
بسم الله الرحمن الرحیم

دانشکده علوم ریاضی

رشته ریاضی کاربردی، گرایش تحقیق در عملیات

پایان‌نامه کارشناسی ارشد

الگوریتم جست‌وجوی هارمونی برای حل مسائل بهینه‌سازی چندهدفه

نگارنده: زهرا پورغراوی

استادان راهنما

دکتر مهرداد غزنوی
دکتر مریم قرآنی

بهمن ۱۳۹۵

فرم شماره 7: صورتجلسه دفاع از پایان نامه تحصیلی دوره کارشناسی ارشد

با تأییدات خداوند متعال و با استعانت از حضرت ولی عصر (عج) ارزیابی جلسه دفاع از پایان نامه کارشناسی ارشد خانم / آقای زهرا پورغراوی به شماره دانشجویی 9304604 رشته ریاضی کاربردی گرایش **تحقیق در عملیات** تحت عنوان الگوریتم جست‌وجوی هارمونی برای حل مسائل بهینه‌سازی چند هدفه که در تاریخ 95/11/11 با حضور هیأت محترم داوران در دانشگاه صنعتی شاهرود برگزار گردید به شرح ذیل اعلام می‌گردد:

قبول (با درجه: عالی (19)) ☑ دفاع مجدد ☐ مردود ☐

نوع تحقیق: نظری ☑ عملی ☐

1ـ عالی (20 ـ 19) 2ـ بسیار خوب (18/99 ـ 18)
3ـ خوب (17/99 ـ 16) 4ـ قابل قبول (15/99 ـ 14)
5- نمره کمتر از 14 غیر قابل قبول

عضو هیأت داوران	نام ونام خانوادگی	مرتبهٔ علمی	امضاء
1ـ استادراهنمای اول	دکتر مهرداد غزنوی	استادیار	
2- استادراهنمای دوم	دکتر مریم قرآنی	استادیار	
3- استاد مشاور			
4- نماینده شورای تحصیلات تکمیلی	دکتر مهدی قوتمند	استادیار	
5- استاد ممتحن اول	دکتر جعفر فتحعلی	دانشیار	
6-ـ استاد ممتحن دوم	دکتر محمد هادی نوری اسکندری	استادیار	

رئیس دانشکده:

تقدیم به:

ارواح پاک پدر و مادر مهربانم؛

پدری که عالمانه به من آموخت تا چگونه در عرصه زندگی، ایستادگی را تجربه نمایم

و مادری که دریای بی کران فداکاری و ایثار بود و وجودش برایم همه مهر.

برادرم که همواره در طول تحصیل متحمل زحماتم بود و تکیه گاه من در مواجهه با مشکلات

و وجودش مایه دلگرمی من می باشد.

دوستان مهربانم که مایه دلگرمی من در این مسیر بودند.

و آزادمردان و زنانی که نیک می اندیشند.

به مصداق "من لم یشکر المخلوق لم یشکر الخالق"

بسی شایسته است از:

استاد گرامی جناب آقای دکتر غزنوی تشکر نمایم که بدون راهنمایی های ایشان، تأمین این پایان نامه برایم بسیار مشکل می نمود.

همچنین از استاد محترم خانم دکتر قرآنی که با حسن خلق زحمت راهنمایی این پایان نامه را بر عهده گرفتند.

استادان گرامی آقایان دکتر فتحعلی و دکتر نوری اسکندری که زحمت داوری این پایان نامه را متقبل شدند، کمال تشکر و قدردانی را دارم.

«شکر خدا که هرچه طلب کردم از خدا، بر منتهای همت خود کامران شدم.»

زهرا پورغراوی

بهمن ۱۳۹۵

تعهد نامه

اینجانب **زهرا پورغراوی** دانشجوی کارشناسی ارشد رشته **ریاضی کاربردی علوم ریاضی** دانشگاه شاهرود، نویسنده پایان‌نامه با عنوان **الگوریتم جست‌وجوی هارمونی برای حل مسائل بهینه‌سازی چندهدفه** ، تحت راهنمایی **مهرداد غزنوی** متعهد می‌شوم:

- تحقیقات در این پایان‌نامه توسط اینجانب انجام شده است و از صحت و اصالت برخوردار است.
- در استفاده از نتایج پژوهش‌های دیگر پژوهش‌گران، به مرجع مورد استفاده استناد شده است.
- مطالب این پایان‌نامه، تا کنون توسط خود، یا فرد دیگری برای دریافت هیچ نوع مدرک یا امتیازی در هیچ‌جا ارایه نشده است.
- حقوق معنوی این اثر، به دانشگاه صنعتی شاهرود تعلق دارد، و مقالات مستخرج با نام‘‘ دانشگاه صنعتی شاهرود ’’ یا ‘‘ Shahrood University of Technology ‘‘ به چاپ خواهد رسید.
- حقوق معنوی تمام افرادی که در به‌دست آوردن نتایج اصلی پایان‌نامه تاثیرگذار بوده‌اند، در مقالات مستخرج از پایان‌نامه رعایت می‌گردد.
- در تمام مراحل انجام این پایان‌نامه، در مواردی که از موجود زنده (یا بافت‌های آنها) استفاده شده است، ضوابط و اصول اخلاقی رعایت شده است.
- در تمام مراحل انجام این پایان‌نامه، در مواردی که به حوزه اطلاعات شخصی افراد دسترسی یافته (یا استفاده شده است)، اصل رازداری و اصول اخلاق انسانی رعایت شده است.

زهرا پورغراوی
بهمن ۱۳۹۵

مالکیت نتایج و حق نشر

- تمام حقوق معنوی این اثر و محصولات آن (مقالات مستخرج، کتاب، برنامه‌های رایانه‌ای، نرم‌افزارها و تجهیزات ساخته شده) متعلق به دانشگاه صنعتی شاهرود می‌باشد. این مطلب باید به نحو مقتضی، در تولیدات علمی مربوطه ذکر شود.
- استفاده از اطلاعات و نتایج موجود در این پایان‌نامه بدون ذکر منبع مجاز نمی‌باشد.

چکیده

الگوریتم جست‌وجوی هارمونی (HS) یک روش فراابتکاری جدید می‌باشد که از فرآیند موسیقی الهام گرفته شده است. این الگوریتم به عنوان یک روش فراابتکاری بهینه‌سازی برای مسائل تک‌هدفه پیشنهاد شد و از زمان ایجاد آن مشخص شد که در مسائل مهندسی و علمی به‌صورت موفقیت آمیزی کاربرد دارد. در این پایان‌نامه، ابتدا دو طرح جست‌وجوی هارمونی (MOHS۱، MOHS۲) برای حل مسائل بهینه‌سازی چندهدفه پیشنهاد می‌شود. در ادامه، برای اثبات اثر بخشی دو طرح پیشنهاد شده از توابع ZDT به عنوان توابع تست مورد استفاده قرار گرفته و نتایج الگوریتم با جواب‌های به‌دست آمده با الگوریتم ژنتیک غیرتسلطی نوع دو (NSGA-II) مقایسه می‌شوند. اگر چه الگوریتم جست‌وجوی هارمونی مزایای زیادی در حل مسائل بهینه‌سازی نشان می‌دهد، پارامترهای آن باید براساس تجربه و ویژگی‌های مسئله توسط کاربران مشخص شوند. این امر موجب مشکلات زیادی برای کاربران مبتدی می‌شود. برای غلبه بر این مشکل، یک الگوریتم جست‌وجوی هارمونی چند هدفه خود انطباق (SAMOHS) براساس واریانس حافظه هارمونی مطرح می‌شود. برای حل مسائل بهینه‌سازی چند هدفه، الگوریتم پیشنهاد شده از روش کوتاه کردن و مرتب‌سازی غیرتسلطی برای به‌روز رسانی حافظه هارمونی استفاده می‌کند. در ادامه، الگوریتم SAMOHS پیشنهادشده با الگوریتم‌های تکاملی چندهدفه دیگر (SPEA۲، MOPSO، NSGA-II) و نیز با الگوریتم‌های جست‌وجوی هارمونی چندهدفه پیشنهادی (MOHS۱، MOHS۲) مقایسه می‌شود.

کلمات کلیدی:جست‌وجوی هارمونی، بهینه‌سازی چند هدفه، تسلط پارتو، بهینگی پارتو، تنظیم پارامتر خود انطباق.

فهرست مطالب

فهرست تصاویر

فهرست جداول

فصل ۱

برخی الگوریتم‌های فراابتکاری در بهینه‌سازی غیرخطی

۱.۱ مقدمه

امروزه بهینه‌سازی[۱] کاربرد فراوانی در رشته‌های مختلف اعم از مهندسی، اقتصاد، مدیریت و ... دارد. بهینه‌سازی فرآیندی برای یافتن بهترین پاسخ برای مسائلی است که به‌صورت ریاضی تعریف شده‌اند. فرآیند بهینه‌سازی به دنبال یافتن بهترین مقدار قابل دست یابی از یک تابع هدف[۲] تعریف شده بر یک دامنه معین از مقادیر است. هر مسئله بهینه‌سازی شامل سه بخش اصلی تابع هدف، محدودیت‌ها و متغیرهای تصمیم است. اگر در یک مسئله بهینه‌سازی تابع هدف و توابع متناظر با قیود تساوی و نامساوی خطی باشند، با یک مسئله بهینه‌سازی خطی[۳] روبه‌رو هستیم، در غیر این‌صورت اگر حداقل یک قید یا تابع هدف خطی نباشند، آن‌گاه مسئله را یک مسئله بهینه‌سازی غیرخطی[۴] می‌نامیم. روش‌های مختلفی برای حل مسائل بهینه‌سازی غیرخطی وجود دارد که هر یک در شرایط و مقتضیات خاص خود عملکرد مناسب را دارا هستند و در واقع هیچ الگوریتم خاصی برای حل کلیه مسائل غیرخطی وجود ندارد. از این رو روش‌های مختلفی برای بهینه‌سازی غیرخطی مورد استفاده قرار می‌گیرد که

---

[۱] Optimization
[۲] Objective Function
[۳] Linear Optimization Problem
[۴] Non-Linear Optimization Problem

الگوریتم‌های فراابتکاری[۵] یکی از آنها می‌باشند.

در روش‌های ریاضی که به‌طور معمول از مفهوم گرادیان یا مشتق تابع هدف در فرآیند بهینه‌سازی استفاده می‌کنند، شرط پیوستگی و مشتق‌پذیری، یکی از اصول اساسی آنها محسوب می‌شود. در کنار روش‌های ریاضی، روش‌های بهینه‌سازی فراابتکاری نیز معرفی شده‌اند که به‌طور معمول مبتنی بر مفاهیم هوش مصنوعی یا محاسبات تکاملی[۶] برگرفته از طبیعت می‌باشند. این روش‌ها این مزیت را دارند که در شرایطی که فضای جست‌وجو نامحدود بوده و یا تابع هدف بسیار پیچیده و دارای چندین نقطه ماکزیمم یا مینیمم سراسری باشد، بدون نیاز به محاسبه مشتق توابع هدف، نقطه یا نقاط بهینه را با سرعت بیشتری نسبت به روش‌های ریاضی، تخمین بزنند.

یک روش بهینه‌سازی کارا، باید وابستگی کمتری به جواب اولیه داشته باشد. برای مثال، در یک مسئله کمینه‌سازی[۷]، علاوه بر تغییرات و جواب‌هایی که سبب کاهش تابع هدف می‌گردند، تغییراتی که افزایش تابع هدف را به همراه دارند نیز تا حدی قابل قبول خواهند بود؛ اما به هر حال چون هدف نهایی، همگرایی جواب‌ها به یک نقطه مینیمم می‌باشد، تغییراتی که سبب افزایش تابع هدف می‌شوند، بایستی کنترل شده و محدود باشند. مطالب این فصل از مراجع [۴]ـ[۲] گرفته شده است.

الگوریتم‌های فراابتکاری، برای خارج شدن از بهینه‌های محلی[۸] یا جلوگیری از قرار گرفتن در بهینه‌های محلی استفاده می‌شوند. در مسائل بهینه‌سازی، تفکیک نقاط بهینه، به دو دسته بهینه سراسری[۹] و بهینه محلی بسیار ضروری می‌باشد. براساس تعریف، نقطه بهینه سراسری، نقطه‌ای از فضای تصمیم[۱۰] است که نسبت به سایر نقاط، کمترین یا بیشترین مقدار تابع هدف را ارائه دهد که حالت اول را مینیمم سراسری و حالت دوم را ماکزیمم سراسری می‌خوانند. همچنین، نقطه بهینه محلی به نقطه‌ای گفته می‌شود که تنها نسبت به نقاط همسایه خود و نه کل تابع، حداقل یا حداکثر باشد که حالت اول را مینیمم محلی و حالت دوم را ماکزیمم محلی می‌گویند. ویژگی مشترک الگوریتم‌های فراابتکاری، استفاده از مکانیزم‌های خروج از بهینه محلی است [۴۱]. الگوریتم‌های فراابتکاری به دو گروه روش‌های مبتنی بر یک جواب[۱۱] و مبتنی بر جمعیت[۱۲] تقسیم می‌شوند. الگوریتم‌های مبتنی بر یک جواب در حین فرآیند جست‌وجو یک جواب را تغییر می‌دهند و بر روی مناطق محلی جست‌وجو تمرکز دارند؛ الگوریتم‌هایی مانند روش جست‌وجوی انطباقی حریصانه[۱۳]، الگوریتم جست‌وجوی ممنوعه[۱۴] و الگوریتم تبرید شبیه‌سازی شده[۱۵] در این دسته از الگوریتم‌ها قرار می‌گیرند. در حالی‌که الگوریتم‌های مبتنی بر جمعیت در حین جست‌وجو یک جمعیت از جواب‌ها را در نظر می‌گیرند و می‌توانند جست‌وجو را به‌طور هم‌زمان در مناطق مختلفی از فضای جواب انجام دهند. الگوریتم‌های مختلفی بر این اساس شکل گرفته‌اند از جمله: الگوریتم

---

[۵] Metaheuristic Algorithms
[۶] Evoloutionary Computing
[۷] Minimization
[۸] Local optimal
[۹] Global optimal
[۱۰] Decision Space
[۱۱] Single-Solution
[۱۲] Population
[۱۳] Greedy Adaptive Search Procedure (GRASP)
[۱۴] Tabu Search (TS)
[۱۵] Simulated Annealing (SA)

ژنتیک[۱۶]، الگوریتم بهینه‌سازی گروهی ذرات[۱۷]، الگوریتم بهینه‌سازی کلونی مورچگان[۱۸].
مطالعات اخیر حاکی از آن است که روش‌های مبتنی بر جمعیت در توابع چندهدفه بسیار مؤثرتر جلوه می‌کند، البته این موضوع ممکن است که به لحاظ عملیات اجرایی درست باشد، ولی از نظر محاسباتی و تئوری، هیچ قطعیتی در این زمینه وجود ندارد.
دو مفهوم اصلی و مهم در الگوریتم‌های فراابتکاری، ”تنوع [۱۹]و تشدید[۲۰]“ می‌باشد. در حقیقت کارایی[۲۱] یک الگوریتم، با مفاهیم بالا سنجیده می‌شوند. تنوع یا گوناگونی، توانایی الگوریتم برای جست‌وجوی مؤثر و هماهنگ کل فضای جست‌وجو می‌باشد، به طوری که تقریباً محلی از فضای جست‌وجو وجود نداشته باشد که بدون بررسی الگوریتم رها شود. هر چه قدرت الگوریتم در بررسی تمام نقاط فضای جست‌وجو بیشتر باشد درجه تنوع آن الگوریتم بیشتر خواهد بود. در عین حال این توانایی الگوریتم در تضاد با مفهوم دیگر الگوریتم به‌نام تشدید می‌باشد. تشدید توانایی الگوریتم در تمرکز بر نقاط و نواحی از فضای جست‌وجو است که مقدار بهینه تابع در آن نواحی قرار دارد. هر الگوریتم موفق فراابتکاری، نیازمند یک تعادل مناسب بین این دو مؤلفه به‌ظاهر متضاد است. چنان‌چه مؤلفه تشدید بسیار قویتر از حد معمول باشد، تنها کسر کوچکی از فضای جواب قابل دیدن است و احتمال بروز خطا افزایش می‌یابد. در این مواقع اغلب از تکنیک‌های مبتنی بر گرادیان مانند روش نیوتن بهره گرفته می‌شود. حال اگر مؤلفه تنوع بیش از حد قوی باشد، همگرایی در الگوریتم به کندی صورت می‌گیرد و دست‌یابی به جواب بهینه دشوارتر می‌شود و ممکن است برخی از جواب‌ها از قلم بیفتند. معمولاً بسیاری از جواب‌ها به‌صورت تصادفی شروع می‌شوند و کم کم با کاهش مؤلفه تنوع، مؤلفه تشدید رو به افزایش می‌رود که این فرآیند به‌طور هم‌زمان انجام می‌شود.
تمام الگوریتم‌های فراابتکاری براساس یک فرآیند تکراری عمل می‌کنند که از یک نقطه اولیه آغاز و تا زمان برآورده شدن معیارهای از پیش تعیین شده‌ای ادامه می‌یابند. قواعد توقف، باید متناسب با شرایط هر مسئله تعیین شود که این امر نیز عمدتاً با آزمون و خطا میسر می‌شود. برهمین اساس، قواعد توقف متعددی که می‌توانند به صورت تکی یا ترکیبی در کلیه الگوریتم‌ها مورد استفاده قرار گیرند، به صورت زیر می‌باشند:

الف) تعیین یک حد آستانه برای تابع هدف؛ که در این حالت به محض اینکه تابع هدف به این میزان برسد، الگوریتم متوقف خواهد شد.

ب) عدم بهبود در تابع هدف پس از گذشت زمان معین یا پس از تعداد تکرارهای معین.

ج) تعداد تکرارهای از پیش تعیین شده.

انتخاب و به‌کارگیری الگوریتم‌های فراابتکاری، وابستگی شدیدی به شرایط مسئله مورد بحث، نوع متغیرها، بزرگی مسئله و شرایط و محیط به‌کارگیری دارد. تمامی الگوریتم‌های فراابتکاری از الگوریتم‌های

[۱۶] Genetic Algorithm (GA)
[۱۷] Particle Swarm Optimization (PSO)
[۱۸] Ant Colony Optimization (ACO)
[۱۹] Diversification
[۲۰] Intensification
[۲۱] Effectiveness

موفق بهینه‌یابی هستند، ولی موضوع اصلی این است که اشتباه در به‌کارگیری این ابزارها، هزینه و زمان زیادی تلف خواهد کرد. بنابراین باید در انتخاب این ابزارها دقت و شرایط مسئله را به‌صورت دقیق بررسی کرد.

۲.۱ الگوریتم ژنتیک

علمی که به بررسی شباهت و تفاوت بین گونه‌های مختلف طبیعی می‌پردازد، ژنتیک نام دارد. علم ژنتیک به ما کمک می‌کند که توارث در طبیعت، شباهت‌ها و تفاوت‌های آن را با الگوریتم ژنتیک بشناسیم. در دهه هفتاد میلادی دانشمندی از دانشگاه میشیگان به‌نام هالند[۲۲]ایده استفاده از الگوریتم ژنتیک را در بهینه‌سازی‌های مهندسی مطرح کرد. پیشرفت کامپیوتر در طی ۵۰ سال گذشته باعث توسعه روش‌های بهینه‌سازی شده است به طوری که دستورهای متعددی در طی این دوره تدوین گردیده است. یکی از معروف‌ترین و کاراترین این رویکردها الگوریتم‌هایی است که با الگو برداری از تکامل ژنتیکی، الگوهایی برای حل مسئله ارائه می‌کنند و به همین دلیل به الگوریتم ژنتیک نیز معروف شده‌اند. این الگوریتم‌ها یک روش جست‌وجوی مؤثر در فضاهای بسیار بزرگ ایجاد می‌کنند که در نهایت منجر به جهت‌گیری به سمت یافتن جواب بهینه می‌شود. ژن[۲۳]یا ماده وراثتی[۲۴]، ماده پیچیده‌ای است که در هنگام تقسیم می‌تواند همانند خود را به وجود آورد. واحدهایی از این ماده وراثتی از پدر و مادر به فرزند[۲۵]منتقل می‌شوند. این واحدها دارای ویژگی‌های بسیار پایدار بوده و به‌طور مشخص موجودی را که صاحب آن است، تحت تأثیر قرار می‌دهند. ایده اصلی الگوریتم، انتقال خصوصیات موروثی توسط ژن‌ها است. ژن‌ها خصوصیات گونه‌ها یا به‌عبارتی خصوصیات افراد را تشکیل می‌دهند. در الگوریتم ژنتیک، هر متغیر تصمیم در قالب یک ژن که می‌تواند حاوی یک یا چند بیت باشد، نمایش داده می‌شود. مثلاً، یک ژن وجود دارد که بیانگر رنگ چشم است. ژن‌ها بر روی کروموزوم‌ها[۲۶]در جایگاه‌های ویژه، مرتب شده‌اند. کروموزوم به دنباله‌ای از بیت‌ها گفته می‌شود که به شکل کد شده یک جواب ممکن از مسئله مورد نظر است. در حقیقت، بیت‌های یک کروموزوم نقش ژن‌ها را در طبیعت بازی می‌کنند. هر کروموزوم به مثابه یک جواب مسئله می‌باشد. فرض کنید مجموعه خصوصیات انسان توسط کروموزوم‌های او به نسل[۲۷]بعدی منتقل می‌شوند. هر ژن در این کروموزوم‌ها نماینده یک خصوصیت است. به‌عنوان مثال، ژن ۱ می‌تواند رنگ چشم، ژن ۲ طول قد، ژن ۳ رنگ مو باشد. حال اگر این کروموزوم بدون تغییر به نسل بعد انتقال یابد، تمامی خصوصیات نسل بعدی شبیه به خصوصیات نسل قبل خواهد بود. در واقع به‌صورت هم‌زمان دو اتفاق برای کروموزوم‌ها می‌افتد. اتفاق اول جهش[۲۸] است. جهش به این صورت اتفاق می‌افتد که بعضی ژن‌ها به‌صورت کاملاً تصادفی تغییر می‌کنند. البته تعداد این گونه ژن‌ها بسیار کم می‌باشد اما در هر حال این تغییر تصادفی بسیار مهم است. مثلاً ژن رنگ چشم می‌تواند به‌صورت

[۲۲] Holland
[۲۳] Gene
[۲۴] Hereditary factor
[۲۵] Offspring
[۲۶] Chromosom
[۲۷] هر تکرار از الگوریتم را یک نسل می‌گویند.
[۲۸] Mutation

تصادفی باعث شود تا در نسل بعدی یک نفر دارای چشمان سبز باشد، در حالی‌که تمامی نسل قبل دارای چشم قهوه‌ای بوده‌اند. علاوه بر جهش اتفاق دیگری که می‌افتد و البته این اتفاق به تعداد بسیار بیشتری نسبت به جهش رخ می‌دهد، چسبیدن ابتدای یک کروموزوم به انتهای یک کروموزوم دیگر است. این مسئله با نام تقاطع[۲۹] شناخته می‌شود. این همان چیزی است که مثلاً باعث می‌شود تا فرزند تعدادی از خصوصیات پدر و تعدادی از خصوصیات مادر را با هم به ارث ببرد و از شبیه شدن تام فرزند به تنها یکی از والدین[۳۰]جلوگیری می‌کند. تعداد مشخصی از ورودی‌ها $x_1, x_2, \ldots, x_n$ که متعلق به فضای نمونه X هستند، را انتخاب می‌کنیم و آنها را در یک عدد برداری $(x_1, x_2, \ldots, x_n)$ نشان می‌دهیم. در مهندسی نرم افزار اصطلاحاً به آنها کروموزوم می‌گویند. به گروه کروموزوم‌ها، جمعیت می‌گوییم. یکی از ویژگی‌های الگوریتم ژنتیک این است که به جای تمرکز بر روی یک نقطه از فضای جست‌وجو یا یک کروموزوم، بر روی جمعیتی از کروموزوم‌ها تمرکز می‌کند. به این ترتیب در هر مرحله، الگوریتم دارای جمعیتی از کروموزوم‌ها بوده که خواص مورد نظر را بیشتر از جمعیت مرحله قبل داشته باشند. بهترین افراد از یک جمعیت مفروض به عنوان والدین برای ایجاد نسل بعدی انتخاب می‌شود، یا این‌که بدترین افراد از جمعیت مفروض برای جایگزینی به وسیله افرادی جدید انتخاب می‌شوند. تقریباً در تمامی کاربردهای الگوریتم‌های ژنتیک، اندازه جمعیت ثابت است و در طول جست‌وجو تغییر نمی‌کند. تنوع یک جمعیت، معیار وجود تعداد جواب‌های متفاوت در آن جمعیت است. معیار منحصر بفردی برای تنوع وجود ندارد. اگر تعداد کروموزوم‌ها خیلی زیاد باشد، الگوریتم بسیار کند خواهد شد. برای هر کروموزوم x یک مقدار برازندگی[۳۱] داریم که آن را $f(x)$ هم می‌نامیم. کروموزوم‌هایی که مقدار برازندگی آنها بیشتر باشد با احتمال بیشتری برای زنده ماندن در طول دوره‌های دیگر و دوباره تولید شدن را دارند. به عبارت دیگر، الگوریتم ورودی هایی را که به جواب بهینه نزدیک‌ترند، نگه می‌دارد و از بقیه صرف‌نظر می‌کند. با استفاده از این تابع، میزان برازندگی هر جواب نشان داده می‌شود. این تابع، پایه را برای انتخاب تشکیل می‌دهد و در نتیجه بهبودها را آسان می‌کند؛ یعنی این تابع، بهبود را تعریف می‌کند. بیشتر اوقات، مسئله اصلی که به وسیله الگوریتم ژنتیک حل می‌شود، یک مسئله بهینه‌سازی است. در این گونه مسائل تابع برازندگی می‌تواند با تابع هدف مسئله، یکسان یا یک تبدیل ساده از آن باشد [۶۲]. به عنوان مثال، چنان‌چه هدف بیشینه کردن یک تابع باشد، مقدار برازندگی، یک تابع صعودی از تابع هدف در نظر گرفته می‌شود و اگر هدف، یافتن مقدار کمینه یک تابع باشد، مقدار برازندگی، یک تابع نزولی از آن قرار داده می‌شود. معمولاً در مواردی که مناسب باشد، تابع برازندگی را در فاصله‌ی $[0, 1]$ نرمال می‌کنند.

در طول فضای جست‌وجو، الگوریتم ژنتیک به دنبال نقطه‌ای است که بیشترین شایستگی را داشته باشد. در مکانیزم جست‌وجو اگرچه مقدار تابع هدف در تمام فضای جست‌وجو محاسبه نمی‌شود، ولی مقدار محاسبه شده در کلیه زیر فضاهایی که آن نقطه به آنها وابسته بوده، دخالت داده می‌شود. چون در این روش برخلاف روش‌های تک مسیری فضای جواب به‌طور همه جانبه جست‌وجو می‌شود، امکان کمتری برای همگرایی به یک نقطه بهینه محلی وجود خواهد داشت.

---

[۲۹] Crossover
[۳۰] Parent
[۳۱] FitnessValue

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

۱.۲.۱ ساختار کلی الگوریتم ژنتیک

الگوریتم ژنتیک یک تکنیک جست‌وجو را برای یافتن جواب‌های نزدیک به بهینه برای مسائل بهینه‌سازی ارائه می‌کند. این الگوریتم با یک جمعیت اولیه از جواب‌ها شروع می‌شود. هر جواب از طریق یک کروموزوم نمایش داده می‌شود و تمامی جواب‌های ممکن باید با استفاده از یک سیستم کد گذاری به کد تبدیل شوند. سپس مجموعه‌ای از اپراتورهای تولید مثل[۳۲]باید تعیین شوند. اپراتورهای تولید مثل، مستقیماً روی کروموزوم‌ها عمل کرده و سپس کروموزوم‌ها تحت اپراتور جهش و رویه ترکیب[۳۳] قرار می‌گیرند. رویه انتخاب[۳۴] برای رقابت افراد در داخل جمعیت به‌کار می‌رود که براساس تابع برازندگی عمل می‌نماید. برای هر کروموزوم، یک مقدار مرتبط با برازندگی جوابی که نمایش می‌دهد، وجود دارد. الگوریتم ژنتیک به دنبال حداکثر کردن مقدار تابع برازندگی است. بعد از اینکه تولید مثل و تابع برازندگی به‌خوبی تعریف شدند، یک الگوریتم ژنتیک براساس یک ساختار مشابه و پایه طراحی می‌شود. این ساختار ساده با تولید یک جمعیت اولیه از کروموزوم‌ها شروع می‌شود. معمولاً جواب اولیه به‌صورت تصادفی تولید می‌شود. سپس الگوریتم ژنتیک یک رویه تکراری را به منظور تکامل جمعیت انجام می‌دهد. هر تکرار شامل مراحل زیر است:

انتخاب

اولین گام شامل انتخاب افرادی از جمعیت برای تولید مثل است. این انتخاب به‌صورت تصادفی با استفاده از یک احتمال متناسب با تابع برازندگی افراد انجام می‌شود. همواره بهترین فرد شانس بیشتری برای انتخاب را نسبت به ضعیف ترین فرد از لحاظ تابع برازندگی دارد. فرآیند انتخاب منجر به تصمیم‌گیری در ارتباط با اینکه کدام یک از افراد تحت عملگرهای تقاطع و جهش برای تولید نسل جدید به‌کار روند، می‌شود.

تولید مثل

در دومین گام، عملگرهای ترکیب مجدد[۳۵] و جهش روی افراد انتخاب شده به‌کار گرفته شده و کروموزوم‌های جدید تولید می‌شوند که در این مرحله فرزندان جدید تولید می‌شوند. تولید مثل فرآیندی است که با استفاده از آن، خصوصیات ژنتیکی والدین به‌منظور تولید فرزندان، با همدیگر ترکیب می‌شوند.

ارزیابی

در این مرحله، میزان برازندگی فرزندان جدید تولید شده، ارزیابی می‌شود.

جابه‌جایی

---

[۳۲] Reproduction
[۳۳] Crossover
[۳۴] Selection pressure
[۳۵] Recombination

در این مرحله، افرادی از جمعیت قبلی حذف می‌شوند و با افراد جدیدی که به تازگی تولید شده‌اند، جابه‌جا می‌شوند.
عملگر جهش روی یک فرد صورت می‌گیرد که در آن بعضی از خصوصیات ژنتیکی فرد به‌صورت تصادفی تغییر یافته و نسخه جدیدی از آن فرد به‌دست می‌آید. الگوریتم زمانی متوقف می‌شود که جمعیت به سمت جواب بهینه همگرا شود و یا به آن برسد. مراحل الگوریتم ژنتیک استاندارد به‌صورت زیر می‌باشند:

۱. شروع: در این مرحله یک جمعیت اولیه شامل n کروموزوم تولید می‌شود.
۲. برازندگی: مقدار برازندگی $f(x)$ هر کروموزوم x از جمعیت ارزیابی می‌شود.
۳. جمعیت جدید: ایجاد یک جمعیت جدید با تکرار مراحل زیر تا زمانی‌که جمعیت جدید کامل شود:
ـ انتخاب: انتخاب دو کروموزوم از جمعیت مطابق با مقدار برازندگی آنها (افراد با برازندگی بیشتر، شانس بیشتری برای انتخاب دارند.)
ـ عملگر تقاطع: با استفاده از عملگر تقاطع احتمالی روی والدین انتخاب شده، فرزندان جدید تولید می‌شوند. اگر عملگر تقاطع انجام نشود، فرزندان همان والدین خود خواهند بود.
ـ جهش: جهش عملگری است که جواب را تولید می‌کند. بعد از اینکه یک عضو در جمعیت جدید به‌وجود آمد، هر ژن آن با احتمال جهش، جهش می‌یابد و ممکن است ژنی از مجموعه ژن‌های جمعیت حذف شود یا ژنی که تا به‌حال در جمعیت وجود نداشته است به آن اضافه شود.
ـ پذیرش: فرزند جدید در جمعیت جدید قرار داده می‌شود.
۴. جابه‌جایی: جمعیت جدید (فرزندان) با جمعیت قبلی (والدین) با هم تشکیل یک نسل جدید می‌دهند. در این حالت بعضی از والدین حذف شده و بعضی از فرزندان جایگزین آن‌ها می‌شوند و دو جمعیت تبدیل به یک جمعیت شده و اندازه جمعیت ثابت می‌ماند.
۵. شرط توقف: اگر شرط توقف برقرار شد، توقف کنید و بهترین جواب را در آخرین جمعیت گزارش کنید.
۶. تکرار: در صورت عدم برقراری شرط توقف، به گام دوم که برازندگی است، برگردید.

۲.۲.۱ اجزای الگوریتم ژنتیک

در قسمت‌های قبل اجزایی مانند کروموزوم، جمعیت و مقدار برازندگی مورد بحث قرار گرفتند. در ادامه به بررسی بقیه این اجزا می‌پردازیم.

۱. **کد گذاری**

کد گذاری فرآیندی است که به واسطه آن، جواب‌ها در فضای فیزیکی مسئله تبدیل به جواب‌هایی با کدهایی می‌شود که برای کامپیوتر و محاسبات قابل فهم است. به عبارت دیگر، جواب واقعی با استفاده از ژن‌های آن بیان می‌شود. البته ژن‌ها نیز به زبان کامپیوتری بیان می‌شوند. متداول‌ترین نوع کد گذاری، کد گذاری باینری است. در این روش، هر جواب با یک رشته از بیت‌ها که در برگیرنده اعداد صفر و یک است، معرفی می‌شود. مثلاً در مسئله کوله‌پشتی، آرایه [۱۰۰۱۱۱۰۰۱۱] را در نظر بگیرید. این آرایه یک مثال از کد گذاری باینری را برای این مسئله نمایش می‌دهد. این کد مشخص می‌کند که از بین ۱۰ شی، اشیا با شماره $1, 4, 5, 6, 9, 10$

انتخاب شده‌اند.

۲. **انتخاب**

انتخاب، فرآیند تعیین دو والد از جمعیت برای عمل تقاطع است. در این مرحله، باید در ارتباط با انتخاب والدین برای عمل تقاطع، نحوه تولید فرزندان و تعداد فرزندان تصمیم‌گیری شود. هدف انتخاب این است که والدین شایسته‌تر انتخاب شده که منجر به تولید فرزندانی با برازندگی بالا شود. انتخاب روشی است که به‌طور تصادفی کروموزوم‌هایی را از جمعیت، برای تولید مثل بیرون می‌آورد. کروموزوم با تابع برازندگی بالاتر، شانس بیشتری برای انتخاب دارد. فشار رویه انتخاب به معنای درجه توجه به والدین بهتر در رویه انتخاب است که توسط گلدبرگ[۳۶] و دب[۳۷] در سال ۱۹۹۱ معرفی شد [۲۷]. فشار بالای رویه انتخاب منجر به تأکید بیشتر در انتخاب والدین با شایستگی بیشتر می‌شود. همگرایی الگوریتم ژنتیک وابستگی بالایی به دامنه و شدت فشار رویه انتخاب دارد. فشار بالای رویه انتخاب منجر به افزایش نرخ همگرایی الگوریتم می‌شود. الگوریتم ژنتیک قادر به یافتن جواب بهینه یا نزدیک به بهینه، تحت دامنه‌ی معقولی از فشار رویه انتخاب می‌باشد. با این وجود اگر فشار رویه انتخاب خیلی کم باشد، نرخ همگرایی خیلی کند خواهد شد و الگوریتم ژنتیک نیاز به زمان بسیار طولانی و غیر ضروری برای رسیدن به جواب بهینه خواهد داشت. در صورتی‌که فشار رویه انتخاب خیلی زیاد باشد، موجب می‌شود که الگوریتم ژنتیک به‌طور نابه‌هنگام به بهینه تقریبی همگرا شود. از طرفی، در هنگام طراحی برنامه انتخاب، باید دقت شود که گوناگونی جمعیت حفظ شود که از همگرایی زودرس الگوریتم جلوگیری شود. به‌طور کلی، دو نوع برنامه انتخاب وجود دارد، رویه مبتنی بر تناسب[۳۸] و رویه‌ی مبتنی بر ترتیب[۳۹]. رویه‌ی مبتنی بر تناسب، والدین را براساس مقدار برازندگی آنها نسبت به مقدار برازندگی سایر والدین جمعیت انتخاب می‌کند. برنامه انتخاب مبتنی بر ترتیب، والدین را نه براساس مقدار خام برازندگی آنها، بلکه براساس رتبه آنها در جمعیت انتخاب می‌کند. در این حالت، فشار رویه انتخاب مستقل از توزیع مقدار برازندگی جمعیت در نظر گرفته شده و مقدار رتبه آنها در جمعیت به‌جای مقدار برازندگی آنها، در نظر گرفته می‌شود. چند روش برای رویه انتخاب وجود دارد: روش چرخ رولت[۴۰]، روش تصادفی، روش رتبه‌بندی[۴۱]، روش انتخاب رقابتی[۴۲] و روش انتخاب بالتزمن[۴۳] که در اینجا به سه مورد از آنها اشاره می‌کنیم.

---

[۳۶] Goldberg
[۳۷] Deb
[۳۸] Proportionate selection
[۳۹] Ordinal-based selection
[۴۰] Roulette whel
[۴۱] Rank selection
[۴۲] Tournament selection
[۴۳] Boltzmann selection

روش‌های انتخاب

(آ) **روش چرخ رولت**

چرخ رولت، وسیله‌ای است که در آن با چرخاندن یک صفحه، شانس برنده شدن هر فرد در بازی، با توجه به دسته‌ی چرخ مشخص می‌شود. دسته‌ی چرخ بر روی سهم هر فرد که بایستد، آن فرد برنده است. در این روش، ابتدا برای هر کروموزوم از جمعیت مقدار شایستگی آن محاسبه می‌شود. در این مدل، سطح چرخ به بخش‌هایی تقسیم می‌شود که تعداد آنها برابر با تعداد اعضای جمعیت و سطح هر بخش متناسب با مقدار برازندگی هر کروموزوم است. سپس چرخ به گردش در می‌آید تا در نقطه‌ای به تصادف متوقف شود. این نقطه، کروموزوم انتخاب شده را مشخص می‌سازد. این شیوه انتخاب سبب می‌شود که با گذشت زمان، تعداد کروموزوم‌های مطلوب در جمعیت افزایش یابد به طوری که میانگین مقدار برازندگی جمعیت، در مقایسه با جمعیت قبل، بیشتر می‌شود. کروموزوم‌های بهتر شانس بیشتری دارند و شانس انتخاب هر کروموزوم، متناسب با میزان برازندگی آن کروموزوم است. در این روش، احتمال انتخاب با میزان شایستگی نسبت مستقیم دارد.

(ب) **روش رتبه‌بندی**

در روش چرخ رولت، مشکلی که ممکن است به‌وجود بیاید این است که اگر برازندگی یک کروموزوم خیلی متفاوت از بقیه باشد، ممکن است شانس بسیار زیاد یا شانس بسیار کمی برای انتخاب داشته باشد. در صورتی‌که یک کروموزوم، مقدارش خیلی بیشتر از بقیه باشد، شانس انتخاب آن خیلی زیاد شده و در نتیجه شانس انتخاب بقیه خیلی کم خواهد شد. روش دیگری که این مشکل را حل کرده است، روش رتبه‌بندی است. در این روش انتخاب، کروموزوم‌ها براساس مقدار برازندگی آنها رتبه‌بندی شده و رتبه آنها به جای برازندگی آنها در چرخ رولت مورد استفاده قرار می‌گیرد. برای این منظور ابتدا کروموزوم‌ها به ترتیب از بدترین به بهترین مرتب می‌شوند، سپس کروموزوم‌ها رتبه‌بندی می‌شوند. در این حالت بدترین کروموزوم، رتبه ۱، کروموزوم ماقبل بدتری، رتبه ۲ و ... تا اینکه به بهترین کروموزوم رتبه n (تعداد کروموزوم‌های موجود در جمعیت) اختصاص داده می‌شود.

احتمال انتخاب کروموزوم i ام به صورت زیر محاسبه می‌شود:

$$P_i = \frac{Rank_i}{\frac{n(n+1)}{2}}$$

(ج) **روش انتخاب رقابتی**

در این روش یک زیر مجموعه کوچکی از کروموزوم‌ها به‌صورت تصادفی انتخاب شده و به رقابت می‌پردازند. سرانجام در این رقابت براساس میزان برازندگی، یکی از آنها پیروز شده و انتخاب می‌شود.

۳. عملگر تقاطع (ترکیب مجدد)

عملگر تقاطع، فرآیندی است که دو والد را در نظر گرفته و براساس آنها یک فرزند جدید تولید می‌شود. این فرآیند، مهمترین مشخصه الگوریتم ژنتیک به شمار می‌آید. عملگر تقاطع با یک احتمال از قبل تعیین شده بر روی کروموزوم‌های والد عمل می‌کند؛ یعنی با این احتمال، عمل تقاطع انجام می‌شود. اگر هیچ تقاطعی صورت نگیرد، فرزندان دقیقاً مشابه والدین خواهند بود. در صورتی که عمل تقاطع صورت گیرد، فرزندان از قسمت‌های مختلف کروموزوم والد، ساخته می‌شوند. این عملیات با این هدف انجام می‌شوند که کروموزوم‌های جدید در بردارنده قسمت‌های مناسب و خوب کروموزوم‌های قبلی خواهند بود. اما بهتر است همیشه بهترین کروموزوم‌های نسل قبلی بدون هیچ تغییری به نسل جدید منتقل شوند.

روش‌های مختلفی برای عملگر تقاطع وجود دارد که در زیر بعضی از آن‌ها را شرح می‌دهیم.

