذایی موجود در خارج از کندو به دست آورند. ارتباط بین زنبورها توسط زبان رقص انجام می‌گیرد. زبان رقص شامل مجموعه ای از حرکات پشت سرهم است که توسط

1- Scouts
1- On- Lookers

زنبورها انجام می‌گیرد. این رقص که رقص چرخشی [1] نام دارد، حاوی اطلاعاتی در مورد کیفیت منبع، مکان و موقعیت آن می‌باشد. در این رقص، تعداد چرخش نمایان‌گر فاصله و مدت زمان چرخش نشان دهنده‌ی کیفیت منبع غذایی است. زنبور تماشاگر می‌تواند تعداد زیادی رقص را مشاهده کند و یکی از منابع غذایی را انتخاب نماید. با توجه به مدت زمان چرخش، احتمال انتخاب منابع غذایی با کیفیت بیشتر بالاتر خواهد بود.

۳-۲ جستجوی غذا در طبیعت

پروسه‌ی جستجوی غذای یک **کلونی** به وسیله ی زنبورهای دیده‌بان آغاز می‌شود که برای جستجوی گلزارهای امیدبخش{دارای امید بالا برای وجود نکتار یا گرده} فرستاده می‌شوند.
زنبورهای دیده‌بان به صورت تصادفی{Random} ازگلزاری به گلزار دیگر حرکت می‌کنند. زنبورها برای پرواز، به انرژی زیادی نیاز دارند. بنابراین، آن‌ها سعی می‌کنند کوتاه‌ترین و بهترین راه را در بین شبکه‌ای از گل‌ها پیدا کنند و چیزی که به آن‌ها کمک می‌کند تا راه خانه را پیدا کند، نور آفتاب است .
در طول فصل برداشت محصول{گل دهی}، **کلونی** با آماده نگه داشتن تعدادی از جمعیت **کلونی** به عنوان زنبور دیده‌بان به جستجوی خود ادامه می‌دهد. هنگامی که جستجوی تمام گلزارها پایان یافت، هر زنبور دیده‌بان، بالای گلزاری که اندوخته‌ی کیفی مطمئنی از نکتار و گرده دارد، رقص خاصی را اجرا می‌کند.
این رقص که به نام "رقص چرخشی"{حرکتی مانند حرکت قرقره} شناخته می‌شود، اطلاعات مربوط به جهت تکه گلزار{نسبت به کندو}، فاصله تا گلزار و کیفیت گلزار را به زنبورهای دیگر انتقال می‌دهد. این اطلاعات زنبورهای اضافی و پیرو را به سوی گلزار می‌فرستد.



شکل2-3 : رقص چرخشي

بیشتر زنبورهای پیرو به سوی گلزارهایی می روند که امید بخش تر هستند و امید بیشتری برای یافتن نکتار و گرده در آن‌ها وجود دارد.

2- Waggle dance

وقتی همه‌ی زنبورها به سمت ناحیه ای مشابه بروند، دوباره به صورت تصادفی {Random} و به علت محدوده‌ی رقصشان در پیرامون گلزار پراکنده می‌شوند تا به موجب این کار سرانجام نه یک گلزار، بلکه بهترین گل‌های موجود درون آن نیز تعیین موقعیت شوند.

۲-۴ الگوریتم کلونی زنبورهای مصنوعی

الگوریتم کلونی زنبورهای مصنوعی (ABC) الهام گرفته از رفتار زنبورها در طبیعت می‌باشد. مشابه با کلونی زنبورهای طبیعی، این الگوریتم نیز از سه گروه زنبورهای کارگر، تماشاگر و پیش‌آهنگ تشکیل شده است.

در ابتدا مجموعه‌ای از منابع غذایی به طور تصادفی انتخاب می‌شوند. زنبورهای کارگر به منابع مراجعه کرده و میزان شهد آنها را محاسبه می‌کنند. سپس این زنبورها به کندو بازگشته و اطلاعات خود را با دیگر زنبورها(تماشاگران) به اشتراک می‌گذارند. در مرحله‌ی دوم بعد از تبادل اطلاعات، هر زنبور کارگر به سمت منبعی می‌رود که قبلا دیده است و ممکن است براساس اطلاعات دیداری که از محیط می‌گیرد یک منبع جدید در همسایگی منبع قبلی انتخاب نماید. بدین معنی که زنبور، با توجه به رنگ و نوع گل تصمیم می‌گیرد که به همان منبع قبلی برود و یا منبع جدیدی را انتخاب نماید. در مرحله‌ی سوم ، تماشاگران با توجه به اطلاعاتی که از زنبورهای کارگر هنگام رقص دریافت کرده‌اند یک محدوده منبع غذایی را بر مبنای شهد آن ترجیح می‌دهند. بعد از رسیدن به محل ممکن است با توجه به اطلاعات دیداری، یک منبع جدید را که در همان اطراف قراردارد انتخاب کنند. زمانی که یک منبع پایان پذیرد یا ترک شود یک منبع جدید که به طور تصادفی توسط پیش‌آهنگان دریافت شده است، جایگزین می‌شود. این چرخه تا برآورده شدن نیازها تکرار خواهد شد. در این مدل در هر چرخه حداکثر یک پیش‌آهنگ وجود دارد و تعداد زنبورهای کارگر و تماشاگر برابر است.

همان‌طور که گفته شد هر یک از زنبورهای کارگر و تماشاگران ممکن است تغیراتی بر روی موقعیت منبع غذایی(راه حل) در حافظه‌ی خود ایجاد کنند و شایستگی آن را محاسبه کرده، در صورتی که میزان شایستگی آن از راه حل قدیمی بیشتر باشد، راه حل جدید انتخاب می‌شود و راه حل قدیمی فراموش می‌شود، در غیر این صورت همان راه حل قبلی باقی خواهد ماند. این تغیرات توسط رابطه زیر به دست می‌آید:

(1_2)

$$v_{ij} = x_{ij} + \phi_{ij}\left(x_{ij} - x_{kj}\right)$$
$$i \neq k, k \in \{1,2,..,BN\} \quad , j \in \{1,2,..,D\}, \phi_{ij} \in [-1\ 1]$$

در این رابطه، ϕ_{ij} یک عدد تصادفی در بازه [1 -1] است. این متغیر ، تولید موقعیت منابع غذایی همسایه، در اطراف $x_{i,j}$ را کنترل می‌کند. در این رابطه BN تعداد زنبورهای کارگر می‌باشد و متغیر K به صورت تصادفی تولید می‌گردد و با i متفاوت خواهد بود.

براساس این رابطه، هر چه تفاوت بین $x_{i,j}$ و $x_{i,k}$ کاهش یابد، انحراف از موقعیت $x_{i,j}$ نیز کاهش خواهد یافت. درحقیقت در این رابطه سعی می‌کنیم یک بعد از ابعاد یکی از موقعیت‌ها را انتخاب کرده و با توجه به میزان ϕ به سمت آن و یا در خلاف جهت آن حرکت کنیم؛ همانند الگوریتم جمعی ذرات (PSO)، با این تفاوت که در اینجا با انتخاب تصادفی سعی می‌کنیم تا حدودی ایجاد تنوع نموده و از قرارگرفتن در بهینه‌ی محلی جلوگیری نماییم.

بعد از اتمام فرایند جستجو، تماشاگران اطلاعات هرکدام از زنبورهای کارگر را ارزیابی می‌کنند و با یک احتمال که متناسب است با میزان کیفیت شهد منبع، یکی از منابع غذایی را انتخاب می‌کنند. این احتمال از رابطه‌ی زیر به دست می‌آید:

در این رابطه fit_i، میزان شایستگی منبع غذایی متناظر با زنبور iام و SN تعداد راه حل‌های موجود می‌باشد.

(2_2)

$$P_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n}$$

در صورتی که یک منبع پایان پذیرد و یا کیفیت یک منبع غذایی مناسب نباشد، زنبور کارگر آن‌ها را رها کرده و به یک پیش‌آهنگ تبدیل می‌شود. این رفتار بدین صورت مدل می‌گردد که اگر شایستگی یک نقطه بعد از چندین تکرار(که تعداد آن را با پارامتر limit نمایش می‌دهیم) بهبود نیابد، بدین معنی است که در یک بهینه‌ی محلی قرار داریم بنابراین آن نقطه حذف می‌شود و یک نقطه جدید به صورت تصادفی تولید می‌گردد.

۲-۴-۱ بهینه‌سازی کلونی زنبورها

سیستم های طبیعی مختلفی به ما یاد می دهند که ارگانیسم‌های خارجی بسیار ساده توان تولید سیستم‌هایی با قابلیت انجام کارهای بسیار پیچیده به کمک برهم کنشهای پویا با هم را دارند. متاهیوریستیک(ابرکشف (کلونی زنبورها (BCO)) در این قسمت از مقاله آورده شده است. کلونی مصنوعی زنبورها در پاره‌ای نزدیک به هم و در مقایسه با کلونی زنبورهای طبیعی متفاوت عمل می‌کنند.

BCO به همان میزان که قابلیت حل مسائل ترکیبی قطعی را دارد، قادر به حل مسائل ترکیبی است که دارای عدم قطعیت نیز می‌باشند. .

توسعه‌ی الگوریتم کشف کننده ی جدید برای حل مسأله‌ی *Ride-Matching* به کمک راه پیشنهاد شده(استفاده از کلونی زنبورها) راهی روشن‌گر برای نشان دادن قابلیت های این روش محسوب می شود.(در فصل‌های بعد، این مساله به تفصیل مورد بررسی قرار خواهد گرفت.)

۲-۴-۲ معرفی کلونی زنبورهای مصنوعی

الگوریتم‌های غیرمبتنی بر فرومون عموما الهام گرفته از رفتار زنبورها هستند. این الگوریتم‌ها از فرومون برای جهت‌یابی در یک محیط ناشناخته استفاده نمی‌کنند، بلکه به منظور جهت‌یابی، ازیک استراتژی مستقیم به نام یکپارچه‌سازی مسیر استفاده می‌کنند.

شمار زیادی از مدل های مهندسی و الگوریتم هایی که برای حل مسائل پیچیده به کار می‌رود بر اساس کنترل و مرکزگرایی بنا شده اند. برخی از سیستم های طبیعی(کلونی‌های حشرات اجتماعی) به ما یاد می‌دهند که یک سری ارگانیسم های ساده‌ی خارجی قابلیت تولید سیستم‌هایی را دارند که به کمک بر هم کنش‌های پویا قابلیت انجام اعمال بسیار پیچیده را دارند.

گروه زنبورها به خاطر استقلال داخلی کلونی و عملکردهای توزیع شده و سیستم درون سازمانی یکی از بهترین کلونی‌ها برای توضیح این مساله شناخته شده است.

در سال‌های اخیر محققان برای تولید سیستم‌های جدید مصنوعی(در حیطه‌ی هوش مصنوعی) شروع به تحقیق درباره‌ی طرز رفتار حشرات اجتماعی کرده‌اند.

BCO (Bee Colony Optimization) که مسیر جدیدی را در هوش جمعی بررسی می‌کند در این قسمت بررسی شده است. هدف اصلی این قسمت بررسی این امکان است که به کمک سیستم مصنوعی زنبورها بتوان قدمی را در پیدا کردن راه حل هایی جامع برای حل مسائلی که با عدم قطعیت مواجه هستند برداشت.

The Bee Colony Optimization : The New Computational Paradigm

Bees In Nature

سیستم سازمانی زنبورها بر اساس یک سری قوائد ساده ی خارجی حشرات بنا شده است. با اینکه نژادهای بسیاری از حشرات مختلف بر روی کره‌ی زمین موجود هستند و همین باعث تفاوت هایی در الگوی

رفتاری آن ها می‌شود ولی با این حال این سری حشرات اجتماعی را می توان دارای قابلیت حل مسائل پیچیده دانست. بهترین مثال برای این حالت روند تولید نکتار(شهد) محسوب می شود که در نوع خود یک فرایند سازماندهی شده‌ی پیشرفته محسوب می‌شود. هر زنبور ترجیح می دهد که راه قبلی زنبور هم کندوی خود را دنبال کند تا اینکه خود به دنبال گل جدید بگردد.

هر کندوی زنبور عسل دارای مکانی معروف به سالن رقص است که در آنجا زنبورها با انجام حرکتی خاص، هم کندویی‌های خود را راضی می کنند تا راه آن ها را برای رسیدن به گل ها برگزینند. اگر یک زنبور تصمیم بگیرد که به دنبال نکتار برود با انتخاب زنبور هم کندوی رقصنده‌ی خود، راه قبلی را دنبال می کند تا به گل برسد. هر زنبور پس از رسیدن به گل‌ها و جمع‌آوری شهد قادر به انجام یکی از کارهای زیر است:

الف: منبع غذا را رها کند و دوباره به دنبال زنبور رقصانی بگردد تا بتواند منبعی جدید پیدا کند.

ب: خود به دنبال منابع غذایی جدید بگردد.

ج: در کندو اقدام به رقصیدن کرده و زنبورهای جدیدی را به دنبال خود بکشاند.

بر اساس احتمالات اندازه گیری شده، زنبور اقدام به انجام یکی از حالات بالا می‌کند. در مکان رقص، زنبورها اقدام به پیشنهاد مکان های مربوط به جمع‌آوری نکتار به دیگران می کنند. مکانیزم انتخاب یک زنبور توسط زنبوری دیگر هنوز شناخته شده نیست ولی تا به امروز روشن شده است که این امر بیشتر مربوط به کیفیت نکتار پیدا شده توسط زنبور رقصنده است.

لوسیچ و تدورووپچ اولین کسانی بودند که از رویه‌های پایه و ساده ی زنبوری برای حل کردن مسائل ترکیبی بهینه‌سازی استفاده کردند. آن‌ها سیستم زنبوری *(BS)* را معرفی کردند و از آن برای حل مساله ی معروف *Travelling Salesman* استفاده کردند. در ادامه به استفاده‌های *BCO* در حل مسائل پیشرفته اشاره خواهیم کرد.

در کلونی مصنوعی طراحی شده توسط ما شباهت ها و تفاوت‌هایی با کلونی‌های واقعی زنبورها در طبیعت وجود دارد. در ادامه به معرفی *FBS (Fuzzy Bee System)* می‌پردازیم که قادر به حل مسائل ترکیبی (طرح شده توسط انسان ها) است. به کمک *FBS*، *Agent*ها در ارتباطات با همدیگر از قوانین تقریبی دلیل‌گرایی و منطق *Fuzzy* استفاده می‌کنند.

The Bee Colony Optimization Metaheuristic

در *BCO* مامورهایی که ما به آن ها "زنبور مصنوعی" می‌گوییم با همدیگر اجتماع می کنند تا بتوانند قادر به حل مسائل مشکل تر باشند. تمامی زنبورهای مصنوعی در ابتدای فرایند جستجو ، در کندوی اصلی قرار دارند. در فرایند جستجو نیز زنبورهای مصنوعی به طور کاملا مستقیم با یکدیگر ارتباط برقرار می کنند. هر زنبور مصنوعی یک سری حرکات محلی خاص انجام داده و به کمک آنها قادر خواهد بود تا راه حلی را برای مشکل فعلی خود پیدا کند.

این زنبورها تک تک راه حل‌های کمکی و زیرپایه‌ای را ارائه می‌دهند تا در آخر با ادغام این راه حل‌ها راه حل اصلی برای حل مساله‌ی ترکیبی به دست بیاید.

روند جستجو از تکرارهای پشت سر هم تشکیل شده است. اولین تکرار زمانی پایان می‌یابد که اولین زنبور راه حل زیر پایه‌ی خود را برای حل مساله‌ی اصلی ارائه دهد.

بهترین راه حل زیرپایه در خلال اولین تکرار انتخاب شده و پس از آن تکرار دوم شروع خواهد شد. در تکرار دوم زنبورهای مصنوعی شروع به پیدا کردن راه حلی جدید برای مساله‌ی زیر پایه می‌کنند و... .

در پایان هر تکرار حداقل یک و یا چند راه حل ارائه شده و جود دارد که آنالیست مقدار همگی آنها را مشخصی می‌کند.

به هنگام حرکت در فضا، زنبورهای مصنوعی ما یکی از دو حرکت "حرکت به سمت جلو" و یا "حرکت به سمت عقب" را انجام می‌دهند.

به هنگام "حرکت به سمت جلو" زنبورها راه و روش‌های جدیدی را برای حل مساله پیدا می‌کنند. آن‌ها این کار را به کمک یک سری جستجوهای شخصی و اطلاعات به دست آمده‌ی گذشته انجام می‌دهند.

بعد از آن، زنبورها عمل "حرکت به سمت عقب" را انجام می‌دهند که همان برگشتن به کندوی اصلی است.

در کندو همگی زنبورها در یک فرایند "تصمیم گیری" شرکت می‌کنند. ما در نظر می گیریم که هر زنبوری قابلیت درک و دریافت اطلاعات زنبورهای دیگر را بر اساس کیفیت دارد. به کمک این روش ، زنبورها این قابلیت را دارند که با استفاده از اطلاعات دیگران، راه‌های بهتر حل مساله را پیدا کنند.

براساس اطلاعات جدیدی که در مورد کیفیت راه حل به دست می‌آید، زنبور می‌تواند تصمیم بگیرد که

الف) منبع راه حل خود را رها کرده و در سالن رقص به دنبال کسی بگردد که منبعی با کیفیت بیشتر در اختیار دارد.

ب) بدون اینکه کسی را جذب کند دوباره به سراغ منبع راه حل خود برود.

ج) در سالن رقص با انجام حرکاتی خاص(رقصیدن) سعی در جمع کردن زنبورهای دیگر به دور خود داشته باشد.

بر اساس میزان کیفیتی که زنبور از منبع خود به دست می‌آورد ، فاکتوری به نام "وفاداری" در وی به وجود می‌آید که در واقع همان وفاداری به راهی است که خود زنبور انتخاب کرده است. بار دومی‌که زنبورهای مصنوعی برای پیدا کودن راه حل مساله به حرکت در می‌آیند این بار سعی در پیدا کردن راه های جدیدی برای حل مساله دارند و بعد از این کار ، دوباره عمل "حرکت به سمت عقب" را انجام داده و به کندو برمی‌گردند و دوباره در کندو در بحثی که در مورد پیدا کردن بهترین راه شکل گرفته شرکت می‌کنند.

این روند زمانی پایان می یابد که یک راه حل تقریبا کامل برای مساله پیدا شود.
مثل برنامه نویسی پویا، *BCO* نیز می‌تواند مسائل ترکیبی بهینه‌سازی را در هر مرحله(تکرار) به میزانی حل کند. هر کدام از مراحل مشخص شده، دارای یک مقدار بهینه‌سازی خاص است. اجازه دهید اشاره کنیم که (3_2)

$$ST=\{st1 + st2 + \dots + stm\}$$

همان‌طور که می‌بینید هر(*Stage* مرحله) شامل یک سری مراحل از قبل انتخاب شده است. در ادامه می‌بینید که به کمک کمیت *B* ما تعداد زنبورهایی را که در این فرایند شرکت می‌کنند را مشخص می‌کنیم و به کمک *I* تعداد کل مراحل (تکرار)هایی را که انجام می‌پذیرند را نشان می‌دهیم. مجموعه‌ی تمامی راه حل‌های زیرپایه را نیز به کمک *Sj* نشان می‌دهیم که در آن *j* دارای مقادیر ۱ تا *m* میباشد.

۲ـ۵ شبه کد

الف) شروع : مشخص کردن تعداد زنبورها *(B)* و تعداد تکرارها *(I)* مشخص نمودن تعداد مراحل *(ST)* . پیدا کردن هر گونه راه حل قابل حل *x* از مساله.

