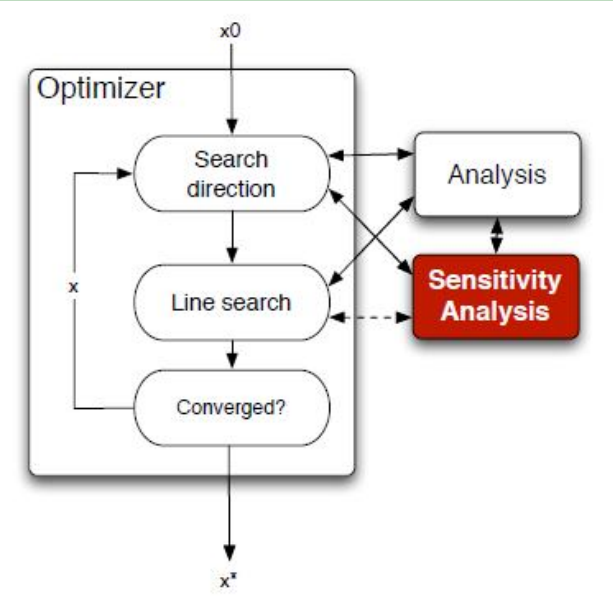
بهینه سازی و الگوریتم های موجود

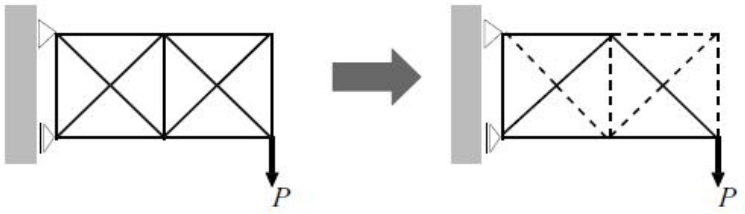
ایمان الیاسیان،دانشجوی دکترای عمران سازه

هدف از بهینه‌سازی یافتن بهترین جواب قابل قبول، با توجه به محدودیت‌ها و نیازهای مسأله است. برای یک مسأله، ممکن است جواب‌های مختلفی موجود باشد که برای مقایسه آنها و انتخاب جواب بهینه، تابعی به نام تابع هدف تعریف می‌شود. انتخاب این تابع به طبیعت مسأله وابسته است. به عنوان مثال، زمان سفر یا هزینه از جمله اهداف رایج بهینه‌سازی شبکه‌های حمل و نقل می‌باشد. به هر حال، انتخاب تابع هدف مناسب یکی از مهمترین گام‌های بهینه‌سازی است. گاهی در بهینه‌سازی چند هدف به طور همزمان مد نظر قرار می‌گیرد؛ این گونه مسائل بهینه‌سازی را که دربرگیرنده چند تابع هدف هستند، مسائل چند هدفی می‌نامند. ساده‌ترین راه در برخورد با این گونه مسائل، تشکیل یک تابع هدف جدید به صورت ترکیب خطی توابع هدف اصلی است که در این ترکیب میزان اثرگذاری هر تابع با وزن اختصاص یافته به آن مشخص می‌شود. هر مسأله بهینه‌سازی دارای تعدادی متغیر مستقل است که آنها را متغیرهای طراحی می‌نامند که با بردار n بعدی x نشان داده می‌شوند. هدف از بهینه‌سازی تعیین متغیرهای طراحی است، به گونه‌ای که تابع هدف کمینه یا بیشینه شود. مسائل مختلف بهینه‌سازی به دو دسته زیر تقسیم می‌شود:

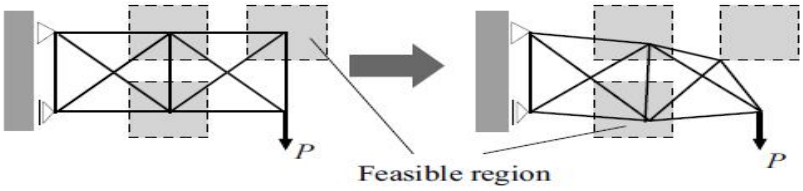
الف) مسائل بهینه‌سازی بی‌محدودیت: در این مسائل هدف، بیشینه یا کمینه کردن تابع هدف بدون هر گونه محدودیتی بر روی متغیرهای طراحی می‌باشد.

ب) مسائل بهینه‌سازی با محدودیت: بهینه‌سازی در اغلب مسائل کاربردی، با توجه به محدودیت‌هایی صورت می‌گیرد؛ محدودیت هایی که در زمینه رفتار و عملکرد یک سیستم می‌باشد و محدودیت‌های رفتاری و محدودیت‌هایی که در فیزیک و هندسه مسأله وجود دارد، محدودیت‌های هندسی یا جانبی نامیده می‌شوند.

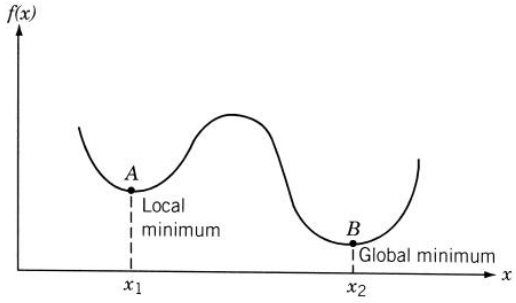




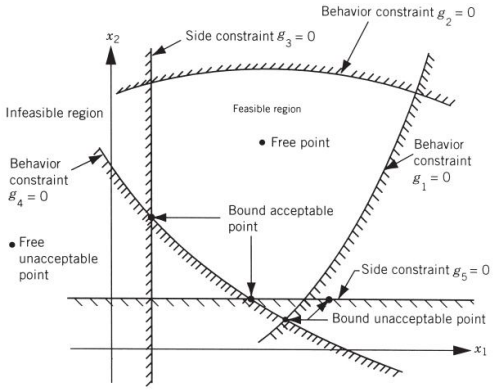
Topology optimization.



Geometry optimization.



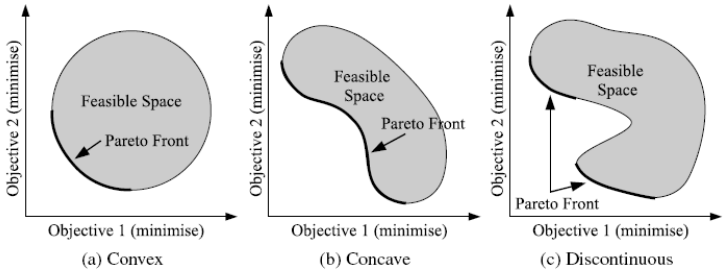
Local and global minima.



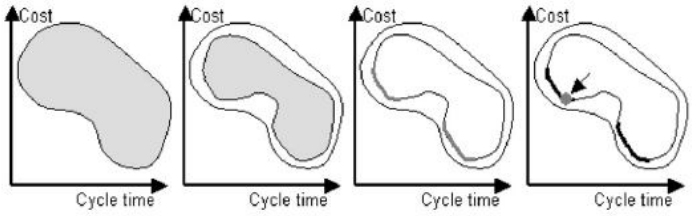
Constraint surfaces in a hypothetical two-dimensional design space.

معادلات معرف محدودیت‌ها ممکن است به صورت مساوی یا نامساوی باشند که در هر مورد، روش بهینه‌سازی متفاوت می‌باشد. به هر حال محدودیت‌ها، ناحیه قابل قبول در طراحی را معین می‌کنند. هدف از بهینه‌سازی یافتن بهترین جواب قابل قبول، با توجه به محدودیت‌ها و نیازهای مسأله است. برای یک مسأله، ممکن است جواب‌های مختلفی موجود باشد که برای مقایسه آنها و انتخاب جواب بهینه، تابعی به نام تابع هدف تعریف می‌شود. انتخاب این تابع به طبیعت مسأله وابسته است. به عنوان مثال، زمان سفر یا هزینه از جمله اهداف رایج بهینه‌سازی شبکه‌های حمل و نقل می‌باشد. به هر حال، انتخاب تابع هدف مناسب یکی از مهمترین گام‌های بهینه‌سازی است. گاهی در بهینه‌سازی چند هدف به طور همزمان مد نظر قرار می‌گیرد؛ این گونه مسائل بهینه‌سازی را که دربرگیرنده چند تابع هدف هستند، مسائل چند هدفی می‌نامند. ساده‌ترین راه در برخورد با این گونه مسائل، تشکیل یک تابع هدف جدید به صورت ترکیب خطی توابع هدف اصلی است که در این ترکیب میزان اثرگذاری هر تابع با وزن اختصاص یافته به آن مشخص می‌شود. هر مسأله بهینه‌سازی دارای تعدادی متغیر مستقل است که آنها را متغیرهای طراحی می‌نامند که با

بردار n بعدی x نشان داده می‌شوند.هدف از بهینه‌سازی تعیین متغیرهای طراحی است، به گونه‌ای که تابع هدف کمینه یا بیشینه شود. می‌باشد.



Representations of different Pareto fronts plotted in the objective space



Proposed optimal configuration and planning process.

روش‌های حل مسایل بهینه‌سازی

- روش‌های تصویری
- روش‌های تحلیلی یا روش‌های کلاسیک
- روش‌های خاص
- روش‌های عددی
- روش‌های برنامه‌ریزی دینامیکی

- روش‌های برنامه‌ریزی دینامیکی
- روش‌های مدرن یا مکاشفه‌ای(ابتکاری)

- بهینه‌سازی خطی Linear Programing
- بهینه‌سازی غیرخطی Nonlinear Programing
- بهینه‌سازی گسسته(شبکه) Discrete (Network) Optimization
- بهینه‌سازی کوادراتیک Quadratic Programing
- برنامه‌ریزی محدب Convex Programing

- بهینه‌سازی کمترین مربعات Least Square Programing
- برنامه‌ریزی دینامیکی Dynamic Programming
- بهینه‌سازی کنترل بهینه گسسته Discrete Optimal Control
- بهینه‌سازی تنک Sparse Optimization
- برنامه‌ریزی نیمه‌معین Semidefinite Programing
- برنامه‌ریزی نیمه‌متناهی Semi-infinite Programing
- بهینه‌سازی ناهموار Nonsmooth Optimization
- بهینه‌سازی سراسری Global Optimization
- بهینه‌سازی برداری Vector Optimization

بررسی روش‌های جستجو و بهینه‌سازی

پیشرفت کامپیوتر در طی پنجاه سال گذشته باعث توسعه روش‌های بهینه‌سازی شده، به طوری که دستورهای متعددی در طی این دوره تدوین شده است. در این بخش، مروری بر روش‌های مختلف بهینه‌سازی ارائه می‌شود.

روش‌های بهینه‌سازی را در چهار دسته وسیع دسته‌بندی می‌کند. در ادامه بحث، هر دسته از این روش‌ها مورد بررسی قرار می‌گیرند.

روش‌های شمارشی

در روش‌های شمارشی (Enumerative Method)، در هر تکرار فقط یک نقطه متعلق به فضای دامنه تابع هدف بررسی می‌شود. این روش‌ها برای پیاده‌سازی، ساده‌تر از روش‌های دیگر می‌باشند؛ اما به محاسبات قابل توجهی نیاز دارند. در این روش‌ها سازوکاری برای کاستن دامنه جستجو وجود ندارد و دامنه فضای جستجو شده با این روش خیلی بزرگ است. برنامه‌ریزی پویا (Dynamic

Programming) مثال خوبی از روش‌های شمارشی می‌باشد. این روش کاملاً غیرهوشمند است و به همین دلیل امروزه بندرت به تنهایی مورد استفاده قرار می‌گیرد.

روش‌های محاسباتی (جستجوی ریاضی یا- Based Method Calculus)

این روش‌ها از مجموعه شرایط لازم و کافی که در جواب مسأله بهینه‌سازی صدق می‌کند، استفاده می‌کنند. وجود یا عدم وجود محدودیت‌های بهینه‌سازی در این روش‌ها نقش اساسی دارد. به همین علت، این روش‌ها به دو دسته روش‌های با محدودیت و بی‌محدودیت تقسیم می‌شوند.روش‌های بهینه‌سازی بی‌محدودیت با توجه به تعداد متغیرها شامل بهینه‌سازی توابع یک متغیره و چند متغیره می‌باشند. روش‌های بهینه‌سازی توابع یک متغیره، به سه دسته روش‌های مرتبه صفر، مرتبه اول و مرتبه دوم تقسیم می‌شوند. روش‌های مرتبه صفر فقط به محاسبه تابع هدف در نقاط مختلف نیاز دارد؛ اما روش‌های مرتبه اول از تابع هدف و مشتق آن و روش‌های مرتبه دوم از تابع هدف و مشتق اول و دوم آن استفاده می‌کنند. در بهینه‌سازی توابع چند متغیره که کاربرد بسیار زیادی در مسائل مهندسی دارد، کمینه‌سازی یا بیشینه‌سازی یک کمیت با مقدار زیادی متغیر طراحی صورت می‌گیرد. یک تقسیم‌بندی، روش‌های بهینه‌سازی با محدودیت را به سه دسته برنامه‌ریزی خطی، روش‌های مستقیم و غیرمستقیم تقسیم می‌کند. مسائل با محدودیت که توابع هدف و محدودیت‌های آنها خطی باشند، جزو مسائل برنامه‌ریزی خطی قرار می‌گیرند. برنامه‌ریزی خطی شاخه‌ای از برنامه‌ریزی ریاضی است و کاربردهای فیزیکی، صنعتی و تجاری بسیاری دارد.در روش‌های مستقیم، نقطه بهینه به طور مستقیم جستجو شده و از روش‌های بهینه‌یابی بی‌محدودیت استفاده نمی‌شود. هدف اصلی روش‌های غیرمستقیم استفاده از الگوریتم‌های بهینه‌سازی بی‌محدودیت برای حل عمومی مسائل بهینه‌سازی با محدودیت می‌باشد.

در اکثر روش‌های محاسباتی بهینه‌یابی، از گرادیان تابع هدف برای هدایت جستجو استفاده می‌شود. اگر مثلاً به علت ناپیوستگی تابع هدف، مشتق آن قابل محاسبه نباشد، این روش‌ها اغلب با مشکل روبه‌رو می‌شوند.

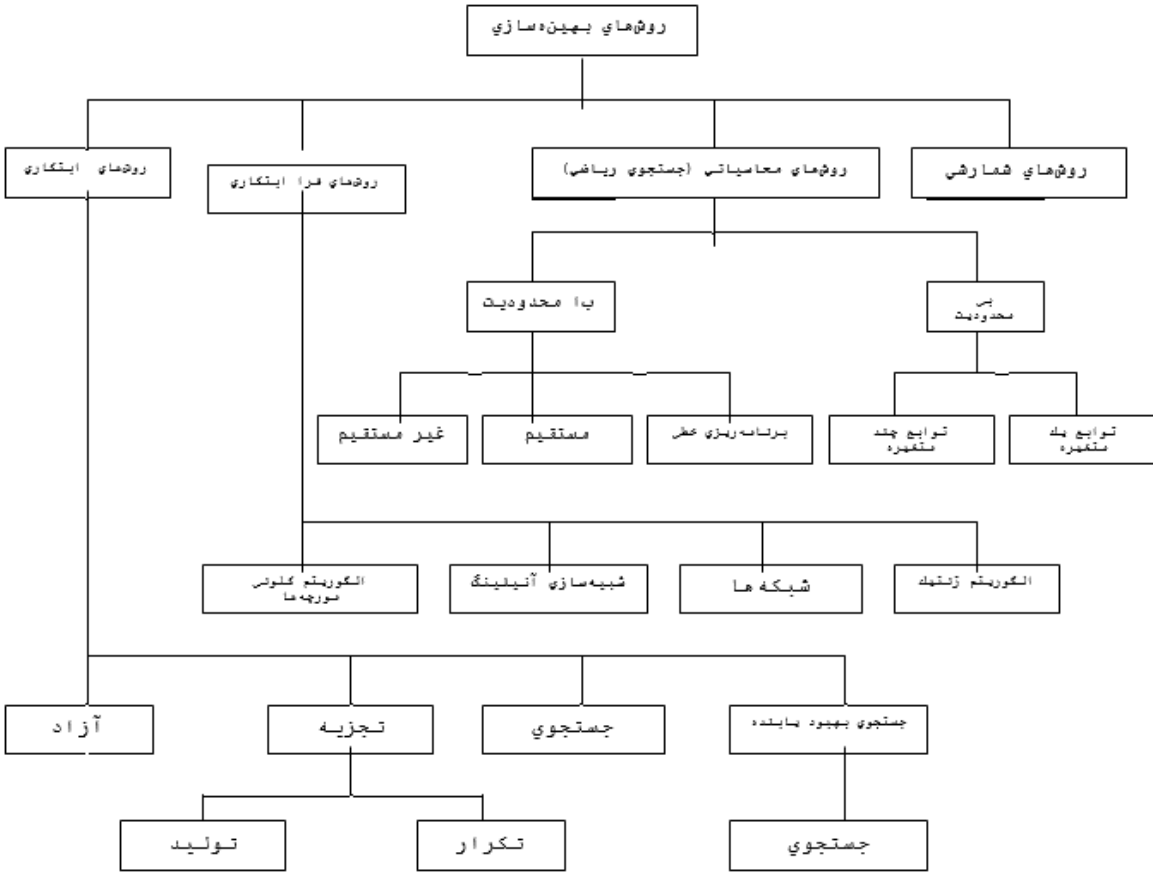
روش‌های ابتکاری و فرا ابتکاری (جستجوی تصادفی)

یک روش ناشیانه برای حل مسائل بهینه‌سازی ترکیبی این است که تمامی جواب‌های امکان‌پذیر در نظر گرفته شود و توابع هدف مربوط به آن محاسبه شود و در نهایت، بهترین جواب انتخاب گردد. روشن است که شیوه شمارش کامل، نهایتاً به جواب دقیق مسأله منتهی می‌شود؛ اما در عمل به دلیل زیاد بودن تعداد جواب‌های امکان‌پذیر، استفاده از آن غیرممکن است. با توجه به مشکلات مربوط به روش شمارش کامل، همواره بر ایجاد روش‌های مؤثرتر و کاراتر تأکید شده است. در این زمینه، الگوریتم‌های مختلفی به وجود آمده است که مشهورترین نمونه آنها، روش سیمپلکس برای حل برنامه‌های خطی و روش شاخه و کرانه برای حل برنامه‌های خطی با متغیرهای صحیح است. برای مسائلی با ابعاد بزرگ، روش سیمپلکس از کارایی بسیار خوبی برخوردار است، ولی روش شاخه و کرانه کارایی خود را از دست می‌دهد و عملکرد بهتری از شمارش کامل نخواهد داشت. به دلایل فوق، اخیراً تمرکز بیشتری بر روش‌های ابتکاری (Heuristic) یا فرا ابتکاری (Metaheuristic) یا جستجوی تصادفی (Random Method) صورت گرفته است. روش‌های جستجوی ابتکاری، روش‌هایی هستند که می‌توانند جوابی خوب (نزدیک به بهینه) در زمانی محدود برای یک مسأله ارائه کنند. روش‌های جستجوی ابتکاری عمدتاً بر مبنای روش‌های شمارشی می‌باشند، با این تفاوت که از اطلاعات اضافی برای هدایت جستجو استفاده می‌کنند. این روش‌ها از نظر حوزه کاربرد، کاملاً عمومی هستند و می‌توانند مسائل خیلی پیچیده را حل کنند. عمده این روش‌ها، تصادفی بوده و از طبیعت الهام گرفته شده‌اند.

مسائل بهینه‌سازی ترکیبی (Optimization Problems Combinational)

در طول دو دهه گذشته، کاربرد بهینه‌سازی در زمینه‌های مختلفی چون مهندسی صنایع، برق، کامپیوتر، ارتباطات و حمل و نقل گسترش یافته است. بهینه‌سازی خطی و غیرخطی (جستجو جهت یافتن مقدار بهینه تابعی از متغیرهای پیوسته)، در دهه پنجاه و شصت از اصلی‌ترین جنبه‌های توجه به بهینه‌سازی بود. بهینه‌سازی ترکیبی عبارت است از جستجو برای یافتن نقطه توابع با متغیرهای گسسته و در دهه ۷۰ نتایج مهمی در این زمینه به دست آمد. امروزه بسیاری از مسائل بهینه‌سازی ترکیبی (مانند مسأله فروشنده دوره‌گرد) که اغلب از جمله مسائل NP-hard هستند، به صورت تقریبی (نه به طور دقیق) در کامپیوترهای موجود قابل

حل می‌باشند. مسأله بهینه‌سازی ترکیبی را می‌توان به صورت زوج مرتب R,C نمایش داد که R مجموعه متناهی از جواب‌های ممکن (فضای حل) و C تابع هدفی است که به ازای هر جواب مقدار خاصی دارد.



روش حل مسائل بهینه‌سازی ترکیبی

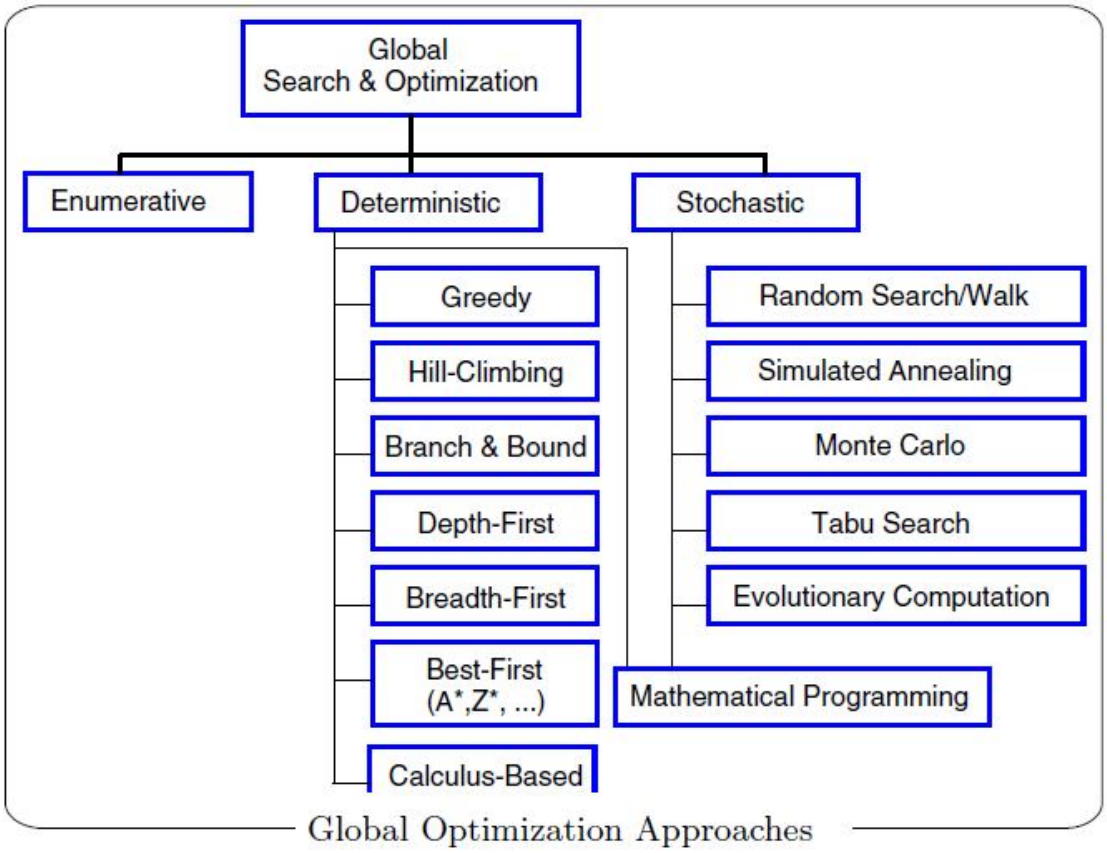
روشن است که شیوه شمارش کامل، نهایتاً به جواب دقیق مسأله منجر می‌شود؛ اما در عمل به دلیل زیاد بودن تعداد جواب‌های امکان‌پذیر، استفاده از آن بی‌نتیجه است. برای آنکه مطلب روشن شود، مسأله مشهور فروشنده دوره‌گرد (TSP) را در نظر می‌گیریم. این مسأله یکی از مشهورترین مسائل در حیطه بهینه‌سازی ترکیبی است که بدین شرح می‌باشد:

تعیین مسیر حرکت یک فروشنده بین N شهر به گونه‌ای که از هر شهر تنها یکبار بگذرد و طول کل مسیر به حداقل برسد، بسیار مطلوب است. تعداد کل جواب‌ها برابر است با ! . فرض کنید کامپیوتری موجود است که می‌تواند تمام جواب‌های مسأله با بیست شهر را در یک ساعت بررسی کند. بر اساس آنچه آورده شد، برای حل مسأله با ۲۱ شهر، ۲۰ ساعت زمان لازم است و به همین ترتیب، زمان لازم برای مسأله ۲۲ شهر، ۱۷/۵ روز و برای مسأله ۲۵ شهر، ۶ قرن ا ست!

به دلیل همین رشد نمایی زمان محاسبه، شمارش کامل روشی کاملاً نامناسب است.

همان طور که گفته شد، با توجه به مشکلات مربوط به روش شمارش کامل، همواره بر ایجاد روش‌های مؤثرتر و کاراتر تأکید شده است. در این زمینه، الگوریتم‌های مختلفی به وجود آمده که مشهورترین آنها، الگوریتم سیمپلکس برای حل برنامه‌های خطی و روش شاخه و کران برای حل برنامه‌های خطی با اعداد صحیح است.

بنابراین در سال‌های اخیر توجه بیشتری بر روش‌های ابتکاری برگرفته از طبیعت که شباهت‌هایی با سیستم‌های اجتماعی یا طبیعی دارد، صورت گرفته است و نتایج بسیار خوبی در حل مسائل بهینه‌سازی ترکیبی NP-hard به دنبال داشته است. در این الگوریتم‌ها هیچ ضمانتی برای آنکه جواب به دست آمده بهینه باشد، وجود ندارد و تنها با صرف زمان بسیار می‌توان جواب نسبتاً دقیقی به دست آورد؛ در حقیقت با توجه به زمان صرف شده، دقت جواب تغییر می‌کند. برای روش‌های ابتکاری نمی‌توان تعریفی جامع ارائه کرد. با وجود این، در اینجا کوشش می‌شود تعریفی تا حد امکان مناسب برای آن عنوان شود:روش جستجوی ابتکاری، روشی است که می‌تواند جوابی خوب (نزدیک به بهینه) در زمانی محدود برای یک مسأله ارائه کند. هیچ تضمینی برای بهینه بودن جواب وجود ندارد و متأسفانه نمی‌توان میزان نزدیکی جواب به دست آمده به جواب بهینه را تعیین کرد.



Global Optimization Approaches

فرایند بهینه سازی

فرموله کردن مسئله: در این مرحله، یک مسئله ی تصمیم گیری، همراه با یک ساختار کلی از آن تعریف می‌شود. این ساختار کلی ممکن است خیلی دقیق نباشد اما وضعیت کلی مسئله را، که شامل فاکتورهای ورودی و خروجی و اهداف مسئله است، بیان می کند. شفاف سازی و ساختاردهی به مسئله، ممکن است برای بسیاری از مسایل بهینه سازی، کاری پیچیده باشد.

مدل سازی مسئله:

در این مرحله یک مدل ریاضی کلی برای مسئله، ساخته می شود. مدل‌سازی ممکن است از مدل های مشابه در پیشینه ی موضوع کمک بگیرد. این گام موجب تجزیه مسئله به یک یا چند مدل بهینه‌سازی می گردد.

بهینه سازی مسئله:

پس از مدل سازی مسئله، روال حل، یک راه حل خوب برای مسئله تولید می کند. این راه‌حل ممکن است بهینه یا تقریباً بهینه باشد. نکته ای که باید به آن توجه داشت این است که راه حل به دست آمده، راه حلی برای مدل طراحی شده است، نه برای

مسئله ی واقعی. در هنگام فرموله کردن و مدلسازی ممکن است تغییراتی در مسئله واقعی به وجود آمده و مسئله ی جدید، نسبت به مسئله ی واقعی تفاوت زیادی داشته باشد.

استقرار مسئله:

راه حل به دست آمده توسط تصمیم گیرنده بررسی می شود و در صورتی که قابل قبول باشد، مورد استفاده قرار می گیرد و در صورتی که راه‌حل قابل قبول نباشد، مدل یا الگوریتم بهینه سازی باید توسعه داده شده و فرایند بهینه سازی تکرار گردد.

هدف الگوریتم های اکتشافی، ارائه راه حل در چارچوب یک زمان قابل قبول است که برای حل مسئله مناسب باشد، ممکن است الگوریتم اکتشافی، بهترین راه حل واقعی برای حل مسئله نبوده ولی می تواند راه حل نزدیک به بهترین باشد. الگوریتم های اکتشافی با الگوریتم های بهینه سازی برای اصلاح کارایی الگوریتم می‌توانند ترکیب شوند. الگوریتم فرا اکتشافی ترکیبی است از الگوریتم های اکتشافی که برای پیدا کردن، تولید یا انتخاب هر اکتشاف در هر مرحله طراحی می‌شود و راه حل خوبی برای مسائلی که مشکل بهینه‌سازی دارند ارائه می‌دهد. الگوریتم های فرا اکتشافی برخی از فرضیات مسائل بهینه سازی که باید حل شود را در نظر می گیرد

MIQP	Mixed Integer Quadratic Programming	برنامه‌ریزی اعداد صحیح مخلوط درجه دو
INLP	Integer Non-Linear Programming	برنامه‌ریزی اعداد صحیح غیرخطی
BNLP	Binary Non-Linear Programming	برنامه‌ریزی دودویی غیرخطی
MINLP	Mixed Integer Non-Linear Programming	برنامه‌ریزی اعداد صحیح مخلوط غیرخطی
SLP	Stochastic Linear Programming	برنامه‌ریزی خطی تصادفی
SQP	Stochastic Quadratic Programming	برنامه‌ریزی درجه دوم تصادفی
MOQP	Multi-Objective Quadratic Programming	برنامه‌ریزی درجه دوم چندهدفه
CP	Convex Programming	برنامه‌ریزی محدب
LSP	Least Square Programming	برنامه‌ریزی کمترین مربعات
LLSP	Linear Least Square Programming	برنامه‌ریزی کمترین مربعات خطی
NLLSP	Non-Linear Least Square Programming	برنامه‌ریزی کمترین مربعات غیرخطی

آزادسازی

آزادسازی (Relaxation)، یکی از روش‌های ابتکاری در بهینه‌سازی است که ریشه در روش‌های قطعی بهینه‌سازی دارد. در این روش، ابتدا مسأله به شکل یک مسأله برنامه‌ریزی خطی عدد صحیح LIP) = Linear Litegar Programming مختلط MIP) = Mixed Integer Programming) (و گاهی اوقات کمی غیر خطی)، فرموله می‌شود. سپس با برداشتن محدودیت‌های عدد صحیح بودن، یک مسأله آزاد شده به دست آمده و حل می‌شود. یک جواب خوب (و نه لزوماً بهینه) برای مسأله اصلی می‌تواند از روند کردن جواب مسأله آزاد شده (برای رسیدن به یک جواب موجه نزدیک به جواب مسأله آزاد شده)، به دست آید؛ اگر چه روند کردن جواب برای رسیدن به یک جواب لزوماً کار آسانی نیست، اما در مورد بسیاری از مدل‌های معمول، به آسانی قابل انجام است.

تجزیه

بسیاری اوقات آنچه که حل یک مسأله را از روش‌های قطعی بسیار مشکل می‌کند، این است که بیش از یک مورد تصمیم‌گیری وجود دارد، مانند موقعیت ماشین‌آلات و تخصیص کار، تخصیص بار به وسائل نقلیه و مسیریابی. هر یک از این موارد تصمیم‌گیری ممکن است به تنهایی پیچیده نباشند، اما در نظر گرفتن همه آنها در یک مدل به طور همزمان، چندان آسان نیست. روش ابتکاری تجزیه (Decomposition) می‌تواند در چنین مسائلی مفید واقع شود. در این روش، جواب به دو یا چند بخش (که فرض می‌شود

از هم مستقل هستند) تجزیه شده و هر یک جداگانه حل می‌شوند؛ سپس یک روش برای هماهنگ کردن و ترکیب این جواب‌های جزیی و به دست آوردن یک جواب خوب ابتکاری، به کار گرفته می‌شود.

تکرار

یکی از روش‌های تجزیه، **تکرار (Iteration)** است. در این روش، مسأله به زیرمسأله‌های جداگانه‌ای تبدیل می‌شود و در هر زمان یکی از زیرمسأله‌ها با ثابت در نظر گرفتن متغیرهای تصمیم موجود در سایر زیرمسأله‌ها در بهترین مقدار شناخته شده‌شان، بهینه می‌شود؛ سپس یکی دیگر از زیرمسأله‌ها در نظر گرفته می‌شود و این عمل به طور متناوب تا رسیدن به یک جواب رضایت‌بخش، ادامه می‌یابد.

روش تولید ستون (Column Generation)

این نیز یکی از روش‌های تجزیه است که عموماً برای مسائلی که دارای عناصر زیادی هستند (مانند مسأله کاهش ضایعات برش با تعداد الگوهای زیاد) کاربرد دارد. در این روش، حل مسأله به دو بخش تقسیم می‌شود:

- یافتن ستون‌ها (یا عناصر) جواب (مثلاً در مسأله کاهش ضایعات برش و یافتن الگوهای برش).
- یافتن ترکیب بهینه این عناصر، با توجه به محدودیت‌ها (در مسأله کاهش ضایعات برش و یافتن ترکیب مناسب الگوها).

یکی از روش‌های معمول برای یافتن ستون‌ها، مقدار متغیرهای دوگانه مسأله اصلی است، اما هر روش دیگری نیز می‌تواند مورد استفاده قرار گیرد

جستجوی سازنده (Constructive Search)

در این روش، با شروع از یک جواب تهی، تصمیم‌ها مرحله به مرحله گرفته می‌شود تا یک جواب کامل به دست آید. هر تصمیم، یک تصمیم آزمند است؛ یعنی قصد دارد با استفاده از اطلاعات به دست آمده از آنچه که تا کنون انجام شده است، بهترین تصمیم را بگیرد.

آنچه که یک الگوریتم سازنده و یک الگوریتم آزمند را از هم متمایز می‌کند، نحوه ساختن جواب‌ها می‌باشد. یک الگوریتم سازنده، جواب را به هر طریق ممکن تولید می‌کند، اما در یک الگوریتم آزمند، جواب مرحله به مرحله و با توجه به یافته‌ها، ساخته می‌شود (در هر مرحله، بخشی از جواب ساخته می‌شود). جستجوی سازنده در مسائلی مانند زمانبندی ماشین و بودجه‌بندی سرمایه کاربرد داشته است. در اینجا مثال مسیریابی کامیون مطرح می‌شود. در این مسأله کالا باید به نقاط مشخصی (هر یک با میزان مشخصی از تقاضا برای کالا) حمل شود؛ مسأله، سازماندهی این نقاط در مسیرهای مشخص با توجه به محدودیت ظرفیت کامیون است.

جستجوی بهبود یافته (Improving Search)

بر خلاف روش جستجوی سازنده، این روش با جواب‌های کامل کار می‌کند. جستجو با یک یا چند جواب (مجموعه‌ای از مقادیر متغیرهای تصمیم) شروع می‌شود و در هر مرحله، حرکت‌ها یا تغییرات مشخصی در مجموعه فعلی در نظر گرفته می‌شود و حرکت‌هایی که بیشترین بهبود را ایجاد می‌کنند، انجام می‌شود و عمل جستجو ادامه می‌یابد. یک مسأله در طراحی این روش، انتخاب جواب اولیه است. گاهی اوقات جواب اولیه یک جواب تصادفی است و گاهی نیز برای ساختن یک جواب اولیه، از روش‌هایی نظیر جستجوی سازنده استفاده می‌شود. مسأله دیگر، تعیین حرکت‌ها یا به عبارتی، تعریف همسایگی (مجموعه جواب‌هایی که با یک حرکت از جواب فعلی قابل دسترسی هستند) در مسأله است.

روش جستجوی همسایه (NS= Neighbourhood Search)

استفاده از الگوریتم مبتنی بر تکرار مستلزم وجود یک سازوکار تولید جواب است. سازوکار تولید جواب، برای هر جواب i یک همسایه به وجود می‌آورد که می‌توان از i به آن منتقل شد. الگوریتم‌های تکراری به عنوان جستجوی همسایه (NS) یا جستجوی محلی نیز شناخته می‌شوند. الگوریتم بدین صورت بیان می‌شود که از یک نقطه (جواب) شروع می‌شود و در هر تکرار، از نقطه جاری به یک نقطه همسایه جابه‌جایی صورت می‌گیرد. اگر جواب همسایه مقدار کمتری داشته باشد، جایگزین جواب

جاری می‌شود (در مسأله حداقل‌سازی) و در غیر این صورت، نقطه همسایه دیگری انتخاب می‌شود. هنگامی که مقدار جواب از جواب تمام نقاط همسایه آن کمتر باشد، الگوریتم پایان می‌یابد. مفهوم روش جستجوی همسایه از حدود چهل سال پیش مطرح شده است. از جمله اولین موارد آن، کارهای کرز می‌باشد که برای حل مسأله فروشنده دوره‌گرد از مفهوم جستجوی همسایه استفاده کرده است. در کارهای اخیر ریوز نیز جنبه‌هایی از این شیوه یافت می‌شود.

اشکالات الگوریتم فوق بدین شرح است:

- ممکن است الگوریتم در یک بهینه محلی متوقف شود، اما مشخص نباشد که آیا جواب به دست آمده یک بهینه محلی است یا یک بهینه سراسری.
- بهینه محلی به دست آمده به جواب اولیه وابسته است و در مورد چگونگی انتخاب جواب اولیه هیچ راه حلی در دسترسی نیست.
- به طور معمول نمی‌توان یک حد بالا برای زمان اجرا تعیین کرد.

البته الگوریتم‌های مبتنی بر تکرار مزایایی نیز دارند؛ از جمله اینکه یافتن جواب اولیه، تعیین مقدار تابع و سازوکار تولید جواب همسایه به طور معمول ساده است. با وجود آنکه تعیین حد بالای زمان اجرا امکان‌پذیر نیست، ولی با اطمینان می‌توان گفت که یک تکرار از الگوریتم در زمان مشخص قابل اجراست.

روش‌های فرا ابتکاری (Metaheuristic) برگرفته از طبیعت

برگرفته از طبیعت روش‌های فرا ابتکاری

الگوریتم های فراابتکاری الگوریتم هایی هستند که با الهام از طبیعت، فیزیک و انسان طراحی شده اند و در حل بسیاری از مسایل بهینه سازی استفاده می شوند. معمولاً از الگوریتم های فراابتکاری در ترکیب با سایر الگوریتم ها، جهت رسیدن به جواب بهینه یا خروج از وضعیت جواب بهینه محلی استفاده می‌گردد. در سال‌های اخیر یکی از مهمترین و امیدبخش‌ترین تحقیقات، «روش‌های ابتکاری برگرفته از طبیعت» بوده است؛ این روش‌ها شباهت‌هایی با سیستم‌های اجتماعی و یا طبیعی دارند. کاربرد آنها برگرفته از روش‌های ابتکاری پیوسته می‌باشد که در حل مسائل مشکل ترکیبی (NP-Hard) نتایج بسیار خوبی داشته است. در ابتدا با تعریفی از طبیعت و طبیعی بودن روش‌ها شروع می‌کنیم؛ روش‌ها برگرفته از فیزیک، زیست‌شناسی و جامعه‌شناسی هستند و به صورت زیر تشکیل شده‌اند:

- استفاده از تعداد مشخصی از سعی‌ها و کوشش‌های تکراری
- استفاده از یک یا چند عامل (نرون، خرده‌ریز، کروموزوم، مورچه و غیره)
- عملیات (در حالت چند عاملی) با یک سازوکار همکاری ـ رقابت
- ایجاد روش‌های خود تغییری و خود تبدیلی

طبیعت دارای دو تدبیر بزرگ می‌باشد:

۱- انتخاب پاداش برای خصوصیات فردی قوی و جزا برای فرد ضعیف‌تر؛

۲- جهش که معرفی اعضای تصادفی و امکان تولد فرد جدید را میسر می‌سازد.

به طور کلی دو وضعیت وجود دارد که در روش‌های ابتکاری برگرفته از طبیعت دیده می‌شود، یکی انتخاب و دیگری جهش. انتخاب ایده‌ای مبنا برای بهینه‌سازی و جهش ایده‌ای مبنا برای جستجوی پیوسته می‌باشد.

از خصوصیات روش‌های ابتکاری برگرفته از طبیعت، می‌توان به موارد زیر اشاره کرد:

۱- پدیده‌ای حقیقی در طبیعت را مدل‌سازی می‌کنند.

۲- بدون قطع می‌باشند.

۳- اغلب بدون شرط ترکیبی همانند (عامل‌های متعدد) را معرفی می‌نمایند.

۴- تطبیق‌پذیر هستند.

خصوصیات بالا باعث رفتاری معقول در جهت تأمین هوشمندی می‌شود. تعریف هوشمندی نیز عبارت است از قدرت حل مسائل مشکل؛ بنابراین هوشمندی به حل مناسب مسائل بهینه‌سازی ترکیبی منجر می‌شود.

در سال‌های اخیر یکی از مهمترین و امیدبخش‌ترین تحقیقات، «روش‌های ابتکاری برگرفته از طبیعت» بوده است؛ این روش‌ها شباهت‌هایی با سیستم‌های اجتماعی و یا طبیعی دارند. کاربرد آنها برگرفته از روش‌های ابتکاری پیوسته می‌باشد که در حل مسائل مشکل ترکیبی (NP-Hard) نتایج بسیار خوبی داشته است.

در ابتدا با تعریفی از طبیعت و طبیعی بودن روش‌ها شروع می‌کنیم؛ روش‌ها برگرفته از فیزیک، زیست‌شناسی و جامعه‌شناسی هستند و به شکل زیر تشکیل شده‌اند:

- استفاده از تعداد مشخصی از سعی‌ها و کوشش‌های تکراری
- استفاده از یک یا چند عامل (نرون، خرده‌ریز، کروموزوم، مورچه و غیره)
- عملیات (در حالت چند عاملی) با یک سازوکار همکاری ـ رقابت
- ایجاد روش‌های خود تغییری و خود تبدیلی

طبیعت دارای دو تدبیر بزرگ می‌باشد:

- انتخاب پاداش برای خصوصیات فردی قوی و جزا برای فرد ضعیف‌تر؛
- جهش که معرفی اعضای تصادفی و امکان تولد فرد جدید را میسر می‌سازد.

به طور کلی دو وضعیت وجود دارد که در روش‌های ابتکاری برگرفته از طبیعت دیده می‌شود، یکی انتخاب و دیگری جهش. انتخاب ایده‌ای مبنا برای بهینه‌سازی و جهش ایده‌ای مبنا برای جستجوی پیوسته می‌باشد.

از خصوصیات روش‌های ابتکاری برگرفته از طبیعت، می‌توان به موارد زیر اشاره کرد:

- پدیده‌ای حقیقی در طبیعت را مدل‌سازی می‌کنند.
- بدون قطع می‌باشند.
- اغلب بدون شرط ترکیبی همانند (عامل‌های متعدد) را معرفی می‌نمایند.
- تطبیق‌پذیر هستند.

خصوصیات بالا باعث رفتاری معقول در جهت تأمین هوشمندی می‌شود. تعریف هوشمندی نیز عبارت است از قدرت حل مسائل مشکل؛ بنابراین هوشمندی به حل مناسب مسائل بهینه‌سازی ترکیبی منجر می‌شود.

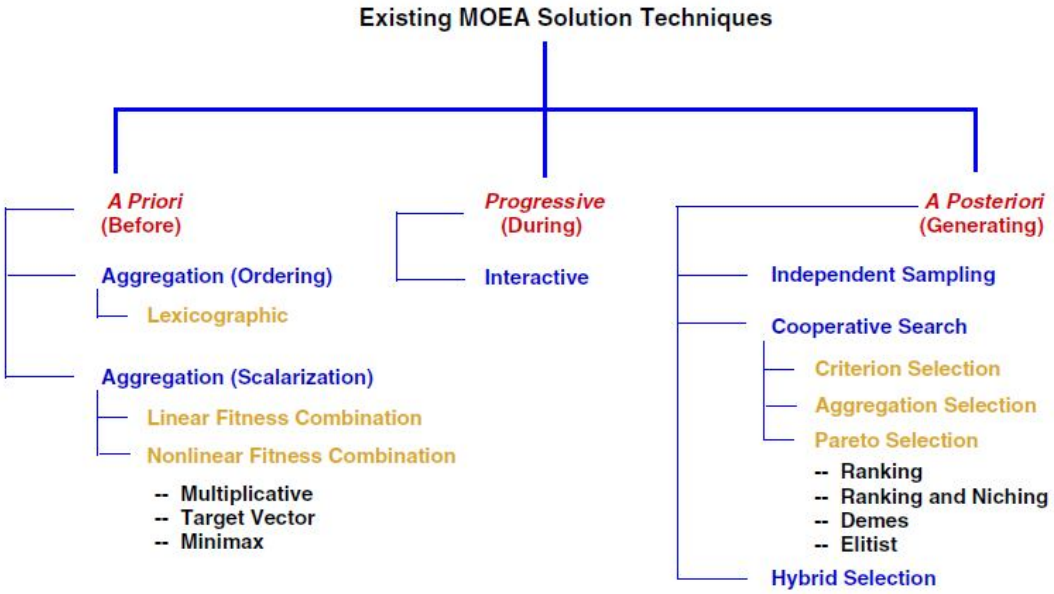
۱-۳- مسأله فروشنده دوره‌گرد (Travelling Salesman Problem = TSP)

در ابتدا به مسأله فروشنده دوره‌گرد (TSP) توجه می‌کنیم؛ در این مسأله مأموری به صورت تصادفی در فضای جستجو (گراف n بعدی) حرکت می‌کند. تنها موقعیت اجباری این است که مأمور باید فهرست شهرهایی را که رفته به جهت اجتناب از تکرار به خاطر بسپارد. در این روش بعد از n حرکت، مأمور به شهر شروع باز می‌گردد و راه‌حلی به دست می‌آید. روش جستجوی تصادفی ممکن است تکراری بوده و یا از شهرهای مختلف شروع شود که شامل الگوریتم چند شروعه (Multistart) می‌شود.

روش‌های فرا ابتکاری می‌توانند مطابق موارد زیر به دست آیند:

- استفاده از شیوه‌ای مبتنی بر علاقه‌مندی برای انتخاب هر حرکت مأمور؛
- استفاده از روش جستجوی محلی (معاوضه موقعیت گره‌ها) برای بهبودی راه‌حل؛
- استفاده از روش جستجوی محلی تصادفی و تنها پذیرش تغییرات بهبود یافته؛
- استفاده از m مأمور که از شهرهای مختلف شروع می‌کنند.
- استفاده از تعدادی مأمور با استخدام غیر قطعی؛
- استفاده از روش‌های گروهی برای قسمت‌بندی فضا و یا مأموران؛
- استفاده از قانون پذیرش بدون قطع برای تغییرات اصلاح نشده؛

- استفاده از اطلاعات آخرین حرکات برای اجرای یک سیستم حافظه‌ای.



برخی الگوریتم های بهینه سازی مهم عبارت انداز:

انواع روش‌های فرا ابتکاری برگرفته از طبیعت

1- الگوریتم ژنتیک

الگوریتم‌های ژنتیک، تکنیک جستجویی در علم رایانه برای یافتن راه‌حل تقریبی برای بهینه‌سازی و مسائل جستجو است. الگوریتم ژنتیک نوع خاصی از الگوریتم‌های تکامل است که از تکنیک‌های زیست‌شناسی فرگشتی مانند وراثت و جهش استفاده می‌کند. این الگوریتم برای اولین بار توسط جان هنری هالند معرفی شد. در واقع الگوریتم‌های ژنتیک از اصول انتخاب طبیعی داروین برای یافتن فرمول بهینه جهت پیش‌بینی یا تطبیق الگو استفاده می‌کنند. الگوریتم‌های ژنتیک اغلب گزینه خوبی برای تکنیک‌های پیش‌بینی بر مبنای رگرسیون هستند. در هوش مصنوعی الگوریتم ژنتیک (یا GA) یک تکنیک برنامه‌نویسی است که از تکامل ژنتیکی به عنوان یک الگوی حل مسئله استفاده می‌کند. مسئله‌ای که باید حل شود دارای ورودی‌هایی می‌باشد که طی یک فرایند الگوبرداری شده از تکامل ژنتیکی به راه‌حلها تبدیل می‌شود سپس راه حلها بعنوان کاندیداها توسط تابع ارزیاب (Fitness Function) مورد ارزیابی قرار می‌گیرند و چنانچه شرط خروج مسئله فراهم شده باشد الگوریتم خاتمه می‌یابد.
الگوریتم ژنتیک بطور کلی یک الگوریتم مبتنی بر تکرار است که اغلب بخش‌های آن به صورت فرایندهای تصادفی انتخاب می‌شوند الگوریتم ژنتیک (Genetic Algorithm) روشی عمومی از روش‌های فرا ابتکاری برای بهینه‌سازی گسسته می‌باشد که مسائل جدول زمانبندی را حل می‌نماید. روش شبیه‌سازی که در ادامه مورد بحث قرار می‌گیرد، راهبرد تکاملی نام دارد. این روش در سال ۱۹۷۵ به وسیله هولند (Holland) و در سال ۱۹۸۹ توسط گولدبرگ (Goldberg) ابداع شده است. ین روش نوعی روش جستجوی همسایه است که عملکردی مشابه ژن دارد. در طبیعت، فرایند تکامل هنگامی ایجاد می‌شود که چهار شرط زیر برقرار باشد:

الف) یک موجود توانایی تکثیر داشته باشد (قابلیت تولید مثل).
ب) جمعیتی از این موجودات قابل تکثیر وجود داشته باشد.
پ) چنین وضعیتی دارای تنوع باشد.
ت) این موجودات به وسیله قابلیت‌هایی در زندگی از هم جدا شوند.

در طبیعت، گونه‌های متفاوتی از یک موجود وجود دارند که این تفاوت‌ها در کروموزوم‌های این موجودات ظاهر می‌شود و باعث تنوع در ساختار و رفتار این موجودات می‌شود. این تنوع ساختار و رفتار به نوبه خود بر زاد و ولد تأثیر می‌گذارد. موجوداتی که قابلیت‌ها و توانایی بیشتری برای انجام فعالیت‌ها در محیط دارند (موجودات متکامل‌تر)، دارای نرخ زاد و ولد بالاتری خواهند بود و طبعاً موجوداتی که سازگاری کمتری با محیط دارند، از نرخ زاد و ولد پایین‌تری برخوردار خواهند بود. بعد از چند دوره زمانی و گذشت چند نسل، جمعیت تمایل دارد که موجوداتی را بیشتر در خود داشته باشد که کروموزوم‌هایشان با محیط اطراف سازگاری بیشتری دارد.

در طی زمان، ساختار افراد جامعه به علت انتخاب طبیعی تغییر می‌کند و این نشانه تکامل جمعیت است.

۲- آنیلینگ شبیه‌سازی شده

این روش توسط متروپولیس (Metropolis) و همکاران در سال ۱۹۵۳ پیشنهاد شده و جهت بهینه‌سازی به وسیله وچی (Vecchi)، گلات (Gelatt) و کرک‌پاتریک (kirkpatrick) در سال ۱۹۸۳ مورد بازبینی قرار گرفته است. این روش در مسائل تاکسی تلفنی کاربرد دارد. الگوریتم آنیلینگ شبیه‌سازی شده (Simulated Annealing) در شکل عمومی، بر اساس شباهت میان سرد شدن جامدات مذاب و حل مسائل بهینه‌سازی ترکیبی به وجود آمده است. در فیزیک مواد فشرده، گرم و سرد کردن فرایندی است فیزیکی که طی آن یک ماده جامد در ظرفی حرارت داده می‌شود تا مایع شود؛ سپس حرارت آن بتدریج کاهش می‌یابد. بدین ترتیب تمام ذرات فرصت می‌یابند تا خود را در پایین‌ترین سطح انرژی منظم کنند. چنین وضعی در شرایطی ایجاد می‌شود که گرمادهی کافی بوده و سرد کردن نیز به آهستگی صورت گیرد. جواب حاصل از الگوریتم گرم و سرد کردن شبیه‌سازی شده، به جواب اولیه وابسته نیست و می‌توان توسط آن جوابی نزدیک به جواب بهینه به دست آورد. حد بالایی زمان اجرای الگوریتم نیز قابل تعیین است. بنابراین الگوریتم گرم و سرد کردن شبیه‌سازی شده، الگوریتمی است تکراری که اشکالات روش‌های عمومی مبتنی بر تکرار را ندارد. در روش آنیلینگ شبیه‌سازی شده، به صورت پی در پی از جواب جاری به یکی از همسایه‌های آن انتقال صورت می‌گیرد. این سازوکار توسط زنجیره مارکوف به صورت ریاضی قابل توصیف است. در این روش، یک مجموعه آزمون انجام می‌گیرد؛ این آزمون‌ها به نحوی است که نتیجه هر یک به نتیجه آزمون قبل وابسته است. در روش آنیلینگ شبیه‌سازی شده، منظور از یک آزمون، انتقال به نقطه جدید است و روشن است که نتیجه انتقال به نقطه جدید تنها وابسته به مشخصات جواب جاری است. روش جستجوی همسایه و روش آنیلینگ شبیه‌سازی شده، هر دو روش‌های تکراری هستند. در الگوریتم آنیلینگ شبیه‌سازی شده، هر بار که شاخص کنترل‌کننده به مقدار نهایی خود می‌رسد، در حقیقت یک عملیات تکراری انجام شده است. در الگوریتم جستجوی همسایه، هنگامی که تعداد تکرارها به سمت بی‌نهایت میل می‌کند، روش به جواب بهینه نزدیک می‌شود. اما عملکرد الگوریتم آنیلینگ شبیه‌سازی شده سریع‌تر است.

۳- شبکه‌های عصبی

شبکه‌های عصبی (Neural Networks) مصنوعی سیستم‌های هوشمندی هستند که از شبکه‌های عصبی طبیعی الهام گرفته شده‌اند. شبکه‌های عصبی مصنوعی در واقع تلاشی برای حرکت از سمت مدل محاسباتی فون نیومن به سمت مدلی است که با توجه به عملکرد و ویژگی‌های مغز انسان طراحی شده است. مدل فون نیومن گرچه هم اکنون بسیار استفاده می‌شود، اما از کمبودهایی رنج می‌برد که تلاش شده است این کمبودها در شبکه‌های عصبی مصنوعی برطرف شود.

در سال ۱۹۴۳ مدلی راجع به عملکرد نورون‌ها (Neuron) نوشته شد که با اندکی تغییر، امروزه بلوک اصلی سازنده اکثر شبکه‌های عصبی مصنوعی می‌باشد. عملکرد اساسی این مدل مبتنی بر جمع کردن ورودی‌ها و به دنبال آن به وجود آمدن یک خروجی است. ورودی‌های نورون‌ها از طریق دندریت‌ها که به خروجی نورون‌های دیگر از طریق سیناپس متصل است، وارد می‌شوند. بدنه سلولی کلیه این ورودی‌ها را دریافت می‌کند و چنانچه جمع این مقادیر از مقداری که به آن آستانه گفته می‌شود بیشتر باشد، در اصطلاح برانگیخته شده یا آتش می‌کند و در این صورت، خروجی نورون روشن یا خاموش خواهد شد.

۴-جستجوی ممنوع

روشی عمومی است که به وسیله گلوور (Glover) در سال ۱۹۸۹ پیشنهاد شده و در حل مسائل برنامه‌ریزی کاری ـ خرید کاربرد دارد. روش جستجوی ممنوع (Tabu Search)، همانند روش آنیلینگ شبیه‌سازی شده بر اساس جستجوی همسایه بنا شده است. در این روش عملکرد حافظه انسان شبیه‌سازی شده است. حافظه انسان با به کارگیری ساختمانی مؤثر و در عین حال ساده از اطلاعات، آنچه را در قبل رؤیت شده، ذخیره می‌کند. این مرکز همچنین فهرستی از حرکات منع شده را تنظیم می‌کند و این فهرست همواره بر اساس آخرین جستجوها منظم می‌شود. این روش از انجام هر گونه عملیات مجدد و تکراری جلوگیری می‌کند. شکل نوین جستجوی ممنوع توسط گلوور مطرح شده است. روش جستجوی مبتنی بر منع، با ایجاد تغییری کوچک در روش جستجوی همسایه به وجود می‌آید. هدف این روش آن است که بخش‌هایی از مجموعه جواب که پیش از این بررسی نشده است، مد نظر قرار گیرد. بدین منظور حرکت به جواب‌هایی که اخیراً جستجو شده، ممنوع خواهد بود. ساختار کلی روش جستجوی ممنوع بدین صورت است که ابتدا یک جواب اولیه امکان‌پذیر انتخاب می‌شود؛ سپس برای جواب مربوط، بر اساس یک معیار خاص مجموعه‌ای از جواب‌های همسایه امکان‌پذیر در نظر گرفته می‌شود در گام بعد، پس از ارزیابی جواب‌های همسایه تعیین شده، بهترین آنها انتخاب می‌شود و جابه‌جایی از جواب جاری به جواب همسایه انتخابی صورت می‌گیرد. این فرایند به همین ترتیب تکرار می‌شود تا زمانی که شرط خاتمه تحقق یابد. در روش جستجوی ممنوع، فهرستی وجود دارد که جابه‌جایی‌های منع شده را نگهداری می‌کند و به فهرست تابو معروف است و کاربرد اصلی آن، پرهیز از همگرا شدن به جواب‌های بهینه محلی است. به عبارت دیگر، به کمک فهرست تابو جابه‌جایی به جواب‌هایی که اخیراً جستجو شده‌اند، ممنوع خواهد شد. فقط بخش‌هایی از مجموعه جواب که پیش از این مورد بررسی قرار نگرفته، مد نظر خواهند بود. در واقع جابه‌جایی از جواب جاری به جواب همسایه امکان‌پذیر زمانی انجام می‌شود که در فهرست تابو قرار نداشته باشد. در غیر این صورت، جواب همسایه دیگری که در ارزیابی جواب‌های همسایه در رده بعدی قرار گرفته است، انتخاب شده و جابه‌جایی به آن صورت می‌گیرد. در روش جستجوی ممنوع بعد از هر جابه‌جایی، فهرست تابو بهنگام می‌شود، به نحوی که جابه‌جایی جدید به آن فهرست اضافه شده و جابه‌جایی که تا n تکرار مشخص در فهرست بوده است، از آن حذف می‌شود. نحوه انتخاب می‌تواند با توجه به شرایط و نوع مسأله متفاوت باشد.

5- سیستم مورچه (Ant System)

این روش در سال ۱۹۹۱ توسط مانیه‌زو (Maniezzo) دوریگو (Dorigo) و کولورنی (Colorni) پیشنهاد شده است که در حل مسأله فروشنده دوره‌گرد و مسائل تخصیص چندوجهی کاربرد دارد. الگوریتم بهینه‌سازی کلونی مورچه‌ها از عامل‌های ساده‌ای که مورچه نامیده می‌شوند، استفاده می‌کند تا به صورت تکراری جواب‌هایی تولید کند. مورچه‌ها می‌توانند کوتاه‌ترین مسیر از یک منبع غذایی به لانه را با بهره‌گیری از اطلاعات فرمونی پیدا کنند. مورچه‌ها در هنگام راه رفتن، روی زمین فرمون می‌ریزند و با بو کشیدن فرمون ریخته شده بر روی زمین راه را دنبال می‌کنند؛ چنانچه در طی مسیر به سوی لانه به یک دوراهی برسند، از آن جایی که هیچ اطلاعی درباره راه بهتر ندارند، راه را به تصادف برمی‌گزینند. انتظار می‌رود به طور متوسط نیمی از مورچه‌ها مسیر اول و نیمی دیگر مسیر دوم را انتخاب کنند. فرض می‌شود که تمام مورچه‌ها با سرعت یکسان مسیر را طی کنند. از آنجا که یک مسیر کوتاه‌تر از مسیر دیگر است، مورچه‌های بیشتری از آن می‌گذرند و فرمون بیشتری بر روی آن انباشته می‌شود. بعد از مدت کوتاهی مقدار فرمون روی دو مسیر به اندازه‌ای می رسد که روی تصمیم مورچه‌های جدید برای انتخاب مسیر بهتر تأثیر می‌گذارد. از این به بعد، مورچه‌های جدید با احتمال بیشتری ترجیح می‌دهند از مسیر کوتاه‌تر استفاده کنند، زیرا در نقطه تصمیم‌گیری مقدار فرمون بیشتری در مسیر کوتاه‌تر مشاهده می‌کنند. بعد از مدت کوتاهی تمام مورچه‌ها این مسیر را انتخاب خواهند کرد.منظور از بهینه سازی یک سامانه کمینه یا بیشینه کردن تابعی است که این تابع معیاری از عملکرد سامانه می باشد. این عمل در نهایت به بهبود کارایی سامانه می انجامد. در طراحی آیرودینامیکی این تابع می تواند ضریب برآ، ضریب پسا، نسبت ضریب برآ به ضریب پسا و یا تابع دیگری باشد. در گذشته طراحان برای طراحی آیرودینامیکی بهینه نیاز به ساخت مدل های بسیاری برای تست در تونل باد داشتند تا بدین وسیله بتوانند عملکرد طراحی نهایی را تایید کنند. توسعه دینامیک سیالات محاسباتی در چند دهه گذشته این امکان را فراهم کرد که طراحان از طریق شبیه سازی عددی، طراحی ها را با سرعت بیشتری

انجام دهند. با این وجود، این روش نیز شامل یک فرآیند سعی و خطا است و در بسیاری از موارد به سامانه بهینه منجر نمی‌شود.به طور کلی می توان از سه مرحله مهم برای بهینه سازی یک سامانه نام برد:

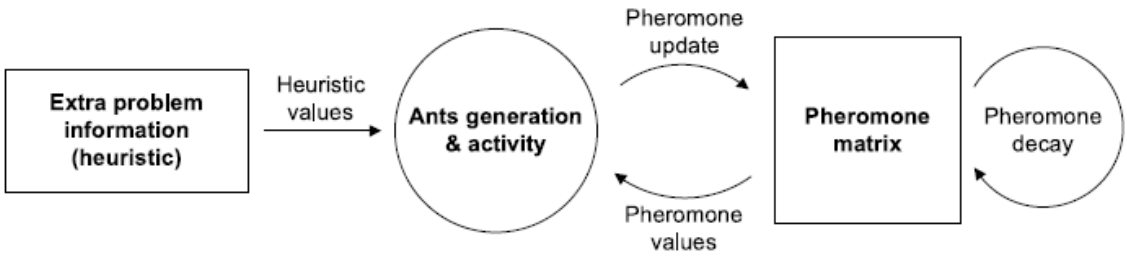
- مرحله اول درک سامانه و متغیرهای مختلفی است که بر روی آن تاثیر می گذارند.
- مرحله دوم انتخاب تابعی به عنوان معیار عملکرد سامانه است. این معیار به متغیرهای سامانه وابسته است و تاثیر زیادی روی کارآیی سامانه دارد.
- مرحله سوم انتخاب مقدار متغیرهای سامانه است و این انتخاب به گونه ای است که سامانه بهینه می شود.

در مرحله اول طراح باید درک صحیحی از سامانه، نحوه عملکرد آن، متغیرهای مختلف تاثیر گذار بر سامانه و تاثیر متقابل متغیرها بر روی یکدیگر داشته باشد. در مرحله دوم باید معیار عملکرد سامانه تعریف شود. در طراحی هواپیما و بالگرد انتخاب های گوناگونی در این زمینه وجود دارد. برای مثال نسبت ضریب برآ به ضریب پسا می تواند معیار عملکرد یک هواپیما باشد. در حالت کلی انتخاب معیار عملکرد به ماموریتی که بر عهده هواپیما می باشد بستگی دارد. در مرحله سوم، طراح با استفاده از یک روش بهینه سازی مناسب به جستجوی مقادیر بهینه برای متغیرهای طراحی می پردازد. انتخاب روش بهینه سازی به عواملی نظیر خطی بودن مساله، تعداد متغیرهای طراحی، تعداد توابع هزینه و مقید یا غیر مقید بودن مساله بستگی دارد. منظور از تابع هزینه تابعی است که معیاری برای کارایی سامانه می باشد. در طراحی مهندسی گاهی لازم است قیدهایی را به مساله تحمیل کنیم. مثلا ممکن است بخواهیم بال یا پره را به گونه ای طراحی کنیم که ضخامت آن از مقدار معینی بیشتر یا کمتر شود. به این نوع طراحی، طراحی مقید گفته می شود. اما در طراحی غیر مقید، قیدی به مساله اعمال نمی‌شود.

روش های بهینه سازی را می توان به دو گروه کلی دسته بندی کرد:

- روش های غیر مبتنی بر محاسبه گرادیان ها
- روش هایی که بر مبنای محاسبه گرادیان ها می باشند.

در روش های نوع اول هیچ اطلاعاتی از گرادیان های تابع هزینه نسبت به متغیرهای طراحی در خلال فرآیند بهینه سازی لازم نیست. و جستجو برای رسیدن به نقطه بهینه با مقایسه مقادیر تابع هزینه در نقاط طراحی مختلف انجام می شود. روش های جستجوی تصادفی مانند الگوریتم ژنتیک و روش های بهینه سازی آنیلینگ در این دسته قرار می گیرند. در روش های نوع دوم گرادیان های تابع هزینه نسبت به متغیرهای طراحی نقشی اساسی را در فرآیند بهینه سازی ایفاء می کنند. روش های تفاضل محدود و بسط سری تیلور مختلط از این نوع هستند. در این روش ها بعد از محاسبه مشتقات تابع هزینه نسبت به متغیرهای طراحی، با استفاده از یک الگوریتم مرتبه اول یا دوم جستجو برای یافتن مقادیر بهینه آغاز می شود. در الگوریتم های مرتبه اول تنها مشتق اول تابع هزینه نسبت به متغیرهای طراحی لازم است. به عنوان مثال می توان به الگوریتم سریع ترین شیب اشاره کرد. در این الگوریتم جستجو در جهت منفی بردار گرادیان انجام می شود. در الگوریتم های مرتبه دوم علاوه بر مشتق اول، مقادیر مشتق دوم تابع هزینه نسبت به متغیرهای طراحی نیز مورد نیاز می باشد. الگوریتم های شبه نیوتنی از این نوع هستند.



Process organisation of the Ant Colony Optimisation Metaheuristic Framework.

الگوریتم کلونی مورچگان

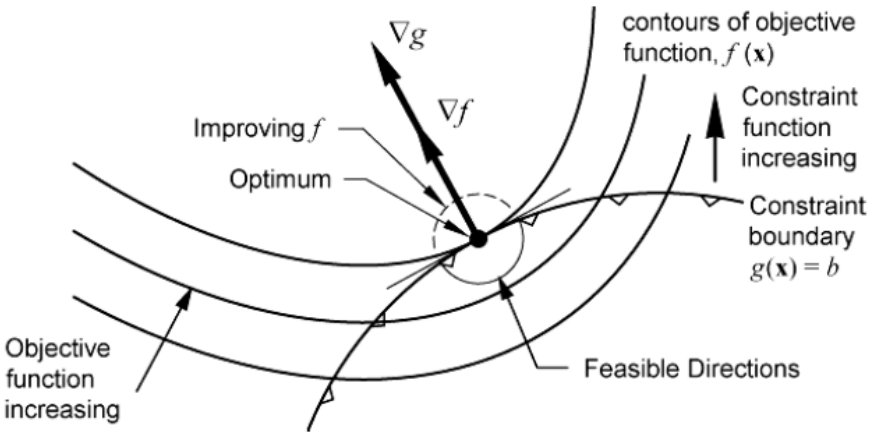
بهینه‌سازی گروه مورچه‌ها یا ACO همانطور که می دانیم مسئله یافتن کوتاهترین مسیر، یک مسئله بهینه سازیست که گاه حل آن بسیار دشوار است و گاه نیز بسیار زمانبر. برای مثال مسئله فروشنده دوره گرد را نیز می‌توان مطرح کرد. در این روش(ACo)، مورچه‌های مصنوعی به‌وسیلهٔ حرکت بر روی نمودار مسئله و با باقی گذاشتن نشانه‌هایی بر روی نمودار، همچون مورچه‌های واقعی که در مسیر حرکت خود نشانه‌های باقی می‌گذارند، باعث می‌شوند که مورچه‌های مصنوعی بعدی بتوانند راه‌حل‌های بهتری را برای مسئله فراهم نمایند. همچنین در این روش می‌توان توسط مسائل محاسباتی-عددی بر مبنای علم احتمالات بهترین مسیر را در یک نمودار یافت.

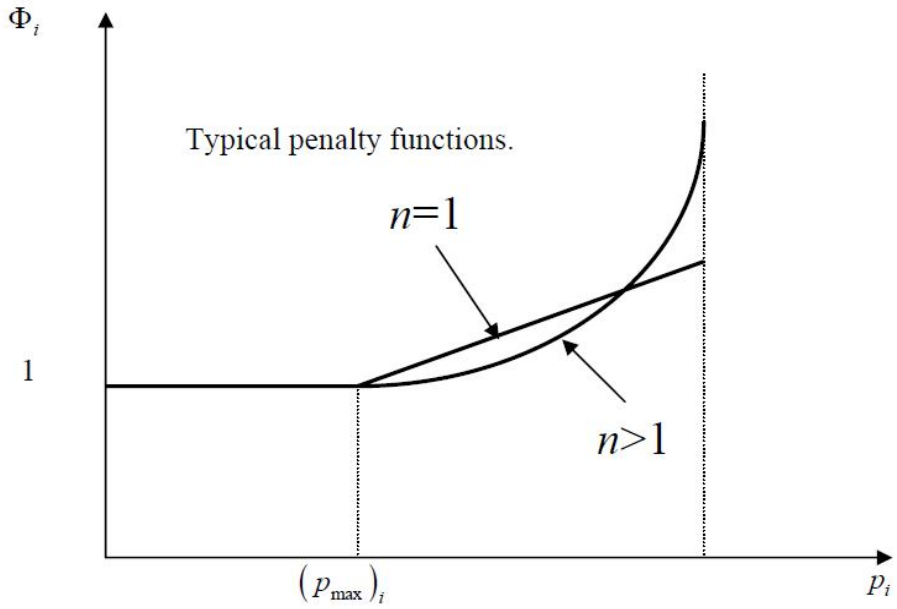
۶- الگوریتم بهینه سازی ازدحام ذرات PSO

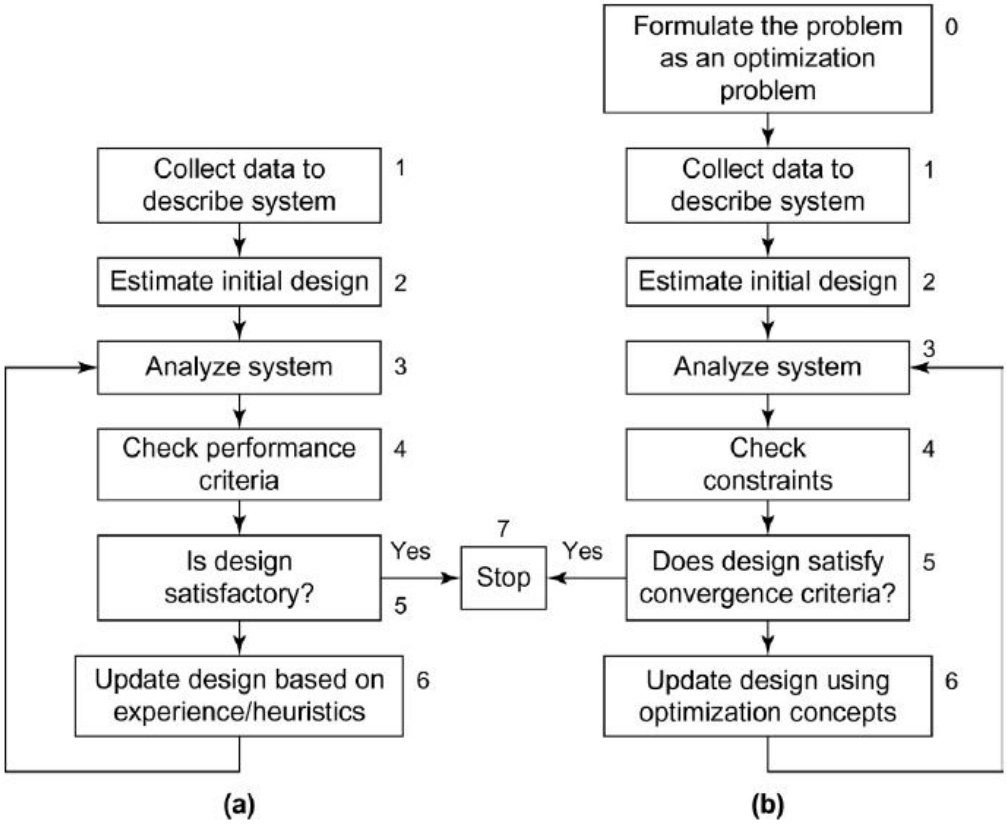
در سال ۱۹۹۵ ابرهارت و کنیدی برای اولین بارPSO به عنوان یک روش جستجوی غیر قطعی برای بهینه سازی تابعی مطرح گشت این الگوریتم از حرکت دسته جمعی پرندگانی که به دنبال غذا می‌باشند، الهام گرفته شده است. گروهی از پرندگان در فضایی به صورت تصادفی دنبال غذا می‌گردند. تنها یک تکه غذا در فضای مورد بحث وجود دارد. هیچ یک از پرندگان محل غذا را نمی‌دانند. یکی از بهترین استراتژی‌ها می‌تواند دنبال کردن پرنده‌ای باشد که کمترین فاصله را تا غذا داشته باشد. این استراتژی در واقع جانمایه الگوریتم است. هر راه‌حل که به آن یک ذره گفته می‌شود، PSO در الگوریتم معادل یک پرنده در الگوریتم حرکت جمعی پرندگان می‌باشد. هر ذره یک مقدار شایستگی دارد که توسط یک تابع شایستگی محاسبه می‌شود. هر چه ذره در فضای جستجو به هدف - غذا در مدل حرکت پرندگان - نزدیکتر باشد، شایستگی بیشتری دارد. همچنین هر ذره دارای یک سرعت است که هدایت حرکت ذره را بر عهده دارد. هرذره با دنبال کردن ذرات بهینه در حالت فعلی، به حرکت خود در فضای مساله ادامه می‌دهد.

۷- الگوریتم کرم شب تاب

الگوریتم کرم شب تاب به عنوان الگوریتم ذهنی مبتنی بر ازدحام، برای وظایف بهینه‌سازی محدود، ارائه شده است. در این الگوریتم از رفتار تابشی کرم‌های شب تاب الهام گرفته شده است. کرم‌های شب تاب در طبیعت به طور دسته جمعی زندگی می‌کنند و همواره کرم کم نورتر به سمت کرم پرنور تر حرکت می‌کند. این الگوریتم یک رویه تکراری مبتنی بر جمعیت را با عوامل بی‌شمار (تحت عنوان کرم‌های شب‌تاب) به کار می‌گیرد. به این عوامل امکان داده می‌شود تا فضای تابع هزینه را به صورت موثرتری نسبت به جستجوی تصادفی توزیع شده، بررسی کنند. تکنیک بهینه‌سازی هوشمند، مبتنی بر این فرضیه است که راه‌حل یک مشکل بهینه‌سازی را، می‌توان به عنوان عاملی (کرم شب‌تاب) در نظر گرفت که به صورت متناسب با کیفیت آن در یک محیط تابیده می‌شود. متعاقباً هر کرم شب‌تاب، همتایان خود را (صرف نظر از جنسیتشان) جذب می‌کند که فضای جستجو را به صورت موثرتری بررسی می‌کند. الگوریتم‌های کرم شب‌تاب نورهای ریتمیک و کوتاه تولید می‌کنند. الگوی نوری هر کدام از کرم‌های شب‌تاب با یکدیگر متفاوت می‌باشند. الگوریتم‌های کرم شب‌تاب از این نورها به دو منظور استفاده می‌کنند. ۱- پروسه جذب جفت‌ها ۲- جذب شکار. به علاوه این نورها می‌توانند به عنوان یک مکانیزم محافظتی برای کرم‌های شب‌تاب باشند.







Comparison of: (a) conventional design method; and (b) optimum design method.