(آ) **تقاطع تک نقطه‌ای**

عملگر تقاطع تک نقطه‌ای، دو کروموزوم را به‌طور تصادفی از یک نقطه شکسته و بخش‌های شکسته شده دو کروموزوم را جابه‌جا می‌کند. به این ترتیب، دو کروموزوم جدید به‌دست می‌آید. به کروموزوم‌های اولیه، کروموزوم‌های ''والد'' و به کروموزوم‌های حاصل شده از عمل جابه‌جایی، کروموزوم ''فرزند'' می‌گویند.

(ب) **تقاطع یکنواخت**

یک ژن از هر دو والد به‌طور مستقل از سایر ژن‌ها، شانس برابر برای حضور در کروموزوم یک فرزند را دارند. در این حالت براساس یک توزیع تصادفی باینری مشخص می‌شود که یک ژن از کدام والد انتخاب شود [۵۷]. مثلاً اگر توزیع باینری ۱ را نشان داد، آن ژن از والد اول و اگر صفر بود از والد دوم انتخاب می‌شود. این عمل برای تمامی ژن‌های یک فرزند انجام می‌شود. در نتیجه فرزندان ترکیبی از ژن‌های والدین خواهند بود. این عملگر با هر ژن به طور مستقل رفتار می‌کند و یک انتخاب تصادفی انجام می‌دهد که طبق آن مشخص می‌شود که هر ژن باید از کدام والد به ارث برده شود. این فرآیند با تولید دنباله‌ای از L متغیر تصادفی از توزیع یکنواخت در بازه‌ی $[۰, ۱]$ پیاده سازی می‌شود.

نمی‌توان گفت که کدام یک از این عملگرها، بهترین عملکرد را در هر مسئله دارد. به هر حال، دانستن نوع تمایلی که به وسیله عملگرهای مختلف نشان داده می‌شود، می‌تواند برای طراحی الگوریتم برای یک مسئله مشخص گران‌بها باشد. براساس مطالعات تجربی، چنان‌چه طول کروموزوم‌های مسئله کوتاه باشد، بهتر است از عملگر تقاطع تک نقطه‌ای استفاده شود و چنان‌چه طول کروموزوم‌ها بلند باشد، بهتر است از عملگر تقاطع یکنواخت بهره جست.

۴. جهش

بعد از تقاطع، کروموزوم‌ها تحت اپراتور جهش قرار می‌گیرند. عملگر جهش از افتادن الگوریتم در بهینه محلی جلوگیری می‌کند. اگر عملگر تقاطع برای جست‌وجو روی جواب‌های اخیر در جهت یافتن جواب بهتر به‌کار گرفته شده است، عملگر جهش برای کمک به جست‌وجوی کل

فضای جست‌وجو در نظر گرفته می‌شود. جهش موجب می‌شود که گوناگونی جمعیت حفظ شده و ساختار ژنتیکی جدیدی در جمعیت با تغییرات تصادفی بعضی از ژن‌ها به‌وجود آید. برای یک نمایش باینری یک جهش ساده می‌تواند به‌صورت عکس مقدار هر ژن با احتمال کوچکی تعریف شود. احتمال جهش، معمولاً در حدود $1/L$ در نظر گرفته می‌شود که L طول کروموزوم است. طول یک کروموزوم برابر است با حاصل ضرب متغیرهای تصمیم در طول هر ژن. معکوس کردن[۴۴]، یکی از انواع عملگرهای جهش است و به این صورت عمل می‌کند که یک بیت از ۰ به ۱ و از ۱ به ۰ تغییر می‌کند.

۵. **جابه‌جایی**

جابه‌جایی آخرین مرحله از چرخه تولید مثل است. دو والد از یک جمعیت با اندازه ثابت، انتخاب شده و دو فرزند را به‌وجود آورده‌اند. زمانی‌که فرزندان تولید شدند، باید روشی تعریف شود که براساس آن مشخص شود که کدام یک از اعضای فعلی جمعیت باید حذف شده و چه فرزندانی باید جانشین آنها شود. این روش بر همگرایی الگوریتم ژنتیک تأثیر زیادی خواهد داشت. روش‌های مختلفی برای انتخاب جمعیت جدید وجود دارد که به‌طور مثال می‌توان از دو روش زیر نام برد.

- تمام اعضای جمعیت جدید از میان کروموزوم‌های فرزندان انتخاب شوند.
- تعدادی از افراد جمعیت مرحله بعد، همان افراد جمعیت مرحله قبل بوده و بقیه از میان فرزندان جدید انتخاب شوند. البته در هر مورد، شایسته‌ترین کروموزوم‌ها انتخاب می‌شود.

۶. **قاعده توقف**

قواعد توقف متعددی برای الگوریتم ژنتیک وجود دارد که در زیر به‌طور خلاصه به بعضی از آن‌ها اشاره می‌کنیم.

- حداکثر تولید نسل. با استفاده از این قاعده، الگوریتم ژنتیک زمانی متوقف می‌شود که تعداد مشخصی از تولید نسل اتفاق افتاده باشد. مثلاً، شمارنده تولید نسل به عدد خاصی مثل ۱۰۰ برسد.
- زمان سپری شده. هنگامی‌که فرآیند الگوریتم ژنتیک زمان خاصی را سپری کرد، الگوریتم متوقف می‌شود.
- عدم بهبود در برازندگی. در این حالت، در صورتی‌که هیچ تغییری در بهترین برازندگی جمعیت بعد از تعداد مشخصی تولید نسل به وجود نیاید، الگوریتم ژنتیک متوقف می‌شود.

[۴۴] Flipping

۳.۲.۱ مثال

حل مسئله کوله‌پشتی با استفاده از الگوریتم ژنتیک

۱) تعریف مسئله

مسئله کوله‌پشتی مسئله‌ای است که در آن یک کوهنورد در هنگام انتخاب اشیا برای کوهنوردی با آن مواجه است. کوله‌پشتی این کوهنورد فضای محدودی داشته و کوهنورد مجبور است تعداد شی محدودی را از بین n شی که در اختیار دارد، انتخاب کند. به دلیل محدودیت فضا، امکان انتخاب همه اشیا وجود ندارد. کوهنورد به دنبال این است که کدام اشیا را انتخاب کند که بیشترین ارزش را داشته باشند. این مسئله را می‌توان به‌صورت زیر فرمول‌بندی کرد:

$$Max \quad z = \sum_{i=۱}^{n} x_i v_i$$

$$\sum_{i=۱}^{n} x_i w_i \leq W$$

$$x_i \in \{۰, ۱\}, \quad i = ۱, ۲, \ldots, n$$

که در آن x_i نشان دهنده حضور یا عدم حضور شی i ام در کوله‌پشتی می‌باشد و v_i نشان دهنده ارزش وجود شی i ام در کوله می‌باشد. W محدودیت حداکثر حجم یا وزن کوله را نشان می‌دهد و w_i حجم یا وزن شی i ام . این مدل به دنبال حداکثر کردن ارزش کوله با رعایت محدودیت حجم آن می‌باشد.

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

شماره شی	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰
ارزش	۶۵	۸۵	۳۲	۹۸	۲۴	۲۶	۵۷	۴۳	۶۴	۲۱
وزن	۵	۵/۵	۲/۵	۶	۱/۵	۱/۸	۳/۵	۳	۴	۲

جدول ۱.۱: مسئله کوله‌پشتی با ۱۰ شی

۲) کد گذاری مسئله

کد گذاری باینری یک روش مناسب برای نشان دادن جواب‌های این مسئله است. برای مثال، از یک آرایه تک بعدی دارای ۱۰ ارزش به‌صورت $[x_۱, x_۲, \ldots, x_{۱۰}]$ استفاده می‌شود. در این آرایه مقادیر صفر یا یک قرار گرفته اند که نشان دهنده حضور یا عدم حضور شی مرتبط در کوله است. تعداد جواب‌ها برابر $۲^{۱۰}$ خواهد بود که تعدادی از آن‌ها امکان‌پذیر نمی‌باشند. با توجه به اینکه تعدادی از جواب‌های تولیدی امکان‌پذیر نمی‌باشند، اشیا داخل کوله براساس نسبت ارزش آنها به وزن آنها به‌صورت صعودی مرتب می‌شوند (v_i/w_i) ، سپس شروع به حذف کردن اشیا از جواب می‌شود تا به یک جواب ممکن تبدیل شود.

۳) حل مسئله کوله‌پشتی

در این مرحله، یک جمعیت اولیه با ۴ کروموزوم به‌صورت تصادفی تولید می‌شوند. نحوه کد برگردانی کدهای داخل جدول ۲.۱، به‌صورت زیر انجام می‌شود. برای مثال کروموزوم ۱ را در نظر بگیرید. مقدار ارزش این کد و وزن کوله به روش زیر محاسبه می‌شود.

$$f([۱۰۰۱۱۰۰۱۰۱]) = ۶۵ + ۹۸ + ۲۴ + ۴۳ + ۲۱ = ۲۵۱$$
$$w([۱۰۰۱۱۰۰۱۰۱]) = ۵ + ۶ + ۱/۵ + ۳ + ۲ = ۱۷/۵$$

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

جدول ۲.۱: جمعیت اولیه تولید شده

با توجه به اینکه وزن اشیا این جواب کمتر از ۲۰ می‌باشد، جواب امکان‌پذیر محسوب می‌شود و نیاز به اصلاح ندارد. احتمال انتخاب هر یک از کروموزوم‌ها به‌صورت حاصل تقسیم مقدار برازندگی آن تقسیم بر مقدار برازندگی کل کروموزوم‌های جمعیت محاسبه شده است. حال با استفاده از رویه انتخاب، کروموزوم‌ها انتخاب شده و فرزندان تولید شده در جدول ۳.۱ نمایش داده شده است. در این گام براساس چرخ رولت، کروموزوم‌های ۱ و ۲ و همچنین کروموزوم‌های ۲ و ۴ برای تولید مثل انتخاب شده‌اند. از تقاطع تک‌نقطه‌ای استفاده شده و نقطه تقاطع براساس یک قاعده تصادفی انتخاب شده است. دو کروموزوم اول در نقطه ۶ و دو کروموزوم دوم در نقطه ۴ تقاطع پیدا کرده‌اند. براساس این رویه، تقاطع ۴ کروموزوم فرزند (شماره ۵ تا ۸) تولید شد. با توجه به اینکه کروموزوم فرزند شماره ۶

والد شماره	والد آرایه	تقاطع نقطه	فرزند شماره	فرزند آرایه	وزن	امکان‌پذیری	$f(x)$
۱	[۱۰۰۱۱۰ : ۰۱۰۱]	۶	۵	[۱۰۰۱۱۰۱۰۰۰]	۱۶	OK	۲۴۴
۲	[۱۱۰۱۰۰ : ۱۰۰۰]	۶	۶	[۱۱۰۱۰۰۰۱۰۱]	۲۱/۵	NO	—
۳	[۱۱۰۱ : ۰۰۱۰۰۰]	۴	۷	[۱۱۰۱۰۱۱۰۱۰]	۲۵/۸	NO	—
۴	[۰۱۰۰ : ۰۱۱۰۱۰]	۴	۸	[۰۱۰۰۰۰۱۰۰۰]	۹	OK	۱۴۲
			۹	[۱۱۰۱۰۰۰۱۰۰]	۱۹/۵	OK	۲۹۱
			۱۰	[۰۱۰۱۰۰۱۰۱۰]	۱۹	OK	۳۰۴

جدول ۳.۱: فرزندان تولید شده در تکرار اول

فرزند شماره	فرزند آرایه	یافته جهش ژن شماره	فرزند شماره	فرزند آرایه	وزن	پذیری امکان	$f(x)$
۵	[۱۰۰۱۱۰۱۰۰۰]	۲,۵	۱۱	[۱۱۰۱۰۰۱۰۰۰]	۲۰	OK	۳۰۵
۸	[۰۱۰۰۰۰۱۰۰۰]	–	۸	[۰۱۰۰۰۰۱۰۰۰]	۹	OK	۱۴۲
۹	[۱۱۰۱۰۰۰۱۰۰]	–	۹	[۱۱۰۱۰۰۰۱۰۰]	۱۹٫۵	OK	۲۹۱
۱۰	[۰۱۰۱۰۰۱۰۱۰]	۳	۱۲	[۰۱۱۱۰۰۱۰۱۰]	۲۱٫۵	NO	—
			۱۳	[۰۱۰۱۰۰۱۰۱۰]	۱۹	OK	۳۰۴

جدول ۴.۱: فرزندان تولید شده در تکرار اول بعد از عمل جهش

و ۷ امکان‌پذیر نمی‌باشند، این کروموزوم‌ها به کروموزوم‌های ۹ و ۱۰ اصلاح شدند. در زیر نحوه اصلاح کروموزوم شماره ۶ نشان داده شده است.

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

مطابق با محاسبات فوق، رویه حذف اشیا از کروموزوم شماره ۶ به ترتیب ۴, ۲, ۸, ۱, ۱۰ صورت می‌گیرد. بر این اساس ابتدا شی شماره ۱۰ حذف می‌شود و وزن کوله به ۱۹.۵ کیلوگرم کاهش می‌یابد. با توجه به اینکه جواب امکان‌پذیر شده است، شی دیگری از کوله حذف نمی‌شود و جواب اصلاح شده است.

جدول ۴.۱ فرزندان تولید شده را بعد از عمل جهش نشان می‌دهد. در این جدول مشاهده می‌کنید که فرزندان اول و چهارم جهش پیدا کرده‌اند. در فرزند اول، دو ژن و در فرزند چهارم، یک ژن جهش یافته است. هر ژن به‌صورت تصادفی، مستقل از سایر ژن‌ها شانس جهش خواهد داشت و این عمل براساس تولید عدد تصادفی انجام می‌گیرد. برای هر ژن مستقل از سایر ژن‌ها، یک عدد تصادفی تولید و در صورتی‌که کمتر از عدد تصادفی جهش باشد، آن ژن جهش پیدا می‌کند. از بین جمعیت فرزندان و جمعیت والدین، باید ۴ عضو انتخاب شود. در این مثال فرض می‌شود که بهترین و بدترین کروموزوم به‌صورت ثابت به جمعیت بعدی منتقل شده و دو کروموزوم دیگر به‌صورت تصادفی براساس مقدار برازندگی آنها انتخاب می‌شوند. جمعیت جدید در جدول ۵.۱ نمایش داده شده است. روند فوق ادامه پیدا می‌کند تا شرط توقف الگوریتم برقرار شود.

کروموزوم شماره	آرایه	وزن	پذیری امکان	$F(x)$	احتمال
۲	[۱۱۰۱۰۰۱۰۰۰]	۲۰	OK	۳۰۵	۰٫۲۹۳
۸	[۰۱۰۰۰۰۱۰۰۰]	۹	OK	۱۴۲	۰٫۱۳۷
۱۳	[۰۱۱۱۰۰۱۰۱۰]	۱۹	OK	۳۰۴	۰٫۲۹۱
۹	[۱۱۰۱۰۰۰۱۰۰]	۱۹٫۵	OK	۲۹۱	۰٫۲۷۹

جدول ۵.۱: جمعیت به‌دست آمده بعد از تکرار اول

۳.۱ الگوریتم شبیه‌سازی تبرید

الگوریتم شبیه‌سازی تبرید[۴۵]اولین بار توسط متروپلیس[۴۶]در سال ۱۹۵۳[۴۰] برگرفته شده از مباحث ترمودینامیک آماری مطرح شد. الگوریتم شبیه‌سازی تبرید در خیلی از مواقع الگوریتم انجماد تدریجی نیز نامیده می‌شود. در این الگوریتم، از یک پارامتر دما برای کنترل الگوریتم در طول بهبود جواب مسئله استفاده می‌شود. چنان‌چه یک ماده جامد در ابتدا ذوب و سپس تا رسیدن دوباره به حالت جامد، سرد شود، خواص ساختاری آن به سرعت سرد شدن و انجمادش بستگی دارد. فرآیند ذوب اولیه و سرد کردن آهسته مواد برای رسیدن به خواص عالی آن را فرآیند تبرید[۴۷] گویند. در سال ۱۹۸۳ کریک پاتریک، گلت و وکچی[۴۸][۳۳] با شبیه‌سازی همین فرآیند روشی را برای جست‌وجوی جواب‌های یک مسئله بهینه‌سازی با هدف همگرایی به یک جواب بهینه سراسری ارائه دادند که به همین جهت این روش به شبیه‌سازی تبرید معروف شده است. در روش‌های جست‌وجوی جواب بهینه مانند روش کاهش گرادیان، تنها جواب‌هایی قابل قبول هستند که سبب بهبود تابع هدف شوند؛ اما این امکان وجود دارد که این روش‌ها به یک جواب بهینه محلی همگرا شوند. ایده ابتکاری روش شبیه‌سازی تبرید برای غلبه بر این مشکل، استفاده از نوعی جست‌وجوی تصادفی است که طی آن نه تنها جواب‌هایی که تابع هدف را بهبود می‌بخشند قابل قبول خواهند بود، بلکه تغییرات و یا جواب‌هایی که ایده‌آل نیستند نیز با در نظر گرفتن یک احتمال، امکان پذیرش را داشته باشند.

الگوریتم شبیه‌سازی تبرید، یک جست‌وجوی فراابتکاری است که در حل مسائل بهینه‌سازی گسسته اثر بخش بوده و برای حل مسائل پیوسته نیز گسترش یافته است. الگوریتم شبیه‌سازی تبرید بر مبنای دو قاعده از فیزیک آماری عمل می‌کند. قاعده اول بر این اساس است که وقتی تعادل ترمودینامیکی به یک دمای مشخص رسید، احتمال یک سیستم فیزیکی برای داشتن یک سطح انرژی E، با فاکتور بالتزمن[۴۹]، $e^{\frac{-E}{k_B T}}$، که در آن k_B توزیع بالتزمن در دمای مربوطه می‌باشد، متناسب است. براساس این قاعده، احتمال پذیرش جواب‌ها در دماهای مختلف، متفاوت بوده و متناسب با تابع معرفی شده می‌باشد. قاعده دوم، نحوه رسیدن به تعادل ترمودینامیکی در یک دمای مشخص را بیان می‌کند. براساس آن، در یک دمای مشخص، تکامل یک سیستم فیزیکی در رسیدن به تعادل ترمودینامیکی در یک دمای مشخص، شبیه‌سازی می‌شود. براساس این قاعده، برای یک جواب یک سری محدود از حرکت یا تغییرات ابتدایی در نظر گرفته می‌شود. اگر یک تغییر شکل منجر به کاهش تابع هدف یا انرژی شود، آن تغییر شکل پذیرفته می‌شود و در صورتی‌که این تغییر شکل منجر به افزایش تابع هدف یا انرژی به اندازه ΔE شود، تغییر شکل با احتمال $e^{\frac{-\Delta E}{T}}$ پذیرفته می‌شود. این پذیرش از طریق تولید یک عدد تصادفی با توزیع یکنواخت تصادفی در فاصله $[۰, ۱]$ و مقایسه آن با تابع تعریف شده انجام می‌شود. در صورتی‌که مقدار عدد به‌دست آمده کوچکتر از تابع تعریف شده باشد، تغییر شکل پذیرفته می‌شود. هر جواب از روی جواب پذیرفته شده قبلی خود تولید می‌شود. با تکرار تولید جواب‌ها و پذیرش آنها با

---

[۴۵] Simulated Annealing
[۴۶] Metropolis
[۴۷] Annealing Process
[۴۸] Kirkapatrick, Gelatt, Vecchi
[۴۹] Boltzman factor

استفاده از قاعده متروپلیس، یک توالی از جواب‌ها به‌وجود می‌آید که یک زنجیره مارکف[۵۰](در این زنجیره هر جواب وابسته به جواب پذیرفته شده قبلی خود است) را به‌وجود آورده‌اند. پایان یافتن این زنجیره با طول محدود و کافی را می‌توان به رسیدن یک سیستم فیزیکی به تعادل ترمودینامیکی در یک دمای مشخص تشبیه کرد. قاعده متروپلیس به این شرح می‌باشد:
در دمای بالا و مراحل اولیه الگوریتم مقدار $e^{\frac{-\Delta E}{T}}$ نزدیک به ۱ بوده و اکثر حرکت‌ها پذیرفته می‌شود و الگوریتم رفتاری شبیه به یک جست‌وجوی تصادفی را خواهد داشت. در دماهای پایین و اواخر الگوریتم، مقدار $e^{\frac{-\Delta E}{T}}$ به صفر نزدیک شده و جواب‌های خوب پذیرفته می‌شوند. هنگامی‌که تعادل ترمودینامیکی برقرار شد، دما به مقدار کمی کاهش یافته و یک زنجیره مارکف جدید در دمای جدید تولید می‌شود. کاهش دما منجر به تغییر توزیع بالتزمن شده و موجب می‌شود که الگوریتم به آرامی به سمت جواب‌های با مقدار تابع هدف یا سطح انرژی کمتر حرکت کند. این امر موجب می‌شود که همگرایی الگوریتم تضمین شود.

۱.۳.۱ ساختار کلی الگوریتم شبیه‌سازی تبرید

در این الگوریتم، $x_۱$ یک جواب اولیه امکان‌پذیر می‌باشد که محدودیت‌های مدل را رعایت کرده است. $f(x)$ مقدار تابع هدف به ازای x بوده و x_c یک جواب جاری در هر مرحله است. $V(x_c)$ مجموعه‌ای است که برای نشان دادن همسایه‌های x_c به‌کار می‌رود. p یک عدد تصادفی است که در زمان برخورد با جواب بدتر به‌منظور بررسی پذیرش یا عدم پذیرش آن جواب به‌کار می‌رود. در گام اول الگوریتم، یک جواب اولیه تولید و مقدار تابع هدف آن محاسبه می‌شود. در این مرحله، جواب اولیه به عنوان جواب جاری (x_c) و بهترین جواب (x^*) در نظر گرفته می‌شود. این دو جواب در طول الگوریتم تغییر خواهند کرد. مقدار تابع هدف جواب اولیه، به عنوان مقدار تابع هدف جواب جاری $(f(x_c))$ و بهترین جواب $(f(x^*))$ در نظر گرفته می‌شود. در این مرحله، دمای اولیه $T_۰$ به عنوان دمای جاری یعنی T_c در نظر گرفته می‌شود.
در گام دوم، در هر دما، یک زنجیره مارکف تولید می‌شود. در این مرحله، جست‌وجوی همسایگی تا زمانی‌که زنجیره مارکف به انتها برسد، انجام می‌شود. به این منظور، برای هر جواب جاری، یک جواب همسایه $x_{c'}$ در همسایگی آن پیدا شده و مقدار تابع هدف آن $f(x_{c'})$ محاسبه می‌شود. در صورتی‌که همسایه جدید، مقدار تابع هدف جاری را بهبود دهد یا برابر آن باشد یعنی $f(x_{c'}) \leq f(x_c)$، جواب همسایه به عنوان جواب جاری پذیرفته می‌شود و جواب جدید به عنوان جواب جاری پذیرفته می‌شود. در غیر این صورت، اگر جواب همسایه منجر به بدتر شدن تابع هدف جاری شود یعنی $f(x_{c'}) > f(x_c)$، همسایه با تابع احتمال $e^{\frac{-\Delta E}{T}}$ ارزیابی می‌شود. در این تابع، مقدار اختلاف انرژی از طریق $\Delta f = f(x_{c'}) - f(x_c)$ محاسبه می‌شود. برای این منظور، مقدار تصادفی p با توزیع یکنواخت تصادفی در فاصله $[۰, ۱]$ تولید شده و سپس با تابع احتمال مقایسه می‌شود. اگر مقدار p از تابع احتمال کمتر باشد، جواب همسایه به عنوان جواب جاری پذیرفته می‌شود. در این مرحله، جواب جاری و بهترین جواب به‌هنگام می‌شوند. به این منظور، بعد از یافتن هر جواب جدید که منجر به بهبود جواب جاری

[۵۰] Markov chain

شود، آن جواب با بهترین جواب مقایسه می‌شود. اگر این جواب از بهترین جوابی که تا به حال پیدا شده است، بهتر باشد یعنی $f(x_{c'}) \leq f(x^*)$ جایگزین جواب بهتر می‌شود. این مرحله تا زمانی ادامه دارد که زنجیره مارکف به انتها برسد و یا به عبارت دیگر، تعداد مشخصی از جست‌وجوها انجام شده باشند. معمولاً برای این زنجیره، یک طول مشخص تعیین شده و زمانی‌که تعداد جست‌وجو به طول مشخص شده برسد، زنجیره پایان می‌یابد. در طول زنجیره، دما ثابت می‌ماند.
در گام سوم، بعد از اتمام یک زنجیره، کاهش دما اتفاق خواهد افتاد. معمولاً برای کاهش دما از یک ضریب ثابت به‌نام r که $0 < r < 1$ استفاده می‌شود. رابطه کاهش دما در این حالت به‌صورت $T = r \cdot T$ تعریف می‌شود.
در گام چهارم، شرط توقف الگوریتم بررسی می‌شود. اگر شرط توقف برقرار بود، الگوریتم متوقف شده و در غیر این صورت، به گام دوم برمی‌گردد. یکی از شرایط توقف که معمولاً در این الگوریتم استفاده می‌شود این است که دمای الگوریتم به یک دمای انتهایی یعنی T_f (دمای انجماد) برسد که به‌صورت $T \leq T_f$ معرفی می‌شود.
کلیه روش‌های فراابتکاری نیاز به اثبات همگرایی دارند. الگوریتم شبیه‌سازی تبرید نیز با توجه به اینکه یک الگوریتم بر پایه جست‌وجوی تصادفی با مکانیزم‌های آماری است، نیاز به اثبات همگرایی دارد. این روش، فرآیند تبرید تدریجی را برای حل یک مسئله بهینه‌سازی، شبیه‌سازی می‌کند. تابع هدف مسئله مشابه انرژی ماده است که باید با کمک تعریف یک دمای مجازی کمینه شود. دما در این حالت یک پارامتر قابل کنترل در الگوریتم است. در سطح معینی از دما، تکرارهای زیادی انجام می‌شود. هنگامی‌که حالت تعادلی حاصل شد، دما با توجه به یک برنامه تبرید کاهش می‌یابد که این کاهش باعث می‌شود تعداد کمتری از جواب‌های نامطلوب پذیرفته شود. الگوریتم شبیه‌سازی تبرید نسبت به سایر الگوریتم‌های فراابتکاری ساده‌تر می‌باشد. این الگوریتم امکان رسیدن به نقطه‌ای نزدیک به بهینه سراسری را خواهد داشت. این نتیجه موجب تمایز این الگوریتم نسبت به سایر الگوریتم های مشابه که هنوز همگرایی آنها تضمین نشده است، می‌شود.

۲.۳.۱ برنامه انجماد

الگوریتم شبیه‌سازی تبرید دارای پارامترهای متعددی است که انتخاب صحیح آن‌ها، نقش بسیار مهمی در بهبود نتایج و همچنین زمان مورد نیاز برای اجرای آن دارد. فرآیند تعیین این پارامترها، برنامه انجماد نامیده می‌شود که طی آن پارامترهای مختلفی از قبیل دمای اولیه، طول زنجیره مارکف، قاعده کاهش دما، قاعده توقف تعیین می‌شوند. برنامه انجماد تاثیر زیادی بر همگرایی الگوریتم شبیه‌سازی تبرید دارد. برنامه انجماد نحوه کنترل دمای الگوریتم را مشخص می‌کند. طراحی مناسب پارامترهای بالا، بر الگوریتم تاثیر داشته و به نوعی یک عیب مسلم این الگوریتم شبیه به سایر الگوریتم‌های فراابتکاری می‌باشد.

- دمای اولیه: استفاده از دمای اولیه بالا موجب می‌شود که الگوریتم رفتاری شبیه به یک جست‌وجوی تصادفی به خود بگیرد و استفاده از دماهای پایین برای دمای اولیه، موجب می‌شود که الگوریتم

تبدیل به یک الگوریتم جست‌وجوی محلی شود. جست‌وجوی محلی یک روش ابتکاری است که از یک جواب اولیه شروع می‌کند و در هر تکرار، جواب فعلی را با یک همسایه جایگزین می‌کند. در این روش، جواب همسایه‌ای که جایگزین جواب فعلی می‌شود، باید تابع هدف را بهبود بخشد. جست‌وجو زمانی پایان می‌یابد که همسایه‌های کاندید بدتر از جواب فعلی باشند و این بدان معنا است که بهینه محلی یافت شده است. زمانی‌که مقدار دمای اولیه بسیار بالا در نظر گرفته شود، در مراحل اولیه الگوریتم، بسیاری از جواب‌ها پذیرفته می‌شوند که همین امر موجب طولانی شدن زمان اجرای آن می‌شود. از سوی دیگر، چنان‌چه دمای اولیه بسیار پایین در نظر گرفته شود، الگوریتم مشابه روش کاهش گرادیان عمل می‌کند و این امکان وجود دارد که تنها به یک جواب بهینه محلی دست یابیم. بنابراین درجه حرارت اولیه سیستم بایستی به گونه‌ای انتخاب شود که موازنه‌ای بین این دو حالت حدی برقرار شود. دمای اولیه برابر با $k\sigma$ براساس آزمایشات اولیه محاسبه می‌شود که در آن σ انحراف معیار توابع هدف جواب‌های به‌دست آمده در آزمایشات اولیه می‌باشد و $k = -۳/ln(p)$ است که احتمال پذیرش p مرتبط با ناحیه بیشتر از ۳σ می‌باشد [۴۴].

- طول زنجیره مارکف: برای رسیدن به یک وضعیت پایدار در هر دما، باید تعداد کافی از حرکت‌ها در نظر گرفته شود که براساس مطالعات نظری بهتر است تعداد تکرارها در هر دما براساس یک تابع نمایی از اندازه مسئله تعریف شود. تعداد تکرارها باید مطابق با اندازه مسئله متناسب با اندازه همسایگی $|N(s)|$ تعیین شود. فرض کنید f_l و f_h نشان دهنده کوچکترین و بزرگترین مقدار تابع هدف به‌دست آمده از تکرارهای انجام شده در دمای جاری باشد. در این‌صورت تعداد تکرارها برای دمای بعدی، L، از طریق رابطه زیر به‌دست می‌آید:

$$L = L_B + \lfloor L_B \cdot F \rfloor$$

که در آن $\lfloor x \rfloor$ نشان دهنده کوچکترین عدد صحیح بزرگتر از x است و L_B مقدار حد پایین برای تعداد تکرارها بوده و $F = ۱ - \exp(-(f_h - f_l)/f_h)$ می‌باشد [۶].

- قاعده کاهش دما: بین کیفیت نتایج و سرعت انجماد رابطه قوی وجود دارد. مقدار دما همواره مقداری مثبت بوده و زمانی‌که تعداد تکرارها به سمت بی‌نهایت میل کند، مقدار دما نیز به سمت صفر میل می‌کند. با رعایت این اصول، قواعدی مانند قاعده خطی، قاعده هندسی، قاعده لگاریتمی و ... برای کاهش دما وجود دارد که در زیر به بعضی از آنها اشاره می‌شود:
  - قاعده خطی: در این قاعده، مقدار دما به‌صورت خطی کاهش می‌یابد که در رابطه زیر فرمول کاهش دما به‌صورت خطی بیان می‌شود: $T_i = T_۰ - i \times \beta$ که در آن $T_۰$ دمای اولیه، i شماره مرحله کاهش دما و β ضریب ثابت بین صفر و یک می‌باشد.
  - قاعده هندسی: در این روش، کاهش دما به‌صورت زیر معرفی می‌شود: $T_{k+۱} = \alpha.T_k$ که در آن α مقداری بین صفر و یک و ضریب ثابتی می‌باشد. این قاعده به دلیل سادگی آن، بیشتر مورد استفاده قرار می‌گیرد. تجربه نشان داده است که α باید در فاصله $[۰/۵, ۰/۹۹]$ قرار داده شود.

- قاعده لگاریتمی: این قاعده نسبت به دو قاعده فوق، سرعت کاهش دمای آن آرام‌تر بوده و همگرایی بیشتری را به سمت بهینه سراسری موجب می‌شود [۱۹]: $T_i = \frac{T_\circ}{log(i)}$.

- قاعده توقف: قاعده توقف یکی از اجزای برنامه انجماد است که بر کیفیت الگوریتم تاثیر دارد. قواعد زیر نمونه‌ای از قواعد توقف رایج می‌باشند:

- رسیدن دمای الگوریتم به یک دمای انتهایی (دمای انجماد) معروف‌ترین قاعده انجماد است. برای مثال، ممکن است رسیدن دمای الگوریتم به دمای $T_f = ۰/۰۱$ به‌عنوان قاعده توقف تعیین گردد.
- در قاعده دوم، شمارنده‌ای برای شمارش تعداد تکرارها اختصاص داده می‌شود. هر گاه مقدار بهترین جوابی که توسط الگوریتم یافت شده است، بهبود پیدا کند، این شمارنده صفر می‌شود. قاعده توقف به‌صورت رسیدن شمارنده به یک حد بالایی از قبل تعیین شده تعریف می‌شود. در این حالت فرض بر این است که اگر الگوریتم تا حد معینی به جست‌وجوی خود ادامه دهد و فایده ای نداشته باشد (بهترین جواب بهبود داده نشود) بهتر است الگوریتم متوقف شود؛ زیرا احتمال یافتن نقطه‌ای بهتر از بهترین نقطه، خیلی کم است [۴۵].
- در این قاعده، برای هر حلقه (تکرارها در یک دما)، اگر تعداد حرکت‌های پذیرفته‌شده از نسبت از قبل تعیین شده‌ای کمتر باشد، یک عدد به شمارنده مرتبط اضافه می‌شود. این شمارنده تعداد حلقه‌های با بهبود کم، بعد از آخرین بهبود در بهترین جواب، را نشان می‌دهد. هر وقت که بهترین جواب، بهبود یابد، این شمارنده صفر می‌شود. به هر حال، اگر این شمارنده به مقدار از قبل تعیین شده‌ای برسد، الگوریتم متوقف می‌شود [۳۰].

۳.۳.۱ مثالی کاربردی از الگوریتم شبیه‌سازی تبرید

مسئله فروشنده دوره‌گرد (TSP)

مسئله فروشنده دوره‌گرد[۵۱]یکی از مسائلی است که بسیار مورد توجه الگوریتم‌های فراابتکاری بوده است. دلیل این موضوع این است که TSP با وجود سادگی در تعریف و مدل‌سازی، حل آن بسیار پیچیده و سخت است. بزرگترین مسئله ای که حل بهینه برای آن پیدا و اثبات شده، تنها شامل فقط چند هزار شهر است. مسئله فروشنده دوره‌گرد را به این صورت می‌توان بیان کرد که: یک فروشنده قرار است از یک شهر شروع به حرکت کرده و از تمامی شهرها عبور کند و سپس به همان شهر برگردد. هدف، یافتن یک مسیر از اولین نقطه، گذر از تمامی شهرها و سپس بازگشت به همان شهر است به طوری‌که طول مسیر حداقل شود. فروشنده باید از تمامی شهرها گذشته و از هر شهر نیز تنها یکبار عبور کند. تابع هدف، طول مسیر عبور شده می‌باشد. فاصله رفت بین شهرها، مساوی فاصله برگشت می‌باشد (ماتریس متقارن می‌باشد). برای مثال، جدول ۶.۱ طول مسافت بین شهرهای یک مسئله فروشنده دوره‌گرد را با ۶ شهر نشان می‌دهد.

[۵۱] Traveling salesman problem

نقاط	۱	۲	۳	۴	۵	۶
۱		۴۵	۶۴	۲۵	۱۴۵	۷۵
۲			۸۵	۸۵	۴۵	۷۸
۳				۹۸	۸۷	۵۴
۴					۱۳۵	۹۸
۵						۶۴
۶						

جدول ۶.۱: فاصله بین شهرها در یک مسئله TSP با ۶ شهر

حال به منظور آشنایی بیشتر با الگوریتم SA، چند مرحله از حل مسئله فوق را با استفاده از این الگوریتم ارائه می‌دهیم. ابتدا باید ساختاری برای معرفی فضا معرفی شود. بهترین ساختار برای معرفی فضای جواب این مسئله، استفاده از کدگذاری ترتیبی در یک ماتریس یک بعدی می‌باشد. این ماتریس یک بعدی، ترتیب گذر فروشنده دوره‌گرد را در طول شهرها نمایش می‌دهد. برای سادگی فرض می‌کنیم که فروشنده می‌خواهد از شهر یک شروع به حرکت کند و در نهایت به این شهر برگردد. بنابراین، ماتریس جواب را می‌توان به‌صورت $[۱, x_۱, x_۲, x_۳, x_۴, x_۵, ۱]$ تعریف کرد؛ که در آن $x_۱$ و $x_۲$ و ... به ترتیب نشان‌دهنده شهرهایی هستند که بعد از شهر ۱ به ترتیب ملاقات خواهند شد. با توجه به این‌که شهر ۱ جایگاه ثابت دارد، می‌توان برای سادگی بیشتر، آن را حذف کرد. در نتیجه ماتریس جواب به‌صورت $[x_۱, x_۲, x_۳, x_۴, x_۵]$ تبدیل می‌شود. فرض کنید الگوریتم با دمای اولیه ۱۰۰ شروع کرده و در هر دما، ۳ حرکت بررسی شود (طول زنجیره مارکف برابر ۳ فرض شود) و برای ایجاد حرکت، دو عضو مجاور به تصادف انتخاب شده و جایگاه آنها تعویض می‌شود. بعد از اتمام هر زنجیره مارکف، مقدار دما با قاعده هندسی و $\alpha = ۰/۹$ کاهش یابد. فرض کنید قاعده توقف، برابر رسیدن دمای الگوریتم به دمای انجماد $T_f = ۱$ باشد. (فرضیات ارائه شده در این قسمت، برای ایجاد سادگی در این مسئله به‌کار رفته و برای رسیدن به جواب خوب با استفاده از الگوریتم SA مناسب نمی‌باشد.)

فرض کنید جواب اولیه به‌صورت $x_۱ = [۲, ۵, ۳, ۴, ۶]$ باشد. مقدار تابع هدف این جواب برابر با $f_۱ = ۴۴۸$ می‌باشد. نقاط جاری و بهترین نیز برابر نقطه اول فرض می‌شوند:

$$x_c = [۲, ۵, ۳, ۴, ۶]\text{، } x^* = [۲, ۵, ۳, ۴, ۶]\text{، } f_c = ۴۴۸ \text{ ، } f^* = ۴۴۸ .$$

زنجیره مارکف اول $T = ۱۰۰$.

حرکت اول: فرض کنید عدد ۴ و ۵ به تصادف انتخاب شده و عضو قرار گرفته در محل ۴ با محل ۵ جابه‌جا شود. در این‌صورت، $x_{c+۱} = [۲, ۵, ۳, ۶, ۴]$ و مقدار تابع هدف آن برابر $f_{c+۱} = ۳۵۴$ به‌دست خواهد آمد. (توجه داشته باشید که در هر تکرار جابه‌جایی روی جواب x_c انجام می‌گیرد.) در نتیجه، حرکت به دلیل $\Delta f \leqslant ۰$ پذیرفته می‌شود. در نتیجه، $x_c = [۲, ۵, ۳, ۶, ۴]$ و $f_c = ۳۵۴$. همچنین، چون مقدار به‌دست آمده از مقدار بهترین کمتر است، جواب زیر را داریم:

$$x^* = [۲, ۵, ۳, ۶, ۴] \text{ و } f^* = ۳۵۴ .$$

حرکت دوم: عضو قرار گرفته در محل ۲ را با عضو قرار گرفته در محل ۳ جابه‌جا می‌کنیم. در این‌صورت، $x_{c+1} = [۲, ۳, ۵, ۶, ۴]$ و مقدار تابع هدف آن $f_{c+1} = ۴۰۴$ به‌دست خواهد آمد. چون $\Delta f > ۰$ پس یک مقدار تصادفی بین صفر و یک براساس توزیع یکنواخت تولید می‌شود $p = ۰/۶۵$ این مقدار با تابع احتمال مقایسه می‌شود. $e^{\frac{-\Delta f}{T}} = e^{\frac{-۵۰}{۱۰۰}} = ۰/۶۱$ و چون مقدار p بزرگتر از تابع احتمال است، حرکت رد می‌شود.

حرکت سوم: عضو قرار گرفته در محل ۳ را با عضو قرار گرفته در محل ۴ جابه‌جا می‌کنیم. در این‌صورت، $x_{c+1} = [۲, ۵, ۶, ۳, ۴]$ و $f_{c+1} = ۳۳۱$ به‌دست خواهد آمد و در نتیجه حرکت به دلیل $\Delta f \leqslant ۰$ پذیرفته می‌شود. در نتیجه $x_c = [۲, ۵, ۶, ۳, ۴]$ و $f_c = ۳۳۱$. همچنین با مقایسه مقدار بهترین و مقدار به‌دست آمده، بهبود حاصل می‌شود و داریم: $x^* = [۲, ۵, ۶, ۳, ۴]$ و $f^* = ۳۳۱$. با توجه به پایان یافتن حلقه داخلی در دمای ۱۰۰، یک مرحله کاهش دما رخ می‌دهد $(T = ۱۰۰ \times ۰/۹ = ۹۰)$ و دما به ۹۰ کاهش می‌یابد.

زنجیره مارکف دوم $T = ۹۰$.

حرکت اول: عضو قرار گرفته در محل ۲ را با عضو قرار گرفته در محل ۱ را جابه‌جا کرده و داریم: $x_{c+1} = [۵, ۲, ۶, ۳, ۴]$ و مقدار تابع هدف آن نیز برابر است با: $f_{c+1} = ۴۴۵$ و $\Delta f = ۱۱۴ > ۰$ است. پس یک مقدار تصادفی بین صفر و یک براساس توزیع یکنواخت تولید می‌شود $(p = ۰/۷۸)$ این مقدار با تابع احتمال مقایسه می‌شود. $e^{\frac{-\Delta f}{T}} = e^{\frac{-۱۱۴}{۹۰}} = ۰/۲۸$ و چون مقدار p بزرگتر از تابع احتمال است، حرکت رد می‌شود.