این راه حل در واقع بهترین و اولین راه حل انتخاب توسط ما خواهد بود.

ب) *Set i:=1 , Until i=I* (و تکرار کن مراحل بعدی را)

ج) *Set j:=m , Until j=m* (و تکرار کن مراحل بعدی را)

حرکت به سمت جلو(رفت) : به زنبورها این امکان را می‌دهد که از کندو بیرون آمده و قابلیت انتخاب *B* راه حل را از مجموعه‌ی راه حل‌های زیرپایه *Sj* در *STj* داشته باشند.

حرکت به سمت عقب(برگشت) : تمامی زنبورها را به کندو برمی‌گرداند. به زنبورها این اجازه را می‌دهد که اطلاعات خود را در مورد کیفیت راه حل های دیگران و خود به اشتراک بگذارند و بدین طریق تصمیم بگیرند که منبع خود را رها کرده یا به دنبال کسی دیگر بیفتند یا به تنهایی به منبع خود برگردند و یا با رقصیدن دیگران را مشتاق به دنبال کردن منبع خود کنند. *Setj:=j+1*

د) اگر بهترین راه حلی *(Xi)* که در *I* امین تکرار به دست آمد، بهتر از بهترین راه اخیر به دست آمده بود آنگاه فاکتور بهترین راه حل را به روز میکنیم *X:=xi*

ه) *Set i:=i+*۱

به طور کل حرکت های جلویی و عقبی در *BCO* می‌توانند نقش فرعی را بگیرند به این معنی که تا زمانی که یکی از فاکتورهای مهم کامل نشده است این دو به کار خود ادامه دهند. این فاکتور مهم به عنوان مثال می‌تواند "بیشترین مقدار رفت و برگشت‌ها" و یا برخی دیگر از موارد مورد نظر توسط خود اپراتور باشد.

در *BCO* زیر مدل‌های مختلفی که به توصیف چگونگی حالات زنبورها می‌پردازد و یا منطق‌گرایی آن‌ها را مشخص می‌کند به راحتی قابلیت توسعه و تست شدن را دارد. به این معنی که الگوریتم‌های متفاوتی از *BCO* را می‌توان طراحی کرد.

این مدل ها می توانند به توصیف چگونگی ترک کردن منبع اولیه توسط زنبورها ، ادامه دادن رفت و برگشت بین کندو و منبع توسط زنبور و یا چگونگی رقصیدن زنبور برای جمع کردن دیگر زنبورها به دو ر خود را توضیح دهند.

The Fuzzy Bee System:

زنبورها در فرایند پیدا کردن بهترین راه حل با مشکلات تصمیم گیری مختلفی مواجه می شوند، مشکلات زیر برخی از مشکلات رایج بین آن‌هاست:

الف) راه حل زیرپایه‌ی بعدی که باید به راه حل اصلی اضافه شود چیست؟

ب) آیا باید راه حل زیرپایه‌ی فعلی را رها کرد و به دنبال راه حل زیرپایه‌ی جدیدی رفت؟

ج) آیا باید به گسترش راه حل زیرپایه‌ی فعلی ادامه داد ولی فعلا به دنبال دیگر زنبورها نرفت ؟

ب سیاری از مدل های تصمیم گیری بر اساس ابزارهای مدلینگ مختلفی به وجود آمده اند. این حالات کاملا منطقی و عقلی هستند و بر اساس این اطلاعات به وجود آمده‌اند که ماموران تصمیم گیر *(Decision Agents)* مامورانی با داشتن بیشترین اطلاعات هستند و همیشه بهترین راه حل را برای پایان دادن به حل مساله در نظر می گیرند. برای اینکه بتوان مدل های حل مساله‌ی مختلفی را به وجود آورد محققان شروع به استفاده از راه‌های بی قاعده‌تری کردند.

مفهوم ساده‌ی منطق فازی[1] که توسط "زاده" معرفی شد، قابلیت بهتری در توضیح مسائلی که با عدم قطعیت ادغام شده اند را دارد. با توجه به اطلاعات فوق ، ما در انتخاب اینکه منطق زنبورها بر چه اساسی صورت می‌گیرد از منطق فازی استفاده می کنیم. زنبورهای مصنوعی ما از منطق‌گرایی تقریبی و منطق فازی برای انجام اعمال خود استفاده می‌کنند.

به هنگام دادن راه حل های زیرپایه‌ی جدید به زنبور مصنوعی ، هرزنبور حالت‌های زیر را برای برقراری ارتباط با راه حل زیرپایه ی فوق در نظر می گیرد: "کم‌جاذبه"، "جذاب"، "خیلی‌جذاب". همچنین ما در نظر می گیریم که یک زنبور مصنوعی می تواند مقادیر خاصی را مانند "کوتاه" ، "متوسط" و "بلند" و یا "ارزان" ، "متوسط" و "گران" در نظر بگیرد.

1- Fuzzy logic

Calculating The Solution Component Attractiveness and Choice Of The Next : Solution Component To Be Added To The Partial Solution

الگوریتم منطق تقریبی برای حل کردن مساله‌ی جذابیت، از قوانین زیر تشکیل شده است:
اگر مقادیر بد دست آمده از راه حل زیر پایه خیلی خوب باشد آنگاه راه حل بد دست آمده خیلی جذاب است.

هدف و امتیاز اصلی استفاده از از این الگوریتم این است که حتی با وجود اینکه اطلاعات به دست آمده ممکن است فقط اطلاعات تقریبی باشند(و نه قطعی) ، می‌توان میزان جذابیت راه حل زیرپایه را به راحتی مشخص کرد. با در نظر گرفتن میزان جذابیت راه حل زیرپایه ی i به توضیح میزان احتمال وقوع می‌پردازیم:

احتمال برای راه حل زیر پایه ی i که به راه حل اصلی الحاق می‌شود برابر است با نسبت میزان جذابیت تقسیم بر تمامی‌جذابیت‌های راه حل‌های زیر پایه‌ی دیگر :

برای اضافه کردن راه حل‌های جدید به راه حل اصلی، زنبورها از نوعی انتخاب به نام *Roulette Wheel Selection*استفاده می‌کنند.

در توصیف مکانیزم مقایسه‌ی راه حل های زیر پایه‌ی زنبور ، ما موضوع "بدی راه حل زیرپایه " را نیز معرفی می کنیم که برابر است با:

- بدی راه حل زیرپایه به وسیله‌ی k امین زنبور
- مقادیر تابع مفعولی از راه حل زیرپایه‌ای که به وسیله‌ی n امین زنبور کشف شده
- مقادیر تابع مفعولی از بهترین راه حل زیرپایه‌ی کشف شده از ابتدای روند جستجو تاکنون
- مقادیر تابع مفعولی از بدترین راه حل زیرپایه‌ی کشف شده از ابتدای روند جستجو تاکنون

الگوریتم منطق تقریبی برای تعیین بدی راه حل زیرپایه از قوانینی به شکل زیر تشکیل شده است:

- اگر راه حل کشف شده بد بود، آنگاه وفاداری کم خواهد شد.

زنبورها از منطق تقریبی و مقایسه‌ی راه حل‌های زیرپایه‌ی کشف شده‌شان با بهترین راه حل زیرپایه و مقایسه‌ی راه حل‌های زیرپایه‌ی کشف شده با بدترین راه حل‌ها از آغاز روند جستجو استفاده می‌کند.

در این روش ، حقایق تاریخی که به وسیله‌ی تمامی اعضای کلونی زنبور به وجود آمده‌اند ، تاثیر قابل توجهی بر راههای آینده‌ی جستجو دارند.

Bee's Partial Solutions Coparison Mechanism

Bee's Decision About Recruiting The Nestmates

از زمان شروع زندگی زنبورها و یا بهتر ا ست بگوییم از زمان شروع زندگی حشرات اجتماعی ، احتمال رخدادی که در آن زنبور به پرواز در طول همان مسیر بدون گرفتن همراه ادامه دهد ، احتمال بسیار کمی‌است*(1>>)* .

زنبورها تا محل رقص بپرواز می‌کنند این نوع ارتباط بین زنبورها منجر به ساخته شدن فاکتوری به نام "هوش جمعی"[1] می‌شود.

در این حالت هنگامی‌که زنبور تصمیم می گیرد که همان مسیر را پرواز نکند ، آن زنبور به سالن رقص رفته و از دیگر زنبورها پیروی خواهد کرد.

Calculating The Number Of Bees Changing The Path

هر راه حل زیرپایه که در ناحیه ی رقص اعلان شده دو ویژگی اصلی داشته است:

الف) مقادیر تابع مفعولی

ب) تعداد زنبورهایی که آن راه حل زیرپایه را اعلان کرده‌اند

این تعداد، یک تعیین کننده ی خوب برای دانش دسته جمعی زنبورهاست. این فرایند نشان می دهد که چگونه کلونی زنبوری راه حل زیرپایه ی خاصی را در نظر می‌گیرد.

الگوریتم منطق تقریبی برای معین کردن جذابیت راه حل زیرپایه ی اعلان شده از قوانین زیر تشکیل شده است:

اگر طول راه اعلان شده کوتاه باشد و تعداد زنبورهای اعلان کننده کم باشد

آنگاه جذابیت راه حل زیرپایه متوسط است.

جذابیت محاسبه شده ی راه در این روش می تواند مقادیری بین ۰ و ۱ را اختیار کند. هر چقدر مقدار محاسبه زیادتر باشد، راه حل اعلان شده جذابیت بیشتری دارد. زنبورها کم و بیش به راه اولیه و قدیمی‌خود وفادارند، هم زمان راه‌های اعلان شده‌ی جدید ، جذابیت کم و بیشی برای آنها خواهد داشت.

الگوریتم منطق تقریبی، برای محاسبه‌ی تعداد زنبورهای جابجا شده شامل قوانینی به فرم زیر است:

اگر وفاداری زنبورها به راه پایین باشد و جذابیت راه بالا باشد آنگاه تعداد زنبورهای جابجا شده از راه به راه بعدی بالا است.

در این روش تعداد زنبورهایی که در طول یک مسیر خاص پرواز می‌کنند، قبل از رفت بعدی تغییر داده می‌شود.

با استفاده از دانش اجتماعی و به اشتراک گذاری اطلاعات، زنبورها بر مسیرهای تضمین شده‌ی جستجو تمرکز می‌کنند و مسیرهای کمتر تضمین شده را کم کم ترک می‌کنند.

1-Collective intelligence

۲ـ۶ الگوریتم بهینه‌یابی جفت‌گیری زنبورهای عسل *(HBMO)* در حل مسائل بهینه‌سازی

جفت‌گیری زنبورهای عسل نیز می‌تواند به عنوان یک روش عمومی بر پایه رفتار حشرات جهت بهینه‌سازی در نظر گرفته شود که در آن الگوریتم جستجو الهام گرفته از فرایند جفت‌گیری در زنبورهای واقعی می‌باشد. رفتار زنبورهای عسل تقابلی بین پتانسیل ژنتیک، محیط فیزیولوژیکی واکولوژیکی و شرایط اجتماعی کندو و یا ترکیبی از موارد فوق می‌باشد. زنبورهای عسل همچنین جهت مدل سازی سیستم‌های بر پایه‌ی اجزاء توسط پرز- یوریب و هیرسبرنر به کار گرفته شده‌اند.

۲-۶-۱ مدل‌سازی جفت‌گیری زنبورهای عسل

یک کندوی زنبور عسل به طور معمول شامل یک ملکه با طول عمر زیاد جهت تخم‌گذاری و تعدادی از صفر تا چندصد زنبورنر(با توجه به فصول) و حدود ۱۰۰۰۰تا ۶۰۰۰۰ زنبور کارگر می‌باشد. ملکه‌ها اصلی‌ترین نقش تولید مثل را در برخی گونه‌های زنبور عسل ایفا می‌کنند و وظیفه تخم‌گذاری را نیز بر عهده دارند. زنبورهای نر پدر کندو می‌باشند. آنها تک جنسی بوده و وظیفه‌ی تشدید ژن‌های مادران، بدون تغییر در ژنتیک آنها را برعهده دارند. وظیفه‌ی کارگرها بچه داری و در برخی موارد تخم‌گذاری می‌باشد. بچه‌ها از تخمهای بارور و نابارور حاصل می‌شوند، به گونه‌ای که از دسته‌ی اول ملکه و زنبورهای کارگر و از دسته دوم زنبورهای نر تولید می‌گردد.

از بین کلیه‌ی زنبورها، فقط ملکه توسط "ژله‌ی سلطنتی" تغذیه می‌شود. ژله‌ی سلطنتی یک ماده‌ی ژله مانند به رنگ سفید- شیری می‌باشد. زنبورهای پرستار این ماده‌ی مغذی را مخفی کرده و تنها جهت تغذیه‌ی ملکه مصرف می‌نمایند. تغذیه‌ی ملکه توسط این ژله، او را نسبت به بقیه‌ی زنبورها در کندو بزرگتر می‌سازد. ملکه، حدود ۵ تا ۶ سال عمر می‌کند در حالی که زنبورهای کارگر کمی بیش از ۶ ماه زندگی می‌نمایند. پرواز جفت‌گیری توسط رقص خاصی از جانب ملکه آغاز می‌گردد. در این پرواز زنبورهای نر به تعقیب ملکه پرداخته و در فضا جفت‌گیری با ملکه را انجام می‌دهند. در یک پرواز جفت‌گیری معمول، هر ملکه با ۷ تا ۲۰ زنبور نر جفت‌گیری می‌نماید. در هر جفت‌گیری اسپرم وارد محفظه‌ی اسپرم ملکه شده و در آنجا جمع‌آوری می‌گردد. هربار که ملکه تخم‌ریزی بارور انجام می‌دهد، مخلوطی از اسپرم جمع شده در محفظه‌ی اسپرم را جهت باروری تخم‌ها خارج می‌سازد.

در حین پرواز جفت‌گیری، ملکه توسط جمعیت انبوهی از زنبورهای نر تعقیب شده و سرانجام زنبورهای نری که موفق به جفت‌گیری با ملکه شوند خواهند مرد، ولی ملکه اسپرم آنها را دریافت می‌نماید. این بدان معناست که ملکه چندین بار و با چندین زنبور نر جفت‌گیری می‌نماید ولی زنبورهای نر تنها قادر به یک بار جفت‌گیری با ملکه می‌باشند. این عمل جفت‌گیری زنبورها را در قیاس با دیگر حشرات منحصر به فرد

می‌سازد. درواقع، پرواز جفت‌گیری می‌تواند به یک مجموعه جابه‌جایی در فضا و مکان(محیط) تشبیه شود که در آن ملکه در نقاط مختلف و با سرعت‌های متفاوت به پرواز درآمده و با زنبورهای نری که در آن لحظه و درآن مکان برخورد می‌نمایند به طور تصادفی جفت‌گیری می‌نماید.

بدیهی است که درآغاز پرواز جفت‌گیری، انرژی ملکه در حد مشخصی بوده و در انتهای مسیر یعنی در زمانی که ملکه به کندو باز می‌گردد انرژی او کاسته شده و نزدیک به صفر می‌شود. از طرف دیگر ممکن است که قبل از به صفر رسیدن انرژی ملکه، حجم محفظه‌ی اسپرم ملکه پرشده و ملکه حتی در صورت دارا بودن انرژی نیز به کندو بازگردد.

در طبیعت نقش کارگرها محدود به بچه داری و تغذیه ملکه می‌باشد، بنابراین در الگوریتم توسعه یافته، هر کارگر به عنوان یک رفتار و تابع کاوشی جهت ترقی نسل و یا مراقبت از یک مجموعه از بچه‌ها عمل می‌نماید. هر زنبور نر به طور احتمالاتی توسط تابع نورد زیر جفت‌گیری می‌نماید:

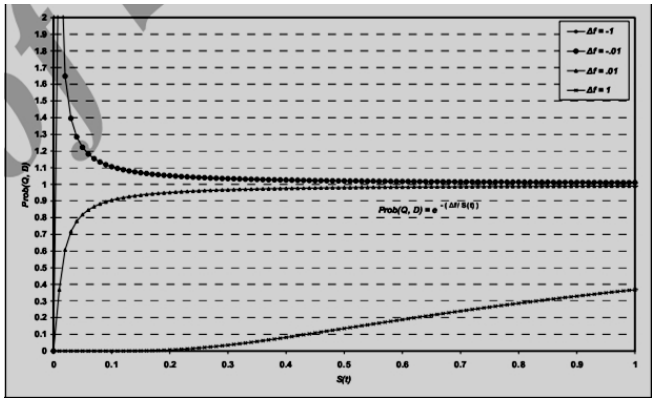
(4_2)

$$Prob\ (Q,D) = e^{\frac{-\Delta(f)}{S(t)}}$$

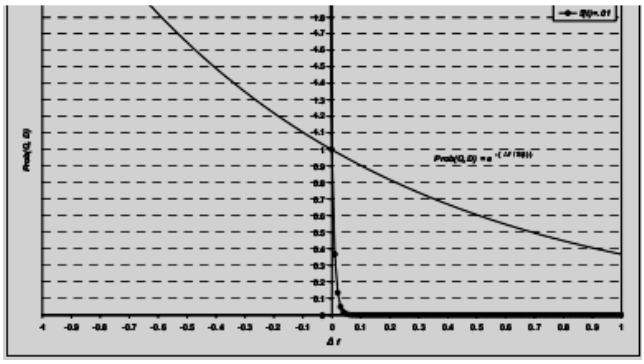
که در آن، $Prop(Q,D)$ احتمال اضافه شدن اسپرم زنبور نر D به حجم محفظه‌ی اسپرم ملکه Q یا احتمال یک جفت‌گیری موفق می‌باشد. $\Delta(f)$، قدر مطلق اختلاف بین تابع هدف زنبورنر D یعنی $f(D))$ و تابع هدف ملکه Q یعنی $f(Q))$ می‌باشد و $S(t)$ سرعت ملکه در لحظه t می‌باشد. واضح است که تابع فوق به عنوان یک تابه نورد شده عمل می‌نماید. این بدان معناست که احتمال جفت‌گیری در ابتدای پرواز جفت‌گیری، که ملکه دارای سرعت زیاد می‌باشد و یا در زمانی که تابع برازش زنبور نر خوب و مناسب بوده و به مقدار تابع برازش ملکه نزدیک باشد بسیار زیاد است. در شکل‌های (۱) و (۲) میزان تغیرات احتمال انتخاب نسبت به تغیرات سرعت ملکه و اختلاف بین تابع هدف ملکه و هر زنبور نر نمایش داده شده است. به تدریج و بعد از هر جابه‌جایی ملکه در فضا، سرعت و انرژی او توسط روابط زیر کاهش می‌یابد:

(5_2)

$$s(t+1) = \alpha \times s(t)$$
$$E(t+1) = E(t) - \gamma$$



شکل 2-4: نمودار احتمال انتخاب زنبورهاي نر بر حسب تغییرات سرعت



شکل 2-5: نمودار احتمال انتخاب زنبورهاي نر برحسب تغییرات مقدار تابع هدف

که د د.

بنابراین این الگوریتم بهینه یابی جفت‌گیری زنبورهای عسل[1] را می‌توان در ۵ گام اساسی زیر خلاصه نمود:

۱-الگوریتم با پرواز جفت‌گیری آغاز می‌شود که در آن ملکه(جواب برتر) به طور احتمالی جفت‌های خود را ازبین زنبورهای نر جهت پر نمودن محفظه‌ی اسپرم خود و در نهایت تولید بچه‌های جدید انتخاب می‌نماید.