روند کلی بهینه سازی گرادیانی

۱. محاسبه گرادیان

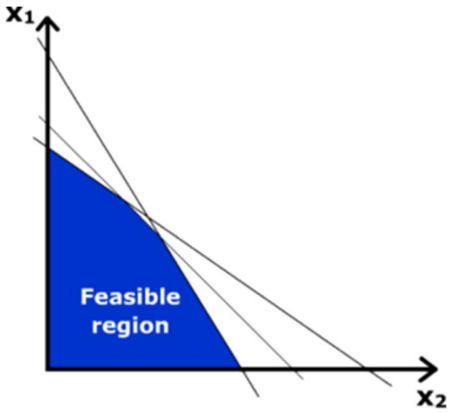
۲. جستجوی خطی در راستای تندترین شیب نزولی و یا یک راستای بهبود یافته بر مبنای تخمین هسیان

این متد تا وقتی که تابع هزینه را بتوان به سادگی محاسبه کرد، به خوبی کار می کند اما در مورد طراحی با استفاده از معادلات ناویر استوکس و یا اویلر و در حالت تعداد زیاد متغیرهای طراحی صدق نمی کند. روش الحاقی یک استراتژی به منظور کاستن از هزینه محاسبه گرادیان ها است و به منظور احتراز از جستجوی خطی از یک فرآیند نزولی پیوسته به همراه هموارسازی گرادیان بهره می گیرد.روش های جستجوی خطی مستلزم آن هستند که راستایی توسط الگوریتم انتخاب و در طول آن جستجو آغاز شود و این کار با انجام تکرار تا رسیدن به مقدار جدیدی برای تابع هدف صورت می گیرد. با انتخاب شدن راستای جستجو، یک طول گام در راستای جستجو ضرب شده تا عمل بهینه سازی به تکرار بعدی پیش رود. این راستای جستجو در جهت منفی گرادیان تابع هدف در هر تکرار می باشد. در روش جستجوی خطی، طول گام به گونه‌ای انتخاب می شود که حداکثر کاهش تابع هدف را باعث شود. راه حل دیگر سعی در دنبال کردن یک مسیر پیوسته در راستای تندترین شیب و در یک سری گام های متوالی خیلی کوچک است.

مسئله حل مجموعه‌ای از نامعادلات خطی از زمان فوریه مطرح بوده‌است. برنامه‌ریزی خطی به عنوان یک مدل ریاضی در زمان جنگ جهانی دوم شکل گرفت تا خرج‌ها و بازگشت‌های مالی را طوری سامان بخشد که به کاهش هزینه‌های ارتش و افزایش خسارات دشمن بینجامد. این طرح تا سال ۱۹۴۷ سری باقی ماند. پس از جنگ، بسیاری از صنایع به استفاده از آن پرداختند. پایه‌گذاران این حوزه جورج دانتزیگ منتشرکنندهٔ روش سیمپلکس در سال ۱۹۴۷، جان فون نویمان مطرح‌کننده نظریه دوگانگی

در همان سال، و لئونید کانتروویچ ریاضیدان روس که از تکنیک‌های مشابهی پیش از دانتزینگ استفاده کرد و نوبل سال ۱۹۵۷ را برد هستند. نخستین بار در سال ۱۹۷۹ لئونید خاچیان نشان داد که مسئله برنامه‌ریزی خطی در مرتبه زمانی چندجمله‌ای قابل حل است. اما پیشرفت اساسی‌تر زمانی حاصل شد که نراندرا کارمارکار یک روش نقطه داخلی جدید برای حل این مسائل معرفی کرد. مثال دانتزینگ برای منتصب کردن هفتاد نفر به هفتاد شغل متمایز کارآمدی برنامه‌ریزی خطی را به نمایش می‌گذارد. توان محاسباتی لازم برای آزمودن همهٔ جایگشتهای ممکن این مسئله بسیار بالاست. این تعداد از تعداد ذرات موجود در عالم بیشتر است. با این حال، پیدا کردن پاسخ بهینه با تبدیل مسئله به یک مسئله برنامه‌ریزی خطی و حل آن با روش سیمپلکس تنها لحظه‌ای طول می‌کشد.

الگوریتم‌ها



مجموعه‌ای از محدودیت‌ها (خطوط مرزی) به صورت نامعادلات خطی روی دو متغیر منجر به ایجاد منطقه‌ای از مقادیر ممکن برای آن دو متغیر روی صفحه می‌شود. این منطقه برای مسائل حل‌شدنی به شکل یک چندضلعی محدب است.

الگوریتم سیمپلکس که توسط جورج دانتزینگ شکل گرفت، مسائل برنامه‌ریزی خطی را به این ترتیب حل می‌کند که یک جواب قابل قبول در یکی از رئوس چندضلعی فراهم می‌کند و سپس در راستای اضلاع چندضلعی به طرف رئوسی با مقدار بالاتری از تابع هدف حرکت می‌کند تا این که به نقطه بهینه برسد. اگرچه در عمل این الگوریتم بسیار کارآمد است و می‌تواند با در نظر گرفتن برخی پیش‌گیری‌های مربوط به جلوگیری از ایجاد دور، با اطمینان جواب بهینه مطلق را بیابد، اما در حالاتی که به اصطلاح بدترین حالت نامیده می‌شوند عملکرد بدی دارد. تا حدی که می‌توان مسائل برنامه‌ریزی خطی طراحی کرد که روش سیمپلکس برای حلشان در برخی مراحل زمانی از مرتبه زمانی نمایی نیاز داشته باشد. حتی در دورانی دانشمندان نمی‌دانستند که این مسائل راه حل چندجمله‌ای هم دارند. سرانجام این مسئله را لئونید خاچیان در سال ۱۹۷۹ با ارائه روش بیضوی حل کرد. این روش در بدترین حالت هم دارای زمان اجرای چندجمله‌ای بود. این روش تأثیر چندانی در جنبهٔ عملی مسئله نداشت چرا که همچنان روش سیمپلکس در همه موارد به جز تعداد محدودی از مسائل بهتر عمل می‌کرد. اما اهمیت نظری روش خاچیان غیرقابل‌انکار بود. این روش الهام‌بخش به وجود آمدن نسل جدیدی از راه‌حل‌ها شد که به آن‌ها روش نقطه داخلی گفته می‌شود. در این روش‌ها نقاط داخلی محدوده قابل بررسی متغیرها پیموده می‌شود و به سمت نقطه بهینه حرکت انجام می‌گیرد.

برنامه‌ریزی خطی استوار

در بسیاری از مسایل بهینه‌سازی، تابع هدف یا قیود، تصادفی و در حال تغییر هستند؛ بنابراین پاسخ بهینهٔ مسئله می‌بایست به گونه‌ای بدست آورده شود که برای تمامی تغییرات توابع بهینه باشد.

یک مسئله برنامه‌ریزی خطی (LP) به صورت زیر فرض می‌شود:

اگر در مسئلهٔ فوق پارامترهای C، ثابت و یقینی نباشند، حل مسئله، به حل یک مسئلهٔ مقاوم در برابر تغییرات پارامترها تبدیل می‌شود. برای حل مسائل بهینه‌سازی به صورت مقاوم دو رهیافت وجود دارد:

۱. بدترین حالت
۲. مدل تصادفی

بدترین حالت

در این روش، بازهٔ تغییرات پارامترها مشخص است و مسئله برای بدترین حالتی که ممکن است رخ دهد، حل می‌شود.
در حالت خاص فرض می‌شود تنها پارامتر متغیر تصادفی باشد. در صورتی کهفرض شود، (عضو یک بیضی باشد).
قید نامساوی را می‌توان به صورت زیر نوشت:
با جایگذاری در خواهیم داشت:
در نتیجه مسئله برنامه‌ریزی خطی مقاوم به یک مسئلهٔ SOCP تبدیل می‌شود:

مدل تصادفی

در این روش، متغیر در بازه مشخص نبوده و دارای توزیع آماری است بنابراین بهینه‌سازی برای بدترین حالت امکان‌پذیر نبوده و مسئله با توجه به توزیع آماری متغیر حل می‌شود.
یک بردار گوسی با میانگین و تابع کوواریانس فرض می‌شود.
قید مسئله به صورت زیر خواهد شد:
تابع CDF متغیر گوسی است.
با جایگذاری قید بدست آمده در مسئله اصلی، مسئله به صورت زیر خواهد شد:
مسئله برنامه‌ریزی خطی مقاوم حاصل به شکل یک مسئلهٔ SOCP است.

بهینه‌سازی ریاضی یا **برنامه‌ریزی ریاضی** در ریاضیات، اقتصاد، مدیریت به برگزیدن بهترین عضو از یک مجموعه از اعضای دست یافتنی اشاره می‌کند. در ساده‌ترین شکل تلاش می‌شود که با گزینش نظام مند داده‌ها از یک مجموعه قابل دستیابی و محاسبه مقدار یک تابع حقیقی مقدار بیشینه و کمینه آن به دست آید. در قلمرو مدیریت اصولاً دو فرض وجود دارد:

۱. نبود محدودیت در منابع
۲. وجود محدودیت در منابع

که اگر فرض نخست را بپذیریم می‌توان از روشهایی چون گرفتن مشتق اول و دوم مقدار بهینه را برآورد کرد و چنانچه فرض دوم پذیرفته شود بسته به نوع مسائل سازمانی واقتصادی می‌توان مدلهایی را چون:مدل خطی، عدد صحیح، آرمانی، غیر خطی، ضریب لاگرانژ، قطعی یا احتمالی و

انواع بهینه‌سازی

روش‌های تحلیلی

روش‌های تحلیلی بیشتر به دنبال حل دقیق مسائل هستند. از این رو شامل مشتق‌گیری و یافتن پاسخ بهینه‌اند فایده اصلی این نوع از الگوریتم‌های بهینه‌سازی تضمین جواب بهینه است، اما استفاده از آنها در مسائل با پیچیدگی بالا یا مسائل بزرگ یا دارای تابع گسسته دشوار است.

روش‌های فراابتکاری

روش‌های فرابتکاری یا فرااکتشافی برای حل مسائل بزرگتر و با توابع بدرفتار مناسب‌ترند. اگرچه این روش‌ها نمی‌توانند رسیدن به جواب بهینه را تضمین کنند. الگوریتم ژنتیک و تصعید شبیه‌سازی شده مثال‌هایی از این الگوریتم‌ها هستند.

هدف بهینه‌سازی

در بهینه‌سازی هدفمان حفظ کردن فرمول یا رابطه‌ای نیست. تنها از دانش قبلی خود استفاده می‌کنیم. به دنبال بیشترین یا کمترین مقدار برای یک کمیت هستیم.

گام‌های حل مسائل بهینه‌سازی ریاضی

۱. گام اول یافتن تابع تغییرات کمیتی است که با آن سر و کار داریم. در این گام باید نسبت به تعیین دامنهٔ تابع دقت کنیم.
۲. گام دوم، پیدا کردن نقاط بحرانی تابعی است که در گام اول یافتیم.
۳. گام سوم، پیدا کردن اکسترمم تابع است. بسته به خواستهٔ مسئله، گاهی این اکسترمم ماکزیمم است و گاهی مینیمم.

بهینه‌سازی استوار

در برنامه‌ریزی ریاضی معمولاً مسائل با پیش فرض قطعی بودن داده‌ها از قبل حل می‌شوند حال آنکه در دنیای واقعی اکثر داده‌ها دچار عدم قطعیت اند. پیش فرض اصلی برنامه‌ریزی‌های ریاضی توسعه مدل بر اساس داده‌های صریحاً معین و برابر با مقداری اسمی است. حال آنکه در این گونه از مدل‌ها اثر عدم قطعیت داده‌ها در کیفیت و امکانپذیر بودن جواب‌ها اثری ندارد. در نتیجه در در مسائل دنیای واقعی ممکن است با تغییر یکی از داده‌ها تعداد زیادی از محدودیت‌ها نقض شده و جواب بدست آمده غیر بهینه یا حتی غیرممکن باشد. در نتیجه این بحث سؤال اصلی ساخت جوابی برای مسئله پیش می‌آید که در مقابل این عدم قطعیت داده‌ها مقاوم باشد که اصطلاحاً این پاسخ‌ها را استوار و این دسته از بهینه‌سازی را بهینه‌سازی استوار می‌نامند ایده اولیه در بهینه‌سازی استوار، در نظر گرفتن بدترین سناریوی ممکن و بهینه‌سازی بر اساس بدترین سناریو است. به عنوان مثال فرض کنید ضرایب در یکی از محدودیت ممکن است تغییر کند. در بهینه‌سازی استوار، بدترین حالتی که ممکن است برای محدودیت با توجه به تغییر ضریب ممکن است پیش بیاید در نظر گرفته شده و طبق آن بهینه‌سازی انجام می‌شود.

مهم‌ترین کاستی این روش محتاطانه عمل کردن آن است. ممکن است این روش کاربرد عملی زیادی نداشته باشد؛ ولی به عنوان ابزاری برای تصمیم‌گیری بسیار مفید خواهد بود. اولین گام‌ها در این مورد توسط Soyster برداشته شد. وی مدلی ارائه کرد تا در آن جوابی ممکن برای تمامی داده‌های متعلق به یک مجموعه محدب ساخته شود؛ ولی مدل ارائه شده به دلیل زیاده روی در محتاطانه عمل کردن با جواب بهینه اسمی فاصله‌ای زیاد داشت. پس از او نیز افرادی مانند Ben-Tal و Nemirovski و Bertsimas به ارائه مدل‌های بهتری پرداختند تا به اندازه ممکن فاصله از مقدار بهینه را کاهش دهند.

گونه‌ها

به طور کلی مدل‌های استوار به دو گونه اصلی **بازه محور** و **سناریو محور** تقسیم‌بندی می‌شوند. در حالت **بازه محور** می‌گوییم، بازه‌ای که در آن قرار است ضریب در بهینه‌سازی تغییر کند چیست. بهینه‌سازی جورج دانتزیگ الگوریتم غیر مرکب یکی از بهترین الگوریتم‌ها برای برنامه‌ریزی خطی است در بهینه‌سازی ریاضیاتی، الگوریتم غیر مرکب دانتزیگ، (یا روش غیر مرکب) یک الگوریتم محبوب برای برنامه‌نویسی خطی است. مجله پردازش در علوم و مهندسی این الگوریتم را به عنوان یکی از ۱۰ الگوریتم برتر در قرن بیستم معرفی کرده‌است. نام این الگوریتم از مفهوم یک غیر مرکب برگرفته شده و توسط ت.س.موتزگین[۷] پیشنهاد شده‌است.

سیمپلیس‌ها در واقع در این الگوریتم مورد استفاده قرار نگرفته‌اند، ولی یک تفسیر از آن این است بر روی مخروط‌های سیمپلیسی، و اینها سیمپلیس‌هایی منتاسب با محدویت‌های اضافی می‌شوند. مخروط‌های سیمپلیسی مورد سؤال، گوشه‌های (یعنی در همسایگی راس‌ها) یک شی هندسی که یک پولی توپ نامیده می‌شوند. شکل این پولی توپ توسط قیدهای ایجاد شده در تابع هدف، مشخص می‌شود.

مرور

الگوریتم غیر مرکب بر روی برنامه‌هایی در فرم استاندارد عمل می‌کند؛ مسئله‌های خطی به این فرم، کمینه شده مشروط بر این که:

با وجود متغیرهای مسئله، c = (c ۱ , … , c n) {{displaystyle \scriptstyle c\;=\;(c_{1},\,\dots ,\,c_{n}\}
ضریب‌های تابع گفته شده می‌باشند، ماتریس A، یک ماتریس p*n می‌باشد و ثابت‌های b = (b ۱ , … , b p)

{displaystyle b_{j}\geq 0\} . ۰ ≤ j b که {({displaystyle \scriptstyle b\;=\;(b_{1},\,\dots ,\,b_{p}\}
یک شیوهٔ مستقیم برای تبدیل هر برنامه خطی به حالت استاندارد وجود دارد که این در از بین رفتن عمومیت تأثیری ندارد. در اصطلاحات هندسی، منطقه محتمل

A x = b , x i ≥ 0 {\displaystyle \mathbf {A} \mathbf {x} =\mathbf {b} ,\,x_{i}\geq 0}

یک (به احتمالی بیکران) پولی توپ محدب است. یک توصیف صفاتی ساده از نقاط کرانی رأس‌های این پولی توپ وجود دارد ، x = (۱ x , … , n x) {({displaystyle \scriptstyle x\;=\;(x_{1},\,\dots ,\,x_{n}\} یک نقطه کرانی است اگر و فقط اگر ستون بردارهای A i {{displaystyle \scriptstyle A_{i}\} ، جایی که به صورت خطی غیر مستقل باشند. در این مبحث، چنین نقطه‌ای به عنوان یک راه حل پایه‌ای امکان‌پذیر(BFS) شناخته می شود.

می‌توان نشان داد که برای یک برنامه خطی در فرم استاندارد، اگر تابع مقصود یک مقدار کمینه در منطقه محتمل داشته باشد، در نتیجه این مقدار را روی (حداقل) یکی از نقاط کرانی خواهد داشت. این در عنوان خودش مسئله را به یک پردازش با حالت محدود تبدیل می‌کند، زیرا تعداد محدودی از نقاط کرانی وجود دارند، در هر حال تعداد نقاط کرانی به‌طور غیرقابل کنترلی برای کوچکترین برنامه‌های خطی نیز بزرگ است. همچنین می‌توان نشان داد که اگر یک نقطه کرانی، نقطه کمینه تابع مقصود نیست، در نتیجه یک گوشه‌ای وجود دارد که نقطه را در بر می‌گیرد به شکلی که تابع مقصود اکیداً نزولی است بر روی گوشه‌ای که از نقطه دور می‌شود.[۱۵] اگر گوشه کراندار باشد، گوشه به نقطه‌ای دیگر در جایی که تابع مقصود مقدار کمتری دارد، متصل می‌شود، در غیر این صورت تابع مقصود زیر گوشه بی‌کران است و برنامه خطی هیچ راه حلی ندارد. الگوریتم غیر مرکب، این بینش را با گذشتن از تمامی گوشه‌های پولی توپ به نقطه‌های کرانی با مقادیر کمر تابعی، به وجود می‌آورد. این ادامه پیدا می‌کند تا وقتی که به مقدار کمینه دست پیدا کند، یا یک راس بی‌کران، مشاهده شود، این کار ادامه پیدا می‌کند، و این نتیجه‌گیری می‌شود که مسئله جوابی ندارد. الگوریتم همیشه به مام می‌رسد به این دلیل که تعداد رأس‌های پولی توپ، متناهی است؛ همچنین به خاطر این که بین رئوس در یک جهت پرش می‌کنیم، امید داریم که تعداد رئوس پیمایش شده، کم باشد.[۱۵] راه حل یک برنامه خطی در دو مرحله به اجرا می‌رسد. در مرحله اول، که به فاز شناخته می‌شود، یک نقطه کرانی برای شروع جستجو ی شود. وابسته به طبیعت برنامه، این ممکن است کاری ناچیز باشد ولی عموماً با ارائه الگوریتم غیر مرکب به یک نسخه تصحیح شده نسبت به برنامه اولیه، این مسئله قابل حل است. نتایج محتمل فاز ۱، یا یک راه حل پایه‌ای عملی که پیدا می‌شود یا این که ناحیه محتمل خالی است. در مرحله دوم، فاز ۲، الگوریتم غیرمرکب به این گونه اعمال می‌شود که راه حل پایه‌ای عملی در فاز ۱ به عنوان نقطه شروع به آن داده می‌شود. نتایج محتمل در فاز ۲ یا یک راه حل پایه‌ای و عملی بهینه است یا یک گوشه بی‌کران که تابع مقصود در زیر آن بی‌کران است.

فرم استاندارد

تبدیل یک برنامه خطی به یکی به حالت استاندارد، می‌تواند به صورت مقابل اعمال شود.[۱۸] ابتدا برای هر متغیر با یک کران پایین‌تر به غیر از صفر، یک متغیر جدید معرفی می‌شود که نشانگر تفاوت بین متغیر و حد می‌باشد. متغیر اولیه می‌تواند پس از آن با جایگزینی حذف شود. برای مثال، داده شده با این محدودیت

x 1 ≥ 5 {\displaystyle x_{1}\geq 5}

با متغیر جدید، y۱، که برابر است با

y 1 = x 1 − 5 x 1 = y 1 + 5 {\displaystyle {\begin{aligned}y_{1}=x_{1}-5\\x_{1}=y_{1}+5\end{aligned}}}

تساوی دوم برای حذف x_۱ از برنامه خطی استفاده می‌شود. در این روش، تمامی قیدهای با کران پایین تبدیل به محدودیت‌های غیر منفی می‌شوند. دوم، برای هر قید باقی‌مانده، یک متغیر جدید به نام متغیر اسلَک، معرفی می‌شود که هر قید را به یک قید

تساوی تبدیل می‌کند. این متغیر نشانگر تفاوت بین دو طرف نامساوی است و غیر منفی در نظر گرفته شده‌است. برای مثال این

نامساوی‌ها

x 2 + 2 x 3 ≤ 3 − x 4 + 3 x 5 ≥ 2 {\displaystyle {\begin{aligned}x_{2}+2x_{3}&\leq 3\\-x_{4}+3x_{5}&\geq 2\end{aligned}}}

با این جایگزین می‌شود

x 2 + 2 x 3 + s 1 = 3 − x 4 + 3 x 5 − s 2 = 2 s 1 , s 2 ≥ 0 {\displaystyle {\begin{aligned}x_{2}+2x_{3}+s_{1}&=3\\-x_{4}+3x_{5}-s_{2}&=2\\s_{1},\,s_{2}&\geq 0\end{aligned}}}

راه آسانتر این است که دستکاری جبری بر روی نامساوی‌ها در این فرم انجام شود. در نامساوی‌هایی که ≤ وجود دارد مانند نامساوی دوم، بعضی نویسندگان ارجاع می‌دهند به متغیری که به متغیر surplus معروف است. سوم، هر متغیر غیر محدود، از برنامه خطی حذف می‌شود. این می‌تواند به دو راه انجام شود، یکی با حل کردن مسئله برای متغیری در یکی از تساوی‌ها یی که وجود دارد و سپس حذف متغیر با جایگزینی. راه دیگر این است که متغیر را با تفاوت دو متغیر محدود جایگزین کرد. برای مثال اگر ۱z_ بدون قید است کس می‌نویسیم

z 1 = z 1 + − z 1 − z 1 + , z 1 − ≥ 0 {\displaystyle {\begin{aligned}&z_{1}=z_{1}^{+}-z_{1}^{-}\\&z_{1}^{+},\,z_{1}^{-}\geq 0\end{aligned}}}

این تساوی برای حذف ۱z_ از برنامه خطی استفاده می‌شود. زمانی که پروسه به اتمام می‌رسد، منطقه محتمل به فرم زیر خواهد بود ۰Ax=b,x_i≥ همچنین بهتر است که درچه A را تعداد ردیف‌ها در نظر بگیریم. این به هیچ وجه باعث از بین رفتن عمومیت نمی‌شود، زیرا در هر حال یا:

A x = b , x i ≥ 0 {\displaystyle \mathbf {A} \mathbf {x} =\mathbf {b} ,\,x_{i}\geq 0}

دارای تساوی‌های اضافی است که می‌توان آن‌ها را حذف کرد، یا سیستم متناقض است و دستگاه خطی هیچ راه حلی ندارد.[۱۹]

جدول مخروطی

یک دستگاه خطی به فرم استاندارد می‌تواند به صورت ماتریسزیر نمایش داده شود

[1 − c T 0 0 A b] {\displaystyle {\begin{bmatrix}1&-\mathbf {c} ^{T}&0\\0&\mathbf {A} &\mathbf {b} \end{bmatrix}}}

ردیف اول نمایانگر تابع مقصود و باقی ردیف‌ها نشانگر قیدها هستند. (در نظر داشته باشید، نویسنده‌های مختلف، روش‌های مختلفی برای برای نوشتن این دستگاه استفاده می‌کنند)اگر ستون‌های A بتوانندبه گونه‌ای چیده شوند که دربر گیرنده ماتریس هویت باشند که از درجه Pاست (تعداد سطرها در A)در نتیجه جدول یا ماتریس در حالت مخروطی قرار دارد.[۲۰] متغیرهای نظیر ستون‌های ماتریس هویت، متغیرهای پایه‌ای نامیده می‌شوند، در حالی که باقی متغیرها غیرپایه ای یا متغیرهای آزاد هستند. اگر متغیرهای غیر پایه‌ای صفر در نظر گرفته شوند، مقادیر متغیرهای پایه‌ای به راحتی قابل بدست آمدن هستند، به این گونه که با گرفتن ورودی‌هایی در b ، راه حل یک راه حل پایه‌ای محتمل است. برعکس، با داشتن یک راه حل امکان‌پذیر، ستون‌های نظیر متغیرهای غیر صفر می‌توانند به یک ماتریس غیر منفرد ارتقا داد. اگر ماتریس متناظر در امعکوس این ماتریس ضرب شود، نتیجه یک جدول یا ماتریس به فرم. بگذارید

[1 − c B T − c D T 0 0 I D b] {\displaystyle {\begin{bmatrix}1&-\mathbf {c} _{B}^{T}&-\mathbf {c} _{D}^{T}&0\\0&I&\mathbf {D} &\mathbf {b} \end{bmatrix}}}

یک جدول به فرم مخروطی باشد. تغییر شکل اضافی جمع سطرها می‌تواند برای حذف ثابت‌های TB c از تابع مقصود به کار می‌رود. این روش به بیرون کشی قیمت معروف است و یک جدول مخروطی را به جای می‌گذارد.

[1 0 − c ¯ D T z B 0 I D b] {\displaystyle {\begin{bmatrix}1&0&-{\bar {\mathbf {c} }}_{D}^{T}&z_{B}\\0&I&\mathbf {D} &\mathbf {b} \end{bmatrix}}}

جایی که zB مقدار تابع مقصود در راه حل محتمل متناظر است. ضریب‌های به روز شده، که با نام ضریب‌های متناسب هزینه اینیز شناخته می‌شوند، نرخ تغییرات تابع مقصود با توجه به متغیرهای غیر پایه‌ای، هستند.

عملیات محوری

عملیات هندسی جابجایی از یک راه حل محتمل به یک راه حل پایه‌ای محتمل همجوار، با عملیات محوری پیاده‌سازی می‌شود. ابتدا، یک عنصر محوری غیر صفر در یک ستون غیر پایه‌ای انتخاب می‌شود. ردیفی که این عنصر را دربردارد ضربدر معکوس آن می‌شود تا این عنصر به ۱ تبدیل شود، و سپس ضرب شده‌های ردیف با ردیف‌های دیگر برای تغییر ورودی‌های ستون به صفر، جمع می‌شوند. نتیجه این می‌شود که، اگر عنصر محوری در ردیف r قرار دارد، در نتیجه ستون، r امین ستون ماتریس هویت خواهد بود. متغیر این ستون الان یک متغیر پایه‌ای است که با متغیری که متناظر با ستون r ام ماتریس هویت، جابجا می‌شود، البته قبل از عملیات. در حقیقت، متغیر متناظر با ستون محوری وارد دستهٔ متغیرهای پایه‌ای شده و نام آن می‌شود متغیر رونده. جدول همچنان به فرم مخروطی است ولی با مجموعه‌ای از متغیرهای پایه‌ای که با یک عنصر تغییر پیدا کرده‌اند.

مسئلهٔ بهینه‌سازی محدب یا بهینه‌سازی کوژ (به انگلیسی: Convex Optimization) به یافتن مقدار حداقل یک تابع محدب (یا حداکثر یک تابع مقعر) از بین مجموعه‌ای محدب گفته می‌شود. مهمترین مزیت این نوع مسائل بهینه‌سازی در این است که هر نقطهٔ بهینهٔ محلی یک نقطه بهینه سراسری نیز است و هر الگوریتم بهینه‌سازی که یک نقطه بهینهٔ محلی را یافت در حقیقت یک نقطه بهینهٔ سراسری را یافته‌است.

مسئله بهینه‌سازی شبه محدب

مسئله بهینه‌سازی شبه محدب، فرم استاندارد زیر را دارد:

$$\min f_o(x) \quad s.t. \quad f_i(x) \leq 0 \quad i=1,\dots,m \quad , Ax=b$$

که قیود نامساوی محدب هستند و تابع هدف نیز شبه محدب می‌باشد (زمانیکه مسئله محدب باشد تابع هدف نیز محدب است). قیود شبه محدب می‌توانند با معادل محدب شان جایگزین شوند. در این نوشتار بعضی از اختلافات پایه‌ای بین مسائل بهینه‌سازی محدب و شبه محدب نشان داده خواهد شد، همچنین نشان داده می‌شود حل یک مسئله بهینه‌سازی شبه محدب چگونه می‌تواند به حل چند دنباله از مسائل بهینه‌سازی محدب کاهش یابد.

پاسخ‌های بهینه محلی و شرایط بهینگی

مهمترین اختلاف بین بهینه‌سازی محدب و شبه محدب این است که مسائل بهینه‌سازی شبه محدب می‌توانند جواب‌های بهینه محلی داشته باشد. این پدیده می‌تواند حتی در ساده‌ترین مورد، کمینه سازی بدون قید یک تابع شبه محدب روی R دیده شود.

برای یک مسئله بهینه‌سازی محدب، x بهینه است اگر:

$$\bigtriangledown f_0(x)^T(y-x) \geq 0 \quad for \, all \, y \in X$$

انواع شرایط بهینگی برای مسائل بهینه‌سازی شبه محدب با توابع هدف مشتق پذیر برقرار است. X را فضای شدنی برای مسئله بهینه‌سازی شبه محدب در نظر بگیرید، در این صورت x بهینه است اگر:

$x \in X , \nabla f 0 (x) T (y - x) > 0 f o r a l l y \in X$ {\displaystyle x\in X,\quad \quad \bigtriangledown f_{0}(x)^{T}(y-x)>0\,\,\,for\,\,\,all\,\,\,y\in X}

دو تفاوت مهم بین معیار فوق و معیار بهینگی برای مسائل محدب وجود دارد:

۱- معیار مسائل شبه محدب برای بهینگی پاسخ شرط کافی است و برقراری آن برای نقطه بهینه ضروری نیست اما برقراری رابطه فوق برای مسائل محدب شرط لازم و کافی برای بهینگی x {\displaystyle x\} می‌باشد.

۲-معیار فوق نیازمند این است که گرادیان تابع هدف غیر صفر باشد در حالی که در رابطه بهینگی مسائل محدب اینگونه نیست، در واقع زمانی که $\nabla f (x) = 0$ {\displaystyle \bigtriangledown f_{0}(x)=0\} ، معیار بهینگی مسائل محدب صادق است و x {\displaystyle x\} نقطه بهینه است.

یک روش کلی برای مسائل بهینه‌سازی شبه محدب وابسته به نمایش مجموعه‌های زیرسطحی از یک تابع شبه محدب است. ϕt : $R \ni t \rightarrow n R$ {\displaystyle \phi _{t}:R^{n}\rightarrow \,\,t\in R\} را به عنوان مجموعه‌ای از توابع محدب که در رابطه زیر صدق می‌کنند در نظر بگیرید.

$f 0 (x) \leq t \Leftrightarrow \phi t (x) \leq 0$ {\displaystyle f_{0}(x)\leq t\Leftrightarrow \phi _{t}(x)\leq 0}

و برای هر x {\displaystyle x\} $\phi t (x)$ ، {\displaystyle \phi _{t}(x)\} یک تابع غیر صعودی از t {\displaystyle t\} است. فرض کنید $p *$ {\displaystyle p^{*}\} جواب بهینه مسئله بهینه‌سازی شبه محدب باشد. اگر مسئله امکان‌سنجی زیر،

$f i n d x s . t . \phi t (x) \leq 0 , f i (x) \leq 0 , A x = b$ {\displaystyle find\quad x\quad s.t.\quad \phi _{t}(x)\quad \leq 0,\quad f_{i}(x)\leq 0,\quad Ax=b}

شدنی باشد سپس $t \geq p *$ {\displaystyle p^{*}\leq t\} را خواهیم داشت. در طرف مقابل اگر مسئله فوق نشدنی باشد می‌توانیم نتیجه بگیریم $t \leq p *$ {\displaystyle p^{*}\geq t\} خواهد بود. مسئله فوق یک مسئله امکان‌سنجی محدب است چون قیود نامساوی محدب هستند و قید تساوی نیز خطی است؛ بنابراین می‌توانیم بررسی کنیم آیا مقدار $p *$ {\displaystyle p^{*}\} کمتر یا بیشتر از مقدار t {\displaystyle t\} ای است که با حل مسئله امکان‌سنجی فوق به دست می‌آید. اگر مسئله امکان‌سنجی شدنی باشد پس خواهیم داشت $t \geq p *$ {\displaystyle p^{*}\leq t\} و هر نقطه شدنی مثل x {\displaystyle x\} برای مسئله شبه محدب نیز شدنی است و رابطه $t \geq f (x)$ {\displaystyle \} را ارضا می‌کند. اگر مسئله امکان‌سنجی محدب نشدنی باشد می‌توان نتیجه گرفت $t \leq p *$ {\displaystyle p^{*}\geq t\} خواهد بود

بهینه‌سازی ترکیباتی (به انگلیسی: Combinatorial Optimization) شاخه‌ای از بهینه‌سازی است که به آن دسته از مسایل بهینه‌سازی می‌پردازد که در آن‌ها مجموعه پاسخ‌های امکان‌پذیر گسسته است یا می‌تواند به صورت گسسته درآید و هدف پیدا کردن بهترین پاسخ از بین این پاسخ‌ها است. بهینه‌سازی ترکیباتی شاخه‌ای از ریاضیات کاربردی و علوم رایانه و مرتبط با تحقیق در عملیات، نظریه الگوریتم و نظریه پیچیدگی محاسباتی است که در محل تلاقی چندین رشته ازجمله هوش مصنوعی، ریاضیات و مهندسی نرم‌افزار قرار دارد. الگوریتم‌های فراابتکاری یا فراتکاملی یا فرااکتشافی نوعی از الگوریتم‌های تصادفی هستند که برای یافتن پاسخ بهینه به کار می‌روند.

روش‌ها و الگوریتم‌های بهینه‌سازی به دو دسته الگوریتمهای دقیق (exact) و الگوریتم‌های تقریبی (approximate algorithms) تقسیم‌بندی می‌شوند. الگوریتم‌های دقیق قادر به یافتن جواب بهینه به صورت دقیق هستند اما در مورد مسائل بهینه‌سازی سخت کارایی کافی ندارند و زمان اجرای آن‌ها متناسب با ابعاد مسائل به صورت نمایی افزایش می‌یابد. الگوریتم‌های تقریبی قادر به یافتن جواب‌های خوب (نزدیک به بهینه) در زمان حل کوتاه برای مسائل بهینه‌سازی سخت هستند. الگوریتم‌های تقریبی نیز به سه دسته الگوریتم‌های ابتکاری (heuristic) و فراابتکاری (meta-heuristic) و فوق ابتکاری (hyper heuristic) بخش‌بندی می‌شوند. دو مشکل اصلی الگوریتم‌های ابتکاری، گیر افتادن آن‌ها در نقاط بهینه محلی، همگرایی زودرس

به این نقاط است. الگوریتم‌های فراابتکاری برای حل این مشکلات الگوریتم‌های ابتکاری ارائه شده‌اند. در واقع الگوریتم‌های فراابتکاری، یکی از انواع الگوریتم‌های بهینه‌سازی تقریبی هستند که دارای راهکارهای برونرفت از نقاط بهینه محلی هستند و قابلیت کاربرد در طیف گسترده‌ای از مسائل را دارند. رده‌های گوناگونی از این نوع الگوریتم در دهه‌های اخیر توسعه یافته‌است[۳]که همه این‌ها زیر مجموعه الگوریتم فراابتکاری می‌باشند.

- ۱ دسته‌بندی الگوریتم‌های فراابتکاری
- ۲ الگوریتم‌های فراابتکاری بر پایه جمعیت
- ۳ الگوریتم‌های متداول فراابتکاری مبتنی بر یک جواب
- ۴ پیاده‌سازی الگوریتم‌های فراابتکاری
- ۵ جستارهای وابسته
- ۶ منابع

دسته‌بندی الگوریتم‌های فراابتکاری

معیارهای مختلفی می‌تواند برای طبقه‌بندی الگوریتم‌های فراابتکاری استفاده شود:

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

الگوریتم‌های فراابتکاری بر پایه جمعیت

از الگوریتم‌های شناخته شده فراابتکاری بر پایه جمعیت می‌توان الگوریتم‌های تکاملی[۵] (الگوریتم ژنتیک، برنامه‌ریزی ژنتیک، ...)، بهینه‌سازی کلونی مورچگان،[۶] کلونی زنبورها،[۷] روش بهینه‌سازی ازدحام ذرات، الگوریتم بهینه‌سازی جنگل[۸]، الگوریتم بهینه سازی Battle Royal[۹]، الگوریتم قهرمانی در لیگهای ورزشی، بهینه‌سازی ملهم از فیزیک نور، الگوریتم ریشه-پاجوش و الگوریتم چکه آبهای هوشمند را نام برد.

در سال‌های اخیر الگوریتم‌های فرابتکاری جدیدی با توجه به موجودات زنده موجود در طبیعت (الهام گرفته از طبیعت) توسعه داده شده‌اند که از معروف‌ترین آن‌ها می‌توان به الگوریتم بهینه سازی گرگ خاکستری ، الگوریتم بهینه سازی سنجاقک ، الگوریتم بهینه سازی گرده افشانی گل ها، الگوریتم بهینه سازی نهنگ یا وال ، الگوریتم بهینه سازی ملخ، و الگوریتم بهینه سازی کلونی پنگوئن های امپراتور نام برد. اخیرا، به نظر می رسد روند توسعه جدیدی از الگوریتم ها با الهام گرفتن از علم دیرینه شناسی و علوم باستان آغاز شده است. این الگوریتم ها که در دسته جدید الهام گرفته از دوران باستان، قرار می گیرند با بررسی ویژگی های دوران باستان سعی می کنند نوعی بهینه سازی ایجاد کنند

الگوریتم‌های متداول فراابتکاری مبتنی بر یک جواب

از الگوریتم‌های متداول فراابتکاری مبتنی بر یک جواب می‌توان الگوریتم جستجوی ممنوعهو الگوریتم تبرید شبیه‌سازی شدهرا نام برد.

پیاده‌سازی الگوریتم‌های فراابتکاری

فرایند طراحی و پیاده‌سازی الگوریتم‌های فراابتکاری دارای سه مرحلهٔ متوالی است که هر کدام از آن‌ها دارای گام‌های مختلفی هستند. در هر گام فعالیت‌هایی باید انجام شود تا آن گام کامل شود. مرحلهٔ ۱ آماده‌سازی است که در آن باید شناخت دقیقی از مسئله‌ای که می‌خواهیم حل کنیم بدست آوریم، و اهداف طراحی الگوریتم فراابتکاری برای آن باید با توجه به روش‌های حل موجود برای این مسئله به‌طور واضح و شفاف مشخص شود. مرحلهٔ بعدی، ساخت نام دارد. مهم‌ترین اهداف این مرحله انتخاب استراتژی حل، تعریف معیارهای اندازه‌گیری عملکرد، و طراحی الگوریتم برای استراتژی حل انتخابی می‌باشد. آخرین مرحله پیاده‌سازی است که در آن پیاده‌سازی الگوریتم طراحی شده در مرحلهٔ قبل، شامل تنظیم پارامترها، تحلیل عملکرد، و در نهایت تدوین و تهیه گزارش نتایج باید انجام شود

در ریاضیات، برنامه‌سازی غیر خطی (NLP Nonlinear programming) فرایند حل مسئله بهینه سازی است که در آن برخی از محدودیت ها یا خود تابع هدف غیر خطی است. این مسئله بهینه سازی، یک سیستم از برابری‌ها و نابرابری‌ها بر روی مجموعه‌ای از متغیرهای ناشناخته حقیقی، در یک تابع هدف که باید کمینه یا بیشینه شود.

روش‌های برنامه سازی غیر خطی[ویرایش]

روش لاگرانژ[ویرایش]

در این روش تابع هدف به صورت تابع F در نظر گرفته می‌شود.

این تابع بر زیرمجموعه‌ای چون X از فضای اقلیدسی تعریف شده‌است (پس عناصر X می‌توانند بردار باشند). این زیرمجموعه، توسط قیود به شکل g(x)=b تعریف می‌شود. تابع لاگرانژ در این حالت عبارت خواهد بود از: ((y)=F(x)+y.(b-g(x،L(x

شرایط لازم برای حل مسئله را می‌توان از طریق یافتن نقاط بحرانی تابع لاگرانژ (ماکزیمم‌سازی بدون قید) به دست آورد.

روش برنامه‌ریزی مرتبه دوم

برنامه‌ریزی مرتبه دوم(QP) روشی برای مینیمم سازی توابع مرتبه دوم n متغیره با m محدودیت خطی نامساوی یا مساوی یا هر دو است. مسائل برنامه‌ریزی مرتبه دوم ساده‌ترین فرم مسائل برنامه‌ریزی غیر خطی با محدودیت نا مساوی می‌باشد.

روش گرادیان کاهش یافته عمومی

این الگوریتم برای محدودیت‌های خطی اصلاح شده که تابع هدف و محدودیت آن‌ها غیر خطی است محسوب می‌شود. در اصل روش محدودیت‌های خطی یا خطی شده را شامل می‌شود و متغیر جدید با محدودیت تعریف خواهد شد. بیشتر روش‌های حل مسائل برنامه‌ریزی غیر خطی عمومی شامل خطی کردن مسئله و به کار بردن تکنیک برنامه‌ریزی خطی است که به‌طور خلاصه مراحل زیر طی می‌شود.

- به دست آوردن مدل با نقاط عملیاتی و خطی کردن تمام محدودیت‌های تابع هدف حول نقاط عملیاتی. بطوریکه مسئله به فرم برنامه‌ریزی خطی تبدیل شود. سپس استفاده از برنامه‌ریزی خطی برای حل مسئله خطی.
- تکرار روش برنامه‌ریزی خطی برای رسیدن به جواب مناسب با خطی کردن توابع محدودیت‌ها و تابع هدف و چنانچه به جواب مناسب نرسید با خطی کردن دوباره محدودیت‌ها و توابع هدف حول نقطه جدید optimum مسئله پیدا می‌شود.

در روش‌های ذکر شده ممکن است روش به همگرایی نرسد و این خود یکی از معایب روش‌های فوق است به واقع بهترین الگوریتم عمومی حاضر استفاده از الگوریتم گرادیان کاهش یافته عمومی است.

کاربرد برنامه سازی غیر خطی

یکی از مسائل پرکاربرد و معمول برای توابع غیر خطی و برنامه‌سازی غیر خطی مسائل مربوط به بهینه‌سازی است. مثلاً بهینه‌سازی هزینه حمل و نقل با انتخاب روش یا روش‌هایی از میان چندین روش نقل و انتقال است که هرکدام ظرفیت‌ها و محدودیت‌های متفاوتی دارند. به عنوان مثال نقل و انتقال نفت خام با انتخاب روش‌های ترکیبی از خط لوله، تانکر، راه آهن، حمل کننده‌های دریایی که هرکدام توابع هزینه‌ای متفاوتی دارند می‌تواند در نهایت به ما یک تابع غیر خطی از هزینه بدهد.

نمایش فرمولی مسئله‌های بهینه‌سازی

مسئله بهینه‌سازی می‌تواند به صورت‌های مختلفی بیان شود مثلاً یکی از ساده‌ترین حالت‌ها این است که به صورت زیر بیان شود:

$\max x \in X f (x) {\displaystyle \max _{x\in X}f(x)}$

برای بیشینه کردن بعضی از متغیرها مانند توان عملیاتی محصول یا

$\min x \in X f (x) {\displaystyle \min _{x\in X}f(x)}$

برای کمینه کردن تابع هزینه در جایی که داریم:

$f : R n \longrightarrow R {\displaystyle f:R^{n}\to R}$

$X \subseteq R n . {\displaystyle X\subseteq R^{n}.}$

روش‌های حل مسئله

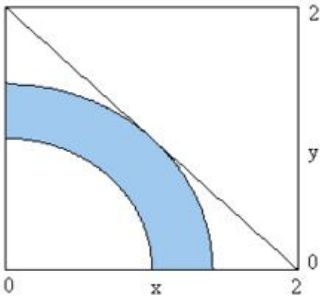
اگر تابع هدف مسئله به صورت f یک تابع خطی باشد و فضای محدوده یک Polytope باشد این مسئله یک مسئله برنامه‌نویسی خطی است که با روش‌های خطی شناخته شده به راحتی حل می‌شود.

اگر تابع هدف یک تابع Concave (مسئله به حداکثر رسانی) یا یک تابع Convex (مسئله به حداقل رسانی) باشد و محدودیت‌ها از نوع محدب (convex) باشد، آنگاه مسئله یک مسئله محدب (convex) نامیده می‌شود و روش‌ها و متدهای عمومی برای بینه سازی محدب (Convex Optimization)می‌توانند در اغلب این مسئله‌ها مورد استفاده قرار گیرند. اگر تابع هدف نسبت تابعی مقعر (Concave) و محدب (Convex) باشد و محدودیت‌ها به صورت محدب باشد، این مسئله می‌تواند به یک مسئله بهینه‌سازی محدب تبدیل شود که در آن از تکنیک‌های برنامه‌نویسی کسری استفاده می‌شود.

روش‌ها و متدهای زیادی برای حل مسئله‌های غیر محدب وجود دارند. یکی از این روش‌ها استفاده از فرمولاسیون‌های ویژه مسائل برنامه‌نویسی خطی است. یکی دیگر از این روش‌ها استفاده از روش شاخه و حد است که مسئله در آن به زیر بخش‌هایی برای حل به صورت محدب یا تقریب خطی تقسیم می‌شود.

مثال‌ها

نمونه دو بعدی



تقاطع خط با فضای محدوده، راه حل را نشان می‌دهد.

یک مسئله ساده می‌تواند با محدودیت‌های زیر تعریف شود:

$$X_1 \geq 0$$

$$X_2 \geq 0$$

$$X_1^2 + X_2^2 \leq 1$$

$$X_1^2 + X_2^2 \geq 2$$

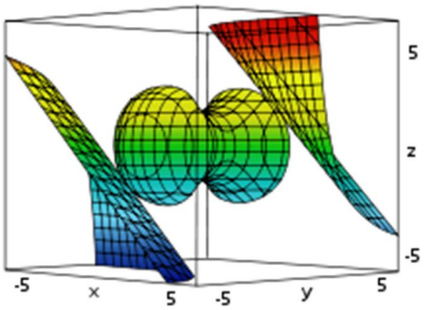
با تابع هدفی به صورت زیر که باید بیشینه شود:

$$f(\mathbf{X}) = X_1 + X_2$$

درحالی که: $\mathbf{X} = (X_1, X_2)$.

حل دو بعدی مسئله.

نمونه سه بعدی



اشتراک سطح بالایی با فضای محدوده در وسط، راه حل را نمایش می‌دهد.

نمونه دیگری از مسئله می‌تواند به صورت زیر تعریف شود:

$$X_1^2 + X_2^2 - X_3^2 \geq 2$$

$$X_1^2 + X_2^2 + X_3^2 \geq 10$$

با تابع هدفی به صورت زیر که باید بیشینه شود:

$$f(\mathbf{X}) = X_1 X_2 + X_2 X_3$$

درحالی که: $\mathbf{X} = (X_1, X_2, X_3)$.

- درس یکم: مقدمه و آشنایی با درس
  - مسئله بهینه سازی خطی و غیر خطی چیست؟
  - کاربرد مسائل بهینه سازی
  - معرفی جعبه ابزار متلب برای حل مسائل بهینه سازی خطی و غیر خطی

- توضیحات اجمالی راجع به درس

- درس دوم: مسائل بهینه سازی خطی
  - مسئله بهینه سازی خطی چیست؟
  - مدل کردن مسئله بهینه سازی خطی و حل آن با استفاده از نرم افزار متلب
  - تشریح پارامترهای مربوط به جعبه ابزارهای مورد نیاز
  - مدل سازی و حل مسئله حمل و نقل با استفاده از روش بهینه سازی خطی با متلب
- درس سوم: حل مسائل بهینه سازی نامقید غیرخطی
  - مسئله بهینه سازی نامقید غیرخطی چیست؟
  - مدل کردن مسئله بهینه سازی نامقید غیرخطی و حل آن با استفاده از نرم افزار متلب
  - تشریح پارامترهای مربوط به جعبه ابزارهای مورد نیاز
  - حل چند مسئله مهم و کاربردی از بهینه سازی نامقید با استفاده از روش های ذکر شده
- درس چهارم: حل مسائل بهینه سازی مقید غیرخطی
  - مسئله بهینه سازی مقید غیرخطی چیست؟
  - مدل کردن مسئله بهینه سازی مقید غیرخطی و حل آن با استفاده از نرم افزار متلب
  - تشریح پارامترهای مربوط به جعبه ابزارهای مورد نیاز
  - حل چند مسئله مهم و کاربردی از بهینه سازی مقید با استفاده از روش های ذکر شده
- درس پنجم: حل مسائل بهینه سازی درجه دوم
  - مسئله بهینه سازی درجه دوم چیست؟
  - مدل کردن مسئله بهینه سازی درجه دوم و حل آن با استفاده از نرم افزار متلب
  - تشریح پارامترهای مربوط به جعبه ابزارهای مورد نیاز
  - مدل مسئله ماشین بردار پشتیبان و حل آن توسط جعبه ابزار متلب
- درس ششم: حل مسئله بهینه سازی کمترین مربعات
  - مسئله بهینه سازی کمترین مربعات چیست؟
  - مدل کردن مسئله بهینه سازی کمترین مربعات و حل آن با استفاده از نرم افزار متلب
  - تشریح پارامترهای مربوط به جعبه ابزارهای مورد نیاز
  - حل یک مسئله کاربردی از بخش کمترین مربعات با استفاده از متلب
- درس یکم: مقدمه ای بر عدم قطعیت، ریسک و استواری
  - تعریف عدم قطعیت
  - انواع عدم قطعیت
  - تعریف ریسک
  - تعریف استواری
  - استواری بهینگی
  - استواری شدنی بودن
- درس دوم: رویکردهای مواجهه با عدم قطعیت زمان
  - برنامه ریزی تصادفی
  - برنامه ریزی تصادفی با محدودیت های احتمالی

- برنامه ریزی تصادفی با دستاویز
- بهینه سازی فازی
- رویکرد برنامه ریزی ریاضی انعطاف پذیر
- رویکرد برنامه ریزی امکانی
- بهینه سازی استوار
- بهینه سازی ترکیبی
- مزایا و معایب انواع رویکردها

▪ درس سوم: بهینه سازی استوار
- روش سویستر (Soyster)
- ارائه مثال کاربردی از روش سویستر
- روش بن تال و نیمروفسکی (Nemirovski & Ben-Tal)
- ارائه مثال کاربردی از روش بن تال و نمیروفسکی
- روش بن تال و همکاران
- ارائه مثال کاربردی از روش بن تال و همکاران
- روش برتسیماس و سیم (Sim & Bertsimas)
- ارائه مثال کاربردی از روش برتسیماس و سیم
- روش مول وی (Mulvey) و همکاران
- ارائه مثال کاربردی از روش مول وی و همکاران
- روش Leung و همکاران
- ارائه مثال کاربردی از روش Leung و همکاران
- روش برتسیماس و تیلا
- ارائه مثال کاربردی از روش برتسیماس و تیلا
- برنامه ریزی استوار جدید
- ارائه مثال کاربردی از برنامه ریزی استوار جدید

▪ درس چهارم: بهینه سازی امکانی استوار
- برنامه ریزی امکانی استوار
- حل تمرین کاربردی از برنامه ریزی امکانی استوار
- مدل برنامه ریزی امکانی استوار مبتنی بر اعتبار
- ارائه مثال کاربردی از برنامه ریزی امکانی استوار مبتنی بر اعتبار
- برنامه ریزی امکانی استوار چندهدفه
- ارائه مثال کاربردی از برنامه ریزی امکانی استوار چندهدفه
- برنامه ریزی امکانی استوار چندهدفه توسعه یافته
- ارائه مثال کاربردی از برنامه ریزی امکانی استوار چندهدفه توسعه یافته

▪ درس پنجم: بهینه سازی فازی – استوار
- برنامه ریزی انعطاف پذیر استوار
- ارائه مثال کاربردی از برنامه ریزی انعطاف پذیر استوار

- برنامه ریزی استوار امکانی - انعطاف پذیر
- ارائه مثال کاربردی از برنامه ریزی استوار امکانی - انعطاف پذیر
- رویکرد استوار ترکیبی با برنامه ریزی انعطاف پذیر - امکانی
- ارائه مثال کاربردی از رویکرد استوار ترکیبی با برنامه ریزی انعطاف پذیر - امکانی

▪ درس ششم: بهینه سازی ترکیبی – استوار

- روش فیچتی و موناکی (Monaci & Fischetti)
- ارائه مثال کاربردی از روش فیچتی و موناکی
- روش Aghezzaf و همکاران
- ارائه مثال کاربردی از روش Aghezzaf و همکاران
- برنامه ریزی تصادفی فازی رباست (Robust)
- ارائه مثال کاربردی از برنامه ریزی تصادفی فازی رباست

بهینه سازی استوار (robust optimization) در سالهای اخیر تحقیقات زیادی جهت در نظر گرفتن عدم قطعیت داده ها در مدلهای ریاضی صورت گرفته است. این تحقیقات به توسعه روشهای بهینه سازی استوار منجر شده است. عدم قطعیت می تواند بر روی بهینگی و موجه بودن مسائل تأثیر بگذارد معمولاً از بهترین برآورد داده ها جهت به کارگیری در مدل های ریاضی استفاده می شود که به این داده ها، داده های اسمی گفته می شود. اولین مدل بهینه سازی استوار توسط سویستر ارائه گردید، این مدل به ساختن جوابهای شدنی برای یک مجموعه محدب می پرداخت. جوابهای مدل سویستر بسیار محافظه کارانه بودند، به گونه ای که در مقابل تضمین استواری جواب از بهینگی صرف نظر می شد. در گام بعدی جهت توسعه بهینه سازی استوار مدل بنتال و نمیروسکی و بطور مستقل از آن القاوی و همکارانش ارائه گردید. مدل آنها دارای دو مشکل بود اول اینکه سختی محاسباتی مسئله را افزایش می داد و دوم اینکه هیچ تضمین احتمالی جهت شدنی بودن مسئله ارائه نمی کرد. چهارچوب کاری مدل بنتال و نمیروسکی غیر خطی بود. برتسیماس و سیم رویکردی را ارائه کردند که در آن تعاملی بین بهینگی و استواری وجود داشت. مدل آنها یک مدل خطی می باشد که به تعدیل سطح محافظه کاری جواب استوار می پردازد. از ویژگی های مدل برتسیماس و سیم می توان به خطی بودن و همچنین قابلیت کنترل محافظه کاری جوابهای استوار به کمک پارامتری به عنوان هزینه استواری اشاره نمود. از دیگر قابلیت های این مدل می توان به قابل استفاده بودن در مسائل عدد صحیح اشاره کرد .در برنامه‌ریزی ریاضی معمولاً مسائل با پیش فرض قطعی بودن داده‌ها از قبل حل می‌شوند حال آنکه در دنیای واقعی اکثر داده‌ها دچار عدم قطعیت اند. پیش فرض اصلی برنامه‌ریزی‌های ریاضی توسعه مدل بر اساس داده‌های صریحاً معین و برابر با مقداری اسمی است. حال آنکه در این گونه از مدل‌ها اثر عدم قطعیت داده‌ها در کیفیت و امکانپذیر بودن جواب‌ها اثری ندارد. در نتیجه در در مسائل دنیای واقعی ممکن است با تغییر یکی از داده‌ها تعداد زیادی از محدودیت‌ها نقض شده و جواب بدست آمده غیر بهینه یا حتی غیرممکن باشد. در نتیجه این بحث سؤال اصلی ساخت جوابی برای مسئله پیش می‌آید که در مقابل این عدم قطعیت داده‌ها مقاوم باشد که اصطلاحاً این پاسخ‌ها را استوار و این دسته از بهینه‌سازی را بهینه‌سازی استوار می‌نامند .ایدهٔ اولیه در بهینه‌سازی استوار، در نظر گرفتن بدترین سناریوی ممکن و بهینه‌سازی بر اساس بدترین سناریو است. به عنوان مثال فرض کنید ضرایب در یکی از محدودیت ممکن است تغییر کند. در بهینه‌سازی استوار، بدترین حالتی که ممکن است برای محدودیت با توجه به تغییر ضریب ممکن است پیش بیاید در نظر گرفته شده و طبق آن بهینه‌سازی انجام می‌شود .مهم‌ترین کاستی این روش محتاطانه عمل کردن آن است. ممکن است این روش کاربرد عملی زیادی نداشته باشد؛ ولی به عنوان ابزاری برای تصمیم‌گیری بسیار مفید خواهد بود .تاریخچه اولین گام‌ها در این مورد توسط Soyster برداشته شد. وی مدلی ارائه کرد تا در آن جوابی ممکن برای تمامی داده‌های متعلق به یک مجموعه محدب ساخته شود؛ ولی مدل ارائه شده به دلیل زیاده روی در محتاطانه عمل کردن با جواب بهینه اسمی فاصله‌ای زیاد

داشت .پس از او نیز افرادی مانند Ben-Tal و Nemirovski و Birtsymas به ارائه مدل‌های بهتری پرداختند تا به اندازه ممکن فاصله از مقدار بهینه را کاهش دهند.

تقسیم بندی بهینه سازی چند هدفه در متلب

مسائل بهینه سازی از نظر تعداد توابع هدف و معیارهای بهینه سازی، به دو نوع تقسیم پذیر هستند: (۱) مسائل بهینه سازی تک هدفه و (۲) مسائل بهینه سازی چند هدفه. در مسائل بهینه سازی تک هدفه، هدف از حل مساله بهبود یک شاخص عملکرد (Performance Index) یگانه است که مقدار کمینه یا بیشینه آن، کیفیت پاسخ به دست آمده را به طور کامل منعکس می کند، اما در برخی موارد، نمی توان صرفا با اتکا به یک شاخص، یک پاسخ فرضی برای مساله بهینه سازی را امتیازدهی نمود. در این نوع مسائل، ناگزیریم که چندین تابع هدف یا شاخص عملکرد را تعریف نماییم و به طور همزمان، مقدار همه آن ها را بهینه کنیم.

فهرست سرفصل ها و رئوس مطالب مطرح شده در این مجموعه آموزشی، در ادامه آمده است:

- مبانی بهینه سازی چند هدفه و بیان تفاوت های آن با مساله بهینه سازی یک هدفه
- تقسیم بندی روش های بهینه سازی چند هدفه
- روش های بهینه سازی چند هدفه کلاسیک
  - روش مجموع وزن دار (Weighted Sum)، مزایا و معایب آن
  - روش برنامه ریزی آرمانی (Goal Programming)
  - روش نیل به آرمان (Goal Attainment)
- روش چبیشف (Chebyshev)، به عنوان حالت کلی روش های مبتنی بر آرمان
- روش تبدیل به قید (ε-Constrainet) (بخوانید Epsilon Constraint)
- مقدمه سازی برای طرح الگوریتم های تکاملی چند هدفه
- جمع بندی و نتیجه گیری های نهایی

درس یکم: مبانی تئوری و روش های کلاسیک بهینه سازی چند هدفه

در این فرادرس، دانشجویان عزیز با مباحث مختلف مبانی تئوری و روش های کلاسیک بهینه سازی چند هدفه آشنا می شوند. درس هایی که در این آموزش به آن پرداخته می شود، مبانی بهینه سازی چند هدفه، روش های بهینه سازی چند هدفه کلاسیک، روش تبدیل به قید یا ε-Constrainet و... است. نقطه قوت این آموزش این است که به طور کامل به توضیح مباحث مربوطه پرداخته شده است و آموزش توسط یکی از بهترین مدرسین متخصص در این زمینه، انجام شده است.

بهینه سازی چند هدفه، یکی از زمینه های بسیار فعال و پرکاربرد تحقیقاتی در میان مباحث بهینه سازی است. غالبا بهینه سازی چند هدفه (Multi-objective Optimization) به نام های بهینه سازی چند معیاره (Multi-criteria Optimization) و بهینه سازی برداری (Vector Optimization) نیز شناخته می شود. روش های فراوانی تاکنون برای حل این مسائل ارائه شده اند که در حالت کلی می توان آن ها را به دو دسته تقسیم نمود:

- روش های کلاسیک که اغلب مساله چند هدفه را به یک مساله یک هدفه تقلیل می دهند.
- روش های تکاملی که اغلب مساله بهینه سازی چند هدفه را واقعا به صورت چند هدفه حل می نمایند.

موضوع بحث آموزش که در این بخش قصد معرفی آن را داریم، مبانی تئوری بهینه سازی چند هدفه و روش های بهینه سازی چند هدفه کلاسیک است. روش های کلاسیک، در برخی متون به نام روش های تجزیه (Decomposition) نیز شناخته می شوند.

درس دوم: الگوریتم ژنتیک چند هدفه (NSGA-II) در متلب

الگوریتم ژنتیک چند هدفه با مرتب سازی نامغلوب (Non-dominated Sorting Genetic Algorithm) یکی از الگوریتم های شاخص و پرکاربرد در زمینه بهینه سازی چندهدفه است. پس از ارائه نسخه اول این الگوریتم در سال ۱۹۹۵، معرفی کنندگان این الگوریتم که از میان آن ها دب (Deb) معروف تر از سایرین است، نسخه دوم آن را در سال ۲۰۰۲ با نام اختصاری NSGA-

II ارائه نمودند.در کنار تمام کارایی هایی که NSGA-II دارد، می توان آن را الگوی شکل گیری بسیاری از الگوریتم های بهینه سازی چند هدفه دانست. این الگوریتم و شیوه منحصر به فرد آن در برخورد با مسائل بهینه سازی چند هدفه، بارها و بارها توسط افراد مختلف برای ایجاد الگوریتم های بهینه سازی چند هدفه جدیدتر، مورد استفاده قرار گرفته است. بدون شک این الگوریتم یکی از اساسی ترین اعضای کلکسیون الگوریتم بهینه سازی چندهدفه تکاملی است که می توان آن ها را نسل دوم این گونه روش ها نامید.در این جا با مباحث مختلف الگوریتم ژنتیک چند هدفه (NSGA-II) در متلب آشنا می شوند. درس هایی که در این آموزش به آن پرداخته می شود، مبانی بهینه سازی چند هدفه با الگوریتم های تکاملی، تفاوت های الگوریتم NSGA اولیه با نسخه دوم یا NSGA-II، پیاده سازی بخش های مختلف الگوریتم NSGA-II و... است. نقطه قوت این آموزش این است که به طور کامل به توضیح مباحث مربوطه پرداخته شده است و آموزش توسط یکی از بهترین مدرسین متخصص در این زمینه، انجام شده است. موضوع بحث آموزشی که در این پست قصد معرفی آن را داریم، مبانی تئوری الگوریتم NSGA-II و شیوه پیاده سازی آن در محیط متلب است.

فهرست سرفصل ها و رئوس مطالب مطرح شده در این مجموعه آموزشی، در ادامه آمده است:

- مروری بر مبانی بهینه سازی چند هدفه با الگوریتم های تکاملی
- بیان رویکردهای کلی در طراحی الگوریتم های تکاملی برای بهینه سازی چند هدفه
- مرور مختصری بر تئوری الگوریتم ژنتیک
- تعریف مفهوم غلبه در محیط چند هدفه
- اعمال تغییرات لازم در الگوریتم ژنتیک تک هدفه برای استفاده از آن در مسائل چند هدفه
- معرفی اجزای مختلف الگوریتم NSGA-II
  - الگوریتم مرتب سازی نا مغلوب و شیوه رتبه بندی (Ranking)
  - مفهوم فاصله ازدحامی (Crowding Distance)
  - شیوه انتخاب والد به صورت رقابت (تورنمنت) دو دویی (Binary Tournament Selection)
  - مرتب سازی جمعیت به صورت چند مرحله ای
  - انتخاب جمعیت جدید از میان اعضای قدیمی و فرزندان
- تفاوت های الگوریتم NSGA اولیه با نسخه دوم (NSGA-II)
- پیاده سازی دو مثال نمونه از توابع استاندارد چند هدفه
- پیاده سازی بخش های مختلف الگوریتم NSGA-II
- بیان مساله کوله پشتی (Knapsack Problem) به صورت چند هدفه و حل آن به صورت چند هدفه
- جمع بندی و نتیجه گیری های نهایی

درس سوم: الگوریتم PSO چند هدفه (MOPSO) در متلب

الگوریتم بهینه سازی ازدحام ذرات (Swarm Optimizatoion Particle) یا به اختصار PSO یکی از مهم ترین الگوریتم های بهینه سازی هوشمند است که در حوزه هوش ازدحامی (Swarm Intelligence) جای می گیرد. این الگوریتم، توسط جیمز کندی و راسل سی ابرهارت در سال ۱۹۹۵ معرفی گردید، و با الهام از رفتار اجتماعی حیواناتی چون ماهی ها و پرندگان که در گروه هایی کوچک و بزرگ کنار هم زندگی می کنند، طراحی شده است. در الگوریتم PSO، اعضای جمعیت جواب ها، به صورت مستقیم با هم ارتباط دارند و از طریق تبادل اطلاعات با یکدیگر و یادآوری خاطرات خوب گذشته، به حل مساله می پردازند.

با توجه به موفقیت های کم نظیر الگوریتم PSO در حل مسائل بهینه سازی تک هدفه، دانشمندان و محققین بسیاری، سعی در استفاده از این الگوریتم برای حل مسائل چند هدفه داشته اند و تاکنون نسخه های متعددی از الگوریتم PSO برای حل مسائل چند هدفه ارائه شده است. یکی از معروف ترین الگوریتم هایی که در این راستا معرفی شده است، کاری است که توسط پرفسور

کوئلو کوئلو (Coello Coello) و همکارانش معرفی شده است. نامی که آن ها برای الگوریتم شان انتخاب کرده اند MOPSO است، که اغلب این اسم انحصارا برای این الگوریتم به کار برده می شود. این الگوریتم در سال ۲۰۰۴ و در مقاله ای که در مجله محاسبات تکاملی IEEE (با نام کامل IEEE Transactions on Evolutionary Computation) چاپ شده است، معرفی گردید. مبانی تئوری الگوریتم MOPSO و شیوه پیاده سازی آن در محیط متلب است.

فهرست سرفصل ها و رئوس مطالب مطرح شده در این مجموعه آموزشی، در ادامه آمده است:

- مروری بر مبانی بهینه سازی چند هدفه با الگوریتم های تکاملی
- مرور مختصری بر تئوری الگوریتم بهینه سازی ازدحام ذرات (PSO)
- ایجاد تغییرات لازم در الگوریتم PSO برای تبدیل آن به نسخه چند هدفه
- معرفی مفهوم مخزن یا بایگانی پاسخ های نا مغلوب
- معرفی ساختار جدولی مورد استفاده در MOPSO
- مفهوم انتخاب مبتنی بر ناحیه به جای انتخاب مبتنی بر فرد
- شیوه انتخاب الگو (Leader) در الگوریتم MOPSO
- شیوه کنترل اندازه آرشیو
- چگونگی شکل دهی توزیع های احتمالی به نحوی که تنوع پاسخ ها حفظ شود
- انتخاب چرخه رولت (Roulette Wheel Selection) و استفاده از آن در MOPSO
- بیان شباهت های موجود میان الگوریتم PESA-II و MOPSO
- پیاده سازی یک مثال نمونه از توابع استاندارد چند هدفه
- پیاده سازی بخش های مختلف الگوریتم MOPSO در متلب
- جمع بندی و نتیجه گیری های نهایی

درس چهارم: پیاده سازی روش های کلاسیک بهینه سازی چند هدفه در متلب

موضوع بحث آموزشی که در این فرادرس قصد معرفی آن را داریم، پیاده سازی روش های بهینه سازی چند هدفه کلاسیک در متلب است. سه روش از میان روش های کلاسیک بهینه سازی چند هدفه، در این آموزش مورد بررسی قرار گرفته و پیاده سازی شده اند. این سه روش عبارتند از:

- روش مجموع وزن دار
- روش برنامه ریزی آرمانی (Goal Programming)
- روش نیل به آرمان (Goal Attainment)

فهرست سرفصل ها و رئوس مطالب مطرح شده در این مجموعه آموزشی، در ادامه آمده است:

- مبانی بهینه سازی چند هدفه و بیان تفاوت های آن با مساله بهینه سازی یک هدفه
- مروری بر روش های بهینه سازی چند هدفه کلاسیک
  - روش مجموع وزن دار (Weighted Sum)، مزایا و معایب آن
  - روش برنامه ریزی آرمانی (Goal Programming)
  - روش نیل به آرمان (Goal Attainment)
  - روش چبیشف، به عنوان حالت کلی روش های مبتنی بر آرمان
  - روش تبدیل به قید (ε-Constrainet) (بخوانید Epsilon Constraint)
- پیاده سازی سه روش از روش های مطرح شده در متلب
  - پیاده سازی روش مجموع وزن دار با استفاده از تابع fminunc

- پیاده سازی روش برنامه ریزی آرمانی با استفاده از تابع fminunc
- پیاده سازی روش نیل به آرمان با استفاده از تابع fgoalattain

- بهبود عملکرد برنامه های نوشته شده با حذف پاسخ های مغلوب
- جمع بندی و نتیجه گیری های نهایی

درس پنجم: نسخه دوم الگوریتم تکاملی مبتنی بر قوت پارتو (Strength Pareto Evolutionary Algorithm 2) به اختصار SPEA2

نسخه دوم الگوریتم تکاملی مبتنی بر قوت پارتو (Strength Pareto Evolutionary Algorithm 2) به اختصار SPEA2، یکی از معروف ترین و پرکاربردترین الگوریتم های بهینه سازی تکاملی چند هدفه است. این الگوریتم، ایده های بسیار مهمی را مطرح نموده است و یک مثال بسیار مناسب از همه ویژگی هایی است که باید یک الگوریتم بهینه سازی چند هدفه داشته باشد.

در آموزش جامع نسخه دوم الگوریتم تکاملی مبتنی بر قوت پارتو (SPEA2) که بخشی از بسته طلایی آموزش بهینه سازی چند هدفه است، پس از مرور کاملی بر مبانی تئوری الگوریتم SPEA2، پیاده سازی عملی و گام به گام این الگوریتم در محیط متلب، مورد بحث و بررسی واقع شده است. مطالب و مباحث این آموزش به زبان فارسی روان، و توسط سید مصطفی کلامی هریس (فارغ التحصیل دکترای مهندسی برق-کنترل، دانشگاه صنعتی خواجه نصیرالدین طوسی) ارائه شده است.

فهرست سرفصل ها و رئوس مطالب مطرح شده در این مجموعه آموزشی، در ادامه آمده است:

- بررسی مبانی تئوری نسخه یکم الگوریتم SPEA
- بررسی کاستی ها و اشکالات موجود در الگوریتم SPEA
- بیان تفاوت های موجود میان SPEA و SEPA2
- چگونگی تعریف برازندگی ترکیبی در SPEA2 برای انتقال معیار کیفیت پاسخ ها و معیار توزیع
- چگونگی امتیازدهی به پاسخ ها در SPEA2
- چگونگی تعریف معیار توزیع (نظم) با استفاده از الگوی KNN (الگوریتم k نزدیک ترین همسایه) در الگوریتم SPEA2
- پیاده سازی گام به گام و عملی الگوریتم SPEA2 در متلب
- اجرای برنامه برای مثال های عددی از مسائل نمونه بهینه سازی چند هدفه

درس ششم: الگوریتم تکاملی چند هدفه مبتنی بر تجزیه (Multiobjective Evolutionary Algorithm based on Decomposition) به اختصار MOEA/D

الگوریتم تکاملی چند هدفه مبتنی بر تجزیه (Multiobjective Evolutionary Algorithm based on Decomposition) به اختصار MOEA/D، یکی از جدیدترین الگوریتم های بهینه سازی تکاملی چند هدفه است، که دارای ماهیتی کاملا متفاوت با سایر الگوریتم های تکاملی چند هدفه ای است که پیش از آن معرفی شده اند. در ساختار این الگوریتم، مساله بهینه سازی چند هدفه به چندین زیر مساله تک هدفه تجزیه می شود و همه این زیر مسائل، به صورت یک جا حل می شوند. اما حل این زیر مسائل، که توسط اعضای یک جمعیت و به صورت موازی انجام می شود، به صورت تعاملی پیش می رود و با استفاده از مولفه های رقابتی (Competitive) و همکاری (Cooperative) که در ساختار الگوریتم گنجانده شده است، تکامل راه حل های پیشنهادی، به مرور انجام می شود.

در آموزش جامع الگوریتم تکاملی چند هدفه مبتنی بر تجزیه (MOEA/D) که بخشی از بسته طلایی آموزش بهینه سازی چند هدفه است، پس از مرور کاملی بر مبانی تئوری الگوریتم MOEA/D، پیاده سازی عملی و گام به گام این الگوریتم در محیط متلب، مورد بحث و بررسی واقع شده است. مطالب و مباحث این آموزش به زبان فارسی روان، و توسط سید مصطفی کلامی هریس (فارغ التحصیل دکترای مهندسی برق-کنترل، دانشگاه صنعتی خواجه نصیرالدین طوسی) ارائه شده است.

فهرست سرفصل ها و رئوس مطالب مطرح شده در این مجموعه آموزشی، در ادامه آمده است:

- مروری بر مبانی روش های تجزیه (Decomposition) در مسائل بهینه سازی چند هدفه
- روش مبتنی بر وزن دهی
- روش های مبتنی بر رویکرد چبیشف (Tchybecheff)
- مبانی تئوری و شیوه عملکرد الگوریتم MOEA/D
- پیاده سازی گام به گام الگوریتم MOEA/D در محیط متلب
- مباحثی در روش های تعیین وزن های اولیه
- تعریف وزن های دوبعدی منظم (توزیع بر روی کمان دایره)
- حل مسائل نمونه بهینه سازی چند هدفه با استفاده از الگوریتم MOEA/D

درس هفتم: نسخه دوم الگوریتم انتخاب مبتنی بر شکل دهی پارتو (Pareto Envelope-based Selection Algorithm II) به اختصار PESA-II

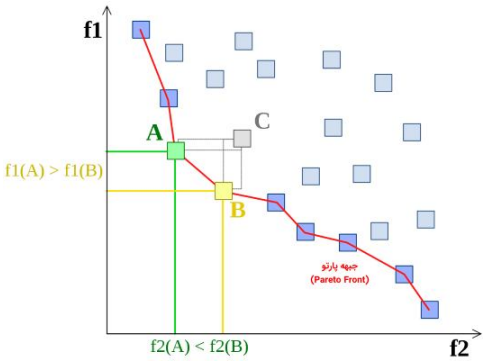
- نسخه دوم الگوریتم انتخاب مبتنی بر شکل دهی پارتو (Pareto Envelope-based Selection Algorithm II) به اختصار PESA-II، یکی از معروف ترین و پرکاربردترین الگوریتم های بهینه سازی تکاملی چند هدفه است. این الگوریتم، ایده های بسیار مهمی را مطرح نموده است و یک مثال بسیار مناسب از همه ویژگی هایی است که باید یک الگوریتم بهینه سازی چند هدفه داشته باشد. از طرفی، ایده اصلی الگوریتم MOPSO نیز از همین الگوریتم گرفته شده است.
- در آموزش جامع نسخه دوم الگوریتم انتخاب مبتنی بر شکل دهی پارتو (PESA-II) که بخشی از بسته طلایی آموزش بهینه سازی چندهدفه است، پس از مرور کاملی بر مبانی تئوری الگوریتم PESA-II، پیاده سازی عملی و گام به گام این الگوریتم در محیط متلب، مورد بحث و بررسی واقع شده است. مطالب و مباحث این آموزش به زبان فارسی روان، و توسط سید مصطفی کلامی هریس (فارغ التحصیل دکترای مهندسی برق-کنترل، دانشگاه صنعتی خواجه نصیرالدین طوسی) ارائه شده است.
- فهرست سرفصل ها و رئوس مطالب مطرح شده در این مجموعه آموزشی، در ادامه آمده است:
- بررسی مبانی تئوری نسخه یکم الگوریتم انتخاب مبتنی بر شکل دهی پارتو (PESA)
- بررسی تفاوت عملکرد نسخه یکم (PESA) و نسخه دوم (PESA-II) از الگوریتم انتخاب مبتنی بر شکل دهی پارتو
- بررسی ساختار احتمالی موجود در PESA و PESA-II و یکسان سازی این دو روش از طریق تعریف فاکتور «فشار انتخاب»
- پیاده سازی گام به گام و عملی الگوریتم PESA-II در متلب
- اعمال تغییرات در برنامه پیاده سازی شده برای بهبود عملکرد آن
- حل یک مساله نمونه بهینه سازی چندهدفه
- هوش مصنوعی
- «بهینه سازی چند هدفه» (Multi-Objective Optimization)، حوزه‌ای از «تصمیم‌گیری چند معیاری» (Multi-Criteria Decision Making) محسوب می‌شود. بهینه سازی چند هدفه با «مسائل بهینه‌سازی ریاضیاتی» (Mathematical Optimization Problems) سروکار دارد که در آن‌ها نیاز است بیش از یک تابع هدف (Objective Function)، به طور همزمان، بهینه‌سازی شوند. بهینه سازی چند هدفه با نام‌های دیگری نظیر «برنامه‌نویسی چند هدفه» (Multi-Objective Programming)، «بهینه‌سازی برداری» (Vector

OPtimization)، «بهینه‌سازی چند معیاری» (Multi-Criteria Optimization)، «بهینه‌سازی چند مشخصه‌ای» (Multi-Attribute Optimization) یا «بهینه‌سازی پارتو» (Pareto Optimization) نیز شناخته می‌شود.