حرکت دوم: عضو قرار گرفته در محل ۲ با ۳ را جابه‌جا کرده، در این‌صورت داریم: $x_{c+1} = [۲, ۶, ۵, ۳, ۴]$ و مقدار تابع هدف آن برابر با $f_{c+1} = ۳۹۷$ به‌دست خواهد آمد. با توجه به این‌که $\Delta f = ۶۶ > ۰$ است، پس یک مقدار تصادفی بین صفر و یک براساس توزیع یکنواخت تولید می‌شود $(p = ۰/۳۴)$ این مقدار با تابع احتمال مقایسه می‌شود. $e^{\frac{-\Delta f}{T}} = e^{\frac{-۶۶}{۹۰}} = ۰/۴۸$ و چون مقدار p بزرگتر از تابع احتمال است، حرکت پذیرفته می‌شود. در نتیجه، $x_c = [۲, ۶, ۵, ۳, ۴]$ و $f_c = ۳۹۷$.

حرکت سوم

عضو قرار گرفته در محل ۴ با ۵ را جابه‌جا کرده، در این‌صورت: $x_{c+1} = [۲, ۶, ۵, ۴, ۳]$ و مقدار تابع هدف آن برابر $f_{c+1} = ۴۸۴$ به‌دست خواهد آمد. چون‌که $\Delta f = ۸۷ > ۰$ است یک مقدار تصادفی بین صفر و یک براساس توزیع یکنواخت تولید می‌شود $p = ۰/۵۳$. این مقدار با تابع احتمال مقایسه می‌شود. $e^{\frac{-\Delta f}{T}} = e^{\frac{-۱۱۴}{۹۰}} = ۰/۳۸$ و چون مقدار p بزرگتر از تابع احتمال است، حرکت رد می‌شود. با توجه به پایان یافتن حلقه داخلی در دمای ۹۰ یک مرحله کاهش دما رخ می‌دهد $(T = ۹۰ \times ۰/۹ = ۸۱)$ و دما به ۸۱ کاهش می‌یابد. سایر مراحل الگوریتم نیز طبق مراحل فوق است. چند نکته در ارتباط با روش حل بالا لازم به ذکر است: اول اینکه، با توجه به دمای اولیه تقریباً پایین الگوریتم، خیلی از جواب‌های بد در مراحل اولیه الگوریتم رد می‌شدند و بهتر است دمای اولیه افزایش یابد. دوم اینکه، به دلیل محدودیت در تعداد حرکت‌ها، الگوریتم نقاط تکراری زیادی دارد و شاید بهتر باشد جابه‌جایی دو عضو مجاور با جابه‌جایی دو عضو (بدون شرط مجاور) جایگزین شود.

۴.۱ الگوریتم بهینه‌سازی ازدحام ذرات

یکی دیگر از الگوریتم‌های فراابتکاری، الگوریتم PSO می‌باشد که مبتنی بر جمعیت است. این الگوریتم توسط کندی[۵۲] و ابرهارت[۵۳] با الهام از رفتار دسته ماهی‌ها و پرندگان در طبیعت ایجاد شده است. این الگوریتم فضای جواب را با سازگار نمودن عوامل متعدد جست‌وجو می‌کند. حرکت ذرات دارای دو مؤلفه اصلی می‌باشد.

۱. جز تصادفی

۲. جز قطعی به لحاظ سرعت و موقعیت بردار جواب

در اینجا تنوع توسط ترکیبی از بردارهای تصادفی کنترل می‌شود و تشدید نیز عمدتاً توسط حرکت قطعی نسبت به بهترین x_i^t به‌روز شده در موقعیت i و در بهترین Gbest بازنمایی می‌شود.

ویژگی‌های مختلفی در این الگوریتم یافت می‌شود؛ این الگوریتم قادر است با استفاده از برخی استراتژی‌ها در سطوح بالا و بهره‌گیری از حافظه، سرعت همگرایی را بیشتر کند و همچنین در کشف فضای جست‌وجوی کارا، بسیار مفید واقع شود. اگر مؤلفه تنوع بزرگ باشد، بخش بیشتری از فضای جست‌وجو را مورد بررسی قرار می‌دهد ولی همگرایی آهسته‌تر صورت می‌گیرد و از سوی دیگر، در سطوح بالا و با تأکید بیشتر بر تشدید، همگرایی الگوریتم به سرعت انجام می‌گیرد، اما لزوماً به مجموعه جواب‌های درست منتهی نمی‌شود. چنان‌چه یکی از اعضا توانست موقعیت بهتری نسبت به سرگروه پیدا کند او به عنوان سرگروه انتخاب می‌شود. عملکرد یک الگوریتم PSO نیز به این گونه است که دسته‌ای از ذرات (به عنوان متغیرهای مسئله بهینه‌سازی) در محیط جست‌وجو پخش می‌شوند. واضح است که بعضی از ذرات، موقعیت بهتری نسبت به ذرات دیگر خواهند داشت. در نتیجه، بر طبق رفتار ذرات هجومی بقیه ذرات سعی می‌کنند موقعیت خود را به موقعیت ذرات برتر برسانند؛ در عین حال که موقعیت ذرات برتر نیز در حال تغییر می‌باشد. تغییر موقعیت هر ذره براساس تجربه خود ذره در حرکات قبلی و تجربه ذرات همسایه صورت می‌گیرد. در واقع هر ذره از برتری یا عدم برتری خود نسبت به ذرات همسایه و همچنین نسبت به کل گروه آگاه است. برای شبیه‌سازی این رفتار، پارامترهای زیر تعریف می‌شود:

الف) Pbest : این پارامتر، بیانگر بهترین موقعیتی است که هر ذره در طول اجرای الگوریتم می‌تواند کسب کرده باشد.

ب) Gbest : این متغیر بهترین موقعیتی را که ذرات در طول اجرای الگوریتم کسب کرده‌اند را نشان می‌دهد.

ج) پارامتر شناخت فردی (c_1): این کمیت باعث می‌شود که ذره به سمت بهترین نقطه‌ای که خود و همسایه‌گانش پیدا کرده‌اند، حرکت کند. این ضریب، به عنوان ضریب تحریک به‌کار می‌رود.

[۵۲] Kennedy
[۵۳] Eberhart

د) پارامتر شناخت اجتماعی ($c_۲$): این ضریب که با عنوان ضریب تحریک نیز به‌کار می‌رود، باعث می‌شود که ذره به سمت بهترین نقطه‌ای که ذرات تا به حال کسب کرده‌اند، حرکت کند.

ه) ضریب لختی (w): این ضریب باعث ایجاد تعادل در جست‌وجوی محلی و سراسری در الگوریتم می‌شود.

و) لغزش (v): این پارامتر، تغییر موقعیت ذره در محیط جست‌وجو را نشان می‌دهد.

اکنون فرض کنید ذره j ام دارای بعد g باشد که به‌صورت زیر نشان شود:

$$x_j = [x_{j,۱}, x_{j,۲}, \ldots, x_{j,g}] \quad (۱.۱)$$

و هر ذره دارای یک Pbest و تمام ذرات دارای یک Gbest به‌صورت زیر می‌باشند:

$$Pbest_j = Pbest_{j,۱} Pbest_{j,۲} \ldots, Pbest_{j,g} \quad (۲.۱)$$

آنگاه تغییر موقعیت ذره براساس مقدار لغزش نیز به‌صورت زیر می‌باشد:

$$\begin{aligned} &v_{i,g}^{(t+۱)} = wv_i, g + c_۱ Rand()(Pbest_{i,g} - x_{i,g}^{(t)}) + c_۲ rand()(Gbest_{i,g} - x_{i,g}^{(t)}) \\ &v_{min} \leq v_{i,g}^{(t+۱)} \leq v_{max} \\ &x_{j,g}^{(t+۱)} = x_{j,g}^{(t)} + v_{i,g}^{(t+۱)} \quad j = ۱, ۲, \ldots, n \quad , g = ۱, ۲, \ldots, m \end{aligned} \quad (۳.۱)$$

مقدار x بیانگر موقعیت ذره، n تعداد ذرات گروه و m تعداد اعضای تشکیل دهنده ذره و توابع $Rand()$ و $rand()$ تولید کننده یک مقدار تصادفی بین صفر و یک می‌باشند. در این روابط باید دقت کرد که بزرگ بودن v_{max} ممکن است باعث شود که ذرات از روی نقطه مینیمم عبور کنند و کوچک بودن آن نیز باعث می‌شود که ذره، حول موقعیت خود به چرخش در آمده و قادر به جست‌وجو در فضای آزمون نشود. مقدار v_{max} معمولاً بین ۱۰٪ تا ۲۰٪ محدوده متغیرها انتخاب می‌شود. از طرف دیگر انتخاب مناسب w باعث تکرار کمتر الگوریتم برای رسیدن به نقطه بهینه می‌شود. در الگوریتم‌های معمولی PSO ضریب w از مقدار ۰٫۹ تا مقدار ۰٫۴ در طول اجرای الگوریتم و براساس رابطه زیر کاهش می‌یابد:

$$w = w_{max} - \frac{w_{max} - w_{min}}{iter_{max}} * iter \quad (۴.۱)$$

از دیگر مشکلات در اجرای این الگوریتم انتخاب مناسب $c_۱$ و $c_۲$ است. در بسیاری از الگوریتم‌ها، مقادیر $c_۱$ و $c_۲$ به گونه‌ای انتخاب می‌شوند که $c_۱ + c_۲ \leq ۴$ باشد.

مراحل اجرای الگوریتم PSO

۱) مقدار $t = ۱$ را اختیار کنید.

۲) تولید بردار x_j^t به‌صورت تصادفی در محدوده مورد نظر.

۳) تعیین مقدار برازندگی $(f(x_j))$ و انتساب این مقدار متغیر به $Pbest_j^t$ برای $j = ۱, ۲, \ldots, n$

۴) مقدار $Gbest^t$ را به‌صورت زیر تعیین کنید:

$$Gbest^t = min(f(Pbest_j^t))$$

۵) مقدار $v_{j,g}^{(t+۱)}$ را از رابطه (۳.۱) محاسبه کنید.

۶) شرط زیر را بررسی کنید:

$$\text{اگر} \quad v_{j,g}^{(t+۱)} \geq v_{j,g}^{max} \quad \text{آنگاه} \quad v_{j,g}^{(t+۱)} = v_{j,g}^{max}$$

۷) مقدار $x_{j,g}^{(t+۱)}$ را از رابطه (۳.۱) محاسبه کنید.

۸) تغییر $Pbest$ را برای $j = ۱, ۲, \ldots, n$ براساس دستورات زیر انجام دهید:

$$\text{اگر} \quad f(Pbest_j) \geq f(x_{j,g}^{(t+۱)}) \quad \text{آنگاه} \quad Pbest_{j,g} = x_{j,g}^{(t+۱)}$$

۹) تغییر $Gbest$ را برای $j = ۱, ۲, \ldots, n$ براساس دستورات زیر انجام دهید:

$$\text{اگر} \quad f(Gbest_j) \geq min(f(Pbest_j)) \quad \text{آنگاه} \quad Gbest = Pbest_j$$

۱۰) مقدار $t = t + ۱$ را اختیار کنید.

۱۱) بررسی شرط پایان مراحل و تعیین x بهینه؛ اگر $t \neq t_{max}$ به گام ۶ بروید و در غیر این صورت مقدار $Gbest$ را در x بهینه قرار بدهید.

فصل ۲

الگوریتم جست‌وجوی هارمونی در مسائل بهینه‌سازی تک‌هدفه

۱.۲ مقدمه

آیا تا به‌حال هنگام گوش دادن به قطعه‌ای زیبا از موسیقی کلاسیک، به این فکر کرده‌اید که بین پخش موسیقی و پیداکردن جواب مناسب برای یک مسئله پیچیده در مباحثی مانند طراحی شبکه آب و یا دیگر مسائل بهینه‌سازی ارتباطی وجود دارد؟ در سالیان اخیر، محققان در اکثر مسائل پیچیده بهینه‌سازی با پیاده‌سازی روش‌های فراابتکاری به نتایج مناسبی دست یافته‌اند.

الگوریتم جست‌وجوی هارمونی(HS[۱]) یکی از ساده ترین و جدیدترین روش‌های فراابتکاری است که در فرآیند جست‌وجوی جواب شدنی بهینه در مسائل بهینه‌سازی از فرآیند موسیقی الهام گرفته است. به‌عبارت دیگر، میان پیدا کردن یک جواب بهینه در مسئله پیچیده و فرآیند اجرای موسیقی تشابه وجود دارد. این روش را ابتدا گیم[۲]و همکاران در سال ۲۰۰۱ [۲۲] میلادی ارائه کردند. مطابق با این روش فراابتکاری، تلاش برای یافتن هارمونی و هم‌آهنگی در موسیقی، مشابه پیدا کردن جواب بهینه در مسائل بهینه‌سازی می‌باشد. هارمونی در موسیقی اصطلاحاً به اجرای نت‌های متفاوت به‌صورت هم‌زمان می‌گویند که در نهایت تبدیل به آهنگی زیبا و موزون از نظر شنیداری می‌شود. در حقیقت می‌خواهیم نشان دهیم بین گوش دادن موسیقی و لذت بردن از آن با پیدا کردن جواب بهینه برای یک مسئله ارتباط

[۱] Harmony Search
[۲] Geem

زیادی وجود دارد. هدف این است که به‌صورت تخصصی مباحث مربوط به موسیقی که می‌تواند مصداقی برای طراحی یک الگوریتم فراابتکاری در راستای دست‌یابی به جواب بهینه یا نزدیک به بهینه باشد را بررسی کنیم. شباهتی نسبی بین موسیقی و مسئله بهینه‌سازی وجود دارد؛ بدین صورت که می‌توان ابزارها و آلات موسیقی را با متغیرهای تصمیم در مسئله بهینه‌سازی، نت‌های موسیقی را با مقدار متغیر و یک قطعه موسیقی کامل را با بردار جواب مسئله متناظر دانست. تلاش برای یافتن این هارمونی و هم‌آهنگی در موسیقی، مثل پیداکردن شرایط بهینه در فرآیند بهینه‌سازی می‌باشد. به عبارت دیگر، خلاقیت یک نوازنده موسیقی برای ایجاد یک قطعه هنری را می‌توان با فرآیند جست‌وجو در یک مسئله بهینه‌سازی مقایسه نمود و از طرف دیگر، می‌توان با استانداردهای زیبایی‌شناسی صوتی، هارمونی کاملاً ایده‌آلی را طراحی کرد، چرا که یک موسیقی‌دان همیشه در نظر دارد در قطعه موسیقی‌ای که می‌نوازد، تمام هارمونی‌ها رعایت شده باشند. مطالب این فصل از پایان‌نامه از مرجع [۵] جمع آوری شده است.

الگوریتم جست‌وجوی هارمونی مانند ژنتیک جزء الگوریتم‌های بهبود دهنده است که مبتنی بر جمعیت است. به‌عبارت دیگر، با نسلی از بردارهای جواب شروع و برای ایجاد نسل‌های جدید از فرآیند انتخاب استفاده می‌شود اما برخلاف الگوریتم ژنتیک، در این روش از همه‌ی بردارهای جواب موجود در حافظه برای تولید جواب جدید استفاده می‌شود. از مزایای دیگر این الگوریتم، همگرایی سریع آن به دلیل ساختار مناسب آن است که در مدت زمان مناسبی فضاهای جواب را با محدوده عملکرد بهتری شناسایی می‌کند. این ویژگی در صورتی‌که مسئله مورد مطالعه از بهینگی محلی برخوردار باشد، دچار مشکل می‌شود و در بهینگی محلی متوقف شده و نمی‌تواند به بهینگی سراسری برسد. دلیل این مشکل عدم کارایی مناسب الگوریتم در جست‌وجوی محلی در مسائل بهینه‌سازی گسسته است. به منظور رفع این مشکل و تطبیق روش حل با مسئله، محققان با تغییر در پارامترهای الگوریتم، انواع مختلفی از این الگوریتم ارائه کردند تا دقت حل و نرخ همگرایی افزایش یابد. PAR[۳](نرخ تنظیم گام) و HMCR[۴] (نرخ بررسی حافظه هارمونی) دو پارامتری هستند که نقش اساسی در تنظیم صحیح بردارهای جواب جدید دارند که در ادامه بیشتر معرفی می‌گردند. به منظور توضیح الگوریتم جست‌وجوی هارمونی لازم است تا روند ایجاد یک قطعه موسیقی ایده‌آل را توسط یک موسیقی‌دان حرفه‌ای بررسی کنیم. وقتی یک موسیقی‌دان در حال آفرینش یک قطعه موسیقی است، سه گزینه را در پیش روی خود دارد:

۱. نواختن یک نغمه و موسیقی معروف دقیقاً به همان ترتیبی که در ذهن موسیقی‌دان است.

۲. نواختن آهنگی شبیه موسیقی معروف، با تغییر دادن جزئی آن چیزی که وجود دارد.

۳. ایجاد یک قطعه موسیقی کاملاً جدید با نت‌هایی جدید.

شباهت بین کار نوازندگان در موسیقی و عملکرد روش‌های بهینه‌یابی را می‌توان به‌صورت جدول ۱.۲ خلاصه کرد. شکل و فرم دادن قطعی و مشخص به سه گزینه فوق، فرآیندی بهینه‌سازی است که گیم و همکاران در سال ۲۰۰۱ [۲۲] با وجود سه مؤلفه متناظر زیر ایجاد کردند: استفاده از حافظه هارمونی[۵]،

[۳] Pitch Adjusting Rate
[۴] Harmony Memory Considering Rate
[۵] Harmony Memory (HM)

موسیقی	بهینه‌سازی
ساز	متغیر تصمیم
دامنه نت	دامنه متغیر تصمیم
نت انتخاب شده	مقدار متغیر تصمیم
هارمونی	بردار جواب
علم زیبا شناختی	تابع هدف
تمرین	تکرار
تجربه	حافظه هارمونی

جدول ۱.۲: مقایسه بین بهینه‌سازی و بداهه‌گویی موسیقی

تنظیم گام صدا[۶] و انتخاب تصادفی[۷].

استفاده و کاربرد حافظه هارمونی بسیار مهم است، زیرا مشخص می‌کند که آیا جواب‌ها و هارمونی‌های جدید می‌توانند قابل قبول باشند یا خیر. به منظور استفاده مؤثر از این حافظه، پارامتری برای نشان دادن احتمال انتخاب از حافظه هارمونی با عنوان HMCR ارائه می‌شود. این پارامتر احتمال پذیرش از حافظه هارمونی است که با توجه به نرخ و میزان آن مورد بررسی قرار می‌گیرد. اگر این نرخ خیلی پایین و نزدیک به صفر باشد، تنها چند جواب خوب انتخاب شده و همگرایی ممکن است کند باشد (تعداد اندکی از متغیرهای برگزیده برای هارمونی که در حافظه هارمونی الگوریتم قرار دارند، انتخاب می‌شوند) اما اگر این نرخ نزدیک به یک باشد، احتمال انتخاب از حافظه هارمونی زیاد و انتخاب تصادفی در کل فضای جست‌وجو کم خواهد شد. در هر دو صورت، الگوریتم به مقادیر بهینه همگرا نشده یا عمل بهینه‌سازی و همگرایی بسیار کند صورت می‌گیرد. به منظور ایجاد تعادل در استفاده از متغیرها و جواب‌های با کیفیت بالاتر موجود در حافظه هارمونی و جست‌وجوی تصادفی در کل فضای جست‌وجو، مقدار مطلوب برای این پارامتر را حدود ۰/۷ تا ۰/۹ در نظر می‌گیرند.

دومین عملگر، تنظیم گام صدا است که دارای پارامترهای پهنای باند[۸](ماکزیمم مقدار تغییر ایجاد شده در متغیر انتخاب شده) برای تعیین مقدار مجاز تغییر توسط عملگر تغییر گام و میزان احتمال تنظیم و تغییر گام PAR می‌باشد. این دو پارامتر در ایجاد یک هارمونی (بردار جواب) جدید مورد استفاده قرار می‌گیرند. تطبیق گام مشابه با اپراتور جهش در الگوریتم ژنتیک است. انتخاب تصادفی در الگوریتم جست‌وجوی هارمونی به افزایش تنوع جواب‌ها کمک می‌کند. استفاده از روش انتخاب تصادفی، می‌تواند الگوریتم را به سوی کشف جواب‌های گوناگون‌تر هدایت کند و این روند می‌تواند ادامه یابد تا بهینگی به‌دست آید. بررسی حافظه مهم می‌باشد چون اطمینان می‌دهد که جواب‌های خوب به‌عنوان مؤلفه‌های جواب‌های جدید در نظر گرفته می‌شوند.

گام صدا هم نقشی مشابه ایفا می‌کند، اما تغییرات را در ناحیه خاصی محدود می‌کند و با جست‌وجوی

---

[۶] Pitch Adjusting
[۷] Randomization
[۸] Band Width (BW)

محلی تطبیق می‌دهد.

۱.۱.۲ مراحل الگوریتم جست‌وجوی هارمونی

در الگوریتم جست‌وجوی هارمونی هر جواب یک هارمونی نامیده می‌شود و به صورت یک بردار n بعدی معرفی می‌شود. یک جمعیت اولیه ابتدا به صورت تصادفی ایجاد شده و در حافظه‌ی الگوریتم ذخیره می‌شود. سپس یک بردار جواب جدید به صورت تصادفی ایجاد می‌شود. در نهایت بردار جواب ایجاد شده با بدترین بردار جواب موجود در حافظه مقایسه و در صورت بهتر بودن با بدترین بردار جواب جابه‌جا می‌شود و به این ترتیب حافظه‌ی هارمونی به روز می‌شود. این فرآیند تا برقرار شدن شرط توقف ادامه می‌یابد. در ادامه به توضیح مفصل این مراحل می‌پردازیم.

۱. **تعریف تابع هدف و پارامترهای الگوریتم**

برای شروع حل با این الگوریتم، تابع برازش و پارامترهای الگوریتم مشخص می‌شوند. سپس به تعریف پارامترهای HMCR و PAR می‌پردازیم. این دو پارامتر بین صفر و یک هستند. مقدار HMCR حدود ۰٫۷ تا ۰٫۹ اختیار می‌شود که بدین معنی است که احتمال انتخاب از حافظه هارمونی برابر با ۰٫۷ تا ۰٫۹ است. مثلاً اگر HMCR برابر با ۰٫۸۵ باشد، الگوریتم با احتمال ۸۵ درصد، بردار جدید را از بین بردارهای مرتب‌شده در حافظه هارمونی انتخاب می‌کند و با احتمال ۱۵ درصد، بردار جدید را به‌صورت تصادفی در محدوده مجاز انتخاب می‌کند. پارامتر PAR مقداری در حدود ۰٫۱ تا ۰٫۵ به خود می‌گیرد به این معنی که احتمال تغییر جزیی مقدار انتخاب شده از حافظه هارمونی به اندازه ۰٫۱ تا ۰٫۵ می‌باشد. برای هر بردار جدید به‌دست آمده باید امتحان شود که آیا لازم است تنظیم گام بر روی آن انجام شود یا خیر. اگر جواب مثبت بود که با احتمال PAR همسایه‌های بالا و پایین نقطه به‌دست آمده را مورد بررسی قرار می‌دهند در غیر این صورت با احتمال $PAR - ۱$ کاری انجام نمی‌دهد. برای تعیین مقدار تغییر بر روی متغیر انتخاب شده از حافظه ماتریس پارامتر دیگری به نام BW نیز تعریف می‌شود.

۲. **ایجاد ماتریس حافظه**

در مرحله دوم، یک ماتریس حافظه هارمونی از چندین جواب (هارمونی) تشکیل می‌شود. شکل کلی این ماتریس حافظه به‌صورت رابطه (۱.۲) است:

$$HM = \begin{bmatrix} x_1^1 & \dots & x_n^1 \\ \vdots & \ddots & \vdots \\ x_1^{HMS} & \dots & x_n^{HMS} \end{bmatrix}_{HMS \times n} \qquad (۱.۲)$$

تعداد ردیف‌ها در این ماتریس حافظه به تعداد HMS[۹] (اندازه حافظه هارمونی) می‌باشد که هر ردیف به معنی یک جواب مسئله است که به صورت تصادفی تولید شده‌اند. در این ماتریس تعداد متغیرها n می‌باشد. این ماتریس حکم موسیقی‌های معروفی را دارد که یک موسیقی‌دان قصد

[۹] Harmony Memory Size

دارد دقیقاً مثل آن‌ها بنوازد. هر ردیف در حکم یک هارمونی یا قطعه موسیقی است که به احتمال زیاد هیچ یک از این هارمونی‌ها بهترین جواب ممکن نمی‌باشند و با استفاده از یک تابع برازش، مقدار برازش یا ارزش هر کدام از این جواب‌ها را به‌دست می‌آوریم. در این مرحله به تعداد HMS بردار جواب در حافظه هارمونی قرار داده می‌شود.

$$\begin{bmatrix} x_1^1 & x_2^1 & \dots & x_{n-1}^1 & x_n^1 & \rightarrow & f(x^1) \\ x_1^2 & x_2^2 & \dots & x_{n-1}^2 & x_n^2 & \rightarrow & f(x^2) \\ \vdots & \dots & \dots & \dots & \dots & \rightarrow & \vdots \\ x_1^{HMS-1} & x_2^{HMS-1} & \dots & x_{n-1}^{HMS-1} & x_n^{HMS-1} & \rightarrow & f(x^{HMS-1}) \\ x_1^{HMS} & x_2^{HMS} & \dots & x_{n-1}^{HMS} & x_n^{HMS} & \rightarrow & f(x^{HMS}) \end{bmatrix}$$

۳. **تولید یک هارمونی یا جواب جدید**

برای ایجاد یک هارمونی یا جواب جدید مانند الگوریتم ژنتیک به چند عملگر احتیاج داریم و تک‌تک به ایجاد متغیرها می‌پردازیم. برای ایجاد مقدار برای متغیر i ام، ابتدا یک عدد تصادفی بین صفر و یک تولید می‌کنیم، این عدد تصادفی با HMCR مقایسه می‌شود. اگر از آن کوچکتر باشد، یک مقدار برای متغیر i ام از ماتریس حافظه و از ستون i ام انتخاب می‌شود. در غیر این صورت، یک مقدار تصادفی از فضای جست‌وجو برای متغیر i ام انتخاب می‌شود. در صورتی‌که از ماتریس حافظه یک مقدار انتخاب شد، سپس عدد تصادفی دیگری تولید می‌شود و با PAR مقایسه می‌شود. در صورتی که عدد تصادفی از PAR کوچکتر باشد، این مقدار انتخاب شده از ماتریس حافظه به مقدار کوچکی با توجه به رابطه (۲.۲) تغییر پیدا می‌کند. برای تعیین مقدار تغییر بر روی متغیر انتخاب شده از حافظه ماتریس، پارامتر دیگری به‌نام BW تعریف می‌شود که با توجه به رابطه (۲.۲) مقدار متغیر جدید به‌دست می‌آید:

$$X_{new} = X_{old} + BW \times \varepsilon \qquad (۲.۲)$$

در اینجا X_{old} به‌عنوان متغیر ذخیره شده در حافظه هارمونی است و X_{new} به‌عنوان متغیر جدید بعد از عملیات تنظیمی و تغییری ظاهر می‌شود. در واقع ε همان عدد تصادفی از یک توزیع یکنواخت در $[-۱, ۱]$ است. به بیان واضح‌تر می‌توان این تنظیم گام را به نوعی شبیه عملگر جهش در الگوریتم ژنتیک دانست.

۴. **به‌روز رسانی حافظه**

به همین صورت تمام متغیرهای یک جواب یا هارمونی تولید می‌شود و سپس ارزش آن هارمونی با توجه به تابع برازش محاسبه شده و با بدترین هارمونی موجود در حافظه ماتریس مقایسه می‌شود. اگر بردار جواب جدید از بدترین بردار جواب در حافظه ماتریس بهتر باشد، هارمونی جدید جایگزین هارمونی قبلی می‌شود.

۵. **شرط توقف**

در الگوریتم جست‌وجوی هارمونی تعداد تکرار[۱۰] (جست‌وجو) به‌عنوان شرط توقف در نظر گرفته

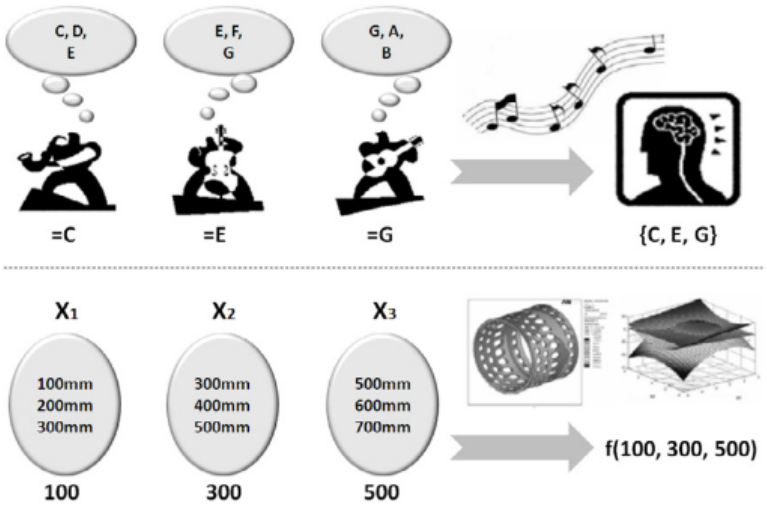
[۱۰] Number Iteration (NI)

می‌شود که پارامتر مربوط به آن در مرحله اول مقداردهی می‌شود. اگر تعداد هارمونی‌های ایجاد شده به NI برسد محاسبه‌ها متوقف شده در غیر این صورت مراحل سوم تا پنجم تکرار می‌شود. از دیگر شروط، تغییر نکردن بهترین هارمونی موجود در ماتریس حافظه هارمونی است. با تکرار این پنج مرحله، الگوریتم به مرور به مقادیر بهینه نزدیک‌تر شده و جواب‌های مسئله بهبود می‌یابند.

۲.۱.۲ بداهه‌گویی موسیقی و بهینه‌سازی

روش ابتکاری جست‌وجوی هارمونی یک الگوریتم بهینه‌سازی نوظهور الهام گرفته از اصول اساسی بداهه‌گویی[۱۱]موسیقی (آفرینش یک قطعه موسیقی) می‌باشد. زمانی‌که موسیقی‌دان‌ها یک هارمونی را می‌سازند، معمولاً ترکیبات آزمایشی متعددی را در حافظه خود ذخیره می‌کنند. فرآیند جست‌وجوی جواب‌های بهینه برای مهندسی مسائل مشابه با این جست‌جوی مؤثر برای یک حالت کاملی از هارمونی می‌باشد [۷۳].

شکل ۱.۲ ساختاری از حافظه‌ی هارمونی را نشان می‌دهد، که هسته‌ی الگوریتم جست‌وجوی هارمونی



شکل ۱.۲: ساختار حافظه‌ی هارمونی و تناسب بین بداهه گویی موسیقی و بهینه سازی.

بوده و مقایسه‌ای بین بداهه‌گویی موسیقی و بهینه‌سازی می‌باشد. بر این اساس، یک گروه سه نفره جاز متشکل از ساکسیفون، باس دوگانه و یک گیتار را در نظر بگیرید. ضربات مورد نظری در حافظه هر موسیقی‌دان قراردارد: نوازنده‌ی ساکسیفون $\{C, D, E\}$؛ باس دوگانه $\{E, F, G\}$؛ گیتارزن $\{G, A, B\}$. اگر نوازنده‌ی ساکسیفون به‌طور تصادفی $\{C\}$ را از حافظه‌اش انتخاب کند ، باس دوگانه $\{E\}$ را انتخاب کند و گیتار زن $\{G\}$ را انتخاب کند، هارمونی جدید $\{C, E, G\}$ ساخته می‌شود. اگر این هارمونی از بدترین هارمونی موجود در حافظه هارمونی بهتر باشد، هارمونی جدید جایگزین هارمونی قبلی می‌شود. این فرآیند تا هارمونی کامل به‌دست آید، ادامه می‌یابد. در یک مسئله بهینه‌سازی واقعی، هر موسیقی‌دان با یک متغیر تصمیم و قطعات مورد نظر با مقادیر متغیر مورد نظر جایگزین می‌شوند. اگر هر متغیر

[۱۱] Improvisation

نشان‌دهنده‌ی قطر یک لوله بین دو گره در یک شبکه‌ی توزیع آب باشد، هر متغیر دارای تعداد مشخصی از قطرهای مطلوب می‌شود.

اگر اولین متغیر یعنی $x_۱$، $\{۱۰۰mm\}$ را از مجموعه $\{۱۰۰mm, ۲۰۰mm, ۳۰۰mm\}$ انتخاب کند و دومین متغیر یعنی $x_۲$ ، $\{۳۰۰mm\}$ را از $\{۳۰۰mm, ۴۰۰mm, ۵۰۰mm\}$ و سومین متغیر $x_۳$ ، $\{۵۰۰mm\}$ را از $\{۵۰۰mm, ۶۰۰mm, ۷۰۰mm\}$ انتخاب کند، این مقادیر بردار جواب جدیدی به‌صورت $\{۱۰۰mm, ۳۰۰mm, ۵۰۰mm\}$ می‌سازند. اگر این بردار جواب از بدترین بردار در حافظه هارمونی بهتر باشد، بردار جدید با بدترین بردار موجود در حافظه هارمونی جایگزین می‌شود. این فرآیند تا زمان رسیدن به محک توقف برای پایان الگوریتم یا یافتن بهینه‌ی سراسری ادامه می‌یابد.

شبه‌کد الگوریتم جست‌وجوی هارمونی

۱) تعریف تابع هدف: $f(x) = f(x_۱, x_۲, x_۳, \ldots, x_n)$.

۲) تعریف نرخ (احتمال) انتخاب از حافظه هارمونی (HMCR)، احتمال تنظیم گام (PAR)، مقدار پهنای باند (BW) ، تعداد تکرارها (NI) و اندازه حافظه هارمونی (HMS).

۳) تولید حافظه هارمونی به‌صورت تصادفی.

۴) به تعداد تکرارها دستورات زیر را تکرار کنید:

برای هر متغیر دو دستور زیر را انجام بدهید:

اگر rand < HMCR آنگاه یک مقدار از ستون i ام ماتریس HM به متغیر i تخصیص بدهید. در غیر این صورت مقداری تصادفی از فضای جست‌وجو به متغیر i تخصیص بدهید. (rand عددی تصادفی بین ۰ و ۱ است.)

اگر rand < PAR آن‌گاه مقدار متغیر انتخاب شده را اندکی تغییر بدهید. (rand عددی تصادفی بین ۰ و ۱ است.)

۵) جواب تولید شده با تابع برازش ارزیابی گردد و مقداری عددی به جواب (هارمونی) جدید تخصیص داده شود.

۶) اگر هارمونی جدید بهتر از بدترین هارمونی در حافظه هارمونی باشد، هارمونی جدید جایگزین بدترین هارمونی در حافظه هارمونی می‌شود.

۷) پایان.

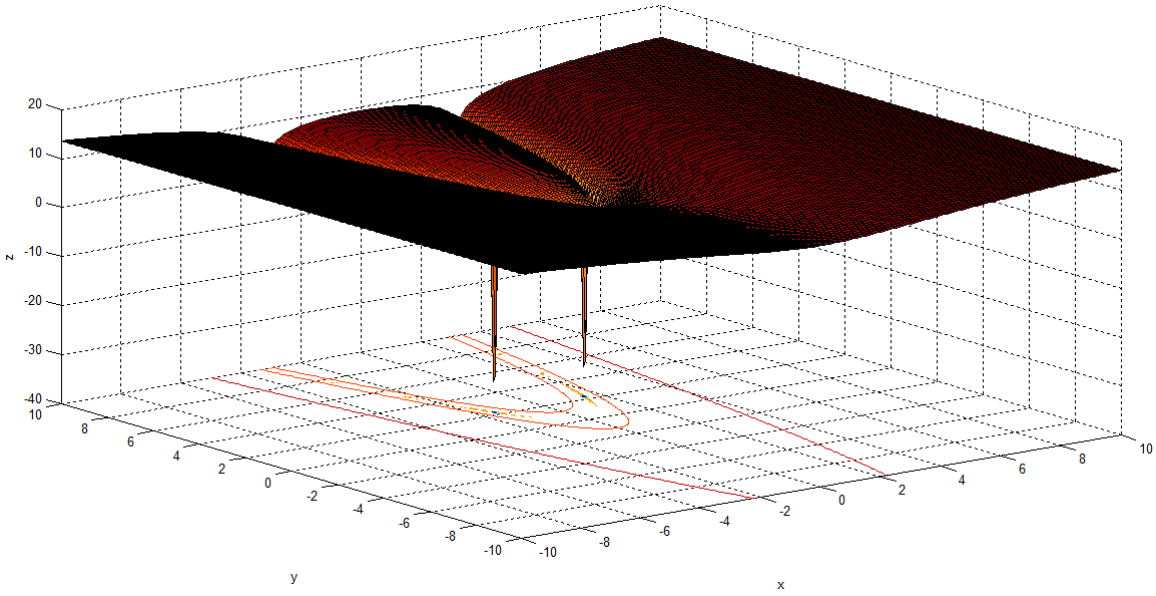
بخش‌های ارائه شده را می‌توان به راحتی و با استفاده از هر زبان برنامه‌نویسی اجرا کرد؛ با این وجود برای دست‌یابی به جواب‌های ساده‌تر، نرم‌افزار متلب در این زمینه کارایی بالاتری دارد. در ادامه با ارائه مثالی مقدار بهینه تابع روزنبراک[۱۲] را به‌دست خواهیم آورد.

مثال ۱.۱.۲. لگاریتم تابع روزنبراک

$$\begin{cases} F(x,y) = \ln[۱ + (۱ - x)^۲ + ۱۰۰(y - x^۲)^۲] \\ s.t. \quad (x,y) \in [-۱۰, ۱۰] \times [-۱۰, ۱۰] \end{cases} \tag{۳.۲}$$

این تابع در سال ۱۹۶۰ میلادی توسط روزنبراک ارائه شد. در بهینه‌سازی ریاضی، این تابع یک تابع غیرمحدب است که برای ارزیابی قدرت همگرایی الگوریتم‌های بهینه‌سازی به‌کار می‌رود. به علت حالت

[۱۲] Rosenbrock Function



شکل ۲.۲: نمایی از لگاریتم تابع روزنبراک

غیر متقارن و شکل دره پیچشی که در شکل ۲.۲ این تابع وجود دارد، یکی از بهترین توابع برای تست مدل‌های ارائه شده می‌باشد، چرا که مدل‌های مبتنی بر گرادیان به احتمال فراوان به مقدار زیادی تکرار برای رسیدن به مینیمم سراسری نیاز دارند. مقدار مینیمم این تابع توسط الگوریتم جست‌وجوی هارمونی با شرایط اولیه زیر به‌دست آمده است. $HMS = ۲۰$، $HMCR = ۰/۹۵$، $PAR = ۰/۷$، $BW = ۱$. بعد از ۱۰۰ تکرار مقدار بهینه این تابع $-۱۰۰۵۳/۳۴۸$ به‌دست آمد. (کد متلب این مثال در ضمیمه پایان‌نامه آورده شده است.)

۲.۲ توسعه الگوریتم جست‌وجوی هارمونی

از زمانی‌که مفاهیم اولیه الگوریتم توسط گیم در سال ۲۰۰۱ [۲۲] به‌وجود آمد، این الگوریتم در طیف متنوعی از مسائل کاربردی به‌کارگرفته شد. به‌عنوان مثال از طراحی سازه برای حل پازل سودوکو گرفته تا ترکیب‌های موسیقی و تصویربرداری پزشکی و مباحث مطرح در حوزه مهندسی صنایع. از زمان پیدایش الگوریتم فراابتکاری جست‌وجوی هارمونی، محققان در پی کشف روش‌هایی هستند تا بتوانند الگوریتم را در حل مسائل مختلف و جدید آسان‌تر کنند. بهبودها و تغییرات صورت گرفته در الگوریتم هارمونی به‌منظور حل گونه‌ای از مسائل زیست محیطی و مرتبط با حفظ منابع طبیعی در ایالت اورگن در کشور آمریکا صورت گرفت. مهدوی و همکاران [۳۸] پیشرفت‌های چشم‌گیری را در الگوریتم جست‌وجوی هارمونی بهبودیافته (IHS [۱۳]) ایجاد کردند که در مسائل بعدی مطرح‌شده بسیار کارا ظاهر گردید. IHS مشابه HS است با این تفاوت که مقادیر پارامترهای PAR و BW به‌صورت پویا در هر تکرار

[۱۳] Improved Harmony Search

جداگانه مطابق روابط (۶.۲)ـ(۴.۲) به‌دست می‌آیند [۱]. در اینجا به برخی از اصلاحات صورت گرفته بر روی الگوریتم اشاره می‌شود:

۱) **بهبودها و اصلاحات صورت گرفته بر روی حافظه هارمونی**

الف) دگرکتین[۱۴] در سال ۲۰۰۸ [۱۵] در تولید حافظه هارمونی به‌جای اینکه تمام اعضای حافظه را تصادفاً تولید کند، به اندازه دو برابر HMS جواب تصادفی تولید کرد و به تعداد HMS از بهترین‌های آن‌ها را در حافظه هارمونی قرار داد.

ب) مهدوی و همکاران [۳۹] به‌جای پارامترهای ثابت، از پارامترهای متغیر برای تنظیم گام صدا استفاده کردند.

$$PAR(j) = PAR_{min} + (PAR_{max} - PAR_{min}) * (\frac{j}{NI}) \qquad (۴.۲)$$

$$BW(j) = BW_{max} exp[ln(\frac{BW_{min}}{BW_{max}})(\frac{j}{NI})] \qquad (۵.۲)$$

$$j = ۱, \ldots, NI. \qquad (۶.۲)$$

که در آن :

PAR_{max} : حداکثر احتمال عملگر تغییر گام.