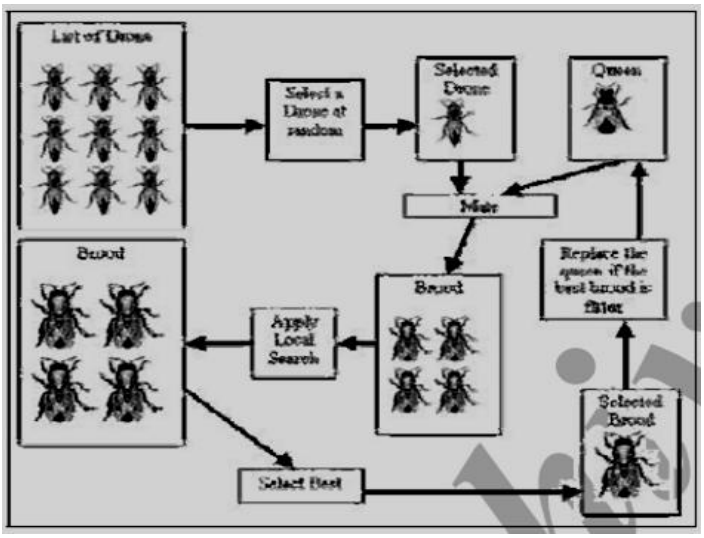
۲-بچه زنبورهای جدید(جوابهای آزمایشی) با جابه‌جایی ژن‌های زنبور نر با ژن‌های ملکه ایجاد می‌شوند.

۳-از کارگرها(توابع کاوشی) جهت جستجوی موضعی(پرورش و ارتقاءنسل بچه زنبورها) استفاده می‌شود.

۴-تابع برازش کارگرها با توجه به میزان ترقی که در نسل زنبورها ایجاد می‌نمایند مرتب می‌شوند.

۵-بچه زنبورهای برتر در این فرایند در صورت برتری نسبت به ملکه‌ی موجود جهت جایگزینی با ملکه و انجام پرواز جفت‌گیری بعدی انتخاب می‌شوند.گام‌های اساسی در الگوریتم HBMO در شکل زیر به نمایش در آمده است.

1- Honey-bee Matikg Optimization Algorithm



شکل 2-6: الگوریتم HBMO

در فرمول بندی ریاضی، زنبور نر و ملکه نمایان‌گر یک رشته از ژن‌ها می‌باشند که آن‌هم نمایان‌گر یک جواب محتمل از مساله می‌باشد. به این ترتیب در هر پرواز جفت‌گیری برخی از ژن‌های زنبور نر بدون تغییر مانده و برخی دیگر به طور تصادفی انتخاب و تغییر خواهند یافت. زنبورهای کارگر که نقش ارتقا نسل بچه زنبورها را دارند به شکل یک سری توابع فراکاوشی تعریف می‌شوند، میزان ارتقاء نسل بچه زنبورها توسط هرتابع فراکاوشی، میزان تابع برازش آن‌ها را تعیین می‌نماید. ملکه‌ها مهم‌ترین نقش را در فرایند جفت‌گیری چه در طبیعت و چه در الگوریتم HBMO بر عهده دارند. هر ملکه با یک رشته ژن، سرعت، انرژی و یک حجم مشخص محفظه‌ی اسپرم شناخته می‌شود. بنابراین انرژی، سرعت و حجم محفظه اسپرم ملکه قبل از هر پرواز جفت‌گیری به صورت تصادفی انتخاب می‌شود. پس از هر جفت‌گیری موفق اسپرم آنها در محفظه‌ی اسپرم ملکه ذخیره می‌گردد. سپس در فرایند تولید بچه‌ها، هر بچه با قرار گیری برخی اززژن‌های نر و کامل شدن مابقی ژن‌ها با ژن‌های ملکه به وجود می‌آید. بدین ترتیب نحوه و روش تخم ریزی ملکه نیز به عنوان یک سری توابع فراکاوشی دیگر تعریف گردیده و به سری توابع مورد نظر جهت ارتقاء نسل اضافه می‌گردند. بدین ترتیب میزان خوبی این توابع نیز با توجه به میزان تاثیر و ارتقائی که در تولید بچه زنبورهای جدید ایجاد می‌نماید محاسبه می‌گردد. در نهایت الگوریتم با مشخص نمودن سه پارامتر کاربر و یک پارامتر از پیش تعیین شده آغاز می‌گردد. پارامتر از پیش تعیین شده تعداد کارگرهاست که بیان‌گر تعداد توابع فراکاوشی در مساله می‌باشد. این پارامتر می‌تواند توسط کاربر جهت تعیین توابع فراکاوشی فعال تغییر یابد. که حدود مجاز تغییرات توسط کاربر بین صفر تا حداکثر تعداد توابع فراکاوشی تعریف می‌شود.

سه پارامتر دیگر تعریف شده توسط کاربر نیز شامل تعداد ملکه‌ها، حجم محفظه‌ی اسپرم ملکه(حداکثر تعداد جفت‌گیری‌های هر ملکه در هر پرواز جفت‌گیری) و حداکثر تعداد بچه زنبورهای حاصل از هر ملکه می‌باشد. انرژی و سرعت ملکه در ابتدای هر پرواز جفت‌گیری به صورت تصادفی تعیین می‌گردد.

بنابراین یک تعداد ملکه به صورت تصادفی انتخاب گردیده و سپس یک سری توابع فراکاوشی با در نظر گرفتن این که ملکه همیشه بهترین زنبور در کندو می‌باشد، به صورت تصادفی جهت بهبود نسل ملکه انتخاب می‌گردد. سپس تعدادی پرواز جفت‌گیری ترتیب داده می‌شود که در هر پرواز، ملکه(ها) با توجه به انرژی و سرعت خود که قبل از هر پرواز به طور تصادفی تعیین می‌گردد، جهت یافتن زنبورهای نر مدنظر خود در فضا حرکت می‌نماید. در ابتدای هر پرواز جفت‌گیری، هر زنبور نر به صورت تصادفی تولید می‌گردد.

جابه‌جایی صورت گرفته در فضا توسط ملکه با توجه به سرعت او می‌باشد که بیان‌گر احتمال آمیزش ژن‌های زنبور نر با ژن ملکه است. در ابتدای هر پرواز، ملکه دارای سرعت بالا بوده و بنابراین می‌تواند گام‌های بلندی را در فضا جهت تعیین جفت مورد نظر خود بردارد. به تدریج که انرژی ملکه کاهش می‌یابد، سرعت او نیز کاهش یافته و در نتیجه محدوده‌ی عمل ملکه در فضا نیز محدودتر خواهد گردید. در هرگام ملکه در فضا، ملکه، زنبور نر موجود در مجاورت خود را توسط تابع احتمالاتی (۱) آزمون می‌نماید.

در صورتی که جفت‌گیری با موفقیت انجام شود(زنبور نر از آستانه‌ی انتخاب ملکه در تابع فوق گذر نماید)، اسپرم زنبور نر موجود وارد محفظه‌ی اسپرم ملکه شده و در آنجا ذخیره می‌گردد. لازم به ذکر مجدد است از آنجا که زنبورهای نر تک جنسی می‌باشند، در هر تولید زنبورهای نر برخی از ژن‌های تشکیل دهنده‌ی آن‌ها به طور تصادفی به گونه‌ای مشخص می‌شود که در طول فرایند بدون تغییر بمانند. زمانی که کلیه‌ی ملکه‌ها پرواز جفت‌گیری خود را به پایان رساندند، تولید بچه‌ها آغاز می‌گردد. جهت تولید بچه‌ها به تعداد مورد نظر که از پیش تعیین شده است هر ملکه با تعدادی از اسپرم‌های درون محفظه‌ی اسپرم خود که به طور تصادفی و توسط توابع مورد نظر تعیین می‌گردد، جفت‌گیری می‌نماید. سپس کارگران با توجه به میزان برازش خود جهت ارتقاء ملکه و بچه زنبورها انتخاب می‌گردند. پس ازتولید کلیه‌ی بچه زنبورها، همگی آنها با توجه به مقدار برازش خود مرتب می‌شوند. بهترین بچه زنبورها درصورت برتری نسبت به ملکه(ها) جایگزین آنها می‌گردند. مابقی بچه زنبورها در ادامه فرایند کشته شده و پرواز جفت‌گیری بعدی آغاز می‌گردد. این عمل تا آنجا ادامه می‌یابد که یا تعداد پروازهای جفت‌گیری از پیش تعیین شده به انتها برسد و یا معیار همگرایی تعیین شده در مساله ارضا گردد که در این صورت حل مساله پایان می‌یابد.

فصل سوم

کاربردهای الگوریتم

الگوریتم زنبورعسل را باید الگوریتمی معرفی کرد که علی‌رغم سن نه چندان زیاد کاربردهای فراوانی برای آن ذکر شده است. در این فصل ما به تفصیل به بررسی کاربردهای الگوریتم در زمینه‌های زیر می‌پردازیم:

- *The Ride-Matching Problem*
- دنیای مجازی در تسخیر زنبور دیجیتال
- بهره برداری بهینه از سد

3-1 The Ride-Matching Problem

شبکه‌های راه شهری در بیشتر کشورها به طرز شدیدی متراکم شده در نتیجه زمان سفر درون شهری و تعداد توقف‌ها افزایش یافته است. همچنین وقفه‌های پیش بینی نشده ، هزینه‌ی سفر درون شهری ، مزاحمت برای رانندگان و مسافران و نیز آلودگی هوا، سطح صدای ناهنجار و تعداد تصادفات ناشی از ترافیک هم افزایش یافته است.

افزایش ظرفیت شبکه‌ی ترافیکی به وسیله‌ی ساختمان‌ها و جاده‌های بیشتر نه تنها هزینه‌های زیادی دارد بلکه آسیب‌های محیطی زیادی نیز در پی خواهد داشت . در نتیجه استفاده‌ی موثرتر از منابع موجود برای حمایت از رشد تقاضای سفر ضروری است.

RideSharing یکی از تکنیک های شناخته شده ی مدیریت رشد سفر *(TDM)* است که توصیه به شریک شدن دو یا چند نفر(با دو یا چند مبدا و مقصد) در یک وسیله ی نقلیه می‌کند. تمام رانندگانی که در *RideSharing* شرکت می‌کنند باید به اپراتور اطلاعات زیر را در مورد تکنیک سفر اشاره شده ارائه د ه‌ند:

الف) ظرفیت وسیله‌ی نقلیه (دو، سه و یا چهار نفر)

ب) روزهایی از هفته که هر فرد برای شرکت در *RS* حاضر است.

ج) مبدا سفر برای هر روز هفته.

د) مسافت سفر برای هر روز هفته.

ه) مقصد مورد نظر و یا زمان رسیدن برای هر روز هفته.

مسأله‌ی *RS* که در این مقاله به آن اشاره شده می‌تواند به روش زیر تعریف شود:

مسیریابی و زمان بندی وسایل نقلیه و مسافران برای تمامی هفته در "بهترین روش ممکن ".

موارد زیر توابع پتانسیل مفعولی هستند:

الف) کمینه کردن کل مسافتی که توسط تمامی اعضای شرکت کننده پیموده می‌شود.

ب) کمینه کردن وقفه‌ی کل
ج) برابر کردن نسبی بهره برداری از وسایل نقلیه
ما وقتی از مساله‌ی بهینه‌سازی ترکیبی معین استفاده می‌کنیم که مقصد در نظر گرفته شده یا زمان رسیدن یا هر دو ثابت هستند(برای مثال "من میخواهم راس ساعت ۸ صبح سوار شوم "). به بیان دیگر در بسیاری از حالت‌های زندگی واقعی، مقصد در نظر گرفته شده و یا زمان رسیدن از منطق فازی تبعیت می کند. در این حالت با مسالهٔ ی *RS* باید به عنوان مسالهٔ ی بهینه‌سازی ترکیبی دارای عدم قطعیت رفتار کرد.

Solving The Ride-Matching Problem By The Fuzzy Bee System:

بیایید هر مسافری که در *RS* شرکت کرده است را به عنوان یک گره در نظر بگیریم. ما مسالهمان را به مراحلی تجزیه می کنیم. اولین سرنشین ماشین(راننده) مرحله ی اول معرفی می‌شود و دومین سرنشین (مسافری) که به *RS* ملحق می‌شود(مرحله ی دوم) و...

در هر رفت زنبور تعداد معینی از گره‌ها را بازدید خواهد کرد و یک راه حل زیر پایه ایجاد می‌کند و پس از آن به کندو برمی‌گردد.

در کندو زنبور در روند تصمیم گیری شرکت خواهد کرد. زنبورها تمام راه حل‌های زیر پایه‌ی تولید شده را مقایسه می‌کنند. بر مبنای کیفیت راه حل های زیرپایه ی تولید شده هر زنبور تصمیم می گیرد که آیا مسیر تولید شده را ترک کند و زنبور سرگردان شود یا به پرواز در طول مسیر کشف شده بدون گرفتن همراه‌ادامه دهد یا برقصد و بدین گونه همراهی بگیرد قبل از آنکه به مسیر کشف شده بازگردد.
بسته به کیفیت راه حل زیر پایه‌ی تولید شده هر زنبور سطح معینی از وفاداری را به راه قبلی کشف شده دارد. به عنوان مثال زنبورهای *B1* و *B2* و *B3* در فرایند تصمیم‌گیری شرکت می کنند. پس از مقایسه ی تمام راه حل‌های زیرپایه ی تولید شده زنبور *B1* تصمیم می‌گیرد راه فعلی را ترک کرده و به زنبور *B2* ملحق شود. زنبورهای *B1* و *B2* با هم در طول مسیری که به وسیله ی زنبور *B2* تولید شده پرواز می کنند. هنگامی که به انتهای مسیر رسیدند آن ها آزاد هستند تا تصمیمی فردی درباره ی گره بعدی که باید بازدید شود بگیرند.

زنبور *B3* به پرواز در طول مسیر بدون گرفتن همراه ادامه می‌دهد. در این روش زنبورها دوباره عمل رفت را انجام می‌دهند. در طی دومین رفت زنبورها تعداد کمی گره بیشتری(نسبت به‌اولین بار) را ملاقات می کنند راه حل‌های زیرپایه‌ی تولید شده‌ی قبلی را توسعه می‌دهند و پس از آن دوباره عمل برگشت را اجرا می‌کنند و به کندو بازمی‌گردند.

در کندو دوباره زنبورها در فرایند تصمیم گیری شرکت می کنند، تصمیم می گیرند، سومین رفت را اجرا می‌کنند و... .

تکرارها هنگامی تمام می‌شود که زنبورها تمامی گره ها را بازدید کرده باشند. هنگام انتخاب گره بعدی که باید در رفت بعدی بازدید شود زنبور گرهی خاصی را به عنوان "کمتر جذاب"، "جذاب" و "خیلی جذاب" در نظر می گیرد که وابسته به نزدیکی مکانی یا زمانی بین دو درخواست از دو مسافر است .
ما این نزدیکی ها را "فاصله‌ی مکانی در مبدا "، "فاصله ی مکانی در مسافت " و "فاصله‌ی زمانی در ورود به مقصد" می‌نامیم.

ما در نظر می‌گیریم که زنبور مصنوعی می تواند فاصله ی خاصی بین دو گره را با عنوان های "کوتاه"، "متوسط" و "طولانی" شناسانی کند.
الگوریتم منطق تقریبی جذابیت گره را با قوانین زیر تعیین می‌کند:
اگر فاصله‌ی مکانی در مبدا کوتاه باشد و فاصله ی مکانی در مسافت کوتاه باشد و فاصله ی زمانی ورود کوتاه باشد آنگاه جذابیت گره بالا است.
بدی راه(که در دومین معادله تعریف شد) در ارتباط با الگوریتم منطق تقریبی برای تعیین وفاداری زنبور به راه کشف شده‌استفاده می شود. الگوریتم منطق تقریبی برای تعیین جذابیت مسیر پیشنهاد شده‌از قوانین زیر تبعیت می‌کند:
اگر طول مسیر پیشنهادی کوتاه باشد و تعداد زنبورهایی که آ ن راه را پیشنهاد داده‌اند کم باشد آنگاه جذابیت آن راه متوسط است.

3-1-1 Numerical Experiment

ما مدل پیشنهادی *RS* را برای شهر *Trani*(شهر کوچک و زیبایی در جنوب ایتالیا) تا شهر *Bari* (مرکز منطقه‌ی *Puglia*) امتحان کردیم. ما اطلاعات مربوط به ۹۷ مسافر که خواستار شرکت در پروژه ی *RideSharing*بودند را جمع‌آوری کرده و برای سادگی مسأله ظرفیت هر اتومبیل را چهار نفر در نظر گرفیم. در این حالت الگوریتم ۹۶=۴*۲۴ نفر از ۹۷ نفر را برای ساختن بهترین راه کنار می‌گذارد. ما از کندویی با ۱۵ زنبور که سریعا (و یکبار) کندو را ترک می کنند استفاده کردیم. زنبورها فقط ۶ مسیر غذایابی را پیدا کردند و دیگر مسیرها رها شدند.

۳-۲ دنیای مجازی در تسخیر زنبور دیجیتال

به زودی دنیای مجازی و کاربران آن شاهد رفت‌وآمد زنبورهای دیجیتال در رایانه‌ها خواهند بود که برخلاف کرم‌ها و ویروس‌های خرابکار، حضورشان جای نگرانی ندارد و فناوری نوین آن‌ها را برای کمک به ما اعزام کرده است. این الگوریتم در دنیای دیجیتال با هدف کنترل و تامین امنیت رایانه ها صورت می‌گیرد.

هوش زنبور عسل یا زنبورهای دیجیتال عنوان ریزبرنامه‌هایی است که اکنون و به همت محققان بر اساس رفتار واقعی زنبور عسل و با هدف تامین امنیت رایانه‌ها مدل‌سازی شده است و از قرار معلوم این ریزبرنامه‌های سرگردان در هیئت مأموران امنیتی با پرسه‌زدن در رایانه‌ها و بخش‌های مختلف آن به کار تفحص و شناسایی مهاجمان و مزاحمان امنیت و سلامت دنیای مجازی یا همان کرم‌ها و ویروس‌های رایانه‌ای می‌پردازند.

دانشمندان در صدد هستند تا ارتشی از زنبورهای دیجیتال خلق کنند که با تمام قوا و همراه با افسران ارشد، گروهبانان دیجیتال و نگهبانان و دیده‌وران خود در اعماق رایانه‌ها به کار جستجوی ویروس‌ها، کرم‌ها و

سایر افزارهای مضر می‌پردازند. دانشمندان معتقدند این نرم‌افزار آنتی‌ویروس جدید می‌تواند ضمن آزاد گذاشتن دسترسی سخت‌افزار، حفاظت بهتری را برای رایانه ایجاد کند.

در همین رابطه، یکی از دانشیاران علوم رایانه دانشگاه‌های امریکا که سهم به سزایی در توسعه‌ی فناوری ریزبرنامه‌های جستجوگر هوشمند موسوم به زنبورهای دیجیتال داشته، درباره‌ی این ایده‌ی فناورانه ضدویروسی خاطرنشان می‌سازد که این زنبورها در واقع برای پی بردن به موردی بسیار اساسی و پایه همچون نرخ و سرعت اتصال استفاده می‌شوند؛ اما در ادامه و با بهره‌برداری واحد پردازش مرکزی رایانه و دیگر عناصر درگیر با آن، کار جمع‌آوری شواهدی که ما را به آلودگی خاصی یا تهدیدی امنیتی رهنمون می‌شوند نیز در دستور کار این ریزبرنامه‌ها قرار می‌گیرد.