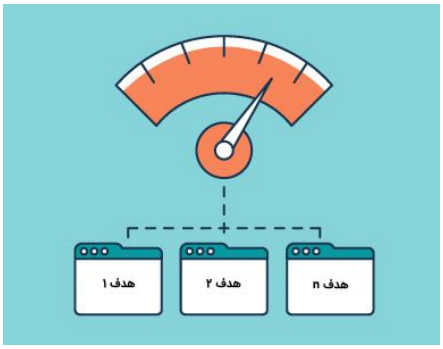
روش‌های بهینه سازی چند هدفه در بسیاری از شاخه‌های علوم و مهندسی به کار گرفته می‌شوند و زمانی مورد استفاده قرار می‌گیرند که برای رسیدن به تصمیمات بهینه در سیستم، نیاز است میان دو یا چند هدف متناقض موازنه (Trade-off) برقرار شود. بدون شک در بسیاری از کاربردهای مهندسی، طراحان فرایند و سیستم‌های مهندسی بر اساس اهداف متناقض، تصمیم‌گیری می‌کنند. به عنوان نمونه، در فرایند طراحی خودرو، علاوه بر اینکه هدف مهندسان طراحی خودرویی است که عملکرد (Performance) حداکثری داشته باشد، به طور همزمان، به دنبال طراحی خودرویی هستند که کمترین میزان آلایندگی و مصرف سوخت را داشته باشد.

در این مورد و موارد مشابه، از آنجایی که بیش از یک تابع هدف باید مورد بررسی قرار بگیرد، نیاز است تا به کارگیری روش‌های بهینه سازی چند هدفه مورد بررسی قرار بگیرد. مهم‌ترین ویژگی در چنین روش‌هایی این است که با به‌کارگیری مدل‌های بهینه سازی چند هدفه (Multi-Objective Optimization)، بیش از یک جواب کاندید (یک جواب ممکن برای مسأله مورد نظر) در اختیار طراحان و مهندسان سیستم قرار گرفته می‌شود؛ هر یک از این جواب‌ها، موازنه میان توابع هدف مختلف را نمایش خواهند داد.

حتی برای یک مسأله بهینه سازی چند هدفه ساده، احتمال اینکه جواب بهینه‌ای (Optimal Solution) یافت شود که به طور همزمان، تمامی توابع هدف تعریف شده در مسأله را بهینه سازی کند، بسیار کم است. در بسیاری از موارد، توابع هدف تعریف شده در مسأله بهینه سازی چند هدفه با یکدیگر در تناقض هستند. در چنین حالتی گفته می‌شود که برای یک مسأله بهینه سازی چند هدفه، «جواب‌های بهینه پارتو» (Pareto Optimal Solutions) وجود خواهد داشت (از لحاظ تئوری، ممکن است بی‌نهایت جواب بهینه پارتو برای یک مسأله بهینه سازی چند هدفه وجود داشته باشد). در بخش‌های بعدی با مفهوم جواب بهینه پارتو آشنا خواهید شد.



مفهومی به نام جواب نامغلوب در سیستم‌های حل مسائل بهینه سازی چند هدفه وجود دارد. در صورتی به یک جواب کاندید برای مسأله بهینه سازی چند هدفه، جواب «نامغلوب» (Non-Dominated) گفته می‌شود که بهبود مقادیر تولید شده توسط یک یا چند تابع هدف از این مسأله (از طریق قرار دادن جواب کاندید در توابع هدف و تولید مقادیر خروجی)، سبب کاهش کیفیت مقادیر تولید شده توسط دیگر توابع هدف همان مسأله شود. به چنین جواب‌هایی، «بهینه پارتو» (Pareto Optimal) گفته می‌شود. بدون در اختیار داشتن اطلاعات اضافی، تمامی جواب‌های بهینه پارتو به یک اندازه خوب هستند و با یکدیگر برابر در نظر گرفته می‌شوند. مفهوم برابری (Equivalency) عدم فرومایگی (Non-Inferiority) برای اولین بار توسط ویلفردو پارتو (Vilfredo Pareto) و فرانسیس وای. اجورث (Francis Y. Edgeworth) و در حوزه اقتصاد معرفی شد. از آن زمان تاکنون، مفهوم بهینه سازی چند هدفه، جای پای خود را در حوزه طراحی و مهندسی مستحکم کرده است. ترجمه تحقیقات ویلفردو پارتو به زبان انگلیسی در سال ۱۹۷۱ منجر به پیاده‌سازی روش بهینه سازی چند هدفه در حوزه مهندسی و «ریاضیات کاربردی» (Applied Mathematics) شد. در طول سه دهه اخیر، به‌کارگیری روش‌های بهینه سازی چند هدفه در بسیاری از حوزه‌های مهندسی و طراحی به رشد ثابت خود ادامه داده است. در این مطلب، با مفاهیم ابتدایی در حوزه بهینه سازی چند هدفه آشنا خواهید شد.



شاید تاکنون، بیشتر مسائل بهینه‌سازی که با آن‌ها برخورد داشته‌اید، مسائلی باشند که تنها از یک تابع هدف برخوردار باشند؛ به مدل‌هایی که برای بهینه‌سازی این دسته از مسائل مورد استفاده قرار می‌گیرند، مدل‌های «بهینه سازی تک هدفه» (Single-Objective) گفته می‌شود. در عمل، در بسیاری از مسائل طراحی و مهندسی، بیش از یک تابع هدف وجود دارد.

در بسیاری از موارد، توابع هدف تعریف شده در یک مسأله بهینه سازی چند هدفه با یکدیگر در تناقض هستند. به عنوان نمونه، یک تیم طراحی مهندسی را در نظر بگیرید. در فرایند طراحی، یکی از اهداف تیم طراحی ممکن است کمینه کردن وزن و به طور همزمان، بیشینه کردن قدرت یک مؤلفه ساختاری خاص در محصول در حال طراحی باشد. علاوه بر این، با در نظر گرفتن فرایند طراحی خودرو، اهداف تیم طراحی ممکن است بیشینه کردن عملکرد خودرو (به عنوان نمونه، افزایش خروجی «گشتاور» (Torque) موتور خودرو جهت شتاب بخشیدن به خودرو در بازه زمانی کوتاه‌تر) و کمینه کردن مصرف سوخت و انتشار آلاینده‌ها باشد؛ دو هدفی که در تناقض با یکدیگر قرار دارند.

دو مثال ذکر شده در بالا، نمونه‌ای از مسائل بهینه سازی چند هدفه هستند که از چند تابع هدف (دو تابع هدف یا بیشتر) تشکیل شده‌اند. از نظر ریاضی، یک مسأله بهینه‌سازی چند هدفه را می‌توان را به شکل زیر فرمول‌بندی کرد:

$$Minimize:f(x)Minimize:f(x)$$
$$Subjectto:gi(x)\leq 0,i=1,mSubjectto:gi(x)\leq 0,i=1,m$$
$$hj(x)=0,j=1,phj(x)=0,j=1,p$$
$$xlk\leq xk\leq xuk,k=1,nxkl\leq xk\leq xku,k=1,n$$

در این رابطه، f(x)f(x) یک بردار متشکل از توابع هدف است که به شکل زیر نمایش داده می‌شود:

$$f(x)=[f1(x),f2(x),.....,fq(x)]Tf(x)=[f1(x),f2(x),.....,fq(x)]T$$

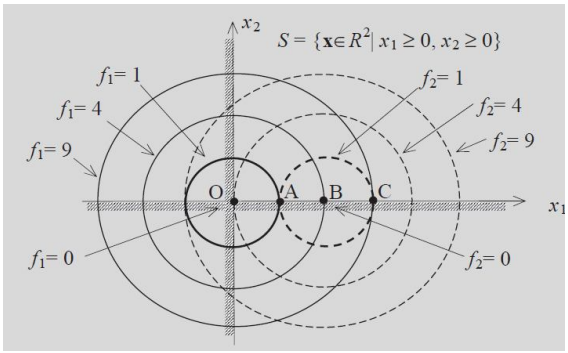
همچنین، qq تعداد توابع هدف در مسأله بهینه سازی چند هدفه را نشان می‌دهد. پیش از اینکه به طور کامل نحوه تولید جواب‌های بهینه پارتو در یک مسأله بهینه سازی چند هدفه شرح داده شود و همچنین، جهت نمایش طبیعت مسائلی که در بهینه سازی چند هدفه با آن‌ها سروکار داریم، ابتدا یک مسأله بسیار ساده در نظر گرفته خواهد شد؛ سادگی این مسأله بهینه سازی چند هدفه از این جهت است که مسأله و جواب‌های آن را می‌توان به صورت بصری نمایش داد. مثال اول: یک مسأله بهینه سازی چند هدفه را در نظر بگیرید که به شکل زیر تعریف می‌شود:

f1(x1,x2)=x21+x22f1(x1,x2)=x12+x22

f2(x1,x2)=(x1−2)2+x22f2(x1,x2)=(x1−2)2+x22

subjectto:x1≥0,x2≥0subjectto:x1≥0,x2≥0

دو تابع هدف f1f1 و f2f2 را می‌توان در «فضای طراحی» (Design Space) مسأله بهینه سازی چند هدفه، به شکل زیر رسم کرد (در مثال اول، فضای طراحی همان صفحه x1−x2x1−x2 است):



خطوط جدا کننده توابع هدف f1f1 و f2f2 (یا همان کانتورهای این توابع) در شکل، دایره‌هایی هستند که به ترتیب در مرکزیت (۰، ۰) و (۰، ۲) قرار دارند. این دو تابع در شکل بالا به ترتیب توسط دایره‌های با خطوط توپر و دایره‌هایی با خطوط خط‌چین نمایش داده شده‌اند. مجموعه جواب‌های امکان‌پذیر (Feasible Set) یا ناحیه جواب‌های امکان‌پذیر این مسأله بهینه سازی چند هدفه، به شکل زیر تعریف می‌شود:

S={x∈R2|x1≥0,x2≥0}S={x∈R2|x1≥0,x2≥0}

مجموعه جواب‌های امکان‌پذیر، «ربع» (Quadrant) اول صفحه x1−x2x1−x2 خواهد بود که در شکل بالا نمایش داده شده است. همان طور که در شکل بالا نیز مشخص است، کمینه (Minimum) تابع f1f1 در نقطه O=(0,0)O=(0,0) و کمینه تابع f2f2 در نقطه B=(2,0)B=(2,0) رخ می‌دهد. با این حال، در نقطه OO مقدار تابع f2f2 برابر با ۴ و در نقطه B، مقدار تابع f1f1 برابر با ۴ است؛ هیچ یک از دو مقدار محاسبه شده، نقطه کمینه توابع f1f1 و f2f2 نیستند. کاملا مشخص است که هیچ نقطه خاصی وجود ندارد که سبب شود توابع f1f1 و f2f2 در مقدار کمینه خود قرار بگیرند.

بهترین جوابی که تا حدودی می‌تواند دو تابع را در نقطه‌ای نزدیک به کمینه قرار دهد، در نقطه A=(1,0)A=(1,0) حاصل می‌شود؛ در این نقطه، مقدار توابع f1f1 و f2f2 برابر با f1=f2=1f1=f2=1 است. اگرچه فاصله گرفتن از این نقطه ممکن است سبب کاهش کیفیت جواب یکی از توابع هدف شود ولی در نقطه مقابل ممکن است سبب افزایش کیفیت جواب تولید شده توسط تابع هدف دیگر شود (ممکن است منجر به تولید مقدار کمینه‌تری برای تابع هدف دوم شود).

شایان توجه است که برای یک مسأله بهینه سازی چند هدفه نظیر مثال اول، هر جوابی که میان نقاط OO و BB قرار بگیرد ممکن است به عنوان یک جواب قابل قبول شناخته شود. زیرا افزایش کیفیت جواب در یکی از توابع هدف مسأله (کمینه کردن تابع اول)، بدون کاهش کیفیت جواب تولید شده توسط تابع هدف دیگر، امکان‌پذیر نخواهد بود.

البته با تغییر دادن نقاطی که خارج از بخش‌بندی خطی میان نقاط OO و BB قرار دارند (تمامی نقاطی که در محدوده x1∉(0,2)x1∉(0,2) و x2=0x2=0 قرار دارند)، می‌توان مقادیر دو تابع هدف f1f1 و f2f2 را به طور همزمان کاهش داد.

به عنوان نمونه، نقطه C=(3,0)C=(3,0) را در نظر بگیرید. در این نقطه، مقدار توابع f1f1 و f2f2 به ترتیب برابر با f1=9f1=9 و f2=1f2=1 هستند. با حرکت کردن از نقطه CC به سمت نقطه BB ، مقدار کمینه‌تری برای دو تابع f1f1 و f2f2 حاصل خواهد شد (افزایش کیفیت جواب‌ها).

شایان توجه است که به نقاط میانOO و BB ، که توسط رابطه زیر تعریف می‌شوند، مجموعه بهینه پارتو (Pareto Optimal Set) برای مسأله بهینه سازی چند هدفه گفته می‌شود (مجموعه SpSp). یک مجموعه بهینه پارتو معمولا حاوی بی‌نهایت جوابی است که به عنوان جواب‌های معتبر برای یک مسأله بهینه سازی چند هدفه شناخته می‌شوند.

Sp={x∈R2|0≤x1≤2,x2=0}Sp={x∈R2|0≤x1≤2,x2=0}

توجه داشته باشید که شاید برای یک مسأله بهینه سازی چند هدفه ساده (مانند مثال اول)، پیدا کردن جواب‌های بهینه پارتو کار ساده‌ای باشد، ولی متاسفانه، برای دیگر مسائل بهینه سازی چند هدفه، چنین امری ممکن است ساده نباشد. به طور کلی، صحبت کردن درباره جواب‌های بهینه پارتو، تنها در فضای طراحی (Design Space) مسأله بهینه سازی چند هدفه کافی نیست. برای درک بهتر مفهوم مسائل بهینه سازی چند هدفه (و تکنیک‌های آن)، لازم است تا مسأله به «فضای معیار» (Criterion Space | فضای هدف) نگاشت شود. در بخش بعدی، با مفاهیم پایه بهینه سازی چند هدفه و فضای معیار بیشتر آشنا خواهید شد.

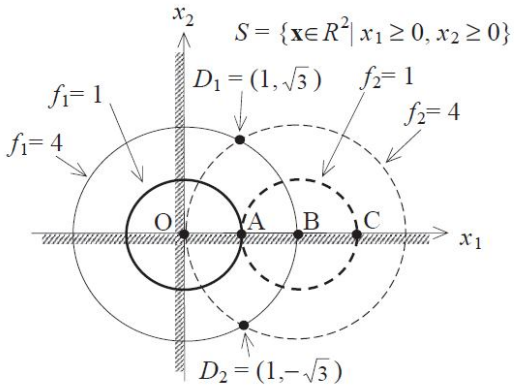
مفاهیم پایه در بهینه سازی چند هدفه

مفهوم اسکالر (عددی) بودن «بهینگی» (Optimality) مسائل بهینه‌سازی «تک‌هدفه» (Single Objective)، به طور مستقیم در بهینه سازی چند هدفه کاربرد ندارد. برای این که بتوان با مفهوم بهینگی در مسائل بهینه سازی چند هدفه آشنا شد، ابتدا باید مفهوم «بهینگی پارتو» (Pareto Optimality) را شرح داد. در این بخش، ابتدا با استفاده از مثال اول، مفاهیمی نظیر فضای طراحی و فضای معیار (یا فضای هدف) شرح داده خواهند شد.

فضای معیار و فضای طراحی در مسائل بهینه سازی چند هدفه

در مثال اول، فضای طراحی یک مسأله بهینه سازی چند هدفه نمایش داده شد. سپس، مجموعه جواب‌های امکان‌پذیر (مجموعه SS) تعریف، کانتورهای توابع هدف (خطوط جدا کننده توابع هدف) رسم و در نهایت، مجموعه بهینه پارتو (مجموعه SpSp) این مسأله بهینه سازی چند هدفه در فضای طراحی شناسایی شد.

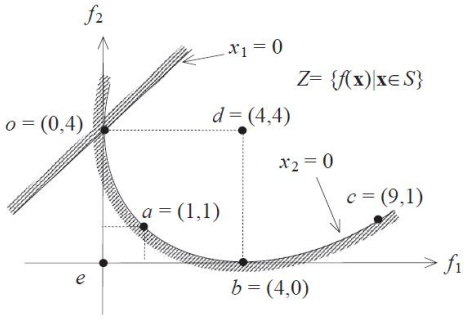
یک روش ممکن دیگر برای حل مسائل بهینه سازی چند هدفه، استفاده از فضای معیار (Criterion Space) است. متناوبا، یک مسأله بهینه سازی چند هدفه را می‌توان در فضایی به نام فضای معیار نمایش داد؛ در این فضا، محورها به وسیله توابع هدف نمایش داده خواهند شد. به عنوان نمونه، شکل زیر نمایش مسأله بهینه سازی چند هدفه مثال اول را در فضای طراحی نمایش می‌دهد.



,

فضای طراحی (Design Space) در مسأله بهینه سازی چند هدفه

شکل زیر، مسأله بهینه سازی چند هدفه مثال اول را این بار در فضای معیار (یا همان صفحه f1−f2f1−f2) نمایش می‌دهد. همانطور که در شکل زیر مشخص است، محورهای این فضا توسط توابع هدف نشان داده شده‌اند.



فضای معیار یا هدف (Criterion Space | Objective Space) در مسأله بهینه سازی چند هدفه

شایان توجه است که قیدهای x1≥0x1≥0 و x2≥0x2≥0 در این مسأله بهینه‌سازی، از طریق حل کردن آن‌ها بر حسب f1f1 و f2f2 ، به فضای معیار نگاشت خواهند شد.

x1(f1,f2)=0.25×(f1−f2)+1≥0x1(f1,f2)=0.25×(f1−f2)+1≥0

x2(f1,f2)=√f1−x21=√f1−(0.25×(f1−f2)+1)2≥0x2(f1,f2)=f1−x12=f1−(0.25×(f1−f2)+1)2≥0

فضای معیار امکان‌پذیر ZZ ، در قالب مجموعه مقادیری از تابع هدف، که متناظر با نقاط امکان‌پذیر (Feasible Points) در فضای طراحی هستند، تعریف می‌شود:

Z={f(x)|x∈S}Z={f(x)|x∈S}

همانطور که در شکل‌های بالا نشان داده شده است، نقاط AA ، BB و OO در فضای طراحی به نقاط aa ، bb و oo در فضای معیار نگاشت شده‌اند. مجموعه بهینه پارتو (Pareto Optimum) در فضای طراحی مسأله بهینه سازی چند هدفه، که به شکل Sp={x∈R2|0≤x1≤2,x2=0}Sp={x∈R2|0≤x1≤2,x2=0} تعریف می‌شود، به منحنی oab در فضای معیار نگاشت و به شکل زیر نمایش داده می‌شود:

Zp={f=(f1,f2)∈R2|0≤f1≤4,0≤f2≤4,√f1−(0.25×(f1−f2)+1)2=0}Zp={f=(f1,f2)∈R2|0≤f1≤4,0≤f2≤4,f1−(0.25×(f1−f2)+1)2=0}

به منحنی oab در فضای معیار، «جواب پارتو» (Pareto Solution)، بهینه پارتو، «جبهه پارتو» (Pareto Front) و یا «مجموعه پارتو» (Pareto Set) گفته می‌شود؛ منحنی oab نمایش دهنده جواب یک مسأله بهینه سازی چند هدفه محسوب می‌شود.

همانطور که در شکل نمایش‌دهنده فضای معیار مثال اول مشخص است، کمینه تابع هدف f1f1 برابر با ۰ است که در نقطه o=(0,4)o=(0,4) واقع شده است. در سوی مقابل، کمینه تابع هدف f2f2 برابر با صفر است که در نقطه b=(4,0)b=(4,0) واقع شده است. علاوه بر این، در صورتی که نیاز باشد تا کمینه‌سازی بیشتری روی مقادیر تولید شده توسط یکی از توابع انجام شد، بدون شک مقدار تابع دیگر افزایش می‌یابد.

همچنین، هیچ یک از نقاط دیگری که روی جبهه پارتو قرار دارند نمی‌توانند به طور همزمان، کمینه‌سازی بیشتری رو مقادیر توابع هدف f1f1 و f2f2 انجام دهند. با این حال، برای نقطه دیگری، نظیر d=(4,4)d=(4,4) ، که روی جبهه پارتو قرار ندارد، این امکان وجود دارد که با حرکت دادن این نقطه به سمت جبهه پارتو، مقدار دو تابع هدف f1f1 و f2f2 را به طور همزمان کاهش داد.

نکته شایان توجه در مورد مسائل بهینه سازی چند هدفه و تولید فضای طراحی و فضای معیار متناسب با مسأله این است که هر نقطه در فضای طراحی را می‌توان به یک نقطه در فضای معیار نگاشت کرد. با این حال، عکس چنین حالتی همیشه صحیح و امکان‌پذیر نیست؛ به عبارت دیگر، هر نقطه در فضای معیار لزوما متناظر با یک نقطه (یا چند نقطه) در فضای معیار نخواهد بود. به عنوان نمونه، نقطه e=(0,0)e=(0,0) در فضای معیار به هیچ نقطه‌ای در فضای طراحی نگاشت نخواهد شد. علاوه بر این، نقطه d=(4,4)d=(4,4) در فضای معیار به دو نقطه D1=(1,3)D1=(1,√3) و D2=(1,−3)D2=(1,−√3) در فضای طراحی نگاشت می‌شود (D2∉SD2∉S و D1∈SD1∈S).

بهینگی پارتو (Pareto Optimality)

به مفهوم تعریف جواب‌های یک مسأله بهینه سازی چند هدفه، بهینگی پارتو (Pareto Optimality) گفته می‌شود. در صورتی به نقطه x⋆x⋆ در فضای طراحی مسأله (SS) بهینه پارتو گفته می‌شود که در مجموعه SS نقطه دیگری وجود نداشته باشد که باعث کمینه‌سازی حداقل یکی از توابع هدف موجود در مسأله بهینه سازی چند هدفه و به طور همزمان، افزایش مقدار یک تابع هدف دیگر شود. در مثال اول این مطلب، بهینه پارتو برابر با رابطه زیر خواهد بود:

x⋆∈Sp={x∈R2|0≤x1≤2,x2=0}x⋆∈Sp={x∈R2|0≤x1≤2,x2=0}

در ادامه، تعریف دقیق‌تری از بهینه پارتو ارائه خواهد شد.

تعریف اول: بهینه پارتو (Pareto Optimality)

در صورتی به نقطه x⋆x⋆ در فضای طراحی مسأله (SS) بهینه پارتو گفته می‌شود که نقطه دیگری مانند x∈Sx∈S وجود نداشته باشد، به طوری که:

fi(x)≤fi(x⋆)∀ifi(x)≤fi(x⋆)∀i

fi(x)≤fi(x⋆)foratleastoneifi(x)≤fi(x⋆)foratleastonei

برای درک بهتر عبارات بالا، فرض کنید که نقطه x⋆=(3,0)x⋆=(3,0) یک بهینه پارتو باشد (این نقطه، همان نقطه CC در فضای طراحی (Design Space) مسأله بهینه سازی چند هدفه مثال اول است). در این نقطه، مقدار تابع f1(x⋆)=9f1(x⋆)=9 و f2(x⋆)=1f2(x⋆)=1 است. حالا، درست بودن شرط زیر را برای یک نقطه دیگر بررسی می‌کنیم؛ در صورتی که شرط زیر برای نقاط دیگر برقرار باشد، نقطه x⋆=(3,0)x⋆=(3,0) را نمی‌توان به عنوان یک نقطه بهینه پارتو در نظر گرفت:

fi(x)≤fi(x⋆)∀iandfi(x)<fi(x⋆)foratleastoneifi(x)≤fi(x⋆)∀iandfi(x)<fi(x⋆)foratleastonei

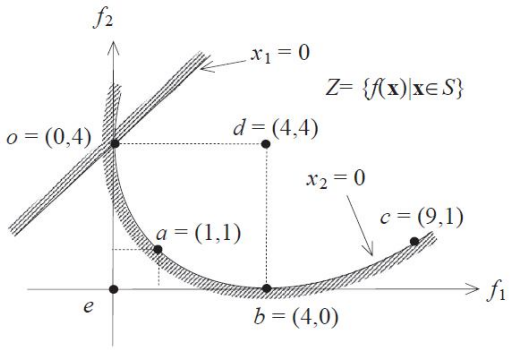
برای چنین کاری، نقطه x⋆=(1,0)x⋆=(1,0) را انتخاب می‌کنیم (این نقطه، همان نقطه AA در فضای طراحی (Design Space) مسأله بهینه سازی چند هدفه مثال اول است). در این نقطه، f1(x⋆)=f2(x⋆)=1f1(x⋆)=f2(x⋆)=1 است. بنابراین، در نقطه x⋆=(1,0)x⋆=(1,0) ، شرط بالا به شکل زیر برقرار است.

fi(x)≤fi(x⋆)∀iandf1(x)=1<f1(x⋆)=9foratleastoneifi(x)≤fi(x⋆)∀iandf1(x)=1<f1(x⋆)=9foratleastonei

با توجه به شرطی که در ابتدای تعریف نقاط بهینه پارتو وضع شد، نقطه x⋆=(3,0)x⋆=(3,0) یک بهینه پارتو نخواهد بود. در نقطه مقابل، در صورتی که نقطه x⋆=(1,0)x⋆=(1,0) را بعنوان یک نقطه بهینه پارتو فرض کنیم، مقادیر توابع f1f1 و f2f2 به ترتیب برابر با f1(x⋆)=f2(x⋆)=1f1(x⋆)=f2(x⋆)=1 خواهد بود. همچنین، هیچ نقطه دیگری وجود ندارد که در آن، مقدار توابع f1f1 و f2f2 کمتر از مقدار آن‌ها در نقطه (1,0)(1,0) باشد. در نتیجه، نقطه x⋆=(1,0)x⋆=(1,0) یک نقطه بهینه پارتو خواهد بود. در واقع، تمامی نقاط موجود در مجموعه SpSp یا همان مجموعه بهینه پارتو در فضای طراحی، شرط بالا را رعایت می‌کنند. در نتیجه، تمامی اعضای این مجموعه بهینه پارتو هستند.

شایان توجه است که تمامی اعضای مجموعه بهینه پارتو ZpZp ، همیشه در مرز (Boundary) فضای معیار ZZ قرار خواهند داشت. با فرض اینکه نقاط کمینه منحصر به فرد باشند و تنها دو تابع هدف در مسأله بهینه سازی چند هدفه داشته باشند

(همانند مثال اول)، مقدار کمینه دو تابع هدف مسأله، نقاط انتهایی جبهه پارتو را تشکیل خواهند داد (نقطه oo و bb در شکل زیر).



یک مفهوم بسیار مرتبط دیگر به بهینگی پارتو، «بهینگی پارتوی ضعیف» (Weak Pareto Optimality) است. در نقاط بهینه پارتوی ضعیف، این امکان وجود دارد که با بهینه‌سازی برخی از توابع هدف مسأله، کیفیت جواب‌های تولید شده توسط دیگر توابع هدف کاهش پیدا نکند.

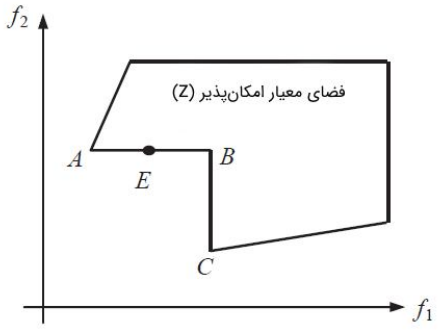
در ادامه، تعریف دقیق‌تری از بهینه پارتوی ضعیف ارائه خواهد شد.

تعریف دوم: بهینه پارتوی ضعیف (Weak Pareto Optimality)

در صورتی به نقطه x⋆x⋆ در فضای طراحی مسأله (SS) بهینه پارتوی ضعیف گفته می‌شود که نقطه دیگری مانند x∈Sx∈S وجود نداشته باشد، به طوری که:

fi(x)<fi(x⋆)∀ifi(x)<fi(x⋆)∀i

به عبارت دیگر، بیک نقطه در فضای طراحی، بهینه پارتوی ضعیف محسوب می‌شود اگر، نقطه دیگری وجود نداشته باشد که بتواند به طور همزمان، تمامی توابع هدف موجود در مسأله را بهینه کند (یا به طور همزمان، کیفیت جواب‌های تولید شده توسط تمامی توابع هدف را افزایش دهد). با این حال، نقاطی نیز ممکن است وجود داشته باشند که کیفیت برخی از توابع هدف موجود در مسأله را افزایش می‌دهند ولی مقادیر تولید شده توسط دیگر توابع را بدون تغییر نگه می‌دارند. مفهوم نقاط بهینه پارتوی ضعیف در شکل زیر نمایش داده شده است.



در این مثال نیز، هدف کمینه‌سازی دو تابع هدف f1f1 و f2f2 است. شایان توجه است که خطوط ABAB و BCBC ، مرز فضای معیار امکان‌پذیر ZZ را تشکیل می‌دهند و به ترتیب، توسط خط‌های افقی و عمودی نمایش داده شده‌اند. در این مثال، تمامی نقاطی که روی خط A−B−CA−B−C قرار دارند، نقاط بهینه پارتوی ضعیف هستند. با این حال، تنها نقاط A و C به عنوان نقاط بهینه پارتو شناخته می‌شوند.

در صورتی که هر نقطه‌ای که رو خط ABAB قرار داشته باشد (به غیر از نقطه AA) را در نظر بگیرید، نظیر نقطه EE ، هیچ نقطه دیگری نظیر XX در فضای معیار امکان‌پذیر وجود نخواهد داشت که در آن:

f1(x)<f1(xE)andf2(x)<f2(xE)f1(x)<f1(xE)andf2(x)<f2(xE)

بنابراین، نقطه EE یک بهینه پارتوی ضعیف محسوب می‌شود. در سوی مقابل، از آنجایی که می‌توان حداقل یک نقطه نظیر xx (به عنوان نمونه، تمامی نقاط روی خط AEAE) پیدا کرد که در آن:

f1(x)<f1(xE)andf2(x)=f2(xE)f1(x)<f1(xE)andf2(x)=f2(xE)

بنابراین، نقطه EE را نمی‌توان یک بهینه پارتو به حساب آورد. به طور مشابه، تمامی نقاطی که رو خط BCBC قرار داشته باشد (به غیر از نقطه CC)، بهینه پارتوی ضعیف محسوب می‌شوند ولی بهینه پارتو نیستند. نکته مهمی که در این بخش باید به آن اشاره شود این است که یک نقطه بهینه پارتو، یک نقطه بهینه پارتوی ضعیف محسوب می‌شود ولی عکس آن همیشه صادق نیست؛ به عبارت دیگر، یک نقطه بهینه پارتوی ضعیف، لزوما یک نقطه بهینه پارتو نخواهد بود.

یکی دیگر از مفاهیم موجود در بهینه سازی چند هدفه، نقاط مغلوب (Dominated) نامغلوب (Non-Dominated) هستند. در ادامه، تعریف دقیق‌تری از این نقاط ارائه خواهد شد.

تعریف سوم: نقاط مغلوب (Dominated) نامغلوب (Non-Dominated)

بردار متشکل از توابع هدف f⋆=f(x⋆)∈Zf⋆=f(x⋆)∈Z نامغلوب شناخته می‌شود، اگر و تنها اگر، بردار دیگری نظیر f∈Zf∈Z وجود نداشته باشد، به طوری که:

fi≤f⋆i∀iandfi<f⋆iforatleastoneifi≤fi⋆∀iandfi<fi⋆foratleastonei

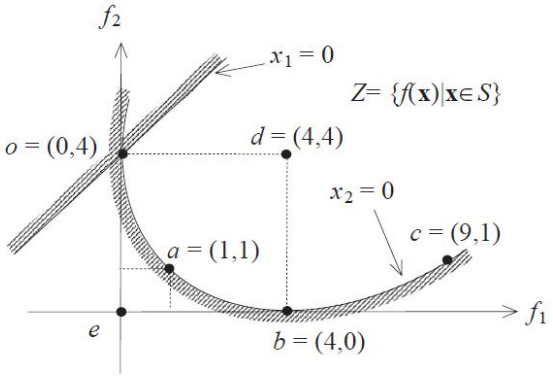
در غیر این صورت، بردار f⋆f⋆ مغلوب شناخته می‌شود. به طور کلی، بهینگی پارتو به هر دو فضای طراحی (Design) معیار اشاره دارد. علاوه بر این مفاهیم، مفهوم دیگری به نام «نقطه منحصر به فرد» (Unique Point) یا «نقطه ایده‌آل» (Utopia Point | Ideal Point) در بهینه سازی چند هدفه وجود دارد. در ادامه، تعریف دقیق‌تری از نقاط ایده‌آل ارائه خواهد شد.

تعریف چهارم: نقطه ایده‌آل یا منحصر به فرد (Unique Point)

در صورتی به یک نقطه نظیر fofo در فضای معیار، نقطه ایده‌آل یا منحصر به فرد (Unique Point) گفته می‌شود که:

foi=min{fi(x)|x∈S},i=1toqfio=min{fi(x)|x∈S},i=1toq

در این رابطه، پارامتر qq تعداد توابع هدف موجود در مسأله بهینه سازی چند هدفه را نشان می‌دهد. نقاط ایده‌آل، از طریق کمینه‌سازی هر یک از توابع موجود در مسأله بهینه سازی چند هدفه، بدون در نظر گرفتن دیگر توابع حاصل می‌شوند. هر بار که یکی از توابع هدف موجود در مسأله بهینه سازی چند هدفه کمینه می‌شود، یک نقطه در فضای طراحی و یک مقدار متناظر با آن تابع هدف به دست می‌آید. به عنوان نمونه، در شکل زیر نقطه ee نقطه ایده‌آل مثال اول محسوب می‌شود.



به طور کلی، احتمال اینکه مقادیر بهینه تمامی توابع هدف موجود در مسأله بهینه سازی چند هدفه، در یک نقطه یکسان در فضای طراحی قرار بگیرد بسیار پایین و کمیاب است. به عبارت دیگر، یک نقطه در فضای طراحی نمی‌تواند تمامی توابع هدف موجود در یک مسأله بهینه‌سازی را کمینه کند. بنابراین، نقطه ایده‌آل تنها در فضای معیار وجود خواهد داشت.

علاوه بر این، پیدا کردن نقاط ایده‌آل، نظیر ee در فضای معیار مثال اول (شکل بالا)، کار بسیار سختی است. به همین خاطر است که می‌گویند نقاط ایده‌آل معمولا دست نیافتنی (Unattainable) هستند. در چنین حالتی، بهترین گزینه پیدا کردن جواب‌هایی (جواب‌های بهینه پارتو) است که بیشترین نزدیکی ممکن را به جواب ایده‌آل در فضای معیار داشته باشند. به چنین جوابی، جواب متوازن (Compromise Solution) گفته می‌شود.

به عنوان نمونه، نقطه aa در شکل بالا، نقطه‌ای در فضای معیار است که منجر به تولید جواب متوازن برای مسأله بهینه سازی چند هدفه مثال اول خواهد شد. میزان نزدیکی میان نقطه ایده‌آل و نقاط دیگر در فضای معیار (نقاطی که منجر به تولید جواب متوازن می‌شوند) را می‌توان توسط روش‌های متفاوتی تعریف کرد. معمولا نقطه‌ای در فضای معیار یک مسأله بهینه سازی چند هدفه به عنوان جواب متوازن شناخته می‌شود که فاصله اقلیدسی (Euclidean Distance) آن از نقطه ایده آل کمینه باشد. فاصله اقلیدسی یک نقطه در فضای معیار، از نقطه ایده‌آل از طریق رابطه زیر محاسبه می‌شود:

$$D(X)=\|f(x)-fo\|=\sqrt{\sum_{i=1}^{q}[fi(x)-foi]^2}D(X)=\|f(x)-fo\|=\sum_{i=1}^{q}[fi(x)-fio]2$$

در این رابطه، foifio یک مؤلفه از نقطه ایده‌ال در فضای معیار را نمایش می‌دهد. جواب‌های متوازن نیز به عنوان جواب‌های بهینه پارتو شناخته می‌شوند.

تولید مجموعه بهینه پارتو

یک مسأله بهینه سازی چند هدفه ساده نظیر مثال اول را به راحتی می‌توان از طریق نگاشت مسأله از فضای طراحی (Design Space) به فضای معیار (Criterion Space) و شناسایی مجموعه جواب‌های بهینه پارتو در فضای معیار، به صورت زیر حل کرد:

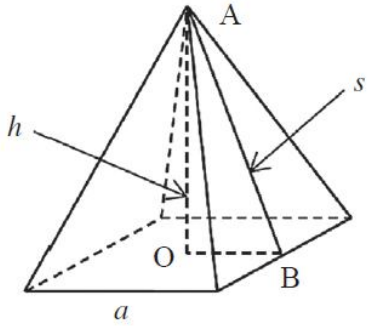
$$Zp=\{f=(f1,f2)\in R2|0\le f1\le 4,0\le f2\le 4,\sqrt{f1}-(0.25\times(f1-f2)+1)2=0\}Zp=\{f=(f1,f2)\in R2|0\le f1\le 4,0\le f2\le 4,f1-(0.25\times(f1-f2)+1)2=0\}$$

همچنین، مجموعه بهینه پارتو (در مثال اول) را می‌توان به راحتی به فضای طراحی اصلی مسأله نگاشت دوباره کرد:

$$Sp=\{x\in R2|0\le x1\le 2,x2=0\}Sp=\{x\in R2|0\le x1\le 2,x2=0\}$$

با این حال، برای مسائل بهینه سازی چند هدفه که پیچیده‌تر از مثال اول هستند، چنین کاری امکان‌پذیر نخواهد بود. بنابراین سؤالی که ممکن است در این بخش پدید آید این است که چگونه می‌توان چنین مسائلی را حل کرد؟ یکی از راه‌های موجود برای حل چنین مسائلی استفاده از رویکردهای Brute-Force است.

مثال دوم: به شکل زیر دقت کنید:



در این بخش با یک مسأله بهینه سازی چند هدفه به نام «هرم» (Pyramid) آشنا خواهید شد. همانطور که در شکل بالا مشخص است، عرض قاعده (Base Width) و ارتفاع (Height) هرم، به ترتیب، توسط aa و hh نمایش داده شده است. طول وتر (Chord Length) مثلث OABOAB نیز برابر با ss است. توابع هدف مسأله طراحی هرم نیز برابر با کمینه کردن مساحت سطح جانبی AA و کمینه کردن مساحت کل هرم TT از طریق دو متغیر عرض قاعده (aa) و ارتفاع (hh) هستند. همچنین حجم هرم باید بیشتر از ۱۵۰۰ اینچ مکعب باشد و کران بالای اندازه عرض قاعده (aa) و ارتفاع (hh) برابر با ۳۰ اینچ است. از نظر ریاضی، مسأله بهینه سازی چند هدفه هرم به شکل زیر تعریف می‌شود:

Minimize:A(a,h)=2as=2aV(a2)2+h2Minimize:A(a,h)=2as=2a(a2)2+h2

T(a,h)=A+a2=2aV(a2)2+h2+a2T(a,h)=A+a2=2a(a2)2+h2+a2

Subjectto:V(a,h)=a2h3≥1500Subjectto:V(a,h)=a2h3≥1500

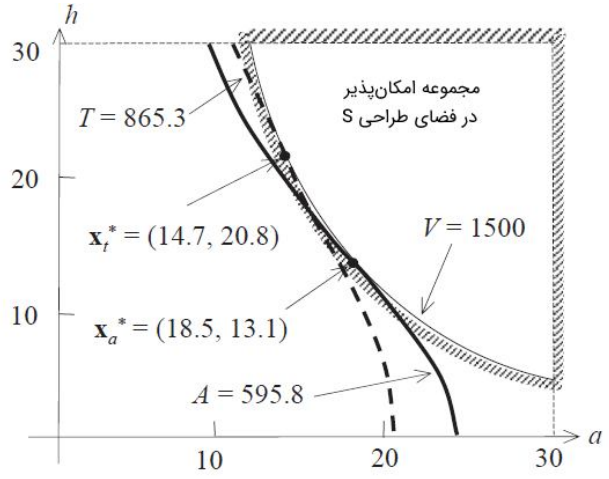
0<a≤300<a≤30

0<h≤300<h≤30

مجموعه امکان‌پذیر برای این مسأله بهینه سازی، در فضای طراحی آن (در چنین حالتی، صفحه a−ha−h) و به شکل زیر تعریف می‌شود:

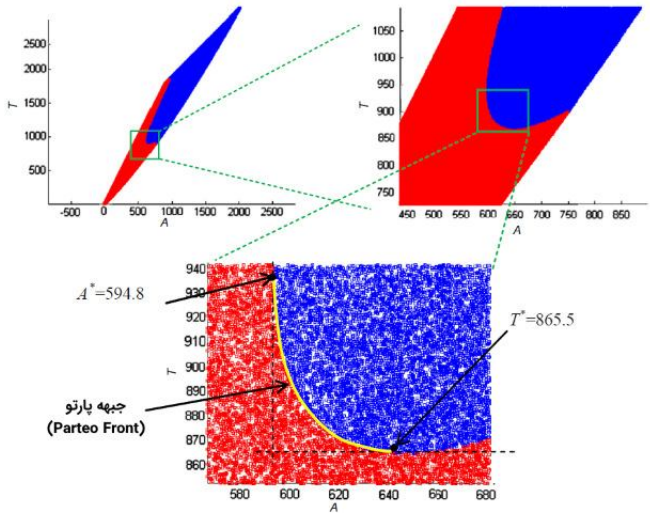
S={(a,h)∈R2|V(a,b)=a2h3≥1500and0<a≤30,0<h≤30}S={(a,h)∈R2|V(a,b)=a2h3≥1500and0<a≤30,0<h≤30}

مجموعه امکان‌پذیر در فضای طراحی مسأله بهینه سازی چند هدفه در شکل زیر نمایش داده شده است. همچنین، مقادیر x⋆t=(14.7,20.8)xt⋆=(14.7,20.8) و x⋆a=(18.5,13.1)xa⋆=(18.5,13.1) به ترتیب، نقاط بهینه برای توابع هدف TT و AA هستند. شایان توجه است که نقاط بهینه x⋆txt⋆ و x⋆axa⋆ از طریق حل کردن جداگانه توابع هدف متناظر با آن‌ها در زبان برنامه‌نویسی متلب (MATLAB Programming Language) به دست آمده‌اند.

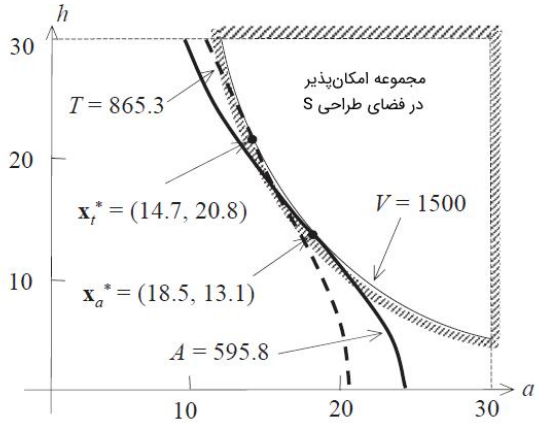


نگاشت این مسأله بهینه سازی چند هدفه از فضای طراحی به فضای معیار کار بسیار سختی است. همچنین، دوباره‌نویسی متغیرهای aa و hh در قالب توابعی بر حسب AA و TT فرایند بسیار پیچیده و مشکلی است. بنابراین، در این مطلب از روش‌های مولد (Generative Method) برای تولید و نمایش جواب‌های امکان‌پذیر و غیرامکان‌پذیر در فضای معیار استفاده می‌شود.برای حل مسأله مورد نظر، ابتدا به ازاء aa و hh یک میلیون نقطه تصادفی تولید می‌شود. سپس، مقدار عددی دو تابع هدف AA و TT برای هر نقطه (aa , hh) محاسبه می‌شود. سپس، حجم هرم VV محاسبه خواهد شد. در صورتی که مقدار حجم محاسبه شده برای یک نقطه بزرگتر یا مساوی ۱۵۰۰ باشد، نقطه‌ای که به وسیله (aa , hh) نمایش داده می‌شود، یک نقطه امکان‌پذیر محسوب خواهد شد؛ در غیر اینصورت، غیرامکان‌پذیر خواهد بود.در مرحله بعد، تمامی نقاط امکان‌پذیر و غیرامکان‌پذیر به طور جداگانه ذخیره و در فضای معیار (صفحه A−TA−T) با دو رنگ متفاوت نمایش داده

می‌شوند. نقاطی که در محل تقاطع مناطق با رنگ‌های مختلف قرار دارند، مرز فضای معیار امکان‌پذیر ZZ هستند. به محض اینکه فضای معیار امکان‌پذیر ZZ شناسایی شد، موقعیت نقاط کمینه توابع هدف AA و TT مشخص و جبهه پارتو (Pareto Front) شناسایی می‌شود. تمامی فرایندهای شرح داده شده در شکل زیر نمایش داده شده است.



در شکل مشخص شده است بسیار نزدیک است.

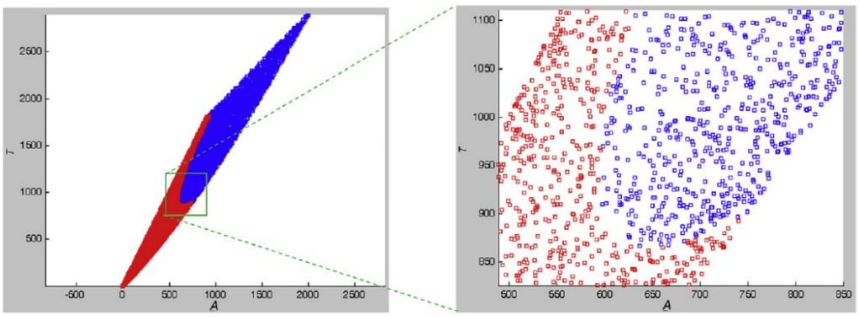


برای صحت‌سنجی نتایج تولید شده، این رویه با تعداد متفاوت نقاط تصادفی (۱۰۰ تا یک میلیون) به ازاء aa و hh آزمایش شده است. نتایج حاصل از آزمایشات انجام شده در جدول زیر نمایش داده شده است.

آزمایشات انجام شده برای تعداد متفاوت نقاط تصادفی (100 تا یک میلیون) به ازاء a و h

نقاط نمونه	مساحت سطح A			مساحت کل T		
	کمینه A	a	h	کمینه T	a	h
10,000,000	594.82	18.486	13.168	865.35	14.675	20.895
1,000,000	594.85	18.627	12.970	865.54	14.559	21.232
100,000	595.68	18.924	12.577	866.26	14.999	20.021
10,000	595.57	18.368	13.360	866.23	14.724	20.787
1000	612.16	19.199	12.728	876.89	15.021	20.336
100	643.25	20.499	11.878	901.44	15.416	20.100

نتایج نشان می‌دهد که وقتی تعداد نقاط نمونه از ۱۰ هزار نمونه بیشتر می‌ود، نتایج تغییر قابل ملاحظه و معناداری نمی‌کنند. با این حال، اگر فضای معیاری که توسط ۱۰ هزار نمونه تولید شده است رسم شود، مرز مجموعه معیار امکان‌پذیر یا همان جبهه پارتو (Pareto Front)، دیگر به خوبی قابل شناسایی نخواهد بود (شکل زیر).



اگرچه روش‌های مولد روش‌های بسیار ساده‌ای برای حل مسائل بهینه سازی چند هدفه هستند، اما عملکرد آن‌ها در مسائل سخت و پیچیده (و حتی مسائل با پیچیدگی متوسط) بسیار ضعیف است. به طور کلی، مسائل مهندسی و طراحی، مسائل بسیار پیچیده‌ای هستند که حل کردن و ارزیابی توابع آن‌ها، بار محاسباتی بسیار قابل توجهی را می‌طلبد. بنابراین، استفاده از روش‌های مولد برای حل مسائل بهینه سازی چند هدفه، از لحاظ محاسباتی، امکان‌پذیر نیست.

روش K نزدیک‌ترین همسایه نزدیک K Nearest Neighbors یا (KNN) یک روش یادگیری موردی است و از جمله ساده‌ترین الگوریتم‌های ***یادگیری ماشین*** می‌باشد که به روش K همسایه نزدیک نیز معروف است. در این الگوریتم یک نمونه با رای اکثریت از همسایه‌هایش دسته‌بندی می‌شود و این نمونه در عمومی‌ترین کلاس مابین k همسایه نزدیک تعیین می‌شود. K یک مقدار مثبت صحیح و عموماً کوچک است. اگر k=1 باشد نمونه به سادگی در کلاس همسایگان نزدیکش تعیین می‌گردد. فرد بودن مقدار k مفید می‌باشد چون با این کار جلوی آراء برابر گرفته می‌شود.
روش k همسایه نزدیک، برای بسیاری از روش‌ها کاربرد دارد، زیرا اثربخش، غیرپارامتریک و دارای پیاده‌سازی راحت می‌باشد. با این حال زمان دسته‌بندی‌اش طولانی است و یافتن مقدار k بهینه مشکل است. بهترین انتخاب از k، وابسته به داده‌ها می‌باشد به طور کلی مقدار بزرگ از k اثر نویز روی دسته‌بندی را کاهش می‌دهد، اما مرز مابین کلاس‌ها کمتر متمایز می‌شود.

شکل بالا مثالی از الگوریتم دسته‌بندی K نزدیک ترین همسایه، با استفاده از بردار ویژگی چندبعدی می‌باشد که مثلث‌ها کلاس اول و مربع‌ها کلاس دوم رانشان می‌دهند. دایره کوچک در داخل دوایر، نمونه تستی را نشان می‌دهد. حال اگر مقدار k=3 باشد (یعنی ۳ همسایه‌ی نزدیک به نمونه) نمونه تستی متعلق به کلاس مثلث و اگر k=5 باشد نمونه متعلق به کلاس مربع می‌باشد.

مراحل آموزش K همسایه نزدیک

مراحل آموزش K نزدیک‌ترین همسایه به صورت زیر می‌باشد، این الگوریتم یک نمونه تستی را بر اساس k همسایه نزدیک دسته‌بندی می‌کند. نمونه‌های آموزشی به عنوان بردارهایی در فضای ویژگی چند بعدی مطرح می‌شوند. فضا به ناحیه‌هایی با

نمونه‌های آموزشی پارتیشن‌بندی می‌شود. یک نقطه در فضا به کلاسی تعلق می‌یابد که بیشترین نقاط آموزشی متعلق به آن کلاس در داخل نزدیک‌ترین نمونه‌ی آموزشی به k در آن باشد.

معمولا فاصله‌ی اقلیدسی یا تشابه کوسینوسی در این روش استفاده می‌شود. در فاز دسته‌بندی K Nearest Neighbors، نمونه تستی به عنوان یک بردار در فضای ویژگی نمایش داده می‌شود و فاصله‌ی اقلیدسی یا تشابه کوسینوسی بردار تستی با کل بردارهای آموزشی محاسبه می‌شود و نزدیکترین نمونه‌ی آموزشی به k انتخاب می‌شود. البته راه‌های زیادی برای دسته‌بندی بردار تستی وجود دارد و بنابراین الگوریتم k همسایه نزدیک کلاسیک، یک نمونه تستی را بر اساس بیشترین آراء از k همسایه‌ی نزدیکش تعیین می‌کند. سه فاکتور مهم در این الگوریتم روش k همسایه‌ی نزدیک، به شرح زیر می‌باشد:

معیار فاصله یا شباهت، برای پیداکردن k همسایه نزدیک استفاده می‌شود.

K تعداد همسایه‌های نزدیک است.

قانون تصمیم‌گیری برای تعیین (شناسایی) یک کلاس برای سند تستی از k همسایه نزدیک می‌باشد.

الگوریتم

در الگوریتم K نزدیک‌ترین همسایه یک نمونه با رای اکثریت از همسایگان خود دسته بندی می شود، با این مورد به کلاس رایج ترین در میان K نزدیک ترین همسایگان خود را با استفاده از تابع فاصله (distance function) اندازه گیری شده است. اگر K = 1، سپس مورد به کلاس نزدیکترین همسایه اختصاص داده می شود.

Distance functions

Euclidean $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$

Manhattan $\sum_{i=1}^{k}|x_i - y_i|$

Minkowski $\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$

همچنین لازم به ذکر است که تمام سه اندازه گیری فاصله تنها برای متغیرهای پیوسته معتبر است. در مورد متغیرهای گسسته باید از فاصله همینگ (Hamming distance) استفاده شود که مسئله استانداردسازی متغیرهای عددی بین ۰ و ۱ را به وجود می آورد در حالی که مخلوطی از متغیرهای عددی و گسسته در مجموعه داده وجود دارد.

Hamming Distance

$$D_H = \sum_{i=1}^{k} |x_i - y_i|$$

$$x = y \Rightarrow D = 0$$

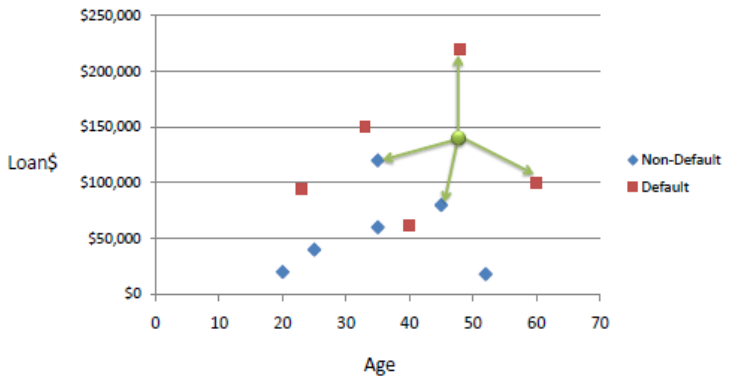
$$x \neq y \Rightarrow D = 1$$

X	Y	Distance
Male	Male	0
Male	Female	1

در روش K نزدیک‌ترین همسایه، انتخاب بهترین مقدار برای K بهتر است با اولین بررسی داده ها انجام شود. به طور کلی، یک مقدار بزرگ K دقیق تر است زیرا نویز کلی را کاهش می دهد اما هیچ تضمینی برای اعتبار آن وجود ندارد. اعتبار سنجی متقاطع یکی دیگر از راه های به دست آوردن یک K خوب با استفاده از یک مجموعه داده مستقل برای اعتبار سنجی K می باشد. به لحاظ تجربه، K مطلوب برای بیشتر مجموعه داده ها بین ۳ تا ۱۰ است. این نتایج بسیار بهتر از NN۱ تولید می کند.

مثال K نزدیک‌ترین همسایه

داده های زیر مربوط به اعتبار بانکی را در نظر بگیرید. سن و وام دو متغیر عددی (پیش بینی کننده) هستند و پیش فرض هدف است.



اکنون می توانیم از مجموعه آموزشی برای طبقه بندی یک مورد ناشناخته (سن = ۴۸ و وام = ۱۴۲،۰۰۰ $) با استفاده از فاصله اقلیدسی استفاده کنیم. اگر K = 1، نزدیک ترین همسایه آخرین مورد در مجموعه آموزش با Default = Y است.

D = Sqrt[(48-33)^2 + (142000-150000)^2] = 8000.01 >> Default=Y

Age	Loan	Default	Distance	
25	$40,000	N	102000	
35	$60,000	N	82000	
45	$80,000	N	62000	
20	$20,000	N	122000	
35	$120,000	N	22000	2
52	$18,000	N	124000	
23	$95,000	Y	47000	
40	$62,000	Y	80000	
60	$100,000	Y	42000	3
48	$220,000	Y	78000	
33	$150,000	Y	8000	1
48	**$142,000**	**?**		

Euclidean Distance

$$D = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

با $K = 3$، دو $Default = Y$ و یک $Default = N$ از سه نزدیکترین همسایگان وجود دارد. پیش بینی برای مورد ناشناخته است دوباره $Default = Y$.

فاصله استاندارد

یکی از معایب عمده در محاسبه اندازه گیری فاصله از مجموعه آموزش در K نزدیک‌ترین همسایه، در مورد متغیرهایی است که دارای مقیاس اندازه گیری متفاوت هستند و یا ترکیبی از متغیرهای عددی و دسته ای وجود دارد. به عنوان مثال، اگر یک متغیر بر مبنای درآمد سالانه در دلار باشد، و دیگری براساس سن سالها است، درآمد تاثیر زیادی بر محاسبه فاصله دارد. یک راه حل این است که مجموعه آموزشی را به صورت زیر تنظیم کنید.

Age	Loan	Default	Distance
0.125	0.11	N	0.7652
0.375	0.21	N	0.5200
0.625	0.31	N	0.3160
0	0.01	N	0.9245
0.375	0.50	N	0.3428
0.8	0.00	N	0.6220
0.075	0.38	Y	0.6669
0.5	0.22	Y	0.4437
1	0.41	Y	0.3650
0.7	1.00	Y	0.3861
0.325	0.65	Y	0.3771
0.7	**0.61**	**?**	

Standardized Variable

$$X_s = \frac{X - Min}{Max - Min}$$

با استفاده از فاصله استاندارد شده در یک مجموعه آموزشی مشابه، نمونه ناشناخته یک همسایه متفاوت را باز می کند که نشانه خوبی از ثبات در K نزدیک ترین همسایه نیست.

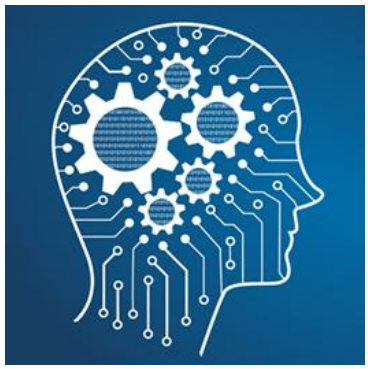
«بهینه سازی ریاضیاتی» (Mathematical Optimization) که به آن «برنامه‌نویسی ریاضیاتی» (Mathematical Programming) نیز گفته می‌شود، فرایندی است که در آن بهترین جواب (با توجه به مجموعه‌ای از معیارها) از میان مجموعه‌ای از جواب‌های ممکن، برای یک مسأله خاص انتخاب می‌شود. امروزه مسائل بهینه سازی در تمامی رشته‌های علمی کَمی (Quantitative Disciplines) نظیر «علوم کامپیوتر» (Computer Science)، مهندسی (Engineering)، «تحقیق در عملیات» (Operations Research)، «اقتصاد» (Economics) و سایر موارد مورد استفاده قرار

می‌گیرند. در طول قرن‌های متمادی، توسعه روش‌های تولید جواب و حل مسأله، یکی از حوزه‌های مهم تحقیقاتی در علم «ریاضیات» (Mathematics) بوده است و اهمیت آن‌ها در طی چند سال گذشته چند برابر شده است.

جهت درک بهتر و اصولی‌تر حوزه بهینه سازی (ریاضیاتی)، ابتدا لازم است تا شناخت کافی در مورد مسائل بهینه سازی ایجاد شود. به بیان ساده، در یک «مسأله بهینه سازی» (Optimization Problems)، هدف «کمینه‌سازی» (Minimizing) یا «بیشینه‌سازی» (Maximizing) یک «تابع حقیقی» (Real Function) است؛ برای چنین کاری سیستم بهینه سازی ریاضیاتی، به طور سیستماتیک، مقادیر ورودی را از یک مجموعه مجاز از مقادیر انتخاب و پس از آن، مقدار تابع حقیقی را محاسبه می‌کند. تعمیم نظریه بهینه سازی (Optimization Theory) و تکنیک‌های آن به دیگر حوزه‌های علمی و تحقیقاتی مرتبط، در حیطه کاربردهای مهم رشته «ریاضیات کاربردی» (Applied Mathematics) محسوب می‌شود. به طور کلی، اصطلاح بهینه سازی به فرایندی اطلاق می‌شود که هدف آن پیدا کردن بهترین مقادیر یک (یا چند) «تابع هدف» (Objective Functions) در یک دامنه تعریف شده است.

مؤلفه‌های یک مدل بهینه سازی

بهینه سازی ابزار مهمی در «تصمیم‌گیری» (Decision Making) و تحلیل سیستم‌های فیزیکی محسوب می‌شود. از نظر ریاضی، یک مسأله بهینه سازی، مسأله پیدا کردن بهترین جواب از میان مجموعه‌ای از جواب‌های کاندید (Candidate) یا امکان‌پذیر (Feasible) است.



ساختن یک مدل بهینه سازی

اولین گام در فرایند بهینه سازی، ساختن یک مدل مناسب برای بهینه سازی است؛ مدل‌سازی (Modelling)، به فرایند شناسایی و نمایش ریاضی «هدف» (Objective)، «متغیرها» (Variables) و «قیدهای» (Constraints) مسأله گفته می‌شود:

- هدف (Objective)، معیار کمی عملکرد سیستم است که قرار است کمینه یا بیشینه شود. به عنوان نمونه، در «تولید» (Manufacturing) هدف کمینه‌سازی هزینه‌های تولید و یا بیشینه‌سازی سود است. در حالی که در برازش داده‌ها (Data Fitting) روی یک مدل، هدف کمینه‌سازی انحراف کل داده‌های مشاهده شده (Observed Data) از داده‌های پیش‌بینی شده (Predicted Data) است.

- متغیرها (Variables) یا به عبارت دیگر «ناشناخته‌ها» (Unknowns)، مؤلفه‌هایی از مدل بهینه سازی محسوب می‌شوند که قرار است مقادیر مناسبی برای آن‌ها یافت شود. به عنوان نمونه در فرایند تولید، متغیرها می‌توانند پارامترهایی نظیر مقدار منابع مصرف شده و یا زمان صرف شده در هر فعالیت باشد. در حالی که در برازش داده‌ها، متغیرها همان پارامترهای مدل خواهند بود.
- قیدها (Constraints) توابعی هستند که روابط میان متغیرها (Variables) را نشان می‌دهد و از این طریق، مقادیر مجاز برای متغیرها مشخص می‌شود. به عنوان نمونه در فرایند تولید، مقدار منابع مصرف شده نمی‌تواند از مقدار منابع در دسترس فراتر برود.

مشخص کردن نوع مسأله بهینه سازی

گام دوم در فرایند بهینه سازی، مشخص کردن نوع و یا طبقه‌بندی مسأله‌ای است که قرار است بهینه‌سازی شود. در ادامه و در بخش‌های بعدی، طبقه‌بندی مسائل بهینه سازی ارائه خواهد شد. پس از این مرحله و مشخص شدن نوع مسأله بهینه‌سازی، نیاز است تا بسته به دامنه و نیازهای مسأله، از میان ابزارهای بهینه‌سازی موجود (تجاری و آکادمیک) برای بهینه‌سازی توابع و پیدا کردن جواب‌های بهینه (یا تقریبی از آن‌ها)، ابزاری که بیشترین تطابق را با نیازهای مسأله دارد، انتخاب کرد و آن را در دامنه مورد نظر مورد استفاده قرار داد.

مسائل بهینه سازی (Optimization Problems)

مسائل بهینه سازی را می‌توان به شکل زیر نمایش داد:

با داشتن: یک تابع به فرم $f:A\rightarrow Rf:A\rightarrow R$ که مقادیر را از مجموعه‌ای نظیر AA (دامنه مسأله) به اعداد حقیقی (Real Numbers) نگاشت می‌کند.

هدف: پیدا کردن عنصری مثل $X0\in AX0\in A$ است، به طوری که شرط

$f(X0)\leq f(X)forall\ X\in Af(X0)\leq f(X)forall\ X\in A$ (برای مسائل کمینه‌سازی) یا شرط

$f(X0)\geq f(X)forall\ X\in Af(X0)\geq f(X)forall\ X\in A$ (برای مسائل بیشینه‌سازی) برقرار باشد.

به فرمول‌بندی مسائل به شکل نشان داده شده، نمایش مسأله بهینه سازی یا نمایش مسأله برنامه‌نویسی ریاضی (Mathematical Programming Problem) گفته می‌شود. نکته بسیار مهم در مورد مسائل بهینه‌سازی این است که بسیاری از مسائل «جهان واقعی» (Real-World) و مسائل تئوری را می‌توان از طریق چارچوب عمومی نمایش داده شده مدل‌سازی کرد. به فرمول بندی مسائل بهینه سازی در قالب زیر توجه کنید:

$$f(X0)\geq f(X)\Leftrightarrow\tilde{f}(X0)\leq\tilde{f}(X)f(X0)\geq f(X)\Leftrightarrow f\sim(X0)\leq f\sim(X)$$
$$\tilde{f}(X):=-f(X),\tilde{f}:A\rightarrow Rf\sim(X):=-f(X),f\sim:A\rightarrow R$$

با توجه به این که فرمول‌بندی مسائل بهینه سازی در قالب مسائل کمینه‌سازی (فرمول‌بندی بالا)، معتبر (Valid) است، در چنین حالتی حل کردن مسائل کمینه سازی (با استفاده از فرمول‌بندی نمایش داده شده) ساده‌تر خواهد بود. با این حال، فرمول‌بندی مسائل بهینه سازی در قالب مسائل بیشینه‌سازی نیز معتبر خواهد بود.

مفاهیم اولیه در بهینه سازی (ریاضیاتی)

روش‌های بهینه سازی، یکی از معتبرترین و قابل قبول‌ترین روش‌های حل مسأله در حوزه‌های مختلف علمی محسوب می‌شوند. به عنوان نمونه، در رشته فیزیک، به مسائلی که از طریق چارچوب بالا فرمول‌بندی شوند، تکنیک‌های «کمینه‌سازی انرژی» (Energy Minimization) گفته می‌شود. به عبارت دیگر در اصل، مقدار تابع ff انرژی سیستمی که مدل‌سازی شده است را نشان می‌دهد. از جمله سیستم‌های بهینه سازی که از مفاهیم موجود در حوزه فیزیک («فیزیک آماری» (Statistical Physics)) استفاده می‌کنند، می‌تواند به الگوریتم «بهینه سازی اکسترمال» (Extremal Optimization) اشاره کرد. در ادامه، کدهای پیاده‌سازی الگوریتم بهینه سازی اکسترمال در زبان Ruby نمایش داده شده است:

همچنین در حوزه «هوش مصنوعی» (Artificial Intelligence) و «یادگیری ماشین» (Machine Learning) نیز بسیار حیاتی است که کیفیت «مدل داده» (Data Model)، به طور پیوسته و توسط یک «تابع هزینه» (Cost Function) ارزیابی شود؛ در این حالت، منظور از کمینه‌سازی تابع هزینه، پیدا کردن مجموعه‌ای از «پارامترهای بهینه» (Optimal Parameters) است که سبب تولید مقدار بهینه خطا (کمترین خطای ممکن) در سیستم می‌شوند.

از جمله الگوریتم‌های بهینه سازی که در دسته «الگوریتم‌های تخمین توزیع» (Estimation of Distribution Algorithms) قرار می‌گیرد، می‌توان به الگوریتم «بهینه سازی بیزی» (Bayesian Optimization) اشاره کرد. در ادامه، کدهای پیاده‌سازی الگوریتم بهینه سازی بیزی در زبان Ruby نمایش داده شده است:

در فرمول‌بندی مسائل بهینه سازی (Optimization Problems)، مجموعه AA زیر مجموعه‌ای از «فضای اقلیدسی» (Euclidean Space) یا RnRn است. معمولا در مسائل بهینه سازی، برای AA مجموعه‌ای از «قیود» (Constraints) تعریف می‌شوند؛ قیود، «برابری‌ها» (Equalities) یا «نابرابری‌هایی» (Inequalities) هستند که تمامی اعضای مجموعه AA باید آن‌ها را ارضا کنند. همچنین، به مجموعه AA یا همان دامنه یک تابع بهینه‌سازی مانند ff (به عبارت دیگر، دامنه تابع هدف مسأله بهینه‌سازی)، «فضای جستجو» (Search Space) یا «مجموعه انتخاب» (Choice Set) نیز گفته می‌شود.

با توجه به تعاریف انجام شده، هر یک از اعضای مجموعه AA ، یک جواب کاندید یا جواب امکان‌پذیر برای تابع بهینه‌سازی ff محسوب می‌شوند. تابع بهینه‌سازی ff با نام‌های دیگری نظیر «تابع هدف» (Object Function)، «تابع زیان» (Loss Function) یا «تابع هزینه» (Loss Function)، «تابع برازندگی» (Fitness Function) و در برخی رشته‌های خاص، «تابع انرژی» (Energy Function) شناخته می‌شود. تابع زیان و تابع هزینه برای مسائل کمینه‌سازی و تابع هدف، برای مسائل بیشینه‌سازی مورد استفاده قرار می‌گیرند.

به یک جواب کاندید یا امکان‌پذیر، که تابع هدف مسأله بهینه سازی را بیشینه یا کمینه کند، «جواب بهینه» (Optimal Solution) گفته می‌شود. همانطور که پیش از این نیز اشاره شد، مسائل بهینه سازی در ریاضیات، معمولا در قالب مسائل «کمینه‌سازی» (Minimization) نمایش داده می‌شوند.

کمینه محلی (Local Minimum)، بیشینه محلی (Local Maximum) و بهینه سراسری (Global Optimum)

یک کمینه محلی X⋆X⋆ تابع ff ، عنصری است که به ازاء آن پارامتری مانند δ>0δ>0 وجود دارد، به طوری که:

∀X∈Awhere‖X−X⋆‖≤δ∀X∈Awhere‖X−X⋆‖≤δ,

و به ازاء تمامی مقادیر XX که قید (Constraint) بالا را ارضا می‌کنند، عبارت f(X⋆)≤f(X)f(X⋆)≤f(X) برقرار است. به عبارت دیگر در برخی نواحی اطراف X⋆X⋆ ، تمامی مقادیر تابع ff ، بزرگتر یا برابر با مقدار عنصر کمینه محلی هستند. همچنین، بیشینه محلی به صورت مشابه و به شکل زیر تعریف می‌شود:

یک بیشینه محلی X⋆X⋆ تابع ff ، عنصری است که به ازاء آن پارامتری مانند δ>0δ>0 وجود دارد، به طوری که:

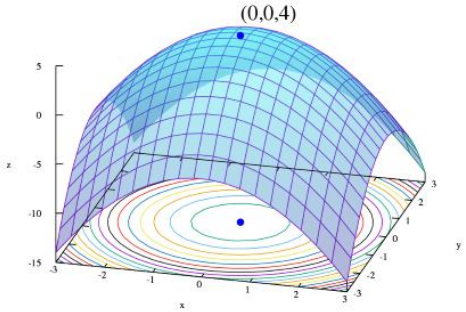
∀X∈Awhere‖X−X⋆‖≤δ∀X∈Awhere‖X−X⋆‖≤δ,

و به ازاء تمامی مقادیر XX که قید (Constraint) بالا را ارضا می‌کنند، عبارت f(X⋆)≥f(X)f(X⋆)≥f(X) برقرار است. به عبارت دیگر در برخی نواحی اطراف X⋆X⋆ ، تمامی مقادیر تابع ff ، کوچکتر یا برابر با مقدار عنصر بیشینه محلی هستند.

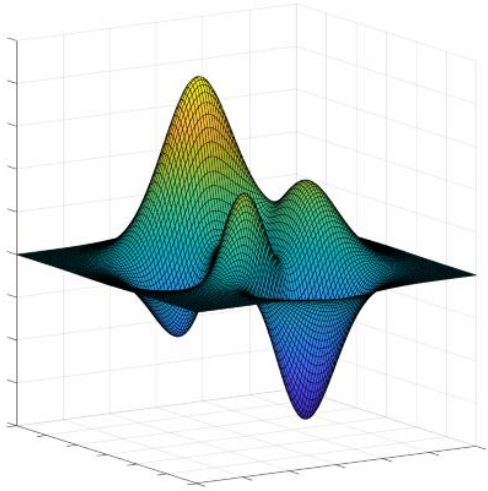
بهینه سازی محدب و غیر محدب

اگرچه عناصر کمینه یا بیشینه محلی بهتر از (یا حداقل برابر با) عناصر همسایگی اطرافشان هستند، ولی یک عنصر بیشینه یا کمینه سراسری (یا همان مقدار بهینه سراسری (Global Optimum)) از تمامی جواب‌های کاندید یا امکان‌پذیر برای یک مسأله بهینه سازی بهتر خواهد بود. به طور کلی، تنها در صورتی که تابع هدف مسأله بهینه سازی «محدب» (Convex) نباشد، ممکن است بیش از یک «کمینه محلی» (Local Minimum) یا «بیشینه محلی» (Local maximum) در مسأله وجود داشته باشد.

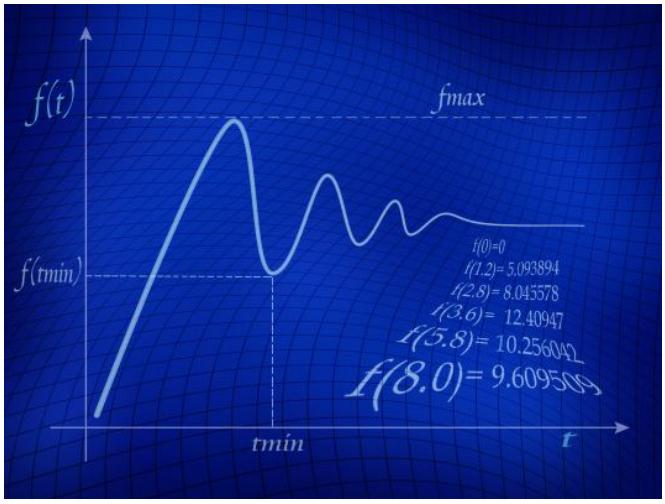
در مسائل «بهینه سازی محدب» (Convex Optimization)، در صورتی که یک جواب بیشینه محلی یا کمینه محلی برای مسأله وجود داشته باشد، این جواب، بهینه سراسری مسأله مورد نظر نیز خواهد بود. با این حال، در یک مسأله بهینه سازی «غیر محدب» (Non-Convex) ممکن است بیش از یک جواب بیشینه محلی یا کمینه محلی وجود داشته باشد که لزوما همه آن‌ها بهینه سراسری مسأله مورد نظر نخواهند بود.



بسیاری از الگوریتم‌هایی که برای حل مسائل بهینه سازی غیر محدب پیشنهاد شده‌اند (از جمله بسیاری از الگوریتم‌های تجاری تولید شده برای حل مسائل بهینه سازی غیر محدب)، به خوبی قادر نیستند میان جواب‌های بهینه محلی (بیشینه محلی یا کمینه محلی) و جواب‌های بهینه سراسری (بیشینه سراسری یا کمینه سراسری) تمایز قائل شوند و تمامی جواب‌های بهینه محلی یافت شده را به عنوان جواب واقعی مسأله بهینه سازی مورد نظر به شمار می‌آورند.



برای غلبه بر چنین معضلی در روش‌های بهینه سازی، زیر شاخه‌ای به نام «بهینه سازی سراسری» (Global Optimization) در حوزه‌های ریاضیات کاربردی و «آنالیز عددی» (Numerical Analysis) پدید آمده است. هدف روش‌های بهینه سازی سراسری، پیاده‌سازی «الگوریتم‌های قطعی» (Deterministic Algorithms) است که قادر هستند «همگرایی» (Convergence) به جواب بهینه سراسری واقعی یک مسئله بهینه سازی محدب را در زمان محدود تضمین کنند.



نمادگذاری‌ها در بهینه سازی ریاضیاتی

در ادامه، مهم‌ترین نمادگذاری‌ها در بهینه سازی ریاضیاتی مورد بررسی قرار می‌گیرند.