PAR_{min} : حداقل احتمال عملگر تغییر گام.

BW_{max} : حداکثر پهنای باند.

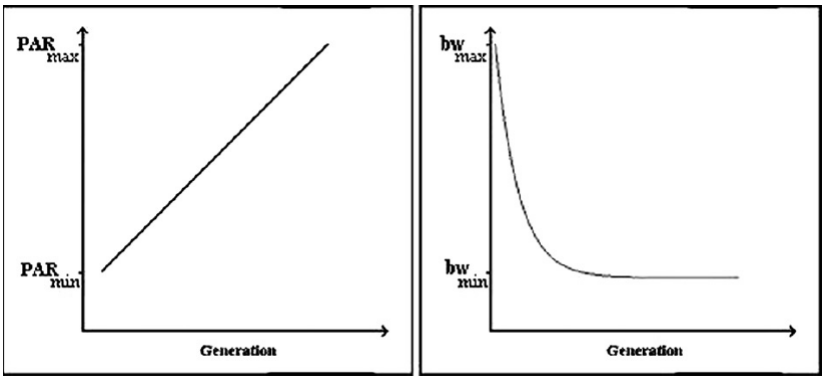
BW_{min} : حداقل پهنای باند.

NI : تعداد کل تکرارهای الگوریتم.

j : شماره تکراری است که الگوریتم در آن قرار دارد.

در این روابط مقدار PAR در ابتدای الگوریتم کم بوده ولی مقدار تغییر متغیر انتخابی (BW) زیاد است. هر چقدر الگوریتم پیش می‌رود مقدار عبارت $(\frac{j}{NI})$ به عدد ۱ نزدیک شده و در انتهای الگوریتم مقدار PAR زیاد می‌شود و به حداکثر مقدار خود می‌رسد و مقدار تغییر متغیر انتخابی BW کم‌شده و این بهبود، مقدار تنوع و تشدید را بسیار بهتر کنترل می‌کند. نقش پارامتر PAR در افزایش تنوع در شروع جست‌وجوی الگوریتم در فضای جواب و نقش BW در جست‌وجوی محلی الگوریتم به منظور افزایش نرخ همگرایی تأثیر گذار است. اگر PAR مقدارش کوچک و BW مقدارش بزرگ باشد، عملکرد الگوریتم ضعیف است و باید برای رفع آن تعداد تکرارها (NI) را افزایش دهیم تا به جواب بهینه دست یابیم. هر چه مقدار پارامتر BW در تکرارهای ابتدایی بزرگتر باشد، موجب می‌شود که الگوریتم، تنوع جست‌وجو در کل فضای جواب را افزایش دهد و در تکرارهای بعدی، به منظور جست‌وجوی محلی مقدار کمتر آن مناسب است. بنابراین مقادیر بزرگ PAR و مقادیر کوچک BW در تکرارهای پایانی منجر به رسیدن به فضای بهینه و همگرایی به بهینه می‌گردد [۱].

[۱۴] Degertekin



شکل ۳.۲: نحوه تغییر پارامترهای PAR و bw.

۲) **بهبودها و اصلاحات صورت گرفته برای ساخت هارمونی جدید**

لی[۱۵] و همکاران در سال ۲۰۰۷ [۳۵] با الهام از الگوریتم ژنتیک، جهش غیریکنواخت را به‌عنوان عملگر جدید تنظیم گام معرفی کردند:

$$x'_{new,i} = \begin{cases} x_{new,i} + \Delta(j, x_i^U - x_{new,i}) & if \quad r_1 \leq 0.5 \\ x_{new,i} - \Delta(j, x_{new,i} - x_i^L) & if \quad r_1 > 0.5 \end{cases} \qquad (۷.۲)$$

که در آن Δ عملگری برای تکرار شماره j و y عبارت است از:

$$\Delta(j, y) = y * r_2 * (1 - (\frac{j}{NI}))^c$$

و c عددی ثابت و $r_1, r_2 \in [0, 1]$ عددهای تصادفی می‌باشند. جهش غیر یکنواخت به این صورت است که یک مقدار کوچک به مقدار فعلی ژن اضافه می‌شود. این مقدار اضافه‌شده، به‌طور تصادفی از یک توزیع نرمال[۱۶] با میانگین صفر و انحراف معیار از پیش تعیین شده به‌دست می‌آید و سپس مقدار به‌دست آمده اگر لازم باشد به دامنه $[L_i, U_i]$ محدود می‌شود. توزیع نرمال دارای این ویژگی است که اکثر تغییرهای ایجاد شده کوچک خواهد بود و این به‌دلیل این است که دو سر انتهایی توزیع، هیچ‌گاه به صفر نمی‌رسند.

۳) **بهبودها و اصلاحات صورت گرفته برای کار با محدودیت‌های مسئله در حین ساختن یک هارمونی جدید**

اردال[۱۷] و همکاران در سال ۲۰۰۸ [۱۷] پس از تولید یک هارمونی جدید از دو روش برای به‌کار بردن محدودیت‌ها استفاده کردند:

روش اول: اگر هارمونی جدید محدودیت‌ها را نقض کرد، می‌توان آن را کنار گذاشت و از آن صرفنظر کرد.

---

[۱۵] Li

[۱۶] Normal distribution

[۱۷] Erdal

روش دوم: اگر مقدار نقض هارمونی جدید از محدودیت‌ها کوچک باشد، این جواب جدید به حافظه هارمونی راه می‌یابد، اما خطای قابل قبول با افزایش تکرارها کاهش می‌یابد.

گائو[۱۸] در سال ۲۰۰۸ [۲۰] روشی ارائه داد که در آن یک هارمونی جدید فقط زمانی به داخل حافظه هارمونی راه می‌یابد که شامل ۳ شرط زیر باشد:

الف) هنگامی‌که هارمونی جدید، از بدترین هارمونی موجود در حافظه هارمونی بهتر باشد.
ب) تعداد هارمونی‌های مشابه با هارمونی تولیدشده در حافظه هارمونی از یک مقدار بحرانی کمتر باشد.
ج) مقدار برازش هارمونی جدید از مقدار میانگین برازش هارمونی‌های مشابه موجود در حافظه هارمونی بهتر باشد.

۴) **بهبودها و اصلاحات صورت گرفته برای شرط لازم پایان دادن به الگوریتم**

چنگ[۱۹] و همکاران در سال ۲۰۰۷ [۷] فقط زمانی تکرارها را خاتمه می‌دادند که مقدار تغییرات بهترین مقدار تابع هدف کمتر از یک مقدار کوچک باشد.

دو مورد از بهبود صورت گرفته بر روی الگوریتم جست‌وجوی هارمونی برای مواجهه با مسائل بهینه‌سازی دارای چند اکسترمم و مسائل دارای محدودیت می‌باشد. حالت ساده و پایه ای الگوریتم در مواجهه با این مسائل دچار ضعف است که برای این نواقص دو الگوریتم هارمونی اصلاح‌شده پیشنهاد می‌شود. اولین الگوریتم اصلاح‌شده با بهره‌گیری از مدیریت حافظه جدید، مسائل دارای چند اکسترمم را حل نموده و دومین الگوریتم اصلاح‌شده با استفاده از تکنیک تسلط پارتو[۲۰] برای حل مسائل دارای محدودیت ارائه می‌شود.

۱.۲.۲ الگوریتم هارمونی اصلاح‌شده برای بهینه‌سازی توابع دارای چند اکسترمم محلی

تعیین جواب بهینه سراسری در مسائل دارای چند اکسترمم با استفاده از الگوریتم جست‌وجوی هارمونی بسیار دشوار می‌باشد؛ زیرا عضوهای حافظه هارمونی بعد از یک یا چند تکرار بدون بهبود، ایستا می‌شوند. بنابراین کلید به‌کارگیری الگوریتم جست‌وجوی هارمونی در بهینه‌سازی مسائل دارای چند اکسترمم حفظ تنوع اعضای حافظه هارمونی است. در واقع کیفیت جواب‌های کاندید به‌عنوان یک عضو جدید حافظه هارمونی تنها بر مبنای برازندگی آن استوار نیست بلکه باید عضوهای موجود در آن تشابه داشته باشد. فرض کنید شکل اعضای جاری حافظه هارمونی به‌صورت $f_i; i = ۱, ۲, \ldots, HMS$ نمایش داده شود. بعد از اینکه یک جواب جدید از مجموعه $(x'_۱, x'_۲, \ldots, x'_n)$ به شکل f' به‌دست آمد، تمام اعضای حافظه هارمونی در بازه $d_i; i = ۱, ۲, \ldots, HMS$ مورد سنجش قرار می‌گیرند:

$$d_i = \left\|(x'_۱, x'_۲, \ldots, x'_n) - (x^i_۱, x^i_۲, \ldots, x^i_n)\right\| \qquad (۸.۲)$$

[۱۸] Gao
[۱۹] Cheng
[۲۰] Pareto Dominance

علامت نُرم در اینجا نماد استاندارد فاصله برای بازه انتخاب شده است. فرض کنید M تعداد اعضای حافظه هارمونی باشد که در همسایگی V از $(x'_1, x'_2, \ldots, x'_n)$ قرار دارد. به عبارت دیگر فقط اعضایی که دارای فاصله کوچک‌تر از V می‌باشند، به حساب می‌آیند. میانگین اعضای حافظه هارمونی به‌صورت زیر محاسبه می‌شود:

$$\bar{F} = \frac{\sum_{i=1}^{M} f_i}{M} \qquad (۹.۲)$$

بنابراین $(x'_1, x'_2, \ldots, x'_n)$ تحت شرایط زیر جایگزین بدترین عضو حافظه هارمونی خواهد شد:

۱. f' بهتر از بدترین عضو HM باشد.

۲. مقدار M از مقدار تعیین‌شده M_v کوچکتر باشد.

۳. f' از $\bar{F}$ بهتر باشد.

روش فوق باعث می‌شود که تنوع جواب‌های حافظه هارمونی حفظ شود و گزینه‌های بهتری برای مسائل دارای چند بهینه محلی پیدا شود. اما این روش نیز دو مشکل اساسی دارد: اولاً پارامترهای V و M_v همیشه به‌صورت وابسته ظاهر می‌شوند و معمولاً براساس روش آزمون و خطا انتخاب می‌شوند. V و M_v می‌توانند بر روی عملکرد بهینه‌سازی توسط روش هارمونی اصلاح‌شده تأثیرگذار باشند. در مواردی که V بسیار کوچک و M_v ثابت باشد، رفتار الگوریتم جست‌وجوی هارمونی و الگوریتم هارمونی اصلاح‌شده شبیه به‌هم است. عملکرد روش الگوریتم جست‌وجوی هارمونی اصلاح‌شده می‌تواند در مواردی که V خیلی بزرگ است، بدتر شود. متأسفانه همانند سایر پارامترهای الگوریتم چگونگی انتخاب بهترین V و M_v هنوز یک مشکل حل نشده است، اگرچه بعضی استراتژی‌های انطباقی می‌توانند جواب‌های موثری باشند. دوماً در بازه‌ی $(x'_1, x'_2, \ldots, x'_n)$ تمام اعضای حافظه هارمونی باید محاسبه شوند. این شرایط در مواردی که اندازه حافظه هارمونی بزرگ است، باعث زمان‌بر شدن حل مسئله می‌شود.

۲.۲.۲ الگوریتم هارمونی اصلاح‌شده برای مسائل بهینه‌سازی دارای محدودیت

تعداد زیادی از مسائل بهینه‌سازی کاربردی مربوط به مسائل بهینه‌سازی با محدودیت است و هدف آن کشف جواب بهینه درون بازه از پیش تعیین شده است به‌گونه‌ای که محدودیت‌ها را برآورده نماید. یک مسئله بهینه‌سازی با محدودیت به شکل رابطه زیر نمایش داده می‌شود:

$$\begin{aligned}
&min \quad f(\vec{\mathbf{x}}) \\
&s.t: \\
&g_i(\vec{\mathbf{x}}) \leq 0; i = 1, 2, \ldots, I \\
&h_j(\vec{\mathbf{x}}) = 0; j = 1, 2, \ldots, J \\
&, \vec{\mathbf{x}} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)
\end{aligned}$$

که در آن $f(\vec{\mathbf{x}})$ تابع هدف، $g_i(\vec{\mathbf{x}})$ محدودیت‌های نامساوی، $h_j(\vec{\mathbf{x}}) = ۰$ محدودیت‌های تساوی می‌باشند. معمولاً سر و کار داشتن با مسائل بهینه‌سازی با محدودیت دشوار است، زیرا محدودیت‌ها در تمامی فضای جست‌وجو به‌صورت بخش‌های مجزا تقسیم می‌شوند. روش‌های متعددی برای بررسی محدودیت‌ها ارائه شده است که یکی از بهترین روش‌ها استفاده از تابع جریمه[۲۱] است. بدین منظور تابع برازش جدید $F(x)$ تعریف می‌شود که حاصل تلفیق تابع برازش اصلی $f(x)$ و چندین تابع جریمه $p(x)$ است. در واقع این توابع جریمه هر کدام مقدار تخطی[۲۲] از محدودیت‌ها را بیان می‌کنند. هر کدام از محدودیت‌ها که اهمیت بیشتری برای ما داشته باشد، توسط تابع جریمه با وزن بیشتر کنترل می‌شود. مثلاً اگر هزینه برآورده نشدن محدودیت اول بسیار بیشتر از هزینه برآورده نشدن محدودیت سوم باشد، تابع جریمه مربوط به محدودیت اول وزن بالاتری خواهد داشت. شکل کلی تابع برازش جدید به‌صورت معادله زیر خواهد بود:

$$F(x) = f(x) + \sum_{i=۱}^{m} w_i p_i(x) \quad i = ۱, \ldots, m \tag{۱۰.۲}$$

در معادله بالا، w_i ها وزن هر کدام از توابع جریمه می‌باشند. اگر محدودیتی برآورده نشود، این عدم برآورده در تابع برازش ظاهر می‌شود تا الگوریتم بعد از چندین تکرار به سمت برآورده این محدودیت‌ها پیش رود. برای تبدیل محدودیت‌ها به توابع جریمه، ابتدا محدودیت‌ها را به‌صورت استاندارد در می‌آوریم. بنابراین دو دسته محدودیت وجود خواهد داشت. در دسته اول که شامل محدودیت‌های کوچکتر یا مساوی است، تابع جریمه را به‌صورت زیر تعریف می‌کنیم:

$$p(x) = max[۰, g(x)]$$

و در دسته دوم، برای محدودیت‌های مساوی، داریم:

$$p(x) = max[۰, |h(x)|]$$

در معادلات بالا، $g(x)$ برابر با مقدار تخطی محدودیت از مقدار تعیین شده است. به عنوان مثال، اگر محدودیتی مانند (۱۱.۲) داشته باشیم،

$$g(x) = {x_۱}^۲ + ۴x_۲ + ۱ \leq ۰ \tag{۱۱.۲}$$

در این حالت، اگر مقدار محدودیت $-۰/۵$ شود، محدودیت برآورده شده و مقدار $p(x)$، صفر خواهد شد. به‌عنوان مثال، اگر مقدار محدودیت برابر با ۱/۱ شود، مقدار تخطی از محدودیت برابر با ۱/۱ ، بوده و مقدار $p(x)$ برابر با ۱/۱ خواهد بود. در محدودیت مساوی، چون تخطی چه مثبت و چه در حالت منفی وجود دارد، از تابع قدرمطلق برای تابع جریمه استفاده می‌شود.

حال روش دیگری برای حل مسائل بهینه‌سازی دارای محدودیت را بررسی می‌کنیم. در این روش، تمامی اعضای حافظه هارمونی مورد استفاده قرار می‌گیرند؛ حتی آنهایی که محدودیت‌ها را برآورده نمی‌کنند.

[۲۱] Penalty Function
[۲۲] Violation

در اینجا مقادیر توابع محدودیت از اعضای حافظه هارمونی به همراه مقادیر تابع هدف در حافظه هارمونی ذخیره شده‌اند. اعضای شدنی[۲۳]حافظه هارمونی را با توجه به مقادیر تابع هدفشان در حافظه هارمونی می‌توان ذخیره کرد. با این‌حال برای عضوهای نشدنی[۲۴]، رتبه‌بندی اعضای حافظه هارمونی براساس روش تسلط پارتو[۲۵] است. در تعریف ۵.۲.۳ از فصل ۳ مفهوم تسلط پارتو آورده شده است.

۳.۲.۲ الگوریتم جست‌وجوی هارمونی اجتماعی

همان‌طور که قبلاً بیان شد، دو مؤلفه مهم در الگوریتم‌های فراابتکاری، تنوع و تشدید می‌باشد که الگوریتم جست‌وجوی هارمونی برای اعمال تشدید در جست‌وجوهای خود از قانون در نظرگیری حافظه استفاده می‌کند. همچنین این الگوریتم برای اعمال تنوع از قوانین انتخاب تصادفی و تنظیم گام بهره می‌برد. الگوریتم با استفاده بهینه از این دو قانون خواهد توانست در تکرارهای کمتر به جواب‌های بهتری برای مسئله بهینه‌یابی مورد نظر دست یابد. قانون در نظرگیری توسط پارامتر HMCR در این الگوریتم اعمال می‌شود. برای هر متغیر با احتمال HMCR یکی از مقادیر ذخیره شده در حافظه هارمونی که متناظر با آن متغیر می‌باشد، را برای آن انتخاب می‌کند. اگر فقط از قانون در نظرگیری حافظه برای تولید بردار جواب‌های جدید استفاده کنیم، آنگاه جواب‌های تولیدی مناسب نخواهد بود (مثلاً الگوریتم همگرایی نارس دارد) زیرا الگوریتم همواره از ترکیب یک مجموعه اعداد که در حافظه هارمونی ذخیره شده‌اند برای تولید بردار جواب‌های جدید استفاده می‌کند. در واقع تنوعی که از ساختار الگوریتم ایجاد می‌شود، فقط بر روی اعداد ذخیره شده در حافظه اعمال می‌شود. برای رفع این مشکل، الگوریتم قانون تنظیم گام را به‌کار می‌گیرد.

قانون تنظیم گام توسط پارامتر PAR در الگوریتم اعمال می‌شود. این قانون الگوریتم را مجبور می‌کند با احتمال PAR بر روی مقادیری که از حافظه انتخاب شده‌اند، عمل تنظیم گام را انجام دهد؛ یعنی به‌جای اینکه همان مقدار انتخاب شده از حافظه را برای آن متغیر انتخاب کنیم، یک مقدار در همسایگی آن مقدار را برای آن متغیر انتخاب کنیم. این عمل باعث می‌شود که مقادیر جدیدی برای متغیرهای تصمیم وارد حافظه هارمونی شود. ضعف الگوریتم جست‌وجوی هارمونی در این است که همواره یک مقدار ثابت به مقادیر انتخاب شده از حافظه، اضافه یا کم می‌کند که این عمل چندان مناسب نیست؛ زیرا نیاز داریم در تکرارهای ابتدایی مقدار زیادی به مقادیر انتخاب شده از حافظه بیفزاییم و در تکرارهای انتهایی مقدار کمی را به این مقادیر از حافظه بیفزاییم یا از آن کم کنیم. با این عمل یک تنوع بهینه در الگوریتم ایجاد خواهد شد که در تکرارهای ابتدایی مقدار این تنوع زیاد و در تکرارهای انتهایی، کم است.

در این قسمت الگوریتم جست‌وجوی هارمونی اجتماعی[۲۶] را به عنوان روشی جدید برای بهبود عملکرد الگوریتم جست‌وجوی هارمونی معرفی می‌کنیم. فرق مهم و اساسی الگوریتم جست‌وجوی هارمونی اجتماعی با الگوریتم جست‌وجوی هارمونی تنها در قسمت قانون تنظیم گام است. در روش جدید سعی بر این است که در تکرارهای کمتر به جواب‌های بهتری برای مسائل مختلف بهینه‌یابی دست پیدا کنیم.

---

[۲۳] اعضایی که تمام محدودیت‌ها را برآورده می‌کنند.

[۲۴] عضوهایی که حداقل یکی از محدودیت‌ها را برآورده نمی‌کنند.

[۲۵] Pareto dominance

[۲۶] Social harmony search algorithm

در روش جدید با استفاده از خاصیت تابع توزیع نرمال، قانون تنظیم گام را اعمال می‌کنیم. دانشمندان اغلب برای مدل کردن متغیرهای تصادفی که با رفتار آنها آشنایی ندارند، از این تابع استفاده می‌کنند. برای یافتن جواب‌های بهتر حول جواب‌های ذخیره در حافظه، از این تابع استفاده شده است. به دلیل اینکه در مسائل بهینه‌یابی در دنیای اطراف ما، شناخت زیادی نسبت به متغیرهای آنها نداریم، این تابع می‌تواند سودمند باشد. تابع چگالی احتمال به صورت زیر تعریف می‌شود:

$$f(x) = \frac{۱}{\sigma\sqrt{۲\pi}} \exp(-\frac{(x-\mu)^۲}{۲\sigma^۲}) \qquad (۱۲.۲)$$

اگر در فرمول بالا، $\sigma = ۱$ و $\mu = ۱$ باشد، به آن تابع توزیع نرمال استاندارد گویند. از آنجایی که به ازای مقادیر مختلف انحراف معیار، تابع توزیع نرمال شکل‌های مختلفی به خود می‌گیرد، این تابع به عنوان یک عملگر برای تنظیم نت‌های موجود در حافظه مورد استفاده قرار می‌گیرد.
الگوریتم جست‌وجوی هارمونی اجتماعی مقادیر انتخاب شده از حافظه را براساس $N(x_i', \sigma_i')$ تنظیم می‌کند و $N(x_i', \sigma_i')$ نشان‌دهنده یک عدد تصادفی است که به وسیله توزیع نرمال با میانه x_i' و واریانس σ_i' تولید شده است. x_i' مقدار انتخاب شده برای متغیر تصمیم i م از درون حافظه هارمونی می‌باشد. σ_i' مقدار واریانس متغیر i ام در درون حافظه است. واریانس با استفاده از فرمول زیر تعریف می‌شود:

$$\sigma_i' = \xi \times \sum_{j=۱}^{HMS} \frac{|x_i' - x_i{}^j|}{HMS - ۱} \quad ; \quad ۱ \le i \le N \qquad (۱۳.۲)$$

که در آن HMS تعداد بردارهای ذخیره شده در حافظه هارمونی را معرفی می‌کند و $x_i{}^j$ مقدار متغیر i ام در بردار جواب j ام است که در حافظه هارمونی ذخیره شده است. ξ یک مقدار ثابت است که وظیفه تنظیم کردن واریانس را بر عهده دارد و بین ۱ تا ۳ می‌تواند انتخاب شود. در روش جدید σ_i' همانند BW عمل می‌کند، با این تفاوت که مقدار آن برای هر متغیر تصمیم در هر تکرار با توجه به مقادیر ذخیره شده برای همان متغیر تصمیم در درون حافظه به‌روز رسانی می‌شود. از دیگر ویژگی‌های روش جدید، می‌توان به سادگی آن اشاره کرد. در این روش به تعیین پارامترهای زیادی در ابتدا شروع عملیات بهینه‌یابی نیاز نداریم و این ویژگی برتری این الگوریتم نسبت به الگوریتم جست‌وجوی هارمونی اصلاحی ارائه شده توسط مهدوی و همکاران [۳۹] می‌باشد. از این جهت نام این الگوریتم، جست‌وجوی هارمونی اجتماعی گذاشته شده است که الگوریتم جدید برای تعیین σ_i' برای هر متغیر، از تمام مقادیر موجود در حافظه هارمونی استفاده می‌کند و مقدار این پارامتر را براساس یک تجربه کاملاً اجتماعی در هر مرحله به‌روز رسانی می‌کند. براساس نتایج به‌دست آمده، در این الگوریتم همواره مقادیر $HMCR = ۰/۹۹$ و $PAR = ۱$ می‌باشند و دیگر نیازی به انجام آنالیز حساسیت بر روی این دو پارامتر نمی‌باشد.
در تمام مسائل بهینه‌یابی مقدار احتمال اعمال قانون انتخاب تصادفی برابر با ۱ درصد در نظر گرفته می‌شود؛ چون‌که قانون تصادفی یک قانون کاملاً تصادفی است و به عنوان یک عملگر تنوع عمل می‌کند و از آنجایی‌که الگوریتم جست‌وجوی هارمونی خود دارای چندین عملگر تنوع است، قانون تنظیم گام که به عنوان عملگر تنوع عمل می‌کند بهبود یافته است و دیگر نیازی به قانون انتخاب تصادفی نخواهد بود. اگر قانون تنظیم گام را با احتمال بیشتری اعمال کنیم، جست‌وجوی الگوریتم حول مقادیر خوب افزایش خواهد یافت که این باعث می‌شود الگوریتم در تکرارهای کمتر به جواب‌های بهتری برسد به

همین دلیل مقدار $PAR = ۱$ در نظر گرفته شده است. شاید اگر در الگوریتم ژنتیک مقدار جهش را ۱ در نظر بگیریم، الگوریتم به تله بیافتد و به همگرایی نارس برسد ولی در الگوریتم جست‌وجوی هارمونی به دلیل ساختار الگوریتم در تولید بردار جواب جدید (یعنی استفاده از بردارهای جواب موجود در حافظه) انتخاب مقدار ۱ برای پارامتر PAR نه تنها مشکلی ایجاد نمی‌کند بلکه در همگرایی الگوریتم به سمت جواب بهینه تأثیر مثبت هم می‌گذارد.

فرض کنید عمل تنظیم نت بر روی اعدادی که با استفاده از قانون تصادفی تولید می‌شوند، اعمال شود. آنگاه به جای اینکه جست‌وجوی الگوریتم حول یک سری جواب‌هایی که در حافظه ذخیره شده‌اند انجام گیرد، بر روی یک سری مقادیری که به‌صورت کاملاً تصادفی انتخاب می‌شوند، انجام می‌گیرد. این عمل باعث می‌شود که سرعت همگرایی الگوریتم کاهش یابد و جواب نهایی نیز، جواب بهینه مطلق نباشد. از این جهت می‌توان به ارزش قانون در نظر گیری حافظه پی برد، زیرا این قانون این اطمینان را ایجاد می‌کند که الگوریتم در نواحی که مستعدتر هستند، جست‌وجوی بیشتری انجام می‌دهد و این بسیار مطلوب می‌باشد. الگوریتم جست‌وجوی هارمونی اجتماعی تمام مراحل الگوریتم جست‌وجوی هارمونی سنتی را دارا است با این تفاوت که قانون تنظیم گام آن به‌صورت زیر عمل می‌کند:

تا زمانی‌که $i <= n$ مراحل زیر را تکرار کن:
۱) اگر $rand <= HMCR$. آنگاه: $x'_i = x^j_i$
که j توزیع یکنواخت تصادفی در بازه $(۱, ..., HMS)$ است. (بررسی حافظه)
۲) اگر $rand <= PAR$ آنگاه $x'_i = N(x'_i, \sigma'_i)$ (تنظیم گام).
۳) در غیر این صورت $x'_i = LB_i + rand * (UB_i - LB_i)$ (انتخاب تصادفی).

به‌طوری‌که $rand$ یک عدد تصادفی، n تعداد متغیرهای تصمیم، HMS تعداد جواب‌های ذخیره شده در حافظه هارمونی، LB کران پایین دامنه هر متغیر تصمیم و UB کران بالای دامنه هر متغیر تصمیم است. در واقع الگوریتم فوق بیان می‌کند که مقداری که برای متغیرهای تصمیم، با استفاده از قانون تنظیم گام، به وسیله توزیع نرمال حاصل می‌شود، ممکن است در دامنه مجاز متغیر مورد نظر نباشد. در این صورت الگوریتم همان مقداری که از حافظه انتخاب شده است را برای متغیر تصمیم مورد نظر در نظر خواهد گرفت. این عمل باعث افزایش تشدید در الگوریتم خواهد شد.

۴.۲.۲ حل مسائل مختلف با استفاده از الگوریتم جست‌وجوی هارمونی

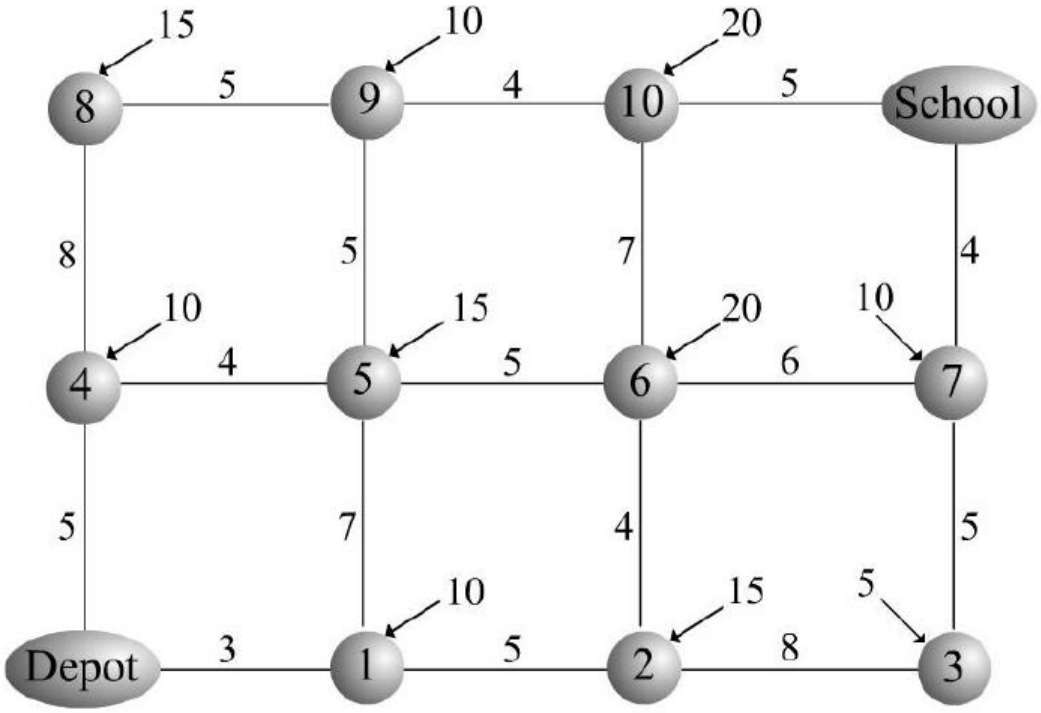
حل مسئله مسیریابی وسایل نقلیه

مسئله مسیریابی وسایل نقلیه (VRP[۲۷]) یک مسئله بهینه‌سازی ترکیبی[۲۸] است که در جست‌وجوی انتخاب کردن، تحویل دادن یا سهولت در سرویس‌دهی به مشتری‌ها در یک منطقه با ناوگان وسایل نقلیه توزیع شده است. در یک حالت خاص مسئله هر وسیله نقلیه یک گنجایش مشخص و هر مشتری یک تقاضای معلوم یک پایانه و یک ماتریس مسافت (طول، قیمت، زمان) در میان مشتریان وجود دارد.

[۲۷] Vehicle Routing Problem

[۲۸] هنگامی‌که متغیرهای تصمیم در مسائل بهینه‌سازی گسسته باشند، مسئله‌ی یافتن جواب‌های بهینه، بهینه‌سازی ترکیبی نامیده می‌شود

هدف VRP بهینه کردن مسیر وسیله نقلیه است که این بهینه‌شدن از طریق کمترین مسافت یا کمترین تعداد وسایل نقلیه حاصل شود. مسئله تعیین مسیر اتوبوس مدرسه (SBRP[۲۹]) یک مورد ویژه از مسئله از VRP است. از دیدگاه مدرسه هدف این مسئله فراهم آوردن سرویس حمل و نقل مناسب و کافی برای دانش‌آموزان است. شبکه این مسئله شامل یک پایانه اتوبوس، یک مدرسه و ۱۰ ایستگاه اتوبوس است که در شکل ۴.۲ نمایش داده شده است. هر ایستگاه تعداد معینی دانش‌آموز دارد و زمان سفر بین دو ایستگاه به دقیقه است.



شکل ۴.۲: فرم شماتیک شبکه مسیریابی حرکت اتوبوس مدرسه.

هدف این مسئله مینیمم کردن تعداد کل و زمان حرکت اتوبوس‌ها است. الگوریتم جست‌وجوی هارمونی حالت بهینه سراسری را بعد از ۱۰۰۰ بار ارزیابی تابع برازش توانست پیدا کند. برای ارزیابی هرچه بهتر الگوریتم، نتایج حاصل از این الگوریتم با الگوریتم ژنتیک مورد مقایسه قرار گرفت. در این مقایسه تعداد ارزیابی‌های توابع ۱۰۰۰ و تعداد اجرای دو الگوریتم ۲۰ بار در نظر گرفته شده است. در الگوریتم هارمونی، پارامترهای PAR و HMCR و HMS در هر بار اجرا متفاوت هستند. در الگوریتم ژنتیک تعداد ارزیابی توابع ۱۰۰۰ (اندازه جمعیت در تعداد نسل‌ها) و تعداد ۲۰ اجرا با مقادیر جمعیت جهش متفاوت در هر بار اجرای الگوریتم است. هر دو الگوریتم می‌توانند جواب بهینه را کشف کنند و در ۲۰ بار اجرای دو الگوریتم، الگوریتم جست‌وجوی هارمونی توانایی رسیدن به دو مرتبه مقدار بهینه را در ۲۰ بار اجرا دارد در حالی‌که الگوریتم ژنتیک در طی ۲۰ مرتبه اجرا، فقط یک‌بار به مقدار بهینه می‌رسد. الگوریتم جست‌وجوی هارمونی فرآیند جست‌وجو را کمی سریع‌تر از ژنتیک انجام می‌دهد. مقایسه هزینه‌های صورت‌گرفته توسط این دو الگوریتم نشان می‌دهد که به‌طور متوسط الگوریتم هارمونی ۳۳۹٫۸ دلار و الگوریتم ژنتیک ۴۰۹٫۵ دلار هزینه در بردارد.

[۲۹] School Bus Routing Problem

مسئله انتخاب بهینه سبد سهام

مسئله انتخاب بهینه سبد سهام یکی از مباحث روز حوزه مالی به شمار می‌رود. مارکویتز با ارائه مقالاتی این مدل را به‌صورت چندهدفه و با محدودیت‌های اندازه ثابت تعداد سهام و حد بالا و پایین درصد هر سهم حل کرده است. مارکویتز در مدل خود به این نکته اشاره داشت که هر سرمایه گذار سعی در افزایش بازگشت سرمایه و کاهش ریسک خرید سهام دارد. وی بازگشت سرمایه را به صورت میانگین بازگشت سهام و ریسک را به‌صورت واریانس تغییرات قیمت سهام تعریف کرد و مسئله را به صورت زیر مدل‌بندی کرد:

$$
\begin{aligned}
&Var = min \sum_{i=۱}^{n} \sum_{j=۱}^{n} w_i w_j \sigma_{ij} \\
&\sum_{i=۱}^{n} w_i \mu_i \geq R^* \\
&\sum_{i=۱}^{n} w_i = ۱, \quad ۰ \leq w_i \leq ۱ \quad for \quad i = ۱, \ldots, n
\end{aligned}
\tag{۱۴.۲}
$$

که در آن، σ_{ij} : کوواریانس بین دارایی های i و j است و از رابطه زیر به‌دست می‌آید.

$$\sigma_{ij} = \rho_{ij} \sigma_i \sigma_j \tag{۱۵.۲}$$

ρ_{ij} ضریب همبستگی بین دارایی های i و j و σ_i واریانس دارایی i است.
n : تعداد دارایی های موجود.
μ_i : بازگشت سرمایه دارایی i.
R^* : بازگشت سرمایه مورد انتظار.
w_i : درصد دارایی i در پرتفولیو (سبد سهام). بنابراین برای شروع مسئله باید میانگین و واریانس هر دارایی و ماتریس ضرایب همبستگی بین هر دو دارایی را داشته باشیم.
این مسئله یک مسئله بهینه‌سازی غیرخطی است که در آن واریانس در تابع هدف قرار داده شده است و برای سطحی از بازگشت سرمایه مقدار بهینه ریسک به‌دست می‌آید. با تغییر مقدار R^* نموداری بر حسب ریسک و بازگشت سرمایه به‌دست می‌آید که به آن مرز کارا[۳۰] می‌گویند. در مدل فوق، خط اول، تابع واریانس (ریسک) را نشان می‌دهد که هدف مینیمم کردن آن می‌باشد. خط دوم نشان دهنده‌ی مجموع سرمایه های بازگشتی که همان بازگشت سرمایه مورد انتظار یا R^* است و خط سوم نشان می‌دهد که مجموع درصد دارایی ها باید ۱ شود. برای به‌دست آوردن نمودار مرز کارا باید مقدار R^* را تغییر داد و برای مقادیر مختلف R^* مقدار بهینه ریسک را محاسبه کرد. راه دیگر برای انجام این‌کار، آوردن فرمول بازگشت سرمایه در تابع هدف و تخصیص ضریب λ به هر یک از فرمول‌های ریسک و بازگشت است.
بنابراین مدل‌بندی مسئله به‌صورت زیر در می‌آید:

[۳۰] Efficient frontier

$$Var = min\lambda(\sum_{i=۱}^{n}\sum_{j=۱}^{n} w_i w_j \sigma_{ij}) - (۱-\lambda)(\sum_{i=۱}^{n} w_i \mu_i)$$

$$s.t. \qquad (۱۶.۲)$$

$$\sum_{i=۱}^{n} w_i = ۱, \quad ۰ \leq w_i \leq ۱ \quad for \quad i = ۱, \ldots, n$$

در این معادله تابع هدف به‌صورت چندهدفه در آمده است و با تغییر مقادیر λ نمودار مرز کارا محاسبه می‌شود. این مسئله توسط روش‌های بهینه‌سازی غیرخطی قابل حل است اما در دنیای واقعی چنین مسئله ای بدون محدودیت نخواهد بود و مسئله از حالت ساده خود خارج شده است و به یک مسئله NP-hard [۳۱] (مسائلی که با افزایش سایز مسئله، زمان حل به صورت نمایی افزایش پیدا می‌کند که از الگوریتم‌های فراابتکاری به عنوان الگوریتم‌های حل این مسائل استفاده می‌شود [۷۰].) تبدیل می‌شود. مسئله دیگری که در این قسمت مدل شده است، دو محدودیت ”اندازه ثابت تعداد سهام (محدودیت کاردینال)” و ”حد بالا و پایین درصد هر سهم” را شامل می‌شود. چنگ و همکاران در مقاله ای که سال ۲۰۰۰ ارائه کرده اند، نشان داده اند که نمودار مرز کارا که در حالت بدون محدودیت پیوسته بوده است، در چنین حالتی به‌صورت گسسته در می‌آید. یعنی مقادیر بازگشت سرمایه‌ای وجود دارند که هیچ سرمایه‌گذاری روی آنها سرمایه‌گذاری نمی‌کند. فرمول‌بندی ریاضی مسئله با محدودیت‌های جدید به‌صورت زیر خواهد بود:

$$Var = min\lambda(\sum_{i=۱}^{n}\sum_{j=۱}^{n} w_i w_j \sigma_{ij}) - (۱-\lambda)(\sum_{i=۱}^{n} w_i \mu_i)$$

$$s.t.$$

$$\sum_{i=۱}^{n} w_i = ۱, \quad ۰ \leq w_i \leq ۱ \quad for \quad i = ۱, \ldots, n \qquad (۱۷.۲)$$

$$\sum_{i=۱}^{n} z_i = K \quad \varepsilon_i z_i \leq w_i \leq \sigma_i z_i$$

$$z_i \in \{۰, ۱\}$$

z_i : اگر دارایی i در سبد سهام موجود باشد، مقدار ۱ و در غیر این‌صورت مقدار صفر را می‌گیرد. (پارامتر تصمیم)

σ_i : بیشترین میزان نسبت سهم دارایی i در سبد سهام.

K : تعداد دارایی های موجود در سبدسهام که مقداری ثابت است.

ε_i : کمترین میزان نسبت سهم دارایی i در سبد سهام.

با توجه به سخت بودن مسئله انتخاب سهد سهام، تا کنون از الگوریتم‌های فراابتکاری متعددی برای حل این مسئله استفاده شده است. الگوریتم ژنتیک که به عنوان یکی از پر کاربردترین الگوریتم‌های فراابتکاری مطرح است در مسئله انتخاب سبد سهام نیز به میزان قابل توجهی به‌کار رفته است. از الگوریتم‌های دیگری که برای حل این دسته از مسائل ارائه شده است، می‌توان به الگوریتم‌های شبیه‌سازی

[۳۱] Nondeterministic Polynomial-hard

تبرید [۳۳]، جست‌وجوی ممنوع[۳۲] [۲۶]، کلونی مورچه‌ها[۳۳] [۶۱]، شبکه عصبی[۳۴] و الگوریتم ازدحام ذرات[۳۵] [۳۲] اشاره کرد.