هریک از این زنبورهای منفرد همچون همتایان زیستی واقعی خود خیلی زرنگ و باهوش نیستند. اما مواردی نظیر سرعت برقراری اتصال، بهره‌برداری سی‌پی‌یو یا یکی از تقریبا ۶۰ مورد جزییات فنی دیگر رایانه همان چیزی است که تمام و کمال از سوی این جانوران مجازی احساس شده و مورد دقت و مراقبت لازم قرار می‌گیرد.

در حقیقت اساس مکانیسم عملکرد این زنبورهای مجازی بر پایه‌ی همان ساز و کاری شکل گرفته که در عالم طبیعت و در دنیای زنبور عسل در جریان است. به عبارتی همان‌طور که زنبورها در مواجهه با موردی غیرعادی مقداری از ماده شیمیایی فرومون را از خود در موضع مربوطه برجای می‌گذارند تا علامت و هشداری برای سایرین باشد، در عالم مجازی نیز زمانی که زنبوری چیزی غیرعادی را ردیابی و کشف می‌کند، از خود فرومون دیجیتال برجای می‌گذارد که در واقع مصداق حس دیجیتال ظریفی است که وجود موردی غیرعادی در آن موضع را اعلام می‌کند و همچنین به سایر زنبورها نیز می‌گوید که باید مورد مربوطه را بازدید و کنترل کنند.

به گفته‌ی محققان، این زنبور عسل دیجیتال، هرگونه فعالیت مشکوک را به زنبورهای قراول و نگهبان گزارش می‌کند که همین مکانیسم دیده‌بانی و گزارش در حقیقت برنامه‌ای طراحی شده برای دیده‌بانی و مراقبت کامل مجموعه‌ای از رایانه‌های متصل به شبکه است.

در این میان و با حفظ سلسله مراتب، وظیفه‌ی زنبورهای نگهبان طبقه‌بندی و مرتب‌سازی تمامی داده‌ها و اطلاعات جمع‌آوری شده از سوی زنبورهای دیگر می‌باشد و در صورت مشکوک بودن اطلاعات واصله، مراتب را با انتقال دادن به یک زنبور گروهبان گزارش می‌کنند. در این مرحله زنبور گروهبان زنگ خطر و آماده‌باش را برای ناظری انسانی که می‌تواند از عهده‌ی مشکل مربوطه برآید به صدا درمی‌آورد.

اما نکته‌ی جالب توجه و تدبیری که از سوی محققان در این برنامه‌ی پایش رایانه‌ای زنبوری پیش‌بینی شده، سیاست پاداش و مجازاتی است که در طبیعت و در جامعه زنبور عسل وجود دارد، به نحوی که زنبورهای نگهبان و گروهبان به زنبورهای کارگر برای یافتن مشکلات موجود پاداش خدمت می‌دهند.

به عبارتی اگر زنبور کارگری نتواند به حد کفایت مشکلات را پیدا کند یک زنبور مرده به حساب می‌آید و از دور خارج می‌شود؛ البته برنامه همواره تعداد حداقلی از این زنبورها را حفظ می‌کند و اصطلاحا دست خود را خالی نمی‌گذارد.

از سوی دیگر، اگر نوع به خصوصی از زنبورها حجم قابل ملاحظه‌ای از مشکلات را پیدا کنند، شمار بیشتری از آن‌ها برای کنترل و نظارت مشکل مربوطه به‌عنوان قوای کمکی اعزام می‌شوند؛ به این ترتیب به خدمات سودمند گروه مزبور پاداش داده می‌شود.

در حقیقت اساس مکانیزم عملکرد زنبورهای مجازی برپایه‌ی همان ساز و کاری شکل گرفته که در عالم طبیعت و در دنیای زنبور عسل در جریان است.

به گفته‌ی محققان، این نظام کامل ضد ویروس رایانه‌ای به طور کامل از روی یک کلونی طبیعی زنبور عسل مدل‌سازی و الگوبرداری شده و در ساختار تشکیلاتی خود از مفهوم هوش گروهی برای یافتن و تشخیص مشکلات بهره می‌برد.

در حقیقت این همان نظام تشکیلات اجتماعی مبتنی بر غریزه و ذکاوت جمعی است که در گونه‌های مختلف جانوری به اشکال مختلف زندگی اجتماعی گروهی، دسته‌ای و کپه‌ای مشاهده می‌شود و راهبردهای دفاعی و تهاجمی چنین گونه‌هایی نیز بر بنیاد همین ساختار جمعی بنا شده است؛ به نحوی که در نظام‌های اجتماعی زنبورهای عسل برای تمامی اعضای جامعه ، کار و وظایفی تعریف شده و کار و هوشیاری ، وظیفه‌ی همیشگی است و از همین رو چنین ساختارهای متشکل جمعی در مواجهه با بلایا و بروز مشکلات بزرگ به واسطه گوش به زنگ بودن تمام اعضا از قابلیت انعطاف‌پذیری و تصمیم‌گیری‌های کلی و سریع برخوردارند.

در دنیای مجازی نیز به‌کارگیری زنبورهای دیجیتال به جای برنامه‌های آنتی‌ویروس سنتی و رایج را باید به حساب انعطاف‌پذیری بالای زنبورهای دیجیتال گذاشت. نرم‌افزارهای آنتی‌ویروس سنتی که در حال حاضر استفاده می‌شوند معمولا کار پایش و تجسس را به شکل دائمی یا بر اساس یک جدول زمانی وضع شده انجام می‌دهند. این در حالی است که به‌رغم موثر و کارآمد بودن کنترل و تجسس دائمی برای تهدیدات، چنین فرآیندی حجم زیادی از منابع رایانه‌ای را درگیر می‌کند؛ منابعی که در واقع بهتر است به جای درگیر شدن در این فرآیند به مصرف انجام کاری در جای دیگر برسند.

به‌اعتقاد کارشناسان هرچند فرآیند اسکن یا تجسس آنتی‌ویروس‌ها در اوقات معین و معمولا در شب‌ها، موجب مصرف بهینه‌ی رایانه می‌شود، اما در عوض یک رایانه را در معرض آسیب‌پذیری بیشتری قرار می‌دهد.

به گفته‌ی طراحان برنامه‌ی آنتی‌ویروس زنبورهای دیجیتال، از آنجا که تعداد زنبورها به موازات تعداد مشکلاتی که پیدا می‌شوند افزایش و کاهش می‌یابند، راهکار مناسبی برای آزاد گذاشتن سخت‌افزار رایانه مهیا می‌کند، طوری که وقتی حمله و تهاجمی در حال وقوع نیست، سخت‌افزار رایانه آزاد است تا به اجرای

محاسبات خود بپردازد. به همین ترتیب در صورتی که تهاجمی صورت گیرد، زنبورهای بیشتری می‌توانند به سرعت ایجاد شوند تا با نیروی کافی از عهده‌ی حمله برآیند.

محققان در فرآیند طراحی و ساخت برنامه‌ی آنتی‌ویروس زنبورهای دیجیتال ابتدا به خلق سه زنبور دیجیتال از میان انواع آن که نهایتا مد نظرشان بود مبادرت کردند. در مرحله‌ی بعد و به منظور آزمایش کارایی و تاثیربخشی، این ویروس‌کش‌های هوشیار، بانکی از رایانه‌ها را مهیای کار کردند و سپس سه ویروس از گونه‌ی کرم‌ها را به داخل این رایانه‌های لینوکس محور مملو از زنبور رها ساختند.

نتایج عملکرد نشان داد سه زنبور دیجیتال حاضر در این رایانه‌ها که پیش از این هیچ ویروسی را ملاقات نکرده بودند، در عین حال موفق به شناسایی و تعیین هویت این ویروس‌ها شدند و تنها از طریق کنترل و مراقبت سه جنبه‌ی بسیار ویژه‌ی رایانه‌ها به این مهم دست یافتند.

فناوری ریزبرنامه‌های تجسسی رایانه‌ای و در راس آن‌ها برنامه‌ی آنتی‌ویروسی زنبورهای دیجیتال در حالی برای محافظت بهتر و بیشتر امنیت رایانه‌ها و شبکه مطرح می‌شود که تاکنون نرم‌افزارهای آنتی‌ویروس رایج و سنتی به‌رغم کارایی‌های متفاوت و بعضا سودمند خود درخصوص موارد مهمی از جمله کارایی نرخ سرعت اتصال و درگیر نکردن سخت‌افزار و درایوهای سخت رایانه در حین فرآیند تجسس و حذف و پاکسازی نتوانسته‌اند خود را کارآمد نشان دهند بنابراین و در مقام مقایسه، قابلیت‌های انعطاف‌پذیری فناوری آنتی‌ویروس ابتکاری محققان، از آزادی بیشتر سخت‌افزار و کاهش آسیب‌پذیری رایانه‌ها خبر می‌دهد و از همه مهم‌تر سهولت دسترسی و سرعت بالای اتصال به شبکه را برای کاربران مهیا خواهد کرد.

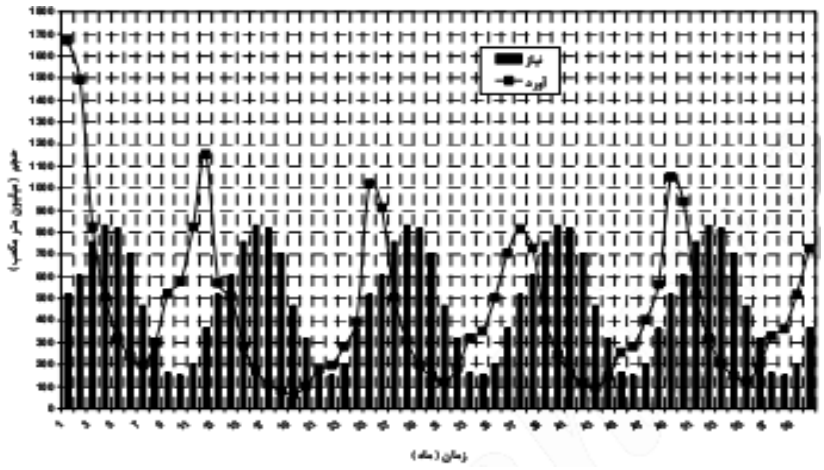
۳-۳ بهینه‌سازی سد

در این بخش، بهره‌برداری بهینه از سد دز در جنوب غربی ایران که درسال ۱۳۴۱ مورد بهره‌برداری قرار گرفته است مدنظر قرار گرفته شده است. میزان آب‌دهی ۶۰ ماهه‌ی رودخانه در محل این سد به همراه میزان متوسط نیاز کشاورزی ماهیانه‌ی پایین دست سد در شکل ۳ـ۱ ارائه شده است. میزان متوسط افت خالص ماهیانه(تبخیر منهای بارش) نیز در شکل ۳ـ۲ ارائه شده است.

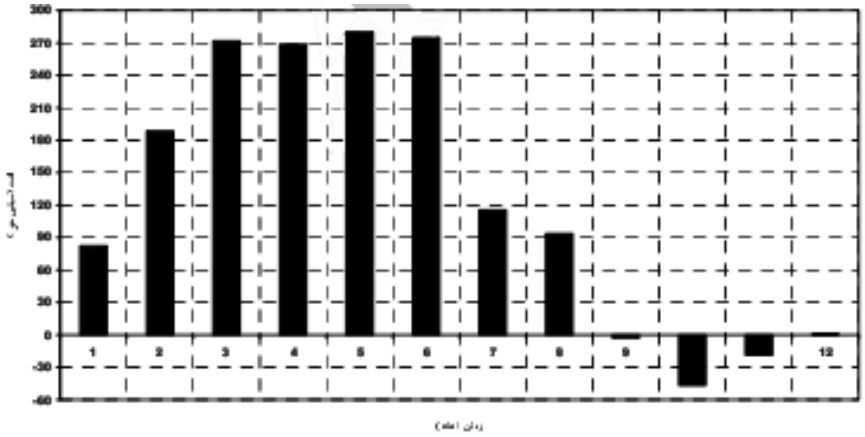
جریان سالیانه‌ی وارد به مخزن برابر با ۵۹۰۰ میلیون مترمکعب و متوسط نیاز سالیانه‌ی پایین دست برابر ۵۳۰۳ میلیون متر مکعب تخمین ذخیره می‌شود. حجم حداقل سد(حجم مرده‌ی آن) برابر با ۸۳۰MCM و حجم حداکثر مخزن نیز برابر با ۳۳۴۰MCM در نظر گرفته شده است. جهت ارائه‌ی رابطه‌ای بین حجم و سطح مخزن، رابطه‌ی خطی زیر در نظر گرفته شده که قابلیت تبدیل حجم مخزن به سطح متناظر به آن را در محدوده‌ی بین سطوح و احجام حداقل تا حداکثر مخزن دارا می‌باشد.

(1-3)

$$A_t = 11.291 = +0.0157 + S_t$$



شکل 3-1: جریان ماهیانه‌ی ورودی به مخزن و نیاز متوسط ماهانه



شکل 3-2: میزان متوسط افت خالص ماهانه

در این بخش مساله‌ی بهره برداری بهینه از مخزن سد دز در ۶۰ پریود زمانی و با ۶۰ متغیر تصمیم گسسته مورد ارزیابی قرار گرفته است.

هدف در این مساله، کمینه‌سازی مجموع مجذور تفاضل رهاسازی از نیاز ماهیانه در کل دوره است.

(2_3)

$$\text{Min TSD}= \sum_{t=1}^{nt}(R_{(t)} - D_{(t)})^2$$

که در آن TSD ، مجموع مجذور تفاضل رهاسازی ماهیانه‌ی $R_{(t)}$ از نیاز $D_{(t)}$ در دوره‌ی t است. در این مثال، حجم فعال مخزن(۲۵۱۰ میلیون متر مکعب) به طور نسبتا یکنواخت به ۱۴ کلاس تقسیم شده است. بدین ترتیب فضای تصمیم از $۶۱^{۱۴}$ حالت شکل گرفته است. با توجه به گسسته بودن متغیرهای تصمیم در این‌گونه موارد استفاده از برنامه‌ریزی پویا(DP) بسیار مطلوب بوده و کاربرد مدل‌های DP می‌تواند ما را به سمت جواب بهینه‌ی مطلق هدایت نماید. مقدار تابع هدف نهایی حاصل از حل این مساله به مدل DP برابر با

۱/۰۷ می‌باشد. اما در اولین گام عملی استفاده از الگوریتم بهینه‌یابی جفت‌گیری زنبورعسل سعی می‌گردد تا قابلیت استفاده از این الگوریتم در حل این‌گونه مسائل نیز مورد آزمون قرار گیرد. با کاربرد الگوریتم HBMO در مساله‌ی فوق و در ۱۰ بار اجرای برنامه، نتایج حاصل برای تابع هدف پس از ۵۰ بار انجام پرواز جفت‌گیری در جدول شکل۳ـ۳ وپارامترهای آماری این مقادیر در جدول شکل ۳ـ۴ به نمایش درآمده است.

	شماره اجرا									
	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰
تابع هدف	۱/۳۴	۱/۲۴	۱/۱۰	۱/۴۳	۱/۱۴	۱/۲۸	۱/۲۷	۱/۱۴	۱/۳۴	۱/۳۲

جدول 3-1: مقادیر تابع هدف در 10 بار اجرا و 50 پرواز جفت‌گیری

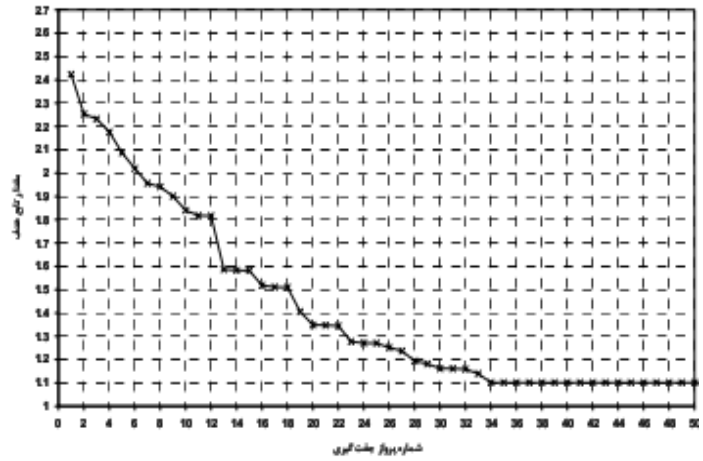
	کمینه	بیشینه	متوسط	انحراف معیار	ضریب تغییرات
تابع هدف	۱/۱۰	۱/۴۳	۱/۲۶	۰/۱۱	۰/۰۸۴

جدول 3-2: پارامترهای آماری تابع هدف در 10 بار اجرا و 50 پرواز جفت‌گیری

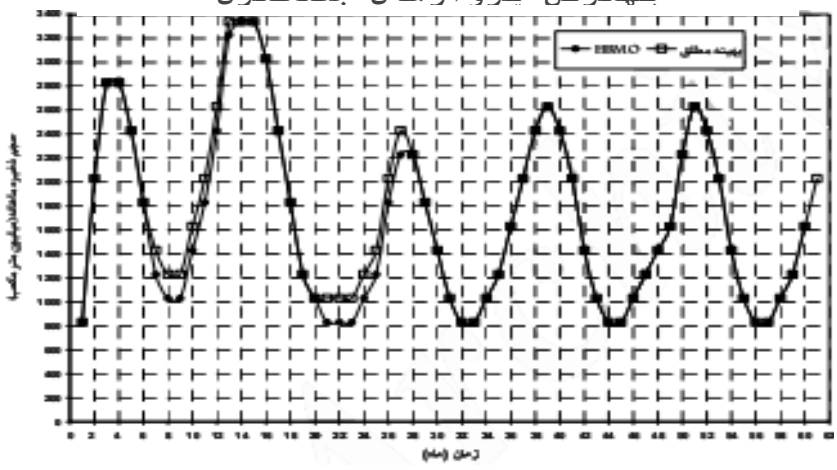
همان‌گونه که ملاحظه می‌گردد بهترین مقدار تابع هدف حاصل در پایان این ۱۰ اجرا برابر ۱/۱ می‌باشد. باید توجه داشت که بهینه‌ی مطلق از نیاز مطلوب کمتر از ۳٪ با بهترین نتیجه‌ی حاصل از حل مساله توسط الگوریتم HBMO اختلاف دارد. همچنین نحوه‌ی تغییرات تابع هدف در طول انجام پروازهای جفت‌گیری در بهترین ۱۰ اجرا در شکل۳ـ۳ ارائه گردیده است.

همگرایی سریع و میزان نزدیکی به حل بهینه‌ی مطلق و تامین نیاز مطلوب در هردوره از جمله‌ی مواردی است که توفیق مدل را در دیگر مسائل طراحی و مدیریت منابع آب نوید می‌دهد.