مقدار کمینه یا بیشینه یک تابع

نمادگذاری زیر را در نظر بگیرید:

minx∈R(x2+1)minx∈R(x2+1)

نمادگذاری نمایش داده شده، مقدار کمینه تابع هدف x2+1x2+1 را، وقتی که مقدار xx از مجموعه اعداد حقیقی RR انتخاب شده باشد، نمایش می‌دهد. مقدار کمینه این تابع برابر با ۱ (یک) است و در نقطه x=0x=0 رخ می‌دهد. به طور مشابه، نمادگذاری زیر:

maxx∈R(2x)maxx∈R(2x)

مقدار بیشینه تابع هدف (2x)(2x) را نمایش می‌دهد. در این تابع، پارامتر xx می‌تواند هر مقدار حقیقی به خود بگیرد. در این حالت، از آنجایی که تابع هدف «بی‌کران» (Unbounded) است، هیچ مقدار بیشینه‌ای برای آن وجود نخواهد داشت. به عبارت دیگر، جواب این مسأله «بی‌نهایت» (Infinity) یا «تعریف نشده» (Undefined) است.

آرگومان‌های ورودی بهینه (Optimal Input Argument)

نمادگذاری زیر را در نظر بگیرید:

argminx∈(−∞,−1](x2+1)argminx∈(−∞,−1](x2+1)

نمادگذاری که در ادامه مشاهده خواهید کرد، هیچ تفاوتی با نمادگذاری بالا ندارد:

argminx(x2+1),subjectto:x∈(−∞,−1]argminx(x2+1),subjectto:x∈(−∞,−1]

نمادگذاری‌های نمایش داده شده، مقدار (یا مقادیر) آرگومان xx در بازه x∈(−∞,−1]x∈(−∞,−1] را نشان می‌دهد؛ مقادیری که تابع هدف x2+1x2+1 را کمینه‌سازی می‌کنند. نکته مهم در مورد نمادگذاری تعریف شده برای مسأله بهینه‌سازی بالا این است که در این نمادگذاری، مقادیر کمینه تابع هدف مشخص نشده‌اند و تنها قید مرتبط با مقادیر ممکن برای آرگومان xx نمایش داده شده است. در این مورد، از آنجایی که مقدار صفر در مجموعه مقادیر امکان‌پذیر آرگومان xx وجود ندارد، مقدار کمینه این تابع در نقطه x=−1x=−1 حاصل می‌شود.

به طور مشابه، برای نمایش آرگومان‌های ورودی بهینه یک مسأله بیشینه‌سازی، نمادگذاری زیر را در نظر بگیرید:

argmaxx∈[−5,5],y∈R(xcosy)argmaxx∈[−5,5],y∈R(xcosy)

مانند حالت قبل، نمادگذاری که در ادامه مشاهده خواهید کرد، هیچ تفاوتی با نمادگذاری بالا ندارد:

argmaxx,y(xcosy),subjectto:x∈[−5,5],y∈Rargmaxx,y(xcosy),subjectto:x∈[−5,5],y∈R

نمادگذاری‌های نمایش داده شده، جفت مقادیر (x,y)(x,y) را نشان می‌دهند که مقدار تابع هدف xcosyxcosy را بیشینه می‌کنند؛ در این تابع هدف، قید خاصی روی آرگومان xx تعریف شده است که بر اساس آن، تنها مقادیر موجود در بازه [−5,5][−5,5] قابل قبول هستند. مانند مورد قبلی، در نمادگذاری تعریف شده برای مسأله بیشینه‌سازی بالا، مقادیر بیشینه تابع هدف مشخص نشده‌اند و تنها قیدهای مرتبط با مقادیر ممکن برای آرگومان xx و yy نمایش داده شده‌اند. در این مورد، جفت مقادیر (x,y)(x,y) که به فرم (5,2kπ)(5,2kπ) و (−5,(2k+1)π)(−5,(2k+1)π) هستند، جواب‌های این مسأله بهینه‌سازی هستند؛ پارامتر kk می‌تواند هر مقدار صحیحی را به خود بگیرد.

بنابراین، عملگرهای argminargmin و argmaxargmax ، به ترتیب بیانگر «آرگومان کمینه تابع هدف» (Argument of Minimum) و «آرگومان بیشینه تابع هدف» (Argument of Maximum) هستند و مشخص می‌کنند که مسأله مورد نظر، یک مسأله کمینه‌سازی است یا یک مسأله بیشینه‌سازی.

انواع مسائل بهینه سازی

همانطور که پیش از این نیز اشاره شد، گام دوم در فرایند بهینه سازی، مشخص کردن نوع و یا طبقه‌بندی مسأله‌ای است که قرار است بهینه‌سازی شود. از آنجایی که هر یک از الگوریتم‌های حل مسائل بهینه سازی (Optimization)، جهت یافتن مقادیر بهینه انواع خاصی از مسائل پیاده‌سازی می‌شوند، مشخص کردن «دسته‌بندی» (Classification) مدل‌های بهینه سازی نقش مهمی در این فرایند ایفا خواهد کرد. در ادامه، برخی از مهم‌ترین دسته‌بندی‌های ارائه شده از مدل‌های بهینه سازی نمایش داده خواهد شد.

مسائل بهینه سازی «پیوسته» (Continuous) و بهینه سازی «گسسته» (Discrete)

برخی از مدل‌های بهینه سازی، جهت بهینه سازی توابعی طراحی شده‌اند که متغیرهای آن‌ها، مقادیر مجاز خود را از یک «مجموعه گسسته» (Discrete Set) اتخاذ می‌کنند (زیر مجموعه‌ای از مقادیر صحیح)، در حالی که مدل‌های دیگر، برای بهینه سازی توابعی پیاده‌سازی شده‌اند که متغیرهای آن‌ها می‌توانند مقادیر حقیقی (Real Values) به خود بگیرند. به مدل‌هایی که از متغیرهای گسسته استفاده می‌کنند، مدل‌های مسائل بهینه سازی گسسته (Discrete Optimization Problems) و به مدل‌هایی که برای بهینه سازی متغیرهای پیوسته مورد استفاده قرار می‌گیرند، مدل‌های مسائل بهینه سازی پیوسته (Continuous Optimization Problems) گفته می‌شود.

حل کردن مسائل بهینه سازی پیوسته معمولا سخت‌تر از حل مسائل بهینه‌سازی گسسته است؛ از آنجایی که سطح توابع پیوسته هموار (Smooth) هستند، این امکان وجود دارد که از توابع هدف و قیدهای تعریف شده برای مقادیر متغیرها در نقطه XX ، جهت استنتاج اطلاعات مرتبط با نقاط موجود در همسایگی این نقطه استفاده کرد و از این طریق، مقادیر بهینه مسأله بهینه سازی پیوسته را پیدا کرد. با این حال، پیشرفت‌های حاصل شده در توسعه الگوریتم‌های بهینه سازی گسسته و افزایش قابل توجه قدرت محاسباتی سیستم‌های کامپیوتری در چند سال اخیر سبب شده است تا الگوریتم‌هایی توسعه یابند که قادر هستند مسائل بهینه سازی گسسته بزرگ و پیچیده را به شکل بسیار کارآمدی حل کنند.

نکته مهم در مورد این دسته از مدل‌های بهینه سازی این است که مدل‌های پیوسته نقش مهمی در پیاده‌سازی مدل‌های بهینه سازی گسسته دارند. زیرا، بسیاری از مدل‌های بهینه‌سازی گسسته، مسائل بهینه سازی را به دنباله‌ای از زیر مسائل پیوسته (Continuous Subproblems) تقسیم‌بندی می‌کنند و برای حل کردن هر یک از این زیر مسائل، از مدل های بهینه سازی پیوسته استفاده می‌شود. بنابراین، یکی از مؤلفه‌های اساسی مدل‌های بهینه سازی گسسته، روش‌های حل مسائل بهینه‌سازی پیوسته هستند. یکی از الگوریتم‌هایی که برای بهینه سازی مسائل پیوسته مورد استفاده قرار می‌گیرد، «الگوریتم بهینه‌سازی فاخته» (Cuckoo Optimization Algorithm) نام دارد. در ادامه، کدهای پیاده سازم این الگوریتم در زبان متلب نمایش داده شده است.

مسائل بهینه سازی «مقید» (Constrained) و بهینه سازی «نامقید» (Unconstrained)

از یک جهت دیگر نیز می‌توان میان مسائل و مدل‌های بهینه سازی قائل شد: مسائلی که هیچ قیدی روی متغیرها تعریف نمی‌کنند و مسائلی که برای متغیرها قید تعریف می‌کنند. درصد قابل توجه از مسائلی که در جهان واقعی با آن‌ها سروکار داریم، از نوع مسائل بهینه سازی نامقید هستند. در چند سال اخیر، مطالعات زیادی برای فرمول‌بندی دوباره (Re-Formulation) مسائل بهینه سازی نامقید در قالب مسائل بهینه سازی مقید ارائه شده است. در این مطالعات، از طریق جایگزین کردن قیدهای مسأله با «عبارات جریمه» (Penalty Terms)، یک مسأله بهینه سازی مقید به یک مسأله بهینه سازی نامقید تبدیل می‌شود.

مسائل بهینه سازی مقید مربوط به کاربردهایی هستند که در آن‌ها قیدهای صریحی (Explicit Constraints) رو متغیرهای مسأله تعریف شده است. قیدهای تعریف شده روی این دسته از مثال می‌توانند کران‌های (Bounds) ساده و یا سیستم‌هایی از معادلات و نامعادلات باشند که روابط میان متغیرهای مسأله را مدل می‌کنند.

همچنین مسائل بهینه‌سازی مقید را می‌توان بر اساس طبیعت قیدهای تعریف شده رو متغیرها (به عنوان نمونه، خطی، غیر خطی، محدب و سایر موارد) و هموار بودن توابع (به عنوان نمونه، مشتق‌پذیر (Differentiable) یا نامشتق‌پذیر (Non-Differentiable))، به طبقه‌های دیگری دسته‌بندی کرد.

مسائل بهینه سازی «قطعی» (Deterministic) و بهینه سازی «تصادفی» (Stochastic)

در مسائل بهینه سازی قطعی، فرض بر این است که داده‌های مسأله داده شده به طور دقیق شناخته شده هستند. با این حال و به دلایل مختلفی، داده‌های بسیاری از مسائل داده شده را به طور دقیق نمی‌توان شناخت (یا در اختیار داشت). اولین دلیل برای در دسترس نبودن داده‌های دقیق مسأله، «خطای اندازه‌گیری» (Measurement Error) داده‌ها است. دلیل دوم و بسیار مهم برای شناخته نبودن داده‌های مسأله این واقعیت است که برخی از داده‌ها، اطلاعات مرتبط با آینده را نمایش می‌دهند (نظیر تقاضا برای یک محصول یا قیمت یک محصول در آینده) که نمی‌توان با قطعیت (Certainty) آنها را شناخت.

در بهینه سازی همراه با عدم قطعیت (Optimization Under Uncertainty) یا بهینه سازی تصادفی (Stochastic Optimization)، عدم قطعیت در مدل ترکیب شده است. تکنیک‌های «بهینه سازی استوار» (Robust Optimization) تنها زمانی می‌توانند مورد استفاده قرار بگیرند که پارامترهای مسأله درون کران‌های مشخصی قرار گرفته باشند؛ در چنین حالتی، هدف پیدا کردن پیدا کردن جوابی است که به ازاء تمامی داده‌ها، امکان‌پذیر (یا قابل قبول) و به نحوی بهینه باشد.

مدل‌های بهینه سازی تصادفی (Stochastic Optimization)، از این واقعیت که «توزیع احتمالی» (Probability Distribution) حاکم بر داده‌ها مشخص و یا قابل تخمین است، بهره می‌برند؛ در چنین حالتی، هدف پیدا کردن یک سیاست (Policy) است که برای تمامی (یا تقریبا تمامی) نمونه‌های داده قابل قبول باشد و عملکرد مورد انتظار مدل (یا سیستم) را بهینه سازی کند.

مسائل بهینه سازی بدون هدف، تک هدفه و چند هدفه

بیشتر مسائل بهینه‌سازی، از یک تابع هدف برخوردار هستند. با این حال، مسائل بهینه سازی جالبی می‌توان پیدا کرد که یا هیچ تابع هدفی ندارند و یا از چندین تابع هدف برخوردار هستند. «مسائل امکان‌پذیری» (Feasibility Problems) مسائلی هستند که در آن‌ها، هدف پیدا کردن مقادیری برای متغیرها است که قیدهای مدل را ارضا کنند؛ در چنین حالتی مدل، تابع هدف خاصی ندارد که بخواهد آن را بهینه سازی کند. «مسائل مکملی» (Complementarity Problems)، مسائل بسیار فراگیری در مهندسی و اقتصاد محسوب می‌شوند؛ در چنین مواردی، هدف مدل‌های بهینه سازی پیدا کردن جوابی است که «شرایط مکملی» (Complementarity Conditions) مسأله را ارضا کند.

«مسائل بهینه سازی چند هدفه» (Multi-Objective Optimization Problems) در بسیاری از حوزه‌های علمی و کاربردی نظیر مهندسی، اقتصاد و «لجستیک» (Logistics) کاربرد دارند و زمانی مورد استفاده قرار می‌گیرند که برای رسیدن به تصمیمات بهینه در سیستم، نیاز است میان دو یا چند «هدف متناقض» (Conflicting Objectives) موازنه برقرار شود. به عنوان نمونه، توسعه یک مؤلفه جدید ممکن است نیازمند کمینه کردن وزن و به طور همزمان، بیشینه کردن قدرت باشد.

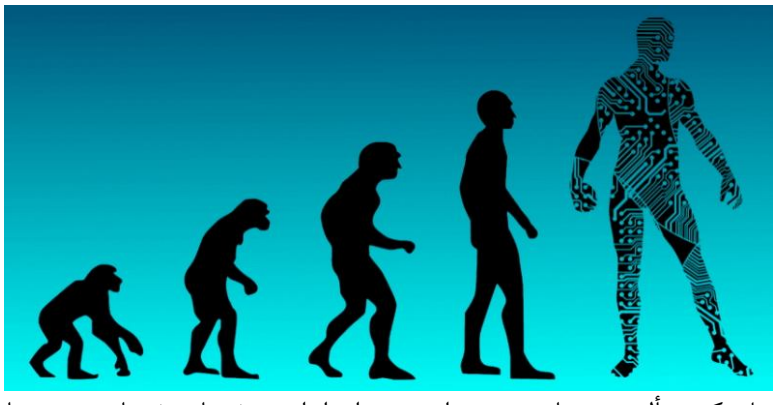
همچنین، جهت انتخاب سبد سرمایه‌گذاری مناسب باید به نحوی عمل کرد که بازگشت مورد انتظار سرمایه، بیشینه و ریسک سرمایه گذاری، کمینه گردد. در عمل، برای بهینه سازی مسائل چند هدفه، از راهکارهایی استفاده می‌شود که در آن‌ها، مسائل چند هدفه (Multi-Objective) به مسائل تک هدفه (Single-Objective) تبدیل می‌شوند؛ در این دسته از راهکارها، از طریق تشکیل یک ترکیب وزن‌دار از توابع هدف مختلف و یا به وسیله جایگزین کردن توابع هدف با قیدهای مشخص، مسائل بهینه سازی چند هدفه به مسائل تک هدفه تبدیل می‌شوند. الگوریتم بهینه سازی فاخته (Cuckoo Optimization Algorithm) یکی از الگوریتم‌های توسعه داده شده برای حل «مسائل بهینه‌سازی غیرخطی» (Non-Linear Optimization Problems) و «مسائل بهینه‌سازی پیوسته» (Continuous Optimization Problems) محسوب می‌شود. این الگوریتم، از زندگی خانواده‌ای از پرندگان به نام «فاخته» (Cuckoo) الهام گرفته شده است. الگوریتم بهینه سازی فاخته براساس شیوه زندگی بهینه و ویژگی‌های جالب این گونه، نظیر تخم‌گذاری و تولید مثل آن‌ها ساخته شده است.

فهرست مطالب این نوشته پنهان کردن

فاخته‌های بالغ و تخم‌های فاخته، جمعیت اولیه الگوریتم بهینه سازی فاخته را تشکیل می‌دهند. فاخته‌های بالغ در لانه پرندگان دیگر تخم‌گذاری می‌کنند. در صورتی که تخم‌های فاخته توسط پرندگان میزبان بالغ شناسایی نشوند و از بین نروند، رشد کرده و به فاخته‌های بالغ تبدیل خواهند شد. فاخته‌های بالغ تحت تأثیر ویژگی‌های محیطی و به امید یافتن محیط بهینه برای زندگی و تولید مثل، به صورت گروهی مهاجرت می‌کنند. در این الگوریتم، محیط بهینه همان «بهینه سراسری» (Global Optimum) در تابع هدف مسأله بهینه‌سازی خواهد بود. این الگوریتم تاکنون در سناریوهای بهینه‌سازی مختلف و کاربردهای جهان واقعی، عملکرد خوبی از خود نشان داده است. در ادامه معرفی مختصری از «الگوریتم‌های تکاملی» (Evolutionary Algorithms) ارائه خواهد شد؛ سپس الگوریتم بهینه سازی فاخته و بخش‌های مختلف آن توضیح داده می‌شود. در نهایت، برخی از کاربردهای این الگوریتم در حل مسائل بهینه‌سازی مورد بررسی قرار می‌گیرد.

مقدمه‌ای از الگوریتم‌های تکاملی

به فرآیند تغییر و دستکاری ورودی‌ها و یا خصوصیات یک دستگاه، عملیات ریاضی یا یک آزمایش علمی که در جهت رسیدن به یک خروجی با جواب بهینه انجام می‌شود، بهینه‌سازی گفته می‌شود. در فرآیند بهینه‌سازی، ورودی، متغیرهای مسأله هستند. خروجی این فرآیند، «برازندگی» (Fitness) نامیده می‌شود و فرآیند یا تابعی که قرار است بهینه شود، «تابع هدف» (Objective Function) نام دارد.

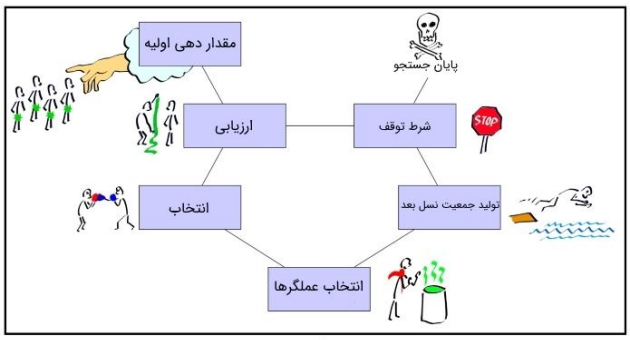


روش‌های متفاوتی برای حل یک مسأله بهینه‌سازی وجود دارد. دسته‌ای از این روش‌ها، روش‌های «بهینه‌سازی ریاضیاتی» (Mathematical Optimization) هستند که بر پایه فرآیندهای «قطعی» (Deterministic) ریاضی بنا نهاده شده‌اند. به این دسته فرآیندهای بهینه‌سازی، «برنامه‌نویسی ریاضی» (Mathematical Programming) نیز گفته می‌شود.

دسته دیگر، روش‌هایی هستند که از فرآیندهای طبیعی الهام گرفته شده‌اند. این دسته روش‌های بهینه‌سازی، مبتنی بر فرآیندهای تصادفی و روش‌های «مولد اعداد تصادفی» (Random Number Generator) هستند. چنین روش‌هایی معمولا کار خود را با مجموعه‌ای از متغیرهای اولیه شروع می‌کنند. این متغیرها به شکل تصادفی (یا شبه تصادفی) مقداردهی اولیه می‌شوند. روش‌های بهینه‌سازی الهام گرفته شده از طبیعت سعی دارند تا در یک فرآیند تکاملی، همگرایی جمعیت اولیه تولید شده از متغیرها را به بهینه سراسری تضمین و از این طریق، جواب بهینه را برای مسأله بهینه‌سازی تولید کنند.

یکی از معروف‌ترین و پر کاربردترین الگوریتم‌های بهینه‌سازی تکاملی، «الگوریتم ژنتیک» (Genetic Algorithm) است. این الگوریتم از عملگرهایی برای تکامل جمعیت اولیه استفاده می‌کند که از تغییرات ژنتیکی طبیعی در موجودات زنده و اصل «انتخاب طبیعی» (Natural Selection) الهام گرفته شده‌اند. از جمله دیگر الگوریتم‌های تکاملی می‌توان به الگوریتم «بهینه‌سازی ازدحام ذرات» (Particles Swarm Optimization) اشاره کرد. این الگوریتم از رفتارهای اجتماعی پرواز پرندگان برای یافتن غذا یا حرکت گروهی و در جهت یکسان ماهی‌ها الهام گرفته شده است.

نمونه موفق دیگری از شبیه‌سازی فرآیندهای طبیعی برای حل مسائل بهینه‌سازی، الگوریتم «بهینه‌سازی کلونی مورچگان» (Ants Colony Optimization) است. این الگوریتم از رفتارهای اجتماعی مورچگان برای یافتن منابع غذایی الهام گرفته شده است؛ رفتار اجتماعی بهینه‌ای که از طریق باقی گذاشتن رد «فرومون» (Pheromone) در محیط حاصل می‌شود. رد فرومون به جا گذاشته شده از مورچگان در محیط، مسیر بهینه از کلونی تا منبع غذایی یافت شده را نشان می‌دهد.



شمای کلی الگوریتم‌های تکاملی

مهم‌ترین مزایای استفاده از الگوریتم‌های تکاملی برای حل مسائل بهینه‌سازی عبارتند از:

این دسته از الگوریتم‌های بهینه‌سازی معمولا در برابر تغییرات پویای محیط عملیاتی بسیار مقاوم هستند. یکی از نقاط ضعف روش‌های سنتی بهینه‌سازی (نظیر بهینه‌سازی‌های ریاضی) این است که کوچک‌ترین تغییر ایجاد شده در محیط، جواب‌های بهینه از پیش یافته شده برای مسأله مورد نظر را عملا منسوخ می‌کند. در چنین حالتی، اجرای دوباره روش بهینه‌سازی برای تولید جواب‌های بهینه و متناسب با شرایط محیطی جدید الزامی است. در نقطه مقابل، الگوریتم‌های محاسبات تکاملی برای تطابق جواب‌های بهینه با شرایط محیطی در حال تغییر، بسیار مناسب هستند.

روش‌های بهینه‌سازی تکاملی در طیف وسیعی از کاربردها (خصوصا کاربردهای جهان واقعی) قابل استفاده هستند. از الگوریتم‌های تکاملی می‌توان برای حل تمامی مسائلی استفاده کرد که در قالب مسائل «بهینه‌سازی تابعی» (Function Optimization) قابل تعریف باشند.

الگوریتم‌های تکاملی را می‌توان با دیگر روش‌های بهینه‌سازی سنتی ترکیب کرد.

الگوریتم‌های تکاملی می‌توانند برای حل مسائلی استفاده شوند که برای آن‌ها جوابی وجود ندارد. یکی از مزیت‌های مهم الگوریتم‌های تکاملی قابلیت به‌کارگیری آن‌ها در دامنه کاربردهایی است که خبره انسانی (دانشی که برای عملکرد صحیح سیستم مورد نیاز است) در آن‌ها وجود ندارد. این قابلیت الگوریتم‌های تکاملی در «روال‌های حل مسأله خودکار» (Automated Problem-Solving Routines) بیشتر خودنمایی می‌کند. محققان به این نتیجه رسیده‌اند که استفاده از خبره انسانی در چنین سناریوهایی لزوما منجر به تولید نتایج بهتر نخواهد شد؛ اگرچه در صورت وجود خبره یا نیاز به قرارگیری یک خبره در کنار سیستم خودکار طراحی شده، حتما باید از مزایای دانش انسانی برای حل مسأله استفاده کرد.

با در نظر گرفتن چنین ویژگی‌هایی، الگوریتم‌های تکاملی برای دامنه وسیعی از کاربردها قابل استفاده خواهند بود. مهم‌ترین این کاربردها عبارتند از:

عملیات و کنترل سیستم‌های نیرو (کنترل بهینه سیستم‌های برق و قدرت، زمان‌بندی مولدهای جریان برق و سایر موارد).

مسائل ترکیبی با پیچیدگی «ان‌پی سخت» (NP-Hard).

بهینه‌سازی استراتژی‌های زمان‌بندی کارها.

فرآیندهای شیمیایی (تشخیص ساختارهای شیمیایی، پیش‌بینی ساختار کریستال‌ها، طراحی داروها و سایر موارد).

مسیریابی خودروها.

مسائل بهینه‌سازی چند هدفه (Multi-Objective Optimization Problems).

مدل‌سازی پارامترهای بهینه.

مسائل «پردازش تصویر» (Image Processing) و «شناسایی الگو» (Pattern Recognition).

الگوریتم بهینه سازی فاخته نیز یکی از پیاده‌سازی‌های موفق از فرآیندهای طبیعی است. این الگوریتم از شیوه زندگی پرنده‌ای به نام فاخته الهام گرفته شده است. شیوه زندگی خاص این پرنده، روش تخم‌گذاری و رشد منحصر به فرد و در نهایت تولید مثل این

گونه، پایه و اساس این الگوریتم بهینه‌سازی را تشکیل می‌دهد. ویژگی شاخص این الگوریتم، شبیه‌سازی مفهوم بقا، مهاجرت برای یافتن منابع غذایی و انتخاب محیط بهینه برای زندگی است. جمعیت الگوریتم بهینه سازی فاخته را فاخته‌های بالغ و تخم‌های فاخته تشکیل می‌دهند.

فاخته‌ها و شیوه زندگی منحصر به فرد این گونه برای تولید مثل

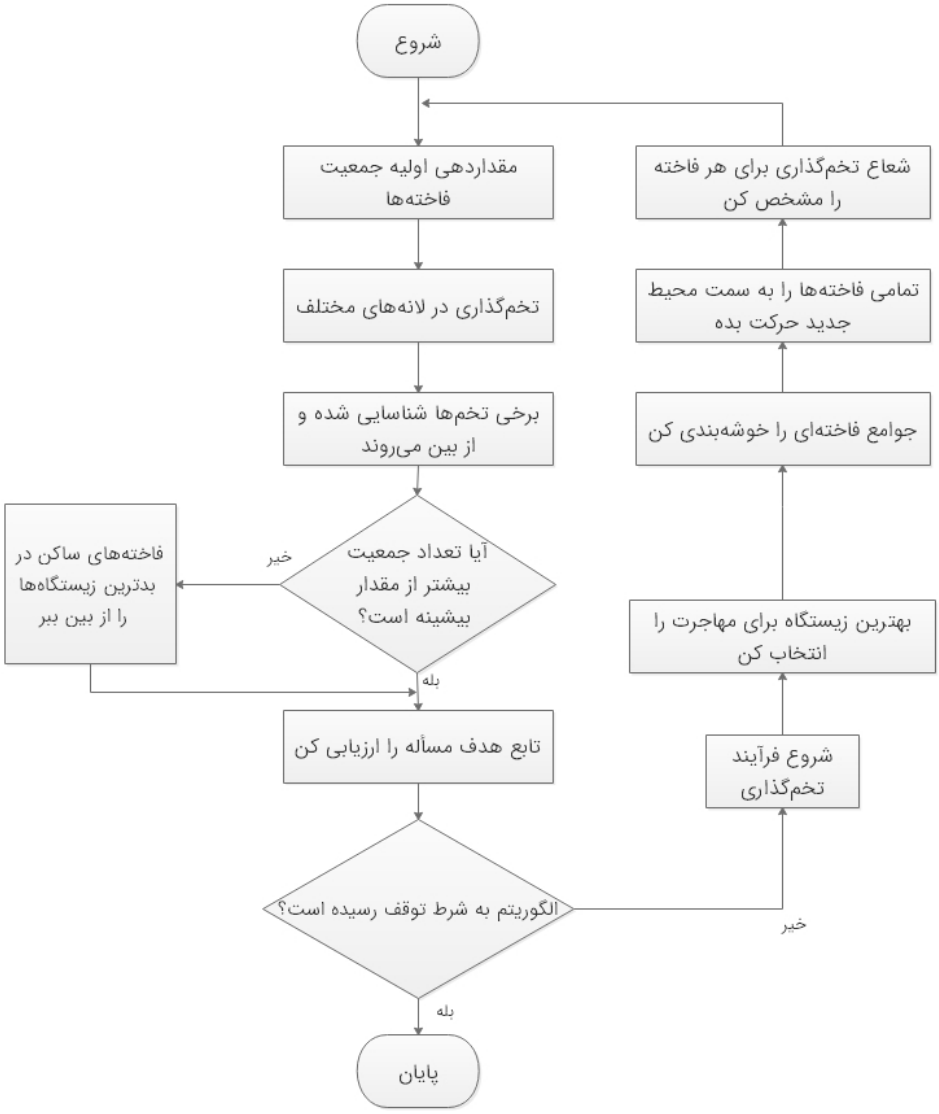
شیوه زندگی بیشتر گونه‌های پرندگان شبیه به هم است. پرنده ماده تخم‌گذاری می‌کند. از آنجا که تخم پرندگان حاوی مقادیر زیادی پروتئین و مواد غذایی است، بیشتر پرندگان مجبور هستند تا برای محافظت از تخم‌های گذاشته شده در برابر انواع مختلف شکارچیان، آن‌ها را در مکان‌های امن نگه‌داری کنند. پیدا کردن مکان مناسب برای مراقبت از تخم‌ها و بزرگ کردن آن‌ها، تا زمانی که به استقلال برسند، یکی از چالش برانگیزترین وظایف غریزی پرندگان گوناگون است. بیشتر پرندگان لانه خود را در میان انبوهی از برگ‌ها و شاخه‌های درختان می‌سازند و از تخم‌های خود در آن‌ها مراقبت می‌کنند.

در این میان، گونه خاصی از پرندگان وجود دارند که برای مراقبت از تخم‌های خود و بزرگ کردن آن‌ها به حیله‌گری متوسل می‌شوند. این دسته پرندگان، «انگل‌های جوجه‌گذاری» (Brood Parasite) نامیده می‌شوند. فاخته‌ها شناخته شده‌ترین گونه از این دسته پرندگان هستند. در ابتدا، فاخته ماده لانه‌ای برای مراقبت از تخم‌هایش پیدا می‌کند. سپس یکی از تخم‌های موجود در لانه پرنده میزبان را با یکی از تخم‌های خود جا به جا می‌کند و با تخم پرنده میزبان از آن منطقه فرار می‌کند؛ فرآیندی که بیش از ده ثانیه به طول نخواهد انجامید.

فاخته ماده لانه‌های مختلفی را زیر نظر قرار می‌دهد تا گونه‌ای از پرندگان را پیدا کند که رنگ و الگوی تخم‌های گذاشته شده توسط آن‌ها شباهت زیادی به تخم‌های خودش داشته باشد. این در حالی است که برخی از گونه‌های فاخته، تخم‌های خود را صرفا در لانه نوع خاصی از پرندگان میزبان قرار می‌دهند. این گونه فاخته‌ها یاد می‌گیرند تخم‌هایی بگذارند که رنگ و الگوی تخم‌های میزبان را با دقت بالایی همانندسازی کنند. با این حال، بسیاری از پرندگان میزبان نیز یاد می‌گیرند تا تخم‌های فاخته را از تخم‌های خود تشخیص دهند. در چنین حالتی، یا تخم‌های فاخته از لانه بیرون انداخته می‌شوند یا اینکه پرنده میزبان لانه خود را رها می‌کند و مجددا در مکان دیگری اقدام به لانه‌سازی می‌کنند. فاخته‌ها دائما در حال بهبود فرآیند همانندسازیتخم‌های خود با تخم‌های میزبان هستند. در نقطه مقابل، پرنده‌های میزبان دائما در حال پیدا کردن راه‌هایی برای شناسایی تخم‌های فاخته از تخم‌های خود هستند. برخی از گونه‌های فاخته مهاجر هستند. این گونه‌ها در فصل‌های خاصی از سال به سمت مقاصد مشخصی پرواز می‌کنند. این گونه فاخته‌ها معمولا در فصل‌های سرد سال به سمت مناطق گرم‌تر پرواز می‌کنند. قوه محرکه آن‌ها برای مهاجرت، پیدا کردن منابع غذایی و شرایط محیطی مناسب برای رشد و بقا است.

الگوریتم بهینه سازی فاخته

فلوچارت الگوریتم بهینه سازی فاخته در شکل زیر نمایش داده شده است. همانند دیگر الگوریتم‌های تکاملی، الگوریتم بهینه سازی فاخته کار خود را با تولید جمعیت اولیه از فاخته‌ها آغاز می‌کند. جمعیت اولیه فاخته‌ها در لانه پرندگان میزبان تخم‌گذاری می‌کنند. تخم‌هایی که بیشترین شباهت را به تخم‌های پرنده میزبان دارند، فرصت رشد و تبدیل شدن به فاخته‌های بالغ را خواهند داشت. سایر تخم‌ها توسط میزبان شناسایی می‌شوند و متعاقبا از بین می‌روند.



فاخته‌ها شرایط محیط زندگی را برای بقا ارزیابی می‌کنند. محیطی که بیشترین تعداد تخم‌های فاخته در آن رشد کنند و به فاخته بالغ تبدیل شوند، بیشترین «سود» (Profit) را به ارمغان خواهد آورد. به عبارت دیگر، فاخته در محیطی تخم‌گذاری می‌کند که نرخ بقای تخم‌ها بیشینه باشد. تخم‌هایی که زنده می‌مانند و به فاخته بالغ تبدیل می‌شوند، در محل زندگی خود جوامعی متشکل از تعدادی فاخته تشکیل می‌دهند. هر جامعه متشکل از فاخته‌ها در یک زیستگاه خاص از محیط زندگی می‌کنند.

بهترین زیستگاه، یا زیستگاهی که بیشترین منابع غذایی و شانس زنده ماندن برای فاخته‌ها را داشته باشد، به عنوان مقصد مهاجرتی دیگر جوامع انتخاب می‌شود. جوامع مهاجرت کننده، نزدیک بهترین زیستگاه سکنی می‌گزینند. با توجه به تعداد تخم‌های هر فاخته و فاصله فاخته تا بهترین زیستگاه، یک «شعاع تخم‌گذاری» (Egg Laying Area) برای هر فاخته تعیین می‌شود.

سپس، هر کدام از فاخته‌ها به طور تصادفی در لانه‌های موجود درون این شعاع، تخم‌گذاری می‌کنند. این کار تا زمانی ادامه پیدا می‌کند که بهترین نقطه با بیشترین سود (بیشترین منابع غذایی و بالاترین شانس زنده ماندن) شناسایی شود؛ بیشتر فاخته‌ها اطراف این نقطه همگرا می‌شوند. در ادامه، مراحل الگوریتم بهینه سازی فاخته توضیح داده خواهند شد.

تولید زیستگاه‌های اولیه فاخته‌ها

برای حل یک مسأله بهینه‌سازی، در مرحله اول باید مقادیر متغیرهای مسأله را در قالب یک آرایه نمایش داد. در الگوریتم بهینه سازی فاخته، به آرایه نمایش دهنده مقادیر متغیرهای مسأله، «زیستگاه» (Habitat) گفته می‌شود. هر زیستگاه، یک نمونه یا یک جواب کاندید برای مسأله بهینه‌سازی مدنظر خواهد بود. جواب‌های کاندید طی یک فرایند تکاملی به جواب بهینه سراسری همگرا می‌شوند. در یک مسأله بهینه‌سازی NVar بُعدی، زیستگاه، آرایه‌ای به ابعاد 1×(Nvar)1×(Nvar) است. این آرایه مکان کنونی فاخته در محیط را نشان می‌دهد. آرایه مذکور به شکل زیر تعریف می‌شود:

habitat=[x1,x2,x3,.....xNvar]habitat=[x1,x2,x3,.....xNvar]

مقادیر متغیرهای (x1,x2,x3,.....xNvar)(x1,x2,x3,.....xNvar) اعشاری هستند. میزان سود یک زیستگاه، از طریق ارزیابی تابع سود fpfp در زیستگاه متشکل از مقادیر (x1,x2,x3,.....xNvar)(x1,x2,x3,.....xNvar) مشخص می‌شود. بنابراین تابع سود برای یک مسأله بهینه‌سازی داده شده به شکل زیر تعریف می‌شود.

Profit=fp(habitat)=fp(x1,x2,x3,.....xNvar)Profit=fp(habitat)=fp(x1,x2,x3,.....xNvar)

چنانکه در رابطه بالا مشهود است، الگوریتم فاخته تابع سود را بیشینه می‌کند. در صورتی که هدف کمینه‌سازی یک تابع هزینه باشد، به راحتی می‌توان از طریق بیشینه‌سازی تابع سود زیر، تابع هزینه را کمینه کرد:

Profit=−Cost(habitat)=−fc(x1,x2,x3,.....xNvar)Profit=−Cost(habitat)=−fc(x1,x2,x3,.....xNvar)

فرآیند تکاملی الگوریتم بهینه سازی فاخته به این صورت است که در ابتدا یک ماتریس با اندازه NPopulation×NVarNPopulation×NVar از زیستگاه‌های کاندید تولید می‌شود. سپس برای هر زیستگاه تولید شده، تعدادی تصادفی تخم فاخته در نظر گرفته می‌شود. در طبیعت، هر فاخته به طور میانگین بین ۵ تا ۲۰ عدد تخم می‌گذارد. این اعداد، مقادیر کمینه و بیشینه (حد بالا و پایین) تعداد تخم‌های مجاز (مقادیر متغیرها) در هر زیستگاه را تشکیل می‌دهند.

عادت دیگر فاخته‌ها در جهان واقعی این است که معمولا در فاصله بیشینه از زیستگاه واقعی زندگی خود تخم‌گذاری می‌کنند. در الگوریتم فاخته، به این فاصله بیشینه، شعاع تخم‌گذاری گفته می‌شود. در یک مسأله بهینه‌سازی، شعاع تخم‌گذاری برای هر فاخته متناسب با تعداد کلی تخم‌ها، حد پایین (VarHighVarHigh) و حد بالای (VarLowVarLow) مقادیر متغیرها محاسبه می‌شود. بنابراین، شعاع تخم‌گذاری به شکل زیر تعریف می‌شود:

ELR=α×No.of.Current.Cuckoo’s.EggsTotal.Number.of.Eggs.×(VarHigh−VarLow)ELR=α×No.of.Current.Cuckoo’s.EggsTotal.Number.of.Eggs.×(VarHigh−VarLow)

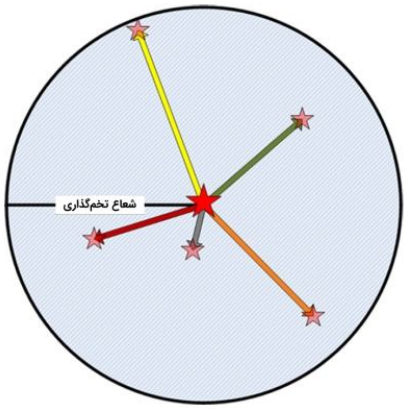
در این رابطه، پارامتر αα عددی صحیح است که مقدار بیشینه شعاع تخم‌گذاری را کنترل می‌کند.

با توجه به بکارگیری الگوریتم‌های فراابتکاری در مباحثی همچون یادگیری ماشین، مهندسی کنترل و ...، «فرادرس» اقدام به انتشار فیلم آموزش الگوریتم بهینه سازی فاخته و پیاده سازی آن در MATLAB کرده که در ادامه متن به آن اشاره شده است.

برای دیدن فیلم آموزش الگوریتم بهینه سازی فاخته و پیاده سازی آن در MATLAB + اینجا کلیک کنید.

روش تخم‌گذاری فاخته‌ها

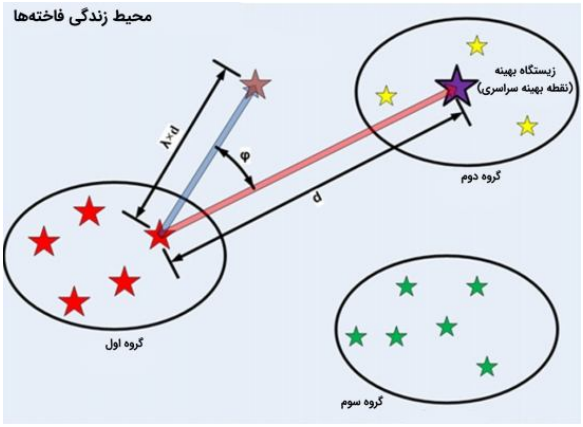
در مرحله بعد، هر فاخته در شعاع تخم‌گذاری خود و به‌طور تصادفی، در لانه پرنده‌های میزبان تخم‌گذاری می‌کند. فرآیند تخم‌گذاری فاخته‌ها در شکل زیر نمایش داده شده است.



پس از پایان فرآیند تخم‌گذاری، تخم‌هایی که شباهت کمتری به تخم‌های پرنده میزبان دارند، توسط پرنده میزبان شناسایی شده و دور انداخته می‌شوند. بنابراین، پس از هر فرایند تخم‌گذاری ρ%ρ% از تخم‌های فاخته از بین می‌روند. این مقدار معمولا برابر ده درصد تخم‌های گذاشته شده است و تخم‌هایی را شامل می‌شود که در محیط‌های حاوی منابع غذایی کم قرار دارند و در نتیجه توانایی همگرایی به جواب بهینه را ندارند. سایر تخم‌ها در لانه پرنده میزبان رشد کرده، سر از تخم بیرون آورده و توسط پرنده میزبان تغذیه می‌شوند. نکته بسیار جالب دیگر در مورد تخم‌گذاری و رشد فاخته‌ها این است که تنها یک تخم در لانه پرنده میزبان رشد می‌کند. وقتی که جوجه فاخته زودتر از جوجه‌های پرنده میزبان سر از تخم بیرون می‌آورد، تخم‌های پرنده میزبان را از لانه بیرون می‌اندازد. شایان توجه است که جوجه فاخته سه برابر بزرگ‌تر از جوجه‌های پرنده میزبان است. بنابراین، حتی در حالتی که جوجه‌های پرنده میزبان زودتر از جوجه فاخته سر از تخم بیرون آورند، جوجه فاخته پس از تولد بیشتر منابع غذایی را مصرف می‌کند. در نتیجه، پس از چند روز جوجه‌های پرنده میزبان از گرسنگی تلف می‌شوند.

مهاجرت فاخته‌ها

جوجه‌های فاخته پس از رشد کردن و بالغ شدن، تا مدتی در زیستگاه و جامعه خود به زندگی ادامه می‌دهند. با این حال، زمانی که دوره تخم‌گذاری آن‌ها آغاز می‌شود به مناطقی مهاجرت می‌کنند که از یک سو منابع غذایی بیشتری در آنجا وجود داشته باشد و از سوی دیگر رنگ و الگوی تخم‌های آن‌ها شباهت زیادی به تخم‌های پرنده میزبان در آن منطقه داشته باشد. جامعه فاخته‌ای که بیشترین سود (بیشترین منابع غذایی و بهترین شرایط محیطی برای زندگی) را داشته باشد، به عنوان مقصد مهاجرت فاخته‌ها در دیگر جوامع انتخاب می‌شود. از آنجا که فاخته‌های بالغ در زیستگاه و محیط زندگی خود پراکنده شده‌اند، به سختی می‌توان تشخیص داد که هر فاخته به کدام جامعه فاخته‌ای تعلق دارد. برای حل این مشکل، جمعیت فاخته‌ای موجود در محیط، توسط الگوریتم خوشه‌بندی K-means گروه‌بندی می‌شوند (در شبیه‌سازی‌های انجام شده، مناسب‌ترین مقدار پارامتر k برای الگوریتم K-means، بین ۳ تا ۵ است). پس از گروه‌بندی جمعیت فاخته‌ای، «سود میانگین» (Mean Profit) گروه‌های فاخته‌ای ایجاد شده محاسبه می‌شود. بهترین زیستگاه موجود در گروه فاخته‌ای که بیشترین سود میانگین را دارد، به عنوان زیستگاه هدف و مقصد مهاجرتی دیگر گروه‌های فاخته‌ای انتخاب می‌شود. فاخته‌ها، تمام مسیر را به سمت زیستگاه مقصد پرواز نمی‌کنند. بلکه بخشی از مسیر را با درجه انحراف خاصی از مقصد حرکت می‌کنند. نحوه حرکت فاخته‌ها به سمت زیستگاه مقصد در شکل زیر آمده است.



همانطور که در شکل بالا مشهود است، هر فاخته تنها λ%λ% از مسیر را با درجه انحراف φφ رادیان پرواز می‌کنند. پارامترهای λλ و φφ به فاخته‌ها کمک می‌کنند تا مناطق بیشتری را برای یافتن زیستگاه بهینه با بیشترین منابع غذایی و بهترین شرایط زندگی جستجو کنند. برای هر فاخته، مقادیر λλ و φφ به شکل زیر تعریف می‌شوند.

λ∼U(0,1)λ∼U(0,1)

φ∼U(−ω,ω)φ∼U(−ω,ω)

در این رابطه، پارامتر λλ یک عدد تصادفی بین ۰ و ۱ است که از توزیع یکنواخت تبعیت می‌کند. پارامتر ωω نیز مقدار درجه انحراف از زیستگاه هدف را مشخص می‌کند (برای هم‌گرایی بهینه جمعیت فاخته‌ها به بهینه سراسری، مقدار π6π6 برای پارامتر ωω مناسب است).

تعادل جمعیت فاخته‌ها در محیط

با توجه به اینکه همیشه باید تعادل در جمعیت فاخته‌ها حفظ شود، پارامتر کنترلی به نام NMaxNMax تعریف می‌شود. این پارامتر وظیفه دارد تا تعداد فاخته‌های موجود در محیط را کنترل و محدود کند. به دلیل کمبود منابع غذایی در محیط، از بین رفتن فاخته‌ها به دست شکارچیان و بعضا نبود لانه مناسب برای رشد جوجه‌ها، وجود پارامتر کنترلی NMaxNMax برای کنترل جمعیت فاخته در الگوریتم بهینه سازی فاخته ضروری است. در الگوریتم بهینه سازی فاخته، تنها تعداد NMaxNMax فاخته (که بیشترین مقدار سود را تولید می‌کنند) در محیط زنده می‌مانند.

همگرایی به جواب بهینه

بعد از تعدادی تکرار در الگوریتم بهینه سازی فاخته، تمامی جمعیت فاخته‌ای به بهترین زیستگاه موجود در محیط همگرا می‌شوند. در این زیستگاه، تخم‌های فاخته بیشترین شباهت را به تخم پرندگان میزبان خواهند داشت و بیشترین منابع غذایی محیط در این ناحیه یافت خواهند شد. به عبارت دیگر، زیستگاه بهینه، بیشترین مقدار سود را تولید خواهد کرد. وقتی که بیش از ۹۵ درصد جمعیت فاخته‌ها به یک زیستگاه همگرا شوند، فرآیند عملیاتی الگوریتم بهینه سازی فاخته به پایان خواهد رسید. در ادامه، شبه کد الگوریتم بهینه سازی فاخته نمایش داده شده است.

شبه کد الگوریتم بهینه سازی فاخته

جمعیت فاخته‌ها از طریق مقداردهی تصادفی به متغیرهای تابع هدف مسأله بهینه‌سازی مورد نظر مقدار دهی اولیه می‌شود.

تعدادی تخم فاخته به هرکدام از فاخته‌ها اختصاص داده می‌شود.

برای هر فاخته، شعاع تخم‌گذاری مشخص می‌شود.

هر فاخته در محدوده شعاع تخم‌گذاری مشخص شده اقدام به تخم‌گذاری می‌کند.

تخم‌های فاخته‌ای که توسط پرنده میزبان شناسایی شوند (رنگ و الگوی این تخم‌ها با تخم‌های پرنده میزبان تفاوت دارد) از بین می‌روند. تخم‌های فاخته باقی مانده، سر از تخم بیرون می‌آورند و رشد می‌کنند (بالغ می‌شوند).

زیستگاه هر کدام از فاخته‌های بالغ ارزیابی می‌شود (میزان سود مشخص می‌شود).
با توجه به پارامتر NMaxNMax ، جمعیت فاخته‌ای موجود در محیط کنترل می‌شود و فاخته‌هایی که در بدترین زیستگاه‌ها زندگی می‌کنند از بین می‌روند. جمعیت فاخته‌ای خوشه‌بندی، سود میانگین گروه‌های فاخته‌ای محاسبه و بهترین زیستگاه به عنوان مقصد مهاجرتی دیگر گروه‌ها انتخاب می‌شود. جمعیت فاخته‌ای به سمت زیستگاه بهینه انتخاب شده مهاجرت می‌کنند. در صورتی که جمعیت فاخته‌ها به نقطه بهینه سراسری همگرا و یا بیش از ۹۵ درصد جمعیت فاخته‌ها به یک زیستگاه خاص همگرا شوند، اجرای الگوریتم متوقف می‌شود؛ در غیر این صورت، اجرای الگوریتم به گام ۲ منتقل شده و الگوریتم دوباره اجرا می‌شود. ویژگی مهم الگوریتم بهینه سازی فاخته، طبیعت تصادفی و احتمالی آن است. نقطه قوت الگوریتم‌های بهینه‌سازی تصادفی در طبیعت احتمالی آن‌ها نهفته است. ویژگی مذکور سبب می‌شود که این دسته الگوریتم‌ها کمتر در دام «بهینه محلی» (Local Optimum) قرار بگیرند. همچنین، برخلاف دیگر الگوریتم‌های بهینه‌سازی (خصوصا ریاضی)، داشتن اطلاعات اضافی در مورد مشتق تابع هدف الزامی نیست.

ارزیابی عملکرد الگوریتم بهینه سازی فاخته روی توابع هدف معیار

در این بخش، الگوریتم بهینه سازی فاخته روی دو تابع هدف معیار ارزیابی می‌شود. هدف از این آزمایش، ارزیابی عملکرد این الگوریتم در همگرایی به جواب بهینه سراسری و همچنین مشخص کردن تعداد تکرارهای لازم برای رسیدن به جواب بهینه سراسری است. توابع هدف استفاده شده با هدف ارزیابی عملکرد الگوریتم‌های بهینه‌سازی در پیدا کردن مقادیر بهینه متغیرها جهت کمینه‌سازی طراحی شده‌اند. در ادامه توابع هدف معیار استفاده شده در این آزمایش تعریف می‌شوند. در تمامی حالات ارزیابی الگوریتم بهینه سازی فاخته، تعداد متغیرهای مسأله برابر ۱۰۰ در نظر گرفته شده است. حد بالا و پایین برای مقادیر ممکن متغیرهای مسأله، به ترتیب برابر با ۵ و ۵- است. تعداد اولیه فاخته‌ها برابر با ۲۰ در نظر گرفته شده است. در هر تکرار از الگوریتم بهینه سازی فاخته، هر فاخته می‌تواند بین ۵ تا ۱۰ عدد تخم بگذارد.

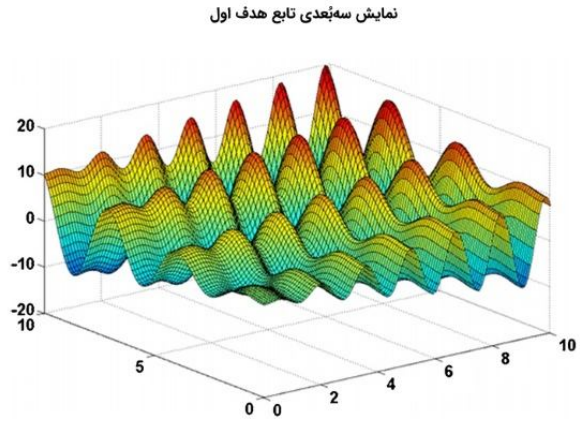
تابع هدف اول

مشخصات و قیدهای عددی تابع هدف اول به شرح زیر است:

$$f=x+\sin(4x)+1.1y+\sin(2y)f=x+\sin\ (4x)+1.1y+\sin\ (2y)$$

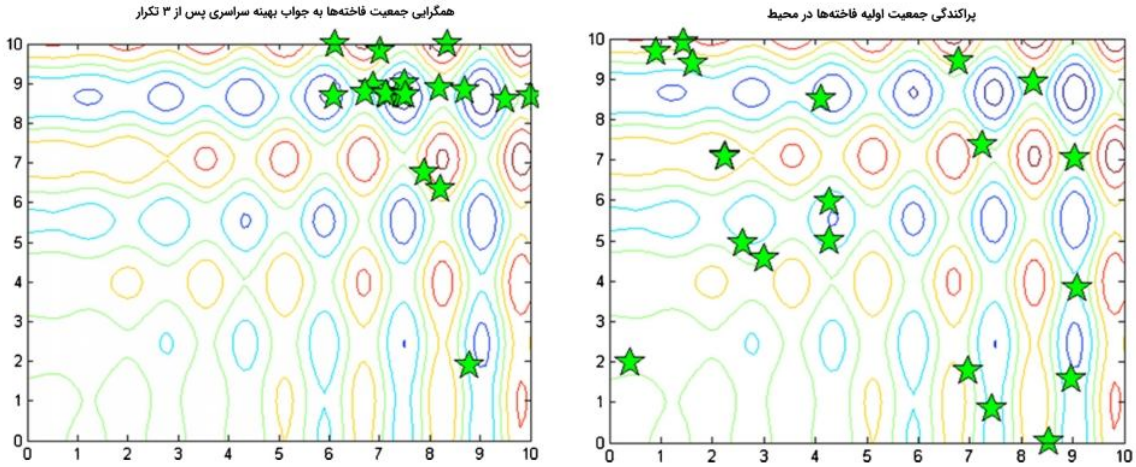
$$0<x,y<0,\min:f(9.039,8.668)=-8.55470<x,y<0,\min:f(9.039,8.668)=-8.5547$$

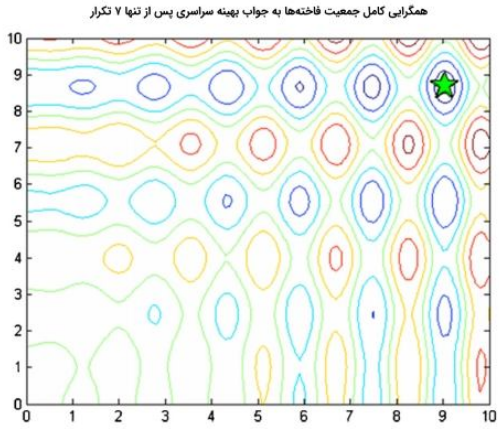
نمایش سه‌بُعدی تابع هدف اول در شکل زیر آمده است.



شکل‌های زیر، نتیجه اجرای الگوریتم بهینه‌ساری فاخته برای بهینه‌سازی تابع هدف اول را نشان می‌دهد. همانطور که در شکل مشهود است، همگرایی تنها طی ۷ تکرار حاصل شده است. تمامی فاخته‌ها به جواب بهینه سراسری همگرا شده‌اند. در تکرارهای ابتدایی الگوریتم، جمعیت فاخته‌ها به سمت زیستگاه حاوی جواب بهینه سراسری مهاجرت می‌کنند. سپس در تکرارهای آتی، فاخته‌ها هر چه بیشتر به زیستگاه حاوی جواب بهینه سراسری نزدیک و نزدیک‌تر می‌شوند. در نهایت، در تکرار هفتم، تقریبا تمام

فاخته‌ها به جواب بهینه سراسری همگرا می‌شوند. در چنین حالتی، الگوریتم به شرط توقف مشخص شده می‌رسد و به کار خود پایان می‌دهد. شکل اول، پراکندگی جمعیت ابتدایی فاخته‌ها در محیط را نشان می‌دهد. شکل دوم، نزدیک شدن جمعیت فاخته‌ها به جواب بهینه سراسری پس از ۳ تکرار را نشان می‌دهد. شکل سوم، همگرایی کامل جمعیت فاخته‌ها را به جواب بهینه سراسری پس از ۷ تکرار نشان می‌دهد.



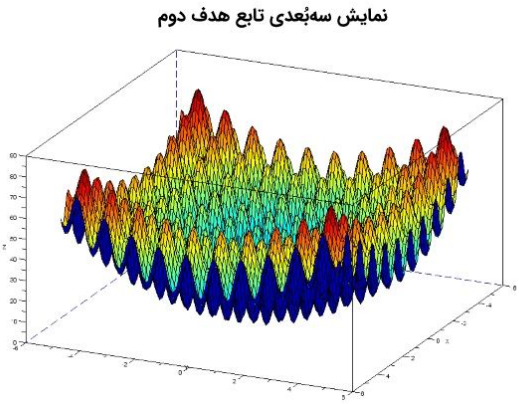


تابع هدف دوم

تابع هدف دوم، یک تابع هدف n-بُعدی است که با هدف کمینه‌سازی طراحی شده است و به آن تابع هدف Rastrigin گفته می‌شود. مشخصات و قیدهای عددی تابع هدف Rastrigin به شرح زیر است:

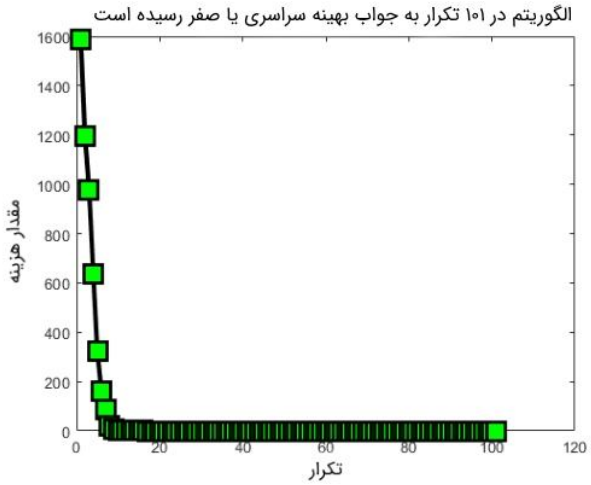
$$f=10n+n\sum i=1(x2i-10\cos(2\pi xi)),n=9f=10n+\sum i=1n(xi2-10\cos\ (2\pi xi)),n=9$$
$$-5.12\leq xi\leq 5.12,f(0,0,....,0)=0-5.12\leq xi\leq 5.12,f(0,0,....,0)=0$$

نمایش سه‌بُعدی تابع هدف دوم در شکل زیر آمده است.



همانطور که در شکل مشاهده می‌شود، تابع هدف Rastrigin یکی از چالش‌برانگیزترین توابع هدف برای مقاصد بهینه‌سازی است. این تابع، تعداد زیادی بهینه محلی دارد و احتمال قرار گرفتن الگوریتم بهینه‌سازی در یکی از بهینه‌های محلی بسیار زیاد است. بنابراین، رسیدن به بهینه سراسری برای بسیاری از الگوریتم‌های بهینه‌سازی بسیار سخت خواهد بود.

شکل زیر نحوه همگرایی الگوریتم بهینه سازی فاخته روی تابع هدف دوم را نشان می‌دهد. همانطور که در شکل مشهود است، در چند تکرار اول، جمعیت فاخته به بهترین زیستگاه موجود در محیط نزدیک می‌شوند. چنین پدیده‌ای، قدرت و سرعت بالای الگوریتم بهینه سازی فاخته در همگرایی به ناحیه حاوی جواب بهینه سراسری را نشان می‌دهد. در نهایت، در تکرار ۱۰۱اُم، الگوریتم بهینه سازی فاخته به جواب بهینه سراسری (مقدار هزینه صفر) همگرا می‌شود.



نتایج نشان داده شده از ارزیابی الگوریتم بهینه سازی فاخته روی توابع هدف معیار، نشان دهنده عملکرد بالای الگوریتم در همگرایی به جواب بهینه سراسری است. همچنین، این الگوریتم در تعداد تکرارهای معدود قادر است تا تقریب بسیار خوبی از جواب بهینه واقعی ارائه دهد. کد الگوریتم بهینه سازی فاخته و توابع هدف معیار در زبان برنامه‌نویسی «متلب» (MATLAB) در ادامه آمده است.

جمع‌بندی

الگوریتم بهینه سازی فاخته، الگوریتمی برای حل مسائل بهینه‌سازی پیوسته و غیر خطی است. این الگوریتم از رفتار گونه خاصی از پرندگان به نام فاخته الهام گرفته شده است. به طور خاص، از ویژگی‌های منحصر به فرد این پرنده در تخم‌گذاری و تولید مثل برای پیاده‌سازی الگوریتم بهینه سازی فاخته استفاده شده است. هر فاخته، زیستگاهی برای زندگی دارد. فاخته‌ها در زیستگاه خود شروع به تخم‌گذاری می‌کنند. تخم‌هایی که از بین نمی‌روند، رشد کرده و بالغ می‌شوند. فاخته برای تولید مثل به مناطقی مهاجرت می‌کند که بیشترین منابع غذایی و بهترین شرایط زندگی را داشته باشد. فاخته‌ها در اطراف زیستگاه بهینه یافت شده اقدام به

تخم‌گذاری می‌کنند. این رویه سبب می‌شود تا محیط بیشتری از زیستگاه بهینه برای یافتن جواب بهینه سراسری جستجو شود. پس از تعدادی چرخه مهاجرتی، بیشتر جمعیت فاخته‌ها به جواب بهینه سراسری مسأله بهینه‌سازی همگرا می‌شوند. نتایج ارزیابی نشان می‌دهد که این الگوریتم، سرعت و دقت بالایی در همگرایی به جواب بهینه توابع هدف معیار دارد. حتی در مواردی که تابع هدف تعداد زیادی جواب بهینه محلی دارد، این الگوریتم قادر است در تعداد کمی تکرار، به تقریب نزدیکی از جواب بهینه سراسری همگرا شود. الگوریتم بهینه سازی فاخته را می‌توان به عنوان پیاده‌سازی موفق از فرآیند الهام گرفته شده از طبیعت در نظر گرفت.

بسیاری از اختراعات بشری از طبیعت الهام گرفته شده‌اند. «شبکه‌های عصبی مصنوعی» (ANN | Artificial Neural Network) نمونه بارز چنین ابداعاتی هستند. یکی دیگر از چنین ابداعاتی، توسعه ایده الگوریتم ژنتیک است. الگوریتم‌های ژنتیک، با «شبیه‌سازی» (Simulating) فرایند تکامل در طبیعت، با هدف یافتن بهترین جواب ممکن برای یک مسأله، به جستجو در «فضای جواب‌های کاندید» (Candidate Solution Space) می‌پردازند. در فرایند جستجو برای یافتن جواب بهینه، ابتدا مجموعه یا جمعیتی از جواب‌های ابتدایی تولید می‌شود. سپس، در «نسل‌های» (Generations) متوالی، مجموعه‌ای از جواب‌های تغییر یافته تولید می‌شوند (در هر نسل از الگوریتم ژنتیک، تغییرات خاصی در ژن‌های کروموزوم‌های تشکیل دهنده جمعیت ایجاد می‌شود). جواب‌های اولیه معمولا به شکلی تغییر می‌کنند که در هر نسل، جمعیت جواب‌ها به سمت جواب بهینه «همگرا» (Converge) می‌شوند. این شاخه از حوزه «هوش مصنوعی» (Artificial Intelligence)، بر پایه مکانیزم تکامل موجودات زنده و تولید گونه‌های موفق‌تر و برازنده‌تر در طبیعت الهام گرفته شده است. به عبارت دیگر، ایده اصلی الگوریتم‌های ژنتیک، «بقای برازنده‌ترین‌ها» (Survival of the Fittest) است. یک کروموزوم، رشته‌ای بلند و پیچیده از «اسید دی‌اکسی ریبونوکلئیک» (Deoxyribonucleic Acid) یا DNA است. عوامل ارثی که ویژگی‌ها یا خصیصه‌های یک «فرد» (Individual) را مشخص می‌کنند، در طول این کروموزوم‌ها نقش یافته‌اند. هر یک از خصیصه‌های موجود در افراد، به وسیله ترکیبی از DNA، در ژن‌های انسان کدبندی می‌شوند. در بدن موجودات زنده، معمولا چهار «پایه» (Base) برای تولید کروموزوم‌ها از روی DNA وجود دارد:

پایه A یا Adenine

پایه C یا Cytosine

پایه G یا Thymine

پایه T یا Guanine

همانطور که الفبا، واحدهای سازنده یک «زبان» (Language) را تعریف می‌کند، ترکیب با معنی از کروموزوم‌ها (و پایه‌های آن‌ها)، دستورالعمل‌های خاصی را برای سلول‌ها تولید می‌کند. تغییرات در کوروموزوم‌ها، طی فرایند تولید مثل اتفاق می‌افتد.

کروموزوم‌های «والدین» (Parents) از طریق فرایند خاصی به نام «ترکیب یا آمیزش» (Crossover)، به طور تصادفی با یکدیگر مبادله می‌شوند. بنابراین، فرزندان برخی از ویژگی‌ها یا خصیصه‌های پدر و برخی از ویژگی یا خصیصه‌های مادر را به ارث می‌برند و از خود به نمایش می‌گذارند.

فرایند نادری به نام «جهش» (Mutation) نیز سبب ایجاد تغییراتی در ویژگی‌ها یا خصیصه‌های موجودات زنده می‌شود. برخی از مواقع، ممکن است خطایی در فرایند کپی کردن کروموزوم‌ها رخ دهد که به آن «میتوز» (Mitosis) نیز گفته می‌شود. غالب جهش‌هایی که در موجودات زنده اتفاق می‌افتد، منجر به از بین رفتن موجودات خواهد شد؛ با این حال، در یک دوره چند میلیون ساله از تکامل، جهش ممکن است منجر به ایجاد گونه‌های جدید و برازنده‌تر از موجودات زنده شود.

انتخاب طبیعی

در طبیعت، موجوداتی که ویژگی‌های برازنده‌تری نسبت به دیگر گونه‌ها دارند، برای مدت بیشتری به بقاء در طبیعت ادامه می‌دهند. چنین ویژگی‌ای، این امکان را در اختیار برازنده‌ترین موجودات زنده قرار می‌دهد تا بر اساس مواد ژنتیکی خود، اقدام به تولید مثل کنند. بنابراین، پس از یک دوره زمانی بلند مدت، جمعیت موجودات زنده به سمتی تکامل پیدا خواهد کرد که در آن، غالب موجودات بسیاری از ویژگی‌های ارثی خود را از «ژن‌های» (Genes) موجودات برتر و تعداد کمی از ویژگی‌های خود را از ژن‌های موجودات «رده پایین» (Inferior) با ژن‌ها یا ویژگی‌های نامرغوب به ارث خواهند برد.

به بیان ساده‌تر، موجودات برازنده‌تر زنده می‌مانند و موجودات نامناسب از بین می‌روند. به این فرایند و نیروی شگفت‌انگیز طبیعی، «انتخاب طبیعی» (Natural Selection) گفته می‌شود. نکته مهم در مورد انتخاب طبیعی و اثبات درست بودن این اصل این است که تحقیقات دانشمندان در مورد «توضیحات مولکولی از تکامل» (Molecular Explanation of Evolution) نشان داده است که گونه‌های مختلف موجودات زنده، خود را با شرایط محیطی تطبیق نمی‌دهند، بلکه صرفا موجودات برازنده‌تر به بقاء خود ادامه می‌دهند.

با توجه به اهمیت روش‌های بهینه‌سازی هوشمند و الگوریتم‌های تکاملی، «فرادرس» اقدام به انتشار فیلم آموزش تئوری و عملی الگوریتم ژنتیک در قالب آموزشی ۱۴ ساعت و ۲۳ دقیقه‌ای کرده که در ادامه متن به آن اشاره شده است.

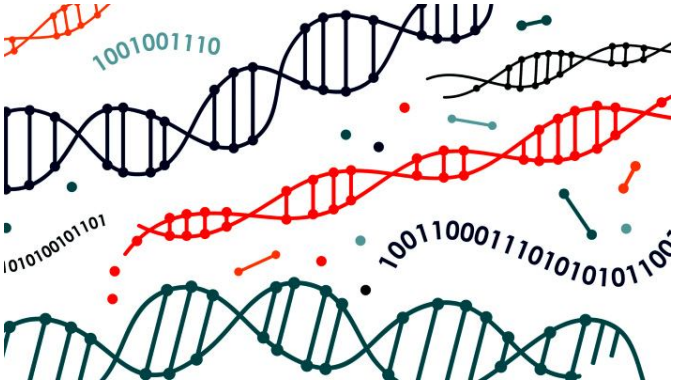
برای دیدن فیلم آموزش تئوری و عملی الگوریتم ژنتیک + اینجا کلیک کنید.

تکامل شبیه‌سازی شده

برای شبیه‌سازی فرایند انتخاب طبیعی توسط سیستم‌های کامپیوتری و حل مسأله با استفاده از الگوریتم‌های الهام گرفته شده از انتخاب طبیعی، ابتدا باید مدل‌های نمایشی جهت مدل‌سازی متغیرهای مسأله تعریف شوند:

نمایشی از یک «موجودیت» (Individual) در هر نقطه از فضای جستجوی مسأله در طول فرایند جستجو برای یافتن جواب بهینه: برای چنین کاری، مفهوم «نسل‌های» (Generation) متوالی از موجودیت‌ها مطرح می‌شود. هر موجودیت یک ساختار داده‌ای خواهد بود که «ساختار ژنتیکی» (Genetic Structure) یک جواب یا فرضیه محتمل/کاندید را نمایش می‌دهد. همانند یک کروموزوم، ساختار ژنتیکی یک موجودیت توسط الفبایی ثابت و محدود توصیف خواهد شد. به عنوان نمونه، الگوریتم ژنتیک از الفبای مبتنی بر رشته‌های «باینری» (Binary)، «مقادیر صحیح» (Integer Values) یا «مقادیر حقیقی» (Real Values) به عنوان تفسیری از جواب‌های محتمل برای یک مسأله خاص استفاده می‌کند.

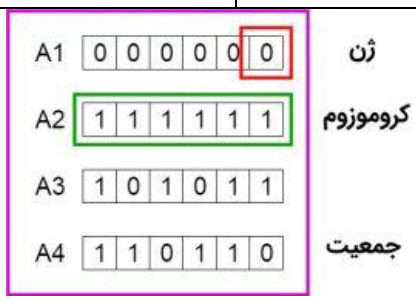
به عنوان نمونه، مسأله «فروشنده دوره‌گرد» (Travelling Salesman) یا TSP را در نظر بگیرید. مسأله فروشنده دورگرد، مسأله پیدا کردن مسیر بهینه برای پیمایش مثلا ۱۰ شهر است. فروشنده می‌تواند مسیر پیمایشی خود را از هر شهری شروع کند. بنابراین، جواب‌های این مسأله «جایگشتی» (Permutation) از شهرهای پیمایش شده خواهد بود: ۱-۳-۲-۵-۴-۸-۱۰-۹-۷-۶-۸.



فرهنگ لغات الگوریتم ژنتیک

در ادامه، برخی از اصطلاحات و واژگان کلیدی در توصیف فرایندهای موجود در الگوریتم ژنتیک نمایش داده شده است. نکته مهم در مورد فرهنگ لغات الگوریتم ژنتیک این است که اصطلاحات و واژگان کلیدی مترادف و معادل یکدیگر، در زمینه‌های موضوعی مرتبط با الگوریتم ژنتیک، غالبا به جای یکدیگر مورد استفاده قرار می‌گیرند.

الگوریتم ژنتیک	توضیح
کروموزوم (رشته، موجودیت)	جواب مسأله
ژن‌ها (بیت‌ها)	بخشی از جواب مسأله
مکان (Locus)	مکان ژن‌ها
آلل‌ها (Alleles)	مقادیر ژن‌ها
فنوتایپ (Phenotype)	جواب کدگشایی شده مسأله
ژنوتایپ (Genotype)	جواب کدبندی شده مسأله



الگوریتم ژنتیک متعارف

در ادامه، با مؤلفه‌های اساسی الگوریتم‌های ژنتیک و نحوه عملکرد آن‌ها در پیدا کردن جواب بهینه یک مسأله خاص مورد بررسی قرار می‌گیرد.

مفاهیم مهم ابتدایی در الگوریتم ژنتیک

الگوریتم‌های ژنتیک، الگوریتم‌های جستجو هستند که بر پایه مفاهیم انتخاب طبیعی و ژنتیک موجودات زنده بنا نهاده شده‌اند. الگوریتم‌های ژنتیک پدیده آمده‌اند تا برخی از فرایندهای مشاهده شده در «تکامل طبیعی» (Natural Evolution) را از طریق الگوریتم‌های کامپیوتری شبیه‌سازی کنند. فرایندهایی که بر اساس انجام عملیات روی کروموزوم‌ها (سیستم‌های ارگانیک جهت کدبندی کردن ساختار ژنتیکی موجودات زنده) شکل گرفته‌اند.

همانطور که پیش از این نیز اشاره شد، الگوریتم‌های ژنتیک در زیر مجموعه الگوریتم‌های جستجو قرار می‌گیرند. با این حال، تفاوت‌های بسیار اساسی با دیگر الگوریتم‌های جستجو دارند. الگوریتم‌های ژنتیک به جای اینکه به طور مستقیم با مقادیر

پارامترهای مسأله سروکار داشته باشند، با نمایشی کدبندی شده از مجموعه پارامترهای مسأله کار می‌کنند و جمعیتی متشکل از نقاط در یک فضای جستجو را برای یافتن جواب‌های مسأله جستجو می‌کنند. همچنین، بدون اینکه از اطلاعات «گرادیان» (Gradient) مرتبط با «تابع هدف» (Objective Function) مسأله اطلاعی داشته باشند، تابع هدف مسأله را بهینه‌سازی می‌کنند. در الگوریتم‌های ژنتیک برای «گذار» (Transition) از یک حالت در فضای مسأله به حالت دیگر، از مکانیزم‌های «احتمالی» (Probabilistic) استفاده می‌شود؛ در حالی که در الگوریتم‌های جستجوی مرسوم، از اطلاعات گرادیان مرتبط با تابع هدف مسأله برای چنین کاری استفاده می‌شود. چنین ویژگی مهمی در الگوریتم‌های ژنتیک، آن‌ها را تبدیل به الگوریتم‌های جستجوی «همه منظوره» (General purpose) کرده است. همچنین، از الگوریتم‌های ژنتیک برای جستجوی فضاهای جستجوی نامنظم و بی‌قاعده استفاده می‌شود. به طور کلی، از الگوریتم‌های ژنتیک برای حل مسأله در کاربردهایی نظیر بهینه‌سازی توابع، «تخمین پارامتر» (Parameter Estimation) و «یادگیری ماشین» (Machine Learning) استفاده می‌شود.

اصول ابتدایی در الگوریتم ژنتیک

اصول کاری الگوریتم ژنتیک، در ساختار الگوریتمی زیر نمایش داده شده است. مهم‌ترین گام لازم برای پیاده‌سازی الگوریتم ژنتیک و انواع مختلف آن عبارتند از: تولید جمعیت (اولیه) از جواب‌های یک مسأله، مشخص کردن تابع هدف، تابع «برازندگی» (Fitness) و به کار گرفتن «عملگرهای ژنتیک» (Genetic Operators) جهت ایجاد تغییرات در جمعیت جواب‌های مسأله. عملگرهای ژنتیک قابل تعریف در الگوریتم ژنتیک، در ادامه معرفی خواهند شد. اصول کاری الگوریتم ژنتیک عبارتست از:

فرموله کردن جمعیت ابتدایی متشکل از جواب‌های مسأله

مقداردهی اولیه و تصادفی جمعیت ابتدایی متشکل از جواب‌های مسأله

حلقه تکرار:

ارزیابی تابع هدف مسأله

پیدا کردن تابع برازندگی مناسب

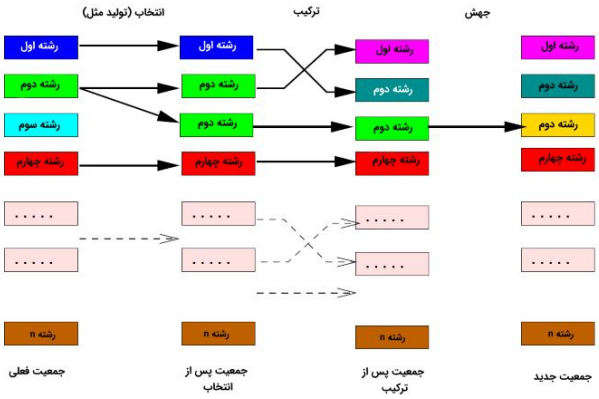
انجام عملیات روی جمعیت متشکل از جواب‌های مسأله با استفاده از عملگرهای ژنتیک

عملگر «تولید مثل» (Reproduction)

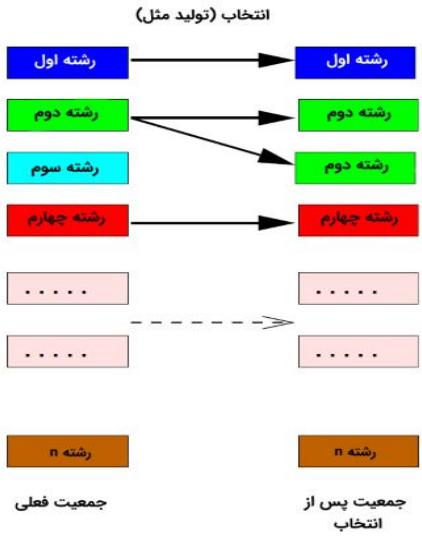
عملگر «ترکیب یا آمیزش» (Crossover)

عملگر «جهش» (Mutation)

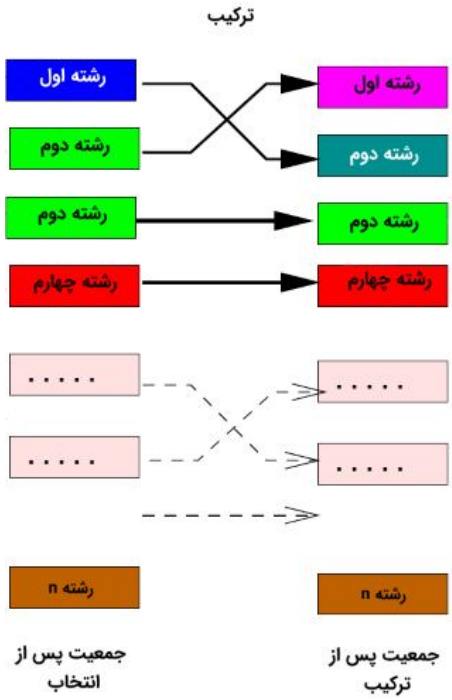
تا زمانی که: شرط توقف ارضا شود.