برای حل این مسئله با استفاده از الگوریتم جست‌وجوی هارمونی ابتدا به تعریف ماتریس حافظه می‌پردازیم. هر سطر حافظه هارمونی در واقع یک جواب (هارمونی) است که برای این مسئله عبارت است از یک سری اعداد تصادفی بین صفر و یک که نرمال‌سازی شده اند. در هر سطر تعداد K عدد حقیقی s_i (نسبت خالی پرتفولیو برای هر دارایی) برای هر یک از دارایی‌ها وجود دارد. پس از نرمال‌سازی، سطر سومی که فرضی است، به جواب اضافه می‌شود که w_i را نشان می‌دهد. برای ارزیابی جواب‌ها، ابتدا باید w_i مربوط به هر دارایی موجود در پرتفولیو محاسبه شود. به هر دارایی باید به اندازه مینیمم مقدار آن نسبت تعلق بگیرد و لذا نسبت $1 - \sum \varepsilon_j$ از پرتفولیو خالی می‌ماند. با استفاده از مقادیر تصادفی s_i می‌توان این تخصیص را صورت داد. ابتدا این مقادیر را نرمال‌سازی کرده، سپس در نسبت خالی پرتفولیو ضرب کرده تا درصد تخصیص یافته هر دارایی به فضای خالی پرتفولیو به‌دست آید. نتیجه را با حد پایین جمع می‌کنیم. به این ترتیب w_i محاسبه می‌شوند. این مقادیر حدود پایین را برآورده کرده و جمعشان ۱ است. اما ممکن است بعضی از دارایی ها از حد مجاز σ_i آنها بالاتر رفته باشد؛ لذا آنهایی که نسبت‌شان از حد محاز بالاتر باشد را به حد مجاز بالا می‌رسانیم و عمل محاسبه w_i را بدون در نظر گرفتن آن دارایی ها انجام می‌دهیم تا زمانی‌که دیگر هیچ دارایی دارای چنین شرایطی نباشد. حال با استفاده از تابع هدف موجود، مقدار ارزیابی، مقدار ریسک و بازگشت سرمایه را می‌توان محاسبه کرد. بهترین جواب موجود در حافظه هارمونی باید با مقدار مشخص λ با مقدار ارزیابی جواب جدید مقایسه شود و در صورت به‌دست آمدن نتیجه بهتر جایگزین جواب قبلی شود.

فرآیند اجرای الگوریتم

ورودی‌های الگوریتم عبارتند از: ماتریس‌های حدود، میانگین و واریانس هر دارایی و ماتریس همبستگی. برای تعداد E مقدار λ با فاصله‌های مساوی بین صفر و یک، گام‌های زیر تکرار می‌شود:

۱) ماتریس حافظه‌ی هارمونی به‌صورت تصادفی تولید می‌شود.

۲) هر سطر ماتریس حافظه ارزیابی می‌شود.
برای تعداد تکرار معینی گام‌های زیر تکرار می‌شود:

الف) برای هر آرایه‌ی i، هارمونی جدید، دو عمل زیر را انجام بدهید:

۱) عددی تصادفی بین صفر و یک تولید شده و با پارامتر HMCR مقایسه می‌شود. در صورتی‌که این مقدار از HMCR کوچکتر باشد، عضوی از سطر i ام ماتریس به‌صورت تصادفی انتخاب خواهد شد و در صورتی که بزرگتر باشد، عددی تصادفی بین صفر و یک برای آرایه‌ی i ام انتخاب خواهد شد.

---

[۳۲] Tabu Search
[۳۳] Ant Colony
[۳۴] Neural network
[۳۵] Partical Swarm

۲) عددی تصادفی بین صفر و یک تولید شده و با PAR مقایسه می‌شود؛ در صورتی که کوچکتر از PAR باشد عملگر تنظیم گام بر روی آرایه i ام عمل می‌شود.

ب) جواب تولید شده ارزیابی می‌شود. در صورتی که این جواب از بدترین عضو حافظه بهتر باشد، جایگزین آن می‌شود.

ج) مقدار ریسک و بازگشت برای این جواب محاسبه می‌شود و مقدار λ به‌روز می‌شود.

بعد از اجرای الگوریتم، مجموعه‌ای از نقاط به تعداد λها به‌دست می‌آید که تقریبی از نمودار مرز کارا است. اگر خطا کم باشد و محدودیت کاردینال اعمال نشود، این نقاط دقیقاً منحنی مرز کارای بدون محدودیت کاردینال را مشخص می‌کند. در صورتی که محدودیت کاردینال در نظر گرفته شود و مسئله با الگوریتم هارمونی حل شود؛ مجموعه ای از نقاط به‌دست می‌آید که تقریبی از نمودار مرز کارا با محدودیت کاردینال می‌باشد.

فصل ۳

الگوریتم جست‌وجوی هارمونی در مسائل بهینه‌سازی چندهدفه

۱.۳ مقدمه

همان طور که در فصل اول نیز اشاره شد، بهینه‌سازی به معنای یافتن بهترین جواب ممکن برای یک مسئله با درنظرگرفتن مجموعه‌ای از شرایط و محدودیت‌ها است. بهینه‌سازی یا برنامه‌ریزی ریاضی در اقتصاد، ریاضیات، علوم کامپیوتر و مدیریت کاربرد فراوان دارد و به انتخاب بهترین جواب در بین وضعیت‌های موجود اشاره می‌کند. یک بخش از بهینه‌سازی، بهینه‌سازی چندهدفه[۱]است؛ اغلب مسائل بهینه‌سازی دنیای واقعی از اهداف متعددی تشکیل شده‌اند که معمولاً با یکدیگر در تضاد می‌باشند که به آنها مسائل بهینه‌سازی چندهدفه (MOPs[۲]) گفته می‌شود. مسائل بهینه‌سازی چندهدفه به‌جای یک جواب، مجموعه‌ای از جواب‌های بهینه را تولید می‌کند که به عنوان مجموعه‌ی بهینه پارتو[۳] شناخته می‌شود. بازده یافتن جواب‌های متعدد اکثر الگوریتم‌های بهینه‌سازی کلاسیک، پایین می‌باشد؛ زیرا این الگوریتم‌ها معمولاً در هر اجرا فقط یک جواب بهینه پارتو را به‌دست می‌آورند. بنابراین، این الگوریتم‌ها به تعداد زیاد اجرا می‌شوند تا مجموعه بهینه پارتو به‌دست آید. بهینه‌سازی چندهدفه، یکی از زمینه‌های بسیار پر کاربرد تحقیقاتی در میان مباحث بهینه‌سازی است و اهمیت فوق العاده‌ای در تقریباً همه مسائل

---

[۱] Multiobjective Optimization
[۲] Multiobjective Optimization Problems
[۳] Pareto optimal

بهینه‌سازی در دنیای واقعی دارد. در مسائل چندهدفه بر خلاف مسائل تک‌هدفه، با یک تابع هدف اسکالر مواجه نیستیم؛ بلکه تابع هدف یک بردار است. اغلب مسائل بهینه‌سازی مهندسی به‌صورت چندهدفه می‌باشند؛ زیرا به‌طور طبیعی دارای اهداف متعددی می‌باشند که باید به‌طور هم‌زمان بهینه (ماکزیمم سازی و یا مینیمم سازی) شوند. بهینه سازی چندهدفه یک زمینه تحقیقاتی مدرن و جالب است که کاربردهای زیادی در علوم مهندسی، اقتصاد، زیست شناسی، حمل و نقل و حتی در پزشکی دارد. بر خلاف مسائل بهینه‌سازی تک‌هدفه، اندازه‌گیری عملکرد مسائل بهینه‌سازی چند هدفه پیچیده‌تر است. سه هدف در بهینه‌سازی چند هدفه وجود دارد:

۱. همگرایی مجموعه بهینه پارتو.

۲. حفظ تنوع در جواب‌های مجموعه بهینه پارتو.

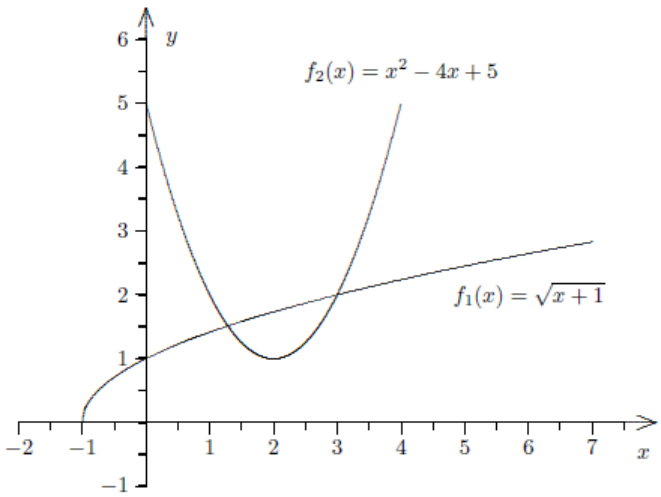
۳. حداکثر سازی مرز توزیع از مجموعه بهینه پارتو.

مثال ۱.۱.۳. می‌خواهیم توابع هدف زیر را در فاصله‌ی $x \geq ۰$ مینیمم کنیم.

$$\begin{cases} f_۱(x) = \sqrt{x+۱} \\ f_۲(x) = x^۲ - ۴x + ۵ = (x-۲)^۲ + ۱ \end{cases} \tag{۱.۳}$$

نمودار این تابع را در شکل ۱.۳ مشاهده می‌کنید. مسئله‌ی بهینه‌سازی زیر را در نظر بگیرید:

$$\begin{aligned} min \quad & (f_۱(x), f_۲(x)) \\ s.t. \quad & x \geq ۰ \end{aligned} \tag{۲.۳}$$



شکل ۱.۳: نمودار توابع $f_۱$ و $f_۲$

نقاط $x_۱ = ۰$ و $x_۲ = ۲$ به ترتیب نقاط بهینه‌ی توابع $f_۱$ و $f_۲$ هستند. اما نقطه‌ی مینیمم این مسئله‌ی دو هدفه کدام است؟

مشکل اصلی که در برخورد با مسائل چندهدفه مطرح است، وجود مجموعه‌ای از جواب‌ها می‌باشد که هر کدام از دیدگاهی می‌توانند بهینه باشند. یک مسئله‌ی چندهدفه برخلاف مسائل تک‌هدفه که دارای یک جواب بهینه هستند، معمولاً چندین جواب بهینه دارند، این جواب‌ها می‌توانند از تعداد کمی تا بی‌نهایت تغییر کنند. این مجموعه از جواب‌ها در حوزه‌ی بهینه‌سازی چندهدفه به مرز پارتو[۴] [۶۰] معروف هستند. مقیاس سنجش برای هر هدف ممکن است با مقیاس سنجش برای بقیه اهداف متفاوت باشد، مثلاً یک هدف، حداکثر کردن سود است که بر حسب پول سنجش می‌شود و هدف دیگر حداقل استفاده از ساعات نیروی کار است که بر حسب ساعت سنجش می‌شود. گاهی این اهداف در یک جهت نیستند و به صورت متضاد عمل می‌کنند. مثلاً تصمیم گیرنده از یک طرف تمایل دارد رضایت کارکنان را افزایش دهد و از طرف دیگر می‌خواهد هزینه‌های حقوق و دستمزد را حداقل کند. مسائل بهینه‌سازی چندهدفه معمولاً به‌صورت غیر مستقیم و با استفاده از تکنیک‌های معمول بهینه‌سازی تک‌هدفه حل می‌شوند و فرآیند حل به این صورت است که ابتدا یک مسئله تک‌هدفه متناظر با مسئله چندهدفه ساخته می‌شود و سپس مسئله تک‌هدفه حل می‌شود.

۲.۳ تعاریف مربوط به مسئله‌ی بهینه‌سازی چندهدفه

تعریف ۱.۲.۳. جواب بهینه

در یک مسئله تک‌هدفه، یک جواب شدنی $\widehat{\mathbf{x}}$ جواب بهینه[۵] نامیده می‌شود، اگر برای همه‌ی $\mathbf{x} \in X$ داشته باشیم: $f(\widehat{\mathbf{x}}) \leq f(\mathbf{x})$.

تعریف ۲.۲.۳. جواب بهینه اکید

در یک مسئله تک‌هدفه، یک جواب شدنی $\widehat{\mathbf{x}}$ جواب بهینه اکید[۶] نامیده می‌شود، اگر برای همه‌ی $\mathbf{x} \in X$ داشته باشیم: $f(\widehat{\mathbf{x}}) < f(\mathbf{x})$.

یک مسئله‌ی بهینه‌سازی چندهدفه در حالت کلی به‌صورت زیر بیان می‌شود:

$$\begin{aligned} &Min \quad \mathrm{F}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})) \\ &s.t. \\ &g_j(\mathbf{x}) \leq 0, j = 1, 2, \ldots, r \\ &h_p(\mathbf{x}) = 0, p = 1, 2, \ldots, q \\ &\mathbf{x} \in \Omega \subset \mathbb{R}^n \end{aligned} \tag{۳.۳}$$

به‌طوری که $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ بردار متغیرهای تصمیم می‌باشد و $f_i, g_j, h_p : \Omega \subset \mathbb{R}^n \longrightarrow \mathbb{R}, \quad \forall i, j, p, \Omega \neq \emptyset$ و به ازای هر $g_j(\mathbf{x})$ ، $j = 1, 2, \ldots, r$ نشان دهنده‌ی محدودیت‌های نامساوی است و به ازای هر $h_p(\mathbf{x}) = 0$ ، $p = 1, 2, \ldots, q$ محدودیت‌های

[۴] Pareto Front
[۵] Optimal Solution
[۶] Strictly Optimal Solution

تساوی را نشان می‌دهند. همچنین به ازای هر $i = ۱, ۲, \ldots, k$ ، توابع $f_i : \Omega \subset \mathbb{R}^n \longrightarrow \mathbb{R}$ ، k تابع هدف را نشان می‌دهند. در حالت $k = ۱$، مسئله‌ی چندهدفه به مسئله‌ی تک‌هدفه تبدیل می‌شود، بنابراین در مسائل بهینه‌سازی چندهدفه همواره فرض می‌کنیم که $k \geq ۲$ است.

تعریف ۳.۲.۳. تصویر ناحیه شدنی و فضای معیار[۷]

مجموعه تمام نقاط شدنی را با X نشان می‌دهیم و آن را مجموعه شدنی[۸]می‌گوییم. فضایی که مجموعه شدنی، زیر مجموعه‌ی آن است فضای تصمیم[۹] نام دارد. به عبارت دیگر:

$$X = \{\mathbf{x} \in \Omega \subset \mathbb{R}^n | g_j(\mathbf{x}) \leq ۰, h_p(\mathbf{x}) = ۰, \forall j = ۱, ۲, \ldots, r, p = ۱, ۲, \ldots, q\}$$

در این مسئله فضای تصمیم $\mathbb{R}^n$ است.

تصویر ناحیه شدنی X تحت تابع $F(\mathbf{x}) = (f_۱(\mathbf{x}), f_۲(\mathbf{x}), \ldots, f_k(\mathbf{x}))$ به‌صورت زیر نمایش داده می‌شود:

$$Y = F(\mathbf{x}) = \{\mathbf{y} \in \mathbb{R}^k : \mathbf{y} = F(\mathbf{x}), \mathbf{x} \in X\}$$

که به آن تصویر مجموعه شدنی یا مجموعه شدنی در فضای معیار گفته می‌شود. فضایی که توابع هدف از آن گرفته شده است، فضای معیار نام دارد. در این مسئله $\mathbb{R}^k$ فضای معیار است.

جهت یافتن جواب یا جواب‌های بهینه نیاز به مقایسه بین اعضای فضای معیار است. اما از آن‌جا که اعضای این فضا دارای بیش از یک مؤلفه می‌باشند، روابط ترتیبی عادی در این فضا برقرار نیستند، بنابراین نیاز به تعریف روابط ترتیبی مؤلفه ای، واضح است.

تعریف ۴.۲.۳. روابط ترتیبی مؤلفه‌ای

از روابط ترتیبی مؤلفه‌ای زیر برای مقایسه در $\mathbb{R}^k$ استفاده شده است: $(\mathbf{y}, \widehat{\mathbf{y}} \in \mathbb{R}^k)$

- $\mathbf{y} < \widehat{\mathbf{y}}$ به معنای $y_l < \widehat{y}_l$ به ازای همه $l = ۱, ۲, \ldots, k$.
- $\mathbf{y} \leqq \widehat{\mathbf{y}}$ به معنای $y_l \leq \widehat{y}_l$ به ازای همه $l = ۱, ۲, \ldots, k$.
- $\mathbf{y} \leq \widehat{\mathbf{y}}$ به معنای $\mathbf{y} \leqq \widehat{\mathbf{y}}$ اما $\mathbf{y} \neq \widehat{\mathbf{y}}$.

با استفاده از ترتیب‌های مؤلفه‌ای فوق، زیر مجموعه‌های $\mathbb{R}^k$ را به‌صورت زیر مشخص می‌کنیم:

$$\mathbb{R}^k_{\geqq} := \{\mathbf{y} \in \mathbb{R}^k : \quad \mathbf{y} \geqq ۰\}$$
$$\mathbb{R}^k_{\geq} := \{\mathbf{y} \in \mathbb{R}^k : \quad \mathbf{y} \geq ۰\} = \mathbb{R}^k_{\geqq} \setminus \{۰\}$$
$$\mathbb{R}^k_{>} := \{\mathbf{y} \in \mathbb{R}^k : \quad \mathbf{y} > ۰\} = int \quad \mathbb{R}^k_{\geqq}$$

به دلیل تضاد و عدم امکان مقایسه در فضای معیار، امکان یافتن جوابی که برای تمام توابع هدف، به‌طور هم‌زمان بهینه باشد، وجود ندارد. به‌عبارت دیگر، می‌توان گفت مسائل بهینه‌سازی چندهدفه

[۷] Criterion Space
[۸] Feasible Set
[۹] Decision Space

بد تعریف[۱۰]هستند. یک مسئله‌ی چندهدفه برخلاف مسائل تک‌هدفه که دارای یک جواب بهینه است، چندین جواب بهینه دارد. در مسائل بهینه‌سازی چندهدفه، فضای معیار دارای بیش از یک بعد است و به‌دنبال جوابی هستیم که در همه ابعاد بهترین باشد. برای یک مسئله بهینه‌سازی چندهدفه، یک جواب بهینه، جوابی است که هیچ یک از مؤلفه‌هایش نمی‌توانند بهبود یابند، مگر اینکه حداقل یکی از مؤلفه‌هایش بدتر شود. این تعریف اولین بار در سال ۱۸۸۱ توسط ادوارت[۱۱]اقتصاددان ایرلندی مطرح شد اما چون ویلفردو پارتو[۱۲]در سال ۱۸۹۶ آن را توسعه داد، به این تعریف اغلب بهینه‌ی پارتو می‌گویند [۷۱]. به‌طور عادی بهترین مفهوم قابل قبول از یک طرح شناخته شده‌ی بهینه، با عنوان بهینه‌ی پارتو است [۶۰]. مفاهیم اساسی برای شناختن مفهوم بهینه‌ی پارتو به‌صورت: تسلط پارتو، بهینگی پارتو[۱۳]، مجموعه بهینه‌ی پارتو و مرز بهینه‌ی پارتو می‌باشند که آنها را معرفی خواهیم کرد [۴۸].

تعریف ۵.۲.۳. تسلط پارتو

بردار u بر بردار v تسلط دارد اگر و تنها اگر u به‌طور جزئی کمتر از v باشد، یعنی

$$\forall i \in \{۱, ۲, \ldots, k\}, u_i \leq v_i \quad \wedge \quad \exists i \in \{۱, ۲, \ldots, k\} \quad s.t \quad u_i < v_i$$

و با $u \preceq v$ نمایش داده می‌شود.

مثلاً فرض کنید دو تابع زیر را داریم: $\overrightarrow{u}$=[۲.۴،۵.۳،۴.۵] و $\overrightarrow{v}$=[۲.۴،۵.۳،۴.۸] بردار u بر بردار v تسلط دارد؛ زیرا در تابع هدف اول و دوم مساوی است و در تابع هدف سوم کوچکتر است.

تعریف ۶.۲.۳. بهینگی پارتو

جواب شدنی $\mathbf{x} \in \Omega$ را بهینه پارتو یا کارا[۱۴] نسبت به Ω گویند اگر و تنها اگر جوابی مانند $\hat{\mathbf{x}} \in \Omega$ وجود نداشته باشد که $u = F(\mathbf{x}) = \{f_۱(\mathbf{x}), \ldots, f_k(\mathbf{x})\}$ بر $v = F(\hat{x}) = \{f_۱(\hat{\mathbf{x}}), \ldots, f_k(\hat{\mathbf{x}})\}$ تسلط داشته باشد. به عبارت دیگر، جواب شدنی $\mathbf{x}$ را بهینه پارتو گویند اگر هیچ جواب شدنی $\hat{\mathbf{x}}$ دیگری وجود نداشته باشد که $F_i(\hat{\mathbf{x}}) \leq F_i(\mathbf{x})$، $\forall i$ و برای حداقل یک $j \in \{۱, \ldots, k\}$ داشته باشیم $F_j(\hat{\mathbf{x}}) < F_j(\mathbf{x})$. اگر $\hat{\mathbf{x}}$ کارا باشد، $f(\hat{\mathbf{x}})$ نقطه‌ی غیرمغلوب[۱۵] نامیده می‌شود. مجموعه نقاط کارا را با X_E و مجموعه نقاط غیرمغلوب را با Y_N نمایش می‌دهیم. در شکل ۲.۳ نقاط مغلوب و غیرمغلوب نشان داده شده است.

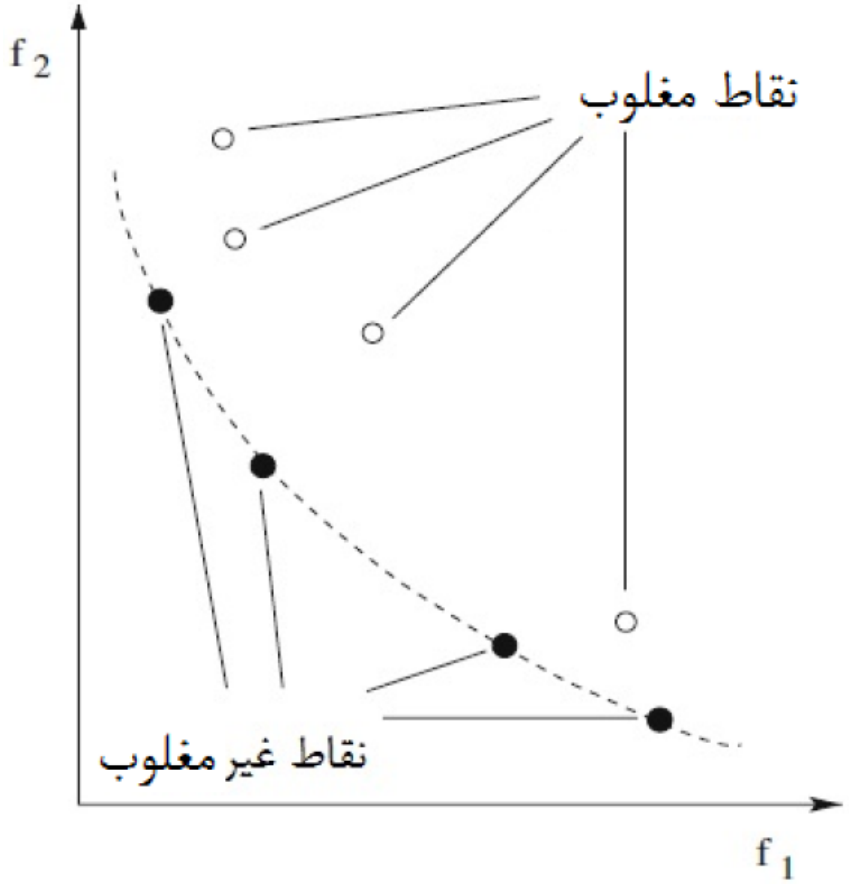
تعریف ۷.۲.۳. مجموعه بهینه پارتو

برای یک مسئله بهینه‌سازی چندهدفه $F(\mathbf{x})$، مجموعه بهینه پارتو P به‌صورت زیر تعریف می‌شود:

$$X_E = P = \{x \in X : \nexists \quad \hat{\mathbf{x}} \in X \quad s.t. \quad F(\hat{\mathbf{x}}) \preceq F(\mathbf{x})\}$$

---

[۱۰] Ill-defined
[۱۱] Edgeworth
[۱۲] Vilffredo Pareto
[۱۳] Pareto Optimality
[۱۴] Efficient
[۱۵] Nondominated Point



شکل ۲.۳: نقاط مغلوب و غیرمغلوب

تعریف ۸.۲.۳. مرز پارتو

برای یک مسئله بهینه‌سازی چندهدفه $\mathrm{F}(\mathbf{x})$ و یک مجموعه بهینه پارتو P، مرز پارتو (PF^*) به‌صورت زیر تعریف می‌شود:

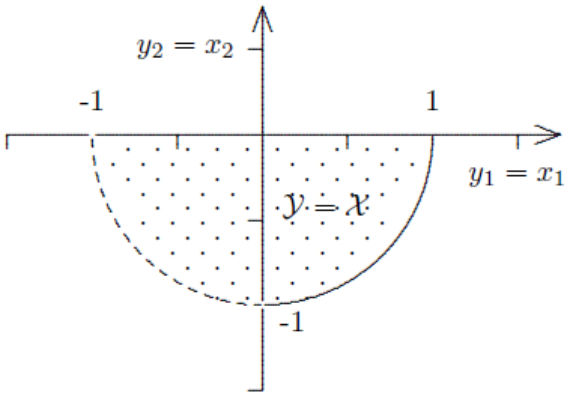
$$PF^* = Y_N := \{u = \mathrm{F}(\mathbf{x}) = \{(f_1(\mathbf{x}), \ldots, f_k(\mathbf{x}))\,|\mathbf{x} \in P\}$$

در ادامه به شرایط وجود نقاط کارا و خواص آنها پرداخته و با یک مثال نشان می‌دهیم که مجموعه‌های X_E و Y_N می‌توانند تهی باشند.

مثال ۹.۲.۳. [۶۴] مسئله بهینه‌سازی دو هدفه با ناحیه شدنی زیر و تابع هدف $f : \mathbb{R}^2 \longrightarrow \mathbb{R}^2$ با ضابطه $f(x_1, x_2) = (x_1, x_2)$ را در نظر بگیرید:

$$X = \left\{ (x_1, x_2) \in \mathbb{R}^n \;\middle|\; \begin{array}{ll} -1 \le x_1 \le 1 & \\ -\sqrt{-x_1^2 + 1} < x_2 \le 0, & if \quad -1 \le x_1 \le 0 \\ -\sqrt{-x_1^2 + 1} \le x_2 \le 0, & if \quad 0 \le x_1 \le 1 \end{array} \right. \qquad (۴.۳)$$

همان‌طور که در شکل ۳.۳ مشاهده می‌کنید با وجود اینکه X و Y محدب هستند و تابع f پیوسته است، هیچ نقطه غیر مغلوبی وجود ندارد، بنابراین این مسئله هیچ نقطه کارایی ندارد و $Y_N = X_E =$

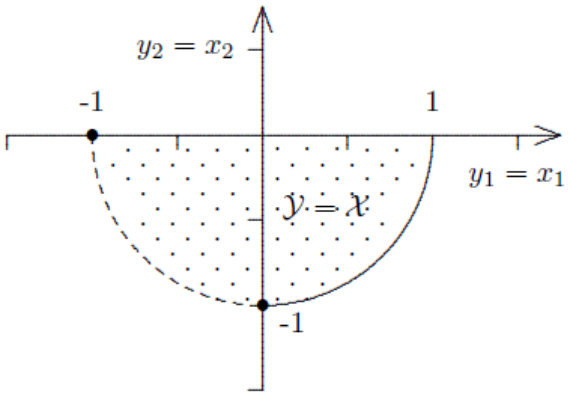


شکل ۳.۳: ناحیه شدنی مسئله اصلی

∅. حال اگر مسئله را با کمی تغییر به‌صورت زیر اصلاح کنیم:

$$X = \left\{ (x_1, x_2) \in \mathbb{R}^n \;\middle|\; \begin{array}{ll} -1 \le x_1 \le 1 & \\ x_2 = 0, \quad if \quad x_1 = -1 & \\ -\sqrt{-x_1^2 + 1} < x_2 \le 0, \quad if \quad -1 < x_1 < 0 & \\ -\sqrt{-x_1^2 + 1} \le x_2 \le 0, \quad if \quad 0 \le x_1 \le 1 & \end{array} \right. \tag{۵.۳}$$

در این صورت همان‌طور که در شکل ۴.۳ مشاهده می‌کنید، مجموعه نقاط غیر مغلوب و مجموعه نقاط کارا تهی نمی‌باشند، یعنی: $Y_N = X_E = \{(-1, 0), (0, -1)\}$.



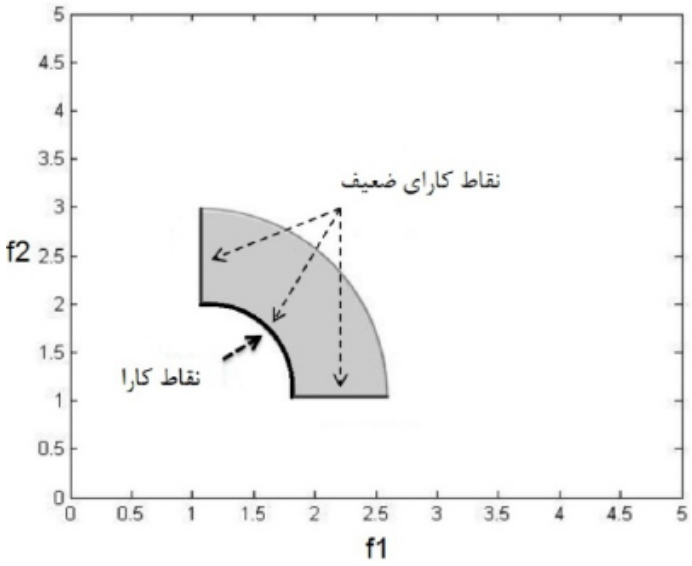
شکل ۴.۳: ناحیه شدنی مسئله اصلاح شده

این مسئله نشان می‌دهد که شرایط وجود جواب‌های کارا و غیر مغلوب از اهمیت ویژه‌ای در مسائل چندهدفه برخوردار است.

تعریف ۱۰.۲.۳. نقاط کارای ضعیف

یک جواب شدنی $\hat{\mathbf{x}} \in X$ کارای ضعیف[۱۶] نامیده می‌شود، اگر هیچ $\mathbf{x} \in X$ وجود نداشته باشد به‌طوری

[۱۶] Weakly Efficient



شکل ۵.۳: نقاط کارا و کارای ضعیف

که $f(\mathbf{x}) < f(\widehat{\mathbf{x}})$. اگر $\widehat{\mathbf{x}}$ کارای ضعیف باشد، آنگاه $f(\widehat{\mathbf{x}})$ نقطه غیرمغلوب ضعیف[۱۷] نامیده می‌شود. مجموعه نقاط کارای ضعیف را با X_{WE} و مجموعه نقاط غیرمغلوب ضعیف را با Y_{WN} نمایش می‌دهیم. در شکل ۵.۳ نقاط کارا و کارای ضعیف نشان داده شده است.

تعریف ۱۱.۲.۳. نقطه کارای اکید

یک جواب شدنی $\widehat{\mathbf{x}} \in X$ را کارای اکید[۱۸] گوییم اگر هیچ $\mathbf{x} \in X$ و $\mathbf{x} \neq \widehat{\mathbf{x}}$ وجود نداشته باشد، به‌طوری که $f(\mathbf{x}) \leqq f(\widehat{\mathbf{x}})$. نقاط کارای اکید را با X_{SE} نمایش می‌دهیم.

همان گونه که از تعریف جواب‌های کارا مشخص است، در یک جواب کارا نمی‌توان یکی از توابع هدف را بهبود داد مگر اینکه این بهبود با بدتر شدن حداقل یکی دیگر از توابع هدف همراه باشد. این تعامل (بده و بستان[۱۹]) بین توابع هدف، با محاسبه میزان افزایش یکی از توابع هدف به ازای کاهش یکی دیگر از توابع هدف به‌دست می‌آید. در بعضی از مواقع این تعامل ممکن است بی‌کران باشد، هنگامی‌که تعامل بین توابع هدف بی‌کران باشد، بهبود دادن یکی از توابع هدف، باعث بی‌نهایت بار بدتر شدن تابع هدف دیگری می‌شود. در عمل، یعنی حداقل یکی از توابع هدف نادیده گرفته می‌شود که مطلوب تصمیم گیرنده نیست، لذا جواب‌های کارای با مقدار تعامل کراندار از اهمیت زیادی برخوردار هستند که این جواب‌ها را کارای سره[۲۰] گویند [۶۴].

تعریف ۱۲.۲.۳. کارایی سره به مفهوم جفرین[۲۱]

یک جواب شدنی $\widehat{\mathbf{x}} \in X$ کارای سره نامیده می‌شود، اگر عدد حقیقی M وجود داشته باشد به‌طوری

[۱۷] Weakly Nondominated
[۱۸] Stricly Efficient
[۱۹] Trade-Off
[۲۰] Properly Efficient
[۲۱] Geoffrion

که برای هر $i \in \{۱, ۲, \ldots, k\}$ و هر $\mathbf{x} \in X$ که در رابطه $f_i(\mathbf{x}) < f_i(\widehat{\mathbf{x}})$ صدق می‌کنند، حداقل یک اندیس $j \in \{۱, ۲, \ldots, k\}$ وجود داشته باشد که $f_j(\widehat{\mathbf{x}}) < f_j(\mathbf{x})$ و $\dfrac{f_i(\widehat{\mathbf{x}}) - f_i(\mathbf{x})}{f_j(\mathbf{x}) - f_j(\widehat{\mathbf{x}})} \leq M$.
مجموعه نقاط کارای سره را با X_{PE} نمایش می‌دهیم. اگر $\widehat{\mathbf{x}}$ کارای سره باشد، آنگاه $f(\widehat{\mathbf{x}})$ نقطه غیرمغلوب سره[۲۲] نامیده می‌شود.

مطابق تعریف بالا، جواب‌های کارای سره آن جواب‌های کارایی هستند که تعاملی کراندار بین توابع هدف را برقرار می‌کنند. همان طور که پیش تر گفته شد، نتایج وجود جواب های غیرمغلوب (جواب های کارا) وجود جواب‌های غیرمغلوب ضعیف (جواب‌های کارای ضعیف) را نیز ارائه می‌کند اما این نتایج وجود نقاط کارای سره را تضمین نمی‌کند.

در ادامه فصل به روش های حل مسائل بهینه‌سازی چندهدفه پرداخته شده است.

۳.۳ روش‌های حل مسائل بهینه‌سازی چندهدفه

تولید جواب‌های بهینه پارتو نقش مهمی در بهینه‌سازی چندهدفه ایفا می‌کنند. از نظر ریاضی، مسئله وقتی حل شده است که جواب‌های بهینه پارتو یافت شوند. امروزه تکنیک‌های زیادی در برنامه‌ریزی ریاضی برای حل مسائل بهینه‌سازی چندهدفه وجود دارد. روش‌های بسیاری برای حل مسائل بهینه‌سازی چندهدفه توسعه یافته‌اند. این روش‌ها براساس معیارهای مختلف به شیوه‌های مختلف دسته‌بندی می‌شوند. در کتاب ارگات[۲۳] [۶۴] به روش‌های اسکالرسازی[۲۴] و روش‌های غیر اسکالرسازی[۲۵] پرداخته شده است. اسکالرسازی یکی از روش‌های قدیمی و متداول در حل مسائل بهینه‌سازی چندهدفه است. روش‌های اسکالرسازی از تکنیک‌های قدرتمند در بهینه‌سازی است. روش‌های اسکالرسازی مسئله چندهدفه را به مسئله‌ای تک‌هدفه تبدیل می‌کند، سپس با استفاده از تکنیک‌های استاندارد موجود برای حل مسائل بهینه‌سازی تک‌هدفه، آنها را حل می‌کند. در این بخش به برخی از روش‌های اسکالرسازی در مسائل بهینه‌سازی چندهدفه می‌پردازیم. روش‌های اسکالرسازی زیادی برای حل مسائل بهینه‌سازی چندهدفه وجود دارند از جمله : روش مجموع وزین[۲۶]، روش ε ـ مقید[۲۷]، روش هیبرید[۲۸]، روش چبیشف وزن‌دار[۲۹] و بسیاری روش‌های دیگر که هر کدام دارای خواص متفاوتی می‌باشند. به‌طور مثال، روش مجموع وزین تنها برای مسائل بهینه‌سازی محدب با انتخاب وزن‌های مناسب، تمام نقاط کارا را محاسبه می‌کند. در حالی که روش چبیشف وزن دار در مسائل محدب و نامحدب، با انتخاب وزن‌های مناسب، تمام نقاط کارا را ارائه می‌دهد.

---
[۲۲] Properly Nondominated
[۲۳] Ehrogott
[۲۴] Scalarization Methods
[۲۵] Nonscalarizing Methods
[۲۶] Weighted Sum Method
[۲۷] ε-Constraint Method
[۲۸] Hybrid Method
[۲۹] Weighted Tchebycheff Method

روش مجموع وزین

روش مجموع وزین ساده‌ترین روش حل مسائل بهینه‌سازی چندهدفه می‌باشد. در این روش وزن‌های نامنفی متناظر با توابع هدف در نظر گرفته می‌شوند و سپس مجموع وزن‌دار توابع هدف روی فضای شدنی مینیمم می‌شوند. مسئله مجموع وزین متناظر با مسئله بهینه‌سازی چندهدفه (۳.۳) به‌صورت زیر بیان می‌شود:

$$\begin{aligned} & min \sum_{i=1}^{k} \lambda_i f_i(\mathbf{x}) \\ & s.t \quad \mathbf{x} \in X \end{aligned} \tag{۶.۳}$$

که در آن $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_k) \in \mathbb{R}^k_{\geq}$ ضرایب وزنی نسبت داده شده به توابع هدف می‌باشند و $\sum_{i=1}^{k} \lambda_i = 1$. در قضیه زیر روابط بین جواب‌های بهینه مسئله (۶.۳) و جواب‌های کارای مسئله (۳.۳) ارائه شده است.

قضیه ۱.۳.۳. [۶۴]
فرض کنید $X \subset \mathbb{R}^n$ یک مجموعه محدب باشد و $f_i : X \longrightarrow \mathbb{R}, \quad i = 1, 2, \ldots, k$ توابعی محدب باشند، در این صورت اگر $\widehat{\mathbf{x}} \in X_{WE}$ ، $\lambda \in \mathbb{R}^k_{\geq}$ وجود دارد به‌طوری که $\widehat{\mathbf{x}}$ جواب بهینه مسئله (۶.۳) است.

قضیه ۲.۳.۳. [۶۴]
فرض کنید $X \subset \mathbb{R}^n$ یک مجموعه محدب باشد و $f_i : X \longrightarrow \mathbb{R}, \quad i = 1, 2, \ldots, k$ توابعی محدب باشند، در این صورت اگر $\widehat{\mathbf{x}} \in X_{PE}$ ، $\lambda \in \mathbb{R}^k_{>}$ وجود دارد به‌طوری که $\widehat{\mathbf{x}}$ جواب بهینه مسئله (۶.۳) است.

قضیه ۳.۳.۳. [۶۴] **جفرین**
فرض کنید $X \subset \mathbb{R}^n$ یک مجموعه محدب باشد و $f_i : X \longrightarrow \mathbb{R}, \quad i = 1, 2, \ldots, k$ توابعی محدب باشند، در این صورت $\widehat{\mathbf{x}} \in X$ جواب کارای سره مسئله (۳.۳) است اگر و تنها اگر X یک جواب بهینه مسئله (۶.۳) با وزن‌های اکیداً مثبت $(\lambda \in \mathbb{R}^k_{>}) \quad \lambda_i > 0, i = 1, 2, \ldots, k$ باشد.

از مزایای روش مجموع وزین، پیاده سازی آسان و کاربرد مناسب در مسائل بهینه‌سازی محدب می‌باشد اما این روش، برای مسائل بهینه‌سازی نامحدب معمولاً ضعیف عمل می‌کند و از آن جا که، تشخیص محدب بودن فضای مسئله بهینه‌سازی، کاری بسیار دشوار است لذا در استفاده از این روش باید محتاط بود. از دیگر معایب روش مجموع وزین این است که حتی اگر فضای شدنی محدب باشد، اوزان مختلف نمی‌توانند نقاط مختلف مرز کارایی را تولید کنند. اغلب به ازای ترکیبات محدب مختلف، نقطه یکسانی به‌دست می‌آید، یعنی نگاشت بین وزن‌های λ و نقاط کارا یک به یک نیستند.

مثال ۴.۳.۳. فرض کنید $X = \{\mathbf{x} \in \mathbb{R}^k_{\geq} : x_1^2 + x_2^2 \geq 1\}$ و $f(\mathbf{x}) = \mathbf{x}$ در این‌صورت $X_E = \{\mathbf{x} \in X : x_1^2 + x_2^2 = 1\}$ اما تنها جواب‌های شدنی که جواب‌های بهینه مسئله (۶.۳) برای هر $\lambda \geq 0$ می‌باشند، نقاط $\widehat{\mathbf{x}}_1 = (1, 0), \widehat{\mathbf{x}}_2 = (0, 1)$ هستند.

در این بخش روش اسکالرسازی دیگری که برای توابع نامحدب نیز قابل اجرا می‌باشد را معرفی می‌کنیم.