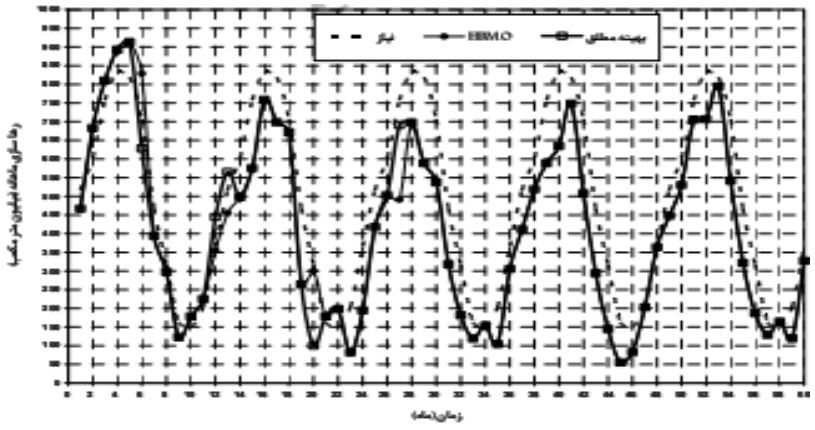
نتایج خروجی مدل و نیز مقدار بهینه‌ی مطلق حاصل از حل مساله به روش DP برای حجم ذخیره‌ی مخزن در هر پریود زمانی در شکل ۳ـ۴ نمایش داده شده است. در شکل ۳ـ۷ نیز میزان رهاسازی ماهیانه از مخزن توسط الگوریتم HBMO پس از ۵۰ پرواز جفت‌گیری و نتایج بهینه‌ی مطلق حل مساله به همراه نیاز ماهیانه در هر دوره موجود می‌باشد.



شکل 3-3: تغییرات تابع هدف در 10
بهترین پروازهای جفتگیری



شکل 3-4: تغییرات حجم مخزن
در هر پریود زمانی



شکل 3-5: تغییرات میزان رهاسازی
از مخزن در هر پریود زمانی

۳-۴ ایده‌ی روباتی[1]

یکی دیگر از کاربردهای این الگوریتم ایده‌ی روباتی ترشکو و لوئنگارو است که در زیر به طور مختصر شرح داده می‌شود.

تلاش‌های زیادی برای مدل کردن رفتارهای خاص و هوشمندانه‌ی تجمع زنبورهای عسل انجام گرفته است. Tereshkoو Loengarov کلونی زنبور را به عنوان یک سیستم پویا درنظر گرفتند که از محیط اطراف اطلاعات جمع‌آوری می‌کند و رفتار خود را براساس این اطلاعات ب ه دست آمده تنظیم می‌نماید. آن‌ها یک ایده‌ی روباتی با توجه به رفتار کاوشی زنبورها مطرح کردند. غالبا این روبات ها به صورت فیزیکی و عملکردی یکسان هستند. در نتیجه هر روبات می تواند به طور تصادفی جایگزین دیگر روبات ها گردد. هرروبات قابلیت تجمع و تحمل خطا را دارا است. با رخ دادن خطا در یک عامل، کار کل سیستم مختل نخواهد شد. روبات های مجزا، مانند حشرات، دارای قابلیت ها وتوانایی‌های محدودی هستند. همچنین دانش محدودی از محیط دارند. به عبارت دیگر تجمع (ازدحام)، هوش جمعی همکارانه را بهبود می دهد. همچنین این آزمایش نشان می دهد که روبات‌های حشره مانند در انجام وظایف حقیقی روبات‌ها، موفق هستند.

۳ـ۵ سایر کاربردها

برخی کاربردهای دیگر الگوریتم در مهندسی عبارتند از:

آموزش شبکه عصبی برای الگو شناسی
زمان بندی کارها برای ماشین های تولیدی
دسته‌بندی اطلاعات
بهینه‌سازی طراحی اجزای مکانیکی
بهینه‌سازی چند گانه
میزان کردن کنترل کننده های منطق فازی برای ربات های ورزشکار

1- The idea of robot

فصل چهارم

کاربرد الگوریتم زنبورعسل در بهینه‌سازی مسائل ریاضی

در این فصل کاربرد الگوریتم به طور خاص در بهینه‌سازی مسائل ریاضی بررسی می‌شود.

برای این کار ابتدا به طورخلاصه به بیان برخی مفاهیم ریاضی خواهم پرداخت و در ادامه با ذکر چند مثال کاربرد عملی آن را نشان خواهم داد.

۴-۱ بهینه‌سازی[1]

در ریاضیات، علوم کامپیوتر و اقتصاد، **بهینه‌سازی** یا **برنامه‌ریزی ریاضی**، به انتخاب عناصر بهینه از یک مجموعه از آلترناتیوهای قابل دست‌یابی می‌پردازد. به عبارت بهتر، به دنبال **یافتن بهترین مقدار قابل دست‌یابی** از یک تابع هدف تعریف شده بر یک دامنه‌ی معین از مقادیر است. در ساده‌ترین حالت، هدف، حداقل یا حداکثرسازی یک تابع حقیقی، با انتخاب نظام‌مند مقادیر حقیقی یا اعداد صحیح از یک مجموعه از مقادیر ممکن است. ساده‌ترین مثال، استفاده از یک تابع هدف حقیقی مقداراست.

تعمیم تئوری بهینه‌سازی و تکنیک‌های فرمول‌بندی بخش بزرگی از ریاضیات کاربردی را شکل می‌دهد. تحقیق در عملیات، برنامه‌ریزی با اعداد صحیح و مختلط، مدل‌های شبکه‌ای، تئوری کنترل، برنامه‌ریزی غیرخطی، نظریه‌ی صف و برنامه‌ریزی پویا برخی شاخه‌های ریاضیات کاربردی مرتبط با بهینه‌سازی هستند که امروزه در مدیریت و اقتصاد کاربرد وسیعی دارند.

اولین تکنیک بهینه‌سازی را کارل فردریش گاوس ابداع کرد. اما عمده ی اصطلاحات مورد استفاده در این حوزه به دوره ی معاصر برمی‌گردد. اصطلاح برنامه‌ریزی خطی را نخستین بار جرج دانتزیگ در ۱۹۴۰ میلادی به‌کاربرد. اصطلاح برنامه‌ریزی در حوزه‌ی بهینه‌سازی به معنای برنامه‌نویسی برای کامپیوتر نیست، با این همه، رایانه‌ها، امروزه به شکل گسترده‌ای در حل مسائل ریاضی مورد استفاده قرار می‌گیرند. ریشه ی این اصطلاح به کاربرد واژه ی «برنامه» در ارتش ایالات متحده برمی‌گردد که در اشاره به طرح های لجستیک و آموزشی به کار می‌رفت که دانتزیگ آن را مورد مطالعه قرار می‌داد.

1- Optimization

۴-۱-۱ شاخه‌های اصلی

برنامه‌ریزی محدب به بررسی حالتی می‌پردازد که تابع هدف محدب است و قیودی اگر وجود داشته باشند یک مجموعه‌ی محدب را شکل می‌دهند. می‌توان آن را حالت خاصی از برنامه‌ریزی غیرخطی یا تعمیمی از برنامه‌ریزی درجه دوم محدب دانست.

برنامه‌ریزی خطی، نوعی از برنامه‌ریزی محدب است که به بررسی مواردی می‌پردازد که در آن تابع هدف خطی است و مجموعه ی قیود با استفاده از معادلات و نامعادلات خطی مشخص می‌شود. چنین مجموعه‌ای اگر کران‌دار باشد چندوجهی یا کثیرالوجوه خوانده می‌شود.

برنامه‌ریزی مخروط مرتبه ی دوم، نوعی از برنامه‌ریزی محدب است که شامل انواع خاصی از برنامه‌های درجه دوم است.

برنامه‌ریزی شبه‌معین نوعی از برنامه‌ریزی محدب و تعمیمی است از برنامه‌ریزی خطی و درجه ی دوم که متغیرهای تحت بررسی آن، ماتریس‌های شبه‌معین است.

برنامه‌ریزی مخروطی شکل عمومی برنامه‌ریزی محدب است. برنامه‌ریزی خطی، برنامه‌ریزی مخروط مرتبه‌ی دوم و برنامه‌ریزی شبه‌معین را می‌توان به‌عنوان حالت خاصی از این نوع برنامه‌ریزی دانست.

برنامه‌ریزی عدد صحیح به بررسی آن دسته از برنامه‌ریزی‌های خطی می‌پردازد که در آن، همه و یا برخی از متغیرها، تنها مقادیر صحیح اتخاذ می‌کنند.

برنامه‌ریزی درجه دوم که در تابع هدف آن جملات از مرتبه ی دو ظاهر می‌شوند، درحالی که مجموعه‌ی مقادیر ممکن به وسیله‌ی معادلات و نامعادلات خطی مشخص می‌شود. حالات خاصی از تابع هدف را می‌توان تحت برنامه‌ریزی محدب بررسی کرد.

برنامه‌ریزی غیرخطی به بررسی مواردی می‌پردازد که در آن تابع هدف یا قیود و یا هردوی آن ها حاوی بخشی غیرخطی هستند.

برنامه‌ریزی تصادفی به بررسی مواردی می‌پردازد که در آن برخی قیود یا پارامترها به متغیرهای تصادفی بستگی دارند.

برنامه‌ریزی ترکیبیاتی مسائلی را مدنظر دارد که مجموعه جواب ه‌ای ممکن گسسته اند و یا می‌توانند به شکل گسسته تبدیل شوند.

برنامه‌ریزی با بعد نامتناهی مواردی را بررسی می‌کند که مجموعه جواب‌های ممکن یک زیر مجموعه از فضا با بعد نامتناهی است. یک نمونه‌ی آن فضای توابع است.

برنامه‌ریزی فصلی که در آن حداقل یک قید باید ارضا شود ولی لزومی برای برقراری همه قیود نیست.

برنامه‌ریزی مسیری اختصاص به بهینه‌یابی مسیر وسایل نقلیه هوایی فضایی دارد.

در برخی زیرشاخه‌ها، تکنیک‌ها اصولا برای بهینه‌سازی پویا، یعنی بهینه‌سازی در طول زمان به کار می‌رود:

حساب تغییرات با درنظرگرفتن اینکه چنانچه تغییر کوچکی در مسیر انتخابی بدهیم، تابع هدف چه تغییری می‌کند، در بهینه‌یابی تابع هدف در نقاط زمانی بسیار به کار گرفته می‌شود.

کنترل بهینه تعمیمی است از حساب تغییرات.

برنامه‌ریزی پویا مواردی را بررسی می‌کند که استراتژی بهینه‌یابی برمبنای تقسیم مساله به مسائل کوچکتر استوار است. معادله‌ای که روابط بین این مسائل کوچکتر را بررسی می‌کند معادله بلمن خوانده می‌شود.

برنامه‌ریزی ریاضی با قیود تساوی که در آن قیود شامل نامعادلات تغییر یابنده و یا comlementarity است.

۴ـ۲: انواع مسائل بهینه‌سازی و تقسیم بندی آن‌ها

در ریاضیات، مساله بهینه‌سازی، مساله‌ی یافتن بهترین جواب از تمام جواب‌های ممکن است. مساله‌ی بهینه‌سازی A چهارتایی (I,f,m,g) است که در آن

I نشان‌دهنده یک مجموعه است

f تابعی است که به هر عضو معلوم $x \in I$ مجموعه‌ای از جواب‌های ممکن نظیر می‌کند

عبارت m(x,y) برای هر $x \in I$ ، و هر جواب ممکن y برای x نشان‌دهنده اندازه y است، که معمولاً یک عدد حقیقی مثبت است

g تابع هدف است که حاوی مینیمم‌سازی یا ماکزیمم‌سازی است.

هدف یافتن جواب بهینه برای X است، یعنی یک جواب ممکن چون Y که

(1_4)

$$m(x,y) = g\{m(x,y')|y' \in f(x)\}$$

برای هر مساله بهینه‌سازی، یک مساله‌ی تصمیم متناظر وجود دارد که در پی یافتن جوابی ممکن برای یک اندازه خاص m_0 است. در الگوریتم‌های تخمین، الگوریتم‌ها برای یافتن جوابهای نزدیک-بهینه برای مسائل پیچیده طراحی می‌شود. گرچه در این مسائل هم می‌توان مسائل تصمیم متناظر تعریف کرد، ولی عموما آن را با مسائل بهینه‌سازی مشخص می‌کنند که طبیعی‌تر هم هست.

یک تقسیم بندی از بهینه‌سازی را در زیر ببنید.

بهینه‌سازی تک بعدی و بهینه‌سازی چند بعدی

اگر تنها یک متغیر در مساله ی بهینه‌سازی وجود داشته باشد، مساله ی بهینه‌سازی تک بعدی و در غیر این صورت چند بعدی خوانده می‌شود.

بهینه‌سازی پویا و بهینه‌سازی ایستا

اگر تابع هزینه ی مساله‌ی بهینه‌سازی تابعی از زمان نباشد، با یک مساله ی بهینه‌سازی ایستا سر و کار داریم. ولی اگر زمان نیز وارد تابع هزینه شود مساله بهینه‌سازی پویا میشود.

بهینه‌سازی مقید و نامقید

اگر متغیرهای مساله ی بهینه‌سازی به مجموعه و یا قید خاصی محدود شده باشند، با یک مساله ی بهینه‌سازی مقید Constrained Optimization سروکار داریم و در غیر این صور ت مساله، بهینه‌سازی نامقید است.

بهینه‌سازی پیوسته و بهینه‌سازی گسسته

یک مساله‌ی بهینه‌سازی گسسته مساله ای است که در آن مقادیر متغیرهای مساله از یک مجموعه معین گسسته هستند. در حالی که، در یک مساله ی پیوسته، مقادیر متغیرها از یک مجموعه ی پیوسته هستند.

بهینه‌سازی تک معیاره و چند معیاره

یک مساله‌ی بهینه‌سازی تک معیاره (Single Objective)، دارای تنها یک تابع هدف می باشد. اما در یک مساله ی چند معیاره (Multi Objective)، تعداد تابع هدف هایی که ب ه طور هم‌زمان بهینه می شوند بیش از یکی است. معمولاً در یک مساله ی بهینه‌سازی چند معیاره، با دادن اهمیتی (وزنی) به هر یک از توابع هدف و جمع بستن آن‌ها، مساله را تبدیل به یک مساله‌ی تک معیاره می کنند. حل مسائل بهینه‌سازی چند هدفه، به تنهایی مبحث مستقل و مهمی از حوزه بهینه‌سازی است.

۴ـ۳: شکل یک مساله‌ی بهینه‌سازی

مساله‌ی بهینه‌سازی به شکل زیر می‌تواند ارائه می‌شود:

با مفروض گرفتن تابع $f : A \rightarrow \mathbf{R}$، که در آن R مجموعه‌ی اعداد حقیقی است. به دنبال یافتن عنصری از x_{opt} در A هستیم به شکلی که برای هر x متعلق به A داشته باشیم $f(x) \geq f(x_{opt})$ و یا آنکه برای هر x متعلق به A داشته باشیم $f(x) \leq f(x_{opt})$

چنین فرمول‌بندی، مساله‌ی بهینه‌سازی یا مساله‌ی برنامه‌ریزی ریاضی خوانده می‌شود. بسیاری از مسائل جهان واقع می‌تواند در این چهارچوب عمومی مدل شود. نوعاً، A زیرمجموعه‌ای از فضای اقلیدسی $\mathbf{R}^n$ است که اغلب با مجموعه‌ای از قیود، یعنی معادلات و نامعادلاتی که اعضای A باید آن‌ها را ارضا کند، مشخص می‌شود. دامنه‌ی f مجموعه‌ی انتخاب یا فضای جستجو خوانده می‌شود، درحالی که عناصر A، جواب‌های ممکن یا قابل دست‌یابی خوانده می‌شوند.

تابع f، عموما تابع هدف خوانده می‌شود. جواب ممکنی که تابع هدف را ماکزیمم کند و یا اگر هدف مینیمم‌سازی باشد، آنرا مینیمم کند، جواب بهینه خوانده می‌شود.

ماکزیمم و مینیمم نسبی

عموماً، وقتی ناحیه‌ی جواب‌های ممکن مساله محدب نباشد، چندین مینیمم و ماکزیمم نسبی وجود خواهد داشت. ماکزیمم نسبی x^* نقطه‌ای است که برای آن یک $\delta > 0$ وجود داشته باشد به‌طوریکه برای هر x که در رابطه زیر صدق کند

(2_4)

$$\|\mathbf{x} - \mathbf{x}^*\| \leq \delta;$$

داشته باشیم

(3_4)

$$f(\mathbf{x}^*) \geq f(\mathbf{x})$$

یعنی در ناحیه‌ی جواب، مقدار تابعی همه نقاط حول (لااقل) یک همسایگی از x^*، کوچکتر یا مساوی مقدار تابع در آن نقطه باشد. مینیمم نسبی نیز به شکل مشابهی تعریف می‌شود.

الگوریتم‌های زیادی برای حل مسائل غیرمحدب ارائه شده است. شاخه‌ای از ریاضیات کاربردی و آنالیز عددی که با ایجاد الگوریتم‌های قطعی -با تضمین همگرایی در زمان متناهی- به جواب بهینه ی واقعی یک مساله غیرمحدب بینجامد، بهینه‌یابی سرتاسری خوانده می‌شود.

نمادگذاری

مسائل بهینه‌سازی غالباً با نمادهای ویژه‌ای نمایش داده می‌شوند. در اینجا چند مثال آورده می‌شود.

(4_4)

$$\min_{x \in \mathbb{R}} (x^2 + 1)$$

که به معنی یافتن مقدار مینیمم تابع هدف $x^2 + 1$ است، که x متعلق به مجموعه‌ی اعداد حقیقی است. به وضوح، حداقل مقدار این تابع، 1 است که در $x=0$ اتفاق می‌افتد.

(5_4)

$$\max_{x \in \mathbb{R}} 2x$$

که به معنی یافتن مقدار ماکزیمم تابع هدف $2x$ است، که x متعلق به مجموعه‌ی اعداد حقیقی است. در این حالت حداکثری وجود ندارد؛ چرا که مجموعه‌ی مقادیر تابع هدف بی‌کران است. پس پاسخ بی‌نهایت یا تعریف نشده است.

(6_4)

$$\operatorname{argmin}_{x \in [-\infty, -1]} x^2 + 1$$

که به دنبال مقدار یا مقادیری از x در بازه $[-\infty,-1]$ است که تابع هدف x^2+1 را مینیمم می‌کند(مقدار مینیمم تابع اهمیتی ندارد). در این حالت جواب $x=-1$ است.

(7_4)

$\text{argmax}_{x\in[-5,5],\, y\in\mathbb{R}}\ x\cdot\cos(y)$

که به دنبال زوج مرتب(هایی) همچون (x,y) است که مقدار تابع هدف $x\cdot\cos(y)$ را ماکزیمم می‌کند با این قید که x در بازه [۵,۵-] باشد (مقدار ماکزیمم تابع اهمیتی ندارد). در این حالت، جواب‌ها، زوج مرتب‌هایی به شکل (kπ۵,۲) و (kπ+π۲و۵-) است، به‌طوری‌که k تمام اعداد صحیح را در برمی‌گیرد.

۴ـ۴: قضایا

قضایای اصلی مساله‌ی بهینه‌سازی را «قضایایی برای اثبات وجود نقاط بهینه»، «قضایایی برای یافتن این نقاط» و «قضایایی برای تحلیل حساسیت این نقاط نسبت به تغییر پارامترهای مساله» تشکیل می دهند. در زیر به مهم‌ترین آن‌ها اشاره می‌شود.

۴ـ۴ـ۱: وجود نقطه‌ی بهینه

قضیه‌ی مقدار بحرانی منتسب به کارل وایرشتراس شرایط وجود نقطه بهینه را بیان می‌کند.

اگر فضای جواب‌های ممکن، فشرده(ویا به طور هم ارز، در فضای اقلیدسی، بسته و کراندار باشد) و ناتهی باشد و از طرفی تابع هدف، پیوسته باشد، آنگاه برای تابع هدف، مقادیر ماکزیمم و مینیمم(بهینه)، در این مجموعه یافت خواهد شد.