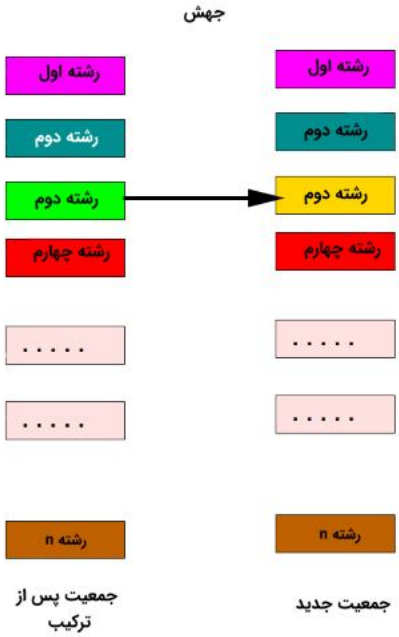
نمای کلی از فرایند تکاملی در الگوریتم ژنتیک پس از تولید مثل، ترکیب و جهش



نمای کلی از اولین مرحله از فرایند تکاملی در الگوریتم ژنتیک پس از تولید مثل



نمای کلی از دومین مرحله از فرایند تکاملی در الگوریتم ژنتیک پس از ترکیب (ادامه تصویر قبلی)



نمای کلی از سومین مرحله از فرایند تکاملی در الگوریتم ژنتیک پس از جهش (ادامه تصویر قبلی)

یکی از مهم‌ترین ویژگی‌های الگوریتم ژنتیک، کدبندی متغیرهای لازم برای توصیف مسأله است. مرسوم‌ترین روش کدبندی متغیرهای مسأله، تبدیل متغیرها به رشته یا برداری از مقادیر باینری، صحیح و یا حقیقی است. بهترین عملکرد الگوریتم‌های ژنتیک معمولا زمانی اتفاق می‌افتد که از نمایش باینری برای کدبندی متغیرهای مسأله استفاده می‌شود.

اگر مسأله‌ای که قرار است جواب‌های بهینه آن مشخص شود بیش از یک متغیر داشته باشد، به تعداد متغیرهای آن مسأله، «کدبندی‌های تک متغیره» (Single-Variable Coding) متناظر با تک تک آن‌ها ایجاد و با یکدیگر ادغام می‌شوند. در چنین حالتی، یک «کدبندی چند متغیره» (Multi-Variable Coding) از مسأله مورد نظر شکل خواهد گرفت. یکی از مهم‌ترین ویژگی‌های الگوریتم‌های ژنتیک، پردازش هم‌زمان چندین جواب کاندید تولید شده برای مسأله تعریف شده است.

بنابراین، در مرحله اول از پیاده‌سازی الگوریتم ژنتیک، مجموعه‌ای متشکل از P موجودیت یا کروموزوم توسط «مولدهای شبه تصادفی» (Pseudo Random Generators) تشکیل می‌شود. هر کدام از موجودیت‌ها با کروموزوم‌های موجود در این جمعیت، یک جواب کاندید و امکان‌پذیر برای مسأله را نمایش می‌دهند. هر کدام از این موجودیت‌ها، یک نمایش برداری از جواب مسأله در یک «فضای جواب» (Solution Space) هستند که به آن‌ها «جواب اولیه» (Initial Solution) نیز گفته می‌شود. از آنجایی که عملیات جستجو از مجموعه‌ای از جواب‌ها (جواب‌های اولیه به طور تصادفی در فضای جواب پراکنده شده‌اند) در فضای جواب مسأله آغاز می‌شود، جستجوی قدرتمند و بدون بایاس در الگوریتم ژنتیک تضمین خواهد شد. در مرحله بعد، تمامی جواب‌های اولیه تولید شده مورد ارزیابی قرار می‌گیرند تا مقدار تابع هدف هر کدام از آن‌ها مشخص شود.

در این مرحله، معمولا یک «تابع جریمه خارجی» (Exterior Penalty Function) به کار گرفته می‌شود تا «مسأله بهینه‌سازی مقید» (Constrained Optimization Problem) به یک مسأله بهینه‌سازی «نامقید» (Unconstrained) تبدیل شود. چنین تبدیلی، بسته به مسائل بهینه‌سازی مختلف (مسائلی که قرار است جواب‌های بهینه آن‌ها تولید شود)، متفاوت خواهد بود.

در مرحله سوم، تابع هدف مسأله به یک تابع برازندگی نگاشت می‌شود. از طریق تابع برازندگی، «مقدار برازندگی» (Fitness Value) هر یک از اعضای جمعیت اولیه مشخص می‌شود. پس از مشخص شدن مقدار برازندگی جواب‌های کاندید، از عملگرهای الگوریتم ژنتیک جهت انجام تغییرات روی جواب‌های کاندید استفاده می‌شود.

اصول کاری الگوریتم ژنتیک

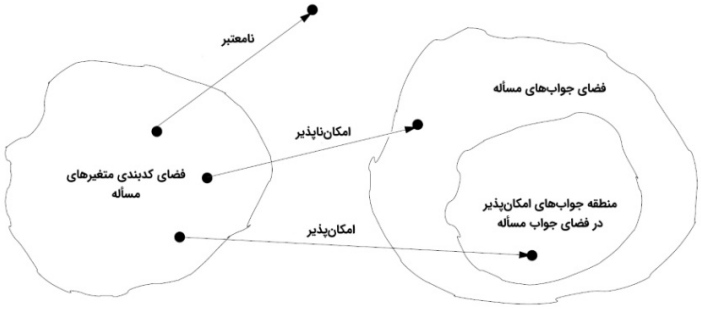
برای نمایش اصول کاری الگوریتم ژنتیک، یک مسأله بهینه‌سازی نامقید در نظر گرفته می‌شود. فرض کنید که مسأله بهینه‌سازی زیر داده شده است و هدف الگوریتم ژنتیک، «بیشینه‌سازی» (Maximizing) تابع زیر باشد:

maximizef(x),xli≤xi≤xui,i=1,2,....,N,maximizef(x),xil≤xi≤xiu,i=1,2,....,N,

در این رابطه، xlixil و xuixiu حد بالا و حد پایین تعیین شده برای مقادیر متغیر xixi است. اگر چه در اینجا از یک مسأله بیشینه‌سازی برای نمایش اصول کاری الگوریتم ژنتیک استفاده است، با این حال، الگوریتم ژنتیک به راحتی قادر به کمینه‌سازی مسائل مختلف نیز خواهد بود. برای پیاده‌سازی الگوریتم‌های ژنتیک، وظایف خاصی باید تعریف و توسط این الگوریتم، جهت بهینه‌سازی توابع مختلف، اجرا شوند.

کدبندی متغیرهای مسأله

برای اینکه الگوریتم ژنتیک قادر باشد تا تابع بالا را بیشینه کند، متغیرهای xixi باید توسط یک ساختار رشته‌ای کدبندی شوند. شایان توجه است که کدبندی متغیرهای مسأله، کاملا ضروری نیست. در تحقیقات انجام شده توسط محققان این حوزه، الگوریتم‌های ژنتیکی پیاده‌سازی شده‌اند که به جای کدبندی متغیرهای مسأله، به طور مستقیم روی خود متغیرها عمل می‌کنند. با این حال، این دسته از الگوریتم‌ها به نوعی استثناء محسوب می‌شوند و در این مطلب، کدبندی متغیرهای مسأله به عنوان یکی از وظایف اصلی الگوریتم‌های ژنتیک مرسوم لحاظ شده است.



کدبندی متغیرهای مسأله در الگوریتم ژنتیک

از کدبندی مبتنی بر رشته‌های باینری، برای کدبندی متغیرهای مسأله استفاده می‌شود. در الگوریتم ژنتیک، طول رشته باینری معمولا بر حسب دقت مطلوب و مورد انتظار از مسأله بهینه‌سازی تعیین می‌شود. به عنوان نمونه، اگر از چهار بیت برای کدبندی یک مسأله بهینه‌سازی دو متغیره استفاده شود، رشته‌های (۰۰۰۰ ۰۰۰۰) و (۱۱۱۱ ۱۱۱۱) به ترتیب متناظر با دو نقطه (xl1,xl2)T(x1l,x2l)T و (xu1,xu2)T(x1u,x2u)T در فضای جواب مسأله خواهند بود؛ زیرا، زیررشته‌های (۰۰۰۰) و (۱۱۱۱) حد بالا و پایین یک نقطه در فضای جستجوی مسأله را نمایش خواهند داد.

در کدبندی متغیرهای مسأله، با استفاده از یک «قانون نگاشت» (Mapping Rule) ثابت، هر رشته هشت بیتی را می‌توان به یک نقطه در فضای جستجوی مسأله نگاشت کرد. معمولا از قانون نگاشت خطی زیر، برای نگاشت یک رشته هشت بیتی به یک نقطه در فضای جستجوی مسأله استفاده می‌شود: xi=xli+xui−xli2β−1∑βj=0γj2jxi=xil+xiu−xil2β−1∑j=0βγj2j

در این رابطه، متغیر xixi به وسیله زیر رشته sisi با طول ββ کدبندی شده است. مقدار کدگشایی شده زیر رشته sisi ، از طریق عبارت β∑j=0γj2j∑j=0βγj2j محاسبه می‌شود. همچنین، sisi به وسیله مقادیر (۰ ، ۱) و رشته ss ، در قالب (sβ−1,sβ−1,....,s1,s0)(sβ−1,sβ−1,....,s1,s0) نمایش داده می‌شوند. به عنوان نمونه، با استفاده از رابطه β∑j=0γj2j∑j=0βγj2j ، مقدار کدگشایی شده برای رشته چهار بیتی (۰۱۱۱)، برابر با مقدار ۷ محاسبه خواهد شد.

شایان توجه است که برای یک رشته ۴ بیتی، از آنجایی که هر بیت تنها می‌تواند مقادیر ۰ یا ۱ به خود اختیار کند، تنها ۱۶ زیر رشته متمایز قابل تعریف خواهد بود. دقت قابل استحصال توسط کدبندی چهار بیتی، تقریبا برابر با یک شانزدهم (۱۶/۱) فضای

جستجوی مسأله است. اما اگر طول رشته به ۵ افزایش پیدا کند، دقت قابل استحصال توسط کدبندی پنج بیتی، تقریبا برابر با یک شانزدهم (۳۲/۱) فضای جستجوی مسأله خواهد شد. بنابراین، طول زیر رشته‌های نمایش دهنده مقادیر متغیرها، وابستگی مستقیم به دقت مطلوب و مورد انتظار از متغیرها دارد. طول زیر رشته‌ها، بسته به دقت مطلوب و مورد انتظار از نتایج الگوریتم، متغیر است؛ هر چقدر طول رشته‌ها بیشتر باشد، دقت نتایج خروجی بیشتر می‌شود. رابطه میان طول رشته ββ و دقت نتایج تولید شده αα از طریق رابطه زیر به دست می‌آید: (xui–xli)10α≤(2β–1)(xiu–xil)10α≤(2β–1) به محض اینکه فرایند کدبندی متغیرهای مسأله به پایان رسید، هر کدام از نقاط متناظر با مقادیر متغیرهای مسأله، x=(x1,x2,.....,xN)Tx=(x1,x2,.....,xN)T ، تولید و در فضای جستجوی مسأله قرار داده خواهند شد. پس از تولید نقاط متناظر با مقادیر متغیرهای مسأله در فضای جستجو، مقدار تابع در نقطه xx ، از طریق جایگذاری xx در یک تابع هدف داده شده یا همان f(x)f(x) ، محاسبه خواهد شد.

تابع برازندگی

همانطور که پیش از این نیز اشاره شد، الگوریتم ژنتیک از اصل بقاء برازنده‌ترین‌ها در طبیعت، برای ایجاد فرایند جستجو و متعاقبا، جستجو در فضای جواب مسأله استفاده می‌کند. بنابراین، الگوریتم ژنتیک برای حل مسائل بهینه‌سازی (بیشینه‌سازی یا کمینه‌سازی) بسیار مناسب خواهد بود. به طور کلی، یک «تابع برازندگی» (Fitness Function) یا F(i)F(i) ، در ابتدا با استفاده از تابع هدف فرموله می‌شود و در عملیات ژنتیکی متوالی در نسل‌های الگوریتم ژنتیک مورد استفاده قرار می‌گیرد.

از دیدگاه زیست‌شناسی، برازندگی یک «مقدار کیفی» (Qualitative Value) است که بازده تولیدِ مثل کروموزوم‌ها را می‌سنجد. در الگوریتم ژنتیک، از تابع برازندگی برای محاسبه شانس (یا احتمال) تولید مثل موجودیت‌ها یا کروموزوم‌های موجود در جمعیت نیز استفاده می‌شود؛ به عبارت دیگر، به عنوان معیاری برای مشخص کردن خوب بودن کروموزوم‌ها یا جواب‌های کاندید مسأله مورد استفاده قرار می‌گیرد (و معمولا این معیار باید بیشینه شود). در الگوریتم ژنتیک، کروموزوم‌ها یا موجودیت‌هایی که بیشترین برازندگی را دارند، نسبت به دیگر کروموزوم‌های موجود در جمعیت، شانس بیشتری برای ترکیب و جهش (عملگرهای ژنتیکی) خواهند داشت. عملکرد صحیح برخی از عملگرهای ژنتیکی منوط به تعریف توابع برازندگی «نامنفی» (Non-Negative) است؛ با این حال، دیگر عملگرهای ژنتیکی، پیش‌شرط نامنفی بودن تابع برازندگی را برای انجام عملیات خود تعریف نمی‌کنند. در مسائل بیشینه‌سازی، می‌توان تابع برازندگی را معادل تابع هدف در گرفت. به عبارت دیگر:

F(i)=O(i)→Objectivefunction(i)=Fitnessfunction(i)F(i)=O(i)→Objectivefunction(i)=Fitnessfunction(i)

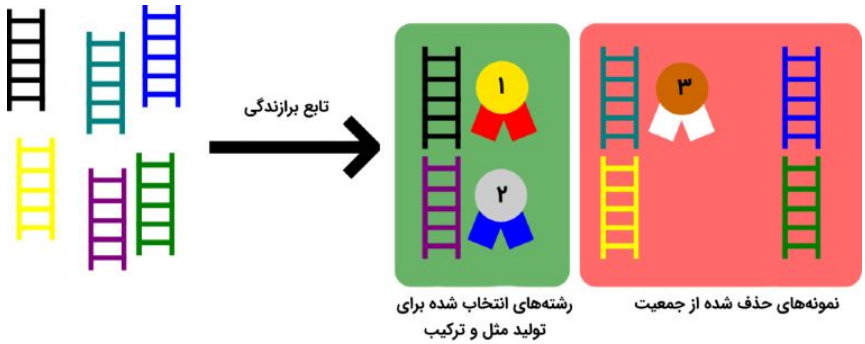
در مسائل «کمینه‌سازی» (Minimization)، برای تولید مقادیر نامنفی در تمامی حالات و جهت منعکس کردن برازندگی نسبی رشته‌های متناظر با کروموزوم‌ها یا موجودیت‌های جمعیت، بسیار حیاتی است که تابع هدف اصلی مسأله به تابع برازندگی نگاشت شود. برای انجام چنین نگاشتی، روش‌های متفاوتی وجود دارد. در ادامه، دو روش شایع و پرکاربرد جهت نگاشت تابع هدف به تابع برازندگی در مسائل کمینه‌سازی، معرفی شده‌اند. F(x)=11+f(x)F(x)=11+f(x)

در این رابطه، F(x)F(x) تابع برازندگی و f(x)f(x) تابع هدف مسأله کمینه‌سازی تعریف شده است. این تبدیل، مکان جواب کمینه در فضای جواب‌های مسأله را تغییر نمی‌دهد ولی یک مسأله کمینه‌سازی را به یک مسأله بیشینه‌سازی معادل آن تبدیل می‌کند.

تابع جایگزین دیگری که جهت تبدیل تابع هدف به مقدار برازندگی کروموزوم ii ، یا F(i)F(i) ، مورد استفاده قرار می‌گیرد، به شکل زیر تعریف می‌شود. F(i)=V–O(i)P∑Pi=1O(i)F(i)=V–O(i)P∑i=1PO(i)

در این رابطه، O(i)O(i) مقدار تابع هدف موجودیت یا کروموزوم ii اُم، PP اندازه جمعیت و VV یک مقدار بزرگ است که نامنفی شدن مقادیر برازندگی را تضمین می‌کند. مقدار VV در این تبدیل، معمولا برابر مقدار بیشینه عبارت O(i)P∑Pi=1O(i)O(i)P∑i=1PO(i) در نظر گرفته می‌شود؛ در چنین حالتی، مقدار برازندگی متناظر با مقدار بیشینه تابع هدف، برابر با صفر می‌شود. این تبدیل نیز مکان جواب کمینه را در فضای جواب‌های مسأله تغییر نمی‌دهد ولی یک مسأله کمینه‌سازی را

به یک مسأله بیشینه‌سازی معادل آن تبدیل می‌کند. به مقدار تابع برازندگی یک رشته، «برازندگی رشته» (String Fitness) نیز گفته می‌شود.



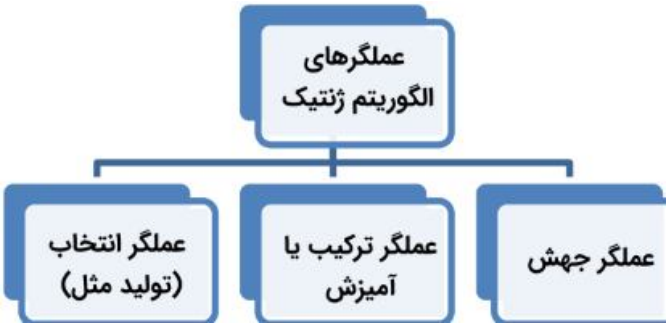
عملگرهای الگوریتم ژنتیک

عملیات بهینه‌سازی در الگوریتم ژنتیک، با یک تولید جمعیت اولیه از «رشته‌های تصادفی» (Random Strings) آغاز می‌شود (این رشته‌ها معادل کروموزوم‌ها یا موجودیت‌ها یا جواب‌های کاندید مسأله هستند). رشته‌های تصادفی، طراحی مسأله یا به عبارت دیگر، «متغیرهای تصمیم» (Decision Variables) مرتبط با یک مسأله را نمایش می‌دهند.

سپس، جمعیت اولیه تحت تأثیر سه دسته عملگر اصلی در الگوریتم ژنتیک قرار می‌گیرند تا جمعیت جدیدی از نقاط در فضای جواب مسأله تولید شود؛ جمعیت جدید، متشکل از کروموزوم‌ها یا موجودیت‌ها یا جواب‌های جدید خواهد بود. عملگرهای اصلی الگوریتم ژنتیک عبارتند از: عملگر تولید مثل، عملگر ترکیب یا آمیزش و عملگر جهش.

همانطور که پیش از این نیز اشاره شد، الگوریتم ژنتیک را می‌توان به عنوان مکانیزمی جهت بیشینه‌سازی تابع هدف در نظر گرفت. این کار، از طریق ارزیابی کروموزوم‌ها یا بردارهای جواب انجام می‌شود. هدف عملگرهای اصلی در الگوریتم ژنتیک، انتخاب، ترکیب و تغییر بردارهای متناظر با جواب‌هایی است که در نسل کنونی، بهترین جواب برای مسأله بهینه‌سازی محسوب می‌شوند. از این طریق، جمعیت جدیدی از کروموزوم‌ها یا بردارهای جواب تولید خواهد شد.

جمعیت جدید تولید شده نیز مورد ارزیابی بیشتر قرار می‌گیرد و اینکار تا «خاتمه یافتن» (Termination) فرایندهای عملیاتی در الگوریتم ژنتیک ادامه پیدا می‌کند. تا زمانی که شرط توقف الگوریتم ژنتیک ارضا نشود، جمعیت کروموزوم‌ها یا بردارهای جواب، به وسیله عملگرهای تولید مثل، ترکیب و جهش دستکاری و ارزیابی می‌شوند. این رویه تا زمانی که معیار توقف الگوریتم ژنتیک ارضا شود، ادامه پیدا می‌کند. یک چرخه (حلقه تکرار) از فرایند دستکاری جمعیت متشکل از کروموزوم‌ها یا بردارهای جواب، توسط عملگرهای تولید مثل، ترکیب و جهش و ارزیابی متعاقب آن‌ها، به عنوان یک «نسل» (Generation) از الگوریتم ژنتیک شناخته می‌شود. در ادامه، هر یک از عملگرهای بالا به تفصیل شرح داده خواهند شد.



عملگر تولید مثل

عملگر تولید مثل یا «انتخاب» (Selection)، عملگری است که بهترین «رشته‌ها» (Strings) در یک جمعیت جدید را کپی می‌کند (منظور، بهترین کروموزوم‌ها یا بردارهای جواب کاندید در یک نسل است). عملگر تولیدِ مثل، معمولا اولین عملگری است

که برای دستکاری جمعیت مورد استفاده قرار می‌گیرد و روی کروموزوم‌ها اعمال می‌شود. این عملگر، رشته‌ها یا کروموزوم‌های خوب جمعیت را انتخاب می‌کند و آن‌ها را در مخزنی که در اصطلاح به آن Mating Pool گفته می‌شود، قرار می‌دهد؛ به همین دلیل است که به عملگر تولید مثل، عملگر انتخاب نیز گفته می‌شود. بنابراین، عملیات تولید مثل در انتخاب طبیعی سبب می‌شود تا رشته‌ها یا کروموزوم‌هایی که برازندگی بهتری نسبت به دیگر کروموزوم‌ها دارند، با تناوب بیشتری خود را تکثیر کنند. تولید مثل رشته‌ها یا کروموزوم‌های موجود در جمعیت فعلی، برای کمک به تولید جمعیت جدید در الگوریتم ژنتیک ضروری است. برای اینکه جمعیت جدید تولید شده در نسل‌های آینده، برازندگی بهتری نسبت به جمعیت نسل فعلی داشته باشند، لازم است تا تناوب تولیدِ مثل در برازنده‌ترین کروموزوم‌های موجود در جمعیت نسل فعلی بیشتر شود. تاکنون، عملگرهای تولید مثل مختلفی برای الگوریتم ژنتیک توسعه داده شده است، ولی مکانیزم‌های عملیاتی آن‌ها تقریبا یکسان است؛ رشته‌ها یا کروموزوم‌های خوب از جمعیت فعلی انتخاب و کپی‌های تولید شده از آن‌ها، به شیوه‌ای احتمالی، در مخزن Mating Pool قرار داده می‌شوند. نکته مهم در مورد عملیات انجام شده توسط عملگر تولید مثل یا ترکیب این است که در این مرحله، رشته‌ها یا کروموزوم‌های جدیدی (با مقادیر متغیر متفاوت از جمعیت اصلی) تشکیل نمی‌شود.

عملگر انتخاب مبتنی بر چرخ رولت

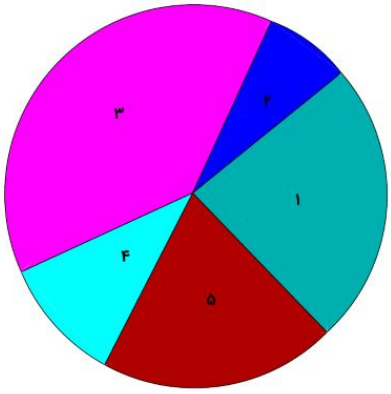
یکی از عملگرهای شایع و پرکاربرد برای تولید مثل در الگوریتم ژنتیک، روش «انتخاب چرخ رولت» (Roulette-Wheel Selection) نام دارد. در این عملگر، احتمال انتخاب یک رشته یا کروموزوم برای قرار گرفتن در مخزن Mating Pool، متناسب با برازندگی آن رشته یا کروموزم محاسبه می‌شود. بنابراین، احتمال قرار گرفتن رشته ii اُم در مخزن Mating Pool، متناسب یا برازندگی آن یا FiFi خواهد بود.

از آنجایی در الگوریتم ژنتیک اندازه جمعیت ثابت است، حاصل جمع تمامی احتمالات محاسبه شده برای انتخاب و قرار گرفتن کروموزوم‌ها یا رشته‌ها در مخزن Mating Pool، برابر با ۱ خواهد بود. بنابراین، احتمال انتخاب و قرار گرفتن رشته ii اُم در مخزن Mating Pool، برابر است با:

pi=Fi∑ni=1Fipi=Fi∑i=1nFi

در این رابطه، nn برابر با اندازه جمعیت است. یک روش ساده برای پیاده‌سازی این عملگر، تصور کردن یک چرخ رولت است که محیط آن، متناسب با برازندگی هر یک از رشته‌ها یا کروموزوم‌ها نشانه‌گذاری شده است. چرخ رولت، nn بار خواهد چرخید. در هر بار چرخش این چرخ، هر نمونه‌ای که توسط نشانه‌گر چرخ نشان داده شود، برای قرار گرفتن در مخزن Mating Pool انتخاب می‌شود. از آنجایی که محیط چرخ رولت، متناسب با برازندگی هر یک از رشته‌ها یا کروموزوم‌ها نشانه‌گذاری شده است، انتظار می‌رود که Fi/¯¯¯¯FFi/F¯ کپی از رشته ii اُم توسط این عملگر، در مخزن Mating Pool تولید شود. برازندگی میانگین جمعیت کروموزوم‌ها نیز به شکل زیر محاسبه می‌شود.

¯¯¯¯F=n∑i=1FiF¯=∑i=1nFi



جمعیت	برازندگی
1	25.0
2	5.0
3	40.0
4	10.0
5	20.0

چرخ رولت توسط کروموزوم‌های موجود در جمعیت فعلی و بر اساس مقدار برازندگی آن‌ها نشانه‌گذاری شده است.
به عبارت دیگر، احتمال انتخاب هر کدام از کروموزوم‌های جمعیت نسل فعلی برابر با مقدار زیر خواهد بود:

$$pi=Fi\sum ni=1Fipi=Fi\sum i=1nFi$$

شکل بالا، عملگر انتخاب چرخ رولت را برای هر کدام از موجودیت یا کروموزوم‌های موجود در جمعیت را نمایش می‌دهد. همانطور که مشاهده می‌شود، برازندگی کروموزوم‌ها متفاوت از یکدیگر است. از آنجایی که موجودیت یا کروموزوم سوم، مقدار برازندگی بالاتری نسبت دیگر کروموزوم‌های موجود در جمعیت دارد، می‌توان انتظار داشت که عملگر انتخاب (تولید مثل) مبتنی بر چرخ رولت، این کروموزوم‌ها را بیش از دیگر کروموزوم‌ها انتخاب کند و در مخزن Mating Pool قرار دهد. جدول زیر، برازندگی هر کدام از کروموزوم‌های شکل بالا، احتمال انتخاب و احتمال تجمعی آن‌ها را نمایش می‌دهد:

جمعیت	برازندگی	احتمال انتخاب	احتمال تجمعی
1	25	25،0	25،0
2	5	05،0	30،0
3	40	40،0	70،0
4	10	10،0	80،0
5	20	20،0	1

به عنوان نمونه، به مثال جمعیت بالا توجه کنید. برای انتخاب nn موجودیت یا کروموزوم از بین جمعیت نسل فعلی، تعداد nn عدد تصادفی بین صفر و یک تولید خواهد شد. سپس، به ازاء هر کدام از مقادیر تصادفی تولید شده، شرط‌های زیر به ترتیب چک می‌شوند: در صورتی که مقدار تصادفی تولید شده کوچکتر از احتمال تجمعی کروموزوم اول (۰،۲۵) باشد، کروموزوم ۱ انتخاب می‌شود. در غیر این صورت، شرط بعدی چک می‌شود.

در صورتی که مقدار تصادفی تولید شده کوچکتر از احتمال تجمعی کروموزوم دوم (۰،۳۰) باشد، کروموزوم ۲ انتخاب می‌شود. در غیر این صورت، شرط بعدی چک می‌شود

در صورتی که مقدار تصادفی تولید شده کوچکتر از احتمال تجمعی کروموزوم سوم (۰،۷۰) باشد، کروموزوم ۳ انتخاب می‌شود. در غیر این صورت، شرط بعدی چک می‌شود.

در صورتی که مقدار تصادفی تولید شده کوچکتر از احتمال تجمعی کروموزوم چهارم (۰،۸۰) باشد، کروموزوم ۴ انتخاب می‌شود. در غیر این صورت، شرط بعدی چک می‌شود.

در نهایت، در صورتی هیچ یک از شرط‌های بالا صحیح نباشند، کروموزوم ۵ انتخاب می‌شود.

عملگر انتخاب باقی مانده تصادفی (بدون جایگذاری)

این عملگر، روش بهتری برای انتخاب رشته‌ها یا کروموزوم‌ها است. ایده اصلی این روش این است که رشته‌ها یا کروموزوم‌ها، بر اساس تعداد دفعات تولید مثل از آن‌ها، از جمعیت جدید حذف و یا در جمعیت جدید تکثیر می‌شوند. برای چنین کاری، پارامتر تعداد «دفعات تولید مثل» (Reproduction Count) به ازاء هر کدام از رشته‌ها یا کروموزوم‌ها محاسبه می‌شود.

پارامتر تعداد دفعات تولید مثل متناظر با هر یک از رشته‌ها یا کروموزوم‌ها، بر اساس مقدار برازندگی (هر کدام از رشته‌ها) و بدون جایگذاری محاسبه خواهد شد. این روش، نسبت به دیگر عملگرهای انتخاب برتر است و برای استفاده در الگوریتم ژنتیک پیشنهاد می‌شود. در این عملگر، ابتدا احتمال انتخاب یا psps به شکل زیر محاسبه می‌شود:

$$ps=F(i)\sum F(i)ps=F(i)\sum F(i)$$

بنابراین، تعداد دفعات تولید مثل مورد انتظار به ازاء هر رشته یا کروموزوم، از طریق رابطه زیر محاسبه می‌شود:

$$ei=ps\times Pei=ps\times P$$

در این رابطه، PP برابر با اندازه جمعیت است. از قسمت کسری مقادیر eiei تولید شده به ازاء هر رشته یا کروموزوم، به عنوان احتمالات انتخاب آن کروموزوم‌ها استفاده می‌شود. به عنوان نمونه، در صورتی که پارامتر تعداد دفعات تولید مثل مورد انتظار یک رشته یا کروموزوم، برابر با ei=1.5ei=1.5 باشد، حتما یک کپی از این رشته در جمعیت جدید وجود خواهد داشت. همچنین، یک کپی دیگر با احتمال ۰٫۵۰، برای قرار گرفتن در جمعیت جدید انتخاب خواهد شد.

این کار تا زمانی انجام می‌شود که پارامتر تعداد دفعات تولید مثل مورد انتظار، به ازاء هر کدام از کروموزم‌ها یا رشته‌های موجود در جمعیت محاسبه شود. کروموزوم‌هایی که پارامتر تعداد دفعات تولید مثل مورد انتظار آن‌ها برابر با صفر باشد، از جمعیت کروموزوم‌ها حذف خواهند شد. کروموزوم‌هایی که پارامتر دفعات تولید مثل مورد انتظار آن‌ها برابر با مقداری غیر صفر باشد، تعداد کپی‌های تکثیر شده از کروموزوم آن‌ها در جمعیت جدید، متناسب با مقدار پارامتر تعداد دفعات تولید مثل مورد انتظار، محاسبه شده خواهد بود. همچنین، اندازه جمعیت جدید (حاصل شده پس از فرایند تولید مثل) ثابت و با اندازه جمعیت قبلی (پیش از عملیات تولید مثل) برابر خواهد بود. با چنین کاری، عملیات تولید مثل در الگوریتم ژنتیک کامل خواهد شد.

تقریبا تمامی عملگرهای انتخاب یا تولید مثل، اصول کاری مشابه یکدیگر دارند و فقط تعداد کپی‌هایی که به هر کروموزوم یا رشته اختصاص می‌دهند متفاوت خواهد بود. در نهایت، این نکته شایان توجه است که در تمامی عملگرهای تولید مثل یا انتخاب برای الگوریتم ژنتیک، رشته‌هایی به تعداد بیشتر در جمعیت جدید کپی می‌شوند که برازندگی آن‌ها، از دیگر رشته‌های موجود در جمعیت بیشتر باشند.

عملگر ترکیب یا آمیزش

از عملگر «ترکیب یا آمیزش» (Crossover)، برای «بازترکیب» (Recombine) دو رشته یا کروموزوم استفاده می‌شود. این کار، با هدف تولید رشته‌ها یا کروموزوم‌های بهتر انجام می‌شود. در عملیات ترکیب در الگوریتم ژنتیک، طریق ترکیب کردن مواد ژنتیکی دو کروموزوم موجود در جمعیت نسل قبل، کروموزوم‌های جدیدی در نسل‌های فعلی می‌شود. به عبارت دیگر، فرایند بازترکیب، ژن‌های موجود در دو کروموزوم را ترکیب و از این طریق، کروموزوم‌های جدیدی در جمعیت فعلی تولید می‌کند.

چنین فرایندی به صورت تکراری و در تمامی نسل‌های یک الگوریتم ژنتیک انجام خواهد شد. در فرایند تولید مثل، معمولا تعداد کپی‌های ایجاد شده از کروموزوم‌هایی که برازندگی بالایی دارند، بیشتر از دیگر کروموزم‌ها خواهد بود. در پایان فرایند تولید مثل، مخزن Mating Pool تشکیل می‌شود (و تمامی کپی‌های تولید شده در آن قرار می‌گیرد).

همانطور که پیش از این نیز اشاره شد، در مرحله تولید مثل، رشته‌ها یا کروموزوم‌های جدیدی (با مقادیر متغیر متفاوت از جمعیت اصلی) در جمعیت تشکیل نمی‌شوند. در این مرحله (و پس از عملیات حاصل از عملگر ترکیب)، رشته‌ها یا کروموزوم‌های جدیدی از طریق تبادل اطلاعات (ژنی) میان رشته‌ها یا کروموزوم‌های موجود در مخزن Mating Pool تشکیل می‌شوند.

به دو کروموزوم یا رشته‌ای که در عملیات ترکیب یا آمیزش مشارکت می‌کنند، کروموزوم‌های «والد» (Parents) گفته می‌شود. همچنین، کروموزوم‌هایی که در اثر فرایند ترکیب یا آمیزش تولید می‌شوند، کروموزوم‌های «فرزند» (Children) نامیده می‌شوند.

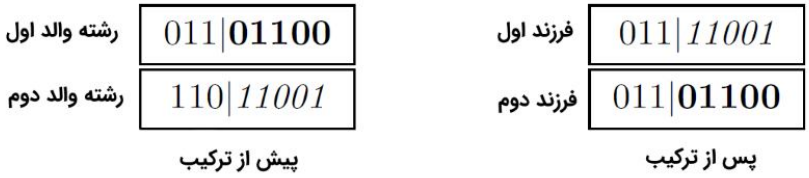
بنابراین، یک نتیجه‌گیری ممکن از عملیات ترکیب می‌تواند این گونه باشد که ترکیب زیررشته‌های خوب (منظور، مجموعه‌ای از ژن‌های خوب در کروموزوم‌های والد) از رشته‌ها یا کروموزوم‌های والدین با یکدیگر، می‌تواند منجر به تولید رشته‌ها یا کروموزوم‌های فرزند خوب شوند. چنین برداشتی، زمانی منطقی و صحیح به شمار می‌آید که عملیات ترکیب زیررشته‌ها، به صورت مشخص و روی ژن‌هایی از رشته‌های والدین صورت بگیرد که ترکیب آن‌ها، سبب تولید رشته یا کروموزوم فرزند خوب می‌شود (ترکیب این زیر رشته‌ها، به صورت احتمالی انجام می‌شود). در صورتی که عملیات انتخاب زیررشته‌ها و ترکیب آن‌ها، بر اساس فرایندهای تصادفی انجام شود، هیچ تضمینی وجود ندارد که فرزندهای حاصل، زیررشته‌های خوب والدین خود را به ارث ببرند. در این حالت، خوب بودن یا خوب نبودن فرزندان، به طور مستقیم، به ژن‌هایی در رشته‌های والدین بستگی دارد که در عملیات ترکیب و تولید فرزندان مشارکت دارند. با این حال، چنین منطقی در الگوریتم ژنتیک نگران کننده نیست؛ زیرا در صورتی که کروموزوم‌های فرزند خوبی توسط عملیات ترکیب یا آمیزش تولید شوند، در نسل‌های بعدی، با احتمال بیشتری انتخاب می‌شوند و کپی‌های بیشتری از

آن‌ها تولید می‌شود. کپی‌های تولید شده نیز در مخزن Mating Pool قرار می‌گیرند تا در مراحل بعدی مورد دستکاری ژنی قرار بگیرند.با این حال، طبیعت تصادفی عملگرهای ژنتیک (نظیر عملگر ترکیب) ممکن است اثر «مخرب» (Detrimental) یا «سودمند» (Beneficial) در کیفیت کروموزوم‌ها یا همان جواب‌های مسأله داشته باشد. در صورتی که کروموزوم‌های فرزند خوبی در نتیجه ترکیب تولید شوند، در جمعیت نسل‌های بعدی، کروموزوم‌های خوبی مشارکت خواهند کرد و برعکس.

بنابراین، برای حفظ برخی از رشته‌ها یا کروموزوم‌های خوبی که در مخزن Mating Pool وجود دارند، تمامی رشته‌ها یا کروموزوم‌های موجود در مخزن Mating Pool، توسط عملگر ترکیب دستکاری نخواهند شد. برای چنین کاری، از مفهومی به نام «احتمال ترکیب» (Crossover Probability) یا pcpc استفاده می‌شود.وقتی که در الگوریتم ژنتیک، پارامتر احتمال ترکیب تعریف می‌شود، یعنی تنها pcpc درصد از رشته‌ها یا کروموزوم‌های موجود در جمعیت، توسط عملگر ترکیب دستکاری می‌شوند. به عبارت دیگر، (1–pc)(1–pc) درصد از رشته‌ها یا کروموزوم‌های موجود در جمعیت، به همان شکل اصلی خودشان در جمعیت نسل فعلی باقی خواهند ماند.در الگوریتم ژنتیک، از عملگر ترکیب برای جستجوی رشته‌ها یا کروموزوم‌های جدید در فضای جواب (فضای جستجو) استفاده می‌شود. البته، عملگر جهش در الگوریتم ژنتیک نیز برای چنین کاربردی مورد استفاده قرار می‌گیرد.

تاکنون، عملگرهای ترکیب متعددی برای الگوریتم ژنتیک توسعه داده شده‌اند. روش‌های «تک نقطه‌ای» (Single Point) و «دو نقطه‌ای» (Two point)، از جمله مهم‌ترین عملگرهای ترکیب در الگوریتم ژنتیک محسوب می‌شوند. در غالب عملگرهای ترکیب توسعه داده شده برای الگوریتم ژنتیک، دو رشته یا کروموزوم به طور تصادفی از مخزن Mating Pool انتخاب می‌شوند و بخش‌هایی از رشته‌های این دو کروموزوم با یکدیگر ترکیب می‌شوند تا رشته‌ها یا کروموزوم‌های جدیدی پدید آیند.

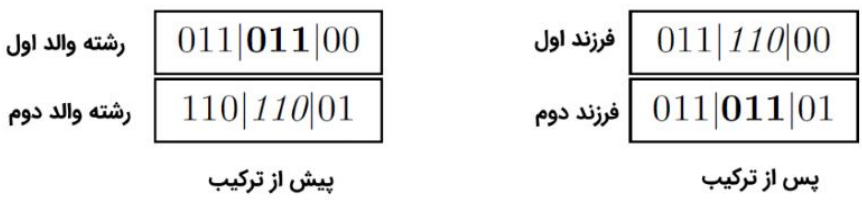
عملیات ترکیب در سطح رشته انجام می‌شود و پس از انتخاب دو رشته یا کروموزوم والد، ژن‌های آن‌ها با یکدیگر مبادله می‌شوند تا کروموزوم‌های فرزند جدید شکل بگیرد. در عملگر ترکیب تک نقطه‌ای، یک ژن به شکل تصادفی، در امتداد یکی از دو رشته یا کروموزوم والد، به عنوان «محل ترکیب» (Crossover Site) در عملیات ترکیب انتخاب می‌شود. سپس، تمامی ژن‌های موجود در سمت راست محل ترکیب رشته اول، با ژن‌های متناظر آن‌ها در کروموزوم یا رشته دوم جا به جا می‌شوند. چگونگی انجام عمل ترکیب تک نقطه‌ای روی کروموزوم‌های والد (توسط عملگر متناظر آن در الگوریتم ژنتیک)، در شکل زیر نمایش داده شده است.



عملیات ترکیب تک نقطه‌ای

همانطور که در شکل بالا مشهود است، در عملگر ترکیب تک نقطه‌ای، یک محل ترکیب به شکل تصادفی انتخاب می‌شود (در شکل بالا، محل ترکیب به شکل یک خط عمودی نمایش داده شده است). تمامی ژن‌هایی که در سمت راست محل ترکیب انتخاب شده (در دو رشته یا کروموزوم) قرار دارند، با یکدیگر مبادله خواهند شد و از این طریق، دو کروموزوم یا رشته جدید تولید می‌شود.

عملگر ترکیب دو نقطه‌ای، حالت خاصی از عملگر ترکیب تک نقطه‌ای محسوب می‌شود؛ با این تفاوت که به جای یک محل ترکیب، دو محل ترکیب در رشته‌ها یا کروموزوم‌های والد انتخاب می‌شوند و ژن‌های قرار گرفته میان محل‌های ترکیب انتخابی، به نحوی که در شکل ۵ نمایش داده شده است، با یکدیگر مبادله می‌شوند.



عملیات ترکیب دو نقطه‌ای

عملگر ترکیب تک نقطه‌ای برای حالتی مناسب است که طول رشته‌ها یا کروموزوم‌ها کوتاه هستند. در صورتی که طول رشته‌ها یا کروموزوها بلند باشند، بهتر است که از عملگر ترکیب دو نقطه‌ای برای تولید کروموزوم‌های جدید استفاده شود. بنابراین، در ادامه این مطلب، از عملگر ترکیب دو نقطه‌ای، به عنوان عملگر پیش‌فرض برای انجام عملیات ترکیب استفاده می‌شود.

هدف اصلی عملیات ترکیب در الگوریتم ژنتیک این است که با ترکیب ژن‌های دو کروموزوم یا رشته والد، به کروموزوم‌ها یا رشته‌های فرزندی دست پیدا شود که به احتمال زیاد، ویژگی‌های بهتری نسبت به کروموزوم‌های والد از خود نشان می‌دهند (برازندگی بهتری نسبت به کروموزوم‌های والد دارند).

عملگر جهش

عملگر جهش یکی از مهم‌ترین فرایندهای تکاملی برای رسیدن به جواب بهینه در الگوریتم ژنتیک محسوب می‌شود. در عملگر جهش، به شکل تصادفی، اطلاعات جدیدی به فرایند جستجو در الگوریتم ژنتیک اضافه می‌شود. چنین ویژگی مهمی به الگوریتم ژنتیک کمک می‌کند تا از قرار گرفتن در دام «بهینه محلی» (Local Optimum) فرار کند.

زمانی که در نسل‌های متوالی، از عملیات ترکیب و تولید مثل به دفعات روی رشته‌ها یا کروموزوم‌ها استفاده می‌شود، جمعیت کروموزوم‌ها یا جواب‌های کاندید به «همگن» (Homogeneous) شدن گرایش پیدا می‌کند. عملگر جهش به الگوریتم ژنتیک کمک می‌کند تا «تنوع» (Diversity) در جمعیت کروموزوم‌ها یا جواب‌های کاندید افزایش پیدا کند.

عملگر جهش ممکن است منجر به تغییرات عمده در کروموزوم‌های فرزندان تولید شده شود و سبب شود که کروموزوم‌ها یا رشته‌های فرزند تولید شده، ژن‌های کاملا متفاوتی نسبت به کروموزوم یا رشته والدین داشته باشند.

به بیان ساده‌تر، عملگر جهش، فرایندی تصادفی برای به هم ریختن و ایجاد اختلال در اطلاعات ژنتیکی محسوب می‌شود. بر خلاف عملگر ترکیب، عملگر جهش در سطح ژن کار می‌کند؛ یعنی، زمانی که ژن‌ها از رشته یا کروموزوم فعلی در رشته یا کروموزوم جدید کپی می‌شوند، این احتمال وجود دارد که هر کدام از این ژن‌ها جهش پیدا کنند. این احتمال معمولا مقدار بسیار کوچکی است که به آن «احتمال جهش» (Mutation Probability) یا pmpm گفته می‌شود.

در صورتی که از نمایش بیتی (صفر و یک) برای مشخص کردن مقادیر متغیرهای مسأله یا ژن‌ها استفاده شود، جهت انجام عملیات جهش، از مکانیزم «پرتاب سکه» (Coin Toss) در الگوریتم ژنتیک استفاده می‌شود؛ یعنی، در صورتی که مقدار تصادفی (بین صفر و یک) از احتمال جهش کمتر باشد، مقدار ژن یا همان بیت ژن معکوس می‌شود. به عبارت دیگر، مقادیر ژنی برابر با یک تبدیل به صفر و مقادیر ژنی برابر با صفر، تبدیل به یک خواهند شد. اگر به مکانیزم جهش دقت شود می‌توان دریافت که عملگر جهش از طریق معکوس کردن تصادفی مقدار ژن‌ها، باعث ایجاد تنوع در جمعیت کروموزوم‌ها یا جواب‌های کاندید می‌شود. معکوس کردن تصادفی مقدار ژن‌ها، هم باعث همگرایی به جواب بهینه بهتر در فضای جستجوی مسأله می‌شود و هم منجر به تغییرات اساسی در کدهای ژنتیکی کروموزوم‌ها می‌شود؛ پدیده‌ای که مسیر رسیدن به جواب بهینه سراسری را در نسل‌های متوالی از الگوریتم ژنتیک هموارتر می‌کند. از طرف دیگر، عملگر جهش ممکن است منجر به تولید رشته‌ها یا کروموزوم‌های ضعیفی شود که به هیچ عنوان توسط عملگر تولید مثل یا انتخاب، انتخاب نخواهند شد. عملگر جهش مکانیزم هوشمندانه‌ای برای جستجوی محلی در فضای جستجوی جواب‌های مسأله به شمار می‌آید. با استفاده از عملگر جهش در الگوریتم ژنتیک، رشته‌ها یا کروموزوم‌های جدیدی در همسایگی رشته یا کروموزوم فعلی ایجاد می‌شوند. در نتیجه، عملگر جهش سبب ایجاد مکانیزم جستجوی محلی اطراف کروموزوم‌های فعلی می‌شود. همچنین، عملگر جهش سبب حفظ تنوع در جمعیت کروموزوم‌ها یا جواب‌های کاندید می‌شود. به عنوان نمونه، به جمعیت زیر توجه کنید. این جمعیت، از ۴ کروموزم تشکیل شده است که هر کدام از آن‌ها، یک رشته ۸ بیتی هستند:

۰۱۱۰۱۰۱۱

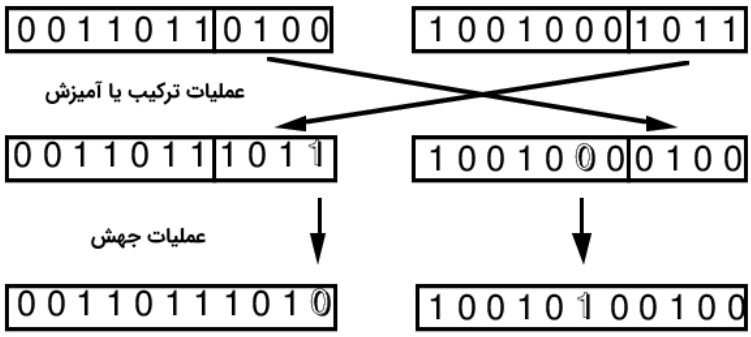
۰۰۱۱۱۱۰۱

۰۰۰۱۰۱۱۰

۰۱۱۱۱۱۰۰

با دقت به جمعیت کروموزومی نمایش داده شده می‌توان دریافت که اولین ژن تمامی رشته‌ها از سمت چپ، صفر (۰) است. در صورتی که ابتدایی‌ترین ژن رشته متناظر با جواب بهینه سراسری برابر با (۱) باشد، هیچ کدام از عملگرهای تولید مثل (انتخاب) و ترکیب قادر به تولید مقدار یک (۱) در این مکان (اولین ژن رشته از سمت چپ) نخواهند بود. استفاده از عملگر جهش در الگوریتم ژنتیک سبب می‌شود تا مقدار اولین ژن این رشته‌ها، با احتمال pmpm از صفر (۰) به یک (۱) تغییر پیدا کند.

سه عملگر معرفی شده برای انجام عملیات تولید مثل (انتخاب)، ترکیب و جهش، عملگرهای ساده و سر راستی هستند. عملگر تولید مثل، رشته خوب را انتخاب و عملگر ترکیب، بهترین زیررشته‌های موجود در رشته‌های خوب جمعیت را با یکدیگر ترکیب می‌کند؛ به امید اینکه جواب‌ها یا رشته‌ها یا کروموزوم‌های حاصل، جواب‌ها یا رشته‌ها یا کروموزوم‌های بهتری نسبت به والدین خود باشند. عملگر جهش، ژن‌های کروموزوم‌ها را به شکل محلی تغییر می‌دهد و از این طریق، منجر به ایجاد جواب‌ها یا رشته‌ها یا کروموزوم‌های بهتر می‌شود.با اینکه هیچ‌گونه تضمینی برای ایجاد جواب‌ها یا رشته‌ها یا کروموزوم‌های خوب در نتیجه عملیات این عملگرها وجود ندارد، با این حال می‌توان از دو مورد زیر تا حد زیادی مطمئن بود: در صورتی که جواب‌ها یا رشته‌ها یا کروموزوم‌های بدی در نتیجه عملگرهای تولید مثل (انتخاب)، ترکیب و جهش حاصل شوند، در عملیات تولیدِ مثل (انتخاب) نسل بعد حذف خواهند شد. در صورتی که جواب‌ها یا رشته‌ها یا کروموزوم‌های خوبی در نتیجه عملگرهای تولید مثل (انتخاب)، ترکیب و جهش حاصل شوند، الگوریتم ژنتیک در نسل‌های متوالی، تاکید بیشتری روی آن‌ها خواهد داشت. استفاده از عملگرهای تولید مثل (انتخاب)، ترکیب و جهش روی رشته‌ها یا کروموزوم‌های نسل حاضر، سبب ایجاد جمعیت نسل بعد خواهد شد. از جمعیت تولید شده نیز برای تولید جمعیت نسل بعد از آن (از طریق عملگرهای تولید مثل (انتخاب)، ترکیب و جهش) استفاده می‌شود. این چرخه در نسل‌های بعدی ادامه پیدا می‌کند و در هر نسل، الگوریتم ژنتیک یک قدم به تولید جواب بهینه سراسری نزدیک‌تر خواهد شد. مقدار برازندگی هر یک از رشته‌ها یا کروموزوم‌های جمعیت جدید نیز از طریق قرار دادن مقادیر متغیرها در تابع برازندگی و کد گشایی آن‌ها به دست می‌آید. مقادیر به دست آمده، برازندگی هر یک از رشته‌ها یا کروموزوم‌های جمعیت نسل‌های جدید را نمایش خواهند داد. محاسبه برازندگی هر یک از رشته‌ها یا کروموزوم‌ها، پایان یک حلقه از الگوریتم ژنتیک خواهد بود که به آن «نسل» (Generation) گفته می‌شود. در هر نسل، در صورتی که برازندگی یک کروموزوم یا جواب بهبود پیدا کند، به عنوان یکی از جواب‌های خوب شناخته خواهد شد. این کار تا زمانی که الگوریتم ژنتیک به جواب بهینه سراسری همگرا شود ادامه پیدا می‌کند.



نمایش عملکرد الگوریتم ژنتیک در بهینه‌سازی توابع

برای اینکه با عملکرد الگوریتم ژنتیک در همگرایی به جواب بهینه بهتر آشنا شویم، یک تابع دو متغیری ساده با استفاده از الگوریتم ژنتیک حل خواهد شد. گام‌های تفصیلی و جواب‌های حاصل شده در ادامه نمایش داده شده‌اند. مسأله کمینه‌سازی زیر را در بازه $0\le x1,x2\le 00\le x1,x2\le 0$ در نظر بگیرید:

f(x1,x2)=(x21+x2−11)+(x1+x22−7)2f(x1,x2)=(x12+x2−11)+(x1+x22−7)2
جواب بهینه سراسری برای مسأله بهینه‌سازی فوق در نقطه (3,2)T(3,2)T حاصل می‌شود. در این نقطه، مقدار تابع هدف نمایش داده شده برابر با صفر خواهد بود.

مرحله اول

برای حل این مسأله توسط الگوریتم ژنتیک، از کدبندی باینری برای نمایش متغیرهای x1x1 و x2x2 استفاده می‌شود. در این مسأله، از ۱۰ بیت برای نمایش هر متغیر استفاده شده است؛ در نتیجه، از آنجایی که مسأله دو متغیری است، طول رشته یا کروموزوم برابر با ۲۰ خواهد بود. با انتخاب ده بیت برای نمایش متغیرهای مسأله، الگوریتم ژنتیک به دقتی برابر با (6−0)(210−1)(6−0)(210−1)0 یا ۰٫۰۰۶ در بازه (۶ ، ۰) دست خواهد یافت. احتمال ترکیب (pcpc) و احتمال جهش (pmpm) به ترتیب برابر با ۰٫۸ و ۰٫۰۵ در نظر گرفته شده است. اندازه جمعیت برابر با ۲۰ و تعداد نسل‌ها نیز برابر با ۳۰ خواهد بود. از یک مولد تصادفی تعبیه شده برای تولید مقادیر تصادفی و از یک «نمونه‌گیری تصادفی بدون جای‌گذاری» (Stochastic Sampling Without Replacement) به عنوان عملگر انتخاب یا تولید مثل استفاده می‌شود. در مرحله بعد، برازندگی تمامی رشته‌ها یا کروموزوم‌های موجود در جمعیت سنجیده می‌شود. در این مثال کاربردی، تنها برازندگی رشته یا کروموزوم اول محاسبه می‌شود.

مرحله دوم

در این مرحله، برازندگی تمامی رشته‌ها یا کروموزوم‌های موجود در جمعیت سنجیده می‌شود. این کار از طریق کد گشایی رشته‌ها یا کروموزوم‌ها انجام می‌شود. زیر رشته اول (۱۰۰۰۰۰۱۱۱۰) با استفاده از تابع زیر به مقداری برابر با (29+23+22+21)(29+23+22+21) یا ۵۲۵ کد گشایی می‌شود. βΣj=0γj2jΣj=0βγj2j
بنابراین، با استفاده از تابع زیر، مقدار پارامتر متناظر، برابر با 5251023⋆(6−0)+52510230⋆(6−0)+0 یا ۳٫۰۹ خواهد بود.
xi=xli+xui−xli2β−1∑βj=0γj2jxi=xil+xiu−xil2β−1∑j=0βγj2j
همچنین، فرض کنید که زیر رشته دوم (۰۰۰۱۱۱۰۰۰۰) به مقداری برابر با ۱۲ کد گشایی می‌شود. مقدار پارامتر متناظر با این زیر رشته، 1121023⋆(6−0)+11210230⋆(6−0)+0 یا ۰٫۶۶ خواهد شد. بنابراین اولین زیر رشته، برابر با نقطه x0=(3.09,0.66)Tx0=(3.09,0.66)T در فضای جستجوی مسأله خواهد بود. با جای‌گذاری این مقادیر در تابع هدف، مقدار تابع هدف برابر با ۱۲٫۸۱۶ برای رشته اول در نقطه x0=(3.09,0.66)Tx0=(3.09,0.66)T تولید خواهد شد.
با توجه به اینکه مسأله بهینه‌سازی تعریف شده در اصل یک مسأله کمینه‌سازی است، مقدار برازندگی رشته اول به شکل F(x(1))=1.0(1.0+12.816)=0.072F(x(1))=1.0(1.0+12.816)=0.072 محاسبه می‌شود. از مقدار برازندگی محاسبه شده، در عملیات تولید مثل (عملگر انتخاب یا تولید مثل) استفاده می‌شود. جدول زیر، جزئیات مرتبط با رشته‌ها یا کروموزوم‌های دوم تا دهم جمعیت را نشان می‌دهد.

جزئیات مربوط به رشته‌های دوم (شماره ۱) تا دهم (شماره ۹) تا مرحله انتخاب یا تولید مثل کروموزوم‌ها

شماره	زیر رشته اول	زیر رشته دوم	پارامتر اول	پارامتر دوم	مقدار تابع هدف	مقدار تابع برازندگی	تعداد کپی‌ها در نسل بعد
1	1100110000	0101111110	4.7	2.2	207.9	0.005	1
2	0010110110	0000110000	1.0	0.2	126.0	0.008	1
3	0100001001	1000011011	1.5	3.1	50.0	0.020	2
4	1011001101	0100111010	4.2	1.8	73.0	0.014	1
5	1001111001	0111100011	3.7	2.8	53.9	0.018	1
6	1000111011	1101111111	3.3	5.2	601.2	0.002	0

7	0011100111	1001110100	1.3	3.6	92.7	0.011	1
8	0110001101	1011111111	2.3	4.5	243.4	0.004	1
9	0010101011	1000110000	1.0	3.2	67.9	0.014	1

مرحله سوم

از آنجایی که الگوریتم ژنتیک هنوز به حد بالای تعداد نسل‌های لازم برای همگرایی به جواب بهینه نرسیده است، کنترل اجرای الگوریتم به مرحله بعد داده می‌شود.

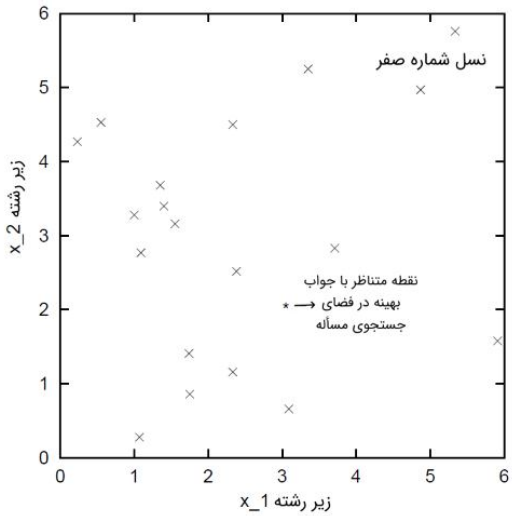
مرحله چهارم

در این مرحله، عملیات انتخاب یا تولید مثل با استفاده از نمونه‌گیری تصادفی بدون جای‌گذاری انجام می‌شود. به عبارت دیگر، تعداد کپی‌های هر رشته یا کروموزم در مخزن Mating Pool، بر اساس برازندگی آن‌ها مشخص می‌شود. به عنوان نمونه، رشته یا کروموزم هفتم (شماره ۶) تعداد ۰ کپی، رشته یا کروموزم دوم تعداد ۱ کپی و رشته یا کروموزم چهارم تعداد ۲ کپی به آن‌ها در مخزن Mating Pool اختصاص داده خواهد شد. دقت کنید از آنجایی که مقدار تابع هدف رشته یا کروموزوم هفتم (شماره ۶) بسیار بالاتر از دیگر رشته‌ها است، تعداد ۰ کپی به آن در مخزن Mating Pool اختصاص داده شده است؛ ولی رشته یا کروموزوم چهارم، تعداد ۲ کپی به آن در مخزن Mating Pool اختصاص داده خواهد شد. به عبارت دیگر، دو کپی از این رشته یا کروموزوم در جمعیت نسل بعد حضور خواهد داشت. با این کار فرایند تولید مثل یا انتخاب به اتمام می‌رسد. در ادامه، از عملگرهای ترکیب و جهش برای دستکاری ژنی کروموزوم‌های والدین و تولید کروموزوم‌های فرزند استفاده می‌شود. دقت داشته باشید پیش از اینکه عملگرهای جهش و ترکیب روی کروموزوم‌ها اعمال شوند، کروموزوم‌هایی که مقدار تابع هدف آن‌ها بالا است از جمعیت حذف می‌شوند. با این حال، پس از انجام عملیات جهش و ترکیب روی کروموزوم‌های انتخاب شده، برازندگی برخی از کروموزوم‌ها بدتر (نظیر کروموزوم شماره ۸) و برازندگی برخی دیگر از آن‌ها بهتر می‌شوند (نظیر کروموزوم شماره ۱). جزئیات مرتبط با رشته‌ها یا کروموزوم‌های اول تا دهم جمعیت، پس از انجام عملیات جهش و ترکیب روی آن‌ها در جدول زیر نمایش داده شده است.

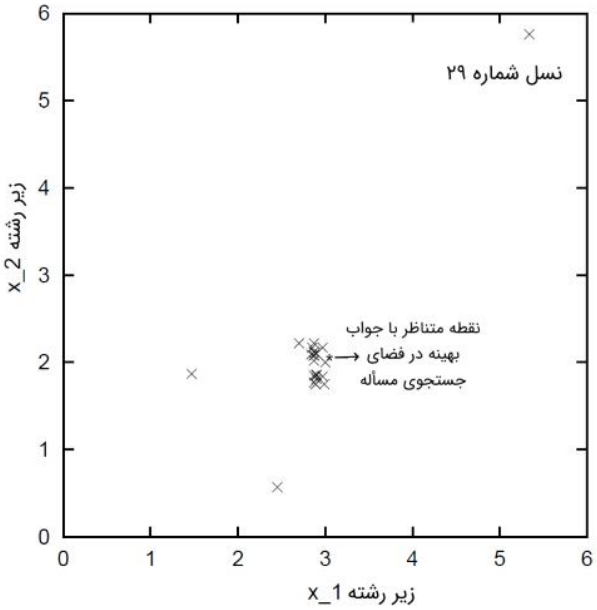
جزئیات مربوط به رشته‌های دوم (شماره ۱) تا دهم (شماره ۹) پس از عملیات ترکیب و جهش روی جمعیت

شماره	زیر رشته اول	زیر رشته دوم	پارامتر اول	پارامتر دوم	مقدار تابع هدف	مقدار تابع برازندگی
1	011010000	00001111110	2.4	0.7	34.6	0.028
2	001011011	11001110000	0.5	1.0	122.5	0.008
3	010000100	10000011001	1.0	3.6	94.0	0.011
4	101000111	11010111110	1.5	0.1	100.6	0.010
5	100111101	10111100011	3.8	4.1	252.2	0.004
6	001110101	11001000011	3.7	2.8	55.0	0.018
7	11001000011	00010110100	1.3	3.4	67.4	0.015
8	111000110	10110111101	5.3	2.6	427.7	0.002
9	001110111	00101010000	1.4	1.9	53.0	0.018

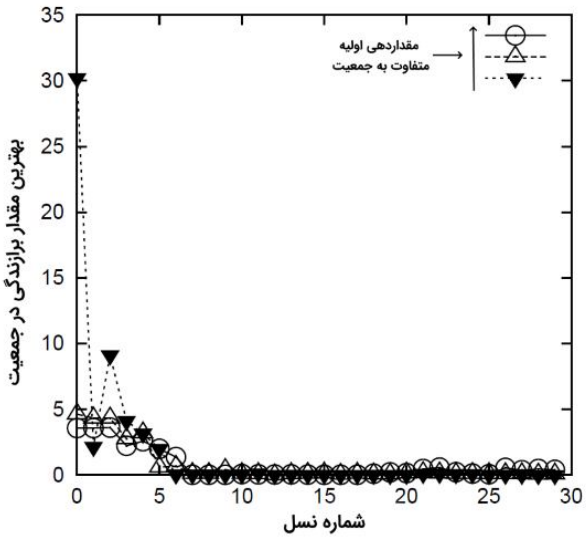
پیشرفت‌های حاصل شده پس از نسل اول معنادار نخواهد بود. با این حال، با مقایسه جواب‌های به دست آمده پس از نسل اول و جواب‌های به دست آمده پس از نسل سی‌اُم، به راحتی می‌توان نتیجه گرفت که پیشرفت‌های بسیار خوبی در زمینه همگرایی به جواب بهینه حاصل شده است.



موقعیت جواب‌های کاندید (جمعیت اولیه) در فضای جستجوی مسأله



همگرایی سریع جواب‌های کاندید به جواب بهینه در الگوریتم ژنتیک پس از ۳۰ نسل

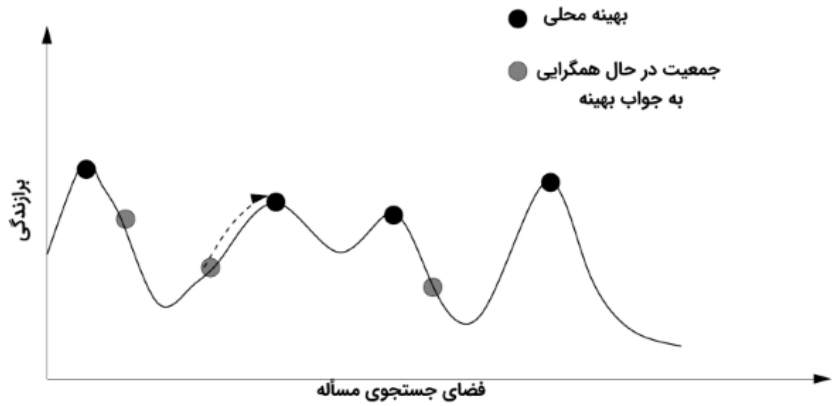


تفاوت الگوریتم ژنتیک و الگوریتم‌های بهینه‌سازی و جستجوی سنتی

الگوریتم ژنتیک، تفاوت‌های بنیادی با الگوریتم‌های جستجو و بهینه‌سازی سنتی دارد. مهم‌ترین این تفاوت‌ها عبارتند از:

الگوریتم ژنتیک از طریق کدبندی مجموعه جواب‌های کاندید، توابع را بهینه و مسائل بهینه‌سازی را حل می‌کند. در حالی که الگوریتم‌های بهینه‌سازی و جستجوی سنتی، از مجموعه جواب‌های کاندید برای بهینه‌سازی استفاده می‌کنند.

الگوریتم ژنتیک، مجموعه‌ای از جواب‌های کاندید را برای یافتن جواب بهینه جستجو می‌کند. در حالی که الگوریتم‌های بهینه‌سازی و جستجوی سنتی، از همان ابتدا به دنبال یک جواب واحد هستند. الگوریتم ژنتیک، از اطلاعات مرتبط با تابع برازندگی برای حل مسائل بهینه‌سازی بهره می‌برد. در حالی که الگوریتم‌های بهینه‌سازی و جستجوی سنتی، از مشتق تابع هدف و دیگر اطلاعات کمکی استفاده می‌کنند. الگوریتم ژنتیک از قوانین گذار «احتمالی» (Probabilistic) برای بهینه‌سازی و جستجو استفاده می‌کنند. در حالی که الگوریتم‌های بهینه‌سازی و جستجوی سنتی، از قوانین «قطعی» (Deterministic) بهره می‌گیرند.



کاربردهای الگوریتم ژنتیک

در صورتی که بتوانید جواب‌های کاندید یک مسأله را در قالب کروموزوم‌ها کدبندی کنید، به راحتی خواهید توانست از الگوریتم ژنتیک برای حل مسأله و مقایسه عملکرد (برازندگی) نسبی جواب‌های بهینه حاصل شده استفاده کنید. نمایش دقیق و مؤثر از متغیرهای مسأله و پیاده‌سازی ساز و کارهای با معنی برای ارزیابی برازندگی جواب‌های کاندید، مهم‌ترین عوامل در موفقیت در کاربردهای تولید شده از الگوریتم ژنتیک است. نقطه قوت الگوریتم ژنتیک، در سادگی و ظرافت آن‌ها به عنوان یک الگوریتم جستجوی قدرتمند و قدرت آن‌ها در کشف سریع جواب‌های خوب برای مسائل سخت و «با ابعاد بالا» (High Dimensional)

نهفته است. الگوریتم ژنتیک زمانی می‌توانند مفید و مؤثر واقع شود که: فضای جستجوی مسأله بزرگ، پیچیده و دارای ساختار بندی ضعیف باشد. دانش دامنه نایاب باشد و یا دانش خبره، جهت محدود کردن فضای جستجو، به سختی کدبندی شود. هیچ‌گونه تحلیل ریاضی در دسترس نباشد. روش‌های جستجوی سنتی با شکست مواجه شوند.

از الگوریتم‌های ژنتیک، جهت حل مسأله و مدل‌سازی استفاده می‌شود. امروزه، از الگوریتم‌های ژنتیک و مشتقات آن، در دامنه وسیعی از مسائل علمی و مهندسی استفاده می‌شود. مهم‌ترین کاربردهای الگوریتم ژنتیک عبارتند از:

بهینه‌سازی: دامنه وسیعی از مسائل بهینه‌سازی نظیر «بهینه‌سازی عددی» (Numerical Optimization) و «بهینه‌سازی ترکیبیاتی» (Combinatorial Optimization).

«برنامه‌نویسی خودکار» (Automatic Programming): برنامه‌نویسی سیستم‌های کامپیوتری برای انجام وظایف خاص و طراحی ساختارهای محاسباتی نظیر «اتوماتای سلولی» (Cellular Automata) و «شبکه‌های مرتب‌سازی» (Sorting Networks). «یادگیری ماشین» (Machine Learning): «دسته‌بندی» (Classification)، «پیش‌بینی» (Prediction)، طراحی «شبکه‌های عصبی مصنوعی» (Artificial Neural Networks) و سایر موارد.

مدل‌های «سیستم ایمنی» (Immune Systems): برای مدل‌سازی جنبه‌های مختلف سیستم ایمنی طبیعی، از جمله برخی جهش‌ها در طول دوره حیات موجودات و اکتشاف خانواده‌های «چندژنی» (Multi-Genes) در طول فرایند تکامل.

مدل‌های اقتصادی: مدل‌سازی فرایند نوآوری، توسعه استراتژی‌های مناقصه و ظهور بازارهای اقتصادی.

< طراحی خودکار آنتن استفاده شده در ماموریت‌های فضایی ناسا (NASA’s Space Technology) توسط الگوریتم‌های تکاملی کدهای پیاده‌سازی الگوریتم ژنتیک در زبان‌های برنامه‌نویسی مختلف

در این بخش، کدهای پیاده‌سازی الگوریتم ژنتیک برای مدل‌سازی و حل «الگوریتم راسو» (Weasel algorithm) [+] نمایش داده شده است. هدف این الگوریتم این است که نشان دهد تغییرات تصادفی در ژن‌ها و ترکیب آن‌ها با یک مکانیزم انتخاب غیر تصادفی، منجر به تولید نتایجی متفاوت و بهتر از شانس خالص در فرایندهای تکاملی خواهد شد. مشخصات این مسأله به شکل زیر است:

جمعیت اولیه: یک آرایه ۲۸ عنصری متشکل از الفبای ABCDEFGHIJKLMNOPQRSTUVWXYZ.

هدف نهایی: تبدیل رشته ورودی یا جمعیت اولیه به رشته METHINKS IT IS LIKE A WEASEL.

تابع برازندگی: تابعی که شباهت آرگومان ورودی را به رشته نهایی یا هدف نهایی مشخص می‌کند.

عملگرهای تکاملی (پیاده‌سازی عملگر انتخاب و جهش ضروری است، ولی پیاده‌سازی عملگر ترکیب اختیاری است)

جمع‌بندی

الگوریتم‌های ژنتیک، به راحتی در دامنه وسیعی از مسائل، از جمله مسائل بهینه‌سازی نظیر «مسأله فروشنده دوره‌گرد» (Travelling Salesman Problem) و مسائل مرتبط با «برنامه‌ریزی» (Scheduling)، می‌توانند مورد استفاده قرار بگیرند. نتایج حاصل از الگوریتم ژنتیک برای برخی از مسائل ممکن است بسیار خوب و برای برخی دیگر از مسائل، ممکن است بسیار ضعیف باشد. اگر تنها از عملگر جهش در الگوریتم ژنتیک استفاده شود، الگوریتم بسیار کند می‌شود. استفاده از عملگر ترکیب، سبب سرعت بخشیدن به فرایند همگرایی در الگوریتم ژنتیک خواهد شد.

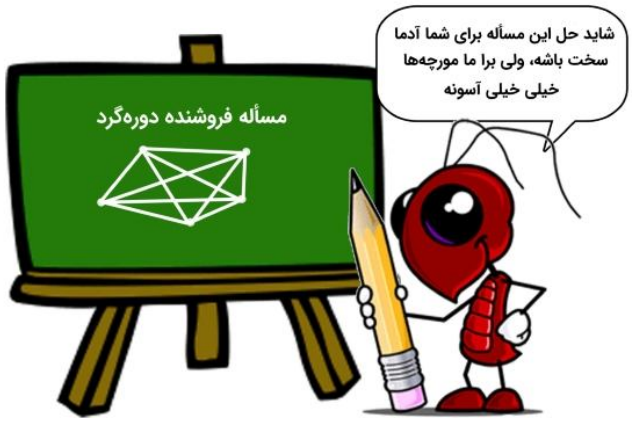
مشکل بهینه محلی در الگوریتم ژنتیک بسیار شایع است. در حالتی که الگوریتم در بهینه محلی قرار بگیرد، استفاده از عملگر جهش سبب می‌شود تا جواب‌های بدتری نسبت به بهینه محلی تولید شود. با این حال، برای فرار از دام بهینه محلی می‌توان از عملگر ترکیب استفاده کرد. البته، از آنجایی که جهش یک فرایند کاملا تصادفی است، این امکان نیز وجود دارد که جهش‌های بزرگی در کروموزوم‌ها ایجاد شوند و آن کروموزوم‌ها یا جواب‌های کاندید از بهینه محلی خارج شوند.

در آینده، ممکن است شاهد توسعه الگوریتم‌های تکاملی باشید که برای حل مسائل خاص مورد استفاده قرار بگیرند. این امر، نقض اصول پیاده‌سازی الگوریتم ژنتیک است که با هدف استفاده همه منظوره و بدون در نظر گرفتن دامنه مسائل قابل حل، توسعه داده

شده‌اند. با این حال، چنین گرایشی در توسعه الگوریتم‌های تکاملی ممکن است منجر به تولید سیستم‌های تکاملی به مراتب قوی‌تر شود. الگوریتم کلونی مورچگان یا در حقیقت «بهینه‌سازی کلونی مورچگان» (Ant Colony Optimization) همانطور که از نام آن مشخص است، بر پایه رفتار طبیعی کلونی‌های مورچگان و مورچگان کارگر شاغل در آن‌ها بنا نهاده شده است. فرآیند یافتن منابع غذایی در کلونی مورچگان بسیار بهینه است. زمانی که مورچه‌ها عملیات کاوش برای یافتن منابع غذایی را آغاز می‌کنند، به طور طبیعی یک مسیر «منطقی» و «بهینه» از آشیانه خود به منابع غذایی پیدا می‌کنند. به عبارت دیگر، جمعیت مورچگان به نحوی همیشه قادر هستند تا یک مسیر بهینه را برای تامین منابع غذایی مورد نیاز بیابند. شبیه‌سازی چنین رفتار بهینه‌ای، پایه و اساس بهینه سازی کلونی مورچگان را تشکیل می‌دهد. در این مطلب، الگوریتم کلونی مورچگان به طور کامل تشریح شده است. باید توجه داشت که نام دقیق این الگوریتم، بهینه سازی کلونی مورچگان است که توسط اغلب افراد به آن الگوریتم مورچگان یا الگوریتم کلونی مورچگان گفته می‌شود.