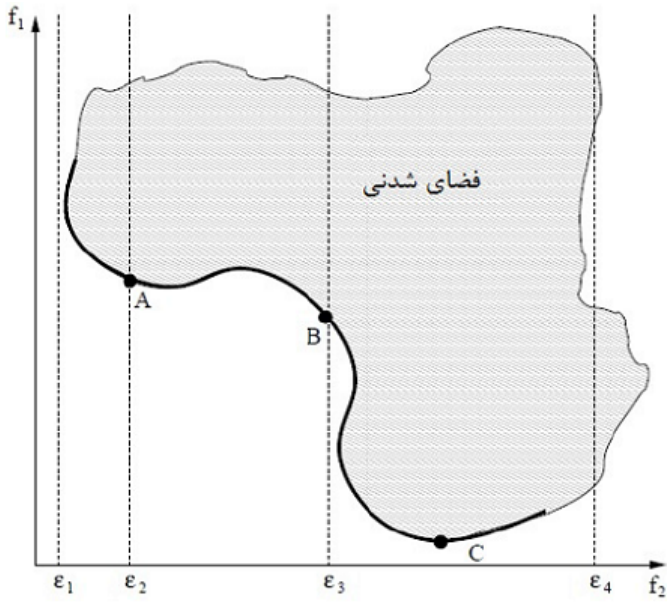
روش εـ مقید

در این روش یکی از توابع هدف مینیمم می‌شود در حالی‌که بقیه توابع هدف در قالب قید ظاهر می‌شوند. این روش اولین بار توسط هایمز و همکاران[۳۰] [۲۸] در سال ۱۹۷۱ مطرح شد و در سال ۱۹۸۳ توسط چان کونگ[۳۱] گسترش پیدا کرد. مسئله ε ـ مقید متناظر با مسئله بهینه‌سازی چندهدفه (۳.۳) به‌صورت زیر بیان می‌شود:

$$\begin{aligned} & min \quad f_j(\mathbf{x}) \\ & s.t \quad f_p(\mathbf{x}) \leq \varepsilon_p, \quad p = ۱, ۲, \ldots, k, p \neq j \\ & \mathbf{x} \in X \end{aligned} \tag{۷.۳}$$

که در آن به ازای هر $p \neq j$ ، $\varepsilon_p \in \mathbb{R}$ کران‌های بالای داده شده‌اند. با تغییر مقدار ε جواب‌های کارای مختلفی را می‌توان یافت. در این روش نسبت به روش مجموع وزین، مجموعه متنوع‌تری از جواب‌های کارا را می‌توان به‌دست آورد.
از معایب این روش وابستگی شدید جواب‌های به‌دست آمده به مقادیر ε است. این مقادیر باید به گونه‌ای انتخاب شوند که بین مقادیر حداقل و حداکثر هر تابع هدف محدود شده قرار گیرند. اضافه شدن قیود در این روش ممکن است ساختار فضای شدنی را به هم بریزد و باعث کاسته شدن کیفیت جواب شود. از طرفی دیگر با افزایش تعداد اهداف اطلاعات بیشتری از مسئله باید داشته باشیم، بنابراین این روش برای مسائل بزرگ و پیچیده بسیار وقت‌گیر و پرهزینه است. یکی از مزایای این روش، به‌دست آوردن مجموعه متنوع تری از جواب‌های کارا با تغییر مقدار ε، نسبت به روش مجموع وزین است. همچنین از دیگر مزایای این روش، مشکل ساز نبودن تفاوت مقیاس اهداف مانند روش مجموع وزین است. به علاوه، همان طور که قبلاً نیز اشاره شد و در شکل ۶.۳ مشاهده می‌شود این روش برای مسائل نامحدب نیز قابل اجرا می‌باشد. از معایب این روش، همان‌طور که در شکل ۶.۳ نشان داده شده است، وابستگی شدید جواب‌های به‌دست آمده به مقادیر انتخابی ε است، این مقادیر باید به گونه ای انتخاب شوند که بین مقادیر حداقل و حداکثر هر تابع هدف محدود شده قرار گیرند. همان گونه که در تصویر مشاهده می‌کنید، مینیمم $f_۱$ زمانی‌که $\varepsilon = \varepsilon_۲$ است در نقطه A است ولی زمانی‌که $\varepsilon = \varepsilon_۳$ است، در نقطه B می‌باشد.
اضافه شدن قیود در این روش، ممکن است ساختار فضای شدنی را به هم بریزد و باعث کاسته شدن کیفیت جواب شود. از طرفی دیگر با افزایش تعداد اهداف، اطلاعات بیشتری باید از مسئله داشته باشیم، بنابراین این روش برای مسائل بزرگ و پیچیده بسیار وقت گیر و پرهزینه است.

[۳۰] Haimes et. al.
[۳۱] Chankong



شکل ۶.۳: وابستگی جواب‌ها به ε در روش εـ مقید

قضیه ۵.۳.۳. شرط کافی برای جواب‌های کارای ضعیف و کارای اکید [۶۴]
فرض کنید $\widehat{\mathbf{x}} \in X$ جواب بهینه مسئله (۷.۳) برای یک $j \in \{1, 2, \ldots, k\}$ باشد،

- آنگاه $\widehat{\mathbf{x}}$ جواب کارای ضعیف مسئله (۳.۳) است. $(\widehat{\mathbf{x}} \in X_{WE})$
- اگر $\widehat{\mathbf{x}}$ جواب یکتای بهینه باشد، آنگاه $\widehat{\mathbf{x}}$ جواب کارای اکید مسئله (۳.۳) است. $(\widehat{\mathbf{x}} \in X_{SE})$

قضیه ۶.۳.۳. شرط لازم و کافی برای جواب‌های کارا [۶۴]
جواب شدنی $\widehat{\mathbf{x}} \in X$ کارا است اگر و تنها اگر $\widehat{\varepsilon} \in \mathbb{R}^k$ وجود داشته باشد به‌طوری که برای همه j ها $\widehat{\mathbf{x}} \in X$ جواب بهینه مسئله (۷.۳) باشد.

نتایج زیر ارتباط بین روش مجموع وزین و روش εـ مقید را نشان می دهد.

نتیجه ۷.۳.۳. فرض کنید $\widehat{\mathbf{x}} \in X$ جواب بهینه $min_{\mathbf{x}\in X} \sum_{i=1}^{k} \lambda_i f_i(\mathbf{x})$ باشد. اگر $\forall j \quad \lambda_j > 0$، آنگاه $\widehat{\varepsilon} \in \mathbb{R}^k$ وجود دارد به‌طوری که $\widehat{\mathbf{x}} \in X$ جواب بهینه مسئله (۷.۳) نیز می‌باشد.

نتیجه ۸.۳.۳. فرض کنید $X \subset \mathbb{R}^n$ یک مجموعه محدب باشد و $f_p : X \to \mathbb{R}, \quad p = 1, 2, \ldots, k$ توابعی محدب باشند. در این صورت اگر $\widehat{\mathbf{x}} \in X$ جواب بهینه مسئله (۷.۳) برای برخی j باشد، آنگاه $\widehat{\lambda} \in \mathbb{R}^k_{\geq}$ وجود دارد به‌طوری که $\widehat{\mathbf{x}} \in X$ جواب بهینه $min_{\mathbf{x}\in X} \sum_{i=1}^{k} \lambda_i f_i(\mathbf{x})$ است.

۴.۳ ساختار الگوریتم جست‌وجوی هارمونی در مسائل بهینه‌سازی چندهدفه

به منظور توضیح مفصل‌تر الگوریتم جست‌وجوی هارمونی، نیاز به ایده آل سازی فرآیند بداهه گویی انجام شده توسط یک موسیقی‌دان حرفه‌ای می‌باشد. هنگامی‌که یک موسیقی‌دان در حال بداهه گویی است،

می‌تواند سه انتخاب داشته باشد:

۱) اجرای هر قطعه از حافظه؛

۲) اجرای یک قطعه نزدیک به قطعات دیگر در حافظه‌اش؛

۳) اجرای یک قطعه‌ی تصادفی از دامنه‌ی همه‌ی قطعه‌های احتمالی.

به‌طور مشابه، هنگامی‌که هر متغیر تصمیم مقداری می‌گیرد، سه انتخاب وجود دارد:

۱) انتخاب هر مقدار از حافظه؛

۲) انتخاب یک مقدار نزدیک به مقدار دیگر در حافظه؛

۳) انتخاب یک مقدار تصادفی از دامنه‌ی همه‌ی مقادیر احتمالی.

گیم و همکاران در سال ۲۰۰۱ [۲۲] برای ایجاد یک روش ابتکاری جدید این سه انتخاب را فرمول بندی کردند. سه مؤلفه‌ی مربوطه عبارت بودند از: استفاده یا بررسی از حافظه، تنظیم و تغییر گام و انتخاب تصادفی. این سه گزینه با استفاده از دو پارامتر نرخ بررسی و نرخ تنظیم و تغییر گام در الگوریتم جست‌وجوی هارمونی، به‌کار می‌روند. با توصیف سه مؤلفه‌ی اصلی الگوریتم جست‌وجوی هارمونی، یعنی: حافظه هارمونی (HM)، نرخ بررسی حافظه‌ی هارمونی (HMCR) و نرخ تنظیم گام (PAR)، بخش‌های زیر هر گام از الگوریتم جست‌وجوی هارمونی را توضیح می‌دهند.

- **فرمول بندی مسئله**

الگوریتم جست‌وجوی هارمونی در ابتدا برای حل مسائل بهینه‌سازی که یک هدف دارند، مورد استفاده قرار گرفت. لذا برای اعمال جست‌وجوی هارمونی استاندارد روی یک مسئله، باید آن را به‌صورت یک مسئله بهینه‌سازی تک هدفه با یک تابع هدف و چند قید فرمول‌بندی کرد:

$$f(\mathbf{x}) = f(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) \qquad (۸.۳)$$

به‌طوری که

$$h_i(\mathbf{x}) = 0; i = 1, \ldots, p \qquad (۹.۳)$$

$$g_i(\mathbf{x}) \geq 0; \quad i = 1, \ldots, q \qquad (۱۰.۳)$$

$$\mathbf{x}_i \in X_i = \{\mathbf{x}_i(1), \mathbf{x}_i(2), \ldots, \mathbf{x}_i(K_i)\} \quad \text{یا} \quad \mathbf{x}_i^L \leq \mathbf{x}_i \leq \mathbf{x}_i^U \qquad (۱۱.۳)$$

الگوریتم جست‌وجوی هارمونی همه‌ی فضای جواب برای یافتن بردار جواب بهینه
$\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$

که تابع هدف در معادله‌ی (۸.۳) را بهینه (مینیمم یا ماکزیمم) می‌کند، جست‌وجو می‌کند. اگر

مسئله دارای قیدهای مساوی یا نامساوی باشد، این قیدها را می‌توان به‌صورت معادله‌های (۹.۳) و (۱۰.۳) در آورد. اگر متغیرهای تصمیم مقادیر گسسته باشند، مجموعه مقادیر ممکن به‌صورت

$$\mathbf{x}_i \in X_i = \{\mathbf{x}_i(۱), \mathbf{x}_i(۲), \ldots, \mathbf{x}_i(K_i)\}$$

بیان می‌شود که K_i تعداد مقادیر مختلف در فضای تعریف متغیر i بوده و از سوی دیگر اگر متغیرها دارای مقادیر پیوسته باشند، مجموعه مقادیر ممکن به‌صورت $\mathbf{x}_i^L \leq \mathbf{x}_i \leq \mathbf{x}_i^U$ نشان داده می‌شود.

- **پیکربندی پارامتر**

هنگامی‌که فرمول‌بندی یک مسئله آماده شد، مقدار پارامترهای الگوریتم باید مشخص شوند. در کنار پارامترهایی که قبلاً ذکر شد، الگوریتم جست‌وجوی هارمونی، پارامترهای دیگری مانند پارامتر FW[۳۲] [۲۳] (تغییر پذیری محدوده‌ی گام) نیز دارد که با پارامتر PAR در تنظیم گام عمل می‌کند. در فصل دوم، از این پارامتر با نام BW نام برده شد که از زمان پیدایش این الگوریتم از این اصطلاح استفاده می‌شد اما در چند سال اخیر، این پارامتر با اصطلاح FW به کار برده می‌شود. یک مقدار کم برای PAR به همراه مقدار کم برای FW می‌تواند همگرایی الگوریتم را کند ساخته و منجر به محدودیت جست‌وجو به یک بخش فضای تحقیق شود. از سوی دیگر مقدار بسیار زیاد برای PAR به همراه یک مقدار وسیع برای FW ممکن است منجر به متفرق شدن جواب‌ها در اطراف یک بهینگی در جست‌وجوی تصادفی شود. به این دلایل معمولاً این دو پارامتر را در بازه‌ی [۰٫۱, ۰٫۵] در نظر می‌گیرند. FW، بین ۱٪ و ۱۰٪ از همه دامنه مقادیر متغیرها محدود می‌شود [۷۴].

- **مقداردهی اولیه‌ی حافظه**

پس از اینکه مسئله فرمول‌بندی شده و پارامترها به درستی مشخص شدند، یک فرآیند ترکیب تصادفی در حافظه انجام می‌شود. الگوریتم جست‌وجوی هارمونی در ابتدا جواب‌های متعددی را به‌صورت تصادفی ایجاد می‌کند. تعداد جواب‌ها باید حداقل برابر با HMS باشد. با این حال، این مقدار ممکن است بیشتر باشد تا مقدار دو برابر یا حتی سه برابر [۱۵]. سپس، بهترین جواب‌های HMS انتخاب می‌شوند. ماتریس حافظه به‌صورت زیر ارائه می‌شود:

$$\left[\begin{array}{cccc|c} x_۱^۱ & x_۲^۱ & \ldots & x_n^۱ & f(x^۱) \\ x_۱^۲ & x_۲^۲ & \ldots & x_n^۲ & f(x^۲) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_۱^{HMS} & x_۲^{HMS} & \ldots & x_n^{HMS} & f(x^{HMS}) \end{array}\right] \qquad (۱۲.۳)$$

- **بداهه‌گویی**

سه گزینه وجود دارد که الگوریتم جست‌وجوی هارمونی هنگام بداهه‌گویی ممکن است انتخاب کند:

[۳۲] Fret Width

(آ) **انتخاب تصادفی**: هنگامی‌که الگوریتم هارمونی مقدار x_i^{new} را برای یک جواب

$$x^{new} = [x_1^{new}, x_2^{new}, \ldots, x_n^{new}]$$

مشخص کند، به‌طور تصادفی یک مقدار را از دامنه همه مقادیر احتمالی

$$\{x_i(1), x_i(2), \ldots, x_i(K_i)\}$$

یا $x_i^L \leq x_i \leq x_i^U$ با احتمال (HMCR-۱) انتخاب می‌کند.

(ب) **بررسی حافظه**: هنگامی‌که الگوریتم جست‌وجوی هارمونی مقداری از x_i^{new} را تعیین کند، به‌طور تصادفی مقدار x_i^j را از $HM(j = 1, 2, \ldots, HMS)$ با احتمال برابر با $HMCR$ انتخاب می‌کند. اندیس تصادفی j ممکن است با استفاده از یک توزیع یکنواخت $U(0, 1)$ محاسبه شود:

$$j \leftarrow int(U(0, 1) * HMS) + 1 \qquad (۱۳.۳)$$

(ج) **تنظیم گام**: پس از اینکه مقدار x_i^{new} به‌طور تصادفی از حافظه هارمونی در فرآیند توصیف شده انتخاب شد، این مقدار ممکن است با احتمال PAR و با اضافه کردن یا کم کردن یک مقدار مفروض تغییر کند. برای متغیرهای گسسته، اگر $x_i(k) = x_i^{new}$، تنظیم گام به‌صورت $x_i(k+m)$ می‌باشد که در آن $m \in \{-1, 1\}$. برای متغیرهای پیوسته، تنظیم گام به‌صورت $x_i^{new} + \Delta$ می‌باشد که در آن $\Delta = U(-1, 1) * FW(i)$.

سه عملیات اساسی ذکرشده به‌صورت زیر بیان می‌شود:

(۱۴.۳)

$$x_i^{new} \leftarrow \begin{cases} \begin{cases} x_i \in \{x_i(1), x_i(2), \ldots, x_i(K_i)\} \\ x_i \in [x_i^L, x_i^U] \end{cases} & w.p. \quad (1 - HMCR) \\ x_i \in HM = \{x_i^1, x_i^2, \ldots, x_i^{HMS}\} & w.p. \quad HMCR * (1 - PAR) \\ \begin{cases} x_i(k+m) & if \quad x_i(k) \in HM \\ x_i + \Delta & if \quad x_i \in HM \end{cases} & w.p. \quad HMCR * PAR \end{cases}$$

که w.p. به‌معنی ''با احتمال'' است.

- **به روزرسانی حافظه**

اگر جواب جدید x^{new} از بدترین جواب در حافظه هارمونی از نظر مقدار تابع هدف بهتر باشد، جواب جدید وارد حافظه شده و بدترین جواب کنار گذاشته می‌شود.

$$(x^{new} \in HM) \wedge (x^{worst} \notin HM) \qquad (۱۵.۳)$$

- **پایان**

اگر الگوریتم جست‌وجوی هارمونی به محک توقف رسید، برای مثال، به ماکزیمم مقدار تکرار یا ماکزیمم زمان اجرایی برسد، فرآیند پایان می‌یابد؛ در غیر این صورت جواب دیگری ایجاد می‌شود.

شبه‌کد الگوریتم در زیر ارائه شده است [۷۴].

شبه‌کد الگوریتم جست‌وجوی هارمونی:

۱. تابع $f(x)$ و پارامترهای $HMCR$، PAR، HMS، NI و FW را وارد کنید.
۲. حافظه هارمونی، HM، را به‌صورت تصادفی مقداردهی اولیه کنید.
۳. برای هر متغیر x_i عملیات زیر را انجام بدهید تا هنگامی‌که شرط توقف برقرار شود:
الف) اگر $U(۰,۱) < HMCR$ آنگاه $x^{new} \leftarrow x_i^j$ که $j \leftarrow int(U(۰,۱) * HMS) + ۱$.
ب) اگر $U(۰,۱) < PAR$ آنگاه x_i^{new} را با $x_i(k+m)$ یا $x_i^{new} + \Delta$ به روز کنید.
ج) در غیر این صورت، به x^{new} یک مقدار تصادفی بدهید.
۴. اگر x^{new} از بدترین جواب در حافظه هارمونی، بهتر بود، x^{new} را جایگزین کنید.
۵. بهترین مقدار x در حافظه هارمونی، به عنوان خروجی است.

۵.۳ حل مسئله چندهدفه با استفاده از جست‌وجوی هارمونی با روش‌هایی که براساس تسلط پارتو نمی‌باشند

هر چند در مراجع جاری آثار کمی را می‌توانیم پیدا کنیم که کاربرد الگوریتم هارمونی را برای حل مسائل چندهدفه خاص پیشنهادمی‌کند، اما اغلب آنها از مفهوم بهینگی پارتو استفاده نمی‌کنند یا روش چندهدفه پیشنهادی را به‌طور صریح معرفی نمی‌کنند. در این بخش، خلاصه ای از کارهای مربوط به الگوریتم جست‌وجوی هارمونی و بهینه‌سازی چندهدفه ارائه می‌شود که از مفهوم بهینه پارتو استفاده نمی‌شود.

گیم و همکاران [۲۴] الگوریتم هارمونی را برای مسیریابی وسیله نقلیه، به‌خصوص در مسیریابی اتوبوس مدرسه به‌کار بردند که در آن هدف حداقل کردن تعداد اتوبوس‌ها و مجموع زمان سفر همه اتوبوس‌ها است در حالی‌که محدودیت‌های ظرفیت اتوبوس‌ها و زمان هر توقف باید برآورده شوند.

گیم و هانگبو[۳۳] [۶۷] از الگوریتم هارمونی برای بهینه‌سازی طراحی لوله‌های گرمایی ماهواره استفاده کردند که این مسئله شامل پیداکردن ابعاد لوله‌ها و دمای اجرای آنها برای حداقل کردن جرم لوله‌ها و حداکثر کردن هدایت حرارتی می‌باشد. همچنین گیم [۲۱] از الگوریتم هارمونی برای بهینه‌سازی طراحی پروژه با هدف مینیمم کردن زمان و هزینه در مسئله توازن زمان–هزینه استفاده کرده است. در کارهای ذکرشده‌ی قبلی، یک تابع تک‌هدفه که از مجموع وزنی اهداف حاصل شده است، مینیمم می‌شود. به‌علاوه، دو کار اول، از مفهوم بهینگی پارتو استفاده نمی‌کنند. با این‌حال، گیم [۲۱] اجرای یک درجه‌بندی را در میان جواب‌های بهینه‌ی پارتو، بدون شرح روش ذکر کرد.

[۳۳] Hwangbo

گائو و همکاران[۳۴] [۶۶] یک نسخه اصلاح‌شده از الگوریتم هارمونی را برای مسائل بهینه سازی تک‌هدفه مقید، با استفاده از مفهوم بهینگی پارتو پیشنهاد دادند. تابعی که باید بهینه شود از اجتماع تابع هدف و مجموع وزنی قیدها نتیجه شده است. کاربرد مفهوم بهینگی پارتو منجر به درجه‌بندی جواب‌های نشدنی می‌شود؛ یعنی امکان تکامل به جواب‌هایی که یک یا چند محدودیت را نقض می‌کنند، داده می‌شود.
سرانجام، ژئو و همکاران[۳۵] [۷۳] یک نسخه چندهدفه از الگوریتم هارمونی را برای طراحی یک نمونه ربات فایل پیکربندی مجدد پیشنهاد کردند. اهدافی که باید در این مسئله مینیمم شوند، پایداری ربات موبایل، مقاومت پیچشی چرخ‌های عقب و جرم ربات موبایل می‌باشند. اگر چه این کار از مفهوم بهینگی پارتو استفاده کرده و مسئله مطابق رابطه ۳.۳ ارائه شده است، چگونگی انجام درجه‌بندی پارتو جواب‌ها را بیان نکرده است و جزییات تغییرات انجام شده روی الگوریتم هارمونی اصلی برای تولید یک نسخه چندهدفه را توضیح نمی‌دهد. با این‌حال، بر طبق اطلاعات ما، این الگوریتم تنها روشی است که به‌طور کامل فرمول‌بندی و حل یک مسئله‌ی بهینه‌سازی چندهدفه واقعی را بیان می‌کند.

۶.۳ طرح‌هایی از الگوریتم جست‌وجوی هارمونی چندهدفه براساس تسلط پارتو

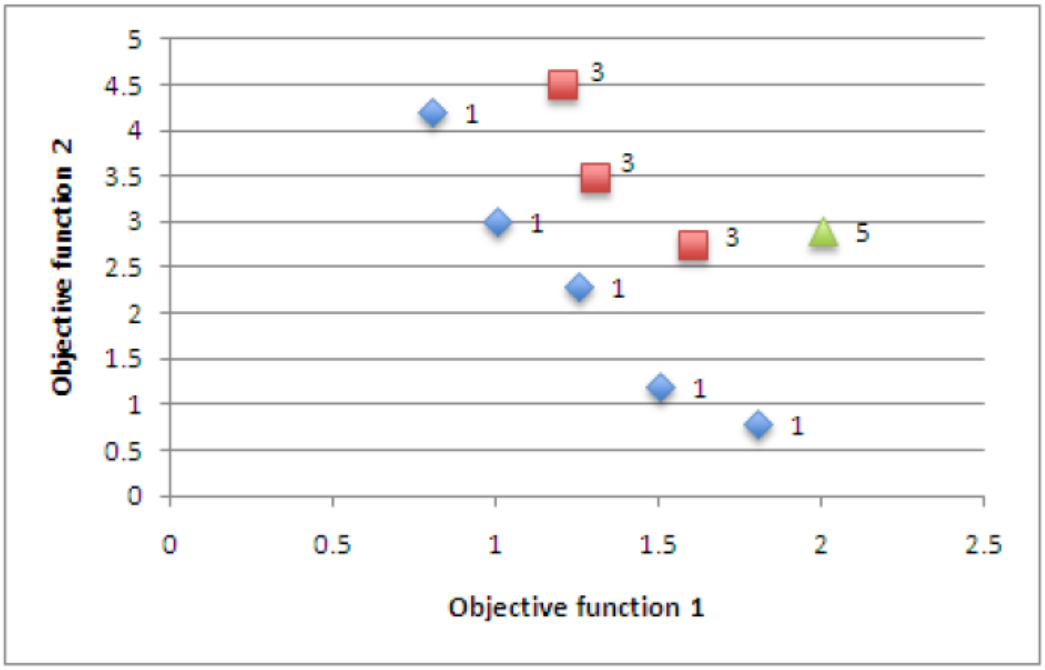
تفاوت اصلی بین الگوریتم هارمونی تک‌هدفه و الگوریتم هارمونی چندهدفه که در این فصل پیشنهاد می‌شود، در روش دسته‌بندی جواب (تخصیص رتبه‌بندی) می‌باشد. تخصیص رتبه‌بندی یعنی نسبت دادن یک عدد به هر جواب است به طوری‌که مشخص می‌کند آن جواب از بقیه بهتر است یا بدتر. تعیین یک رتبه برای جواب‌ها به این فرم است که بهترین جواب‌ها رتبه‌های پایین‌تری از بدترین‌ها دارند. در یک مسئله تک‌هدفه، رتبه‌بندی جواب‌ها با استفاده از تابع هدف تعیین می‌شود. با این‌حال، برای مسائل چندهدفه روش‌های متعددی گسترش پیدا کردند که شامل مفاهیم بهینگی پارتو می‌شدند [۶۸]. روش تخصیص رتبه‌بندی انتخاب شده برای مسائل الگوریتم هارمونی چندهدفه روش پیشنهادشده توسط فونسه‌کا و فلمینگ [۶۵] می‌باشد که در آن رتبه‌بندی یک جواب x_i برای یک تکرار t طبق معادله زیر تعریف می‌شود:

$$rank(x_i, t) = ۱ + p_i^{(t)} \qquad (۱۶.۳)$$

که $p_i^{(t)}$ تعداد جواب‌ها در تکرار فعلی است که بر جواب موردنظر تسلط دارند. بنابراین به جواب‌های غیرمغلوب[۳۶] حافظه هارمونی رتبه‌بندی معادل ۱ تخصیص داده می‌شود در حالی‌که جواب‌های مغلوب[۳۷] رتبه‌بندی معادل $q \in \{۲, \ldots, HMS\}$ می‌گیرند. شکل ۷.۳ یک مثال فرضی را نشان می‌دهد که نقاط مختلفی در فضای هدف نشان می‌دهد که در آن، دو تابع باید بهینه شده و درجه‌بندی فونسه‌کا ـ فلمینگ به هر نقطه تخصیص داده می‌شود.

---

[۳۴] Gao et al
[۳۵] Xu
[۳۶] Non-dominated solutions
[۳۷] Dominated Solutions



شکل ۷.۳: درجه‌بندی فونسه‌کا_فلمینگ برای نقاطی در فضای هدف دوبعدی

۱.۶.۳ MOHS۱:جست‌وجوی هارمونی چندهدفه، اولین طرح

این الگوریتم چندهدفه به گونه‌ای است که نیاز به تغییرات قابل توجه در رفتار الگوریتم هارمونی تک‌هدفه نمی‌باشد. تفاوت در استفاده از حافظه هارمونی به‌عنوان منبعی برای بهترین جواب‌های متوازن پیدا شده در یک نقطه‌ی مفروض در زمان می‌باشد، که رتبه‌بندی آنها مطابق با روش فونسه‌کا_فلمینگ است. این الگوریتم با مقداردهی اولیه‌ی حافظه هارمونی با جواب‌هایی که به‌طور تصادفی به‌وجود آمده‌اند، شروع می‌شود. سپس با محاسبه‌ی درجه‌بندی این جواب‌ها ادامه می‌یابد.

هزینه عملیاتی درجه‌بندی جواب‌ها و مقایسه هر جواب با جواب‌های دیگر $O(HMS^2)$ می‌باشد. در هر تکرار با استفاده از مقادیر متغیرهای تصمیم جواب‌های حافظه هارمونی، الگوریتم سعی می‌کند تا یک جواب ارزیابی شده را پیدا کند. این روند معادل با روند الگوریتم هارمونی تک‌هدفه است. برای این جواب جدید تولید شده، یک رتبه‌بندی با بررسی جواب‌های ذخیره شده در حافظه هارمونی محاسبه می‌شود. اگر این رتبه‌بندی بهتر از بدترین جواب رتبه‌بندی شده در حافظه هارمونی باشد، جواب جدید جایگزین بدترین جواب در حافظه می‌شود. اگر بیش از یک مورد وجود داشته باشد یکی به‌طور تصادفی از میان بدترین‌ها برای جایگزینی انتخاب می‌شود. این روند دارای هزینه‌ی مجانب $O(HMS)$ برای محاسبه‌ی مجدد درجه‌بندی جواب‌ها در حافظه هارمونی می‌باشد.

در هر تکرار، جواب‌های غیرمغلوب ذخیره شده در حافظه با تقریبی مجموعه‌ی پارتو با مسئله‌ای که باید حل شوند، متناظر می‌شوند. زمانی‌که به معیار توقف می‌رسد، جواب‌های با درجه‌بندی مساوی ۱، (جواب‌های غیرمغلوب)، که در حافظه هارمونی ذخیره شده‌اند، به عنوان بهترین تقریب برای جواب‌های بهینه پارتو مسئله چندهدفه مورد نظر معرفی می‌شوند [۴۳].

شبه‌کد الگوریتم ۱ $MOHS$

۱. تابع $f(x)$ و پارامترهای $HMCR$، PAR، HMS، NI و FW را وارد کنید.
۲. حافظه هارمونی، HM، را به‌صورت تصادفی مقداردهی اولیه کنید.
۳. تا زمانی‌که شرط توقف برقرار شود، عملیات زیر را انجام بدهید:
الف) یک جواب جدید، S را تولید کنید. برای این جواب تولید شده، یک رتبه‌بندی را با استفاده از حافظه هارمونی محاسبه کنید.
ب) اگر این رتبه‌بندی از بدترین جواب رتبه‌بندی شده بهتر بود، حافظه هارمونی را با S به روز کنید.
۴. بهترین جواب‌ها را از حافظه هارمونی خارج کرده و به عنوان بهینه پارتو، P در نظر می‌گیریم.

۲.۶.۳ MOHS۲: جست‌وجوی هارمونی چندهدفه، دومین طرح

ایده‌ی اصلی این طرح، تولید یک حافظه‌ی $HM_۲$ جدید در هر تکرار با تعداد جواب‌های برابر با حافظه‌ی اصلی $HM_۱$ می‌باشد. از اجتماع دو حافظه $H_u \leftarrow HM_۱ \cup HM_۲$ تنها نیمی از جواب‌ها به‌عنوان مؤلفه‌های حافظه برای تکرار بعدی پذیرفته می‌شوند. الگوریتم مطرح‌شده مشابه با الگوریتم هارمونی تک‌هدفه، با تولید جواب‌هایی با مقادیر تصادفی برای متغیرها تا زمان پرشدن $HM_۱$ آغاز می‌شود. هر تکرار با ایجاد همه‌ی جواب‌هایی که به $HM_۲$ تعلق خواهند داشت، آغاز می‌شود که اعضای $HM_۲$ با استفاده از روش انتخاب مشابه با الگوریتم HS تک‌هدفه انتخاب می‌شوند. یعنی مقادیر هر جواب محاسبه‌شده در $HM_۲$، با استفاده از مقادیر متغیرهای تصمیم موجود در $HM_۱$ ، به‌عنوان مقادیر مورد بررسی، تولید می‌شوند. زمانی‌که جواب‌های $HM_۲$ تولید شدند، رتبه‌بندی فونسه‌کا_فلمینگ روی H_u انجام می‌شود. زمانی‌که فرآیند تخصیص رتبه‌بندی پایان یافت، بهترین جواب‌های H_u انتخاب می‌شوند. به منظور انجام این امر، ابتدا جواب‌های H_u در مرزهایی گروه‌بندی می‌شوند که هر مرز شامل جواب‌هایی با رتبه‌بندی یکسان می‌باشد. سپس جواب‌های این مرزها به‌ترتیب صعودی به $HM_۱$ منتقل می‌شوند: یعنی جواب‌های موجود در مرز با پایین ترین درجه ابتدا و سپس مرزهای با رتبه‌بندی بالاتر به‌صورت متوالی قرار می‌گیرند. تا زمانی‌که تعداد جواب‌ها در یک مرز برابر یا بزرگتر از فضای موجود در $HM_۱$ باشد، این فرآیند ادامه می‌یابد. اگر اندازه‌ی مرز از فضای موجود در $HM_۱$ بزرگتر باشد، فرآیند کوتاه کردن که برای الگوریتم SPEA۲[۳۸][۵۴] اجرا شده است، به‌کار گرفته می‌شود. فرآیند کوتاه کردن، تعداد جواب‌ها را تا زمانی‌که اندازه‌ی مرز برابر با فضای موجود در $HM_۱$ باشد، کاهش می‌دهد. با تکمیل انتقال جواب‌ها، $HM_۱$ دارای مجموعه‌ی جدیدی از جواب‌ها برای تکرار بعدی می‌باشد. زمانی‌که معیار توقف الگوریتم تأمین می‌شود، جواب‌های $HM_۱$ با رتبه‌بندی معادل ۱، (جواب‌های غیرمغلوب) به عنوان بهترین تقریب مجموعه بهینه پارتو معرفی می‌شوند [۴۳].

[۳۸] Strength Pareto Evolutionary Algorithm

شبه‌کد الگوریتم ۲ $MOHS$

۱. تابع $f(x)$ و پارامترهای $HMCR$، PAR، HMS، NI و FW را وارد کنید.
۲. حافظه هارمونی، HM_1 را به‌صورت تصادفی مقداردهی اولیه کنید.
۳. تا زمانی‌که شرط توقف برقرار شود، عملیات زیر را انجام بدهید:
۱. حافظه هارمونی، HM_2 را خالی کنید.
۲. تا زمانی‌که HM_2 پر نشده است، یک جواب جدید S را از حافظه هارمونی HM_1 تولید کرده و آن را در HM_2 ذخیره کنید.
۳. $H_u \leftarrow HM_1 \cup HM_2$.
۴. رتبه‌بندی پارتو از H_u را با استفاده از تعریف فونسه‌کا–فلمینگ حساب کنید.
۵. HM_1 را خالی کنید.
۶. $R \leftarrow 1$ و همه جواب‌های از H_u با رتبه R را خارج کرده و در F قرار بدهید.
۷. تا زمانی‌که HM_1 فضایی بزرگتر یا مساوی $|F|$ (عدد اصلی مجموعه F) دارد، انجام بدهید:
الف) همه جواب‌ها را از F به HM_1 انتقال داده و $R \leftarrow R + 1$.
ب) همه جواب‌های H_u با رتبه R خارج کرده و در F قرار بدهید.
۸. فضای باقی‌مانده در HM_1 را در T قرار دهید. اگر $T > 0$ آنگاه F را به اندازه T کوتاه کنید و هر جواب F را به HM_1 انتقال دهید.

فرآیند کوتاه کردن

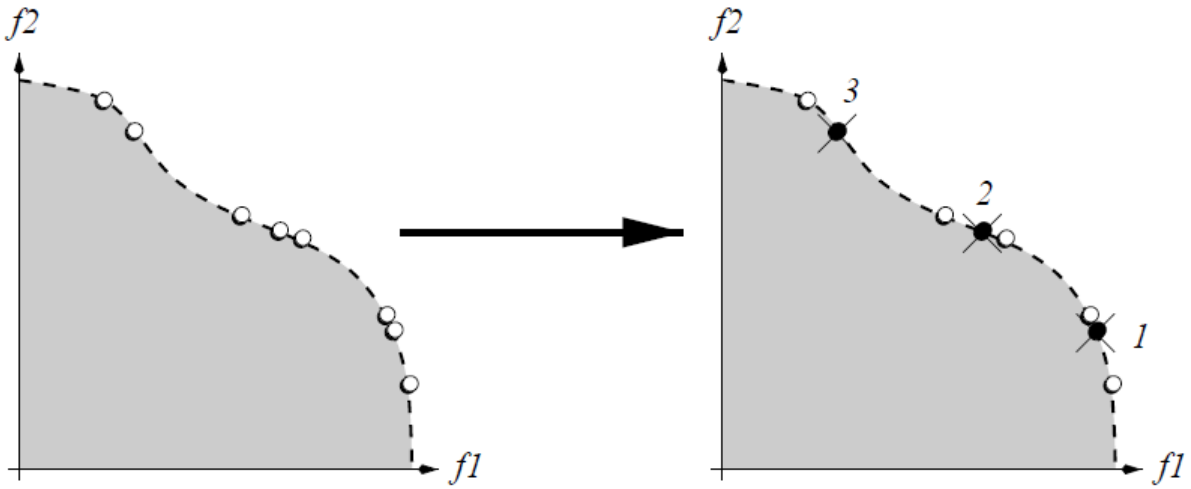
فرض کنید مقدار باقی‌مانده در حافظه هارمونی ۱ برابر F باشد $(|HM_1| = F)$ و ما بخواهیم این اندازه را به اندازه $T < F$ برسانیم، فرآیند کوتاه کردن به‌صورت زیر عمل می‌کند:
در هر تکرار یک بردار i از HM_1 حذف می‌کنیم به گونه‌ای که $i \leq_d j$ برای هر $j \in HM_1$ که در آن

$$i \leq_d j \longleftrightarrow \forall 0 < k < F : \quad {\delta_i}^k = {\delta_j}^k$$

یا

$$\exists 0 < k < F : \left[\left(\forall 0 < l < k : \quad {\delta_i}^l = {\delta_j}^l\right) \wedge {\delta_i}^k < {\delta_j}^k\right]$$

که در آن ${\delta_i}^k$ فاصله بردار i تا k–امین نزدیک‌ترین همسایه‌اش در HM_1 است. در واقع در هر مرحله برداری برای حذف کردن انتخاب می‌شود که مینیمم فاصله تا بردار دیگری را دارد. اگر چندین بردار این خاصیت را داشتند، دومین فاصله را در نظر می‌گیریم و این فرآیند تکرار می‌شود. در شکل ۸.۳ فرآیند کوتاه کردن نشان داده شده است [۵۴].



شکل ۸.۳: مثالی از فرآیند کوتاه کردن در الگوریتم SPEA۲.

۷.۳ نتایج محاسباتی

برای ارزیابی الگوریتم‌های هارمونی چندهدفه مطرح‌شده، یک مجموعه از شش تابع آزمون[۳۹] با ویژگی‌هایی مانند تحدب[۴۰]، عدم‌تحدب[۴۱]، گسستگی[۴۲] و غیریکنواختی[۴۳] انتخاب شدند. این توابع، مینیمم‌سازی دو هدف را بررسی کرده و به عنوان ''توابع ZDT '' شناخته می‌شوند که ZDT برگرفته از اسامی معرفان آنها، یعنی Zitzler ، Deb و Thiele [۵۶] می‌باشد.

هر یک از این شش تابع، ZDT۱، ZDT۲، ZDT۳، ZDT۴، ZDT۵ و ZDT۶ برای مقایسه‌ی دو الگوریتم هارمونی چندهدفه پیشنهاد شده با الگوریتم تکاملی شناخته‌شده NSGA-II (الگوریتم ژنتیک دسته‌بندی غیرمغلوب نوع دو [۴۴]) معرفی شده توسط دب و همکاران[۴۵] [۱۰] و موجود در [۱۲] مورد استفاده قرار گرفتند. هر یک از توابع تست، خود شامل سه تابع $f_۱$ ، g و h می‌باشند که در ادامه تعریف می‌شوند.

$$\begin{cases} min \quad \tau(\mathbf{x}) = (f_۱(x_۱), f_۲(\mathbf{x})) \\ s.t. \quad f_۲(\mathbf{x}) = g(x_۲, \ldots, x_n)h(f_۱(x_۱)g(x_۲, \ldots, x_n)) \\ , \quad \mathbf{x} = (x_۲, \ldots, x_n) \end{cases} \tag{۱۷.۳}$$

تابع $f_۱$ فقط از اولین متغیر تصمیم تشکیل می‌شود، تابع g از $n-۱$ متغیر باقی‌مانده و پارامترهای h مقادیر توابع $f_۱$ و g هستند. توابع تست در این سه تابع و همچنین در تعدادی از متغیرها و در مقادیر متغیرها ممکن است متفاوت باشد. اما قبل از اینکه این توابع را معرفی کنیم، نیاز به معرفی مفاهیمی

[۳۹] Test Functions
[۴۰] Convexity
[۴۱] Non-convexity
[۴۲] Discreteness
[۴۳] Non-uniformity
[۴۴] Nondominated Sorting Genetic Algorithm II
[۴۵] Deb et al.

داریم که در این توابع از آنها استفاده می‌شود.

تعریف ۱.۷.۳. یک مجموعه از بردارهای تصمیم $X' \subseteq X$ را در نظر بگیرید. مجموعه X' یک مجموعه بهینه پارتو موضعی (محلی) نامیده می‌شود اگر و تنها اگر

$$\forall a' \in X' : \nexists a \in X : a \prec a' \wedge \parallel a - a' \parallel < \varepsilon \quad \wedge \parallel f(a) - f(a') \parallel < \delta$$

که $\parallel . \parallel$ اندازه فاصله متناظر است، $\prec$ نماد مفهوم تسلطی ، $\varepsilon > ۰$ و $\delta > ۰$.

تعریف ۲.۷.۳. مجموعه X' یک مجموعه بهینه پارتو سراسری است اگر و تنها اگر

$$\forall a' \in X' : \nexists a \in X : a \prec a'$$

باید توجه داشت که یک مجموعه بهینه پارتو سراسری لزوماً شامل همه جواب‌های بهینه پارتو نیست. مجموعه مربوطه از بردارهای هدف به عنوان مرز بهینه پارتو نشان داده می‌شود. حال به معرفی شش تابع تست فوق می‌پردازیم [۵۶].

- تابع تست $\tau_۱$ یک مرز بهینه پارتو محدب دارد.

$$\begin{cases} f_۱(x_۱) = x_۱ \\ g(x_۲, \ldots, x_n) = ۱ + ۹.\sum_{i=۲}^{n} \frac{x_i}{n-۱} \\ h(f_۱, g) = ۱ - \sqrt{\frac{f_۱}{g}} \end{cases} \qquad (۱۸.۳)$$

مرز بهینه پارتو به‌وسیله $g(\mathbf{x}) = ۱$ تشکیل شده است و $n = ۳۰$ و $x_i = [۰, ۱]$.

- تابع تست $\tau_۲$ همتای غیرمحدب تابع $\tau_۱$ است.

$$\begin{cases} f_۱(x_۱) = x_۱ \\ g(x_۲, \ldots, x_n) = ۱ + ۹.\sum_{i=۲}^{n} \frac{x_i}{n-۱} \\ h(f_۱, g) = ۱ - (\frac{f_۱}{g})^۲ \end{cases} \qquad (۱۹.۳)$$

که در آن $n = ۳۰$ و $x_i = [۰, ۱]$ ، مرز بهینه پارتو به‌وسیله $g(\mathbf{x}) = ۱$ تشکیل شده است.