۴ـ۵: کاربرد الگوریتم در مثال‌های ریاضی

۴ـ۵ـ۱: تابع سینوسی نامقید

اولین مثال عددی بهینه‌سازی یک مساله‌ی بیشینه سازی سینوسی نامقید با متغیرهای تصمیم پیوسته و بدون حضور متغیرهای حالت می‌باشد. در رابطه‌های (۹)،(۱۰)و(۱۱) به ترتیب تابع هدف و محدوده‌ی مجاز متغیرهای تصمیم ارائه گردید است و در شکل ۴ـ۱ شکل رویه‌ی این تابع به تصویر کشیده شده‌است.

(8_4)

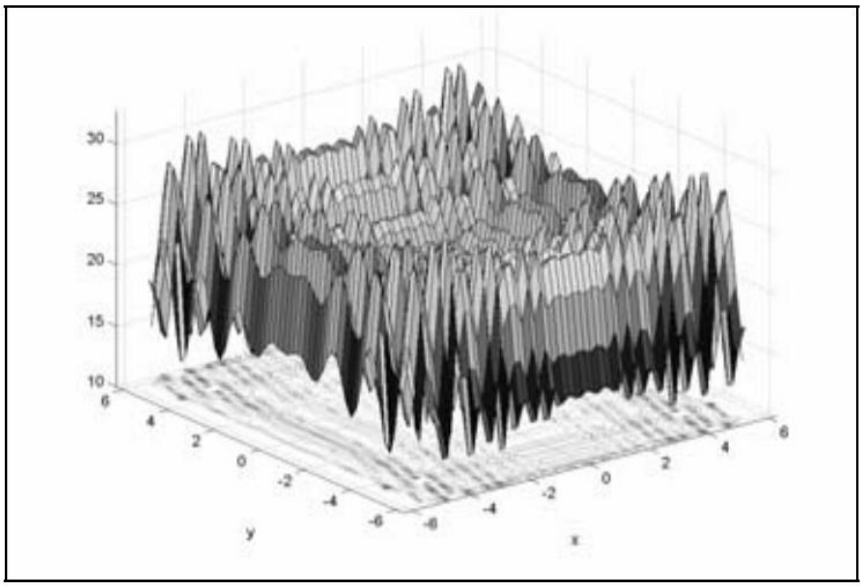
$$Max\ f(x_1,x_2)=21.5+x_1\sin(4\pi x_1)+x_2\sin(20\pi x_2)$$

$$-3\le x_1\le 12$$

$$4.1\le X_2\le 5$$

این تابع دارای دو متغیر تصمیم می‌باشد. از دیگر مشخصات این تابع می‌توان به چند قله‌ای[1] بودن و تفکیک ناپذیر بودن[2] آن اشاره کرد. با حل این مساله توسط روش الگوریتم ژنتیک بهترین جواب حاصل به شکل زیر می‌باشد.

F(11/631407,5/724824)=38/818208



شکل 4-1: رویه‌ي تابع سینوسي نامقید

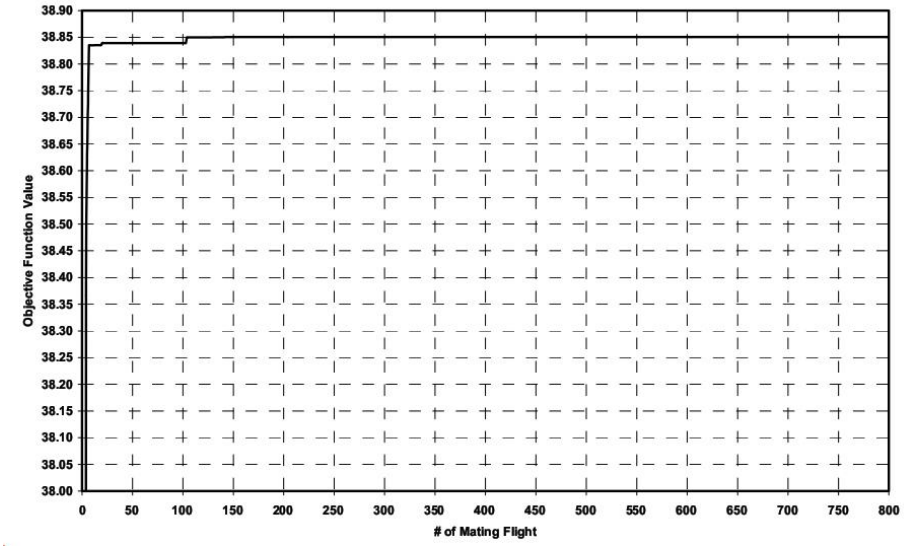
با کابرد الگوریتم HBMO در حل مثال فوق نتایج حاصل از همگرایی جواب که تغیرات مقدار تابع هدف در طول انجام پروازهای جفت‌گیری می‌باشد در شکل ۴ـ۲ ارائه شده است .همان‌گونه که ملاحظه می‌گردد، در حدود ۱۰۰ پرواز جفت‌گیری که حدود ۲۲۰۰۰ ارزیابی تابع می‌باشد مقدار تابع هدف به میزان قابل توجهی به جواب بهینه همگرا شده است که این خود بیانگر توانایی، قابلیت و سرعت زیاد الگوریتم در دست‌یابی سریع به جواب نزدیک به بهینه می‌باشد. در ادامه‌ی فرآیند و پس از انجام ۸۰۰ پرواز جفت‌گیری جواب مساله به سمت جواب بهینه همگرا شده و ثابت می‌ماند، لازم به توضیح است که جواب حاصل در این تعداد پرواز جفت‌گیری به شرح ذیل حاصل گردیده است:

F(11/62555.5/725043)=38/85029446

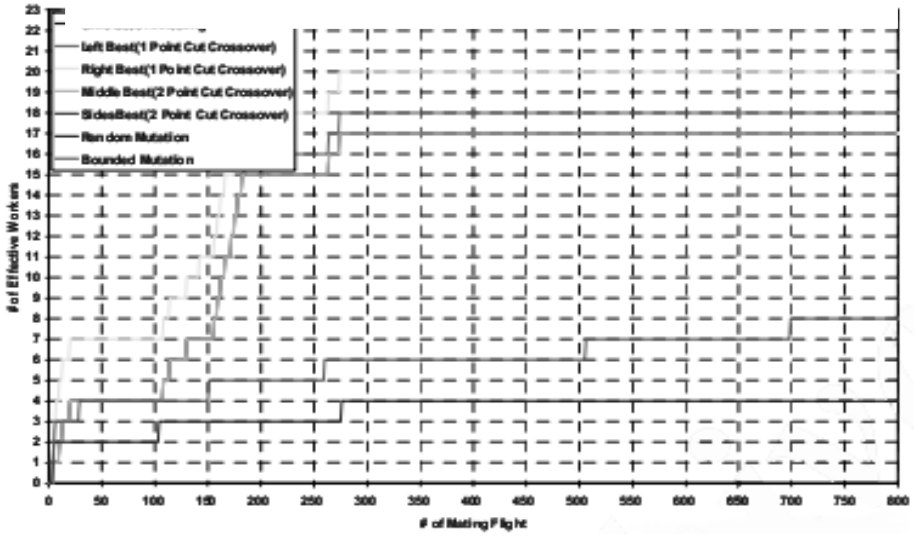
همان‌گونه که ملاحظه می‌گردد جواب مساله در حدود ۰/۰۸٪ بزرگ‌تر(مناسب‌تر) از جواب گزارش شده از کابرد الگوریتم ژنتیک در این مساله می‌باشد. شکل ۴ـ۳ بیانگر تعداد تجمعی موفقیت توابع در طول پروازهای جفت‌گیری می‌باشد. همان‌گونه که ملاحظه می‌گردد از انجام فرآیند نورد موفق، اثری دیده نمی‌شود و نوع تخم‌ریزی ملکه بیشترین و مهم‌ترین اثر و در ادامه، عملکرد زنبورهای کارگر دومین تاثیر را

1- Multi-Modality
2- Non-Separable

در پیشرفت و ترقی مجموعه‌ی جواب دارا می‌باشد. همچنین در جدول شکل ۴ـ۴ مقادیر تابع هدف و هریک از دو متغیر موجود در مساله در ۱۰بار اجرای برنامه و در پایان ۸۰۰ پرواز جفت‌گیری ارائه گردیده است. از طرفی در شکل ۴ـ۲، پارامترهای آماری این مقادیر ارائه گردیده است.



شکل 4-2: تغییرات مقدار تابع هدف در طول پروازهای جفت‌گیری



شکل 4-3: تعداد تجمعی موفقیت توابع در طول پروازهای جفت‌گیری

	شماره اجرا									
	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰
تابع هدف	۳۸.۸۵۰۲۹۴۱	۳۸.۸۵۰۲۹۲۴	۳۸.۸۵۰۲۷۲۳	۳۸.۸۵۰۲۹۳۸	۳۸.۸۵۰۲۹۴۴	۳۸.۸۵۰۲۹۲۵	۳۸.۸۵۰۲۹۱۰	۳۸.۸۵۰۲۹۴۳	۳۸.۸۵۰۲۸۷۴	۳۸.۸۵۰۲۹۱۶
متغیر اول	۱۱.۶۲۵۵۶۰۰	۱۱.۶۲۵۵۰۰۰	۱۱.۶۲۵۷۰۰۰	۱۱.۶۲۵۵۱۹۸	۱۱.۶۲۵۵۵۰۰	۱۱.۶۲۵۵۳۳۴	۱۱.۶۲۵۴۸۴۱	۱۱.۶۲۵۵۴۳۷	۱۱.۶۲۵۴۶۹۰	۱۱.۶۲۵۶۰۰۰
متغیر دوم	۵.۷۲۵۰۴۰۰	۵.۷۲۵۰۴۰۰	۵.۷۲۵۰۴۳۵	۵.۷۲۵۰۴۷۰	۵.۷۲۵۰۴۲۸	۵.۷۲۵۰۳۱۳	۵.۷۲۵۰۴۷۹	۵.۷۲۵۰۴۰۰	۵.۷۲۵۰۳۱۷	۵.۷۲۵۰۴۶۰

جدول 4-1: مقادیر تابع هدف و دومتغیر تصمیم در 10 اجرا

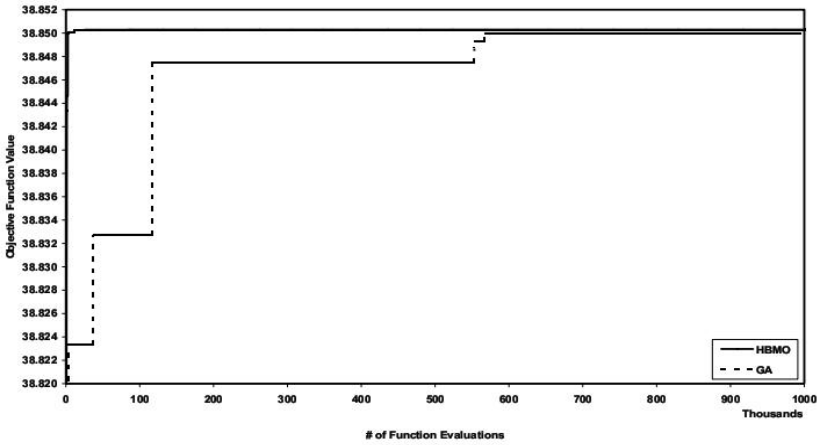
	کمینه	بیشینه	متوسط	انحراف معیار	ضریب تغییرات
مقدار تابع هدف	۳۸.۸۵۰۲۷۲۳	۳۸.۸۵۰۲۹۴۴	۳۸.۸۵۰۲۹۰۴	۰.۰۰۰۰۰۶۶۸۱۵	۰.۰۰۰۰۰۰۱۷۲۰
مقدار متغیر اول	۱۱.۶۲۵۴۶۹۰	۱۱.۶۲۵۷۰۰۰	۱۱.۶۲۵۵۴۶۰	۰.۰۰۰۰۶۶۳۰۶۴	۰.۰۰۰۰۰۵۷۰۳۵
مقدار متغیر دوم	۵.۷۲۵۰۳۱۳	۵.۷۲۵۰۴۷۹	۵.۷۲۵۰۴۱۰	۰.۰۰۰۰۰۵۷۸۳۰	۰.۰۰۰۰۰۱۰۱۰۱

جدول 4-2: پارامترهاي آماري تابع هدف و دومتغير تصميم در 10 اجرا 800 پرواز جفتگيري

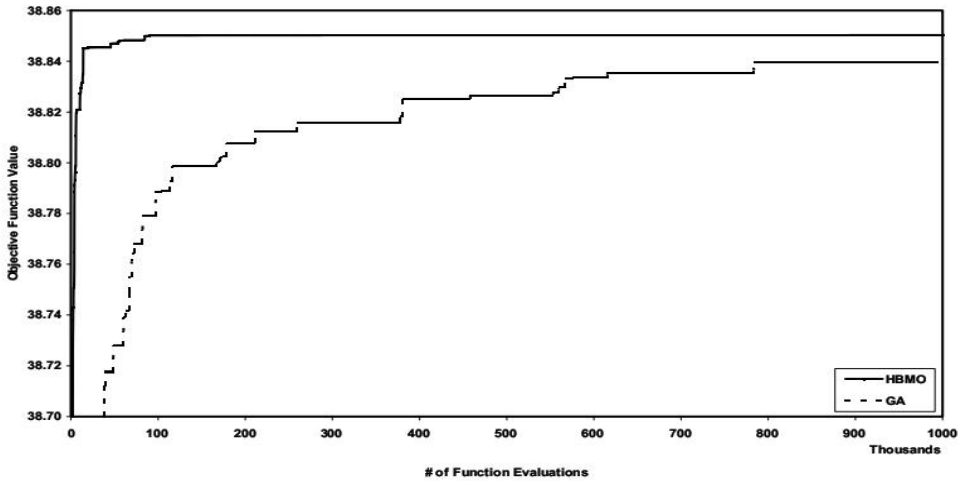
همان‌گونه که ملاحظه می‌گردد مقادیر تابع هدف و متغیرهای حاصل در پایان این ۱۰ اجرا بسیار به یک‌دیگر نزدیک بوده و دارای انحراف معیار و ضریب تغییرات بسیار کوچکی می‌باشد. همان‌گونه که ملاحظه می‌گردد. نزدیکی میزان مقادیر تابع هدف و متغیرها، امیدواری‌هایی را در استفاده از الگوریتم مذکور فراهم می‌آورد، زیرا که این مطلب بیانگر قابلیت الگوریتم در نزدیکی به جواب بهینه‌ی مطلق حتی در حل مسائل دارای چندین بیشینه و کمینه‌ی موضعی می‌باشد. جهت مقایسه‌ی الگوریتم HBMO با سایر روش‌های فراکاوشی و تحولی موجود، یک الگوریتم ژنتیک بر پایه‌ی نخبه‌گرایی تهیه گردیده و مورد آزمون قرار گرفته‌است. باتوجه به حساسیت الگوریتم ژنتیک به مقادیر احتمالات برش ژنی و جهش ژنی و نیز تعداد نقاط برش ژنی، و تاثیر قابل توجه این مقادیر بر نتایج حاصل از اجرای مدل توسط الگوریتم ژنتیک لازم است که بر روی مقادیر مختلف این موارد آنالیز حساسیت صورت گیرد. لذا در این مساله، الگوریتم ژنتیک در حالات مختلف این مقادیر و با ۱۰ اجرا برای هر حالت مورد بررسی قرارگرفته است. نتایج و پارامترهای آماری حاصل در جدول شکل ۴ـ۶ ارائه گریده است. با توجه به نتایج حاصل در نهایت مقادیر ۸۰٪ برای احتمال برش ژنی و ۴۰٪ برای احتمال جهش ژنی انتخاب گردیده است .

احتمال برش ژنی (٪)	احتمال جهش ژنی (٪)	حداقل	متوسط	حداکثر	انحراف معیار	ضریب تغییرات
۴۰	۴۰	۳۸.۶۹۲	۳۸.۷۷۵	۳۸.۸۴۲	۰.۰۵۵	۰.۰۰۱۴
۴۰	۵۰	۳۸.۶۰۹	۳۸.۷۴۳	۳۸.۸۴۷	۰.۰۸۴	۰.۰۰۲۲
۴۰	۶۰	۳۸.۷۱۹	۳۸.۷۷۷	۳۸.۸۴۹	۰.۰۴۷	۰.۰۰۱۲
۶۰	۴۰	۳۸.۷۴۰	۳۸.۷۹۴	۳۸.۸۴۳	۰.۰۳۹	۰.۰۰۱۰
۶۰	۵۰	۳۸.۶۴۷	۳۸.۷۵۶	۳۸.۸۵۰	۰.۰۶۹	۰.۰۰۱۸
۶۰	۶۰	۳۸.۶۶۵	۳۸.۷۱۷	۳۸.۷۸۳	۰.۰۴۰	۰.۰۰۱۰
۸۰	۴۰	۳۸.۷۴۲	۳۸.۸۰۴	۳۸.۸۵۰	۰.۰۴۱	۰.۰۰۱۱
۸۰	۵۰	۳۸.۷۰۰	۳۸.۷۸۷	۳۸.۸۴۶	۰.۰۵۰	۰.۰۰۱۳
۸۰	۶۰	۳۸.۷۱۰	۳۸.۷۷۷	۳۸.۸۲۴	۰.۰۳۴	۰.۰۰۰۹

جدول 4-3: پارامترهاي آماري مقادير تابع هدف در 10 اجرا توسط الگوريتم ژنتيك با احتمالات مختلف



شكل 4-4: تغييرات حداكثر مقدار تابع هدف در 10 بار



شکل 4-5: تغییرات متوسط مقدار تابع در 10 بار اجرا و در طول دفعات ارزیابی تابع هدف

۴-۵-۲: تابع توانی مقید

دیگر مثال مورد بحث در این قسمت یک تابع بهینه‌سازی غیرخطی از نوع کمینه‌سازی و با دو متغیر تصمیم است که به صورت پیوسته تعریف شده‌اند. این مساله یک مثال مقید با دو قید است که هریک از آن‌ها به صورت یک رویه تعریف شده‌اند. مثال مورد بحث یک مثال تک قله‌ای و تفکیک ناپذیر میباشد و تابع هدف، قیودات مساله و محدوده‌ی مجاز متغیرهای تصمیم در روابط زیر ارائه گردیده‌اند:

(9_4)

$$0 \leq x_2 \leq 6$$

$$0 \leq x_1 \leq 6$$

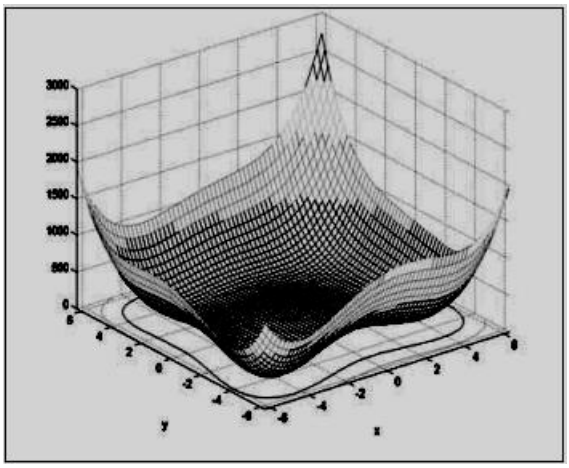
$$\mathrm{Min} f_1 \quad (x_1, \mathrm{x}_2) = (x_1^2 + \mathrm{x}_2 - 11)^2 + (\mathrm{x}_1 + \mathrm{x}_2^2 - 7)^2$$

S.T:

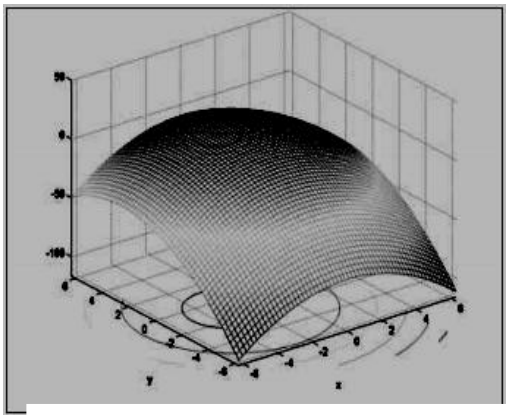
$$\mathrm{g}_1(\mathrm{x}) \equiv \ (5.059 - x_1^2 - (\mathrm{x}_2 - 2.5)^2 \geq 0$$

$$\mathrm{g}_2(x) \equiv (\mathrm{x}_1 - 0.05)^2 + (\mathrm{x}_2 - 2.5)^2 - 4.84 \geq 0$$

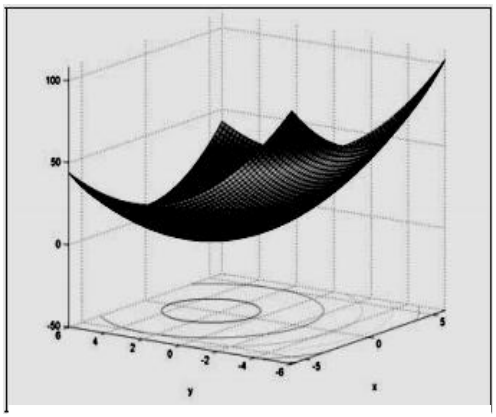
شکل رویه‌ی تابع و قیود آن در شکل ۴ـ۶ و شکل رویه‌ی قیودات مساله نیز به ترتیب در شکل های ۴ـ۷ و ۴ـ۸ به نمایش در آمده‌اند:



شکل 4-6: رویه‌ي تابع توانی مقید



شکل 4-7: رویه‌ي قید $g_1(\mathbf{x})$

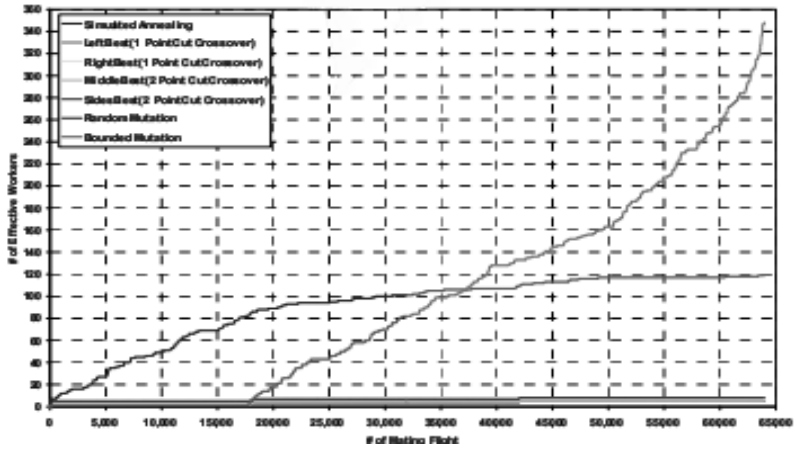


شکل 4-8: رویه‌ي قید $g_2(\mathbf{x})$

تابع هدف نامقید $f_1(x_1, x_2)$ دارای جواب کمینه‌ی مطلق [illegible] با مقدار صفر در این نقطه می‌باشد. اما با توجه به حضور قیدهای فوق، جواب مذکور ناشدنی بوده و جواب بهینه در نقطه‌ی $x^* = (2.246826, 2.381865)$ و با تابع هدفی برابر با f_1^* = ۱۳/۵۹۰۸۵ ظاهر می‌شود.

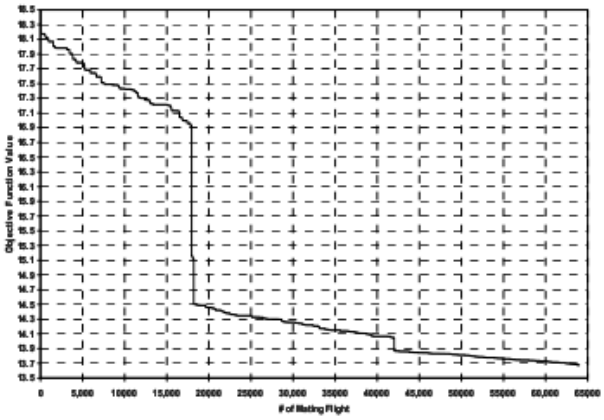
محدوده‌ی شدنی جواب ، یک محدوده‌ی بسیار باریک و در حدود ۰/۷٪ از کل فضای جستجو را شامل می‌شود. جواب بهینه در شرط دوم مساله نهفته است.

با کاربرد الگوریتم HBMO میزان تغییرات تابع هدف در طول پروازهای جفت‌گیری در شکل ۴ـ۱۱ ارائه شده است. کاهش حاصل در محدوده‌ی پرواز جفت‌گیری ۱۸۰۰۰، به دلیل قرارگرفتن جواب در محدوده‌ی شدنی و کاهش مقدار تابع هدف به دلیل از بین رفتن مقدار جریمه می باشد.



شکل 4-9: تغییرات مقدار تابع هدف در طول پروازهای جفت‌گیری

جهت ارائه‌ی نمایشی از وضعیت و عملکرد توابع مورد استفاده در این مثال مقدار تجمعی توابع موثر در طول انجام پروازهای جفت‌گیری در شکل ۴ـ۱۲ ارائه شده است.



شکل 4-10: تعداد تجمعي موفقيت توابع در طول انجام پروازهاي جفتگيري

ملاحظه می‌گردد که توابع نورد و انواع روش های تولیدمثل ملکه دارای تاثیرات و عملکرد بسیار ضعیفی نسبت به عملکرد کارگرها در طول پروازهای جفت‌گیری می‌باشد. شایان ذکر است که عملکرد جهش تصادفی تا حدود پرواز جفت‌گیری ۱۳۰۰۰ دارای روند صعودی مناسبی می‌باشد اما عملکرد جهش محصور تا این لحظه بسیار کم می‌باشد اما پس از آن و با قرار گرفتن جواب بهینه در محدوده‌ی شدنی جواب عملکرد جهش تصادفی کمرنگ‌تر گردیده و جهش محصور با یک روند صعودی شدید به بهبود وضعیت جواب می‌پردازد. مقدار تابع هدف حاصل از الگوریتم حاضر پس از ۶۵۰۰۰ پرواز جفت‌گیری برابر با ۱۳/۶۲۳۰۵ می‌باشد که حدود ۰/۲٪ با جواب بهینه اختلاف دارد. نتایج حاصل از کاربرد الگوریتم HBMO در ۱۰ بار اجرای برنامه در جدول شکل ۴ـ۱۳ و نتایج آماری این ۱۰ اجرا در جدول شکل ۴ـ۱۴ ارائه شده است.

	شماره اجرا									
	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰
تابع هدف	۱۴.۳۴۹۶۳۰۵	۱۴.۴۳۳۴۴۱۵	۱۴.۳۶۵۳۰۱۶	۱۴.۶۲۲۶۳۹۰	۱۳.۸۶۷۰۲۸۵	۱۳.۹۹۷۳۴۳۶	۱۳.۹۰۸۳۸۶۱	۱۴.۱۸۷۴۳۳۳	۱۳.۹۸۱۵۷۷۹	۱۳.۷۷۱۹۵۴۹
متغیر اول	۲.۲۳۳۴۵۷۱	۲.۲۳۲۴۷۵۹	۲.۲۳۳۲۴۰۰	۲.۲۳۰۱۹۱۳	۲.۲۴۰۰۴۱۵	۲.۲۳۸۱۱۰۲	۲.۲۳۹۳۹۷۹	۲.۲۳۵۵۳۶۰	۲.۲۳۸۳۰۰۰	۲.۲۴۱۶۳۷۲
متغیر دوم	۲.۲۲۸۹۵۱۷	۲.۲۲۰۳۹۷۳	۲.۲۲۷۵۰۰۰	۲.۲۰۳۲۲۱۸	۲.۲۸۹۹۳۵۵	۲.۲۷۰۰۹۳۸	۲.۲۸۳۲۹۸۱	۲.۲۴۶۲۷۶۰	۲.۲۷۲۶۰۰۰	۲.۳۰۷۳۲۷۲
مقدار تابع هدف پس از ۶۵۰۰۰ پرواز جفت گیری						۱۳/۶۲۳۰۵				

جدول 4-4: مقادیرتابع هدف و دو متغیر تصمیم در 10 اجرا و 800 پرواز جفتگیري

	کمینه	بیشینه	متوسط	انحراف معیار	ضریب تغییرات
تابع هدف	۱۳.۷۷۱۹۵۴۹	۱۴.۶۲۲۶۳۹۰	۱۴.۱۴۸۴۷۳۷	۰.۲۸۳۶۹۵۸۴۵۵	۰.۰۲۰۰۵۱۳۳۹۲
متغیر اول	۲.۲۳۰۱۹۱۳	۲.۲۴۱۶۳۷۲	۲.۲۳۶۲۳۸۷	۰.۰۰۳۷۹۰۲۰۰۸	۰.۰۰۱۶۹۴۸۹۹۹
متغیر دوم	۲.۲۰۳۲۲۱۸	۲.۳۰۷۳۲۷۲	۲.۲۵۴۹۶۰۱	۰.۰۳۴۴۵۷۹۹۱۲	۰.۰۱۵۲۸۰۹۷۵۸

جدول 4-5: پارامترهاي آماري تابع هدف و دو متغیر تصمیم در 10 اجرا ودر 800 پرواز جفتگیری

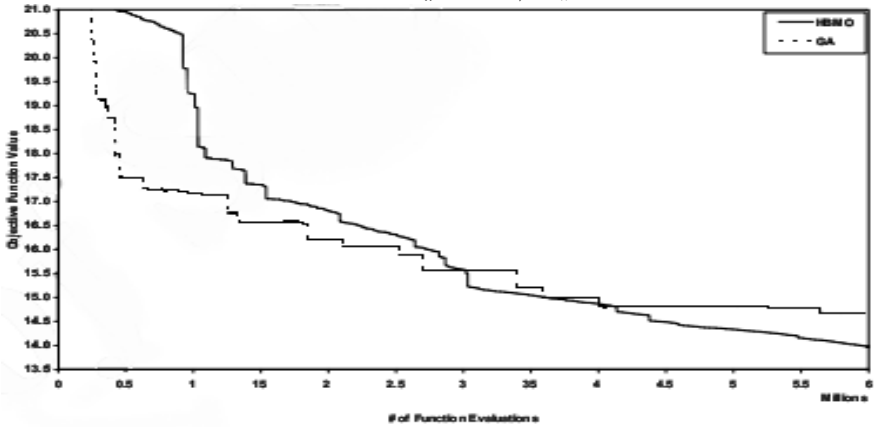
ملاحظه می‌گردد که مقادیر تابع هدف در هریک از دومتغیر و پس از ۶۰۰۰ پرواز جفت‌گیری و درپایان ۱۰ اجرا دارای انحراف معیار و ضریب تغییرات کوچکی بوده و کلیه‌ی ۱۰ اجرا به نحو مناسبی به جواب بهینه همگرا گردیده‌اند به گونه‌ای که بهترین جواب ۱/۳٪ و بدترین جواب ۷/۶٪ با جواب بهینه اختلاف دارند.

در این مساله نیز الگوریتم ژنتیک در حالات مختلف این مقادیر و با ۱۰ اجرا برای هرحالت مورد بررسی قرار گرفته است. نتایج و پارامترهای آماری حاصل از این ۱۰ اجرا در جدول شکل ۴-۱۵ ارائه گردیده است. با توجه به نتایج حاصل در نهایت مقادیر ۴۰٪ برای احتمال برش ژنی و ۶۰٪ برای احتمال جهش ژنی انتخاب گردیده است.در ادامه با قرارگیری مقادیر مطلوب احتمالات برش و جهش ژنی در الگوریتم و اجرای مجدد الگوریتم با تعداد زیادتر نسل در ۱۰ اجرا نتایجی حاصل گردیده است. در شکل‌های ۴ـ۱۱و ۴ـ۱۲ به ترتیب نحوه‌ی تغییرات تابع هدف(روند همگرایی) مقادیر متوسط و حداقل ۱۰ اجرا برحسب تعداد دفعات ارزیابی تابع هدف برای بار الگوریتم ژنتیک و الگوریتم HBMO ارائه شده است. همان‌گونه که ملاحظه می‌شود چه در مقادیر حداقل و چه درمقادیر متوسط در ابتدای حل که احتمال عدم ارضای قیود وجود دارد الگوریتم HBMOجواب‌های ضعیف‌تری را نسبت به الگوریتم ژنتیک نشان می‌دهد. اما با پیشرفت روند حل

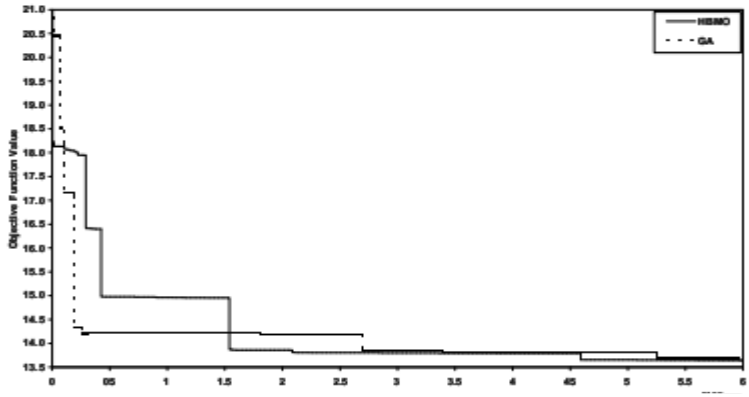
وقرارگرفتن جواب در محدوده شدنی، الگوریتم HBMO به سرعت به جواب بهینه نزدیک شده و از جواب حاصل از GA سبقت میگیرد.

احتمال برش ژنی (٪)	احتمال جهش ژنی (٪)	حداقل	متوسط	حداکثر	انحراف معیار	ضریب تغییرات
۴۰	۴۰	۱۸.۱۳۰	۲۳.۱۳۲	۳۳.۶۰۵	۵.۶۵۷	۰.۲۴۵
۴۰	۵۰	۱۸.۹۲۰	۲۵.۲۴۶	۳۵.۱۷۷	۵.۳۳۳	۰.۲۱۱
۴۰	۶۰	۱۴.۵۰۸	۱۸.۵۴۳	۲۲.۴۶۸	۲.۸۵۰	۰.۱۵۴
۶۰	۴۰	۱۴.۳۳۴	۲۵.۴۱۶	۳۶.۲۶۷	۷.۵۴۷	۰.۲۹۷
۶۰	۵۰	۱۴.۱۹۸	۲۰.۲۰۲	۳۳.۴۸۸	۵.۶۱۰	۰.۲۷۸
۶۰	۶۰	۱۵.۷۹۸	۲۰.۰۴۲	۲۴.۲۴۹	۲.۹۶۸	۰.۱۴۸
۸۰	۴۰	۲۰.۰۱۰	۲۷.۲۸۷	۳۵.۵۲۸	۵.۶۸۱	۰.۲۰۸
۸۰	۵۰	۱۶.۱۹۵	۲۴.۳۹۰	۳۲.۵۱۵	۴.۷۰۰	۰.۱۹۳
۸۰	۶۰	۱۳.۸۲۸	۱۹.۷۰۲	۲۵.۰۲۳	۴.۱۵۷	۰.۲۱۱

جدول 4-6: پارامترهاي آماري مقادير تابع هدف در 10 بار اجرا توسط الگوريتم ژنتيك با احتمالات مختلف



شکل 4-11: تغییرات متوسط مقادیر تابع هدف در 10 بار اجرا و در طول تعداد دفعات ارزیابی توابع



شکل 4-12: تغییرات حداقل مقادیر تابع هدف در 10 اجرا و در طول تعداد دفعات ارزیابی

۴-۶: ارزیابی الگوریتم

برای اثبات شایستگی الگوریتم بد نیست بدانید که الگوریتم فوق در پیچیده‌ترین توابع ریاضی مورد بررسی قرار گرفته است ونتایج حاصل از آن به موفق بودن الگوریتم رای داده**اند.**

۴ـ۶ـ۱: تابع Griewank

این تابع درمینیمم جهانی خود دارای مقدار صفر است. محدوده ی اولیه برای تابع (۲و۲-) است. تابع Griewank اصطلاحی است که وابستگی متقابل بین متغیرها را تولید می کند . هدف غلبه بر شکست تکنیکهایی که هر متغیر را به طور مستقل بهینه‌سازی می‌کند. بهینه‌ی تابع Griewank به طور منظم توزیع شده است.

(10_4)

$$f_1(\vec{x}) = \frac{1}{4,000}\left(\sum_{i=1}^{D}(x_i^2)\right) - \left(\prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}})\right) + 1$$

۴ـ۶ـ۲: تابع Rastrigin

این تابع هم درمینیمم جهانی خود دارای مقدار صفر است. محدوده‌ی اولیه برای تابع(۲و۲-) است. این تابع مبتنی بر تابع Sphere، به علاوه ی مدولاسیون کسینوس، مینیمم محلی بسیاری را تولید می کند. بنابراین تابع مرکب است. نقاط مینیمم به طور منظم توزیع شده‌است. قسمت دشوار در پیدا کردن راه حل‌های بهینه در این تابع این است که یک الگوریتم بهینه‌سازی به راحتی می‌تواند به سمت بهینه‌ی جهانی شدن در بهینه‌ی محلی به دام بیفتد.

(11_4)

$$f_2(\vec{x}) = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10)$$

۴ـ۶ـ۳: تابع Rosenbrock

این تابع نیز دارای مقدارصفر در مینیمم جهانی خود است. محدوده ی اولیه برای تابع (۲و۲-) است. بهینه‌ی جهانی در داخل دره‌ی ژرف و باریک به شکل سهمی می‌باشد. از آنجا که همگراشدن بهینه‌ی جهانی مشکل است، متغیرها به شدت وابسته هستند وسطح شیب دار به طور کلی به سمت نقطه ی مطلوب نیست. این مساله‌ای است که بارها و بارها برای آزمایش کردن عملکرد الگوریتم های بهینه‌سازی مورد استفاده قرار گیرد.

(12_4)

$$f_3(\vec{x}) = \sum_{i=1}^{D} 100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2.$$

۴ـ۶ـ۴: تابع Ackley

این تابع هم درمینیمم جهانی مقدارش صف است. محدوده‌ی اولیه برای تابع(۲و۲-) است. Ackley یک اصطلاح نمایی است که سطح خود را با مینیمم محلی متعدد پوشش می‌دهد. الگوریتمی که استفاده می‌شود در یک بهینه‌ی محلی به دام خواهد افتاد، اما هر راهبرد جستجو که منطقه ی گسترده‌تر را تجزیه و تحلیل می‌کند قادر به عبور از میان دره‌ی بهینه و دست‌یابی به نتایج بهتر خواهد بود

(13_4)

$$f_4(\vec{x}) = 20 + e - 20e^{\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right)} - e^{\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)}$$

۴ـ۶ـ۵: تابع Schwefel

این تابع در بهینه ی جهانی خود دارای مقدار صفر است. محدوده دهی اولیه برای تابع (۲و۲-) است. سطح تابع Schwefel از تعداد زیادی قله و دره تشک یل شده است. تابع بهترین مینیمم ثانویه به دور از

مینیمم جهانی دارد که در آن بسیاری از الگوریتم‌های جستجو به دام افتاده‌اند. علاوه بر این، مینیمم جهانی نزدیک مرزهای دامنه است.