فهرست مطالب این نوشته پنهان کردن

دو مورچه را فرض کنید که در حال حرکت از آشیانه به منبع غذایی، از طریق دو مسیر کاملا متفاوت از هم هستند. مورچه‌ها در ضمن حرکت خود به سمت منبع غذایی، ردی از «فرومون» (Pheromone) در محیط منتشر می‌کنند که به‌طور طبیعی و با گذر زمان متلاشی می‌شود. مورچه‌ای که (به‌طور تصادفی) کوتاهترین مسیر به سمت منبع غذایی را انتخاب کرده، سفر برگشتی به سمت آشیانه را زودتر از دیگر مورچه‌ها آغاز می‌کند. در چنین حالتی، این مورچه در مسیر بازگشت به آشیانه، دوباره شروع به منتشر کردن فرومون در محیط می‌کند و از این طریق، رد فرومون به جا گذاشته در کوتاهترین مسیر را تقویت می‌کند.
مورچه‌های دیگر، به‌طور غریزی، قوی‌ترین مسیر فرومون موجود در محیط را دنبال و رد فرومون در این مسیر را تقویت می‌کنند. پس از گذشت مدت زمان مشخصی، نه تنها رد فرومون موجود در کوتاه‌ترین مسیر متلاشی نمی‌شود، بلکه، با انباشته شدن رد فرومون دیگر مورچه‌ها، بیش از پیش تقویت می‌شود. مسیری که قوی‌ترین رد فرومون در آن به جا گذاشته شده باشد، به مسیر پیش فرض برای حرکت مورچه‌ها از کلونی به منبع غذایی و برعکس تبدیل می‌شود.
روش بهینه‌سازی کلونی مورچگان، مدلی برای پیاده‌سازی روش‌های بهینه‌سازی ارائه می‌دهد. تاکنون، پیاده‌سازی‌های موفق متفاوتی از این روش بهینه‌سازی ارائه شده است. الگوریتم‌هایی نظیر «سیستم مورچگان» (Ant System)، «سیستم کلونی مورچگان» (Ant Colony System) و «سیستم مورچگان Min-Max» از جمله مهم‌ترین و موفق‌ترین پیاده‌سازی‌های صورت گرفته از این روش بهینه‌سازی محسوب می‌شوند.



مقدمه‌ای بر الگوریتم کلونی مورچگان

در این مطب سعی بر آن است تا ویژگی‌ها و مراحل پیاده‌سازی روش الگوریتم کلونی مورچگان شرح داده شود؛ روشی «فرا اکتشافی» (Metaheuristic) که از رفتار بهینه مورچه‌ها الهام گرفته شده است. الگوریتم مورچگان، روشی بسیار قدرتمند برای حل «مسائل بهینه‌سازی ترکیبیاتی» (Combinatorial Optimization Problems) محسوب می‌شود.
الگوریتم‌های مشتق شده از الگوریتم کلونی مورچگان، زیر مجموعه‌ای از روش‌های «هوش ازدحامی» (Swarm Intelligence) هستند. این دسته روش‌ها، حوزه تحقیقاتی و مطالعاتی به شمار می‌آیند که به مطالعه الگوریتم‌های الهام گرفته شده از مفهوم «رفتارهای ازدحامی» (Swarm Behaviors) می‌پردازند. الگوریتم‌های هوش ازدحامی از مجموعه‌ای از موجودیت‌های فردی ساده تشکیل شده‌اند که از طریق «خودسازماندهی» (Self-Organizing) با یکدیگر تعامل و همکاری می‌کنند. منظور از خودسازماندهی، نبود سیستم کنترل مرکزی برای کنترل و ایجاد هماهنگی میان اعضای یک سیستم هوش ازدحامی است.
گذار از ویژگی‌های زیستی به الگوریتم‌های کامپیوتری پیش از آنکه الگوریتم کلونی مورچگان مورد بررسی قرار داده شود، لازم است برخی از مفاهیم مرتبط با فرآیندهای زیستی موجود در مورچه‌ها مطالعه شود؛ مفاهیمی که سبب درک بهتر خواننده از رفتارهای بهینه این موجود در هنگام تعامل با محیط می‌شوند. در ابتدا، برخی از مشاهدات به دست آمده از مطالعه رفتار مورچه‌ها

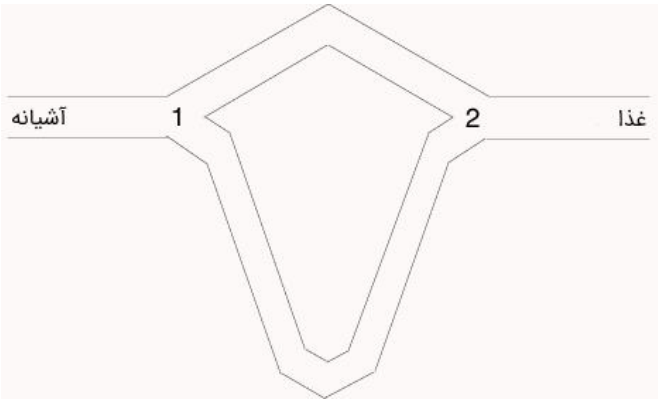
در طبیعت مورد بررسی قرار داده می‌شود. در ادامه نشان داده خواهد شد که این مشاهدات چگونه الهام‌بخش تولید الگوریتم کلونی مورچگان شده‌اند.

رفتار مورچگان

حشره‌شناس فرانسوی، پیر پائول گراس (Pierre-Paul Grassé) اولین محققی بود که رفتار اجتماعی حشرات را مورد بررسی قرار داد. ایشان در بازه سال‌های اولیه دهه ۴۰ میلادی تا اواخر دهه ۵۰ میلادی، به مطالعه و تحقیق در مورد رفتار «موریانه‌ها» (Termites) پرداخت. در جریان این مطالعات، او به این موضوع پی برد که این گونه حشره‌ای می‌تواند به سیگنال‌های «محرک» (Stimuli) محیطی واکنش نشان دهد. چنین محرک‌هایی، به نوبه خود، نوعی واکنش برنامه‌نویسی شده ژنتیکی در این موجودات را فعال می‌کند. این دانشمند، پی برد که تاثیرات این واکنش‌های ژنتیکی (از سوی موریانه) نه تنها می‌تواند به عنوان محرکی جدید برای خود موریانه عمل کند (ایجاد واکنش‌های متعاقب دیگر در پاسخ به محرک‌های محیطی)، بلکه دیگر موریانه‌های موجود در کلونی را نیز تحت تاثیر قرار می‌دهد و واکنش‌های خاصی را در آن‌ها برمی‌انگیزد. ایشان از اصطلاح «نشانه‌ورزی» (Stigmergy) برای توصیف این نوع ارتباط غیر مستقیم میان موریانه‌ها استفاده کرد. در این ارتباط کد شده ژنتیکی، گونه‌های کارگری حشرات از طریق عملکرد حاصل از تعامل با محیط تحریک می‌شوند و واکنش‌های ژنتیکی خاصی از خود نشان می‌دهند.

این دسته از ارتباطات ژنتیکی و غیر مستقیم میان حشرات، از دو جهت با دیگر وسایل ارتباطی متفاوت است:

طبیعتِ فیزیکی و غیر نمادین اطلاعات مبادله شده میان حشرات: این اطلاعات، مترادف با تغییرات حالات محیطی-فیزیکی است که در اثر تعامل حشره با محیط صورت می‌پذیرد. طبیعتِ (ذات) محلی اطلاعات مبادله شده: این دسته از اطلاعات تنها توسط حشراتی قابل دسترسی است که از محیط حاوی این اطلاعات بازدید کرده باشند یا در همسایگی بلافاصل این محیط قرار گرفته باشند. به عبارت دیگر، چنین اطلاعاتی توسط تمامی حشرات موجود در کلونی قابل دریافت نیست و یک فاکتور جغرافیایی در تبادل اطلاعات میان حشرات مؤثر است. نمونه‌هایی از چنین ارتباطات غیر مستقیمی در کلونی مورچگان نیز قابل مشاهده است. در بسیاری از گونه‌های مورچگان، زمانی که مورچه‌ها از کلونی خود به سمت منابع غذایی حرکت می‌کنند، ماده‌ای را روی زمین منتشر می‌کنند که به آن فرومون گفته می‌شود. مورچه‌های دیگر قادر هستند تا فرومون منتشر شده در محیط را ببویند. وجود فرمون در محیط، مسیر انتخابی توسط مورچگان را تحت تاثیر قرار می‌دهد. به عبارت دیگر، مورچه‌ها معمولا تمایل دارند مسیر حاوی فرومون غلیظ‌تر را دنبال کنند. فرومون منتشر شده، سبب ایجاد «رد فرومون» (Pheromone Trail) در محیط می‌شود. رد فرومون به مورچه‌ها این امکان را می‌دهد که بتوانند منابع غذایی مناسبی که پیش از این توسط دیگر مورچه شناسایی شده را پیدا کنند. در آزمایش جالبی که برای مطالعه بهتر این ویژگی مورچه‌ها طراحی شده بود، دانشمندان میان کلونی مورچگان و منابع غذایی، دو پل با طول‌های متفاوت قرار دادند. در ابتدا، جمعیت مورچه‌ها شروع به جستجو در محیط برای یافتن منابع غذایی کردند. مورچه‌هایی که به طور کاملا تصادفی مسیر کوتاه‌تر را انتخاب کرده بودند، سریع‌تر به منابع غذایی رسیدند. این دسته از مورچه‌ها، مسیر بازگشت به کلونی را سریع‌تر و در زمان کمتری پیمودند. در نتیجه، رد فرومون منتشر شده در مسیر کوتاه‌تر تقویت شد. از آن‌جا که مورچه‌ها مسیر حاوی فرومون با غلظت بیشتر را ترجیح می‌دهند، پس از مدت زمان خاصی جمعیت مورچگان، مسیر کوتاه‌تر را با احتمال بیشتری انتخاب می‌کردند. بنابراین، به کمک مکانیزم منتشر کردن و دنبال کرد رد فرومون در محیط، مورچه‌ها به سرعت به مسیر کوتاه‌تر همگرا شدند.



این آزمایش نتایج بسیار شگفت انگیزی برای محققان به ارمغان آورد. آن‌ها کشف کردند که مورچه‌ها از رفتار بهینه ازدحامی مبتنی بر «بازخورد مثبت» (Positive Feedback) برای یافتن کوتاه‌ترین مسیر از کلونی به منابع غذایی استفاده می‌کنند. به چنین مکانیزمی، «خودکاتالیز» یا «اتوکاتالیز» (Autocatalysis) نیز گفته می‌شود. در ادامه، توضیح داده خواهد شد که مطالعات انجام شده روی جمعیت مورچه‌ها، چگونه منجر به توسعه الگوریتم‌های بهینه‌سازی می‌شود. به‌عبارت دیگر، یک الگوریتم بهینه‌سازی الهام گرفته شده از رفتار بهینه مورچگان، باید چه ویژگی‌هایی داشته باشد.

ویژگی‌های یک الگوریتم بهینه‌سازی الهام گرفته شده از رفتار بهینه مورچگان

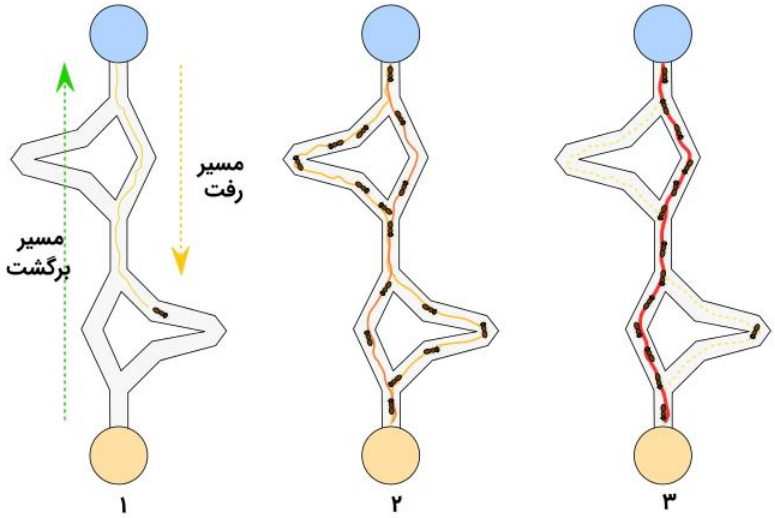
محققان، در پی مشاهده نتایج آزمایش «پل باینری» (Binary Bridge Experiment)، در صدد آن برآمدند تا مدلی ریاضی را برای توصیف رفتار بهینه مورچگان توسعه دهند. با فرض اینکه پس از گذشتن tt واحد زمانی از زمان آغاز آزمایش، m1m1 مورچه از پل اول و m2m2 مورچه از پل دوم استفاده کرده باشند، احتمال (p1p1) اینکه مورچه m+1m+1 پل اول را انتخاب کند برابر است با: p1(m+1)=(m1+k)h((m1+k)h+(m2+k)h)p1(m+1)=(m1+k)h((m1+k)h+(m2+k)h)

در این رابطه، پارامترهای kk و hh ، پارامترهای مورد نیاز برای «برازش» (Fitting) مدل روی داده‌های آزمایش است. همچنین، احتمال اینکه همان مورچه m+1m+1 پل دوم را انتخاب کند، از طریق رابطه p2(m+1)=1−p1(m+1)p2(m+1)=1−p1(m+1) به دست می‌آید. «شبیه‌سازی مونته کارلو» (Monte Carlo Simulation) انجام شده روی داده‌های آزمایش نشان می‌دهد که مدل توسعه داده شده با مقادیر پارامترهای k≈20k≈20 و h≈2h≈2 ، برازش بسیار خوب و مناسبی از داده‌های آزمایش تولید می‌کند. از این مدل ساده می‌توان برای طراحی «مورچه‌های مصنوعی» (Artificial Ants) استفاده کرد که مسائل بهینه‌سازی را به شیوه مشابه حل می‌کنند.

همان‌طور که پیش از این اشاره شد، ارتباطات غیر مستقیم میان مورچه‌های واقعی از طریق فرومون منتشر شده در محیط انجام می‌شود. به طور مشابه، مورچه‌های مصنوعی می‌توانند فرآیند انتشار فرومون در محیط را از طریق دستکاری مقادیر متغیرهای فرومون متناظر با متغیرهای مسأله شبیه‌سازی کنند. متغیرهای مسأله می‌توانند متناظر با حالاتی در نظر گرفته شوند که مورچه‌های مصنوعی در طول فرآیند تولید جواب بهینه در محیط، با آن‌ها برخورد می‌کنند. همچنین، براساس مدل ارتباطی مبتنی بر فرمون (مدل نشانه‌ورزی)، مورچه‌های مصنوعی به اطلاعات مرتبط با متغیرهای فرومون (متغیرهای مسأله) تنها دسترسی محلی خواهند داشت. بنابراین، مورچه‌های مصنوعی به دو روش قادر خواهند بود خود را با ویژگی‌های شاخص مدل ارتباطی ذکر شده (نشانه‌ورزی) سازگار کنند: ایجاد تناظر میان متغیرهای توصیف کننده حالات محیطی و مقادیر مختلف برای متغیرهای مسأله. فراهم کردن دسترسی (فقط) محلی به این متغیرها برای عوامل (مورچه‌های مصنوعی).

یکی دیگر از ویژگی‌های رفتاری مهم جمعیت مورچه‌ها که می‌تواند توسط مورچه‌های مصنوعی (عامل‌های روش بهینه‌سازی) مورد بهره‌وری قرار بگیرد، جفت‌شدگی (رابطه متقابل) میان مکانیزم خودکاتالیز و «ارزیابی ضمنی» (Implicit Evaluation) جواب‌ها است. منظور از ارزیابی ضمنی در رفتار بهینه مورچه‌ها این است که مسیرهای کوتاه‌تر، زودتر از مسیرهای بلندتر توسط مورچه‌ها

پیموده می‌شوند؛ در نتیجه، رد فرمون منتشر شده در این مسیر، سریع‌تر توسط مورچه‌های دیگر تقویت می‌شود. مسیر کوتاه‌تر برای مورچه‌های مصنوعی، معادل برازندگی بیشتر یا هزینه کمتر است. بنابراین، جفت‌شدگی میان مکانیزم خودکاتالیز و ارزیابی ضمنی، در صورتی که برای تولید عملگرهای تکاملی مناسب استفاده شود، مکانیزم فوق‌العاده قدرتمندی برای تضمین عملکرد بهینه در الگوریتم‌های بهینه‌سازی مبتنی بر جمعیت خواهد بود.



یکی از ویژگی‌های مهم روش الگوریتم کلونی مورچگان، شباهت زیاد میان مکانیزم‌های زیستی مورچگان و رفتارهای شبیه‌سازی شده از کلونی مورچگان در آن است. به عبارت دیگر، شبیه‌سازی و ترکیب ویژگی‌های رفتاری نظیر نشانه‌ورزی، ارزیابی ضمنی جواب‌ها و مکانیزم خودکاتالیز، پایه و اساس الگوریتم مورچگان را تشکیل می‌دهند. بنابراین، شباهت زیادی میان مورچه‌های مصنوعی و مورچه‌های واقعی وجود دارد.

شباهت‌های میان کلونی واقعی مورچگان و کلونی مصنوعی مورچگان (روش بهینه‌سازی)

جمعیت کلونی‌های واقعی و کلونی‌های مصنوعی مورچگان، از نمونه‌هایی تشکیل شده‌اند که برای رسیدن به یک هدف خاص با هم همکاری می‌کنند. یک کلونی، جمعیتی متشکل از عامل‌های ساده، مستقل و «ناهمگام» (Asynchronous) است که با یکدیگر برای یافتن جواب‌های بهینه یک مسأله بهینه‌سازی، همکاری و تعامل می‌کنند.

در کلونی مورچگان واقعی، مسأله، پیدا کردن منابع غذایی است. در کلونی مورچه‌های مصنوعی، مسأله، پیدا کردن جواب‌های بهینه برای مسأله بهینه‌سازی داده شده است. یک مورچه (چه واقعی و چه مصنوعی)، به تنهایی قادر به پیدا کردن جواب خواهد بود؛ ولی فقط همکاری میان نمونه‌ها از طریق مکانیزم «نشانه‌ورزی» (Stigmergy)، رسیدن به جواب بهینه را تضمین می‌کند.

وجود مکانیزمی معادل تبخیر فیزیکی اثر فرومون در روش‌های بهینه‌سازی مورچگان (همانند کلونی واقعی مورچگان)، به مورچه‌های مصنوعی این امکان می‌دهد که مسیرهای از پیش پیموده شده را فراموش کرده و بیشتر بر روی جهت‌های جستجوی جدید و امیدوارکننده تمرکز کنند. در کلونی مصنوعی مورچگان (همانند کلونی واقعی)، مورچه‌های مصنوعی، جواب‌های کاندید را با حرکت ترتیبی از یک حالت خاص مسأله به حالتی دیگر (تغییر مقادیر متغیرهای مسأله) تولید می‌کنند.

مورچه‌های واقعی، بر اساس غلظت فرومون در محیط و نیز یک سیاست تصمیم‌گیری تصادفی، مسیر حرکت خود را مشخص می‌کنند. مورچه‌های مصنوعی نیز مرحله به مرحله و از طریق تغییر مقادیر متغیرهای مسأله و نیز تصمیم‌گیری‌های تصادفی، اقدام به تولید جواب‌های کاندید برای مسأله مورد نظر می‌کنند.

تفاوت‌های میان کلونی واقعی مورچگان و کلونی مصنوعی مورچگان (روش بهینه سازی)

در کلونی واقعی، مورچه‌ها یا ماده‌ای به نام فرومون در محیط منتشر می‌کنند و یا به آن واکنش نشان می‌دهند؛ در کلونی مصنوعی، مورچه‌های مصنوعی، مقادیر عددی متناظر با حالت‌های مختلف مسأله (متغیرهای مسأله) را تغییر می‌دهند. به مقادیر عددی، «فرومون‌های مصنوعی» (Artificial Pheromones) و به دنباله‌ای از این مقادیر عددی، «رد مصنوعی فرومون» (Artifical Pheromone Trail) گفته می‌شود. برخلاف مورچه‌های واقعی، مورچه‌های مصنوعی در یک جهان گسسته فعالیت می‌کنند. آن‌ها، پی در پی و در یک فضای متناهی متشکل از حالات (متغیرهای) مسأله در حال حرکت و تولید جواب‌های کاندید هستند. فرآیند به‌روز رسانی مقادیر فرومون‌ها (نظیر انتشار و تبخیر فرومون) در کلونی مصنوعی مورچگان، دقیقا همانند کلونی واقعی مورچه‌ها انجام نمی‌شود. برخی مواقع، به‌روز رسانی فرومون، فقط توسط تعدادی از مورچه‌های مصنوعی و معمولا تنها پس از تولید یک جواب برای مسأله بهینه‌سازی انجام می‌شود. برخی از پیاده‌سازی‌های انجام شده از الگوریتم کلونی مورچگان، مکانیزم‌های اضافی را شامل می‌شوند که در جهان واقعی وجود ندارند (نظیر مکانیزم جستجوی محلی، مکانیزم عقب‌نشینی و سایر موارد).

روش فرا اکتشافی الگوریتم کلونی مورچگان

الگوریتم کلونی مورچگان، به عنوان یک روش فرا اکتشافی برای حل «مسائل بهینه‌سازی ترکیبیاتی» (Combinatorial Optimization Problems) طراحی شده و تاکنون، در موارد بسیاری، برای حل این دسته از مسائل استفاده شده است.

مدل فرومون برای حل مسائل بهینه‌سازی ترکیبیاتی

با فرض داشتن یک مسأله بهینه‌سازی ترکیبیاتی، اولین قدم در بهینه‌سازی با استفاده از الگوریتم کلونی مورچگان، تعریف یک مدل مناسب است. از این مدل، برای تعریف یکی از مهم‌ترین واحدهای الگوریتم کلونی مورچگان یا همان «مدل فرومون» (Pheromone Model) استفاده می‌شود. مدل لازم برای یک مسأله بهینه‌سازی ترکیبیاتی، به شکل زیر تعریف می‌شود:

AmodelP=(S,Ω,f)AmodelP=(S,Ω,f)

یک مدل فرومون برای یک مسأله بهینه‌سازی ترکیبیاتی شامل موارد زیر خواهد بود:

یک فضای جستجو به نام SS ، که روی مجموعه‌ای متناهی از متغیرهای گسسته تصمیم و مجموعه‌ای از قیدهای تعریف شده روی متغیرهای مسأله به نام ΩΩ تعریف می‌شود.

یک تابع هدف f:S→R+0f:S→R0+ که قرار است کمینه شود. به این نکته باید توجه کرد که کمینه‌سازی یک تابع هدف مثل ff ، دقیقا برابر با بیشینه‌سازی تابع هدف −f−f خواهد بود. در نتیجه، هر مسأله بهینه‌سازی ترکیبی، در قالب یک مسأله کمینه‌سازی قابل بازنویسی خواهد بود.

فضای جستجو به این شکل تعریف می‌شود: مجموعه‌ای از متغیرهای تصمیم گسسته (متغیرهای مسأله) XiXi ، i=1,...,ni=1,...,n ، با مقادیر {vji∈Di={v1i,.....,v|(Di)|i}vij∈Di={vi1,.....,vi|(Di)|} داده شده است. مقداردهی متغیرهای مسأله (اختصاص دادن مقدار vjivij به متغیر XiXi)، توسط نماد Xi←vjiXi←vij نمایش داده می‌شود. یک جواب کاندید s∈Ss∈S ، به یک مقداردهی کامل به متغیرهای مسأله گفته می‌شود که در آن، هر متغیر تصمیم مقادیری دارد که تمام قیدهای تعریف شده در مجموعه ΩΩ برای یک مسأله بهینه‌سازی ترکیبیاتی داده شده را ارضا می‌کند.

اگر مجموعه ΩΩ تهی باشد، به PP مسأله بهینه‌سازی ترکیبی «نامقید» (Unconstrained) گفته می‌شود؛ در غیر این صورت، به آن مسأله بهینه‌سازی «مقید» (Constrained) گفته می‌شود.

یک جواب کاندید s∗∈Ss∗∈S ، یک «جواب بهینه سراسری» (Global Optimum Solution) برای مسأله بهینه‌سازی ترکیبیاتی داده شده است، اگر و تنها اگر شرط f(s∗)≤f(s)∀s∈Sf(s∗)≤f(s)∀s∈S را ارضا کند. مجموعه تمامی جواب‌های بهینه سراسری توسط نماد S∗⊆SS∗⊆S نمایش داده می‌شوند. حل یک مسأله بهینه‌سازی ترکیبیاتی، مستلزم پیدا کردن حداقل یک جواب بهینه s∗∈S∗s∗∈S∗ است.

از مدل تعریف شده بالا، برای توسعه مدل فرومون در الگوریتم کلونی مورچگان استفاده می‌شود. متغیر تصمیم مقداردهی اولیه شده Xi=vjiXi=vij (به عبارت دیگر، یک مقدار مانند vjivij از دامنه DiDi به متغیر XiXi اختصاص داده می‌شود)، یک

«مؤلفه جواب کاندید» (Candidate Solution Component) نامیده می‌شود و به وسیله نماد $c_{i,j}$ نمایش داده می‌شود. مجموعه تمامی مؤلفه‌های جواب کاندید ممکن، توسط CC نمایش داده می‌شوند. سپس، هر پارامتر «رد فرومون» (Pheromone Trail)، T_{ij} ، متناظر با یکی از مؤلفه‌های جواب کاندید $c_{i,j}$ در نظر گرفته می‌شود. مجموعه تمامی پارامترهای رد فرومون، توسط TT نمایش داده می‌شوند. مقدار یک پارامتر رد فرومون T_{ij} توسط نماد τ_{ij} نمایش داده شده و «مقدار فرومون» (Pheromone Value) نامیده می‌شود.

مقادیر فرومون، در طی فرآیند جستجو، توسط الگوریتم کلونی مورچگان استفاده و به روز رسانی می‌شوند. همچنین، این پارامترها امکان مدل‌سازی توزیع احتمالی مؤلفه‌های مختلف جواب را به الگوریتم مورچگان می‌دهند. مقادیر فرومون و جواب کاندید (مقادیر متغیرهای مسأله)، عملا دو مفهوم کاملا به هم وابسته هستند. در طول فرآیند تکاملی، مقادیر فرومون برای جواب‌های کاندیدی که به جواب بهینه سراسری نزدیک‌تر باشند، افزایش می‌یابند و برعکس. در نتیجه، رد فرومون جواب‌های خوب، قوی‌تر می‌شود و جهت جستجو برای یافتن جواب بهینه برای مورچه‌های مصنوعی دیگر مشخص می‌شود.

نمایش فضای مسأله بهینه‌سازی در قالب گراف کاملا متصل

در الگوریتم کلونی مورچگان، مورچه‌های مصنوعی، یک جواب کاندید برای مسأله بهینه‌سازی ترکیبیاتی داده شده را از طریق پیمایش گرافی به نام «گراف ساختاری» (Construction Graph) تولید می‌کنند. این گراف، توسط $GC(V,E)$ نمایش داده می‌شود. یک گراف ساختاری کاملا متصل، از مجموعه‌ای از «رأس‌ها» (Vertices) و مجموعه‌ای از «یال‌ها» (Edges) تشکیل شده است. مجموعه مؤلفه‌های جواب کاندید CC ممکن است متناظر با مجموعه رأس‌ها VV در گراف GC یا مجموعه یال‌ها EE در گراف GC باشند.

مورچه‌ها، برای تولید جواب کاندید، در راستای یال‌های گراف، از یک رأس به رأس دیگر حرکت می‌کنند و از این طریق، به طور تدریجی یک «جواب کاندید جزئی» (Partial Candidate Solution) را تولید می‌کنند. علاوه بر این، مورچه‌ها در هنگام حرکت در محیط، روی یال‌ها یا رأس‌هایی که در گراف ساختاری پیمایش می‌کنند، مقدار مشخصی فرومون منتشر می‌کنند. مقدار فرومون منتشر شده در محیط $\Delta\tau$ ، به کیفیت جواب کاندید تولید شده بستگی دارد. مورچه‌های بعدی، از اطلاعات فرومون منتشر شده به عنوان راهنما برای پیدا کردن مناطق بهتر در فضای جستجو و حرکت به سمت آن‌ها استفاده می‌کنند. از یک سو، این مناطق، به احتمال زیاد حاوی جواب بهینه سراسری هستند و از سوی دیگر، در صورت حرکت مورچه‌های مصنوعی به سمت این مناطق، با سرعت بیشتری می‌توانند به جواب بهینه سراسری همگرا شوند.

شبه کد الگوریتم کلونی مورچگان در ادامه آمده است.

پارامترهای الگوریتم کلونی مورچگان تنظیم شده و ردهای فرومون مقداردهی اولیه می‌شوند.

تا زمانی که شرط توقف ارضا نشده باشد:

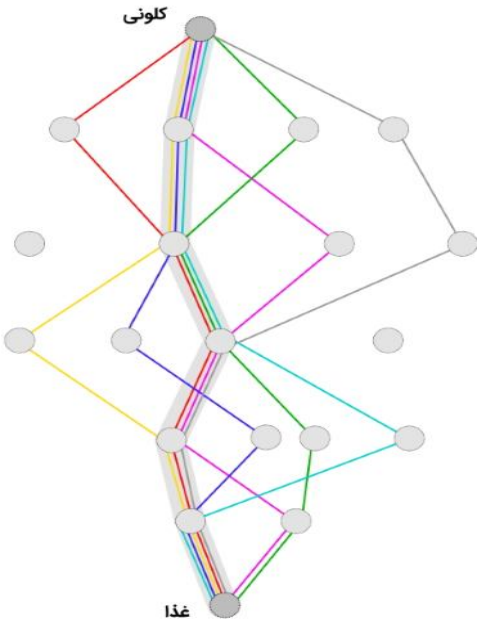
مرحله اول یا مرحله «تولید جواب‌های کاندید» (Construct Ant Solution) را شروع کن.

مرحله دوم یا مرحله «جستجوی محلی جواب‌ها» (Local Search) را شروع کن. در این مرحله، از جواب‌های بهینه محلی استفاده می‌شود تا مشخص شود کدام یک از فرومون‌ها باید به روز رسانی شوند. این مرحله اختیاری است و در برخی از پیاده‌سازی‌های انجام شده از الگوریتم کلونی مورچگان وجود ندارد.

مرحله سوم یا مرحله «به روز رسانی فرومون» (Pheromone Update) را انجام بده.

در صورتی که شرط توقف ارضا شده باشد، اجرای الگوریتم را متوقف کن؛ در غیر این صورت، مراحل را مجددا انجام بده.

الگوریتم کلونی مورچگان از دو بخش تشکیل شده است. در بخش اول، مقادیر پارامترهای مسأله تنظیم و متغیرهای جمعیت اولیه مقداردهی می‌شوند. بخش دو شامل یک حلقه تکرار است که سه مرحله الگوریتم کلونی مورچگان را اجرا می‌کند. در هر حلقه از الگوریتم کلونی مورچگان، جواب‌های کاندید توسط تمامی مورچه‌های مصنوعی ساخته می‌شوند. جواب‌های تولید شده، از طریق یک الگوریتم جستجوی محلی بهبود می‌یابند. در مرحله آخر، فرومون‌ها به روز رسانی می‌شوند.



در ادامه، سه مرحله اصلی الگوریتم کلونی مورچگان با جزئیات بیشتری توضیح داده می‌شود.

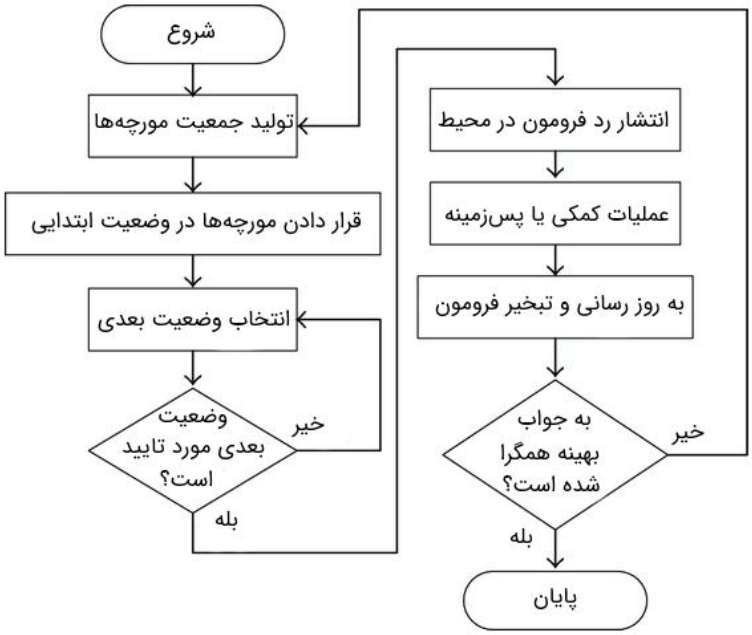
مرحله اول: «تولید جواب‌های کاندید» (Construct Ant Solution)

در این مرحله، مجموعه‌ای از mm مورچه مصنوعی، جواب‌های کاندید مسأله بهینه‌سازی ترکیبیاتی را با استفاده از عناصر یک مجموعه متناهی از «مؤلفه‌های جواب کاندید» در دسترس C={cij},i=1,....,n,j=1,.....,|Di|C={cij},i=1,....,n,j=1,.....,|Di| تولید می‌کنند. مرحله تولید جواب‌های کاندید با تولید یک جواب کاندید جزئی sp=∅sp=∅ آغاز می‌شود. سپس، در گام‌های بعدی از تولید جواب‌های کاندید، جواب کاندید جزئی تولید شده spsp ، با اضافه کردن یک مؤلفه از مجموعه «همسایگان امکان‌پذیر» (Feasible Neighbors) N(sp)⊆CN(sp)⊆C ، گسترده‌تر می‌شود.

فرآیند تولید جواب‌های کاندید را می‌توان به عنوان یک مسیر در گراف ساختاری GC=(V,E)GC=(V,E) در نظر گرفت. به عبارت دیگر، منظور از گسترش جواب بهینه، مشخص کردن مسیرهای حرکتی ممکن برای مورچه مصنوعی در گراف ساختار مدل فرومون است. از طریق چنین روشی، مناطق همسایگی جواب کاندید جزئی جستجو می‌شود تا بهترین مسیر در جهت جواب بهینه سراسری مشخص شود. مسیرهای مجاز در گراف GCGC ، به طور ضمنی توسط مکانیزم تولید جواب تعریف می‌شوند. مکانیزم تولید جواب، مجموعه همسایگان امکان‌پذیر N(sp)⊆CN(sp)⊆C را نسبت به هر کدام از جواب‌های جزئی و به طور مجزا تعریف می‌کند. در هر مرحله از تولید جواب‌های کاندید، نحوه انتخاب مؤلفه‌های مجموعه همسایگان امکان‌پذیر برای گسترش جواب کاندید جزئی، به صورت کاملا احتمالی انجام می‌شود. قوانین لازم برای انتخاب یک مؤلفه از مجموعه همسایگان امکان‌پذیر، در پیاده‌سازی‌های مختلف از الگوریتم کلونی مورچگان، متفاوت از هم هستند. با این حال، شناخته شده‌ترین قانون، مربوط به الگوریتم «سیستم مورچگان» (Ant System) است:

p(cij|sp)=ταij.η(cij)β∑cil∈N(sp)ταil.η(cil)β,∀cij∈N(sp)p(cij|sp)=τija.η(cij)β∑cil∈N(sp)τilα.η(cil)β,∀ci j∈N(sp)j∈N(sp) در این رابطه، τijτij مقادیر فرومون متناظر با مؤلفه cijcij و η(.)η(.) ، تابعی است که در هر مرحله از تولید جواب کاندید، یک مقدار به اصطلاح «اکتشافی» (Heuristic) را به هر [مؤلفه] جواب کاندید cij∈N(sp)cij∈N(sp) اختصاص می‌دهد. مقادیر اکتشافی تولید شده توسط تابع η(.)η(.) ، «اطلاعات اکتشافی» (Heuristic Information) نامیده می‌شوند. همچنین، پارامترهای αα و ββ ، پارامترهایی با مقادیر مثبت هستند که مقدارشان، میزان اهمیت نسبی (وزن) اطلاعات فرومون (مقادیر متغیرهای یک جواب کاندید) و اطلاعات اکتشافی، در تولید مقدار احتمالی رابطه بالا را مشخص می‌کنند. این رابطه،

تعمیمی از مدل ریاضی ارائه شده در بخش‌های قبل، برای توصیف رفتار بهینه مورچگان در آزمایش «پل باینری» (Binary Bridge Experiment) محسوب می‌شود. فلوچارت ساده الگوریتم کلونی مورچگان در شکل زیر آمده است.



مرحله دوم: «جستجوی محلی جواب‌ها» (Local Search)

بسته به پیاده‌سازی انجام شده از الگوریتم کلونی مورچگان، پس از تولید جواب‌های کاندید و پیش از اینکه فرومون‌ها به روز رسانی شوند، ممکن است فرآیندهای اضافی برای تضمین عملکرد بهینه الگوریتم ضروری باشند. بنابراین، این مرحله، اختیاری است. طبیعت این فرآیندها، ازدحامی است. یعنی توسط فقط یک مورچه مصنوعی قابل انجام نیست. به این دسته از فرآیندها، «عملیات کمکی یا پس‌زمینه» (Daemon Actions) گفته می‌شود. شایع‌ترین عملیات پس‌زمینه در الگوریتم‌های مبتنی بر الگوریتم کلونی مورچگان، به‌کارگیری جستجوی محلی در جواب‌های کاندید تولید شده است. به‌عنوان نمونه، از جواب بهینه شده محلی برای تصمیم‌گیری در مورد به روز رسانی مقادیر فرومون‌ها استفاده می‌شود.

مرحله سوم: «به روز رسانی فرومون» (Pheromone Update)

هدف مرحله به روز رسانی فرومون، افزایش مقادیر فرومون متناظر با جواب‌های کاندید خوب و بهینه و کاهش مقادیر فرومون متناظر با جواب‌های بد است. چنین کاری، از طریق دو فرآیند عمده انجام می‌شود:

کاهش مقادیر فرومون متناظر با تمامی جواب‌های کاندید از طریق فرآیند «تبخیر فرومون» (Pheromone Evaporation)

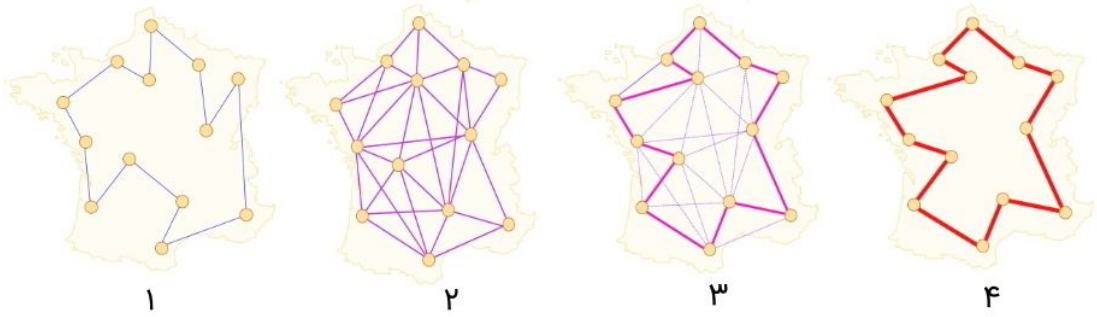
افزایش مقادیر فرومون متناظر با جواب‌های کاندید عضو مجموعه «جواب‌های خوب» یا SupdSupd

این دو فرآیند از طریق رابطه زیر کنترل می‌شوند. به رابطه زیر، «قانون به روز رسانی فرومون» گفته می‌شود.

$$\tau ij \leftarrow (1-\rho)\tau ij + \rho \sum s \in Supd | cij \in sF(s) \tau ij \leftarrow (1-\rho)\tau ij + \rho \sum s \in Supd | cij \in sF(s)$$

بخش اول این رابطه، فرآیند تبخیر فرومون (کاهش مقدار فرومون تمامی جواب‌های کاندید) را کنترل می‌کند. بخش دوم آن، تنها مقادیر فرومون متناظر با جواب‌های کاندید عضو مجموعه «جواب‌های خوب» یا SupdSupd را افزایش می‌دهد. در این رابطه، SupdSupd مجموعه جواب‌های کاندیدی را شامل می‌شود که برازندگی بالایی دارند؛ یعنی به جواب بهینه سراسری نزدیک‌تر هستند. پارامتر $\rho \in (0,1] \rho \in (0,1]$ ، «نرخ تبخیر» (Evaporation Rate) نام دارد و $F:S \rightarrow R+0 F:S \rightarrow R0+$ ، «تابع برازندگی» (Fitness Function) نام دارد. به بیان ساده، این فرآیند منجر به افزایش مقدار فرومون متناظر با مورچه‌هایی می‌شود

که در بهترین مسیرهای موجود به سمت جواب بهینه قرار دارند (به جواب بهینه نزدیک‌ترند) و دارای برازندگی بالاتری (هزینه کمتر یا سود بیشتر) هستند. در نتیجه، مورچه‌های دیگر نیز به این مسیر همگرا می‌شوند. وجود پارامتر تبخیر فرومون، برای جلوگیری از «همگرایی سریع و نابالغِ» (Rapid and Premature Convergence) الگوریتم کلونی مورچگان ضروری است. پارامتر تبخیر، نوعی مکانیزم «فراموشی» (Forgetting) در فرآیند بهینه‌سازی فراهم می‌کند و سبب تاکید بیشتر بر جستجو و کاوش مناطق جدید، در فضای جستجوی الگوریتم‌های پیاده‌سازی شده مبتنی بر الگوریتم کلونی مورچگان می‌شود.



تبخیر رد فرومون پس از چند تکرار از روش بهینه‌سازی کلونی مورچگان

بیشتر تفاوت موجود میان پیاده‌سازی‌های گوناگون انجام شده از الگوریتم کلونی مورچگان، به قانون به روز رسانی فرومون‌ها و نحوه مشخص کردن جواب‌های کاندید عضو مجموعه «جواب‌های خوب» یا SupdSupd مرتبط است. یکی از مهم‌ترین بخش‌های مرحله به روز رسانی فرومون‌ها، انتخاب اعضای مجموعه «جواب‌های خوب» SupdSupd است.

در بیشتر الگوریتم‌های تکاملی مبتنی بر الگوریتم کلونی مورچگان، SupdSupd زیرمجموعه‌ای از {SiterU{sbs}SiterU{sbs} تعریف می‌شود. در این رابطه، SiterSiter مجموعه جواب‌های کاندید تولیده شده در تکرار کنونی است، و sbssbs مجموعه بهترین جواب‌های کاندید پیدا شده از اولین تکرار الگوریتم تاکنون است. به عنوان نمونه، در قانون به روز رسانی فرومون الگوریتم سیستم مورچگان، مجموعه جواب‌های کاندید عضو مجموعه «جواب‌های خوب» از طریق رابطه Supd←SiterSupd←Siter مشخص می‌شوند.در صورتی که در قانون به روز رسانی فرومون‌ها، تاکید بیشتری روی استفاده از جواب‌های خوب هر نسل برای به روز رسانی باشد، سرعت دستیابی به جواب‌های خوب بهینه افزایش می‌یابد. با این حال، احتمال همگرایی نابالغ افزایش پیدا می‌کند. معمولا توصیه می‌شود که در هنگام پیاده‌سازی قانون به روز رسانی فرومون، مکانیزم‌هایی طراحی شوند تا از همگرایی سریع و نابالغ الگوریتم جلوگیری شود.

مزایای روش الگوریتم کلونی مورچگان

همکاری گروهی میان مورچه‌ها برای تولید جواب‌های بهینه، طبیعت مبتنی‌بر «توازی» (Parallelism) و «همبستگی» (Solidarity) این روش فرا اکتشافی را نشان می‌دهد.

بازخورد مثبت ایجاد شده از طریق انتشار فرومون در محیط، سبب همگرایی سریع به جواب‌های خوب برای مسأله بهینه‌سازی می‌شود.برای استفاده در کاربردهای پویا (کاربردهایی که نیاز به انطباق سریع با تغییرات محیطی ضروری است) مناسب است.

همگرایی به جواب بهینه، تضمین شده است.

معایب روش الگوریتم کلونی مورچگان

تجزیه و تحلیل نظری این روش بسیار سخت است.

این روش، بر پایه دنباله‌ای از تصمیمات تصادفی ولی وابسته به هم بنا نهاده شده است.

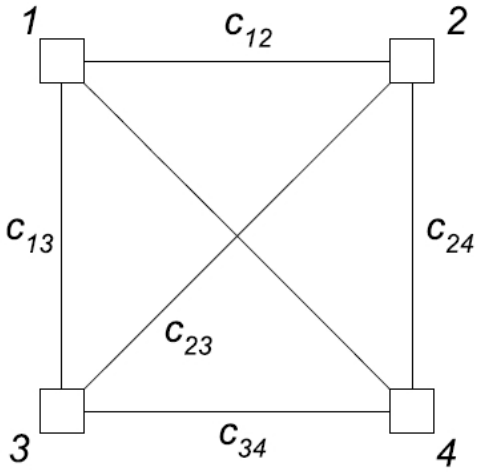
زمان لازم برای همگرایی به جواب بهینه نامشخص است.

استفاده از الگوریتم کلونی مورچگان برای حل «مسأله فروشنده دوره‌گرد» (TSP)

یکی از محبوب‌ترین روش‌ها برای نمایش چگونگی عملکرد روش فرا اکتشافی الگوریتم کلونی مورچگان، استفاده از آن در حل مسأله فروشنده دوره‌گرد است. این مسأله، از مجموعه‌ای از شهرها (مکان‌ها) و یک فروشنده دوره‌گرد تشکیل شده است. این فروشنده اجازه دارد از هر شهر تنها یک‌بار عبور کند. فاصله میان شهرها داده شده است و هدف پیدا کردن «تور همیلتونی» (Hamilton Cycle) با طول کمینه است. پیچیدگی این مسأله برابر با «ان‌پی هارد» (NP-hard) است.

کاربرد بهینه‌سازی کلونی مورچگان برای حل مسأله فروشنده دوره‌گرد ساده و صریح است. حرکت میان شهرها (مکان‌ها)، مؤلفه‌های جواب کاندید است؛ یعنی، حرکت از شهر ii به شهر jj ، مولفه جواب کاندید Cij≡CjiCij≡Cji مسأله خواهد بود. گراف ساختاری GC=(V,E)GC=(V,E) از طریق ایجاد تناظر میان مجموعه شهرها با مجموعه رأس‌ها V در گراف ساختاری تعریف می‌شود.از آنجا که هیچ محدودیتی در امکان حرکت از یک شهر به هر شهر دیگری وجود ندارد، گراف ساختاری تشکیل شده یک گراف کاملا متصل است و تعداد رأس‌های موجود در گراف برابر تعداد شهرهای تعریف شده در مسأله خواهد بود.

همچنین، اندازه یال‌های گراف متناسب با فاصله شهرها (با رأس‌ها نمایش داده می‌شوند) از یکدیگر است. فرومون نیز متناظر با مجموعه یال‌ها E در گراف ساختاری خواهد بود.نمونه‌ای از یک گراف کامل متصل برای مسأله فروشنده دروه‌گرد با تعداد چهار شهر در شکل زیر آمده است.



مورچه‌ها به شکلی که در ادامه بیان شده است، اقدام به تولید جواب‌های مسأله می‌کنند. هر کدام از مورچه‌ها، از یک شهر (یک رأس در گراف) کاملا تصادفی شروع می‌کنند. سپس، در هر گام از فرآیند تولید جواب، در راستای یال‌های گراف به حرکت می‌پردازند. هر مورچه، مسیر پیموده شده در گراف را به خاطر می‌سپارد و در گام‌های بعدی، یال‌هایی را برای حرکت در گراف انتخاب می‌کند که به مکان‌های (رأس‌های) از پیش پیموده شده منتهی نشوند. به محض اینکه تمامی رأس‌های گراف توسط یک مورچه پیمایش شد، یک جواب کاندید تولید می‌شود.در هر گام از فرآیند تولید جواب، مورچه‌ها به طور احتمالی، از میان یال‌های در دسترس (یال‌های پیموده نشده و منتهی به رأس‌هایی که از آن‌ها گذر نکرده)، یک یال را برای پیمایش انتخاب می‌کنند. نحوه محاسبه احتمال انتخاب یال‌ها، به پیاده‌سازی انجام شده از الگوریتم کلونی مورچگان بستگی دارد. پس از اینکه تمامی مورچه‌ها یک جواب کاندید تولید کردند، فرومون روی یال‌ها، براساس «قانون به روز رسانی فرومون» به روز رسانی می‌شود.

پیاده‌سازی الگوریتم کلونی مورچگان برای حل مسأله فروشنده دوره‌گرد در متلب

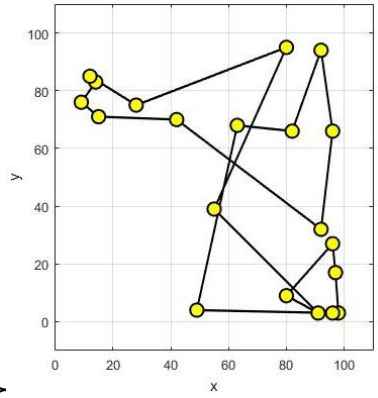
در این بخش، کد متلب پیاده‌سازی روش بهینه‌سازی کلونی مورچگان برای حل مسأله فروشنده دوره‌گرد آورده شده است. تعداد شهرها (مکان‌ها)، برای مسأله فروشنده دوره‌گرد، برابر با ۲۰ خواهد بود. مختصات مکانی شهرها در ادامه آمده است.

MATLAB

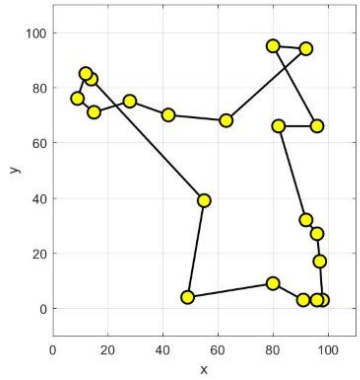
```
1x=[82 91 12 92 63 9 28 55 96 97 15 98 96 49 80 14 42 92 80 96];
```

```
2
3       y=[66 3 85 94 68 76 75 39 66 17 71 3 27 4 9 83 70 32 95 3];
```

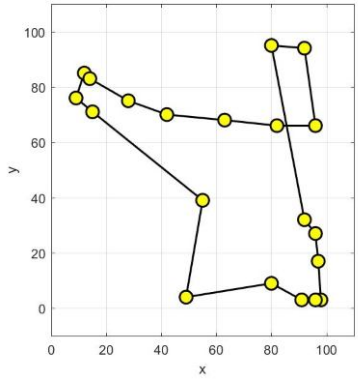
تعداد تکرارهای بیشینه برای رسیدن به جواب بهینه سراسری مسأله فروشنده دوره‌گرد، برابر با ۲۰۰ در نظر گرفته شده است. وزن اطلاعات فرومون (پارامتر آلفا) برابر با ۱ و وزن اطلاعات اکتشافی نیز برابر با ۱ است؛ یعنی اطلاعات فرومون و اطلاعات اکتشافی، به یک اندازه، در تولید احتمال انتخاب هر یک از مؤلفه‌های مجموعه همسایگان امکان‌پذیر برای گسترش جواب کاندید جزئی، سهیم هستند. تعداد جمعیت مورچه‌ها، برابر با ۴۰ است. در هر تکرار، از مقدار برازندگی جواب‌های تولید شده توسط مورچه‌ها برای به روز رسانی مقادیر فرومون استفاده می‌شود.نرخ تبخیر، برای تولید مکانیزم تکاملی و همگرا شونده به جواب بهینه سراسری، برابر با ۰،۰۵ در نظر گرفته شده است. این مقدار، تضمین می‌کند که الگوریتم بهینه‌سازی، جواب‌های کاندید بد را به راحتی فراموش کند و به جستجو برای یافتن جواب بهینه سراسری در دیگر مناطق فضای جستجو ادامه دهد. شکل‌های زیر نحوه همگرایی روش بهینه‌سازی کلونی مورچگان به جواب بهینه سراسری، برای حل مسأله فروشنده دوره‌گرد را نمایش می‌دهند.



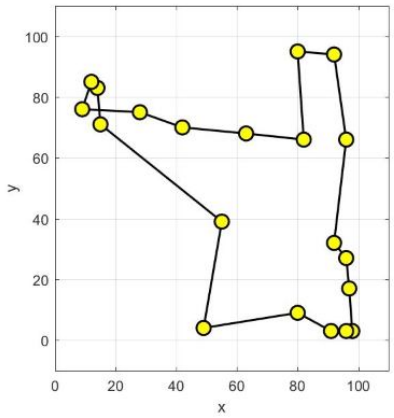
حرکت به سمت جواب بهینه سراسری پس از تکرار ۱



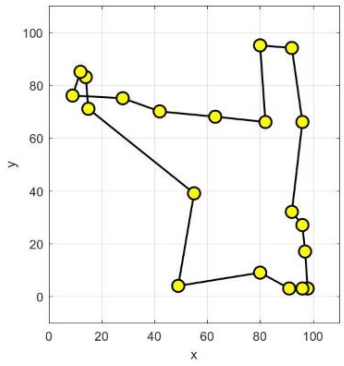
حرکت به سمت جواب بهینه سراسری پس از تکرار ۲۵



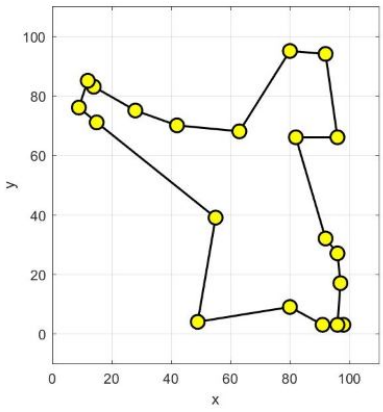
حرکت به سمت جواب بهینه سراسری پس از تکرار ۵۰



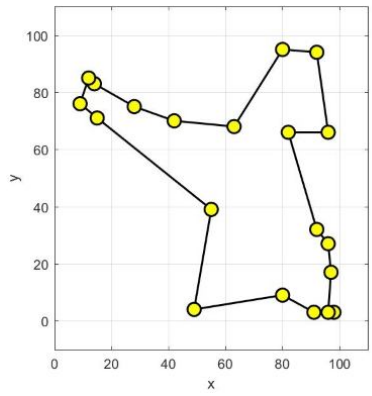
حرکت به سمت جواب بهینه سراسری پس از تکرار ۷۵



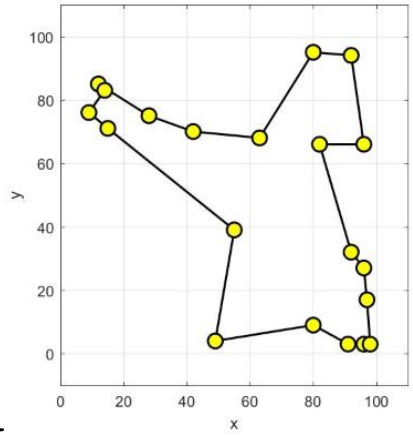
حرکت به سمت جواب بهینه سراسری پس از تکرار ۱۰۰



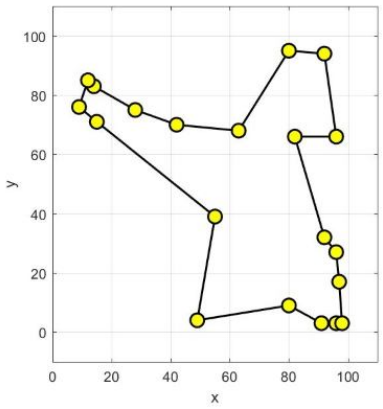
حرکت به سمت جواب بهینه سراسری پس از تکرار ۱۲۵



حرکت به سمت جواب بهینه سراسری پس از تکرار ۱۵۰

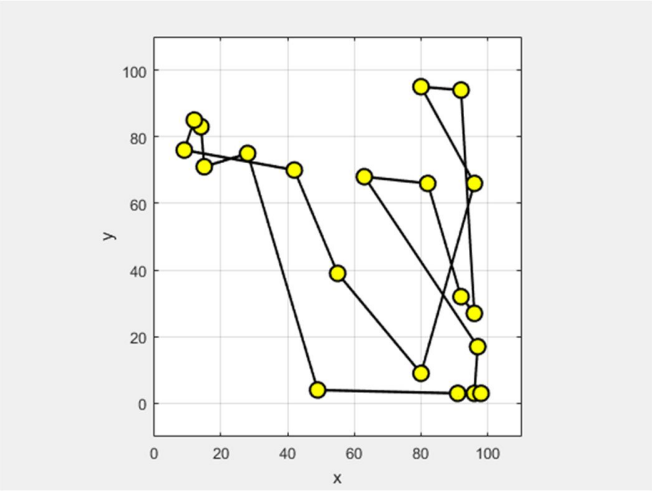


حرکت به سمت جواب بهینه سراسری پس از تکرار ۱۷۵

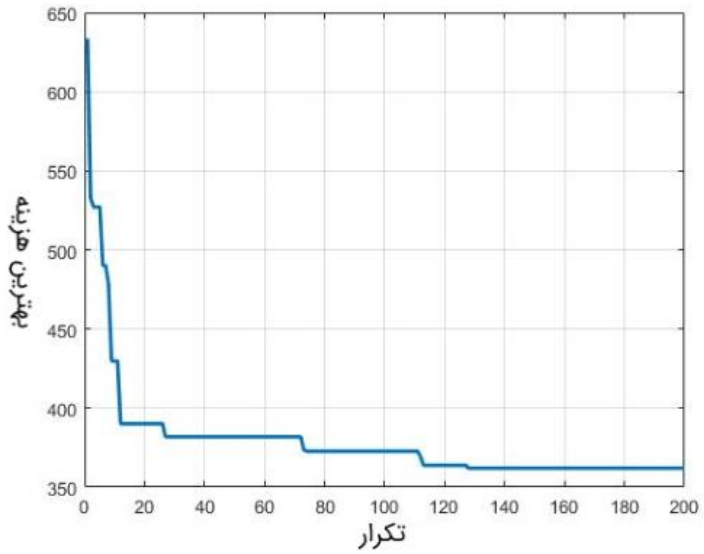


از پایان اجرا

نحوه همگرایی روش بهینه‌سازی کلونی مورچگان به مسیر بهینه در مسأله فروشنده دوره‌گرد در تصویر متحرک زیر نمایش داده شده است.



الگوریتم کلونی مورچگان، در تکرار ۱۲۸ به جواب بهینه سراسری (مقدار هزینه برابر با ۳۶۲.۰۳۸) همگرا شده است. شکل زیر، نحوه همگرایی به هزینه بهینه را نشان می‌دهد.



در ادامه، توابع مورد نیاز برای اجرای روش بهینه‌سازی کلونی مورچگان نمایش داده شده‌اند. شایان توجه است که برای اجرای این روش بهینه‌سازی در متلب، ابتدا باید هر کدام از توابع ارائه شده در زیر را در قالب یک فایل تابعی قابل خواندن توسط متلب ذخیره و سپس آن‌ها را اجرا کرد.

کاربردهای الگوریتم کلونی مورچگان

الگوریتم کلونی مورچگان، می‌تواند برای حل مسائل بهینه‌سازی ترکیبیات مختلفی استفاده شود. این الگوریتم تاکنون در کاربردهایی نظیر مسیریابی وسایل نقلیه و بهینه‌سازی استراتژی برنامه‌ریزی کارها مورد استفاده قرار گرفته است. بسیاری از الگوریتم‌های بهینه‌سازی پیاده‌سازی شده بر مبنای این روش، برای حل مسائل بهینه‌سازی متغیرهای حقیقی، مسائل تصادفی و مسائل بهینه‌سازی چندهدفه به کار گرفته شده‌اند. آزمایشات انجام شده برای تولید جواب بهینه سراسری مسأله فروشنده دوره‌گرد،نشان می‌دهد که این روش می‌تواند جواب بسیار دقیق و نزدیک به جواب بهینه سراسری را تولید کند. از آنجا که این الگوریتم بر پایه ساختار گرافی بنا نهاده شده است، به راحتی می‌تواند خود را با تغییرات پویای محیطی سازگار کند؛ ویژگی‌ای که سبب مزیت این الگوریتم به دیگر الگوریتم‌های بهینه‌سازی نظیر الگوریتم ژنتیک و الگوریتم «شبیه‌سازی تبرید» (Simulated

Annealing) شده است. همچنین، قابلیت تطابق با شرایط در حال تغییر محیط عملیاتی، این الگوریتم را به گزینه مناسبی برای کاربردهایی نظیر مسیریابی شبکه و برنامه‌ریزی سیستم‌های حمل و نقل شهری تبدیل کرده است. مهم‌ترین کاربردهای الگوریتم مورچگان عبارتند از:

- مسائل بهینه‌سازی استراتژی برنامه‌ریزی کارها (Job-Scheduling Problems).
- بهینه‌سازی مسیریابی وسایل نقلیه.
- «پردازش تصویر» (Image Processing).
- مسائل بهینه‌سازی اندازه دستگاه‌ها در طراحی فیزیکی دستگاه‌های نانوالکترونیکی.
- بهینه‌سازی شکل آنتن‌ها (به عنوان نمونه در برچسب‌های RF-ID).
- «دسته‌بندی» (Classification).
- «داده‌کاوی» (Data Mining).

جمع‌بندی

الگوریتم کلونی مورچگان، یکی از روش‌های فرا اکتشافی است که بر پایه رفتار ازدحامی مورچه‌های واقعی برای یافتن منابع غذایی الهام گرفته شده است. مؤلفه اساسی الگوریتم کلونی مورچگان، مدل فرومون است. روش بهینه‌سازی کلونی مورچگان، یک مدل برای پیاده‌سازی الگوریتم‌های بهینه‌سازی ارائه می‌دهد. تاکنون، الگوریتم‌های بهینه‌سازی متفاوتی بر پایه این روش بهینه‌سازی ارائه شده‌اند. از مهم‌ترین پیاده‌سازی‌های انجام شده می‌توان به الگوریتم‌هایی نظیر «سیستم مورچگان» (Ant System)، «سیستم کلونی مورچگان» (Ant Colony System) و «سیستم مورچگان Min-Max» اشاره کرد. ویژگی مهم این روش، اثرگذاری و انعطاف‌پذیری فوق‌العاده آن در حل مسائل بهینه‌سازی است.

«الگوریتم‌های الهام گرفته شده از طبیعت» (Nature-Inspired Algorithms)، در زمره قدرتمندترین الگوریتم‌های «بهینه‌سازی» (Optimization) قرار دارند. در این مطلب، یک الگوریتم بهینه‌سازی به نام «الگوریتم کرم شب تاب» (| FA Firefly Algorithm) مورد بررسی قرار می‌گیرد. ویژگی مهم الگوریتم کرم شب تاب، که آن را از برخی از الگوریتم‌های بهینه‌سازی مشابه متمایز می‌کند، عملکرد بسیار خوب آن در جستجوی جواب‌های بهینه مرتبط با مسائل و توابع «چندمُدی» (Multimodality) است. چنین ویژگی مهمی در الگوریتم کرم شب تاب سبب شده است تا این الگوریتم، به انتخاب ایده‌آلی برای کاربردهای بهینه‌سازی چندمُدی تبدیل شود.

فهرست مطالب این نوشته پنهان کردن

۵. پیاده‌سازی الگوریتم کرم شب تاب در زبان‌های برنامه‌نویسی مختلف
۵,۱. پیاده‌سازی الگوریتم کرم شب تاب در زبان متلب
۵,۲. پیاده‌سازی الگوریتم کرم شب تاب در زبان جاوا
۵,۳. پیاده‌سازی الگوریتم کرم شب تاب در زبان C++/C
۶. دوره ویدیویی آموزش الگوریتم کرم شب تاب یا Firefly Algorithm در متلب
۷. جمع‌بندی

در این مطلب، ابتدا مکانیزم‌ها و مؤلفه‌های الگوریتم کرم شب تاب (جهت «همگرایی» (Convergence) به جواب «بهینه سراسری» (Optimal Solution))، با جزئیات کامل و دقیق مورد بررسی قرار می‌گیرند. سپس، الگوریتم کرم شب تاب با دیگر الگوریتم‌های «فرا اکتشافی» (Meta-Heuristics) نظیر «بهینه‌سازی ازدحام ذرات» (| Particle Swarm Optimization PSO) مقایسه خواهد شد. شبیه‌سازی‌ها و نتایج حاصل نشان می‌دهد که الگوریتم کرم شب تاب عملکرد به مراتب بهتری نسبت به روش‌های «بهینه‌سازی ازدحامی» (Swarm Optimization) از خود نشان می‌دهد. در نهایت، کاربردهای الگوریتم کرم شب تاب در حوزه‌های مختلف و جهت‌گیری پژوهشی آن در آینده مورد بحث و بررسی قرار می‌گیرد.

«بهینه‌سازی» (Optimization)، به فرآیند تغییر و دستکاری ورودی‌ها، عملیات ریاضی یا خصوصیات یک دستگاه گفته می‌شود که با هدف رسیدن به یک خروجی یا جواب بهینه انجام می‌شود. ورودی فرآیند بهینه‌سازی، متغیرهای مسأله‌ای هستند که قرار است به وسیله یکی از الگوریتم‌های خانواده روش‌های «بهینه‌سازی عددی» (Numerical Optimization) پردازش و جواب‌های بهینه آن مشخص شود. خروجی فرآیند بهینه‌سازی، «برازندگی» (Fitness) نامیده می‌شود؛ به فرآیند یا تابعی که قرار است توسط الگوریتم‌های بهینه‌سازی پردازش شود، «تابع هدف» (Objective Function) گفته می‌شود.

الگوریتم‌های بهینه‌سازی عددی مبتنی بر فرایندهای طبیعی، به یکی از مهم‌ترین زیر شاخه‌های حوزه «محاسبات تکاملی» (Evolutionary Computation) تبدیل شده‌اند. این دسته از روش‌های محاسباتی، الگوریتم‌هایی هستند که برای حل مسائل مختلف و بهینه‌سازی عددی آن‌ها پیاده‌سازی شده‌اند. یکی از مهم‌ترین نمونه‌های چنین الگوریتم‌هایی، الگوریتم ژنتیک (و دیگر الگوریتم‌های مبتنی بر این روش محاسباتی) است که از مکانیزم‌های الهام گرفته شده از تکامل زیستی (نظیر «جهش» (Mutation)، «ترکیب» (Recombination)، «تولید مثل» (Reproduction)، «انتخاب طبیعی» (Natural Selection) و «بقای بهترین‌ها» (Survival of Fittest)) برای تولید جواب‌های کاندید، جهت حل یک مسأله بهینه‌سازی، استفاده می‌کنند. به این مکانیزم‌ها، عملگرهای تکاملی نیز گفته می‌شود.

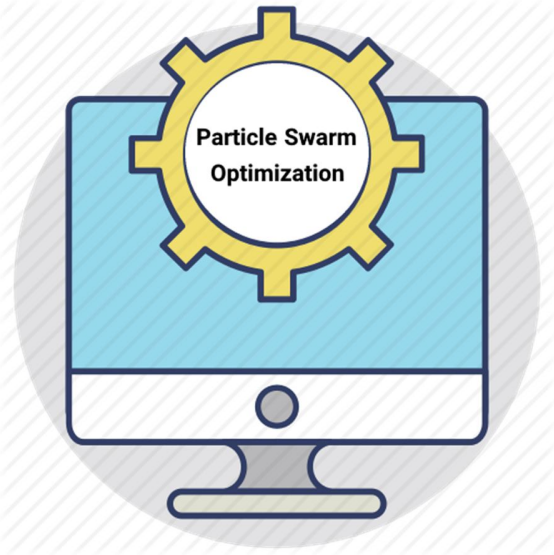
جواب‌های کاندید، نقش نمونه‌ها را در جمعیت ایفا می‌کنند. «تابع هزینه» (Cost Function) نیز برازندگی محیطی را که جواب کاندید در آن قرار دارد (یک جواب کاندید برای مسأله بهینه‌سازی) مشخص می‌کند. الگوریتم‌هایی نظیر الگوریتم ژنتیک، الگوریتم کلونی مورچگان، الگوریتم کرم شب تاب، الگوریتم بهینه‌سازی فاخته و سایر موارد از جمله مهم‌ترین الگوریتم‌های تکاملی هستند.

در چند سال اخیر، الگوریتم‌های الهام گرفته شده از فرایندهای زیستی (Biologically-inspired Algorithm) به یکی از قدرتمندترین تکنیک‌ها و ابزارهای «بهینه‌سازی عددی» (Numerical Optimization) تبدیل شده‌اند؛ به ویژه، در حل مسائل با درجه پیچیدگی NP-Hard، نظیر «مسأله فروشنده دوره‌گرد» (Travelling Salesman Problem)، بسیار قدرتمند ظاهر شده‌اند و در مدت زمان بسیار معقولی به جواب بهینه سراسری همگرا می‌شوند. در این میان، الگوریتم‌های مبتنی بر «هوش ازدحامی» (Swarm Intelligence) و «الگوریتم‌های فرا اکتشافی چند عامله» (Multi-Agent Meta-Heuristic Algorithm) جایگاه ویژه‌ای در حوزه بهینه‌سازی عددی دارند و به عنوان یکی از داغ‌ترین سوژه‌های تحقیقاتی در این حوزه قلمداد می‌شوند. الگوریتم‌هایی نظیر بهینه‌سازی ازدحام ذرات (PSO)، که زیر مجموعه الگوریتم‌های فرا اکتشافی چند عامله و مبتنی بر هوش ازدحامی است، جزء پیشرفته‌ترین الگوریتم‌های توسعه داده شده در بهینه‌سازی و دیگر کاربردهای مشابه هستند. الگوریتم بهینه‌سازی ازدحام ذرات (PSO)، در سال ۱۹۹۵، توسط دو دانشمند به نام‌های کندی (Kennedy) و ابرهارت (Eberhart) ابداع شده است. الگوریتم بهینه‌سازی ازدحام ذرات بر پایه «رفتارهای ازدحامی» (Swarm Behaviors) گونه خاصی از موجودات زنده نظیر رفتارهای اجتماعی پرواز پرندگان برای یافتن غذا یا حرکت گروهی و در جهت یکسان ماهی‌ها الهام گرفته شده است. همانطور که پیش از این نیز اشاره شد، به چنین رفتاری در موجودات زنده، که الهام بخش توسعه برخی از الگوریتم‌های بهینه‌سازی معروف نیز هستند، هوش ازدحامی (Swarm Intelligence) گفته می‌شود.با اینکه الگوریتم بهینه‌سازی ازدحام ذرات وجه شباهت‌های زیادی با «الگوریتم ژنتیک» (Genetic Algorithm) دارد، ولی از لحاظ ساختاری، به مراتب ساده‌تر از این الگوریتم است؛ دلیل این امر این است که در الگوریتم بهینه‌سازی ازدحام ذرات، از «عملگرهای تکاملی» (Evolution Operators) نظیر «ترکیب یا آمیزش» (Crossover) و «جهش» (Mutation) استفاده نمی‌شود. در این الگوریتم، از مکانیزم‌های تکاملی نظیر «تصادفی بودن» (Randomness) اعداد حقیقی و «ارتباطات سراسری» (Global Communications) میان ذرات ازدحامی، جهت رسیدن به تکامل و دست‌یابی به جواب بهینه سراسری استفاده می‌شود. از سوی دیگر، از آنجایی که تمامی نمایش‌های تولید شده از جمعیت، «کدبندی‌های» (Encodings) متناظر از موجودیت‌های «جمعیت» (Population) و مکانیزم‌های تکاملی استفاده شده برای همگرایی به جواب‌های بهینه (در الگوریتم بهینه‌سازی ذرات)، مبتنی بر عملیاتی هستند که عمدتا روی «اعداد حقیقی» (Real Numbers) انجام می‌شوند، پیاده‌سازی آن نیز راحت‌تر از الگوریتم ژنتیک خواهد بود. در این مطلب، هدف ارائه مکانیزم‌ها و مؤلفه‌های الگوریتم کرم شب تاب جهت حل مسائل بهینه‌سازی عددی و جزئیات مرتبط با آن‌ها است. همچنین در این مطلب، مطالعه مقایسه‌ای از الگوریتم کرم شب تاب با الگوریتم‌هایی نظیر بهینه‌سازی ازدحام ذرات و دیگر الگوریتم‌های بهینه‌سازی مشابه ارائه خواهد شد. بنابراین، ساختار مطلب پیش رو به این صورت است که ابتدا الگوریتم بهینه‌سازی ازدحام ذرات و جزئیات آن معرفی خواهد شد. سپس، جزئیات مرتبط با مکانیزم‌ها و مؤلفه‌های الگوریتم کرم شب تاب ارائه خواهد شد. در مرحله آخر نیز، عملکرد الگوریتم کرم شب تاب با دیگر الگوریتم‌های بهینه‌سازی نظیر الگوریتم بهینه‌سازی ازدحام ذرات بررسی خواهد شد. نتایج مقایسه عملکرد الگوریتم کرم شب تاب با دیگر الگوریتم‌های بهینه‌سازی نشان می‌دهد که این الگوریتم، به شکل بهتر و کارآمدتری می‌تواند به بهینه سراسری توابع چندمُدی همگرا شود؛ در نتیجه، الگوریتم کرم شب تاب انتخاب مناسب‌تری برای حل مسائل بهینه‌سازی چندمُدی و همگرایی به بهینه‌های سراسری این دسته از مسائل خواهد بود. بنابراین با در نظر گرفتن حالت ذکر شده، می‌توان نتیجه گرفت که الگوریتم بهینه‌سازی ازدحام ذرات، حالت خاصی از الگوریتم کرم شب تاب محسوب می‌شود (در ادامه، به این موضوع بیشتر پرداخته خواهد شد).

الگوریتم بهینه‌سازی ازدحام ذرات (PSO)

پیش از اینکه الگوریتم کرم شب تاب و ساختار آن مورد بررسی قرار بگیرد، الگوریتم بهینه‌سازی ازدحام ذرات به طور مختصر معرفی خواهد شد. مطالعه ویژگی‌های الگوریتم بهینه‌سازی ازدحام ذرات به خوانندگان و مخاطبان این مطلب کمک می‌کند تا

شباهت‌ها و تفاوت‌های میان این دو الگوریتم و همچنین، نقاط ضعف و قوت آن‌ها در حل مسائل مختلف بهینه‌سازی را به شکل بهتری درک کنند.



الگوریتم استاندارد بهینه‌سازی ازدحام ذرات

در الگوریتم بهینه‌سازی ازدحام ذرات (PSO)، به هر کدام از «عوامل» (Agents) موجود در جمعیت «ذره» (Particles) گفته می‌شود. در الگوریتم PSO، به موازات شکل‌گیری شبه تصادفی «مسیرهای تکه‌ای» (Piecewise Paths) متناظر با هر یک از عوامل توسط «بردارهای مکانی» (Positional Vectors)، فضای جستجوی «توابع هدف» (Objective Functions) از طریق «تنظیم» (Adjust) کردن «مسیرهای» (Trajectories) هر کدام از عوامل موجود در جمعیت (ذرات) جستجو می‌شود (برای یافتن جواب بهینه یا ناحیه در بر گیرنده جواب بهینه). در حال حاضر بیش از ۲۰ نسخه مختلف از الگوریتم بهینه‌سازی ازدحام ذرات معرفی شده است. در این بخش، ساده‌ترین و البته محبوب‌ترین نسخه الگوریتم بهینه‌سازی ازدحام ذرات (الگوریتم استاندارد بهینه‌سازی ازدحام ذرات) معرفی خواهد شد.

حرکت ذرات در فضای جستجو به دو «مؤلفه» (Components) اساسی بستگی دارد: یک مؤلفه تصادفی (Stochastic) و یک مؤلفه قطعی (Deterministic). هر کدام از ذرات موجود در جمعیت، علاوه بر اینکه تمایل دارد به صورت تصادفی در فضای مسأله حرکت کند، به طور همزمان، به سمت نقطه‌ای که بهینه سراسری یا پارامتر g⋆g⋆ در آن قرار گرفته است، جذب می‌شود. به عبارت دیگر، هر کدام از ذرات موجود در جمعیت، به سمت مکان بهترین نقطه‌ای (در فضای مسأله) که ذرات تا کنون در آن قرار گرفته‌اند (بهترین نقطه، بر اساس تاریخچه حرکت ذرات در محیط سنجیده می‌شود) یا پارامتر x⋆ixi⋆ ، جذب می‌شوند. زمانی که یکی از ذرات موجود در جمعیت (ذره ii)، مکانی را در فضای جستجو پیدا کند که از دیگر مکان‌های جستجو شده در این فضا بهتر باشد، مکان جدید یافته شده به عنوان بهترین نقطه‌ای (در فضای مسأله) که ذره ii تا کنون در آن قرار گرفته است، به‌روزرسانی می‌شود. این مکان جدید در فضای مسأله، به عنوان بهترین جواب ممکن برای تمامی nn ذره موجود در جمعیت شناخته خواهد شد. هدف نهایی الگوریتم بهینه‌سازی ازدحام ذرات این است که جواب بهینه سراسری را از میان مجموعه بهترین جواب‌های ممکن یافت شده (توسط ذرات) مشخص کند. فرایند یافتن بهترین جواب‌های ممکن (توسط ذرات) تا زمانی ادامه پیدا می‌کند که پس از تعداد تکرارهای مشخصی، دیگر بهبود قابل ملاحظه‌ای در بهترین جواب ممکن ایجاد نشود. بهترین جواب موجود در مجموعه جواب‌های بهینه یافت شده، به عنوان جواب بهینه سراسری مسأله مشخص می‌شود.