- تابع تست $\tau_۳$ نشان دهنده ویژگی جدایی پذیری است. مرز بهینه پارتوی آن شامل چندین بخش محدب ناپیوسته است.

$$\begin{cases} f_۱(x_۱) = x_۱ \\ g(x_۲, \ldots, x_n) = ۱ + ۹.\sum_{i=۲}^{n} \frac{x_i}{n-۱} \\ h(f_۱, g) = ۱ - \sqrt{\frac{f_۱}{g}} - (\frac{f_۱}{g})\sin(۱۰\pi f_۱) \end{cases} \qquad (۲۰.۳)$$

که در آن $n = ۳۰$ و $x_i = [۰, ۱]$ و مرز بهینه پارتو به‌وسیله $g(\mathbf{x}) = ۱$ تشکیل شده است. معرفی تابع مثلثاتی در تابع h سبب ناپیوستگی بودن در مرز بهینه پارتو می‌شود. هر چند در فضای پارامتری هیچ ناپیوستگی وجود ندارد.

- تابع تست $\tau_۴$ شامل ۲۱[۹] مرز بهینه پارتو موضعی است و تست‌ها برای الگوریتم‌های تکاملی توانایی مقابله با ویژگی چند مودالی را دارند.

$$\begin{cases} f_۱(x_۱) = x_۱ \\ g(x_۲, \ldots, x_n) = ۱ + ۱۰(n-۱) + \sum_{i=۲}^{n}(x_i^{۲} - ۱۰(\cos ۴\pi x_i)) \\ h(f_۱, g) = ۱ - \sqrt{\frac{f_۱}{g}} \end{cases} \qquad (۲۱.۳)$$

که $n = ۱۰$ ، $x_۱ \in [۰, ۱]$ و $x_۲, \ldots, x_m \in [-۵, ۵]$ مرز بهینه پارتو به وسیله $g(\mathbf{x}) = ۱$ تشکیل شده و بهترین مرز پارتو محلی با $g(\mathbf{x})$=۱.۲۵ است. توجه داشته باشید که کل مجموعه بهینه پارتو موضعی در فضای هدف قابل تشخیص نیست.

- تابع تست $\tau_۵$ یک مسئله گمراه کننده را شرح می‌دهد و آن را از توابع تست دیگر تشخیص می‌دهد که در آن x_i یک رشته دودویی را نمایش می‌دهد.

$$\begin{cases} f_۱(x_۱) = ۱ + u(x_۱) \\ g(x_۲, \ldots, x_n) = \sum_{i=۲}^{n} v(u(x_i)) \\ h(f_۱, g) = \frac{۱}{f_۱} \end{cases} \qquad (۲۲.۳)$$

که $u(x_i)$ تعداد یک‌ها در بردار بیت x_i را می‌دهد و

$$v(u(x_i)) = \begin{cases} ۲ + u(x_i) & if \quad u(x_i) < ۵ \\ ۱ & if \quad u(x_i) = ۵ \end{cases}$$

که $n = ۱۱$ ، $x_۱ \in \{۰, ۱\}^{۳۰}$ و $x_۲, \ldots, x_m \in \{۰, ۱\}^{۵}$. مرز بهینه پارتو با $g(\mathbf{x}) = ۱۰$ تشکیل شده است در حالی که بهترین مرز بهینه پارتو گمراه کننده به وسیله جواب‌هایی برای $g(\mathbf{x}) = ۱۱$ تشکیل می‌شود.

- تابع تست $\tau_۶$ شامل دو مشکل ناشی از غیر یکنواختی فضای جست‌وجو است: اولاً، جواب‌های بهینه پارتو در طول مرز بهینه پارتو سراسری غیر یکنواخت توزیع شده‌اند. ثانیاً، تراکم جواب‌ها از مرز بهینه پارتو دور است.

$$\begin{cases} f_۱(x_۱) = ۱ - \exp(-۴x_۱)\sin^{۶}(۶\pi x_۱) \\ g(x_۲, \ldots, x_n) = ۱ + ۹.(\frac{\sum_{i=۲}^{m} x_i}{n-۱})^{۰/۲۵} \\ h(f_۱, g) = ۱ - (\frac{f_۱}{g})^{۲} \end{cases} \qquad (۲۳.۳)$$

که $n = ۱۰$ و $x_i \in [۰, ۱]$. مرز بهینه پارتو به وسیله $g(\mathbf{x}) = ۱$ تشکیل می‌شود و غیرمحدب است.

معیارهای عملکرد

مقایسه روش‌های بهینه‌سازی مختلف تجربی، مستلزم مفهوم عملکرد (کارایی) [۴۶] است. در مورد بهینه‌سازی چند هدفه، تعریف کیفیت مهم‌تر از مسائل بهینه‌سازی تک‌هدفه است؛ زیرا، هدف بهینه‌سازی خود شامل چندین هدف است.

- فاصله بین مرز پارتو به‌دست آمده با مرز پارتو بهینه باید مینیمم شود.
- یک توزیع خوب (در اکثر موارد یکنواخت) برای جواب پیدا شده، مورد نظر است. ارزیابی این ضابطه براساس یک معیار فاصله مشخص انجام می‌شود.
- وسعت مرز غیرتسلطی به‌دست آمده باید ماکزیمم شود، یعنی برای هر تابع هدف، دامنه وسیعی از مقادیر باید با جواب‌های غیرتسلطی پوشش داده شود [۵۶].

تعریف ۳.۷.۳. فرض کنید یک مجموعه از بردارهای تصمیم غیرمغلوب $X' \subseteq X$ و یک پارامتر همسایگی $\sigma > 0$ و یک متر فاصله $\| \cdot \|$ داده شده باشد. سه تابع برای ارزیابی کیفیت X' مربوط به فضای تصمیم را معرفی می‌کنیم.

۱. تابع M_1 میانگین فاصله X' تا مرز بهینه پارتو $\bar{X} \subseteq X$ را نشان می‌دهد.

$$M_1(X') := \frac{1}{|X'|} \sum_{a' \in X'} \min\{\| a' - \bar{a} \|; a' \in X'\}$$

۲. تابع M_2 توزیع نسبت به تعداد جواب‌های غیرمغلوب پیداشده را نشان می‌دهد.

$$M_2(X') := \frac{1}{|X'-1|} \sum_{a' \in X'} |\{b' \in X'; \|a' - b'\| > \sigma\}|$$

۳. تابع M_3 وسعت مرز توصیف شده به وسیله X' را بررسی می‌کند.

$$M_3(X') := \sqrt{\sum_{i=1}^{m} max\{\| a'_i - b'_i \|; a', b' \in X'\}}$$

به‌طور مشابه، سه متر M_1^*، M_2^* و M_3^* را روی فضای هدف معرفی می‌کنیم. فرض کنید $Y', \bar{Y} \subseteq Y$ به ترتیب مجموعه بردارهای هدف متناظر با X' و $\bar{X}$ باشند و σ^* و $\| \cdot \|^*$ همانند قبل تعریف شده باشند، آنگاه

۱. M_1^*: فاصله بین مرز پارتو محاسبه شده و مرز پارتو بهینه.

$$M_1^*(Y') := \frac{1}{|Y'|} \sum_{p' \in Y'} \min\{\| p' - \bar{p} \|^*; p' \in Y'\}$$

[۴۶] Performance

۲. $M_۲^*$: توزیع مرز پارتو محاسبه شده.

$$M_۲^*(Y') := \frac{۱}{\mid Y' - ۱ \mid} \sum_{p' \in Y'} \mid \{q' \in Y'; \|p' - q'\|^* > \sigma^*\}\mid$$

۳. $M_۳^*$: توسعه مرز پارتو محاسبه شده.

$$M_۳^*(Y') := \sqrt{\sum_{i=۱}^{n} max\{\| p'_i - q'_i \|^*; p', q' \in Y'\}}$$

$M_۱$ و $M_۱^*$ شهودی هستند در حالی‌که $M_۲$، $M_۲^*$، $M_۳$ و $M_۳^*$ نیاز به توضیح دارند. تابع‌های توزیع $M_۲$ و $M_۲^*$ یک مقدار در بازه $[۰, \mid X' \mid]$ و $[۰, \mid Y' \mid]$ دارند، نشان‌دهنده تعداد σ جایگاه (σ^* جایگاه) در X' (Y') می‌باشد. بدیهی است که هر چقدر که مقدار پارامتر توزیع بیشتر باشد، توزیع بهتری برای یک پارامتر توضیح خواهیم داشت. به عنوان مثال، $M_۲^*(Y') = |Y'|$ به این مفهوم است که برای هر بردار تابع هدف، هیچ بردار هدف دیگری در فاصله σ^* آن قرار ندارد. توابع $M_۳$ و $M_۳^*$ ماکزیمم وسعت در هر بعد را نشان می‌دهد که دامنه‌ای که مرز در آن توسعه داده شده است را تقریب می‌زند.

۱.۷.۳ ترکیب‌بندی پارامترهای الگوریتم‌های مقایسه شده

مقادیر انتخاب شده برای پارامترهای الگوریتم‌های هارمونی چندهدفه مطرح شده یعنی HMCR، PAR، HMS و FW براساس پیشنهادات یانگ[۴۷] [۷۴]، گائو [۶۶] و گیم [۲۵، ۲۳] انتخاب شده‌اند. مقادیر این متغیرها در جدول ۱.۳ آورده شده است. مقادیر انتخاب شده برای الگوریتم تکاملی NSGA-II یعنی اندازه‌ی جمعیت، احتمال تقاطع، احتمال جهش، اندیس توزیع برای تقاطع و اندیس توزیع برای جهش از [۴۳] انتخاب شده‌اند. مقادیر این متغیرها در جدول ۲.۳ نشان داده شده‌اند.

پارامتر	مقدار
$HMCR$	۰٫۹۵
PAR	۰٫۱
HMS	۱۰۰
$FW(ZDT۵)$	۴۰٪
FW برای ZDT های دیگر	۱٪

جدول ۱.۳: پارامترهای الگوریتم جست‌وجوی هارمونی چند هدفه

[۴۷] Yang

پارامتر	مقدار
اندازه جمعیت	۱۰۰
احتمال تقاطع	۰/۹
احتمال جهش برای ZDT۵	$\frac{۱}{b}$
احتمال جهش برای ZDT های دیگر	$\frac{۱}{n}$
اندیس توزیع برای تقاطع	۱۵
اندیس توزیع برای جهش	۲۰

جدول ۲.۳: پارامترهای الگوریتم NSGA-II، m تعداد متغیرهای مسئله و b جمع تعداد بیت‌های همه‌ی متغیرها (برای یک مسئله‌ی دودویی) است.

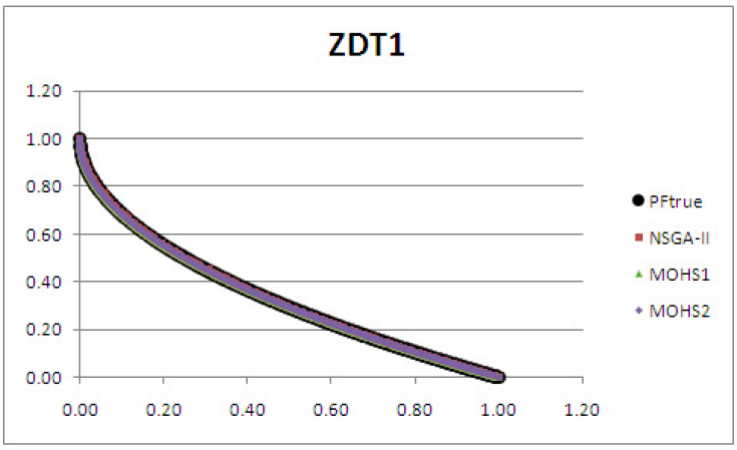
۲.۷.۳ نتایج

برای هر تابع تست، هر یک ازسه الگوریتم (MOHS۱، MOHS۲، NSGA-II) ۱۰ بار با ۱۰ ثانیه برای هر بار اجرا، اجرا شدند. نتایج تجربی در جداول ۳.۳، ۴.۳ و ۵.۳ نشان داده شده‌اند. این سه جدول مقادیر میانگین به‌دست آمده با همه ۱۰ اجرا را برای هر الگوریتم و هرتابع آزمون همراه با انحراف معیار (انحراف استاندارد) مربوطه (در پرانتز) نشان می‌دهند. باید ذکر کرد که این مقادیر با استفاده از مرز بهینه پارتو و طبق روش گفته شده در [۴۳، ۳۶] نرمال شده‌اند. در اینجا مقدار بهینه برابر یک بوده و هر چقدر مقدار فاصله از یک دورتر باشد، آن بدتر است. به علاوه شکل‌های ۱۴.۳ـ۹.۳ مقایسه‌ای بین مرزهای پارتو محاسبه شده برای هریک از سه الگوریتم و مرز بهینه پارتو دقیق[۴۸] (PF_{true}) هر تابع ZDT [۴۸] را نشان می‌دهد. همان گونه که در جدول ۳.۳ می‌بینید، هیچ یک از الگوریتم‌ها در همه توابع ZDT از بقیه الگوریتم‌ها بهتر نمی‌باشد. اما الگوریتم MOHS۱ به‌طور متوسط بهترین نتایج را داشته، بعد از آن MOHS۲ و در نهایت الگوریتم NSGA-II می‌باشد. برجسته‌ترین اختلاف برای تابع تست ZDT۵ قابل مشاهده می‌باشد که الگوریتم‌های MOHS۱ و MOHS۲ از نظر اندازه به‌وضوح برتر از NSGA-II می‌باشند که به‌طور قابل ملاحظه‌ای در میانگین نقش دارند. این اختلاف در شکل ۱۳.۳ بهتر قابل مشاهده می‌باشد. اگر چه تفاوت‌های عددی کوچکی برای مسائل دیگر وجود دارند، این اختلاف‌ها همان‌گونه که در شکل‌های ۹.۳، ۱۲.۳ و ۱۴.۳ مشاهده می‌شوند، به‌صورت ترسیمی ناچیز می‌باشند.

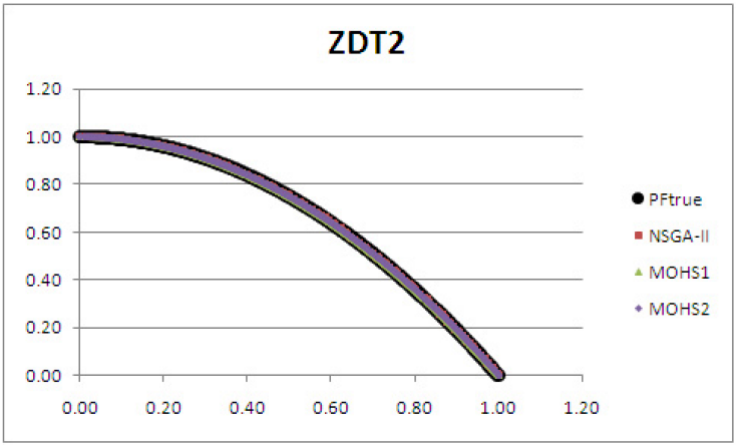
با بررسی مقادیر نشان داده شده در جدول ۴.۳ می‌توان مشاهده کرد که بهترین الگوریتم از نظر معیار توزیع $M_۲^*$، MOHS۲ می‌باشد و پس از آن با فاصله کمی NSGA-II بوده و MOHS۱ آخرین مکان را دارد. برای هرمسئله، اختلاف مقادیر خیلی کم می‌باشد، اگر چه مقدار MOHS۱ به‌طور متناوب برای همه‌ی مسائل مقادیر کمتری دارد. به‌طور ترسیمی، معیار توزیع روش مشابه با $M_۱^*$ رفتار می‌کند که در تصاویر ۱۴.۳ـ۹.۳ قابل مشاهده می‌باشد.

[۴۸] True Pareto Front

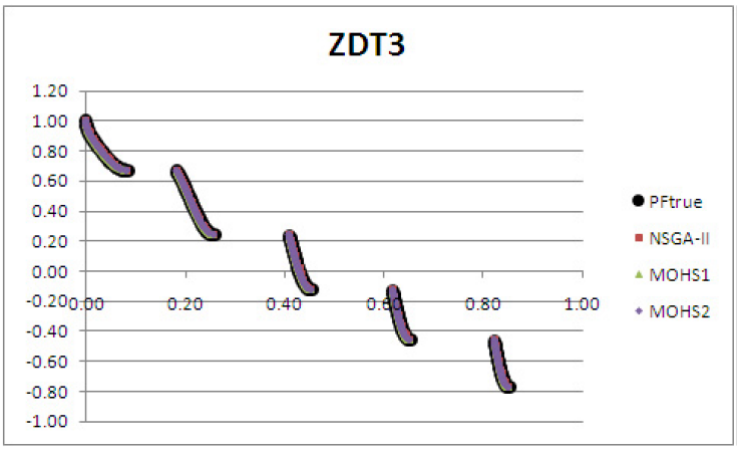
برای معیار گسترش (وسعت)، همان گونه که در جدول ۵.۳ نشان داده شده است، الگوریتم NSGA-II به‌طور میانگین بهترین بوده و پس از آن MOHS۲ و در آخر MOHS۱ می‌باشند. یک مقدار غیر عادی وجود دارد که بیشتر از یک می‌باشد. این به علت این است که گسترش مرز پارتو محاسبه شده از گسترش مرز پارتو بهینه نظری بیشتر می‌باشد که این به خصوص برای NSGA-II و ZDT۵ قابل مشاهده می‌باشد. این مورد، همان گونه که در شکل ۱۳.۳ قابل مشاهده می‌باشد، ناشی از این واقعیت است که NSGA-II یک جواب بسیار دور از بقیه جواب‌ها پیدا می‌کند که لزوماً مشخصه خوبی نیست. به‌طور خلاصه، هیچ کدام از الگوریتم‌ها برای هر مسئله و یا هر معیار بهتر از بقیه عمل نمی‌کند اما الگوریتم‌های پیشنهادی MOHS در مقایسه با گزینه‌ی کاملاً جدید مطرح، به‌نام NSGA-II بسیار رقابتی می‌باشند.



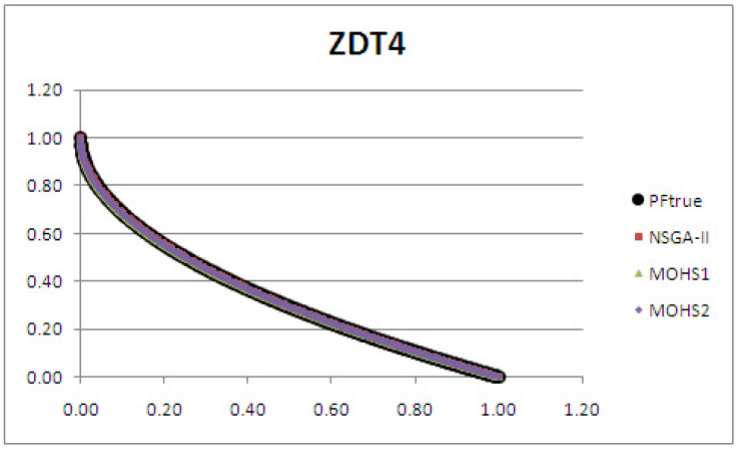
شکل ۹.۳: مقایسه بین مرزهای پارتو محاسبه شده سه الگوریتم و بهینه پارتو تابع ZDT۱.



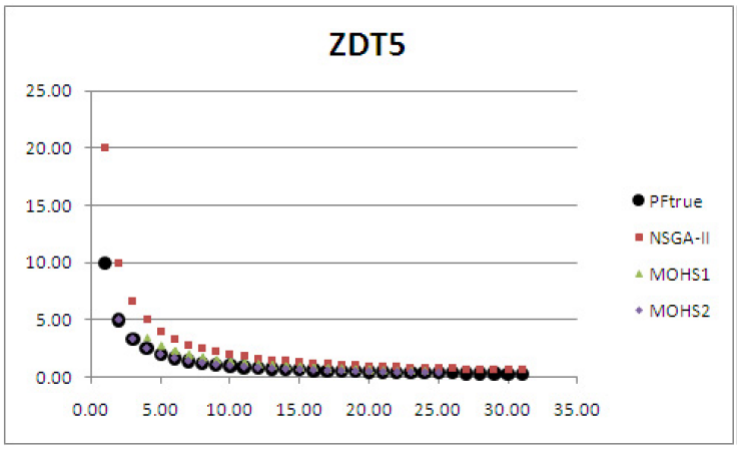
شکل ۱۰.۳: مقایسه بین مرزهای پارتو محاسبه شده سه الگوریتم و بهینه پارتو تابع ZDT۲.



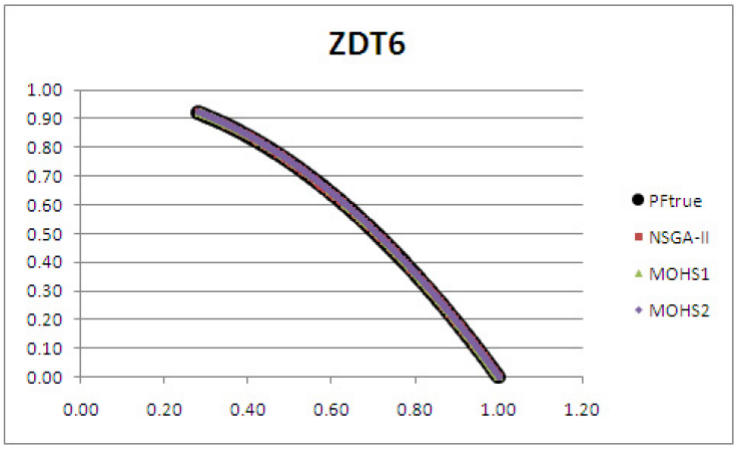
شکل ۱۱.۳: مقایسه بین مرزهای پارتو محاسبه شده سه الگوریتم و بهینه پارتو تابع ZDT۳.



شکل ۱۲.۳: مقایسه بین مرزهای پارتو محاسبه شده سه الگوریتم و بهینه پارتو تابع ZDT۴.



شکل ۱۳.۳: مقایسه بین مرزهای پارتو محاسبه شده سه الگوریتم و بهینه پارتو تابع ZDT۵.



شکل ۱۴.۳: مقایسه بین مرزهای پارتو محاسبه شده سه الگوریتم و بهینه پارتو تابع ZDT۶.

تابع	NSGA-II	MOHS۱	MOHS۲
$ZDT۱$	۰٫۹۹۱۸(۰٫۰۰)	۰٫۹۹۲۲(۰٫۰۰)	۰٫۹۹۲۲(۰٫۰۰)
$ZDT۲$	۰٫۹۹۵۷(۰٫۰۰)	۰٫۹۹۶۰(۰٫۰۰)	۰٫۹۹۶۲(۰٫۰۰)
$ZDT۳$	۰٫۹۵۷۱(۰٫۰۰)	۰٫۹۵۳۳(۰٫۰۰)	۰٫۹۵۲۹(۰٫۰۰)
$ZDT۴$	۰٫۹۹۵۹(۰٫۰۰)	۰٫۹۹۶۳(۰٫۰۰)	۰٫۹۶۸۸(۰٫۰۴)
$ZDT۵$	۰٫۰۸۵۸(۰٫۰۶)	۰٫۶۲۲۳(۰٫۰۰)	۰٫۶۳۱۴(۰٫۱۶)
$ZDT۶$	۰٫۹۴۲۰(۰٫۰۰)	۰٫۹۵۰۰(۰٫۰۰)	۰٫۹۳۸۵(۰٫۰۰)
میانگین (σ)	۰٫۸۲۸۱(۰٫۳۶)	۰٫۹۱۸۴(۰٫۱۵)	۰٫۹۱۳۳(۰٫۱۴)

جدول ۳.۳: نتایج میانگین به‌دست آمده برای متر $M_۱^*$ نرمال شده: فاصله بین مرز پارتو محاسبه‌شده و مرز بهینه پارتو دقیق (انحراف استاندارد در پرانتزها نشان داده شده‌اند).

تابع	NSGA-II	MOHS۱	MOHS۲
$ZDT۱$	۰٫۸۲۶۲(۰٫۰۰)	۰٫۸۲۲۵(۰٫۰۰)	۰٫۸۲۶۱(۰٫۰۰)
$ZDT۲$	۰٫۸۲۴۷(۰٫۰۰)	۰٫۸۱۰۷(۰٫۰۰)	۰٫۸۲۶۳(۰٫۰۰)
$ZDT۳$	۰٫۸۳۴۶(۰٫۰۰)	۰٫۸۳۱۲(۰٫۰۰)	۰٫۸۳۷۵(۰٫۰۰)
$ZDT۴$	۰٫۸۲۷۱(۰٫۰۰)	۰٫۸۱۹۱(۰٫۰۰)	۰٫۸۲۵۸(۰٫۰۴)
$ZDT۵$	۰٫۸۰۲۲(۰٫۰۶)	۰٫۷۷۸۷(۰٫۰۳)	۰٫۸۲۰۴(۰٫۰۲)
$ZDT۶$	۰٫۸۲۴۱(۰٫۰۰)	۰٫۶۷۱۹(۰٫۰۳)	۰٫۸۲۲۵(۰٫۰۰)
میانگین(σ)	۰٫۸۲۳۲(۰٫۰۱)	۰٫۷۸۹۰(۰٫۰۶)	۰٫۸۲۶۴(۰٫۰۱)

جدول ۴.۳: نتایج میانگین به‌دست آمده برای متر $M_۲^*$ نرمال شده : توزیع مرز پارتو محاسبه شده (انحراف استاندارد در پرانتزها نشان داده شده‌اند.)

تابع	NSGA-II	MOHS۱	MOHS۲
$ZDT۱$	۱٫۰۰۰۰(۰٫۰۰)	۰٫۹۹۴۳(۰٫۰۰)	۱٫۰۰۰۰(۰٫۰۰)
$ZDT۲$	۱٫۰۰۰۰(۰٫۰۰)	۰٫۹۹۳۶(۰٫۰۱)	۱٫۰۰۰۰(۰٫۰۰)
$ZDT۳$	۱٫۰۰۰۰(۰٫۰۰)	۰٫۹۹۸۳(۰٫۰۰)	۰٫۹۹۹۹(۰٫۰۰)
$ZDT۴$	۱٫۰۰۰۰(۰٫۰۰)	۰٫۹۹۱۶(۰٫۰۱)	۱٫۰۰۸۷(۰٫۰۱)
$ZDT۵$	۱٫۱۱۵۳(۰٫۰۰)	۰٫۷۲۲۴(۰٫۰۳)	۰٫۸۱۴۲(۰٫۰۴)
$ZDT۶$	۰٫۹۹۹۹(۰٫۰۰)	۱٫۰۰۰۰(۰٫۰۰)	۱٫۰۰۰۰(۰٫۰۰)
میانگین(σ)	۱٫۰۱۹۲(۰٫۰۵)	۰٫۹۵۰۰(۰٫۱۱)	۰٫۹۷۰۵(۰٫۰۸)

جدول ۵.۳: نتایج میانگین به‌دست آمده برای متر $M_۳^*$ نرمال شده: گسترش مرز پارتو محاسبه‌شده (انحراف استاندارد در پرانتزها نشان داده شده‌اند.)

فصل ۴

یک الگوریتم جست‌وجوی هارمونی چندهدفه براساس واریانس حافظه هارمونی

۱.۴ مقدمه

اگر چه الگوریتم جست‌وجوی هارمونی مزایای زیادی در حل مسائل بهینه‌سازی نشان داده است، اما پارامترهای آن باید براساس تجربه و ویژگی‌های مسئله توسط کاربران تخصیص داده شوند. این امر موجب مشکلات زیادی برای کاربران مبتدی می‌شود. به منظور غلبه بر این مشکل، در این فصل یک الگوریتم جست‌وجوی هارمونی چندهدفه خود انطباق (SAMOHS[۱]) براساس واریانس حافظه هارمونی مطرح می‌شود [۹]. در الگوریتم SAMOHS یک پهنای باند خود انطباق اصلاح شده به‌کار رفته است. به علاوه تنظیم پارامتر خود انطباق براساس تنوع واریانس حافظه هارمونی، برای HMCR و PAR مطرح می‌شود. برای حل مسائل بهینه‌سازی چندهدفه، الگوریتم SAMOHS پیشنهاد شده از روش کوتاه کردن [۲] [۱۰] و مرتب سازی غیرتسلطی [۳] [۵۴] برای به‌روز رسانی حافظه هارمونی استفاده می‌کند. برای نشان دادن کارایی الگوریتم SAMOHS این الگوریتم با مسائل معیاری زیادی تست شده است و برای حل یک مسئله بهینه‌سازی مهندسی به‌کار رفته است. نتایج محاسباتی نشان می‌دهند

[۱] Self-Adaptive Multi-Objective Harmony Search
[۲] Truncating
[۳] Non-dominated sorting

که SAMOHS قابل رقابت با الگوریتم‌های تکاملی چندهدفه دیگر (MOEAs[۴]) در اجرای تنوع[۵] و اجرای همگرایی می‌باشد. در عمل تأثیر اندازه حافظه هارمونی بر اجرای الگوریتم SAMOHS نیز تشریح می‌شود. در چند دهه گذشته، الگوریتم‌های تکاملی، توجهات زیادی را در میان محققان برای حل مسائل بهینه‌سازی چندهدفه به خود جلب کرده است. الگوریتم‌های تکاملی چندهدفه زیادی مانند NSGA-II [۱۰]، PESA-II [۵۹][۶]، SPEA۲ [۵۴][۷] و M-PAES [۳۴][۸] توسعه پیدا کرده‌اند. دلیل توسعه سریع الگوریتم‌های تکاملی چند هدفه این است که قادر به یافتن جواب‌های بهینه پارتو متعدد در تنها یک اجرا می‌باشند.

اخیراً تلاش‌های زیادی برای گسترش الگوریتم هارمونی به منظور حل مسائل بهینه‌سازی چندهدفه صورت گرفته است. در مرجع [۲۱] از الگوریتم هارمونی برای حل مسائل بهینه‌سازی چند هدفه با مبادله کردن[۹] زمان ـ هزینه[۱۰] استفاده کرده‌اند. الگوریتم ارائه شده دارای تنوع کمی از جواب‌های غیر تسلطی می‌باشد، زیرا تنها از مقایسه براساس تسلط استفاده کرده و مقایسه تنوع را نادیده می‌گیرد. الگوریتم جست‌وجوی هارمونی چند هدفه مطرح شده در مرجع [۴۷] قادر به همگرا شدن به جواب‌های بهینه پارتو می‌باشد. با این حال، نرخ همگرایی کمی داشته و نیاز به تکرار زیادی دارد. این نرخ همگرایی پایین را می‌توان با استفاده از یک الگوریتم جست‌وجوی هارمونی توسعه داده شده (IHS) [۳۸] رفع کرد که نرخ همگرایی آن بسیار بالا است. دو طرح مفصل برای اعمال الگوریتم هارمونی اصلی به منظور حل مسائل بهینه‌سازی چندهدفه توسط ریکارت و همکاران [۱۱] [۴۳] مطرح شد، با این حال این الگوریتم‌ها در برخی مواقع عملکرد ضعیفی از نظر همگرایی و تنوع دارند. قابل ذکر است که این دو طرح در فصل قبل پایان‌نامه بررسی شده‌اند. پاولسکی و همکاران[۱۲] [۳۷] چهار نوع الگوریتم هارمونی برای حل مسائل بهینه‌سازی چندهدفه پیشنهاد داده‌اند. اما، این چهار نوع قادر به تولید جواب‌های بهینه پارتو درست و به خوبی توزیع شده نمی‌باشند. اجرای ضعیف همه الگوریتم‌های هارمونی چندهدفه ذکر شده می‌تواند نشان دهنده این واقعیت باشد که این الگوریتم‌ها قابلیت‌های تحقیق ضعیفی برای مسائل بهینه‌سازی چندهدفه دارند. به‌علاوه، پارامترهای کنترلی این الگوریتم‌های هارمونی چند هدفه باید مطابق با ویژگی های مسئله و تجربه توسط کاربران تنظیم شوند. در نتیجه، این امر مسئولیت بزرگی برای کاربران جدید بوده و کاربرد و توسعه الگوریتم هارمونی چند هدفه را به تأخیر می‌اندازد. با در نظر گرفتن کمبودهای الگوریتم‌های هارمونی چند هدفه ذکر شده در بالا، یک الگوریتم جست‌وجوی هارمونی چند هدفه خود انطباق بر اساس واریانس حافظه هارمونی (SAMOHS) در این فصل ارائه شده است. تغییر واریانس جمعیت تأثیر زیادی بر توانایی اکتشافی الگوریتم هارمونی داشته [۶۹] و واریانس حافظه هارمونی تا اندازه ای منعکس کننده تنوع حافظه هارمونی می‌باشد [۴۲]. با الهام از این تحقیق‌ها، یک مکانیزم

---

[۴] Multi-Objective Evolutionary Algorithms
[۵] Diversity
[۶] Pareto Envelope-based Selection Algorithm-II
[۷] Strength Pareto Evolutionary Algorithm-II
[۸] Memetic Pareto Archived Evolution Strategy
[۹] Trade-Off
[۱۰] Time-Cost
[۱۱] Ricart et al.
[۱۲] Pavelski et al.

خود انطباق جدید بر اساس تغییر واریانس حافظه هارمونی در الگوریتم SAMOHS به کار گرفته شده است.
مکانیزم خود انطباق مطرح شده دارای ۳ پیشرفت اصلی می‌باشد:

۱. هر متغیر تصمیم دارای پارامترهای کنترلی مختص به خود می‌باشد که در طول فرآیند تحقیق به‌طور انطباقی به‌روز می‌شوند.

۲. یک تنظیم پارامتر خود انطباق بر اساس تغییر واریانس حافظه هارمونی برای HMCR و PAR به‌کار گرفته شده است.

۳. یک پهنای باند خود انطباق اصلاح شده (BW) برای جست‌وجوی هارمونی مطرح شده است.

با استفاده از مکانیزم خود انطباق، الگوریتم SAMOHS دارای سازگاری[۱۳] و پایداری[۱۴] بهتر می‌باشد. به علاوه قابلیت‌های جست‌وجوی موضعی و سراسری الگوریتم SAMOHS ارتقاء یافته اند و مسئولیت تنظیم پارامترها کمتر شده است. به منظور حل مسائل بهینه‌سازی چندهدفه یک روش کوتاه کردن و مرتب سازی غیر تسلطی برای به‌روز رسانی مؤثر حافظه هارمونی و حفظ تنوع جواب‌های غیر تسلطی در حافظه هارمونی مورد استفاده قرار گرفته است [۹].
در فصل سوم مسائل بهینه‌سازی چند هدفه و الگوریتم جست‌وجوی هارمونی در مسائل بهینه‌سازی چند هدفه ذکر شد. در ادامه این فصل انگیزه‌ها و چارچوب الگوریتم SAMOHS و نتایج محاسباتی مطرح می‌شوند.

۲.۴ الگوریتم SAMOHS پیشنهادی

هنگامی‌که الگوریتم جست‌وجوی هارمونی برای حل مسائل بهینه‌سازی چندهدفه به‌کار گرفته می‌شود، تنظیم کردن پارامترهای کنترلی آن سخت است. از یک طرف، اجرای الگوریتم هارمونی به مقادیر پارامترهای کنترلی به‌شدت حساس است. به این معنی که، تفاوت مقادیر پارامترها می‌تواند باعث اجرای متفاوت بر حسب دقت همگرایی، نرخ همگرایی و پایداری شود. از سوی دیگر، ثابت کردن مقادیر پارامترها برای حفظ تعادل بین تشدید و تنوع در طول فرآیند جست‌وجو نامناسب است. اگر چه استراتژی‌های تنظیم پارامترهای پویا[۱۵] و مکانیزم‌های خود انطباق متنوعی برای الگوریتم جست‌وجوی هارمونی تک هدفه پیشنهاد شده است، اما تحقیقات خیلی کمی برای الگوریتم هارمونی چند هدفه وجود دارد. به علاوه، یک استراتژی تنظیم پارامترهای پویا، هنوز برخی اشکالات مانند اینکه همه‌ی متغیرهای تصمیم پارامتر کنترلی در طول فرآیند جست‌وجو را استفاده می‌کنند، دارند [۳۸]. پیدا کردن یک تعادل خوب بین اکتشاف[۱۶] و استخراج[۱۷] مشکل است زیرا هر متغیر تصمیم مشخصات خودش را دارد. اکتشاف

[۱۳] Adaptability
[۱۴] Robustness
[۱۵] Dynamic
[۱۶] Exploration
[۱۷] Exploitation

یا تنوع، قسمت‌هایی از فضای جست‌وجو است که بررسی نشده‌اند و مورد جست‌وجو قرار می‌گیرند. اما، استخراج یا تشدید، محدوده‌ای است که در آن بهترین جواب پیدا شده و جست‌وجو بیشتر در محدوده امیدبخش نسبت به کل فضا انجام می‌شود.

با بررسی این مشکلات یک مکانیزم خود انطباق جدید برای الگوریتم هارمونی چند هدفه در این فصل پایان‌نامه پیشنهاد شده است. مکانیزم خود انطباق پیشنهادی، سه بهبود اصلی شامل موارد زیر را دارد:

۱. هر متغیر تصمیم پارامترهای کنترلی خودش را دارد که به‌طور انطباق در طول فرآیند جست‌وجو به روز می‌شوند.

۲. تنظیم پارامتر خود انطباق برای HMCR و PAR براساس تغییرات واریانس حافظه هارمونی به‌کار گرفته می‌شوند.

۳. یک BW خود انطباق اصلاح شده برای الگوریتم هارمونی پیشنهاد می‌شود.

جزییات الگوریتم SAMOHS پیشنهاد شده به شرح زیر است [۹].

۱.۲.۴ پیاده‌سازی الگوریتم SAMOHS

در این بخش الگوریتم SAMOHS پیشنهاد شده براساس واریانس حافظه هارمونی ارائه می‌شود. باید توجه شود که به روز کردن حافظه هارمونی برای مسائل بهینه‌سازی چند هدفه از تک هدفه متفاوت است. برای به روز کردن حافظه هارمونی در مسائل بهینه‌سازی چندهدفه حافظه هارمونی بعد از تولید هارمونی‌های جدید، به جای یک هارمونی، در هر تکرار به روز می‌شود. فرآیند اجرای الگوریتم SAMOHS پیشنهاد شده را می‌توان به صورت زیر خلاصه کرد.

۱. **مقداردهی اولیه پارامترهای الگوریتم.**
ماکزیمم تعداد تکرارها را T و شمارنده تکرار را $t = ۱$ قرار دهید، حافظه هارمونی اولیه HM_t را به‌طور تصادفی تولید کنید و همه هارمونی‌ها در HM_t را محاسبه کنید. کمترین مقادیر را به عنوان $HMCR_{min}$ و بیشترین مقدار را به عنوان $HMCR_{max}$ مشخص کنید. همچنین، برای PAR مقادیر $[PAR_{mean۱}, PAR_{std۱}]$ و $[PAR_{mean۲}, PAR_{std۲}]$، برای K مقادیر $[K_{min۱}, K_{max۱}]$ و $[K_{min۲}, K_{max۲}]$ و HMS را مشخص کنید.

۲. **به روز کردن پارامترهای کنترلی.**
مقادیر پارامترهای کنترلی $HMCR_{j,t}$، $PAR_{j,t}$، $BW_{j,t}$ را برای هر متغیر محاسبه کنید که طبق مکانیزم خود انطباق پیشنهاد شده در بخش ۲.۲.۴ اختصاص داده می‌شوند.

۳. **تولید یک هارمونی جدید.**
فرآیند تولید یک هارمونی جدید در الگوریتم ۱ را با استفاده از پارامترهای $HMCR_{j,t}$، $PAR_{j,t}$، $BW_{j,t}$ به اندازه حافظه هارمونی اجرا کنید و یک حافظه هارمونی جدید ${HM_t}^{new}$ تشکیل دهید.

الگوریتم ۱. فرآیند تولید (بداهه‌گویی) در الگوریتم جست‌وجوی هارمونی
برای $j = ۱, \ldots, n$ انجام دهید:
۱. اگر $rand() < HMCR$ آنگاه $x_{new,j} \in \{x_{۱,j}, x_{۲,j}, \ldots, x_{HMS,j}\}$ (بررسی حافظه)،
در غیر این صورت $x_{new,j} = LB_j + rand() * (UB_j - LB_j)$ (انتخاب تصادفی).
۲. اگر $rand() < PAR$ آنگاه $x_{new,j} = x_{new,j} \pm rand() * BW$ (تنظیم گام).
که $rand()$ یک عدد تصادفی تولید شده از یک توزیع یکنواخت در بازه $[۰, ۱]$ است.

۴. **تعیین مقدار برازش**
مقدار برازش هر هارمونی در HM_t^{new} را تعیین کنید.

۵. **به روز رسانی حافظه هارمونی**
جزئیات فرآیند به روز کردن حافظه هارمونی در بخش ۳.۲.۴ شرح داده می‌شود.

۶. **بررسی شرط توقف**
اگر به شرط $t \geq T$ برسیم، مجموعه A را برابر با مجموعه‌های غیرتسلطی $HM_{t+۱}$ قرار دهید و مجموعه غیرتسلطی A را استخراج کنید که الگوریتم پایان می‌پذیرد؛ در غیر این صورت، شمارنده تکرار را افزایش دهید $(t = t + ۱)$ و به مرحله دوم بروید.