(14_4)

$$f_5(\vec{x}) = D * 418.9829 + \sum_{i=1}^{D} -x_i \ \sin(\sqrt{|x_i|})$$

نتیجه گیری و پیشنهادات

با توجه به گوناگونی مسائل بهینه سازی اعم از تک هدفی یا چندهدفی، مقید یا نامقید، فضای جواب شدنی مقعر یا محدب، توابع هدف قابل یا غیرقابل جداسازی، متغیرهای پیوسته یا گسسته و...، لازم است درتوسعه‌ی هرالگوریتم و به خصوص در مراحل مقدماتی آن، چگونگی عملکرد و صحت الگوریتم در اکثر موارد فوق الذکر مورد ارزیابی و تایید قرار گیرد. ذکر این نکته ضروری است که با توجه به نوپابودن الگوریتم حاضر، امکان دست‌یابی به مقادیر بهتر معیارهای مذکور در آینده و با توسعه‌ی بیشتر الگوریتم دور از دسترس به نظر نمی‌رسد. گرچه نتایج حاضر نیز نشان از قابلیت‌های بالقوه‌ی الگوریتم داشته وتلاش در جهت توسعه‌ی آن را نمایان می‌سازد. تاکنون مطالعات اندکی جهت کاربرد رفتار زنبورهای عسل در دنیای واقعی بهینه‌سازی صورت گرفته است. در این پایان نامه الگوریتم زنبورهای عسل به عنوان یک الگوریتم بهینه‌سازی مدنظر قرار گرفته است و کاربرد این الگوریتم در چند مثال پایه‌ی بهینه‌سازی ریاضی مقید و نامقید با انواع پیچیدگی‌های موجود ارائه گردید تا کارایی الگوریتم در حل مثال‌های بهینه‌سازی ریاضی با انواع پیچیدگی‌های موجود مورد توجه قرار گیرد. نتایج حاصل از حل این مثال‌ها، گویای قابل قیاس بودن این الگوریتم با روش‌های دیگر، از جمله الگوریتم ژنتیک توسعه یافته می‌باشد.

همچنین کارایی مدل مذکور در حل مسائل موجود در مهندسی آب از قبیل مساله بهره‌برداری از مخازن سدها نیز آزمایش گردیده است که نتایج حاصل از آن بسیار رضایت‌بخش می باشد. در مجموع نتایج به دست آمده از حل مسائل مختلف توسط الگوریتم حاضر و مقایسه‌ی آن با نتایج حاصل از الگوریتم ژنتیک و روش برنامه‌ریزی پویا، نشان دهنده‌ی قابلیت بالای مدل جهت مطالعات بیشتر و کاربرد آن در دیگر مسائل بهینه‌سازی می‌باشد.

از مزایا و معایب این الگوریتم می توان به موارد زیر اشاره کرد:

مزایا:

۱. کارآمدی بسیار در پیدا کردن جوابی بهینه
۲. قادر به حل مسائل با جواب موضعی

معایب:

۱. از تعداد هماهنگی از متغیرها استفاده می کند

۲. پارامتر های وابسته ی کمی می توان تعریف کرد

پیشنهاد:

در این الگوریتم چون برای تولید فرزندان فقط یک مادر که همان ملکه است انتخاب می‌گردد و ملکه با تعدادی از افراد جمعیت که پدرها هستند با استفاده از اپراتور crossover فرزندان مختلف را تولید می‌کند در نتیجه تعداد ازدواج‌ها بسیار کمتر از الگوریتم ژنتیک معمولی است که این امر باعث می‌گردد تا سرعت این الگوریتم بسیار زیاد باشد. بنابراین پیشنهاد می‌گردد تا برای سامانه‌های مرتبه بالا صرفا از الگوریتم زنبورعسل استفاده گردد.

در انتها می‌توان به این نکته اشاره کرد که این الگوریم نوعی از الگوریتم‌هاست که به صورت کاوشی برای جستجوی جواب خود با توجه به محیط تعریف شده عمل می‌کند. یعنی این الگوریتم محدودیتی به یک حیطه‌ی علمی ندارد. تنها باید به این نکته توجه کرد که این الگوریتم را باید برای جستجو به کار برد و حتی می‌تواند در سطح حرفه‌ای در ساخت، رمزگشایی و بسیاری از مسائل دیگر نیز استفاده شود.

پیوست:

کد برنامه‌ی مربوط به الگوریتم زنبور عسل به زبان C

```
#include <stdlib.h>

#include <math.h>

#include <conio.h>

#include <time.h>

/* Control Parameters of ABC algorithm*/

#define NP 20 /* The number of colony size (employed bees+onlooker bees)*/

#define FoodNumber NP/2 /*The number of food sources equals the half of the colony size*/

#define limit 100  /*A food source which could not be improved through "limit" trials is abandoned by its employed bee*/

#define maxCycle 2500 /*The number of cycles for foraging {a stopping criteria}*/

/* Problem specific variables*/

#define D 100 /*The number of parameters of the problem to be optimized*/

#define lb -100 /*lower bound of the parameters. */

#define ub 100 /*upper bound of the parameters. lb and ub can be defined as arrays for the problems of which parameters have different bounds*/

#define runtime 30  /*Algorithm can be run many times in order to see its robustness*/

double Foods[FoodNumber][D]; /*Foods is the population of food sources. Each row of Foods matrix is a vector holding D parameters to be optimized. The number of rows of Foods matrix equals to the FoodNumber*/

double f[FoodNumber];  /*f is a vector holding objective function values associated with food sources */

double fitness[FoodNumber]; /*fitness is a vector holding fitness (quality) values associated with food sources*/

double trial[FoodNumber]; /*trial is a vector holding trial numbers through which solutions can not be improved*/

double prob[FoodNumber]; /*prob is a vector holding probabilities of food sources (solutions) to be chosen*/

double solution [D]; /*New solution (neighbour) produced by v_{ij}=x_{ij}+\phi_{ij}*(x_{kj}-x_{ij}) j is a randomly chosen parameter and k is a randomlu chosen solution different from i*/
```

```
double ObjValSol; /*Objective function value of new solution*/

double FitnessSol; /*Fitness value of new solution*/

int neighbour, param2change; /*param2change corrresponds to j, neighbour corresponds to k in equation v_{ij}=x_{ij}+\phi_{ij}*(x_{kj}-x_{ij})*/

double GlobalMin; /*Optimum solution obtained by ABC algorithm*/

double GlobalParams[D]; /*Parameters of the optimum solution*/

double GlobalMins[runtime]; /*GlobalMins holds the GlobalMin of each run in multiple runs*/

double r; /*a random number in the range [0,1)*/

/*a function pointer returning double and taking a D-dimensional array as argument */

/*If your function takes additional arguments then change function pointer definition and lines calling "...=function(solution);" in the code*/

typedef double (*FunctionCallback)(double sol[D]);

/*benchmark functions */

double sphere(double sol[D]);

double Rosenbrock(double sol[D]);

double Griewank(double sol[D]);

double Rastrigin(double sol[D]);

/*Write your own objective function name instead of sphere*/

FunctionCallback function = &sphere;

/*Fitness function*/

double CalculateFitness(double fun)
        double result=0;
        if(fun>=0)
        {
                result=1/(fun+1);
        }
        else
        {
                result=1+fabs(fun);
        }
```

```
            return result;
 }
/*The best food source is memorized*/
void MemorizeBestSource()
{
  int i,j;
            for(i=0;i<FoodNumber;i++)
            {
            if (f[i]<GlobalMin)
                        {
      GlobalMin=f[i];
      for(j=0;j<D;j++)
        GlobalParams[j]=Foods[i][j];
                          }
            }
 }

/*Variables are initialized in the range [lb,ub]. If each parameter has different range, use arrays lb[j], ub[j] instead of lb and ub */
/* Counters of food sources are also initialized in this function*/
void init(int index)
{
  int j;
  for (j=0;j<D;j++)
{
      r = (   (double)rand() / ((double)(RAND_MAX)+(double)(1)) );
      Foods[index][j]=r*(ub-lb)+lb;
                        solution[j]=Foods[index][j];
}
            f[index]=function(solution);
```

```
            fitness[index]=CalculateFitness(f[index]);
            trial[index]=0;
}

/*All food sources are initialized */
void initial()
{
            int i;
            for(i=0;i<FoodNumber;i++)
            {
            init(i);
            }
            GlobalMin=f[0];
   for(i=0;i<D;i++)
   GlobalParams[i]=Foods[0][i];
}
void SendEmployedBees()
{
  int i,j;
  /*Employed Bee Phase*/
   for (i=0;i<FoodNumber;i++)
        {
        /*The parameter to be changed is determined randomly*/
        r = ((double)rand() / ((double)(RAND_MAX)+(double)(1)) );
        param2change=(int)(r*D);
        /*A randomly chosen solution is used in producing a mutant solution of the solution i*/
        r = (   (double)rand() / ((double)(RAND_MAX)+(double)(1)) );
        neighbour=(int)(r*FoodNumber);
       /*Randomly selected solution must be different from the solution i*/
```

```
while(neighbour==i)
{  r = (   (double)rand() / ((double)(RAND_MAX)+(double)(1)) );
neighbour=(int)(r*FoodNumber);        }
for(j=0;j<D;j++)
solution[j]=Foods[i][j];
/*v_{ij}=x_{ij}+\phi_{ij}*(x_{kj}-x_{ij}) */
r = (   (double)rand() / ((double)(RAND_MAX)+(double)(1)) );
solution[param2change]=Foods[i][param2change]+(Foods[i][param2change]-
Foods[neighbour][param2change])*(r-0.5)*2;
/*if generated parameter value is out of boundaries, it is shifted onto the boundaries*/
if (solution[param2change]<lb)
  solution[param2change]=lb;
if (solution[param2change]>ub)
  solution[param2change]=ub;
ObjValSol=function(solution);
FitnessSol=CalculateFitness(ObjValSol);
/*a greedy selection is applied between the current solution i and its mutant*/
if (FitnessSol>fitness[i])
{    /*If the mutant solution is better than the current solution i, replace the solution with
the mutant and reset the trial counter of solution i*/
trial[i]=0;
for(j=0;j<D;j++)
Foods[i][j]=solution[j];
f[i]=ObjValSol;
fitness[i]=FitnessSol; }
else
{   /*if the solution i can not be improved, increase its trial counter*/
   trial[i]=trial[i]+1;  }
}
/*end of employed bee phase*/
```

```
}
/* A food source is chosen with the probability which is proportioal to its quality*/
/*Different schemes can be used to calculate the probability values*/
/*For example prob(i)=fitness(i)/sum(fitness)*/
/*or in a way used in the metot below prob(i)=a*fitness(i)/max(fitness)+b*/
/*probability values are calculated by using fitness values and normalized by dividing maximum fitness value*/
void CalculateProbabilities()
{
    int i;
    double maxfit;
    maxfit=fitness[0];
  for (i=1;i<FoodNumber;i++)
       {
         if (fitness[i]>maxfit)
         maxfit=fitness[i];
       }
for (i=0;i<FoodNumber;i++)
       {
        prob[i]=(0.9*(fitness[i]/maxfit))+0.1;
       }
}
void SendOnlookerBees()
{  int i,j,t;
  i=0;
  t=0;
  /*onlooker Bee Phase*/
  while(t<FoodNumber)
       {       r = (   (double)rand() / ((double)(RAND_MAX)+(double)(1)) );
       if(r<prob[i]) /*choose a food source depending on its probability to be chosen*/
```

```
{  t++;
/*The parameter to be changed is determined randomly*/
r = ((double)rand() / ((double)(RAND_MAX)+(double)(1)) );
param2change=(int)(r*D);
/*A randomly chosen solution is used in producing a mutant solution of the solution i*/
r = (   (double)rand() / ((double)(RAND_MAX)+(double)(1)) );
neighbour=(int)(r*FoodNumber);
/*Randomly selected solution must be different from the solution i*/
while(neighbour==i)
{
r = (   (double)rand() / ((double)(RAND_MAX)+(double)(1)) );
neighbour=(int)(r*FoodNumber);
}
for(j=0;j<D;j++)
solution[j]=Foods[i][j];
/*v_{ij}=x_{ij}+\phi_{ij}*(x_{kj}-x_{ij}) */
r = (   (double)rand() / ((double)(RAND_MAX)+(double)(1)) );
solution[param2change]=Foods[i][param2change]+(Foods[i][param2change]-
Foods[neighbour][param2change])*(r-0.5)*2;
/*if generated parameter value is out of boundaries, it is shifted onto the boundaries*/
if (solution[param2change]<lb)
  solution[param2change]=lb;
if (solution[param2change]>ub)
  solution[param2change]=ub;
ObjValSol=function(solution);
FitnessSol=CalculateFitness(ObjValSol);
/*a greedy selection is applied between the current solution i and its mutant*/
if (FitnessSol>fitness[i])
{     /*If the mutant solution is better than the current solution i, replace the solution with
the mutant and reset the trial counter of solution i*/
```

```
        trial[i]=0;
        for(j=0;j<D;j++)
        Foods[i][j]=solution[j];
        f[i]=ObjValSol;
        fitness[i]=FitnessSol;        }
        else
        {   /*if the solution i can not be improved, increase its trial counter*/
           trial[i]=trial[i]+1;        }
        } /*if */
        i++;
        if (i==FoodNumber-1)
        i=0;        }/*while*/
        /*end of onlooker bee phase     */
}
/*determine the food sources whose trial counter exceeds the "limit" value. In Basic ABC, only one scout is allowed to occur in each cycle*/
void SendScoutBees()
{int maxtrialindex,i;
maxtrialindex=0;
for (i=1;i<FoodNumber;i++)
        {
         if (trial[i]>trial[maxtrialindex])
         maxtrialindex=i;
        }
if(trial[maxtrialindex]>=limit)
{ init(maxtrialindex);
}}
/*Main program of the ABC algorithm*/
int main()
{int iter,run,j;
```

```
double mean;
mean=0;
srand(time(NULL));
for(run=0;run<runtime;run++)
{
initial();
MemorizeBestSource();
for (iter=0;iter<maxCycle;iter++)
   {   SendEmployedBees();
   CalculateProbabilities();
   SendOnlookerBees();
   MemorizeBestSource();
   SendScoutBees();    }
for(j=0;j<D;j++)
{printf("GlobalParam[%d]: %f\n",j+1,GlobalParams[j]);}
printf("%d. run: %e \n",run+1,GlobalMin);
GlobalMins[run]=GlobalMin;
mean=mean+GlobalMin;}
mean=mean/runtime;
printf("Means of %d runs: %e\n",runtime,mean);
getch();
}
double sphere(double sol[D])
{int j;
double top=0;
for(j=0;j<D;j++){
top=top+sol[j]*sol[j];}
return top;
}
```

```
double Rosenbrock(double sol[D])
{  int j;
double top=0;
for(j=0;j<D-1;j++)
{top=top+100*pow((sol[j+1]-pow((sol[j]),(double)2)),(double)2)+pow((sol[j]-
1),(double)2);}
return top;
}
 double Griewank(double sol[D])
 {
        int j;
        double top1,top2,top;
        top=0;
        top1=0;
        top2=1;
        for(j=0;j<D;j++)
        { top1=top1+pow((sol[j]),(double)2);
        top2=top2*cos((((sol[j])/sqrt((double)(j+1)))*M_PI)/180);   }
        top=(1/(double)4000)*top1-top2+1;
        return top; }
double Rastrigin(double sol[D])
 {
        int j;
        double top=0;
        for(j=0;j<D;j++)
        {top=top+(pow(sol[j],(double)2)-10*cos(2*M_PI*sol[j])+10);          }
        return top;
 }
```

فهرست منابع

۱. جوادی محمد - الگوریتم ژنتیک - انتشارات دانشگاه امام حسین(ع).
۲. نشریه بین المللی علوم مهندسی دانشگاه علم و صنعت ایران ـ شماره۲ـ جلد۱۹ـ تابستان ۱۳۸۷ .
۳. شاهمیری ا ـ الگوریتم‌ژنتیک ـ روزنامه جام‌جم- ضمیمه کلیک- شماره ۲۱۰ـ تهران ۱۳۸۷.
۴. باوری ا صالحی م ـ الگوریتم‌های‌ژنتیک و بهینه‌سازی سازه‌های مرکب ـ انتشارت عابد ـ تهران ۱۳۸۷.

5. Pham, D.T., Karaboga, D.: Intelligent Optimisation Techniques. Springer, London (2000)

6. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI (1975)

7. Benatchba, K., Admane, L., Koudil, M.: Using bees to solve a data-mining problem expressed (2005)

8. Ant Algorithms forDiscrete Optimization Marco Dorigo and Gianni Di Caro
IRIDIA, Universit´e Libre de BruxellesBrussels, Belgium{mdorigo,gdicaro}@ulb.ac.be
Luca M. GambardellaIDSIA, Lugano, Switzerlandluca@idsia.ch

9. Ant Algorithm for Grid Scheduling Problem Stefka Fidanova and Mariya Durchova
IPP – BAS, Acad. G. Bonchev, bl.25A, 1113 Sofia, Bulgaria stefka@parallel.bas.bg, mabs@parallel.bas.bg

10. An Ant Colony Optimization algorithm to solve a 2-machine bicriteria flowshop scheduling problem
Vincent T'kindt *, Nicolas Monmarche, Fabrice Tercinet, Daniel Laugt
Laboratoire d'Informatique, Ecole d'Ingenieurs en Informatique pour l'Industrie, 64 avenue Jean Portalis, 37200 Tours, France Received 1 October 2000; accepted 1 May 2001

11. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm Dervis Karaboga • Bahriye Basturk

12. www.pdfdatabase.com
13. www.matlabsit.com
14. www.4shared.com
15. www.ariabook.ir
16. www.mellifera.ir
17. www.mohammadnasri.com
18. www.gpwiki.com
19. http://artificial.ir/intelligence
20. www.itna.com
21. www.daneshju.ir
22. www.wikipedia.com

Abstract

Man has always looked for inspiration to the living world around. One of the best known projects, plans Lyvrnardv Prvazansan the first da Vinci (1519-1452) sketch of a flying machine based on body building will draw a bat. Four hundred years later the bird machine Klman address that was used with the engine and propellers instead of wings.

In recent decades, evolutionary methods and Frakavshy as a search tool and business optimization in various fields such as science and engineering has been used. Breadth of scope, ease of use and ability to achieve near optimal solution is absolute, including the reasons for this success.

Collective intelligence, the branch of artificial intelligence is based on the collective behavior of decentralized systems and self-organizing map is built. An example of collective intelligence, is the bee colony.

One application of this algorithm, multiple optimization problems is why some say the bees optimization algorithm. In this paper, bee colony algorithm used and the results produced by the algorithm are compared.

Subject of his bee colonies of bees mating process is divided into two parts and searching for food.

Keywords: evolutionary algorithms, collective intelligence, and the bee colony optimization

Payame Noor University

usage Honeybees algorithm in the optimization math problems

A Project Report

Presented to:

Department of Information Technology and Communication

Faculty of Engineering

Payame Noor University of Najafabad

In Partial Fulfillment of The Requirement for the degree of

Bachelor of science in

Information Technology Engineering

Advisor:

Mrs Elham Horry

By:

Fariborz Shamskia

September 2012