جهت نمایش حرکت ذرات در فضای مسأله، از x⋆ixi⋆ برای مشخص کردن بهترین نقطه (جواب) یافت شده توسط ذره ii و از رابطه زیر:

g⋆≈minormax{f(xi)}(i=1,2,....,n)g⋆≈minormax{f(xi)}(i=1,2,....,n)

برای مشخص کردن جواب بهینه سراسری (بهترین جواب موجود در مجموعه جواب‌های بهینه یافت شده توسط ذرات) استفاده می‌شود.

با فرض اینکه xixi و vivi به ترتیب بردار مکان و «سرعت» (Velocity) ذره ii باشند، بردار سرعت جدید از طریق رابطه زیر به دست می‌آید:

vt+1i=vti+αϵ1⊙(g⋆−xti)+βϵ2⊙(x∗i−xti)vit+1=vit+αϵ1⊙(g⋆−xit)+βϵ2⊙(xi∗−xit)

در این رابطه، ϵ2ϵ2 و ϵ1ϵ1 دو بردار تصادفی هستند و هر کدام از عناصر این بردارها، مقداری تصادفی بین ۰ و ۱ به خود می‌گیرند. «ضرب آدامار» (Hadamard Product) دو ماتریس u⊙vvu⊙v در قالب «ضرب عنصر به عنصر» (Entry-wise Product) تعریف و به شکل زیر نمایش داده می‌شود:

|u⊙v|ij=uijvij|u⊙v|ij=uijvij

پارامترهای αα و ββ ، «پارامترهای یادگیری» (Learning Parameters) یا «ثابت‌های شتاب» (Acceleration Constants) نام دارند. این دو پارامتر معمولا به شکل α≈β≈2α≈β≈2 مقداردهی می‌شوند. مقادیر ابتدایی پارامتر Xt=0iXit=0 ، با استفاده از حدهای بالا و پایین a=min(xi),b=max(xj)a=min(xi),b=max(xj) مشخص می‌شود. همچنین، مقدار ابتدایی پارامتر Vt=0iVit=0 برابر با صفر در نظر گرفته می‌شود (Vt=0i=0Vit=0=0).

با توجه به روابط فوق و مقادیر پارامترها، مکان جدید ذره ii در فضای جستجو، از طریق رابطه زیر مشخص می‌شود:

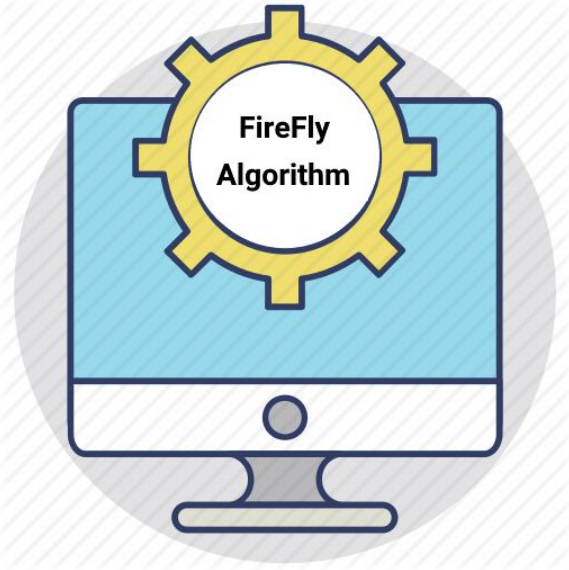
Xt+1i=Xti+Vt+1iXit+1=Xit+Vit+1

اگرچه پارامتر ViVi می‌تواند هر مقداری به خود بگیرد، ولی معمولا مقدار این پارامتر از بازه [0,Vmax][0,Vmax] انتخاب می‌شود.تاکنون نسخه‌های مختلفی از الگوریتم بهینه‌سازی ازدحام ذرات معرفی شده است. مهم‌ترین بهبودی که در نسخه‌های مختلف الگوریتم بهینه‌سازی ازدحام ذرات و در مؤلفه‌های آن ایجاد شده است، استفاده از «تابع اینرسی» (Inertia Function) به فرم θ(t)θ(t) است. در نسخه‌هایی از الگوریتم بهینه‌سازی ازدحام ذرات که از تابع اینرسی استفاده می‌کند، به جای پارامتر VtiVit از عبارت θ(t)Vtiθ(t)Vit استفاده می‌شود؛ در این عبارت، پارامتر θθ مقادیری بین ۰ و ۱ به خود می‌گیرد.

در ساده‌ترن حالت، تابع اینرسی را می‌توان به عنوان یک ثابت در الگوریتم بهینه‌سازی ازدحام ذرات در نظر گرفت که مقادیری به شکل θ≈0.5≈0.9θ≈0.5≈0.9 به خود می‌گیرد. اضافه کردن تابع اینرسی (به شکل یک پارامتر ثابت) را می‌توان به عنوان یک «جرم مجازی» (Virtual Mass) تصور کرد که حرکت ذرات در فضای جستجوی مسأله را پایدار می‌کند؛ در نتیجه، انتظار می‌رود که در چنین حالتی، الگوریتم بهینه‌سازی ازدحام ذرات با سرعت بیشتری به جواب بهینه سراسری همگرا شود (یا ذرات موجود در جمعیت، با سرعت بیشتری به سمت ناحیه دربرگیرنده جواب بهینه سراسری همگرا شوند).

الگوریتم کرم شب تاب

در این بخش، مکانیزم‌ها و مؤلفه‌های الگوریتم کرم شب تاب (جهت «همگرایی» (Convergence) به جواب «بهینه سراسری» (Optimal Solution))، با جزئیات کامل و دقیق مورد بررسی قرار می‌گیرد.



رفتار کرم‌های شب تاب

یکی از صحنه‌های جذاب و زیبایی که گردشگران و ساکنین مناطق استوایی یا معتدل، در آسمان شب‌های فصل تابستان، با آن موجه می‌شوند، «نور چشمک‌زن» (Flashing Light) کرم‌های شب تاب است. تا به امروز، چیزی حدود ۲ هزار گونه متفاوت از کرم‌های شب تاب در دنیا به ثبت رسیده است و بیشتر آن‌ها، نورهای چشمک‌زن «متناوب» (Rhythmic) و کوتاهی را تولید می‌کنند. معمولا، هر یک از گونه‌های کرم شب تاب، الگوهای نور چشمک‌زن یکتا و منحصر به فردی تولید می‌کنند.

نور چشمک‌زنی که توسط کرم‌های شب تاب ساطع می‌شود، در نتیجه فرایند زیستی به نام «لومینسانس زیستی یا فسفر افکنی» (Bioluminescence) ایجاد می‌شود که سبب ایجاد پدیده شب تابی در کرم‌های شب تاب می‌شود. تاکنون، بحث‌ها و مطالعات زیادی در مورد کارکرد این نورهای چشمک‌زن انجام شده است ولی دانشمندان نتوانستند به نظریه واحدی در رابطه با کارکرد واقعی آن‌ها دست پیدا کنند. با این حال، دو کارکرد بنیادی فرایند لومینسانس زیستی و شب تابی حاصل از آن، عبارتند از:

جذب کردن جنس مخالف برای جفت‌گیری و آمیزش (کارکرد ارتباطی (Communication) فرایند لومینسانس زیستی)

جذب کردن طعمه‌های محتمل به سمت خود علاوه بر این، محققان دریافته‌اند که کرم‌های شب تاب از نورهای چشمک‌زن به عنوان یک مکانیزم محافظتی جهت ارسال هشدار به دیگر کرم‌های شب تاب موجود در محیط استفاده می‌کنند. ریتم یا تناوب نور چشمک‌زن، نرخ چشمک زدن نور و مدت زمان چشمک زدن نور توسط کرم‌های شب تاب، بخش‌های مختلف سیستم ارتباطی میان کرم‌ها (جهت جفت‌گیری با جنس مخالف یا هشدار به کرم‌های شب تاب موجود در محیط در مورد خطرات محتمل) را شکل می‌دهد.به عنوان نمونه، در هنگام جفت‌گیری کرم‌های شب تاب، کرم ماده (از یک گونه خاص) به الگوی چشمک زن منحصر به فرد کرم‌های مذکر (از همان گون خاص) واکنش نشان می‌دهد. همچنین، در برخی از گونه‌های کرم شب تاب نظیر «فوتوریس» (Photuris)، کرم ماده این قابلیت را دارد که الگوی چشمک زن منحصر به فرد دیگر گونه‌ها (جهت جفت‌گیری) را تقلید کند. با چنین کاری، کرم شب تاب ماده می‌تواند کرم‌های مذکر گونه‌های دیگر را، که برای جفت‌گیری به سمت کرم ماده حرکت کرده‌اند، به دام بیاندازد و بخورد. یکی از نکاتی که باید در مورد الگوی چشمک زن نوری کرم‌های شب تاب به خاطر داشته باشید این است که «شدت نور» (Light Intensity) در یک فاصله مشخص، rr ، از «منبع نور» (Light Source)، از «قانون مربع معکوس» (Inverse Square Law) تبعیت می‌کند. به عبارت دیگر، شدت نور، II ، با افزایش فاصله، rr ، مطابق با رابطه I∝1r2I∝1r2 کاهش پیدا می‌کند. علاوه بر این، «هوا» (Air) نور را جذب می‌کند که به نوبه خود سبب می‌شود تا شدت نور، با افزایش فاصله، ضعیف و ضعیف‌تر شود.

ترکیب این دو عامل مهم سبب می‌شود تا کرم‌های شب تاب، تنها از فاصله مشخصی قابل مشاهده باشند؛ معمولا، در تاریکی شب، نور چشمک‌زن کرم‌های شب‌تاب از فاصله چند صد متری قابل رؤیت است که برای مشاهده شدن توسط دیگر کرم‌ها و برقراری ارتباط با آن‌ها کفایت می‌کند. نور چشمک‌زن تولید شده توسط کرم‌های شب تاب را می‌توان به گونه‌ای فرمول‌بندی (Formulate) کرد که متناظر با «تابع هدفی» (Objective Function) باشد که قرار است توسط الگوریتم‌های بهینه‌سازی بهینه شوند؛ چنین کاری به محققان اجازه می‌دهد تا بتوانند الگوریتم‌های بهینه‌سازی جدید را فرمول‌بندی و پیاده‌سازی کنند. در ادامه این مطلب، ابتدا مفاهیم مقدماتی و نحوه فرمول‌بندی کردن الگوریتم کرم شب تاب مورد بررسی قرار می‌گیرد. سپس، جزئیات مرتبط با پیاده‌سازی و تحلیل الگوریتم کرم شب تاب بررسی خواهد شد.

مفاهیم مرتبط با الگوریتم کرم شب تاب

در این بخش، جهت پیاده‌سازی الگوریتم بهینه‌سازی الهام گرفته شده از رفتار کرم شب تاب (Firefly-inspired Optimization Algorithm)، ویژگی‌های مشخصه مرتبط با رفتار کرم شتاب و الگوی نور چشمک‌زن تولید شده توسط آن‌ها فرمول‌بندی خواهد شد. جهت ساده‌سازی فرمول‌بندی کردن الگوریتم کرم شب تاب، از قواعد زیر استفاده می‌شود:

تمامی کرم‌های شب تاب، «تک جنسی» (Unisex) هستند. به عبارت دیگر، کرم‌های شب تاب، فارغ از جنسیت آن‌ها، مجذوب دیگر کرم‌های شب تاب موجود در فضای مسأله خواهند شد.

در الگوریتم کرم شب تاب (FA)، «جذابیت» (Attractiveness) یک کرم شب تاب متناسب با «روشنایی» (Brightness) آن کرم خواهد بود. به عبارت دیگر، به ازاء هر دو کرم شب تاب چشمک‌زن، کرمی که نور کمتری دارد به سمت کرمی که نور بیشتری دارد جذب خواهد شد. بنابراین، جذابیت کرم شب تاب، متناسب با روشنایی آن خواهد بود.

وقتی که فاصله بین دو کرم افزایش پیدا می‌کند، مقدار جذابیت (Attractiveness) و روشنایی (Brightness) آن‌ها کاهش پیدا می‌کند. به عبارت دیگر، وقتی که فاصله دو کرم شب تاب از یکدیگر بیشتر می‌شود، علاوه بر اینکه جذابیت آن‌ها برای یکدیگر کاهش پیدا می‌کند، روشنایی (قابل رؤیت) آن‌ها (برای یکدیگر) نیز کاهش پیدا می‌کند. در صورتی که روشنایی یک کرم شب‌تاب خاص از دیگر کرم‌ها بیشتر باشد، به طور تصادفی در محیط حرکت خواهد کرد (به سمت هیچ یک از کرم‌های دیگر جذب نخواهد شد). روشنایی یک کرم شب تاب، از ویژگی‌های مشخصه تابع هدف تأثیر می‌پذیرد و یا به وسیله آن مشخص می‌شود. در مسائل «بیشنیه‌سازی» (Maximization)، روشنایی را می‌توان متناسب با مقدار «تابع برازندگی» (Fitness Function) مشخص کرد.

شایان توجه است که این امکان وجود دارد که بتوان روشنایی کرم‌های شب تاب را، به شیوه‌ای مشابه با تابع برازندگی در «الگوریتم‌های ژنتیک» (Genetic Algorithms) تعریف کرد.

بر اساس این سه قاعده، می‌توان گام‌های اساسی مورد نیاز برای پیاده‌سازی الگوریتم کرم شب تاب را فرمول‌بندی کرد. گام‌های اساسی الگوریتم کرم شب تاب در «شبه کد» (Pseudo-Code) زیر نمایش داده شده است.

الگوریتم کرم شب تاب (Firefly Algorithm | FA)

تابع هدف: f(x),x=(x1,.....,xd)Tf(x),x=(x1,.....,xd)T .

تولید جمعیت اولیه از کرم‌های شب تاب: xi,(i=1,2,3,.....,n)xi,(i=1,2,3,.....,n) .

پارامتر شدت نور lili در xixi ، از طریق جای‌گذاری مقدار xixi در تابع f(xi)f(xi) به دست می‌آید.

تعریف کردن «ضریب جذب نور» (Light Absorption Coefficient) یا γγ .

حلقه While: تا زمانی که (t<MaxGenerationt<MaxGeneration)

حلقه For: (fori=1tonfori=1ton) به ازاء تمامی nn کرم شب تاب موجود در جمعیت

حلقه For: (forj=1toiforj=1toi) به ازاء تمامی nn کرم شب تاب موجود در جمعیت

شرط if: در صورتی که (lj>lilj>li)، کرم شب تاب ii به سمت کرم شب تاب jj حرکت داده می‌شود (در فضای جستجوی d-بُعدی مسأله). پایان شرط if

مقدار پارامتر جذابیت (Attractiveness) بر اساس فاصله rr و مطابق با رابطه exp(−γr)exp(−γr) محاسبه و به‌روزرسانی می‌شود. راه حل‌های (جواب‌های کاندید) جدید ارزیابی و پارامتر شدت نور (Light Intensity) به‌روزرسانی می‌شود.
پایان حلقه For.
پایان حلقه For.
کرم‌های شب تاب ارزیابی و بهترین جواب کاندید در این نسل (Generation) مشخص می‌شود.
پایان حلقه While.
جواب‌های نهایی «پس پردازش» (PostProcess) و مصورسازی‌های مرتبط با خروجی نهایی انجام می‌شود.
از جهات خاصی، شباهت‌های مفهومی میان الگوریتم کرم شب تاب (و الگوریتم‌های مشتق شده از آن) و «الگوریتم غذایابی باکتریایی» (Bacteria Foraging Algorithm | BFA) وجود دارد. الگوریتم غذایابی باکتریایی (Bacteria Foraging Algorithm) از رفتار جمعی باکتری‌ها در پیدا کردن منابع غذایی لازم برای بقاء الهام گرفته شده است. در الگوریتم BFA، بخشی از جذابیت میان باکتری‌ها (جذب شدن باکتری‌ها به سمت یکدیگر)، مبتنی بر «برازندگی» (Fitness) آن‌ها و بخش دیگر مبتنی بر فاصله میان آن‌ها است. در حالی که در الگوریتم کرم شب تاب (و نسخه‌های مختلف آن)، پارامتر جذابیت (Attractiveness) به تابع هدف و زوال (Decay) یکنواخت مقدار جذابیت، متناسب با افزایش فاصله میان کرم‌های شب تاب مرتبط است.
با این حال، با توجه به پراکندگی عوامل (منظور موجودیت‌های موجود در جمعیت یا همان کرم‌های شب تاب) موجود در الگوریتم کرم شب تاب در فضای جستجوی مسأله، قابل تنظیم بودن رؤیت‌پذیری عوامل نسبت به یکدیگر (بر اساس فاصله میان آن‌ها) و تنوع موجود در جمعیت با توجه به جذابیت متغیر عوامل نسبت به یکدیگر، انتظار می‌رود که الگوریتم کرم شب تاب بتواند به شکل بهتر و کارآمدتری، فضای جستجوی مسأله را مورد کاوش قرار دهد.

جذابیت (Attractiveness) در الگوریتم کرم شب تاب

در هنگام فرمول‌بندی کردن الگوریتم کرم شب تاب، نیاز است تا به دو مسأله اساسی توجه شود: «تغییرات شدت نور» (Variation of Light Intensity) و فرمول‌بندی کردن جذابیت متقابل کرم‌های شب تاب. برای ساده‌سازی فرآیند فرمول‌بندی کردن الگوریتم کرم شب تاب، می‌توان فرض کرد که جذابیت یک کرم شب تاب، بر اساس روشنایی این عامل (کرم شب تاب) مشخص می‌شود؛ که روشنایی کرم شب تاب نیز به نوبه خود، به تابع هدف «کدبندی شده» (Encoded) جهت حل یک مسأله خاص مرتبط است. در ساده‌ترین حالت و برای یک مسأله بیشینه‌سازی، می‌توان مقدار پارامتر روشنایی II یک کرم شب تاب را، که در مکان خاص XX قرار دارد، از طریق رابطه‌ای نظیر I(X)∝f(x)I(X)∝f(x) به دست آورد. با این حال، مقدار پارامتر جذابیت ββ یک کرم شب تاب نسبی است و باید توسط دیگر کرم‌ها مشخص شود (به عبارت دیگر، فاصله میان کرم‌های شب تاب، نقش مستقیمی در جذابیت آن‌ها (جذب آن‌ها به سمت یکدیگر) خواهد داشت). بنابراین، پارامتر جذابیت ββ ، بر اساس فاصله rijrij میان کرم شب تاب ii و کرم شب تاب jj ، تغییر پیدا خواهد کرد. علاوه بر این، هر چقدر که فاصله از منبع نور بیشتر شود، شدت نور (Light Intensity) کاهش پیدا می‌کند. همچنین، نور هنگام گذر از «واسطه‌هایی» (Medium) نظیر هوا، توسط آن جذب می‌شود.در نتیجه، الگوریتم کرم شب تاب باید به گونه‌ای فرمول‌بندی شود که بر اساس «درجات جذب» (Degrees of Absorption) مختلف، مقدار پارامتر جذابیت (Attractiveness) نیز تغییر پیدا کند.
در ساده‌ترین حالت ممکن، پارامتر شدت نور I(r)I(r) بر اساس «قانون مربع معکوس» (Inverse Square Law) تغییر پیدا خواهد کرد:

I(r)=Isr2I(r)=Isr2

در این قانون، پارامتر IsIs شدت نور در منبع را نشان می‌دهد. با در اختیار داشتن یک واسط (نظیر هوا) با ضریب ثابتِ جذب نور، γγ ، شدت نور II ، بر اساس فاصله rr و از طریق رابطه زیر تغییر پیدا خواهد کرد:

I=I0e−γrI=I0e−γr

در این رابطه، I0I0 شدت نور اصلی را نشان می‌دهد. برای اجتناب از رخ دادن «تکینی» (Singularity) در رابطه Isr2Isr2 و در نقطه r=0r=0 ، می‌توان «اثر ترکیبی» (Combined Effect) قانون مربع معکوس و اصل جذب نور توسط واسط هوا را با استفاده از تابع زیر (که به فرم گاوسی (Gaussian) است) تقریب زد:

I(r)=I0e−γr2I(r)=I0e−γr2

بعضی مواقع به تابعی نیاز است که بتواند به طور یکنواخت و با نرخ آهسته‌تری کاهش پیدا کند. در چنین حالتی، می‌توان از «تقریب» (Approximation) زیر استفاده کرد:

I(r)=I01+γr2I(r)=I01+γr2

در یک فاصله کوتاه (فاصله بین دو کرم شب تاب بسیار کم باشد)، دو فرم ارائه شده از تابع I(r)I(r) تقریبا با یکدیگر برابر هستند. چنین پدیده‌ای به این دلیل است که برای مقدار r=0r=0 ، «بسط‌های سری» (Series Expansions) این دو تابع، تا مرتبه O(r3)O(r3) ، معادل یکدیگر خواهند بود:

11+γr2≈1−γr2+12γ2r4+....11+γr2≈1−γr2+12γ2r4+....

e−γr2≈1−γr2+12γ2r4+....e−γr2≈1−γr2+12γ2r4+....

از آنجایی که جذابیت یک کرم شب تاب، متناسب با شدت نوری است که توسط کرم‌های شب تاب مجاور (همسایه) مشاهده می‌شود، پارامتر جذابیت ββ یک کرم شب تاب را می‌توان از طریق رابطه زیر تعریف کرد:

β(r)=β0e−γr2β(r)=β0e−γr2

در این رابطه، β0β0 برابر با جذابیت کرم شب تاب در فاصله r=0r=0 است. از آنجایی که محاسبه رابطه 1(1+r2)1(1+r2) از محاسبه یک «تابع نمایی» (Exponential Function) سریع‌تر است، به جای تابع بالا می‌توان از تابع زیر استفاده کرد و مقدار پارامتر جذابیت را از طریق رابطه زیر محاسبه کرد:

β(r)=β01+γr2β(r)=β01+γr2

رابطه بالا، پارامتری به نام فاصله مشخصه (Characteristic Distance) یا مقیاس طول (Length Scale) را به صورت Γ=1γΓ=1γ تعریف می‌کند که توسط آن، پارامتر جذابیت، به طرز قابل ملاحظه‌ای از فرم β0β0 ، به فرم β0e−1β0e−1 تغییر پیدا می‌کند.

در هنگام پیاده‌سازی الگوریتم کرم شب تاب (و نسخه‌های مختلف آن)، تابع β(r)β(r) می‌تواند به شکل «توابع نزولی یکنواخت» (Monotonically Decreasing Functions)، نظیر شکل «تعمیم داده شده» (Generalized) زیر نمایش داده شود:

β(r)=β0e−γrm,(m≥1)β(r)=β0e−γrm,(m≥1)

در صورتی که پارامتر γγ ثابت فرض شود، وقتی که m→∞m→∞ ، پارامتر فاصله مشخصه یا مقیاس طول به شکل Γ=γ−1mΓ=γ−1m تغییر پیدا می‌کند. متقابلا، با در اختیار داشتن مقدار پارامتر فاصله مشخصه یا مقیاس طول در یک مسأله بهینه‌سازی، پارامتر ΓΓ می‌تواند به شکل γ=1Γmγ=1Γm و به عنوان یک «مقدار اولیه» (Initial Value) معمولی مورد استفاده قرار بگیرد.

فاصله (Distance) و «حرکت» (Movement) در الگوریتم کرم شب تاب

فاصله (Distance) میان دو کرم شب تاب ii و jj ، که به ترتیب در مختصات مکانی xixi و xjxj قرار دارند، برابر با «فاصله دکارتی» (Cartesian Distance) میان آن‌ها به فرم زیر است:

rij=‖xi−xj‖=rij=‖xi−xj‖=

√∑k=1d(xi,k−xj,k)2rij=‖xi−xj‖=∑dk=1(xi,k−xj,k)2

در این رابطه، xi,kxi,k برابر با kk -اُمین مؤلفه مختصات مکانی مرتبط با ii -اُمین کرم شب تاب (xixi) است. در یک فضای جستجوی دوبُعدی، فاصله میان دو کرم شب تاب ii و jj از طریق رابطه زیر محاسبه می‌شود:

rij=‖xi−xj‖=√(xi−xj)2−(yi−yj)2rij=‖xi−xj‖=(xi−xj)2−(yi−yj)2

حرکت (Movement) کرم شب تابی (به عنوان نمونه، کرم شب تاب ii) که جذب یک کرم شب تاب جذاب‌تر (روشن‌تر) دیگر شده است، از طریق رابطه زیر مشخص می‌شود:

$$xi=xi+\beta 0e-\gamma r2ij(xj-xi)+\alpha(rand-12)xi=xi+\beta 0e-\gamma rij2(xj-xi)+\alpha(rand-12)$$

در اینجا، عبارت دوم سمت راست رابطه، تأثیر پارامتر جذابیت را در محاسبه مقدار حرکت کرم شب تاب ii نشان می‌دهد.

همچنین، عبارت سوم جهت تصادفی‌سازی (Randomization) به کار می‌رود که در آن αα نقش پارامتر تصادفی‌سازی را ایفا می‌کند. عبارت rand نیز یک «مولد اعداد تصادفی» (Random Number Generator) است که مقادیر تصادفی تولید شده توسط آن، از «توزیع یکنواخت» (Uniform Distribution) در بازه [0,1][0,1] تبعیت می‌کنند.

در پیاده‌سازی‌های انجام شده از الگوریتم کرم شب تاب جهت حل مسائل بهینه‌سازی، مقدار پارامتر β0=1β0=1 در نظر گرفته شده است و مقدار پارامتر αα ، از بازه α∈[0,1]α∈[0,1] انتخاب می‌شود. علاوه بر این، مقدار عبارت تصادفی‌سازی (عبارت سوم در رابطه بالا) را می‌توان بر اساس یک «توزیع نرمال» (Normal Distribution) به فرم N(0,1)N(0,1) یا هر توزیع دلخواه دیگر محاسبه کرد.

همچنین، در شرایطی که مقیاس‌های عددی متغیرهای مسأله تفاوت فاحشی با یکدیگر داشته باشند، به عنوان نمونه، مقیاس عددی یکی از متغیرهای مسأله برابر با [10−5,105][10−5,105] باشد و مقیاس عددی یک متغیر دیگر، برابر یا [−0.01,0.01][−0.01,0.01] باشد، در چنین حالتی، بهتر است که پارامتر αα در عبارت تصادفی‌سازی، با پارامتر αSkαSk جا به جا شود. مؤلفه‌های SkSk در این پارامتر، بردار «پارامترهای مقیاس‌گذاری» (Scaling Parameters) نام دارد. در صورتی که مسأله بهینه‌سازی مد نظر d-بُعدی باشد، پارامترهای مقیاس‌گذاری Sk(K=1,2,...,d)Sk(K=1,2,...,d) باید بر اساس مقیاس‌های واقعی مسأله بهینه‌سازی مورد نظر مشخص شوند. در رابطه مشخص کردن حرکت (Movement) کرم‌های شب تاب، پارامتر γγ ، تغییرات جذابیت کرم‌های شب تاب در الگوریتم کرم شب تاب را مشخص می‌کند. همچنین، مقدار این پارامتر نقش بسیار مهمی در مشخص کردن سرعت همگرایی، رفتار الگوریتم کرم شب تاب در جستجوی فضای مسأله و حل مسأله بهینه‌سازی داده شده دارد. در تئوری، مقدار پارامتر γγ از طریق γ∈[0,∞)γ∈[0,∞) مشخص می‌شود، ولی در عمل، این پارامتر به وسیله γ=O(1)γ=O(1) مقداردهی می‌شود؛ این مقدار به وسیله پارامتر فاصله مشخصه یا مقیاس طول (پارامتر ΓΓ) سیستمی که قرار است بهینه‌سازی شود، مشخص می‌شود. بنابراین، در بیشتر کاربردها، این پارامتر معمولا مقداری بین 0.010.01 و 100100 به خود می‌گیرد.

مقیاس‌گذاری (Scaling) مسائل بهینه‌سازی و تجزیه و تحلیل مجانبی

شایان ذکر است که محاسبه فاصله rr در الگوریتم کرم شب تاب (و پیاده‌سازی‌های مختلف آن)، که در بخش قبل توضیح داده شد، محدود به استفاده از «فاصله اقلیدسی» (Euclidean Distance) نیست و می‌توان از هر معادله دلخواهی برای محاسبه فاصله میان کرم‌های شب تاب استفاده کرد. به عبارت دیگر، بسته به نوع مسأله بهینه‌سازی که قرار است توسط الگوریتم کرم شب تاب حل شود، می‌توان معادلات مختلفی را برای محاسبه فاصله rr در فضای n-بُعدی استفاده کرد.

به عنوان نمونه، در مسائل «زمان‌بندی کار» (Job Scheduling)، فاصله rr می‌تواند به عنوان پارامتر «تأخیر زمانی» (Time Lag) و یا «بازه زمانی» (Time Interval) تعریف شود. همچنین، در شبکه‌های پیچیده نظیر اینترنت و «شبکه‌های اجتماعی» (Social Networks)، که مسائل در قالب «گراف» (Graph) نمایش داده می‌شوند، فاصله rr را می‌توان به عنوان ترکیبی از «درجه خوشه‌بندی محلی» (Degree of Local Clustering) و «متوسط نزدیکی رأس‌ها» (Average Proximity of Vertices) تعریف کرد. به عبارت دیگر، هر معیاری که به شکل مؤثر بتواند مقادیر مطلوب و مورد نظر در مسأله بهینه‌سازی را مشخص کند، می‌تواند به عنوان فاصله rr تعریف شود.مقدار پارامتر مقیاس طول (پارامتر ΓΓ)، باید متناسب با مقیاس مسأله بهینه‌سازی مورد نظر، در الگوریتم کرم شب تاب در نظر گرفته شود. در صورتی که ΓΓ ، مقیاس مسأله بهینه‌سازی مورد نظر برای یک جمعیت متشکل از تعداد زیادی کرم شب تاب (n≫mn≫m ، در اینجا nn تعداد کرم‌های شب در الگوریتم کرم شب تاب و

mm تعداد «بهینه‌های محلی» (Local Optimums) را نشان می‌دهد) باشد، بهتر است که مکان‌های اولیه nn کرم شب تاب موجود در جمعیت، به شکل نسبتا یکنواختی در فضای جستجوی (Search Space) توزیع شده باشند؛ توصیه می‌شود که نحوه توزیع شدگی کرم‌های شب تاب در فضای جستجوی مسأله (مقداردهی اولیه جمعیت در الگوریتم کرم شب تاب)، تا حدودی شبیه به مقداردهی اولیه (Initialization) در «شبیه‌سازی‌های مونته کارلو» (Monte-Carlo Simulations) باشد.

هر چقدر که تعداد تکرارهای (نسل) بیشتری از الگوریتم کرم شب تاب انجام می‌شود، کرم‌های شب تاب به شیوه‌ای تصادفی، به بهینه‌های محلی (از جمله بهینه‌های سراسری؛ بهینه سراسری در مسائل بهینه‌سازی، زیر مجموعه‌ای از بهینه‌های محلی محسوب می‌شوند) مسأله بهینه‌سازی مورد نظر همگرا می‌شوند. با مقایسه بهترین جواب‌ها در میان بهینه‌های محلی، جواب بهینه سراسری مسأله بهینه‌سازی توسط الگوریتم کرم شب تاب مشخص می‌شود.

در این بخش، هدف ثابت کردن این موضوع است که وقتی $n\to\infty$ و $t\gg1$ باشد، الگوریتم کرم شب به جواب بهینه سراسری مسأله بهینه‌سازی مورد نظر همگرا می‌شود (در اینجا، nn برابر با تعداد کرم‌های شب تاب در جمعیت اولیه و tt برابر با تعداد نسل‌های (تکرار) الگوریتم کرم شب تاب است). در آزمایشات عملی انجام شده رو توابع بهینه‌سازی مرجع، الگوریتم کرم شب تاب عملکرد بسیار خوبی از خود نشان می‌دهد؛ به گونه‌ای که در کمتر از ۵۰ تا ۱۰۰ تکرار (نسل)، این الگوریتم به جواب بهینه سراسری همگرا می‌شود. نتایج ارزیابی الگوریتم کرم شب تاب با استفاده از «توابع بهینه‌سازی معیار» (Benchmark Functions)، در بخش‌های بعدی نمایش داده خواهد شد. برای اثبات همگرایی الگوریتم کرم شب تاب به جواب بهینه سراسری، دو مورد محدود کننده مهم باید مورد بررسی قرار گرفته شوند:

زمانی که $\gamma\to0$، پارامتر $\gamma\gamma$ به سمت صفر میل می‌کند.

زمانی که $\gamma\to\infty$، پارامتر $\gamma\gamma$ به سمت بی‌نهایت میل می‌کند.

در حالتی که $\gamma\to0$، پارامتر $\gamma\gamma$ به سمت صفر میل می‌کند، پارامتر جذابیت مقداری ثابت و برابر صفر خواهد بود ($\beta=\beta0$) و پارامتر مقیاس به سمت بی‌نهایت میل می‌کند ($\Gamma\to\infty$). چنین پدیده‌ای را می‌توان به این شکل تصور کرد که شدت نور (Light Intensity) کرم‌های شب تاب در آسمان کاهش پیدا نخواهد کرد. به عبارت دیگر، یک کرم شب تاب چشمک‌زن در هر نقطه‌ای از ناحیه مورد نظر (معادل فضای جستجوی مسأله) قابل رؤیت خواهد بود. بنابراین، الگوریتم کرم شب تاب به راحتی قادر خواهد بود به یک نقطه «بهینه» (Optimum) در فضای جستجوی مسأله همگرا شود (معمولا این بهینه، همان بهینه سراسری مسأله خواهد بود). وقتی که $\gamma\to0$، پارامتر $\gamma\gamma$ به سمت صفر میل می‌کند، الگوریتم کرم شب تاب متناظر با الگوریتم بهینه‌سازی ازدحام ذرات (PSO) خواهد بود. همچنین، عملکرد و کارایی این حالت خاص از الگوریتم کرم شب تاب، همانند الگوریتم PSO خواهد بود.

در حالتی که $\gamma\to\infty$، پارامتر $\gamma\gamma$ به سمت بی‌نهایت میل می‌کند، $\beta(r)\to\delta(r)$ («تابع دلتای دیراک» (Dirac Delta Function)) و پارامتر مقیاس به سمت صفر میل می‌کند ($\Gamma\to0$). این بدین معنی است که جذابیت یک کرم شتاب، برای دیگر کرم‌های شب تاب موجود در محیط (فضای جستجوی مسأله) صفر است و یا اینکه کرم‌های شب تاب «نزدیک بین» (Short-Sighted) هستند. چنین پدیده‌ای معادل پرواز کردن کرم‌های شب تاب در یک ناحیه مه‌آلود (Foggy)، به صورت کاملا تصادفی است. در چنین حالتی، دیگر کرم‌های شب تاب نیز قابل رؤیت نخواهند بود و هر کرم شب تاب، به طور کاملا تصادفی در محیط پرسه می‌زند. بنابراین در چنین شرایطی، الگوریتم کرم شب تاب به یک روش «جستجوی کاملا تصادفی» (Completely Random Search) تبدیل می‌شود.

این دو مورد، حالات بسیار خاص از الگوریتم کرم شب تاب محسوب می‌شوند؛ به عبارت دیگر، نقاط حداکثری (Extremes) و استثناء در الگوریتم کرم شب تاب محسوب می‌شوند. به طور کلی، الگوریتم کرم شب تاب همیشه بین این دو حالت خاص قرار می‌گیرد. همچنین، با تنظیم مناسب پارامترهای $\gamma\gamma$ و $\alpha\alpha$، الگوریتم کرم شب تاب قادر خواهد بود عملکرد بهتری نسبت به الگوریتم بهینه‌سازی ازدحام ذرات (PSO) و روش جستجوی تصادفی (Random Search) از خود نشان دهد.

یکی از ویژگی‌های مهم الگوریتم کرم شب تاب این است که می‌تواند به شکل بسیار مؤثری، علاوه بر یافتن «بهینه سراسری» (Global Optimum)، به طور همزمان بهینه‌های محلی مسائل بهینه‌سازی را نیز پیدا کند. چنین مزیت مهمی در الگوریتم کرم شب تاب (در طول فرایند بهینه‌سازی توابع هدف یک مسأله بهینه‌سازی خاص)، در بخش‌های بعدی نمایش داده می‌شود.
مزیت دیگر الگوریتم کرم شب که آن را از دیگر الگوریتم‌های بهینه‌سازی مرسوم متمایز می‌کند این است که کرم‌های شب تاب مختلف، تقریبا مستقل از یکدیگر عمل می‌کنند. چنین ویژگی مهمی، الگوریتم کرم شب تاب را به انتخابی ایده‌آل برای «پیاده‌سازی‌های موازی» (Parallel Implementation) از الگوریتم‌های تکاملی تبدیل می‌کند. همچنین، از آنجایی که کرم‌های شب تاب به شکل فشرده‌تری حول بهینه‌های (Optimums) موجود در فضای جستجوی مسأله جمع می‌شوند، بهتر از الگوریتم ژنتیک و الگوریتم بهینه‌سازی ازدحام ذرات عمل می‌کنند (به عنوان نمونه، کروموزم‌های تعریف شده در الگوریتم ژنتیک، در اثر عملگرهای تکاملی به ویژه عملگر ترکیب یا آمیزش، در فضای جستجو پرش می‌کنند، در حالی که در الگوریتم کرم شب تاب چنین فرایندی مشاهده نمی‌شود). در پیاده‌سازی موازی صورت گرفته از الگوریتم کرم شب تاب، تعامل میان «زیر ناحیه‌ها» (Subregions) به حداقل می‌رسد.

بهینه‌سازی چندمُدی با بهینه‌های چندگانه

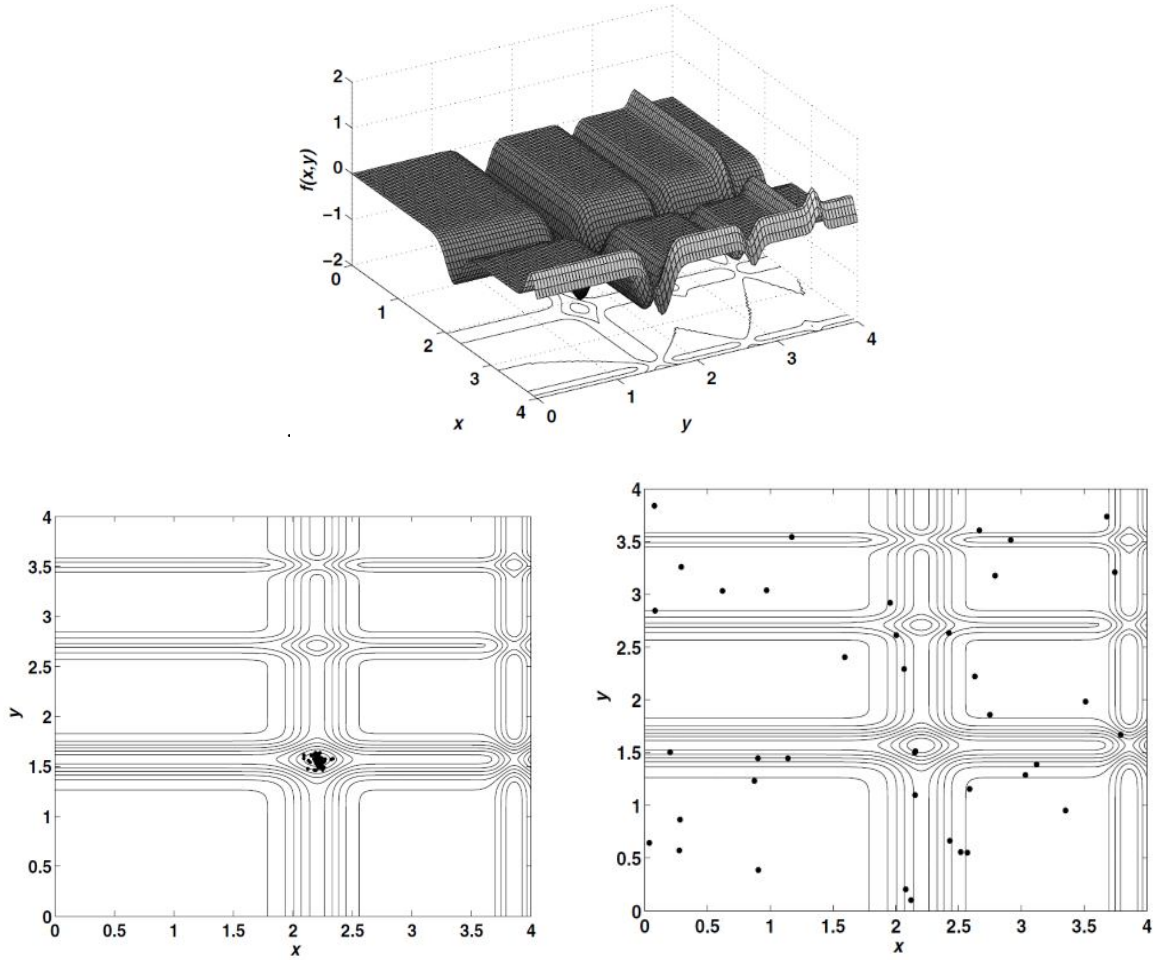
برای اینکه نحوه کارکرد الگوریتم شب تاب در بهینه‌سازی توابع مورد بررسی قرار بگیرد، از توابع معیار مختلفی استفاده شده است تا صحت عملکرد این الگوریتم سنجیده شود (نتایج حاصل از ارزیابی عملکرد الگوریتم کرم شب تاب در ادامه نمایش داده شده است). همچنین، کدهای پیاده‌سازی این الگوریتم در زبان‌های برنامه‌نویسی مختلف نمایش داده خواهد شد.

ارزیابی و سنجش عملکرد الگوریتم کرم شب تاب

در این مطلب، به عنوان نمونه، از الگوریتم کرم شب تاب برای پیدا کردن بهینه سراسری تابع Michalewicz استفاده می‌شود. فرم ریاضی تابع Michalewicz، به شکل زیر است:

$$f(x)=-d\sum i=1\sin(xi)[\sin(ix2i\pi)]2mf(x)=-\sum i=1dsin\ (xi)[sin\ (ixi2\pi)]2m$$

در این تابع، $m=10m=10$ و $d=1,2,....d=1,2,....$ است. بهینه سراسری تابع، $f*\approx-1.0801f*\approx-1.0801$ ، در یک فضای دوبُعدی، در نقطه $(2.20319,1.57049)(2.20319,1.57049)$ رخ می‌دهد. نقطه بهینه سراسری تابع Michalewicz، پس از ۱۰ تکرار الگوریتم کرم شب تاب و به ازاء ۴۰ کرم شب تاب موجود در جمعیت (تابع، حدودا ۴۰۰ بار، برای یافتن نقطه بهینه ارزیابی می‌شود) یافت خواهد شد. در این شبیه‌سازی و ارزیابی تابع هدف Michalewicz با استفاده از الگوریتم کرم شب تاب)، مقادیر پارامترها برابر با $\alpha=0.2\alpha=0.2$ ، $\gamma=1\gamma=1$ و $\beta0=1\beta0=1$ در نظر گرفته شده‌اند.

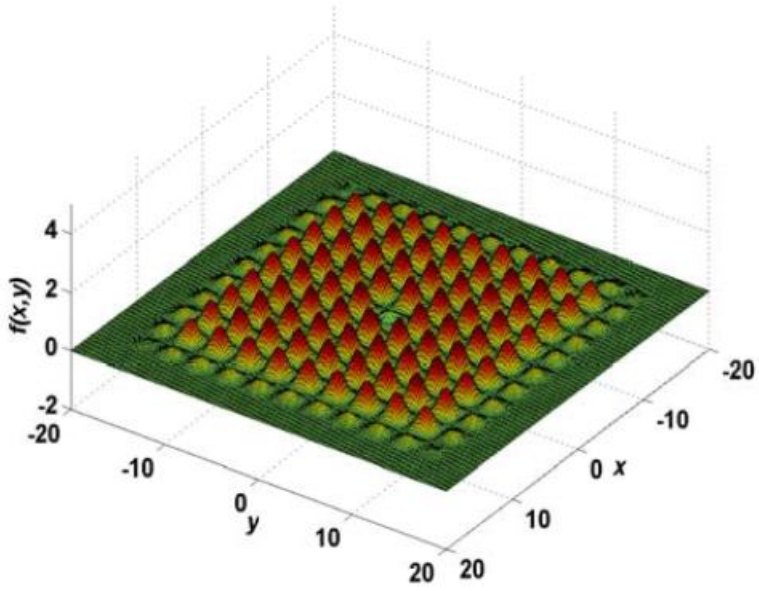


مقداردهی اولیه جمعیت متشکل از ۴ کرم شب تاب

در مرحله بعد عملکرد الگوریتم کرم شب تاب در پیدا کردن نقطه بهینه سراسری توابع سخت‌تر و پیچیده‌تر ارزیابی می‌شود. نتایج ارزیابی روی این دسته از توابع نشان می‌دهد که الگوریتم کرم شب تاب عملکرد بهتری نسبت به دیگر الگوریتم‌های فرا اکتشافی موجود دارد. به عنوان نمونه، از الگوریتم کرم شب تاب برای پیدا کردن بهینه سراسری تابع Yang استفاده می‌شود. این تابع، یک تابع چندمُدی است که الگویی شبیه به یک موج ایستاده دارد. فرم ریاضی تابع Yang، به شکل زیر است:

$$f(x)=[e-\sum_{i=1}^{d}(x_i/a)^{2m}-2e^{-\sum_{i=1}^{d}x_i^2}]\cdot\prod_{i=1}^{d}\cos^2 x_i, m=5$$

همانطور که در شکل زیر مشهود است، این تابع چندمُدی، چندین بهینه محلی (کمینه (Valley) و بیشینه (Peak) محلی) دارد. نقطه بهینه سراسری این تابع، $f^*=-1$ ، در نقطه $(0,0,\ldots,0)$ و در ناحیه $-20\le x_i\le 20$ قرار دارد. در این تابع، $i=1,2,\ldots,d$ و $a=15$ است. نمایش سه‌بُعدی تابع Yang در شکل زیر ارائه شده است.



مقایسه الگوریتم کرم شب تاب با الگوریتم ژنتیک و PSO

مطالعات مختلف نشان می‌دهد که الگوریتم بهینه‌سازی ازدحام ذرات (PSO) می‌تواند عملکرد به مراتب بهتری نسبت به الگوریتم ژنتیک (GA) و الگوریتم‌های بهینه‌سازی مرسوم، در حل بسیاری از مسائل بهینه‌سازی از خود نشان دهد. یک دلیل محتمل برای عملکرد بهینه الگوریتم بهینه‌سازی ازدحام ذرات، توانایی بهترین تخمین یافت شده از جواب بهینه (در هر مرحله)، در مخابره کردن (Broadcasting) یا ارتباط برقرار کردن با دیگر ذرات موجود در جمعیت است؛ چنین ویژگی مهمی سبب می‌شود تا الگوریتم به شکل بهتر و سریع‌تری به جواب بهینه سراسری همگرا شود. در این مرحله، نتایج ارزیابی الگوریتم کرم شب تاب و مقایسه آن با نتایج حاصل از الگوریتم ژنتیک و PSO نمایش داده می‌شود. برای چنین کاری، از توابع معیار استاندارد استفاده شده است. برای پیاده‌سازی و ارزیابی الگوریتم ژنتیک و مقایسه آن با الگوریتم کرم شب تاب، از الگوریتم ژنتیک استاندارد استفاده شده است؛ در الگوریتم ژنتیک استاندارد از اصل «نخبه‌گرایی یا الیتیسم» (Elitism) استفاده نشده است. «احتمال جهش» (Mutation Probability | pmpm) برابر با pm=0.05pm=0.05 و «احتمال ترکیب» (Crossover Probability | pcpc) برابر با pc=0.95pc=0.95 در نظر گرفته شده است. برای پیاده‌سازی و ارزیابی الگوریتم بهینه‌سازی ازدحام ذرات (PSO) و مقایسه آن با الگوریتم کرم شب تاب، از نسخه استاندارد الگوریتم PSO استفاده شده است. در الگوریتم استاندارد بهینه‌سازی ازدحام ذرات، از پارامترهای یادگیری α≈β≈2α≈β≈2 استفاده شده است و ویژگی «تصحیح اینرسی» (Inertia Correction) به کار گرفته نشده است. در مرحله ارزیابی و مقایسه عملکرد الگوریتم کرم شب تاب با دیگر الگوریتم‌های بهینه‌سازی مشابه، مقادیر متفاوتی برای اندازه جمعیت در نظر گرفته شد (n=15to200n=15to200). با این حال، در بیشتر شبیه‌سازی‌های انجام شده و برای غالب توابع استفاده شده، مقدار n=15to50n=15to50 ، به عنوان مناسب‌ترین مقدار برای اندازه جمعیت مشخص شد. بنابراین، از مقدار ثابت n=40n=40 ، به عنوان اندازه جمعیت، در تمامی شبیه‌سازی‌ها و ارزیابی‌ها استفاده شده است.

برای اینکه تحلیل‌های آماری معناداری روی نتایج حاصل از ارزیابی الگوریتم‌ها انجام شود، پس از پیاده‌سازی الگوریتم‌ها، شبیه‌سازی‌های گسترده‌ای روی آن‌ها انجام شد و هر کدام از الگوریتم‌ها حداقل ۱۰۰ بار اجرا شدند. اجرای الگوریتم‌ها تنها زمانی متوقف می‌شود که تغییرات مقادیر توابع معیار، از یک حد آستانه داده شده، ϵ≤10−5ϵ≤10−5 کمتر باشند. نتایج حاصل از شبیه‌سازی‌های انجام شده و مقایسه عملکرد الگوریتم‌های تکاملی در جدول زیر نمایش داده شده است.

فرم ریاضی توابع مشخص شده در جدول زیر، در ادامه نمایش داده شده است:

فرم ریاضی تابع Michalewicz:

$$f(x)=-d\sum_{i=1}\sin(xi)[\sin(ix2i\pi)]2mf(x)=-\sum_{i=1}dsin\ (xi)[sin\ (ixi2\pi)]2m$$

فرم ریاضی تابع Yang:

$$f(x)=[e-\sum_{di=1}(xi/a)2m-2e-\sum_{di=1x2i}]\cdot d\prod_{i=1}\cos2xi,m=5f(x)=[e-\sum_{i=1}d(xi/a)2m-2e-\sum_{i=1dxi2}]\cdot\prod_{i=1dcos2}\ xi,m=5$$

فرم ریاضی تابع Rosenbrock:

$$f(x)=n-1\sum_{i=1}[100(xi+1-x2i)2+(xi-1)2]f(x)=\sum_{i=1n-1}[100(xi+1-xi2)2+(xi-1)2]$$

فرم ریاضی تابع De Jong:

$$f(x)=n\sum_{i=1x2i}-5.12\le xi\le5.12f(x)=\sum_{i=1nxi2}-5.12\le xi\le5.12$$

فرم ریاضی تابع Schwefel:

$$f(x)=n\sum_{i=1}-xi.\sin(\sqrt{|xi|})-500\le xi\le500f(x)=\sum_{i=1n}-xi.\sin\ (\sqrt{|xi|})-500\le xi\le500$$

فرم ریاضی تابع Ackley:

$$f(x)=-20\ e^{\left(-0.2\times\sqrt{\frac{1}{n}\sum_{i=1x2i}}\right)}-e^{\left[\frac{1}{n}\sum_{i=1}\cos(2\pi xi)\right]}+20+e1f(x)=-20\ e^{(-0.2\times\sqrt{\frac{1}{n}\sum_{i=1nxi2}})}-e^{[\frac{1}{n}\sum_{i=1ncos}\ (2\pi xi)]}+20+e1$$

فرم ریاضی تابع Rastrigin:

$$f(x)=10n+n\sum_{i=1}(x2i-10\cos(2\pi xi)),n=9f(x)=10n+\sum_{i=1n}(xi2-10\cos\ (2\pi xi)),n=9$$

$$-5.12\le xi\le5.12,f(0,0,....,0)=0-5.12\le xi\le5.12,f(0,0,....,0)=0$$

فرم ریاضی تابع Easom:

$$f(x)=-\cos(x1)\cos(x2)e[-(x1-\pi)2-(x2-\pi)2]f(x)=-\cos\ (x1)\cos\ (x2)e[-(x1-\pi)2-(x2-\pi)2]$$

فرم ریاضی تابع Griewank:

$$f(x)=14000n\sum_{i=1x2i}-n\prod_{i=1}\cos(xi\sqrt{i})+1f(x)=14000\sum_{i=1nxi2}-\prod_{i=1ncos}\ (xii)+1$$

فرم ریاضی تابع Shubert:

$$f(x)=5\sum_{i=1}isin[(i+1)x+i]f(x)=\sum_{i=15}isin\ [(i+1)x+i]$$

اعداد نمایش داده شده در جدول زیر به این فرم هستند:

[تعداد متوسط ارزیابی‌های تابعی انجام شده ± انحراف از معیار (نرخ موفقیت الگوریتم)]

بنابراین وقتی امتیاز عملکرد الگوریتم کرم شب تاب در بهینه‌سازی تابع Michalewicz به شکل 3752±725(99%)3752±725(99%) نمایش داده شده باشد، به چه معناست؟ این بدین معنی است که تعداد متوسط ارزیابی‌های تابعی انجام شده توسط الگوریتم کرم شب تاب برابر با ۳۵۷۲ است و انحراف معیار آن برابر با ۷۲۵ است. همچنین، نرخ موفقیت الگوریتم شب تاب در همگرایی به نقطه بهینه سراسری برابر با ۹۹ درصد است.

همانطور که در جدول بالا قابل مشاهده است، الگوریتم کرم شب تاب، کارآیی به مراتب بهتری نسبت به الگوریتم ژنتیک و الگوریتم بهینه‌سازی ازدحام ذرات (PSO) از خود نشان می‌دهد. همچنین، نرخ موفقیت الگوریتم کرم شب تاب در همگرایی به جواب بهینه سراسری به مراتب بالاتر از دیگر الگوریتم‌ها است. در سیستم‌های محاسبات امروزی (لپ‌تاپ، کامپیوتر شخصی و سایر موارد)، ارزیابی عملکرد الگوریتم‌ها در بهینه‌سازی توابع، تقریبا به صورت «آنی» (Instantaneous) انجام می‌شود.

به عنوان نمونه، انجام ۱۰ هزار ارزیابی تابعی روی یک پردازنده (کامپیوترهای شخصی)، چیزی حدود ۵ ثانیه زمان می‌برد. حتی با در نظر گرفتن امکانات گرافیکی جهت نمایش مکان کرم‌های شب تاب (نمونه‌های موجود در جمعیت) در فضای جستجو، به صورت تعاملی، اجرای الگوریتم تنها چیزی حدود چند دقیقه زمان خواهد برد.

شایان ذکر است که این امکان وجود دارد که بتوان از روش‌های رسمی «آزمون فرض آماری» (Statistical Hypothesis Testing) جهت صحت‌سنجی «معنادار بودن» (Significance) نتایج حاصل استفاده کرد.

جمع‌بندی

در این مطلب، نحوه فرمول‌بندی کردن الگوریتم کرم شب تاب ارائه شد. همچنین، شباهت‌ها و تفاوت‌های این الگوریتم با الگوریتم بهینه‌سازی ازدحام ذرات (PSO) مورد بررسی قرار گرفت. در مرحله بعد، نحوه پیاده‌سازی و مقایسه این الگوریتم با دیگر الگوریتم‌های مشابه نظیر الگوریتم ژنتیک (GA) و الگوریتم PSO شرح داده شد. نتایج ارزیابی و شبیه‌سازی الگوریتم‌های تکاملی در همگرایی به جواب بهینه سراسری توابع معیار مختلف نشان می‌دهد که الگوریتم PSO عملکرد به مراتب بهتری نسبت به الگوریتم‌های تکاملی مرسوم، نظیر الگوریتم ژنتیک، از خود نشان می‌دهد، با این حال، الگوریتم کرم شب تاب، نه تنها کارایی و عملکرد بهتری، در رسیدن به جواب بهینه توابع مختلف از خود نشان می‌دهد، بلکه نرخ موفقیت (Success Rate) بالاتری نسبت به الگوریتم ژنتیک و الگوریتم PSO دارد. نتایج ارائه شده، نشان دهنده این احتمال است که الگوریتم کرم شب تاب، الگوریتم قدرتمندی در حل مسائل بهینه‌سازی مختلف، حتی مسائل NP-Hard باشد.

الگوریتم کرم شب تاب استاندارد (پایه)، الگوریتم بسیار کارآمد و مؤثری در همگرایی به جواب بهینه توابع مختلف محسوب می‌شود. با این حال، در آزمایشات و شبیه‌سازی‌های انجام شده، بعضا مشاهده شده است که حتی با نزدیک شدن کرم‌های شب تاب به ناحیه حاوی جواب بهینه (همگرایی جمعیت به جواب بهینه)، تغییرات زیادی در جواب‌های تولید شده توسط الگوریتم (جهت حل مسأله بهینه‌سازی مورد نظر) مشاهده می‌شود. یک دلیل ممکن برای چنین پدیده‌ای، عامل تصادفی‌سازی در فرمول‌بندی الگوریتم کرم شب تاب است. یک راه حل ممکن برای رفع چنین نقیصه‌ای و بهبود عملکرد الگوریتم کرم شب تاب و از همه مهم‌تر افزایش کیفیت جواب‌های تولید شده، کاهش تأثیر عامل تصادفی‌سازی به صورت تدریجی است. همچنین، با متغیر کردن پارامتر تصادفی‌سازی αα و تدریجی کردن کاهش این پارامتر (هنگام نزدیک‌تر شدن هر چه بیشتر کرم‌های شب تاب به بهینه سراسری)، این امکان برای الگوریتم فراهم می‌شود تا بتواند همگرایی به بهینه سراسری را بهبود بخشیده و جواب‌های بهینه با کیفیتی تولید کند. شایان توجه است که با ایجاد تغییرات اندکی در الگوریتم کرم شب تاب، می‌توان قابلیت حل کردن «توابع چندهدفه» (Multi-Objective Functions) را به این الگوریتم اضافه کرد. همچنین، ایجاد تغییرات مناسب در الگوریتم کرم شب تاب و استفاده از آن در مسائل «بهینه‌سازی ترکیبیاتی» (Combinatorial Optimization)، می‌تواند یکی از حوزه‌های ارتقاء الگوریتم کرم شب تاب قلمداد شود. آموزش تئوری و عملی الگوریتم ژنتیک (Genetic Algorithm) یا GA، به طور قطع شناخته شده ترین روش بهینه سازی هوشمند و الگوریتم تکاملی است که کاربردهای فراوانی در رشته های مختلف علمی و مهندسی دارد. اهمیت این الگوریتم در محاسبات تکاملی و هوش محاسباتی به قدری است که اولین کلمه ای که پس از عبارت الگوریتم تکاملی به ذهن می رسد، الگوریتم ژنتیک است. بسیاری از افراد، سایر روش های بهینه سازی هوشمند را نسخه های تغییر یافته ای از الگوریتم ژنتیک می شناسند و قائل به اصالت وجود و ماهیت سایر الگوریتم ها نیستند. این ابزار محاسباتی، در اوایل دهه ۱۹۷۰ از دل نتایجی پدید آمد که از تلاش های مهندسین و دانشمندان آن روزگار برای شبیه سازی فرایند تکامل صورت پذیرفته بود. مبتکر ایده الگوریتم های ژنتیک، جان هالند (John L. Holland) بود و پس از وی، یکی از شاگردهایش به نام دیوید گلدبرگ (David Goldberg)، تلاش فراوانی برای توسعه الگوریتم های ژنتیک انجام داده است. تاکنون محصولات متنوعی برای آموزش مباحث تئوری و عملی الگوریتم ژنتیک بر روی فرادرس ارائه شده اند. در این پست قصد داریم جدیدترین و کامل ترین محصول فرادرس را که مربوط به آموزش تئوری و عملی الگوریتم ژنتیک در متلب (MATLAB) است، به حضور مخاطبین محترم معرفی نماییم. این فرادرس که مشتمل بر بیش از ۶ ساعت برنامه آموزشی است، قطعا مرجعی بی نظیر برای آموزش مباحث تخصصی مرتبط با الگوریتم های ژنتیک و از همه مهم تر، پیاده سازی الگوریتم های ژنتیک در محیط متلب است.این محصول شامل همه مطالبی است که شما باید در مورد الگوریتم ژنتیک بدانید و از نظر برنامه نویسی نیز کامل ترین

محصولی است که تاکنون در مورد الگوریتم های ژنتیکی ارائه شده است. عناوین بخش های مختلف این بسته آموزشی در ادامه آمده است:

درس یکم: الگوریتم های ژنتیک در متلب - مباحث تئوری و عملی

در درس یکم، دانشجویان عزیز، با مباحث مختلف الگوریتم های ژنتیک در متلب آشنا می شوند. دروسی که در این آموزش به آن پرداخته می شود، مروری بر مبانی علم ژنتیک، معرفی اجزا و ساختار پایه الگوریتم های ژنتیک، بررسی جامع انواع روش های انتخاب والدین و... است. نقطه قوت این آموزش این است که به طور کامل به توضیح مباحث مربوطه پرداخته شده است و آموزش توسط یکی از بهترین مدرسین متخصص در این زمینه، انجام شده است.

فهرست سرفصل ها و رئوس مطالب مطرح شده در درس یکم در ادامه آمده است:

- مروری بر مبانی علم ژنتیک و منشا الهام الگوریتم های تکاملی و الگوریتم های ژنتیک
- معرفی اجزا و ساختار پایه الگوریتم های ژنتیک
- بررسی جامع انواع شرایط خاتمه در روش های بهینه سازی و الگوریتم های عددی
- بررسی جامع انواع ساختارهای ممکن برای تلفیق و انتخاب اعضای جمعیت جدید
  - روش تلفیق، مرتب سازی و حذف
  - روش سهم های از پیش تعیین شده
  - روش تلفیق و انتخاب تصادفی
  - روش اعمال همزمان تقاطع و جهش بر روی اعضای جمعیت
- بررسی جامع انواع روش های انتخاب والدین
  - انتخاب تصادفی
  - انتخاب بر اساس شایستگی
    - مفاهیم ابتدایی و شرایط توزیع احتمالی گسسته برای انجام انتخاب
    - روش های مختلف برای تعریف توزیع احتمالی، از جمله روش بولتزمن (Boltzmann methods)
    - فشار انتخاب و قواعد عملی برای تنظیم آن
    - چرخه رولت (Roulette Wheel) و شیوه عملکرد آن
    - ساده سازی مکانیزم چرخه رولت به همراه بیان نکات مهم برای پیاده سازی
  - انتخاب رقابتی (Tournament Selection)
    - بررسی تاثیر اندازه تورنومنت (Tournament) در عملکرد این عملگر انتخاب
    - فشار انتخاب در انتخاب رقابتی
- انواع اپراتورها برای مسائل مختلف
  - مسائل باینری و گسسته
    - بررسی انواع تقاطع قابل استفاده در مسائل باینری و گسسته
    - چگونگی انجام جهش در فضای باینری و گسسته
  - مسائل پیوسته (اعداد حقیقی)
    - اپراتور تقاطع حسابی برای مسائل پیوسته
    - بهبود عملگر تقاطع با افزودن پارامتر برون گرایی
    - چگونگی انجام عمل جهش در فضای پیوسته
    - جهش نرمال (گاوسی)

- تنظیم گام جهش بر اساس قانون یک پنجم (One Fifth Rule)
- بررسی مفهوم تعداد دفعات فراخوانی تابع (Number of Function Evaluations) و یا به اختصار NFE
- جمع بندی و نتیجه گیری های نهایی

سرفصل های مورد بحث در بخش عملی:

- پیاده سازی الگوریتم ژنتیک باینری
  - پیاده سازی برای حل یک مساله باینری نمونه
  - پیاده سازی انواع روش های تقاطع
    - تقاطع تک نقطه ای (Single Point Crossover)
    - تقاطع دو نقطه ای (Double Point Crossover)
    - تقاطع یکنواخت (Uniform Crossover)
    - ترکیب تصادفی سه نوع تقاطع به صورت ساده و با استفاده از چرخه رولت
  - پیاده سازی جهش برای مسائل باینری
  - پیاده سازی روش های مختلف انتخاب والد
    - انتخاب تصادفی
    - انتخاب توسط چرخه رولت (Roulette Wheel)
    - انتخاب رقابتی (Tournament Selection)
    - ایجاد رابط گرافیک کاربری برای انتخاب یکی از سه روش انتخاب در هنگام اجرای برنامه
  - افزودن محاسبه تعداد دفعات فراخوانی تابع هدف یا NFE به الگوریتم ژنتیک
- پیاده سازی الگوریتم ژنتیک پیوسته
  - پیاده سازی برای حل یک مساله پیوسته نمونه
  - پیاده سازی تقاطع حسابی (Arithmetic Crossover) و بهبود عملکرد آن
  - پیاده سازی جهش نرمال یا گاوسی
- جمع بندی و نتیجه گیری نهایی

درس دوم: حل مساله مکان یابی هاب با استفاده از الگوریتم ژنتیک

در درس دوم، آشنایی دانشجویان عزیز، با مباحث مختلف حل مساله مکان یابی هاب (Hub Location Problem) با استفاده از الگوریتم ژنتیک صورت می گیرد. دروسی که در این آموزش نام می بریم، مساله مکان یابی هاب، ارائه مدل ریاضی مساله، پیاده سازی مساله مکان یابی هاب و... است. توضیح کامل مباحث مربوطه و آموزش توسط یکی از بهترین مدرسین متخصص در این زمینه از نقاط قوت این آموزش به شمار می رود.

فهرست سرفصل ها و رئوس مطالب مطرح شده در درس دوم در ادامه آمده است:

- مروری بر مساله مکان یابی هاب (Hub Location Allocation) و طراحی شبکه های حسگر بی سیم (Wireless Sensor Network Design)
- ارائه مدل ریاضی مساله
- بحث بر روی روش های کلی برای حل مسائل بهینه سازی
- پیاده سازی مساله مکان یابی هاب در قالب یک مساله بهینه سازی باینری
- حل مساله توسط الگوریتم ژنتیک باینری
- ترسیم شکل مربوط به راه حل مساله به صورت آنلاین (همگام با اجرای برنامه)

- جمع بندی و نتیجه گیری نهایی

درس سوم: حل مساله حمل و نقل با استفاده از الگوریتم ژنتیک

در درس سوم، دانشجویان عزیز، با مباحث مختلف حل مساله حمل و نقل با استفاده از الگوریتم ژنتیک آشنا می شوند. دروسی که در این آموزش به آن پرداخته می شود، مساله حمل و نقل، ارائه مدل ریاضی مساله، پیاده سازی مساله حمل و نقل به صورت یک مساله پیوسته حقیقی و... است. نقطه قوت این آموزش این است که به طور کامل به توضیح مباحث مربوطه پرداخته شده است و آموزش توسط یکی از بهترین مدرسین متخصص در این زمینه، انجام شده است.

فهرست سرفصل ها و رئوس مطالب مطرح شده در درس سوم در ادامه آمده است:

- مروری بر مساله حمل و نقل (Transportation Problem)
- ارائه مدل ریاضی مساله
- بحث بر روی روش های کلی برای حل مسائل بهینه سازی مقید
- چگونگی تعریف تخطی برای انواع قیدهای مساوی و نامساوی
- بررسی انواع تابع جریمه
  - تابع جریمه جمع شونده (Additive Penalty)
  - تابع جریمه ضرب شونده (Multiplicative Penalty)
  - تابع جریمه ترکیبی (Mixed Penalty)
- چگونگی تغییر در ساختار پاسخ، برای محدود کردن جستجو به فضای شدنی (Feasible)
- ارائه مکانیزم کلی برای حل مسائل بهینه سازی با استفاده از الگوریتم های هوشمند
- ایجاد مکانیزمی برای ذخیره اطلاعات مساله در فایل و فراخوانی آن
- پیاده سازی مساله حمل و نقل به صورت یک مساله پیوسته حقیقی
- حل مساله توسط الگوریتم ژنتیک پیوسته
- ایجاد تغییرات در مساله حمل و نقل و تبدیل آن به یک مساله ترکیبی پیوسته و باینری
- نحوه تعریف کروموزم برای مسائلی که چندین نوع متغیر از انواع مختلف دارند
- بررسی نسخه های دیگری از مساله حمل و نقل و حل آن ها توسط الگوریتم ژنتیک
- جمع بندی و نتیجه گیری نهایی

درس چهارم: حل مساله تخصیص درجه دو یا QAP با استفاده از الگوریتم ژنتیک

در درس چهارم قصد داریم فیلم آموزشی جدیدی را که مربوط به حل مساله تخصیص درجه دو (Quadratic Assignment Problem) و یا به اختصار QAP است، معرفی نماییم. این مساله یکی از مسائل پایه ای در بهینه سازی ترکیبی (Combinatorial Optimization) و تحقیق در عملیات است. از نظر دسته بندی کلی، این مساله در میان مسائل مکان یابی تاسیسات قرار می گیرد و کاربردهای فراوانی در رشته های مختلف علمی و فنی دارد.

فهرست سرفصل ها و رئوس مطالب مطرح شده در درس چهارم در ادامه آمده است:

- مروری بر مساله تخصیص درجه دو (Quadratic Assignment Problem) یا QAP
- ارائه مدل ریاضی مساله
- مروری بر روش های برخورد با مسائل مقید با قید ساختاری
- بیان مساله QAP به صورت یک مساله جایگشتی
- نحوه ایجاد راه حل های جایگشتی و تبدیل آن ها به راه حلی برای مساله QAP
- روش انجام تقاطع برای کروموزوم های جایگشتی

- انواع عملگرهای جهش برای کروموزوم های جایگشتی
  - تعویض دو عضو یا Swap
  - معکوس سازی بخش میان دو عضو یا Reversion
  - حذف و جایگذاری یا Insertion
- پیاده سازی مساله QAP به صورت یک مساله جایگشتی
- انجام تغییرات لازم بر روی الگوریتم ژنتیک برای حل مسائل جایگشتی
- حل مساله QAP با استفاده از الگوریتم ژنتیک
- ترسیم شکل مربوط به راه حل مساله به صورت آنلاین (همگام با اجرای برنامه)
- بررسی تاثیر ضرایب وزنی بر پاسخ مساله
- استراتژی های طراحی و مدیریتی در یک شبکه تخصیص
- جمع بندی و نتیجه گیری های نهایی

درس پنجم: حل مساله کوله پشتی با الگوریتم ژنتیک

مساله کوله پشتی (Knapsack Problem) یکی از مسائل معروف در ریاضیات کاربردی و تحقیق در عملیات است، که به نام Backpack Problem نیز شناخته می شود. این مساله دارای نسخه های مختلفی است و حالات پیچیده تر آن، در حل مسائل روزمره و صنعتی، کاربردهای فراوانی دارد. گذشته از اهمیت عملی این مساله، نسخه های استانداردی نیز برای این مساله تعریف شده اند که برای ارزیابی عملکرد الگوریتم های بهینه سازی، مورد استفاده قرار می گیرد. در این فیلم آموزشی با استفاده از الگوریتم ژنتیک به حل مساله کوله پشتی باینری پرداخته شده است.

فهرست سرفصل ها و رئوس مطالب مطرح شده در درس پنجم در ادامه آمده است:

- مروری بر مساله کوله پشتی (Knapsack Problem)
- ارائه مدل ریاضی مساله کوله پشتی باینری
- پیاده سازی مساله کوله پشتی در قالب یک مساله بهینه سازی باینری
- حل مساله توسط الگوریتم ژنتیک باینری
- جمع بندی و نتیجه گیری های نهایی

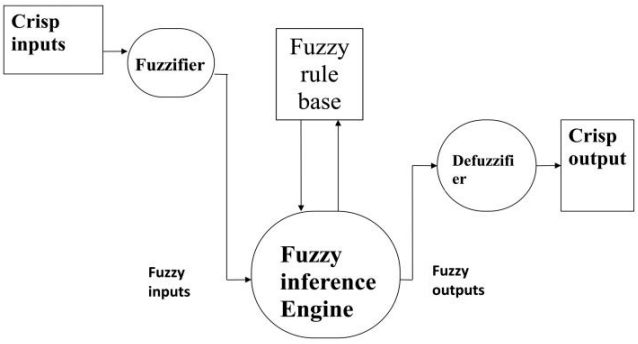
درس ششم: شناسایی سیستم و مدل سازی سیستم های غیر خطی با استفاده از الگوریتم ژنتیک

به جرات می توان گفت که یکی از مسائل مهم در تمامی رشته های علمی و مهندسی، مساله مدل سازی است و در بسیاری از رشته ها، این مساله مهم ترین مساله ای است که در آن حوزه تعریف می شود. تبدیل داده های برگرفته از مشاهدات، به مدلی ریاضی که بتواند سیستم مورد مطالعه را توصیف نماید، گاهی اوقات بزرگ ترین هدفی است که یک محقق دارد. اگر هدف از مدل سازی، ایجاد یک مدل دینامیکی، به صورت مجموعه ای از معادلات دیفرانسیلی باشد، مساله مدل سازی غالبا با نام شناسایی سیستم (System Identification) شناخته می شود. شناسایی سیستم مختص هیچ رشته خاصی نیست، اما اغلب مهندسین کنترل و ریاضیدان ها به کار توسعه الگوریتم ها و روش های شناسایی مبادرت می ورزند.

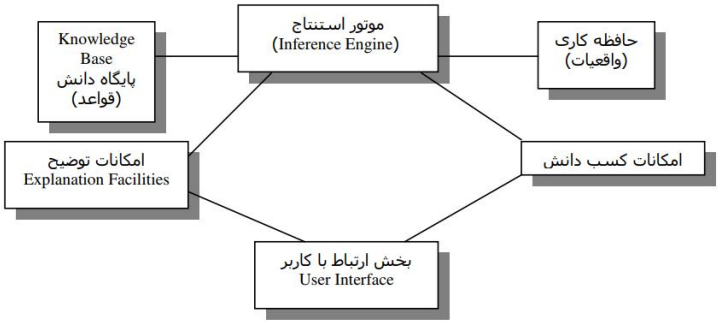
روش های مختلفی برای حل مساله شناسایی سیستم و مدل سازی سیستم های دینامیکی به وجود آمده اند که در حوزه های مختلفی نظیر: مهندسی برق، مهندسی پزشکی، مهندسی مکانیک، مهندسی شیمی و فرایند، اقتصاد، هواشناسی و کشاورزی کاربرد دارند.فهرست سرفصل ها و رئوس مطالب مطرح شده در درس ششم در ادامه آمده است:

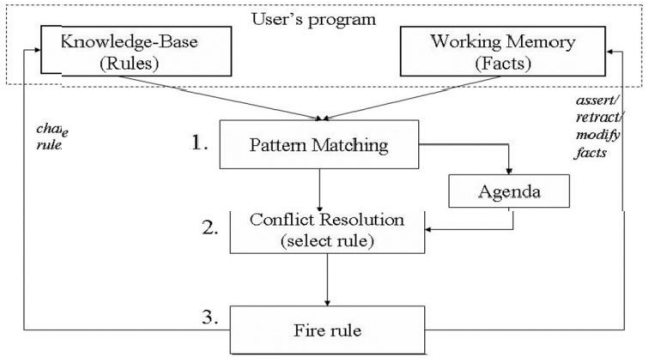
- مروری بر مفاهیم مدل سازی و شناسایی سیستم
- تعریف تابع خطا و تبدیل مساله شناسایی سیستم به مساله بهینه سازی
- معرفی سیستم غیر خطی شکار و شکارچی (Predator - Prey)

- بررسی خواص دینامیکی مدل شکار و شکارچی
- مقدمه ای بر شبیه سازی سیستم های دینامیکی
- تقریب اویلر برای حل عددی معادلات دیفرانسیل و شبیه سازی سیستم های دینامیکی
- پیاده سازی تقریب اویلر در محیط متلب
- ایجاد یک مدل فرضی و شبیه سازی آن برای تعریف یک مساله مدل سازی
- پیاده سازی تابع خطای مدل سازی
- اتصال تابع خطای تعریف شده به الگوریتم ژنتیک
- حل مساله شناسایی سیستم غیر خطی با استفاده از الگوریتم ژنتیک
- ایجاد مکانیزمی برای نمایش لحظه به لحظه نتایج مدل سازی به صورت نمودارهای گرافیکی
- جمع بندی و نتیجه گیری نهایی

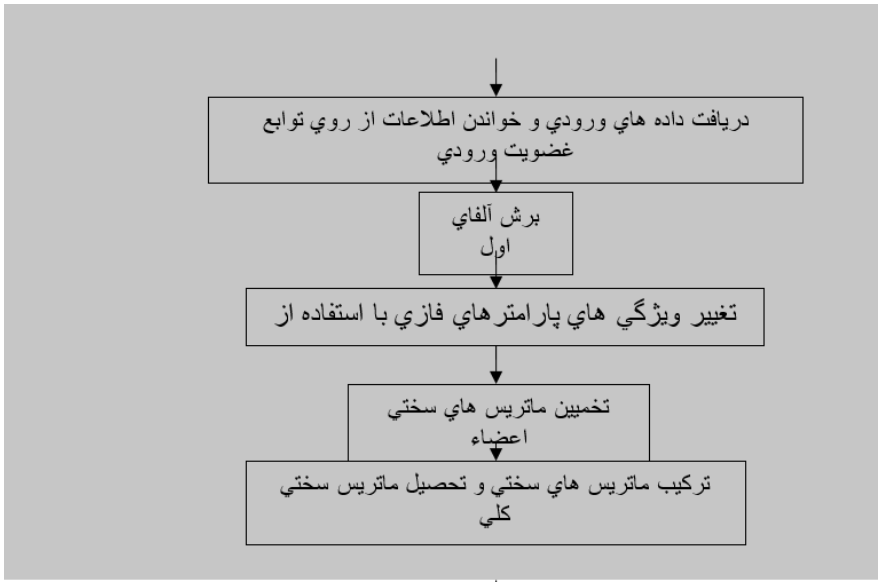


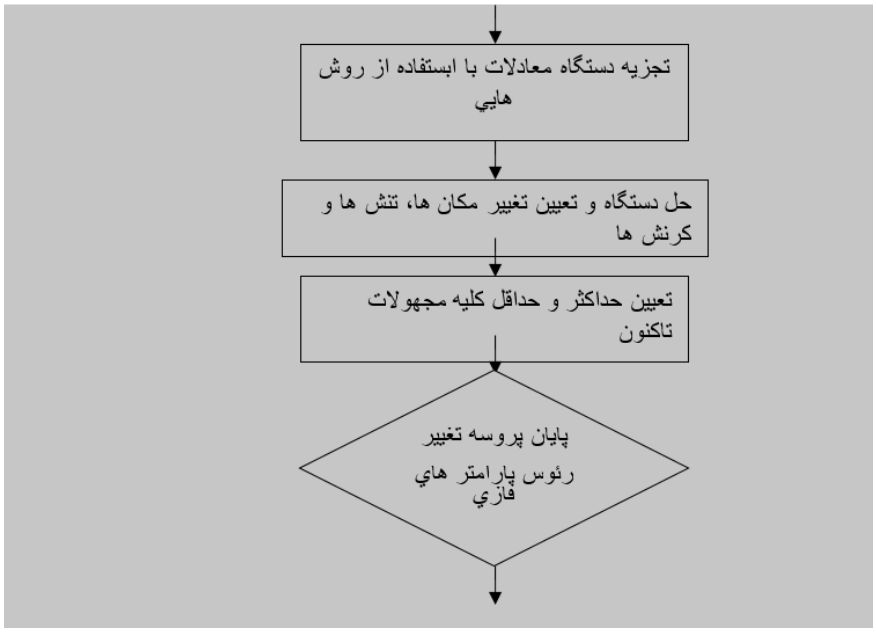
Structure of a fuzzy inference system.

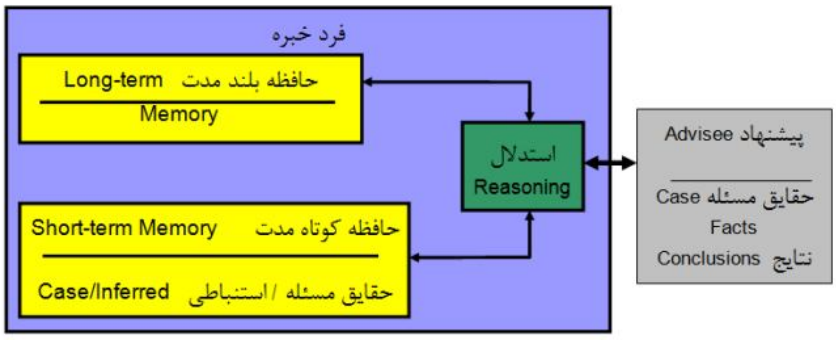




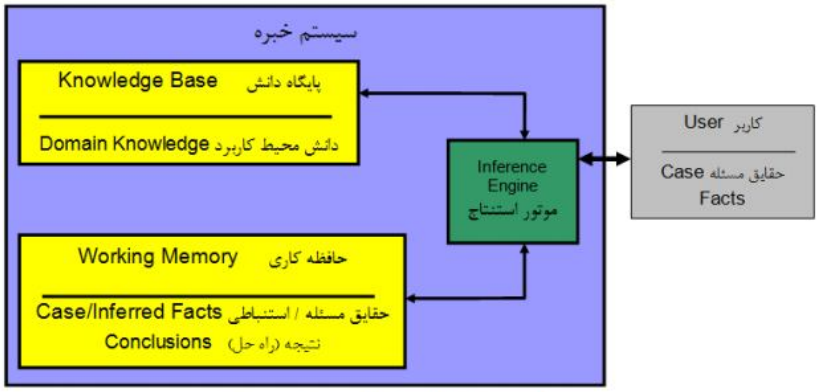
چرخه استنتاج در CLIPS



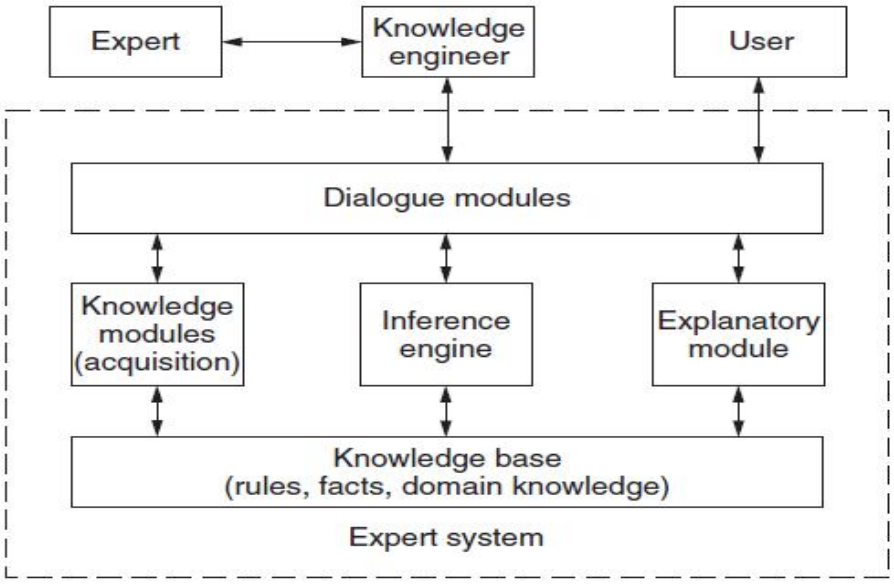




نمایش روش استدلال در یک فرد خبره



نمایش روش استدلال در یک سیستم خبره



Expert system.

ساختار ANFIS

ANFIS(adaptive network-based fuzzy inference system) شبکه تطبیق پذیر و قابل آموزشی است که به لحاظ عملکرد کاملا مشابه سیستم استنتاج فازی است.برای سادگی کار فرض می کنیم که سیستم فازی ما دو ورودی x و y دارد و خروجی آن z است. حال اگر قوانین به صورت زیر باشند :

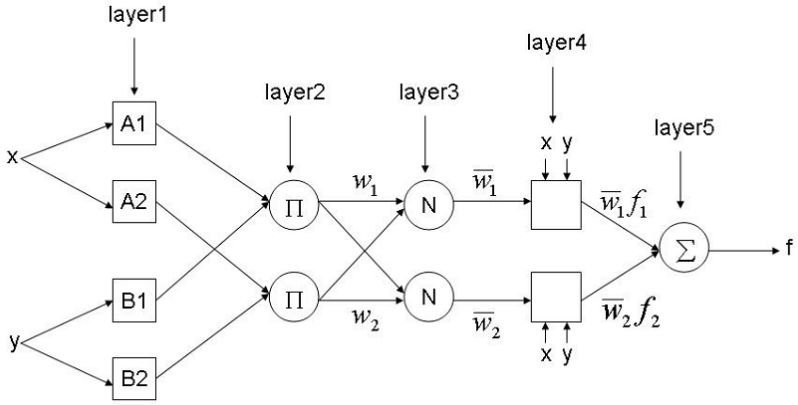
$$Rule1: if\ x\ is\ A_1\ and\ y\ is\ B_1\ then\ f_1 = p_1x + q_1y + r_1$$

$$Rule2: if\ x\ is\ A_2\ and\ y\ is\ B_2\ then\ f_2 = p_2x + q_2y + r_2$$

و اگر برای غیر فازی ساز از غیر فازی ساز میانگین مراکز استفاده کنیم خروجی به صورت زیر خواهد بود:

$$f = \frac{w_1 f_1 + w_2 f_2}{w_1 + w_2} = \overline{w}_1 f_1 + \overline{w}_2 f_2 \quad st \quad \overline{w}_1 = \frac{w_1}{w_1 + w_2},\ \overline{w}_2 = \frac{w_2}{w_1 + w_2}$$

ساختار معادل ANFIS به صورت زیر خواهد بود:



لایه ۱: در این لایه ورودی ها از توابع عضویت عبور(membership functions) می کنند.

$$O_{1,i} = \mu A_i(x), \qquad for\ i = 1,2$$

$$O_{1,i} = \mu B_i(x), \qquad for\ i = 3,4$$

توابع عضویت هر تابع پارامتری مناسبی می تواند باشد که در اکثر موارد توابع گاوسین انتخاب می شوند. مثل تابع زنگی شکل عمومی:

$$\mu A(x) = \frac{1}{1 + \left| \frac{x - c_i}{a_i} \right|^{2b_i}}$$

که a و b و c مجموعه پارامترها هستند. پارامتر های این لایه به پارامتر های اولیه (premise parameters) معروف هستند.

لایه ۲: خروجی این لایه ضرب سیگنال های ورودی است که در واقع معادل قسمت اگر قوانین هستند.

$$O_{2,i} = w_i = \mu A_i(x)\mu B_i(y), \quad i = 1,2$$

لایه ۳: خروجی این لایه نرمالیزه شده لایه قبلی است:

$$O_{3,i} = \overline{w}_i = \frac{w_i}{w_1 + w_2}, \quad i = 1,2$$

لایه ۴:

$$O_{4,i} = \overline{w}_i f_i = \overline{w}_i (p_i x + q_i y + r_i)$$

لایه ۵: خروجی این لایه خروجی کلی سیستم است:

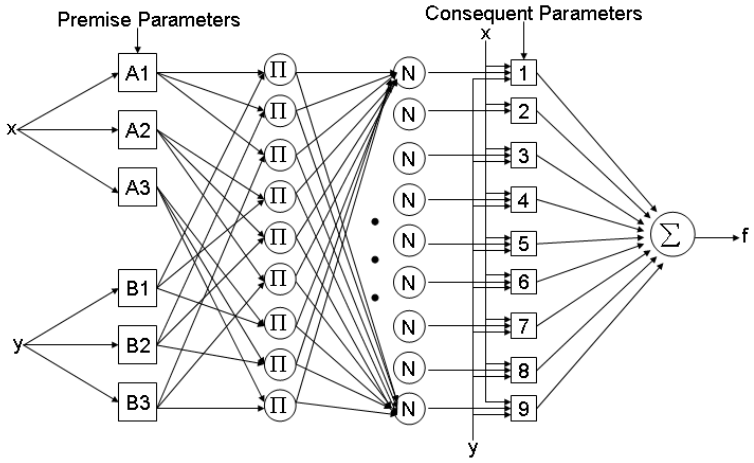
$$O_{5,i} = \sum_i \overline{w}_i f_i = \frac{\sum_i w_i f_i}{\sum_i w_i}$$

اکنون یک شبکه تولید شده است که معادل سیستم استنتاج فازی سوگنو است.

حال قرار است روش های آموزش چنین شبکه ای بررسی شود.

برای این کار ابتدا در لایه ۱ تمام قوانین موجود را تشکیل می دهیم.به طور مثال اگر ۲ ورودی داشته باشیم که هر کدام ۳ تابع عضویت داشته باشد ۹ قانون باید تشکیل دهیم.

که به صورت زیر خواهد بود.



روشهای مختلف آموزش ANFIS

۱- Gradient Descent

۲- ترکیبی (hybrid)

۳- Levenberg-Marquardt

روش ترکیبی(hybrid)

در این روش از ترکیب روش گرادیان نزولی و حداقل مربعات خطا (LSE) استفاده می شود.

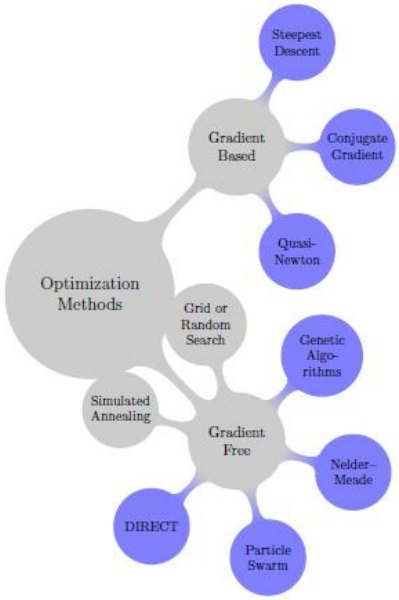
روش دیگری نیز برای محاسبه پارامتر ها وجود دارد که یک روش بازگشتی است (RLSE) این روش به دو دلیل به وجود آمده است.

۱- گاهی ممکن است که ماتریس معکوس پذیز نباشد.

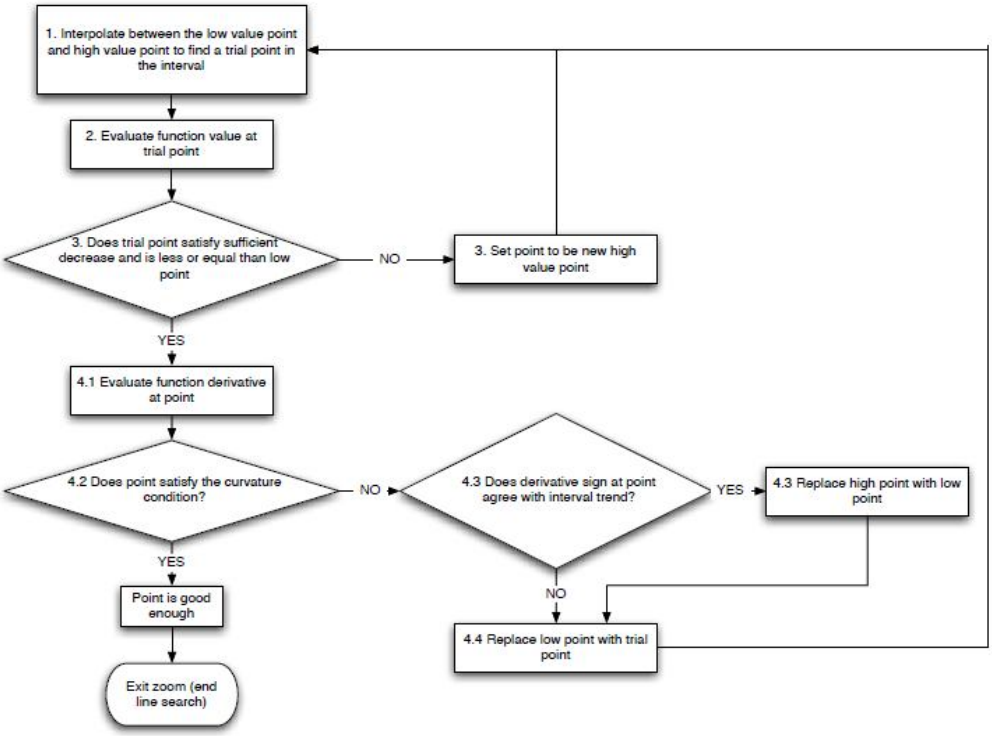
۲- فرض کنید را حساب کرده ایم اگر یک جفت داده ورودی خروجی جدید به سیستم اضافه شود در روش قبل دوباره باید کل پارامتر ها محاسبه شوند ولی در این روش فقط مقدار پارامتر جدید به دست می آید.

طریقه آموزش ANFIS با روش ترکیبی

در هر تکرار feedforward می رویم تا وقتی که ماتریس A که در روش LSE گفته شد به دست آید. خروجی ها را هم که داریم . سپس توسط روش ترکیبی پارامتر ها را به دست می آوریم. لازم به ذکر است که همه داده های آموزشی باید اعمال شوند و همجنین پارامتر های اولیه(premise patameters) ثابت نگه داشته می شوند. سپس پارامترهای تالی (conclusion parameters) ثابت نگه داشته می شوند و پارامترهای اولیه توسط گرادیان نزولی تنظیم می شوند.



Optimization methods for nonlinear problems



"Zoom" function in the line search algorithm satisfying the strong Wolfe conditions

Algorithm 2 Line search algorithm

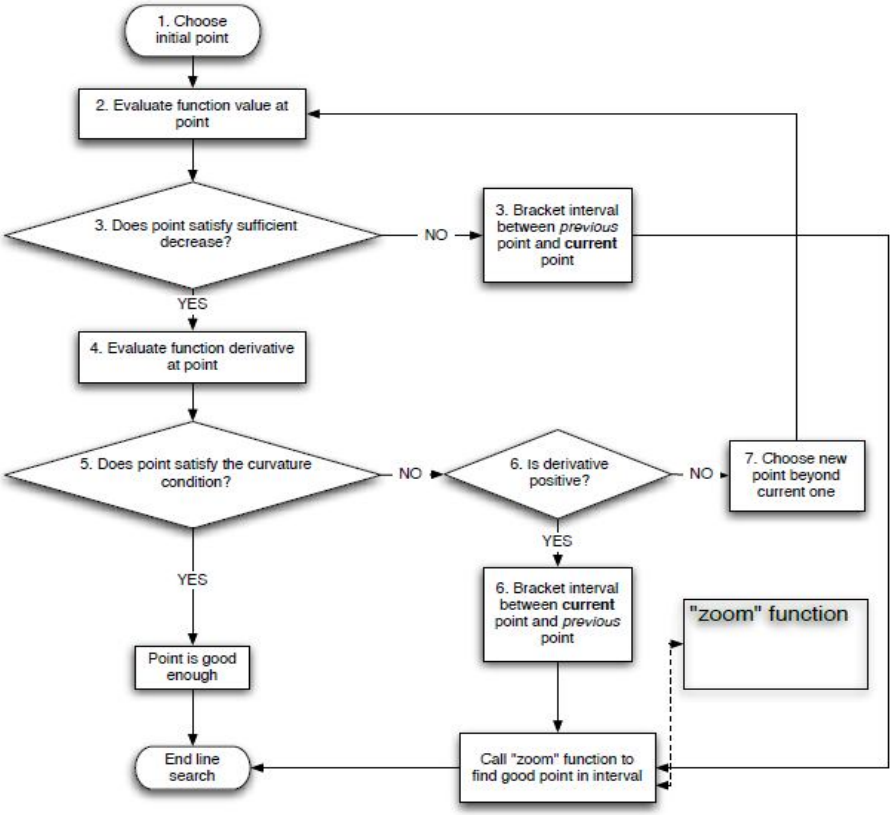
Input: $\alpha_1 > 0$ and $\alpha_{\max}$
Output: α_*

$\alpha_0 = 0$
$i \leftarrow 1$
repeat
  Evaluate $\phi(\alpha_i)$
  if $[\phi(\alpha_i) > \phi(0) + \mu_1 \alpha_i \phi'(0)]$ or $[\phi(\alpha_i) > \phi(\alpha_{i-1})$ and $i > 1]$ **then**
    $\alpha_* \leftarrow \texttt{zoom}(\alpha_{i-1}, \alpha_i)$
    return α_*
  end if
  Evaluate $\phi'(\alpha_i)$
  if $|\phi'(\alpha_i)| \leq -\mu_2 \phi'(0)$ **then**
    return $\alpha_* \leftarrow \alpha_i$
  else if $\phi'(\alpha_i) \geq 0$ **then**
    $\alpha_* \leftarrow \texttt{zoom}(\alpha_i, \alpha_{i-1})$
    return α_*
  else
    Choose α_{i+1} such that $\alpha_i < \alpha_{i+1} < \alpha_{\max}$
  end if
  $i \leftarrow i + 1$
until

Algorithm 3 Zoom function for the line search algorithm

Input: $\alpha_{\text{low}}, \alpha_{\text{high}}$
Output: α_*

$j \leftarrow 0$
repeat
  Find a trial point α_j between α_{low} and α_{high}
  Evaluate $\phi(\alpha_j)$
  if $\phi(\alpha_j) > \phi(0) + \mu_1 \alpha_j \phi'(0)$ or $\phi(\alpha_j) > \phi(\alpha_{\text{low}})$ **then**
    $\alpha_{\text{high}} \leftarrow \alpha_j$
  else
    Evaluate $\phi'(\alpha_j)$
    if $|\phi'(\alpha_j)| \leq -\mu_2 \phi'(0)$ **then**
      $\alpha_* = \alpha_j$
      return α_*
    else if $\phi'(\alpha_j)(\alpha_{\text{high}} - \alpha_{\text{low}}) \geq 0$ **then**
      $\alpha_{\text{high}} \leftarrow \alpha_{\text{low}}$
    end if
    $\alpha_{\text{low}} \leftarrow \alpha_j$
  end if
  $j \leftarrow j + 1$
until



Line search algorithm satisfying the strong Wolfe conditions

Algorithm 4 General algorithm for smooth functions

Input: Initial guess, x_0
Output: Optimum, x^*
  $k \leftarrow 0$
  while Not converged **do**
    Compute a search direction p_k
    Find a step length α_k, such that $f(x_k + \alpha_k p_k) < f(x_k)$ (the curvature condition may also be included)
    Update the design variables: $x_{k+1} \leftarrow x_k + \alpha_k p_k$
    $k \leftarrow k + 1$
  end while

Algorithm 5 Steepest descent

Input: Initial guess, x_0, convergence tolerances, $\varepsilon_g, \varepsilon_a$ and ε_r.
Output: Optimum, x^*
  $k \leftarrow 0$
  repeat
    Compute the gradient of the objective function, $g(x_k) \equiv \nabla f(x_k)$
    Compute the normalized search direction, $p_k \leftarrow -g(x_k)/\|g(x_k)\|$
    Perform line search to find step length α_k
    Update the current point, $x_{k+1} \leftarrow x_k + \alpha_k p_k$
    $k \leftarrow k + 1$
  until $|f(x_k) - f(x_{k-1})| \leq \varepsilon_a + \varepsilon_r |f(x_{k-1})|$ and $\|g(x_{k-1})\| \leq \varepsilon_g$

Algorithm 6 Nonlinear conjugate gradient method

Input: Initial guess, x_0, convergence tolerances, $\varepsilon_g, \varepsilon_a$ and ε_r.
Output: Optimum, x^*
  $k \leftarrow 0$
  repeat
    Compute the gradient of the objective function, $g(x_k)$
    if k=0 **then**
      Compute the normalized steepest descent direction, $p_k \leftarrow -g(x_k)/\|g(x_k)\|$
    else
      Compute $\beta \leftarrow \frac{g_k^T g_k}{g_{k-1}^T g_{k-1}}$
      Compute the conjugate gradient direction $p_k = -g_k/\|g(x_k)\| + \beta_k p_{k-1}$
    end if
    Perform line search to find step length α_k
    Update the current point, $x_{k+1} \leftarrow x_k + \alpha_k p_k$
    $k \leftarrow k + 1$
  until $|f(x_k) - f(x_{k-1})| \leq \varepsilon_a + \varepsilon_r |f(x_{k-1})|$ and $\|g(x_{k-1})\| \leq \varepsilon_g$

Algorithm 7 Modified Newton's method

Input: Initial guess, x_0, convergence tolerances, $\varepsilon_g, \varepsilon_a$ and ε_r.
Output: Optimum, x^*

$k \leftarrow 0$
repeat
  Compute the gradient of the objective function, $g(x_k)$
  Compute the Hessian of the objective function, $H(x_k)$
  Compute the search direction, $p_k = -H^{-1}g_k$
  Perform line search to find step length α_k, starting with $\alpha = 1$
  Update the current point, $x_{k+1} \leftarrow x_k + \alpha_k p_k$
  $k \leftarrow k + 1$
until $|f(x_k) - f(x_{k-1})| \leq \varepsilon_a + \varepsilon_r |f(x_{k-1})|$ and $\|g(x_{k-1})\| \leq \varepsilon_g$

Algorithm 8 Quasi-Newton method with DFP update

Input: Initial guess, x_0, convergence tolerances, $\varepsilon_g, \varepsilon_a$ and ε_r.
Output: Optimum, x^*

$k \leftarrow 0$
$V_0 \leftarrow I$
repeat
  Compute the gradient of the objective function, $g(x_k)$
  Compute the search direction, $p_k \leftarrow -V_k g_k$
  Perform line search to find step length α_k, starting with $\alpha \leftarrow 1$
  Update the current point, $x_{k+1} \leftarrow x_k + \alpha_k p_k$
  Set the step length, $s_k \leftarrow \alpha_k p_k$
  Compute the change in the gradient, $y_k \leftarrow g_{k+1} - g_k$
  $A_k \leftarrow \frac{V_k y_k y_k^T V_k}{y_k^T V_k y_k}$
  $B_k \leftarrow \frac{s_k s_k^T}{s_k^T y_k}$
  Compute the updated approximation to the inverse of the Hessian, $V_{k+1} \leftarrow V_k - A_k + B_k$
until $|f(x_k) - f(x_{k-1})| \leq \varepsilon_a + \varepsilon_r |f(x_{k-1})|$ and $\|g(x_{k-1})\| \leq \varepsilon_g$

Davidon{Fletcher{Powell (DFP) Method

Algorithm 9 Trust region algorithm

Input: Initial guess x_0, convergence tolerances, $\varepsilon_g, \varepsilon_a$ and ε_r, initial size of the trust region, h_0
Output: Optimum, x^*

$k \leftarrow 0$
repeat
Compute the Hessian of the objective function $H(x_k)$, and solve the quadratic subproblem:

$$\begin{aligned} \text{minimize} \quad & q(s_k) = f(x_k) + g(x_k)^T s_k + \frac{1}{2} s_k^T H(x_k) s_k \\ \text{w.r.t.} \quad & s_k \\ \text{s.t.} \quad & -h_k \leq s_k \leq h_k, \quad i = 1, \ldots, n \end{aligned}$$

Evaluate $f(x_k + s_k)$ and compute the ratio that measures the accuracy of the quadratic model,

$$r_k \leftarrow \frac{f(x_k) - f(x_k + s_k)}{f(x_k) - q(s_k)} = \frac{\Delta f}{\Delta q}$$

if $r_k < 0.25$ **then**
$h_{k+1} \leftarrow \frac{\|s_k\|}{4}$ {Model is not good; shrink the trust region}
else if $r_k > 0.75$ and $h_k = \|s_k\|$ **then**
$h_{k+1} \leftarrow 2h_k$ {Model is good and new point on edge; expand trust region}
else
$h_{k+1} \leftarrow h_k$ {New point with trust region and the model is reasonable; keep trust region the same size}
end if
if $r_k \leq 0$ **then**
$x_{k+1} \leftarrow x_k$ {Keep trust region centered about the same point}
else
$x_{k+1} \leftarrow x_k + s_k$ {Move center of trust region to new point}
end if
$k \leftarrow k + 1$
until $|f(x_k) - f(x_{k-1})| \leq \varepsilon_a + \varepsilon_r |f(x_{k-1})|$ and $\|g(x_{k-1})\| \leq \varepsilon_g$

Algorithm 10 General algorithm

Check termination conditions. if x_k satisfies the optimality conditions, the algorithm terminates successfully.
Minimize the penalty function. With x_k as the starting point, execute an algorithm to solve the unconstrained subproblem

$$\begin{aligned} \text{minimize} \quad & \pi(x, \rho_k) \\ \text{w.r.t.} \quad & x \end{aligned}$$

and let the solution of this subproblem be x_{k+1}.
Increase the penalty parameter. Set ρ_{k+1} to a larger value than ρ_k, set $k = k + 1$ and return to 1.

Algorithm 11 SQP algorithm

Input: Initial guess (x_0, λ_0), parameters $0 < \eta < 0.5$
Output: Optimum, x^*
  $k \leftarrow 0$
  Initialize the Hessian estimate, $B_0 \leftarrow I$
  repeat
    Compute p_k and $p_{\hat{\lambda}}$ by solving (5.55), with B_k in place of W_k
    Choose μ_k such that p_k is a descent direction for ϕ at x_k
    $\alpha_k \leftarrow 1$
    while $\phi(x_k + \alpha_k p_k, \mu_k) > \phi(x_k, \mu_k) + \eta \alpha_k D\left[\phi(x_k, p_k)\right]$ **do**
      $\alpha_k \leftarrow \tau_\alpha \alpha_k$ for some $0 < \tau_\alpha < 1$
    end while
    $x_{k+1} \leftarrow x_k + \alpha_k p_k$
    $\hat{\lambda}_{k+1} \leftarrow \hat{\lambda}_k + p_{\hat{\lambda}}$
    Evaluate f_{k+1}, g_{k+1}, c_{k+1} and A_{k+1}
    $s_k \leftarrow \alpha_k p_k$, $y_k \leftarrow \nabla_x \mathcal{L}(x_{k+1}, \lambda_{k+1}) - \nabla_x \mathcal{L}(x_k, \lambda_{k+1})$
    Obtain B_{k+1} by using a quasi-Newton update to B_k
    $k \leftarrow k + 1$
  until Convergence

Algorithm 12 Nelder–Mead Algorithm

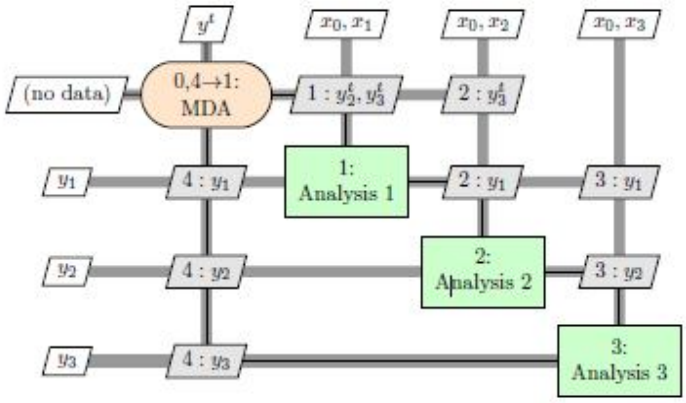
Input: Initial guess, x_0
Output: Optimum, x^*
  $k \leftarrow 0$
  Create a simplex with edge length c
  repeat
    Identify the highest (x_w: worst), second highest (x_l, lousy) and lowest (x_b: best) value points with function values f_w, f_l, and f_b, respectively
    Evaluate x_a, the average of the point in simplex excluding x_w
    Perform *reflection* to obtain x_r, evaluate f_r
    if $f_r < f_b$ **then**
      Perform *expansion* to obtain x_e, evaluate f_e.
      if $f_e < f_b$ **then**
        $x_w \leftarrow x_e$, $f_w \leftarrow f_e$ (accept expansion)
      else
        $x_w \leftarrow x_r$, $f_w \leftarrow f_r$ (accept reflection)
      end if
    else if $f_r \leq f_l$ **then**
      $x_w \leftarrow x_r$, $f_w \leftarrow f_r$ (accept reflected point)
    else
      if $f_r > f_w$ **then**
        Perform an *inside contraction* and evaluate f_c
        if $f_c < f_w$ **then**
          $x_w \leftarrow x_c$ (accept contraction)
        else
          Shrink the simplex
        end if
      else
        Perform an *outside contraction* and evaluate f_c
        if $f_c \leq f_r$ **then**
          $x_w \leftarrow x_c$ (accept contraction)
        else
          Shrink the simplex
        end if
      end if
    end if
    $k \leftarrow k + 1$
  until $(f_w - f_b) < (\varepsilon_1 + \varepsilon_2 |f_b|)$

Algorithm 13 DIRECT Algorithm

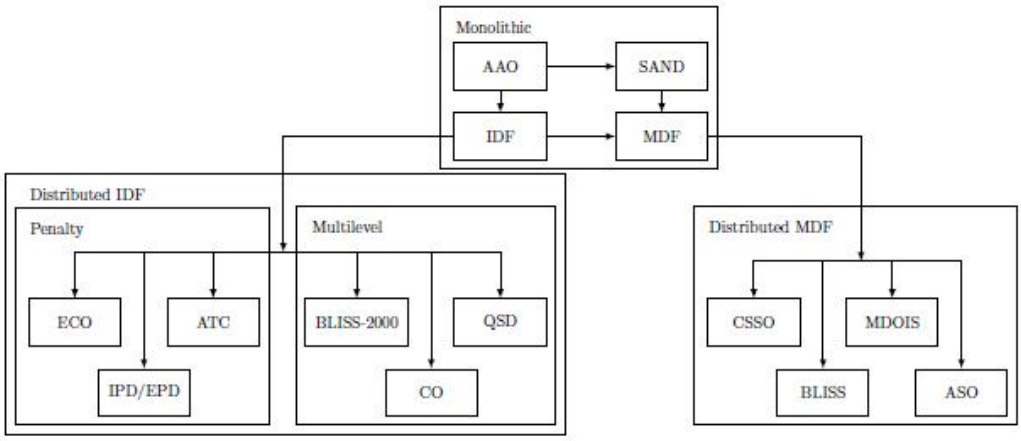
Input: Initial guess, x_0
Output: Optimum, x^*
$k \leftarrow 0$
repeat
Normalize the search space to be the unit hypercube. Let c_1 be the center point of this hypercube and evaluate $f(c_1)$.
Identify the set S of potentially optimal rectangles/cubes, that is all those rectangles defining the bottom of the convex hull of a scatter plot of rectangle diameter versus $f(c_i)$ for all rectangle centers c_i
for all Rectangles $r \in S$ **do**
Identify the set I of dimensions with the maximum side length
Set δ equal one third of this maximum side length
for all $i \in I$ **do**
Evaluate the rectangle/cube at the point $c_r \pm \delta e_i$ for all $i \in I$, where c_r is the center of the rectangle r, and e_i is the i^{th} unit vector
end for
Divide the rectangle r into thirds along the dimensions in I, starting with the dimension with the lowest value of $f(c \pm \delta e_i)$ and continuing to the dimension with the highest $f(c \pm \delta e_i)$.
end for
until Converged

Algorithm 14 Block Gauss–Seidel multidisciplinary analysis algorithm

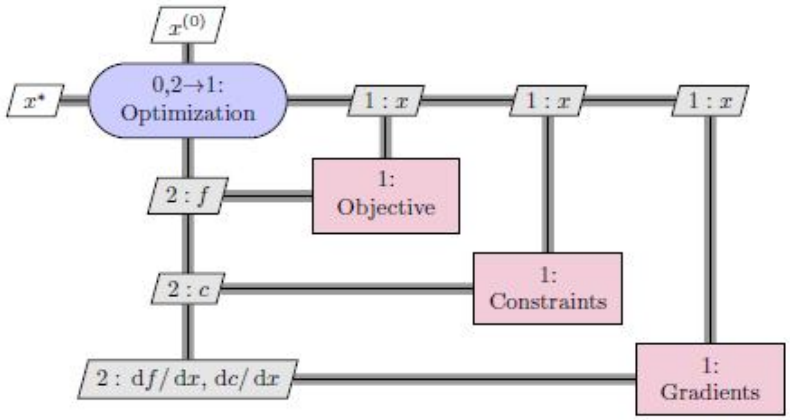
Input: Design variables x
Output: Coupling variables, y
0: Initiate MDA iteration loop
repeat
1: Evaluate Analysis 1 and update $y_1(y_2, y_3)$
2: Evaluate Analysis 2 and update $y_2(y_1, y_3)$
3: Evaluate Analysis 3 and update $y_3(y_1, y_2)$
until $4 \rightarrow 1$: MDA has converged



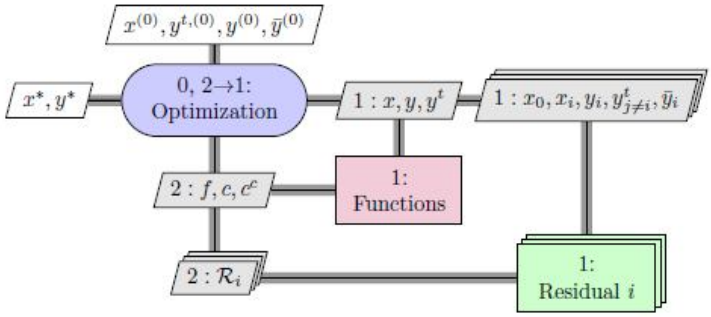
A block Gauss–Seidel multidisciplinary analysis (MDA)



Classification of the MDO architectures.



A gradient-based optimization procedure.



XDSM for solving the AAO problem.

The All-at-Once (AAO) Problem Statement

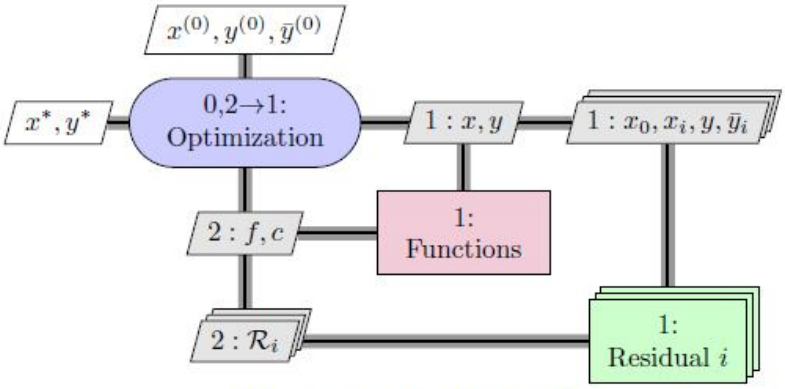


Diagram for the SAND architecture.

Simultaneous Analysis and Design (SAND)

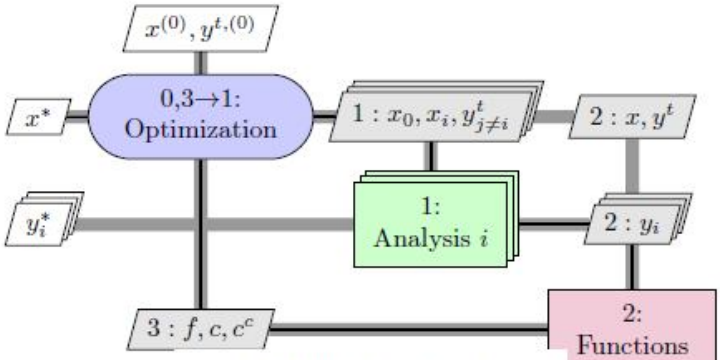


Diagram of the IDF architecture.

Individual Discipline Feasible (IDF)

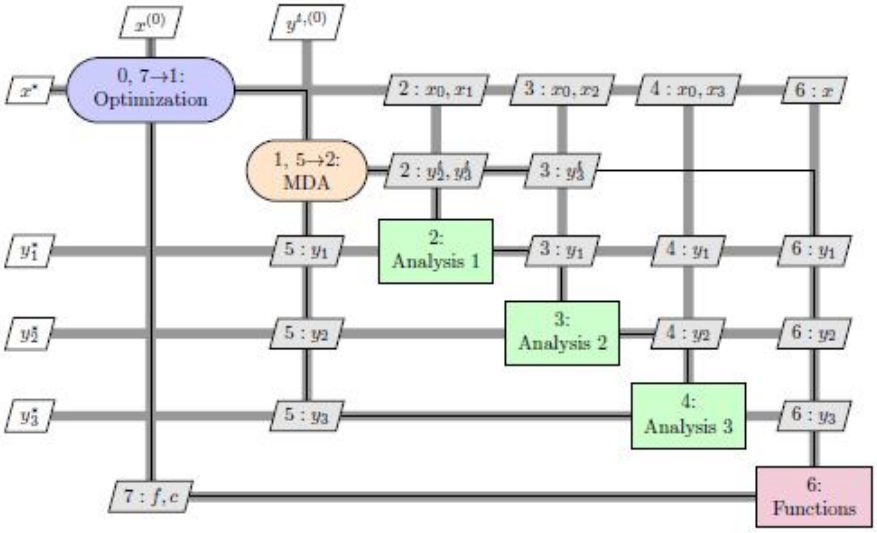
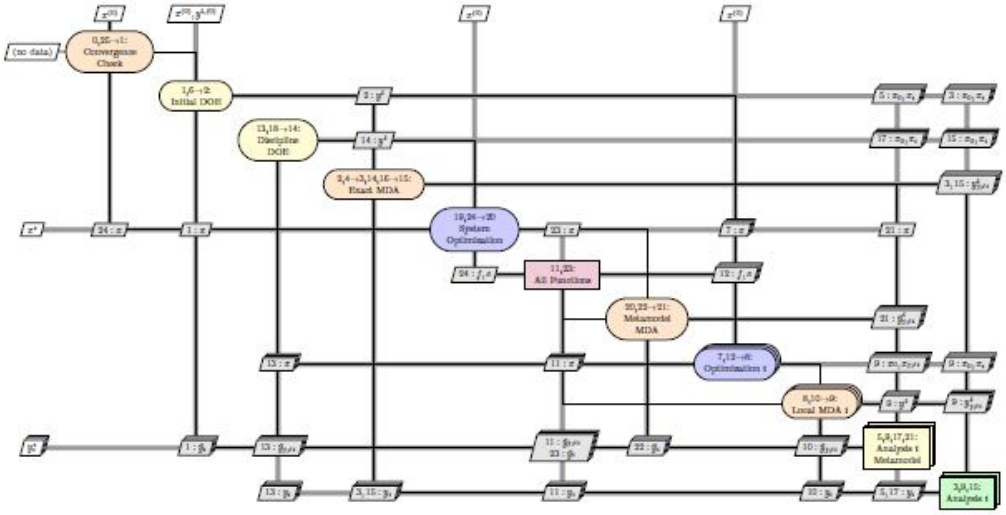


Diagram for the MDF architecture with a Gauss–Seidel multidisciplinary analysis.

Multidisciplinary Feasible (MDF)



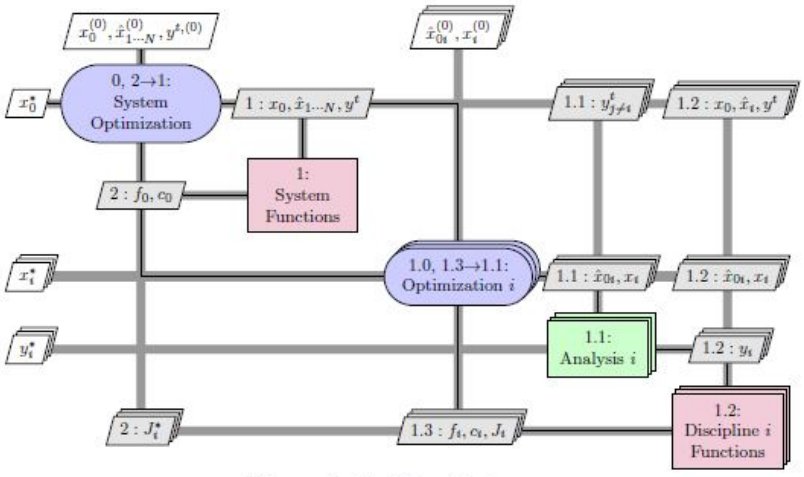
: Diagram for the CSSO architecture.

Concurrent Subspace Optimization (CSSO)

Algorithm 15 CSSO

Input: Initial design variables x
Output: Optimal variables x^*, objective function f^*, and constraint values c^*

0: Initiate main CSSO iteration
repeat
  1: Initiate a design of experiments (DOE) to generate design points
  for Each DOE point **do**
    2: Initiate an MDA that uses exact disciplinary information
    repeat
      3: Evaluate discipline analyses
      4: Update coupling variables y
    until $4 \rightarrow 3$: MDA has converged
    5: Update the disciplinary surrogate models with the latest design
  end for $6 \rightarrow 2$
  7: Initiate independent disciplinary optimizations (in parallel)
  for Each discipline i **do**
    repeat
      8: Initiate an MDA with exact coupling variables for discipline i and approximate coupling variables for the other disciplines
      repeat
        9: Evaluate discipline i outputs y_i, and surrogate models for the other disciplines, $\bar{y}_{j \neq i}$
      until $10 \rightarrow 9$: MDA has converged
      11: Compute objective f_0 and constraint functions c using current data
    until $12 \rightarrow 8$: Disciplinary optimization i has converged
  end for
  13: Initiate a DOE that uses the subproblem solutions as sample points
  for Each subproblem solution i **do**
    14: Initiate an MDA that uses exact disciplinary information
    repeat
      15: Evaluate discipline analyses.
    until $16 \rightarrow 15$ MDA has converged
    17: Update the disciplinary surrogate models with the newest design
  end for $18 \rightarrow 14$
  19: Initiate system-level optimization
  repeat
    20: Initiate an MDA that uses only surrogate model information
    repeat
      21: Evaluate disciplinary surrogate models
    until $22 \rightarrow 21$: MDA has converged
    23: Compute objective f_0, and constraint function values c
  until $24 \rightarrow 20$: System level problem has converged
until $25 \rightarrow 1$: CSSO has converged



: Diagram for the CO architecture.

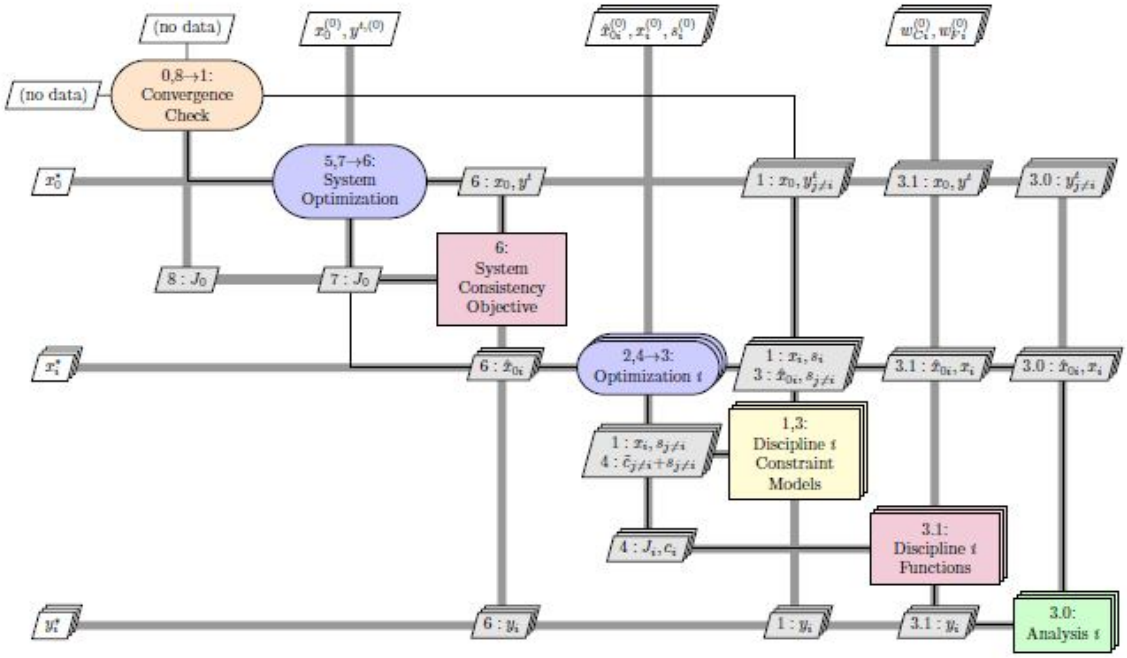
Collaborative Optimization (CO)

Algorithm 16 Collaborative optimization

Input: Initial design variables x

Output: Optimal variables x^*, objective function f^*, and constraint values c^*

0: Initiate system optimization iteration
repeat
  1: Compute system subproblem objectives and constraints
  for Each discipline i (in parallel) **do**
    1.0: Initiate disciplinary subproblem optimization
    repeat
      1.1: Evaluate disciplinary analysis
      1.2: Compute disciplinary subproblem objective and constraints
      1.3: Compute new disciplinary subproblem design point and J_i
    until 1.3 $\rightarrow$ 1.1: Optimization i has converged
  end for
  2: Compute a new system subproblem design point
until 2 $\rightarrow$ 1: System optimization has converged



XDSM for the ECO architecture

Algorithm 17 Enhanced collaborative optimization

Input: Initial design variables x

Output: Optimal variables x^*, objective function f^*, and constraint values c^*

0: Initiate ECO iteration

repeat

for Each discipline i **do**

1: Create linear constraint model

2: Initiate disciplinary subproblem optimization

repeat

3: Evaluate nonlocal constraint models with local duplicates of shared variables

3.0: Evaluate disciplinary analysis

3.1: Compute disciplinary subproblem objective and constraints

4: Compute new disciplinary subproblem design point and J_i

until $4 \rightarrow 3$: Disciplinary optimization subproblem has converged

end for

5: Initiate system optimization

repeat

6: Compute J_0

7: Compute updated values of x_0 and y^t.

until $7 \rightarrow 6$: System optimization has converged

until $8 \rightarrow 1$: The J_0 is below specified tolerance

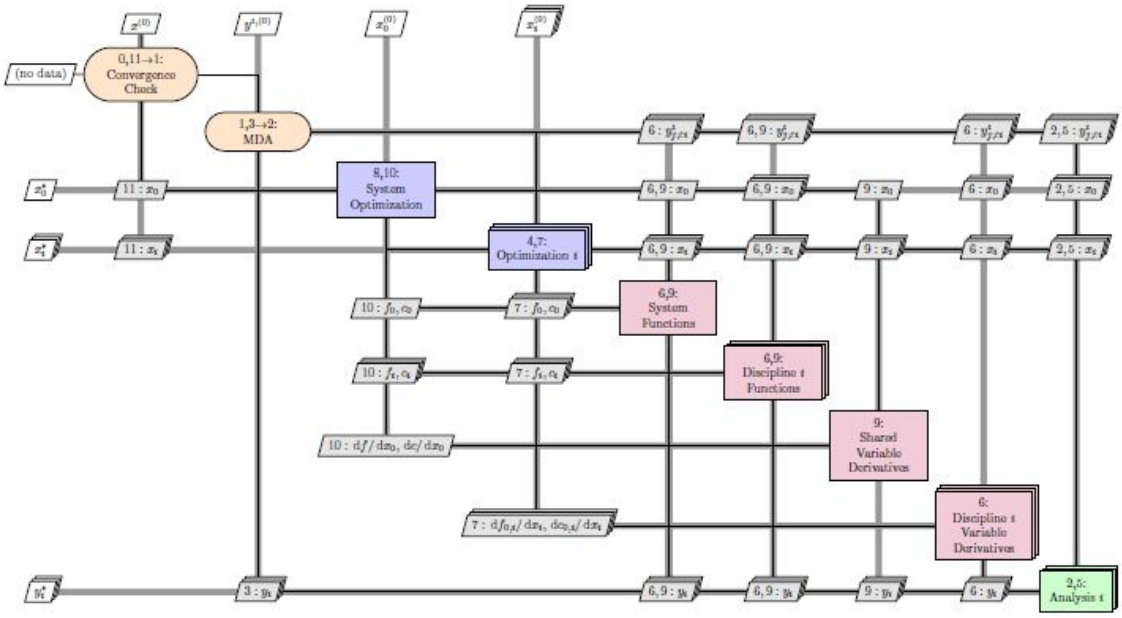


Diagram for the BLISS architecture

Algorithm 18 BLISS

Input: Initial design variables x
Output: Optimal variables x^*, objective function f^*, and constraint values c^*

0: Initiate system optimization
repeat
  1: Initiate MDA
  repeat
    2: Evaluate discipline analyses
    3: Update coupling variables
  until 3 → 2: MDA has converged
  4: Initiate parallel discipline optimizations
  for Each discipline i **do**
    5: Evaluate discipline analysis
    6: Compute objective and constraint function values and derivatives with respect to local design variables
    7: Compute the optimal solutions for the disciplinary subproblem
  end for
  8: Initiate system optimization
  9: Compute objective and constraint function values and derivatives with respect to shared design variables using post-optimality analysis
  10: Compute optimal solution to system subproblem
until 11 → 1: System optimization has converged

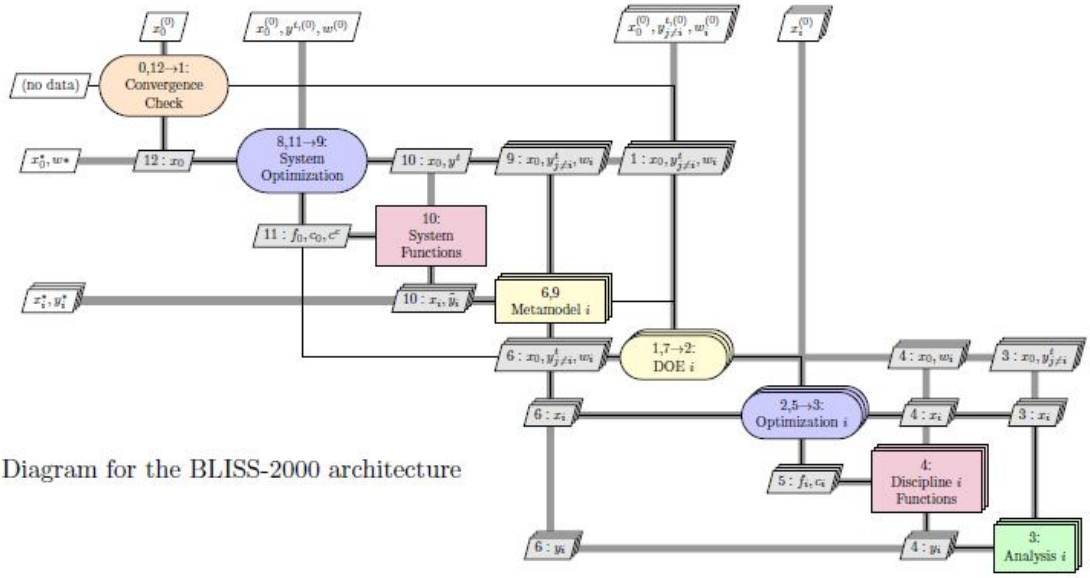


Diagram for the BLISS-2000 architecture

Algorithm 19 BLISS-2000

Input: Initial design variables x
Output: Optimal variables x^*, objective function f^*, and constraint values c^*

```
0: Initiate system optimization
repeat
  for Each discipline i do
    1: Initiate a DOE
    for Each DOE point do
      2: Initiate discipline subproblem optimization
      repeat
        3: Evaluate discipline analysis
        4: Compute discipline objective f_i and constraint functions c_i
        5: Update the local design variables x_i
      until 5 → 3: Disciplinary optimization subproblem has converged
      6: Update surrogate model of optimized disciplinary subproblem with new solution
    end for  7 → 1
  end for
  8: Initiate system subproblem optimization
  repeat
    9: Interrogate surrogate models with current values of system variables
    10: Compute system objective and constraint function values
    11: Compute a new system design point
  until 11 → 9 System subproblem has converged
until 12 → 1: System optimization has converged
```

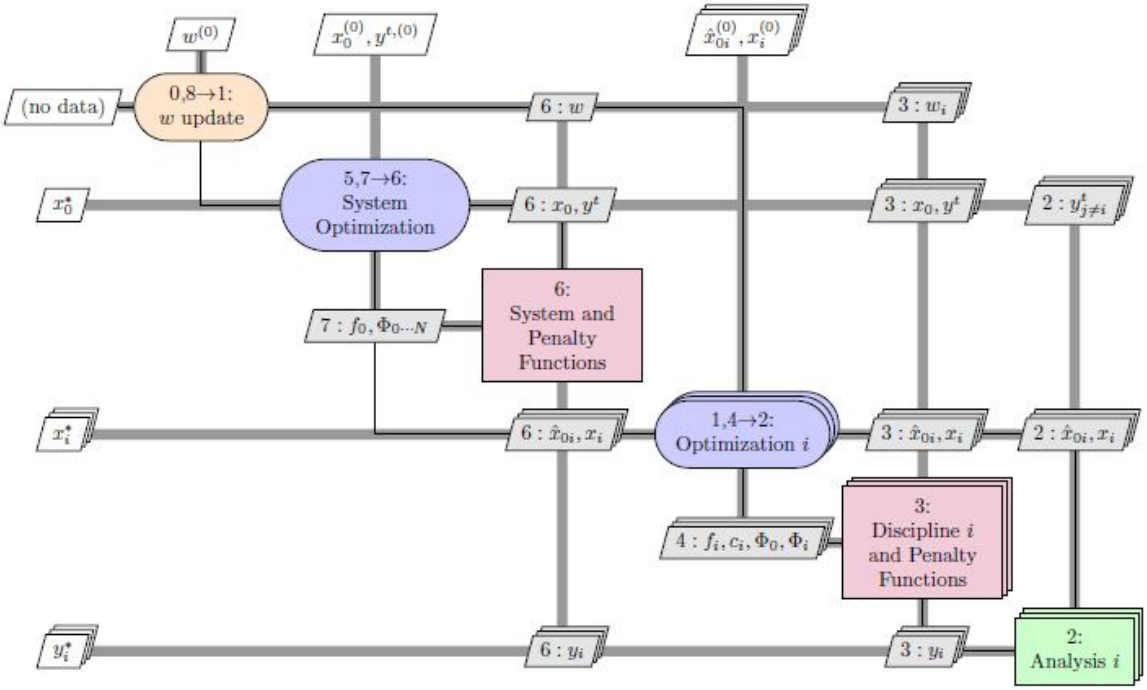


Diagram for the ATC architecture

Algorithm 20 ATC

Input: Initial design variables x
Output: Optimal variables x^*, objective function f^*, and constraint values c^*
0: Initiate main ATC iteration
repeat
for Each discipline i **do**
1: Initiate discipline optimizer
repeat
2: Evaluate disciplinary analysis
3: Compute discipline objective and constraint functions and penalty function values
4: Update discipline design variables
until $4 \rightarrow 2$: Discipline optimization has converged
end for
5: Initiate system optimizer
repeat
6: Compute system objective, constraints, and all penalty functions
7: Update system design variables and coupling targets.
until $7 \rightarrow 6$: System optimization has converged
8: Update penalty weights
until $8 \rightarrow 1$: Penalty weights are large enough

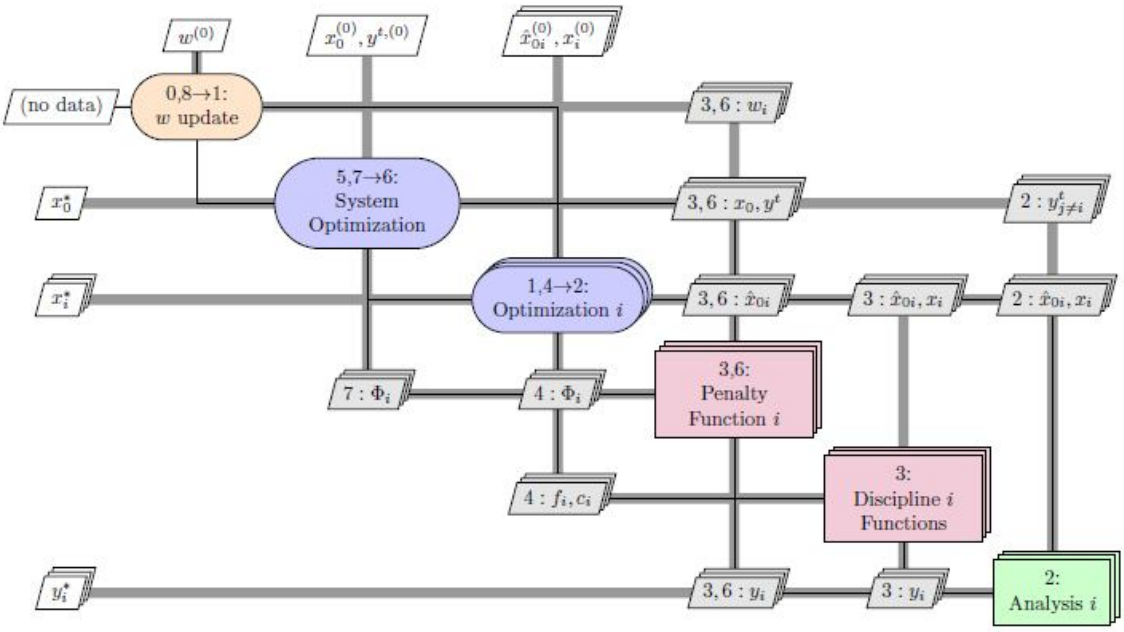


Diagram for the penalty decomposition architectures EPD and IPD

Algorithm 21 EPD and IPD

Input: Initial design variables x
Output: Optimal variables x^*, objective function f^*, and constraint values c^*
0: Initiate main iteration
repeat
  for Each discipline i **do**
    repeat
      1: Initiate discipline optimizer
      2: Evaluate discipline analysis
      3: Compute discipline objective and constraint functions, and penalty function values
      4: Update discipline design variables
    until $4 \rightarrow 2$: Discipline optimization has converged
  end for
  5: Initiate system optimizer
  repeat
    6: Compute all penalty functions
    7: Update system design variables and coupling targets
  until $7 \rightarrow 6$: System optimization has converged
  8: Update penalty weights.
until $8 \rightarrow 1$: Penalty weights are large enough

MDO of Independent Subspaces (MDOIS)

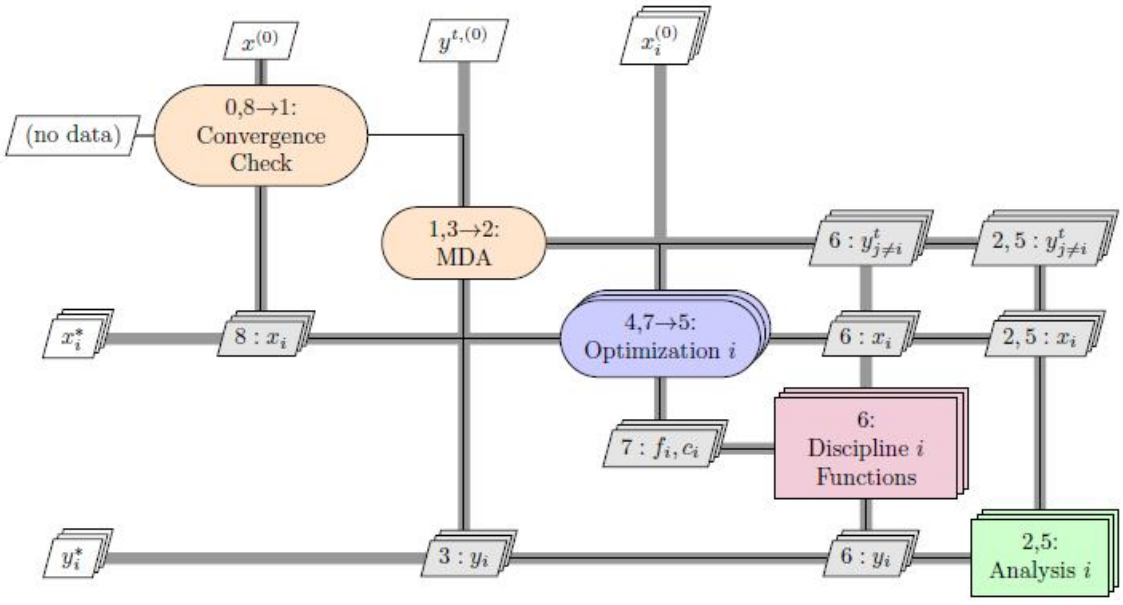


Diagram for the MDOIS architecture

Algorithm 22 MDOIS

Input: Initial design variables x
Output: Optimal variables x^*, objective function f^*, and constraint values c^*
0: Initiate main iteration
repeat
  repeat
    1: Initiate MDA
    2: Evaluate discipline analyses
    3: Update coupling variables
  until $3 \rightarrow 2$: MDA has converged
  for Each discipline i **do**
    4: Initiate disciplinary optimization
    repeat
      5: Evaluate discipline analysis
      6: Compute discipline objectives and constraints
      7: Compute a new discipline design point
    until $7 \rightarrow 5$: Discipline optimization has converged
  end for
until $8 \rightarrow 1$ Main iteration has converged

Algorithm 23 QSD

Input: Initial design variables x
Output: Optimal variables x^*, objective function f^*, and constraint values c^*
0: Initiate system optimization
repeat
  1: Compute system objectives and constraints
  for Each discipline i **do**
    1.0: Initiate discipline optimization
    repeat
      1.1: Evaluate discipline analysis
      1.2: Compute discipline objective and constraints
      1.3: Update discipline design point
    until $1.3 \rightarrow 1.1$: Discipline optimization has converged
  end for
  2: Compute a new system design point
until $2 \rightarrow 1$: System problem has converged

Quasiseparable Decomposition (QSD)

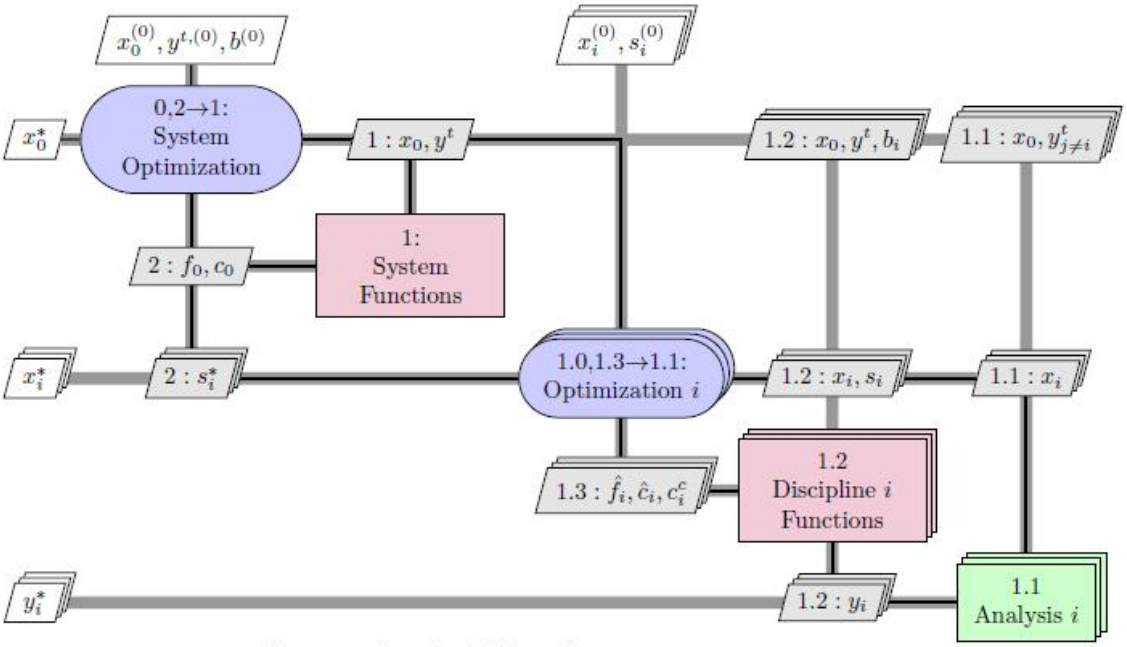


Diagram for the QSD architecture

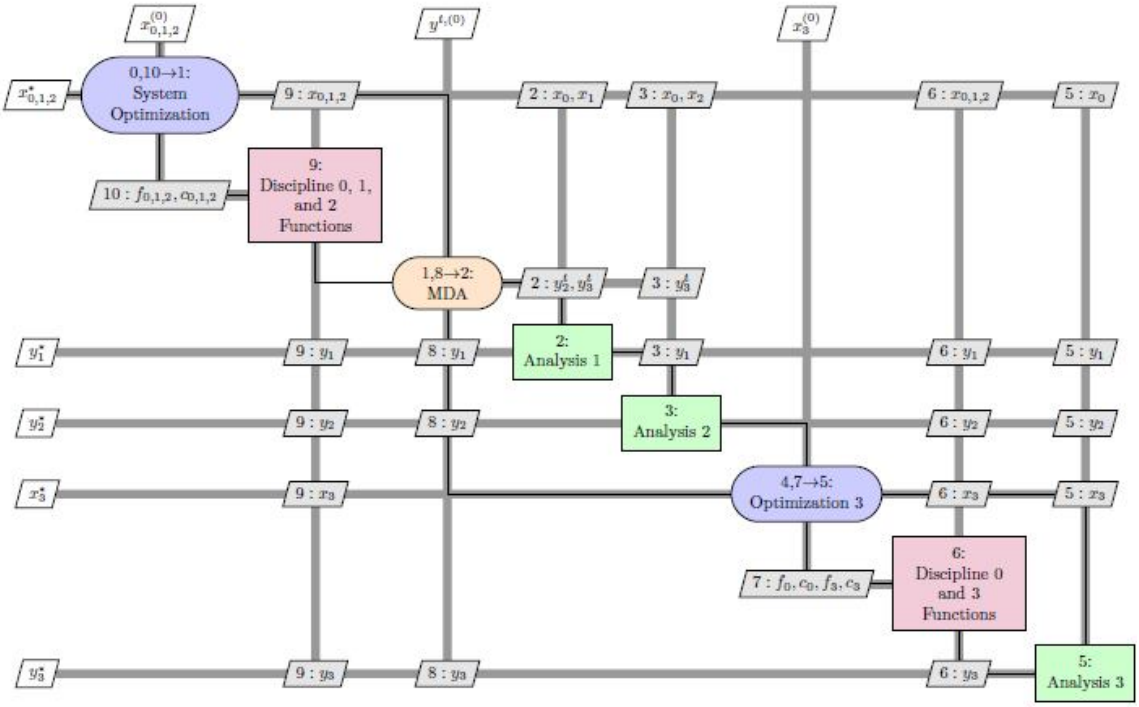
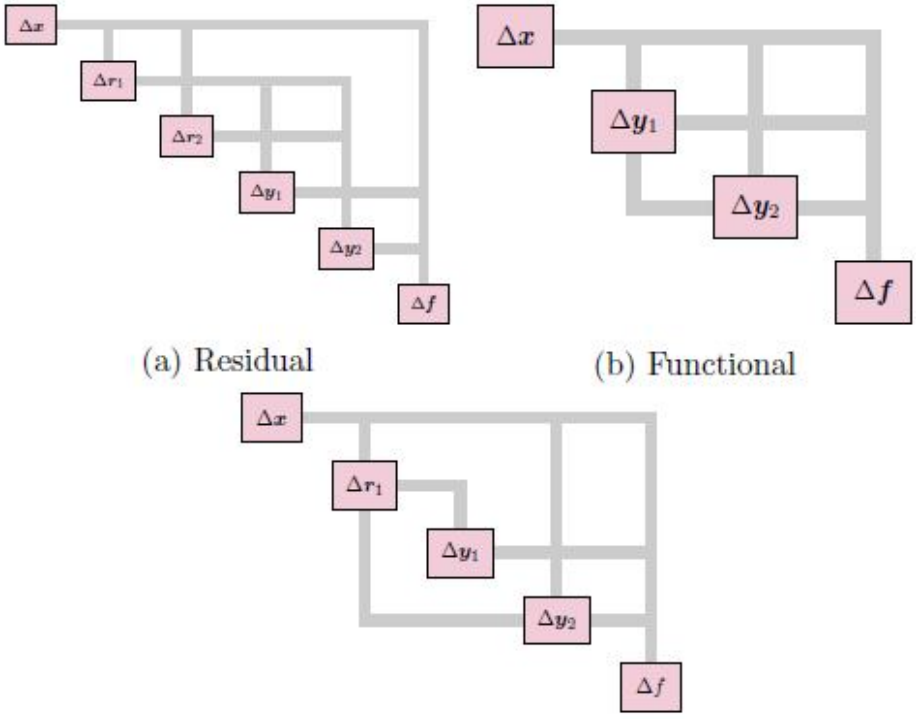


Diagram for the ASO architecture

Asymmetric Subspace Optimization (ASO)

Algorithm 24 ASO

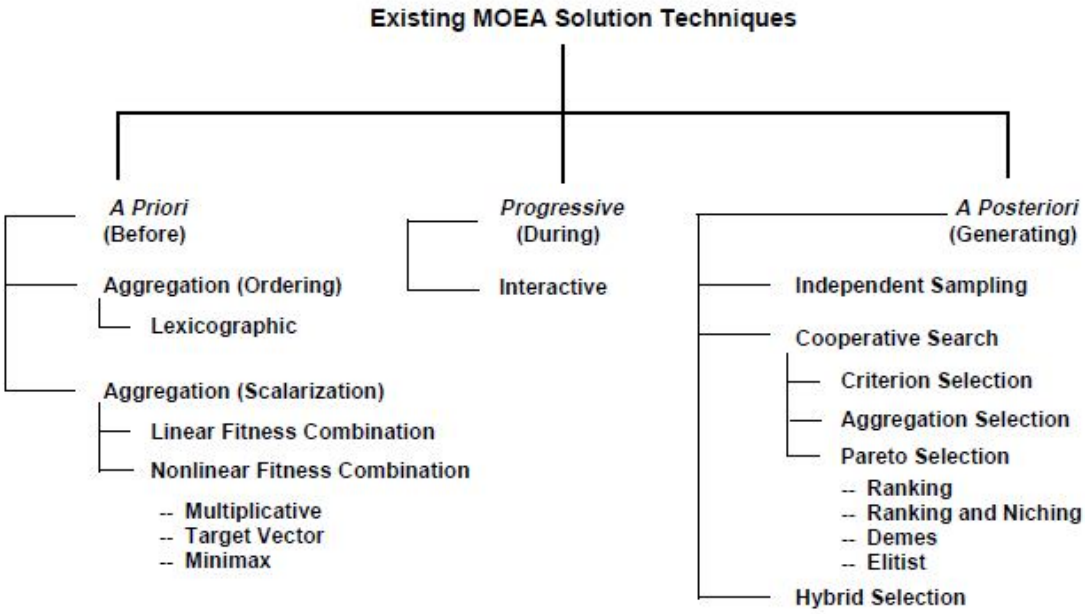
Input: Initial design variables x
Output: Optimal variables x^*, objective function f^*, and constraint values c^*
0: Initiate system optimization
repeat
1: Initiate MDA
repeat
2: Evaluate Analysis 1
3: Evaluate Analysis 2
4: Initiate optimization of Discipline 3
repeat
5: Evaluate Analysis 3
6: Compute discipline 3 objectives and constraints
7: Update local design variables
until 7 $\rightarrow$ 5: Discipline 3 optimization has converged
8: Update coupling variables
until 8 $\rightarrow$ 2 MDA has converged
9: Compute objective and constraint function values for all disciplines 1 and 2
10: Update design variables
until 10 $\rightarrow$ 1: System optimization has converged



(a) Residual (b) Functional

(c) Hybrid

The different approaches for handling coupled multidisciplinary systems



Multi Objective Evolutionary Algorithms

مراجع

[1] D. E. Glover, Genetic Algorithm and Simulated Annealing, page 12-31, Morgan Kaufmann, Los Altos, CA, 1987.
[2] Lissa W. Light and Peter G. Anderson, “Typewriter keyboards via simulated annealing”, AI Expert, September 1993.
[3] Peter Klausler, www.visi.com/~pmk/evovled.html, September 2005.
[4] M. O. Wagner, B Yannou, S. Kehl, D. Feillet, and J. Eggers, Ergonomic Modeling and optimization of keyboard arrangement with an ant colony algorithm”, European Journal of Operation research, 2003.
[5] P. S. Deshwall and K. Deb, Design of an Optimal Hindi Keyboard for Convenient and Efficient Use. Technical Report on KanGAL Report No. 2003005, Indian Institute of Technology, Kanpur, 2003.
[6] D. A. Norman and D. E. Rumelhart, Cognitive Aspects of Skilled Typing, Springer-Verlag, New York, 1983.
[7] J. S. Goetti, A.W. Brugh, and B. A. Julstrom, “Arranging the Keyboard with a Permutaion-Coded Genetic Algorithm”, Proc. of the 2005 SCM Symposium on Applied Computing, Volume 2, pp. 947-951, 2005
https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
https://www.saedsayad.com/k_nearest_neighbors.htm