$x_۱$	$x_۲$	...	x_j	...	x_n
$HMCR_{۱,t}$	$HMCR_{۲,t}$	...	$HMCR_{j,t}$	...	$HMCR_{n,t}$
$PAR_{۱,t}$	$PAR_{۲,t}$	...	$PAR_{j,t}$	...	$PAR_{n,t}$
$K_{۱,t}$	$K_{۲,t}$	...	$K_{j,t}$	...	$K_{n,t}$

جدول ۱.۴: قالب کدگذاری متغیر

۲.۲.۴ مکانیزم خود انطباق جدید برای الگوریتم SAMOHS

از آن‌جایی‌که پارامترهای HMCR، PAR و BW نقش اساسی روی عملکرد الگوریتم هارمونی ایفا می‌کنند و میزان کردن این پارامترها واقعاً سخت است، در این بخش یک مکانیزم خود انطباق جدید بر پایه تغییر واریانس حافظه هارمونی پیشنهاد شده است. لازم به تأکید است که تنظیم پارامتر خود انطباق پیشنهادی از الگوریتم تکامل تفاضلی تک هدفه[۱۸] [۵۱] الهام گرفته شده است. هر چند که تنظیم پارامتر خود انطباق پیشنهادی براساس واریانس حافظه هارمونی است. جزئیات مکانیزم خود انطباق پیشنهادی برای الگوریتم SAMOHS در زیر توصیف شده اند.

ابتدا، $HMCR_{j,t}$ ، $PAR_{j,t}$ و $K_{j,t}$ برای هر متغیر تصمیم x_j کدگذاری شده اند که در جدول ۱.۴ نشان داده شده است. در اینجا t تکرار جاری را نشان می‌دهد و $K_{j,t}$ یک ضریب متغیر است که برای

[۱۸] Single Objective Differential Evolution Algorithm

اصلاح فاصله پهنای باند، $BW_{j,t}$، استفاده می‌شود. سپس، پارامترهای $HMCR_{j,t}$، $PAR_{j,t}$ و $K_{j,t}$ در طول فرآیند جست‌وجو به‌طور انطباقی به‌صورت زیر به روز می‌شوند.

اگر تغییر VAR_j مورد ۱ یا مورد ۲ باشد، آنگاه $HMCR_{j,t}$، $PAR_{j,t}$ و $K_{j,t}$ به ترتیب با معادلات (۴.۴) ، (۵.۴) و (۶.۴) بازنویسی می‌شوند. در غیر این صورت، $HMCR_{j,t}$، $PAR_{j,t}$ و $K_{j,t}$ همگی با معادله (۱.۴) بازنویسی می‌شوند. در اینجا، مورد ۱ و مورد ۲ را به صورت زیر تعریف می‌کنیم:

مورد ۱: مقدار واریانس حافظه هارمونی VAR_j برای j مین متغیر، در α تکرار متوالی (در این فصل $\alpha = ۳$ به‌کارگرفته می‌شود) کاهش می‌یابد.

مورد ۲: مقدار واریانس حافظه هارمونی VAR_j برای j مین متغیر، در β تکرار متوالی (در این فصل $\beta = ۴$ به‌کارگرفته می‌شود) افزایش می‌یابد.

$$\begin{cases} HMCR_{j,t} = HMCR_{j,t-۱} \\ PAR_{j,t} = PAR_{j,t-۱} \\ K_{j,t} = K_{j,t-۱} \end{cases} \tag{۱.۴}$$

مقدار واریانس j مین متغیر در حافظه هارمونی جاری به صورت ریاضی می‌تواند به فرم زیر تعریف شود:

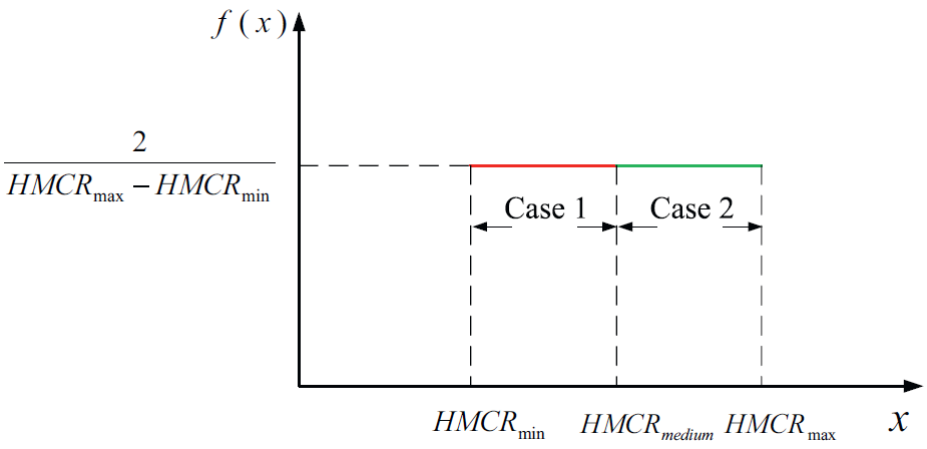
$$VAR_j = \frac{۱}{HMS} \sum_{i=۱}^{HMS} (x_{i,j} - \bar{x}_j)^۲ \tag{۲.۴}$$

که در آن،

$$\bar{x}_j = \frac{۱}{HMS} \sum_{i=۱}^{HMS} x_{i,j} \tag{۳.۴}$$

$\bar{x}_j$ مقدار میانگین حافظه هارمونی j مین متغیر است و $x_{i,j}$، مقدار j مین متغیر تصمیم از i مین هارمونی ذخیره شده در حافظه هارمونی است.

$HMCR_{j,t}$ به‌صورت معادله (۴.۴) محاسبه می‌شود و در شکل ۱.۴ نیز نشان داده شده است.



شکل ۱.۴: توزیع احتمال یکنواخت پیوسته برای $HMCR_{j,t}$.

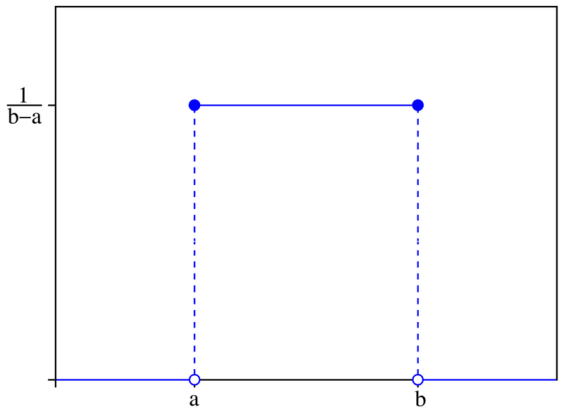
$$HMCR_{j,t} = \begin{cases} U[HMCR_{min}, HMCR_{medium}]; VAR_{j,t} < VAR_{j,t-۱} < \cdots < VAR_{j,t-\alpha} \\ U[HMCR_{medium}, HMCR_{max}]; VAR_{j,t} > VAR_{j,t-۱} > \cdots > VAR_{j,t-\beta} \end{cases} \quad (۴.۴)$$

که $U[U_{min}, U_{max}]$ تابع با توزیع یکنواخت با حد پایین U_{min} و حد بالا U_{max} را نشان می‌دهد و همچنین، $HMCR_{medium}$ مقدار متوسط بازه $[HMCR_{min}, HMCR_{max}]$، $VAR_{j,t}$ مقدار واریانس حافظه هارمونی j مین متغیر در تکرار جاری است و $VAR_{j,t-۱}$ مقدار واریانس حافظه هارمونی j مین متغیر در تکرار قبل را نشان می‌دهد.

تعریف ۱.۲.۴. تابع چگالی احتمال

تابع چگالی احتمال یک توزیع یکنواخت پیوسته چنین می‌باشد و نمودار آن نیز در شکل ۲.۴ نمایش داده شده است.

$$f(x) = U[a,b] = \begin{cases} \frac{۱}{b-a} & for \quad a \le x \le b \\ ۰ & for \quad o.w \end{cases}$$



شکل ۲.۴: تابع چگالی احتمال توزیع یکنواخت پیوسته.

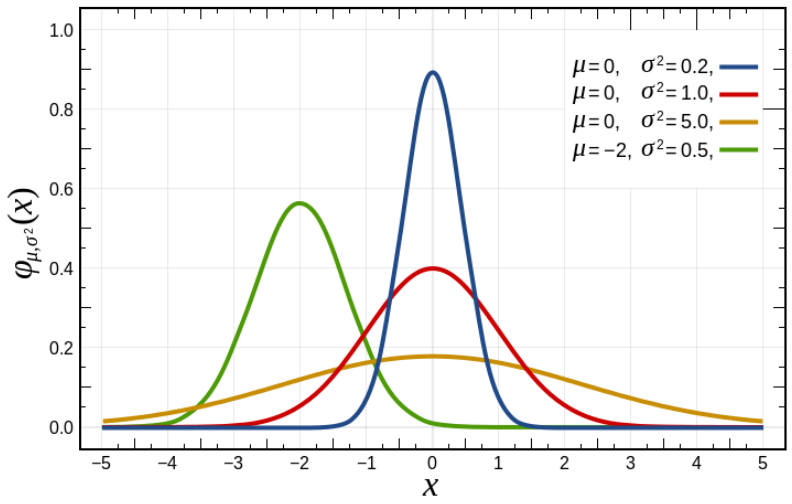
تذکر ۲.۲.۴. از معادله (۴.۴) و شکل ۱.۴ مشاهده می‌شود که تنظیم پارامتر برای $HMCR_{j,t}$ از انواع دیگر الگوریتم‌های جست‌وجوی هارمونی متفاوت است. در الگوریتم جست‌وجوی هارمونی اصلی، هنگامی که واریانس حافظه هارمونی به سرعت کاهش می‌یابد، الگوریتم جست‌وجوی هارمونی نمی‌تواند این روند را متوقف کند؛ زیرا مقدار پارامترهایش ثابت شده اند. در نتیجه، تنوع حافظه هارمونی به سرعت پایین می‌آید و الگوریتم هارمونی به بهینه‌ی موضعی گرفتار می‌شود. از طرف دیگر، اگر واریانس حافظه هارمونی به طور پیوسته افزایش یابد، این کار به جست‌وجوی محلی ضعیفی منجر خواهد شد. بنابراین، تنظیم پارامترها با کمک معادله (۴.۴) قادر است بر این مشکل غلبه کند. هنگامی‌که واریانس حافظه هارمونی به سرعت کاهش یابد، به $HMCR_{j,t}$ یک مقدار کم اختصاص خواهد یافت، که این مقدار کم $HMCR_{j,t}$ می‌تواند جواب‌های جدید بیشتری با اجرای عملیات انتخاب تصادفی تولید کند. در نتیجه، تنوع حافظه هارمونی افزایش می‌یابد. به روش مشابه، هنگامی‌که واریانس حافظه هارمونی

به سرعت افزایش می‌یابد، به $HMCR_{j,t}$ یک مقدار بزرگ اختصاص داده می‌شود که این مقدار بزرگ $HMCR_{j,t}$ باعث افزایش توانایی جست‌وجوی محلی می‌شود.

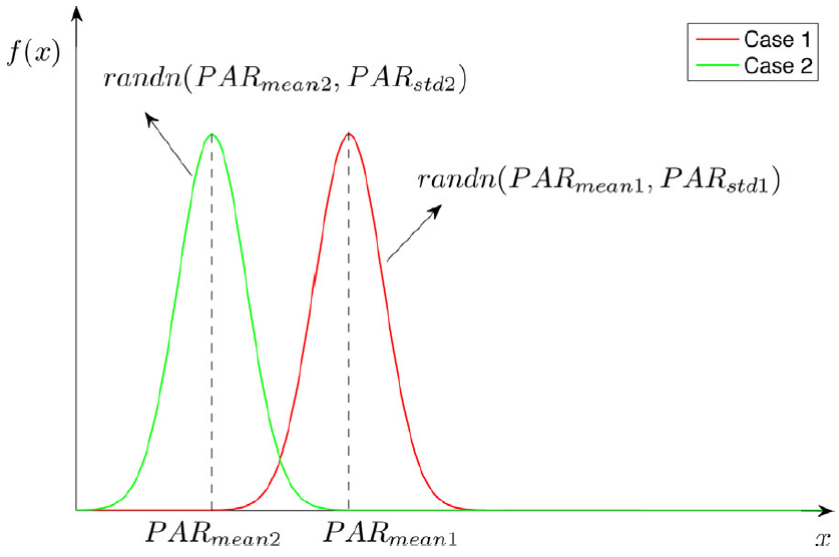
تعریف ۳.۲.۴. تابع توزیع نرمال

تابع چگالی احتمال توزیع نرمال با میانگین μ و واریانس $\sigma^۲$ به‌صورت زیر تعریف می‌شود.

$$f(x) = \frac{۱}{\sigma\sqrt{۲\pi}} \exp -\frac{(x-\mu)^۲}{۲\sigma^۲}$$



شکل ۳.۴: تابع چگالی احتمال توزیع نرمال.



شکل ۴.۴: توزیع احتمال نرمال پیوسته برای $PAR_{j,t}$.

$PAR_{j,t}$ به‌صورت زیر تنظیم می‌شود و در شکل ۴.۴ نشان داده شده است.

(۵.۴)

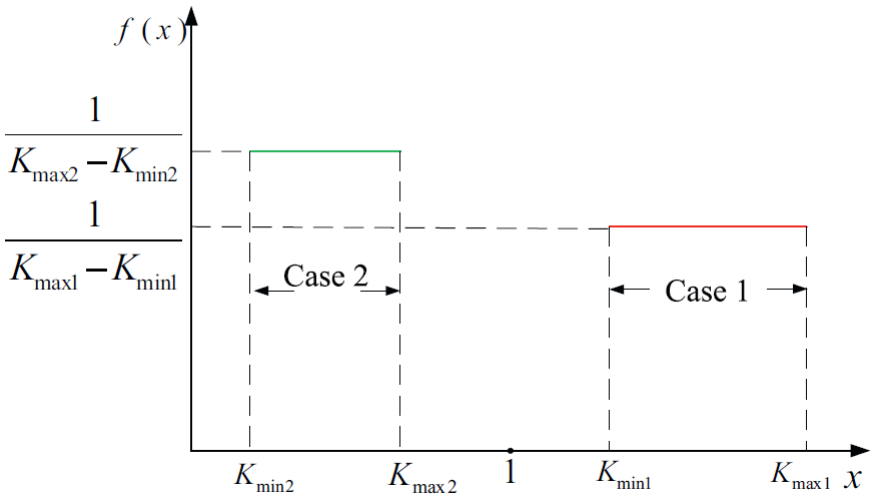
$$PAR_{j,t} = \begin{cases} randn(PAR_{mean۱}, PAR_{std۱}); VAR_{j,t} < VAR_{j,t-۱} < \cdots < VAR_{j,t-\alpha} \\ randn(PAR_{mean۲}, PAR_{std۲}); VAR_{j,t} > VAR_{j,t-۱} > \cdots > VAR_{j,t-\beta} \end{cases}$$

که $randn(PAR_{mean}, PAR_{std})$ نشان دهنده یک تابع توزیع نرمال با میانگین PAR_{mean} و انحراف استاندارد PAR_{std} است.

تذکر ۴.۲.۴. از معادله (۵.۴) و شکل ۴.۴ مشاهده می‌شود که تنظیم پارامتر برای $PAR_{j,t}$ شبیه $HMCR_{j,t}$ است. در مورد ۱، به $PAR_{j,t}$ یک مقدار نسبتاً بزرگ برای افزایش تنوع حافظه هارمونی اختصاص یافته است. در حالی‌که، در مورد ۲، به $PAR_{j,t}$ یک مقدار نسبتاً کوچک در جهت کاهش سرعت اختصاص داده شده است. بنابراین، تنظیم پارامتر $PAR_{j,t}$ برای نگهداری یک تعادل خوب بین تنوع و تشدید مفید است.

$K_{j,t}$ به‌صورت زیر تنظیم می‌شود و در شکل ۵.۴ نشان داده شده است.

$$K_{j,t} = \begin{cases} U[K_{min۱}, K_{max۱}]; VAR_{j,t} < VAR_{j,t-۱} < \cdots < VAR_{j,t-\alpha} \\ U[K_{min۲}, K_{max۲}]; VAR_{j,t} > VAR_{j,t-۱} > \cdots > VAR_{j,t-\beta} \end{cases} \quad (۶.۴)$$



شکل ۵.۴: توزیع احتمال یکنواخت پیوسته برای $K_{j,t}$.

که $U[,]$ یک تابع با توزیع یکنواخت همانند معادله (۴.۴) است.

در این فصل قرار می‌دهیم، $K_{min۱} = ۱/۵$ و $K_{max۱} = ۲$ و $K_{min۲} = ۰/۱$ و $K_{max۲} = ۰/۵$. به این معنی که به ضریب $K_{j,t}$ به ترتیب یک مقدار در محدوده‌ی $[۱/۵, ۲]$ و $[۰/۱, ۰/۵]$ برای مورد ۱ یا مورد ۲ اختصاص داده می‌شود. همان گونه که قبلاً ذکر شد، ضریب $K_{j,t}$ برای اصلاح BW به کار گرفته می‌شود. همانند HMCR و PAR هر متغیر تصمیم در طول فرآیند جست‌وجو BW خودش را دارد.

برای افزایش بیشتر توانایی جست‌وجوی هارمونی، BW خود انطباق اصلاح شده که در معادله (۷.۴) داده شده است، پیشنهاد می‌شود. BW برای هر متغیر به صورت زیر محاسبه می‌شود:

$$BW_{j,t} = K_{j,t} \times \sqrt{VAR_j} \qquad (۷.۴)$$

که $\sqrt{VAR_j}$ انحراف استاندارد j مین متغیر در حافظه هارمونی جاری و $BW_{j,t}$ نشان دهنده‌ی فاصله‌ی پهنای باند j مین متغیر برای تکرار جاری است.

تذکر ۵.۲.۴. موخوفادیای و همکاران [۱۹] [۶۹] نشان دادند که استفاده از انحراف استاندارد حافظه هارمونی جاری برای BW مفید است. به وضوح از معادلات (۶.۴) و (۷.۴) و شکل ۵.۴ مشاهده می‌شود، هنگامی‌که به ضریب $K_{j,t}$ یک مقدار نسبی بزرگتر از ۱ در مورد ۱ اختصاص داده می‌شود، $BW_{j,t}$ بزرگتر از $\sqrt{VAR_j}$ است. در حالی‌که در مورد ۲، $BW_{j,t}$ کوچکتر از $\sqrt{VAR_j}$ است وقتی‌که به ضریب $K_{j,t}$ یک مقدار نسبی، کمتر از ۱ اختصاص داده می‌شود. استفاده از یک BW بزرگ، هنگامی‌که واریانس حافظه هارمونی به سرعت کاهش می‌یابد، واقعاً مفید است. به همین طریق، BW کوچک وقتی استفاده می‌شود که واریانس حافظه هارمونی به سرعت افزایش می‌یابد. بنابراین توانایی اکتشاف و استخراج الگوریتم جست‌وجوی هارمونی با معرفی تنظیماتی برای پارامترهای $K_{j,t}$ افزایش می‌یابد.

تذکر ۶.۲.۴. از آنجایی که یک توزیع نرمال برای PAR استفاده می‌شود، مقادیر این پارامتر در اطراف یک مقدار مشخص در حین فرآیند جست‌وجو به‌شدت متمرکز شده است. این اطمینان را می‌دهد که عمل تنظیم گام می‌تواند در یک سطح نسبی پایدار نگه داشته شود که برای حفظ تنوع حافظه هارمونی مفید است. برای پارامتر HMCR یک توزیع یکنواخت انتخاب شده است. در مقایسه با توزیع نرمال، توزیع یکنواخت برای تنوع بخشیدن به HMCR مؤثرتر است و بنابراین از همگرایی نارس که تحت تمرکز HMCR اتفاق می‌افتد، جلوگیری می‌کند. برای پارامتر K نیز یک توزیع یکنواخت استفاده می‌شود. دلیل آن این است که K متنوع برای تنوع BW مفید است. تنوع مقدار BW برای تنوع حافظه هارمونی مفید است.

۳.۲.۴ به روزرسانی حافظه هارمونی

در ابتدا حافظه هارمونی جدید HM_t^{new} با حافظه‌ی هارمونی جاری HM_t به‌صورت $HM_t \cup HM_t^{new}$ ترکیب می‌شود که اندازه‌ی آن $HMS \times ۲$ است. سپس روی $HM_t \cup HM_t^{new}$ یک روش مرتب‌سازی غیرتسلطی اجرا می‌شود [۱۰]. (توجه شود که می‌توان از فرآیند مرتب‌سازی فونسه‌کا_فلمینگ که در فصل سوم به آن اشاره شد نیز استفاده کرد).

جزئیات فرآیند مرتب‌سازی غیرتسلطی به صورت زیر توصیف می‌شود:

الف) برای هر جواب p در $HM_t \cup HM_t^{new}$ دو مورد زیر را محاسبه کنید:

- شمارنده تسلطی n_p، یعنی تعداد جواب‌هایی که بر p مسلط هستند.

[۱۹] Mukhopadhyay et al.

- s_p یک مجموعه از جواب‌ها در $HM_t \cup HM_t^{new}$ که p بر آنها مسلط است.

ب) همه‌ی جواب‌های $HM_t \cup HM_t^{new}$ که شمارنده تسلطی آنها صفر است، متعلق به اولین مرز هستند.

پ) برای هر جواب p با $n_p = ۰$، هر عضو q از مجموعه‌ی s_p را انتخاب کنید و قرار بدهید:

۱ $n_q = n_q - ۱$. لذا هر عضو q با $n_q = ۰$ را در یک لیست جدای Q قرار بدهید. این اعضا در Q متعلق به مرز دوم هستند.
فرآیند فوق با هر عضو از Q ادامه می‌یابد و مرز سوم مشخص می‌شود. این فرآیند تا زمانی ادامه می‌یابد که همه‌ی مرزها شناخته شوند.
بعد از اینکه فرآیند مرتب‌سازی غیرتسلطی کامل شد، به هر هارمونی در $HM_t \cup HM_t^{new}$ یک رتبه اختصاص داده می‌شود. هارمونی‌های در $HM_t \cup HM_t^{new}$ به ترتیب رتبه بندی خود، به $HM_{t+۱}$ منتقل می‌شوند. برای انتخاب دقیق هارمونی‌ها به اندازه HMS از آخرین مرز غیرتسلطی، یک فرآیند کوتاه کردن در [۵۴] (که در فصل سوم به آن اشاره شد) استفاده شده است که به کاهش اندازه هارمونی‌ها منجر می‌شود تا اندازه آخرین مرز با فضای موجود در $HM_{t+۱}$ برابر شود. $HM_{t+۱}$ برای بداهه‌گویی بعدی استفاده می‌شود.

۳.۴ نتایج محاسباتی و تحلیل‌ها

در این بخش، چندین مسئله معیار برای ارزیابی الگوریتم SAMOHS پیشنهادی اجرا می‌شوند . ابتدا مقایسه بین SAMOHS با الگوریتم‌های تکاملی چند هدفه دیگر شرح داده و بحث می‌شود. ثانیاً مقایسه بین الگوریتم SAMOHS با الگوریتم‌های هارمونی چند هدفه دیگر انجام می‌شود. ثالثاً الگوریتم SAMOHS برای طراحی لوله‌های گرمایشی ماهواره به کار برده می‌شود و سرانجام میزان حساسیت الگوریتم SAMOHS نسبت به اندازه حافظه هارمونی بررسی می‌شود.

۱.۳.۴ مقایسه بین الگوریتم SAMOHS با الگوریتم‌های تکاملی چند هدفه دیگر

به منظور ارزیابی اجرای الگوریتم SAMOHS پیشنهاد شده، الگوریتم SAMOHS با سه الگوریتم کلاسیک NSGA-II [۱۰] ، SPEA۲ [۵۴] و MOPSO [۸] [۲۰] مقایسه می‌شود.

مسائل تست

در این بخش، چهارده مسئله تست کلاسیک که در الگوریتم‌های تکاملی چند هدفه به‌طور مکرر استفاده می‌شوند، انتخاب شده‌اند. همه آنها مسائل نامقید هستند که شامل پنج مسئله دو هدفه، ZDT۱-

[۲۰] Multi-Objective Particle Swarm Optimization

ZDT۴ و ZDT۶ [۵۶] و سه مسئله دو هدفه دیگر (برگرفته از شافر[۲۱]، فونسه‌کا[۲۲] و کورساو[۲۳] [۱۰])، و شش مسئله سه هدفه، DTLZ۱، DTLZ۲ و DTLZ۴-DTLZ۷ (بر گرفته از اسامی سازندگان آنها یعنی Deb، Thiele ، Laummans و Zitzler [۱۱]) می‌باشند. در فصل سوم، مسائل ZDT۱- ZDT۶ معرفی شدند. در این قسمت، مسائل دیگر را معرفی می‌کنیم. همه این توابع هدف، مینیمم می‌شوند.

- **مسئله شافر**

مسئله شافر یک مسئله بهینه‌سازی محدب است که به صورت (۸.۴) تعریف می‌شود:

$$\begin{cases} f_۱(x) = x^۲ \\ f_۲(x) = (x-۲)^۲ \end{cases} \tag{۸.۴}$$

که در آن کران‌های متغیر $[-۱۰^۳, ۱۰^۳]$ بوده و $x^* \in [۰, ۲]$ جواب‌های بهینه مسئله هستند.

- **مسئله فونسه‌کا**

مسئله فونسه‌کا، یک مسئله بهینه‌سازی غیرمحدب است.

$$\begin{cases} f_۱(x) = ۱ - \exp\left(-\sum_{i=۱}^{n}(x_i - \frac{۱}{\sqrt{n}})^۲\right) \\ f_۲(x) = ۱ - \exp\left(-\sum_{i=۱}^{n}(x_i + \frac{۱}{\sqrt{n}})^۲\right) \\ -۴ \le x_i \le ۴, \quad i = ۱, ۲, \ldots, n. \end{cases} \tag{۹.۴}$$

در مسئله فوق، $\forall i \quad x_i^* = \frac{-۱}{\sqrt{n}}$ ، نقطه مینیمم تابع $f_۱$ و $\forall i \quad x_i^* = \frac{۱}{\sqrt{n}}$ ، نقطه مینیمم تابع $f_۲$ است. مجموعه بهینه پارتو در $x_i^* = \left[\frac{-۱}{\sqrt{n}}, \frac{۱}{\sqrt{n}}\right]$ تشکیل شده است.

- **مسئله کورساو**

مسئله کورساو یک مسئله بهینه‌سازی غیرمحدب است که کران‌های متغیرها در بازه $[-۵, ۵]$ تغییر می‌کنند و به صورت زیر تعریف می‌شود.

$$\begin{cases} f_۱(x) = \sum_{i=۱}^{n-۱}\left(-۱۰ \exp\left(-۰/۲\sqrt{x_i^۲ + x_{i+۱}^۲}\right)\right) \\ f_۲(x) = \sum_{i=۱}^{n}\left(|x_i|^{۰/۸} + ۵\sin x_i^۳\right) \end{cases} \tag{۱۰.۴}$$

- **مسئله تست DTLZ۱**

[۲۱] Schaffer
[۲۲] Fonseca
[۲۳] Kursawe

$$\begin{cases} min & f_۱(x) = \frac{۱}{۲}x_۱x_۲\dots x_{M-۱}(۱+g(x_M)) \\ min & f_۲(x) = \frac{۱}{۲}x_۱x_۲\dots(۱-x_{M-۱})(۱+g(x_M)) \\ \vdots & \vdots \\ min & f_{M-۱}(x) = \frac{۱}{۲}x_۱(۱-x_۲)(۱+g(x_M)) \\ min & f_M(x) = \frac{۱}{۲}(۱-x_۱)(۱+g(x_M)) \\ s.t & ۰ \le x_i \le ۱, \forall i = ۱,۲,\dots,n, \\ & g(x_M) = ۱۰۰\left[|x_M| + \sum_{x_i \in X_M}(x_i - ۰/۵)^۲ - \cos(۲۰\pi(x_i - ۰/۵))\right]. \end{cases}$$

جواب بهینه پارتو، متناظر با $x_M = ۰$ است که در این مسئله، تعداد کل متغیرها برابر است با $n = M + k - ۱$.

- **مسئله تست DTLZ۲**

$$\begin{cases} min & f_۱(x) = (۱+g(x_M))\cos(\frac{\pi}{۲}x_۱)\cos(\frac{\pi}{۲}x_۲)\dots\cos(\frac{\pi}{۲}x_{M-۲})\cos(\frac{\pi}{۲}x_{M-۱}) \\ min & f_۲(x) = (۱+g(x_M))\cos(\frac{\pi}{۲}x_۱)\cos(\frac{\pi}{۲}x_۲)\dots\cos(\frac{\pi}{۲}x_{M-۲})\sin(\frac{\pi}{۲}x_{M-۱}) \\ min & f_۳(x) = (۱+g(x_M))\cos(\frac{\pi}{۲}x_۱)\cos(\frac{\pi}{۲}x_۲)\dots\sin(\frac{\pi}{۲}x_{M-۲}) \\ \vdots & \vdots \\ min & f_{M-۱}(x) = (۱+g(x_M))\cos(\frac{\pi}{۲}x_۱)\sin(\frac{\pi}{۲}x_۲) \\ min & f_M(x) = (۱+g(x_M))\sin(\frac{\pi}{۲}x_۱) \\ s.t & ۰ \le x_i \le ۱, \forall i = ۱,۲,\dots,n, \\ & g(x_M) = \sum_{x_i \in X_M}(x_i - ۰/۵)^۲. \end{cases}$$

جواب بهینه پارتو، متناظر با $x^*_M = ۰/۵$ است و همه مقادیر تابع هدف این مسئله باید در معادله (۱۱.۴) صدق کنند.

$$\sum_{i=۱}^{M}(f_i^*) = ۱ \qquad (۱۱.۴)$$

- **مسئله تست DTLZ۴**

در اینجا، پارامترهای $\alpha = ۱۰۰$ و $k = ۱۰$ پیشنهاد شده‌اند. همچنین، متغیرهای $x_۱$ تا $x_{M-۱}$

در بازه [۰ ,۱] تغییر می‌کنند.

$$\begin{cases} min & f_۱(x) = (۱ + g(x_M)) \cos(\frac{\pi}{۲} x_۱^{\alpha}) \cos(\frac{\pi}{۲} x_۲^{\alpha}) \dots \cos(\frac{\pi}{۲} x_{M-۲}^{\alpha}) \cos(\frac{\pi}{۲} x_{M-۱}^{\alpha}) \\ min & f_۲(x) = (۱ + g(x_M)) \cos(\frac{\pi}{۲} x_۱^{\alpha}) \cos(\frac{\pi}{۲} x_۲^{\alpha}) \dots \cos(\frac{\pi}{۲} x_{M-۲}^{\alpha}) \sin(\frac{\pi}{۲} x_{M-۱}^{\alpha}) \\ min & f_۳(x) = (۱ + g(x_M)) \cos(\frac{\pi}{۲} x_۱^{\alpha}) \cos(\frac{\pi}{۲} x_۲) \dots \sin(\frac{\pi}{۲} x_{M-۲}^{\alpha}) \\ \vdots & \vdots \\ min & f_{M-۱}(x) = (۱ + g(x_M)) \cos(\frac{\pi}{۲} x_۱^{\alpha}) \sin(\frac{\pi}{۲} x_۲^{\alpha}) \\ min & f_M(x) = (۱ + g(x_M)) \sin(\frac{\pi}{۲} x_۱^{\alpha}) \\ s.t & ۰ \le x_i \le ۱, \forall i = ۱, ۲, \dots, n, \\ & g(x_M) = \sum_{x_i \in X_M} (x_i - ۰/۵)^۲. \end{cases}$$

- **مسئله تست DTLZ۵**

تابع g با متغیرهای $k = |x_M| = ۱۰$ پیشنهاد شده است. در این مسئله، تعداد متغیرهای تصمیم به اندازه $n = M + k - ۱$ وجود دارد.

$$\begin{cases} min & f_۱(x) = (۱ + g(x_M)) \cos(\frac{\pi}{۲} \theta_۱) \cos(\frac{\pi}{۲} \theta_۲) \dots \cos(\frac{\pi}{۲} \theta_{M-۲}) \cos(\frac{\pi}{۲} \theta_{M-۱}) \\ min & f_۲(x) = (۱ + g(x_M)) \cos(\frac{\pi}{۲} \theta_۱) \cos(\frac{\pi}{۲} \theta_۲) \dots \cos(\frac{\pi}{۲} \theta_{M-۲}) \sin(\frac{\pi}{۲} \theta_{M-۱}) \\ min & f_۳(x) = (۱ + g(x_M)) \cos(\frac{\pi}{۲} \theta_۱) \cos(\frac{\pi}{۲} \theta_۲) \dots \sin(\frac{\pi}{۲} \theta_{M-۲}) \\ \vdots & \vdots \\ min & f_{M-۱}(x) = (۱ + g(x_M)) \cos(\frac{\pi}{۲} \theta_۱) \sin(\frac{\pi}{۲} \theta_۲) \\ min & f_M(x) = (۱ + g(x_M)) \sin(\frac{\pi}{۲} \theta_۱) \\ \theta_i = \frac{\pi}{۱ + ۴ g(r)} (۱ + ۲ g(r) x_i) \forall i = ۲, ۳, \dots, (M - ۱) \\ s.t & ۰ \le x_i \le ۱, \forall i = ۱, ۲, \dots, n, \quad \theta_i = \frac{\pi}{۲} x_i, \forall i = ۱, ۲, \dots, (M - ۱) \\ g(x_M) = \sum_{x_i \in X_M} (x_i - ۰/۵)^۲, & g(r) = {x_M}^۲. \end{cases}$$

- **مسئله تست DTLZ۶**

مسئله تست DTLZ۶ با اصلاح تابع g در مسئله تست DTLZ۵ سخت‌تر شده است. با این حال، در مسئله DTLZ۶ تابع g به صورت زیر تعریف می‌شود.

$$g(x_M) = \sum_{x_i \in X_M} {x_i}^{۰/۱}$$

اندازه بردار x_M برابر ۱۰ انتخاب شده است و تعداد کل متغیرهای مسئله با تعداد کل متغیرها در مسئله تست DTLZ۵ یکسان است.

- **مسئله تست DTLZ۷**

این مسئله تست یک مجموعه گسسته از ناحیه بهینه پارتو دارد که تعداد آنها در فضای جست‌وجو

$۲^{M-۱}$ است.

$$\begin{cases} min \quad f_۱(x_۱) = x_۱ \\ min \quad f_۲(x_۲) = x_۲ \\ \quad \vdots \quad \quad \vdots \\ min \quad f_{M-۱}(x_{M-۱}) = x_{M-۱} \\ min \quad f_M(x)(۱ + g(x_M))h(f_۱, f_۲, \ldots, f_{M-۱}, g) \\ s.t \quad g(x_M) = ۱ + \frac{۹}{|x_M|} \sum_{x_i \in X_M} x_i, \\ h(f_۱, f_۲, \ldots, f_{M-۱}, g) = M - \sum_{i=۱}^{M-۱} \left[\frac{f_i}{۱+g}(۱ + \sin(۳\pi f_i)) \right], \\ ۰ \leq x_i \leq ۱, \forall i = ۱, ۲, \ldots, n. \end{cases}$$

شاخص‌های اجرایی (توابع ارزیابی کیفیت جواب‌ها)

در این فصل، سه شاخص کیفیت که به‌طور مکرر در الگوریتم‌های تکاملی چندهدفه برای ارزیابی اجرای الگوریتم‌های بهینه‌سازی چندهدفه استفاده می‌شوند، به‌کارگرفته می‌شوند. شاخص‌های کیفیت شامل فاصله تولیدی (GD[۲۴]) [۷۲] ، ابرحجم (HV[۲۵]) [۵۵] و گستردگی (Δ[۲۶]) هستند.
شاخص همگرایی GD به عنوان معیاری برای اندازه گیری میزان نزدیکی پاسخ‌های غیرتسلطی پیدا شده به جواب‌هایی که در مجموعه بهینه پارتو هستند، معرفی شده و به‌صورت زیر محاسبه می‌شود:

$$GD = \frac{۱}{n} \left(\sum_{i=۱}^{n} d_i^p \right)^{\frac{۱}{p}} \qquad (۱۲.۴)$$

که n تعداد پاسخ‌های یافته شده غیرتسلطی و d_i فاصله اقلیدسی بین جواب i ام روی مرز پارتوی تقریبی و نزدیک‌ترین جواب به آن در مرز پارتو اصلی است. دراین فصل فرض می‌کنیم $p = ۲$. واضح است که مقدار $GD = ۰$ نشان می‌دهد که همه عناصر تولید شده در مرز پارتو هستند. یک مقدار کوچک‌تر از شاخص GD همگرایی بهتر به مرز پارتو را نشان می‌دهد. (توجه شود که شاخص GD همان شاخص $M_۱$ است که در فصل قبلی معرفی شد). شاخص گستردگی (Δ) [۱۰] برای اندازه‌گیری میزان پراکندگی جواب‌های غیرتسلطی به‌دست آمده است.

$$\Delta = \frac{d_f + d_l + \sum_{i=۱}^{N-۱} |d_i - \bar{d}|}{d_f + d_l + (N-۱)\bar{d}} \qquad (۱۳.۴)$$

که d_f و d_l فواصل اقلیدسی (اندازه در فضای هدف) بین جواب‌های رأسی (نقاط روی مرز پارتو واقعی که یکی از مؤلفه‌های آنها کمترین مقدار است و در فضای دو بعدی نقاط ابتدا و انتهای مرز پارتو هستند) و جواب‌های مرزی مجموعه جواب‌های غیرتسلطی به‌دست آمده می‌باشند، N تعداد جواب‌های غیرتسلطی

[۲۴] Generational Distance
[۲۵] HyperVolume
[۲۶] Spread

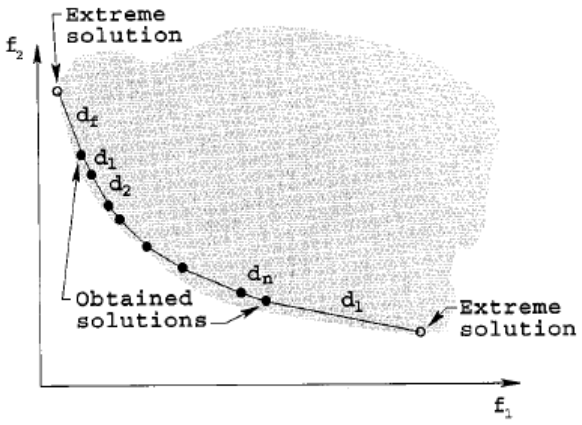
به‌دست آمده، d_i فاصله‌ی اقلیدسی بین جواب‌های مجاور در مجموعه جواب‌های غیرتسلطی به‌دست آمده و $\bar{d}$ میانگین همه d_i ها است. مقدار صفر برای این شاخص نشان می‌دهد که همه اعضا از مجموعه بهینه پارتو دارای فاصله مساوی هستند. یک مقدار کوچک‌تر از این شاخص توزیع بهتر و تنوع جواب‌های غیرتسلطی را نشان می‌دهد. به هر حال، شاخص فوق فقط برای مسائل دو هدفه کار می‌کند و نمی‌توان به‌طور مستقیم برای ارزیابی مسائل بیش از دو هدف استفاده کرد. این شاخص برای مسائل با بیش از دو هدف با محاسبه فاصله از یک نقطه داده شده به نزدیک‌ترین همسایه گسترش داده می‌شود. شاخص Δ تعریف شده در [۷۵] استفاده می‌شود که به‌صورت زیر می‌باشد.

$$\Delta = \frac{\sum_{i=1}^{m} d(E_i, \Omega) + \sum_{X \in \Omega} |d(X, \Omega) - \bar{d}|}{\sum_{i=1}^{m} d(E_i, \Omega) + (|\Omega| - m)\bar{d}} \qquad (۱۴.۴)$$

$$d(X, \Omega) = min_{\substack{Y \in \Omega \\ Y \neq X}} \quad \| F(X) - F(Y) \| \qquad (۱۵.۴)$$

$$\bar{d} = \frac{1}{|\Omega|} \sum_{X \in \Omega} d(X, \Omega) \qquad (۱۶.۴)$$

که $(E_1, \ldots, E_m)$، m جواب رأسی در یک مجموعه از جواب‌های بهینه پارتوی شناخته شده هستند، Ω یک مجموعه از جواب‌های غیرتسلطی است و m تعداد اهداف است.



شکل ۶.۴: استاندارد تنوع Δ.

شاخص HV، حجم (اندازه در فضای هدف) تحت پوشش با اعضایی از جواب‌های غیرتسلطی برای مسائلی که همه اهداف آنها باید مینیمم شود، را محاسبه می‌کند. به‌صورت ریاضی برای هر جواب $x_i \in \Omega$ یک مکعب v_i با یک نقطه مرجع ساخته می‌شود و جواب x_i به عنوان گوشه قطری از مکعب در نظر گرفته می‌شود. شاخص HV به‌صورت ریاضی در زیر تعریف می‌شود:

$$HV = \bigcup_{i=1}^{|\Omega|} v_i \qquad (۱۷.۴)$$

که Ω مجموعه غیر تسلطی به‌دست آمده است. لازم است تأکید کنیم جواب‌هایی که بر نقطه مرجع مسلط نمی‌شوند، در محاسبه HV کنار گذاشته می‌شوند. الگوریتم‌های با مقادیر HV بزرگ، پسندیده

هستند. از آنجایی که محاسبه HV به یک نقطه مرجع وابسته است، مقدار HV یک مجموعه از جواب‌ها، به وسیله یک مجموعه مرجع از جواب‌های بهینه پارتو با همان نقطه‌ی مرجع نرمال‌سازی می‌شود [۵۰]. مقدار HV بعد از نرمال‌سازی در بازه‌ی $[۰, ۱]$ محدود شده است.

تنظیمات پارامتر الگوریتم

طبق مکانیزم خود انطباق پیشنهاد شده در این فصل، در گام اولیه یک بازه برای HMCR در نظر گرفته می‌شود. معمولاً یک مقدار بزرگ برای HMCR در نظر گرفته می‌شود (یعنی $HMCR \geq ۰/۹$) [۳۸]. لذا، مقدار $HMCR_{min}$ و $HMCR_{max}$ به ترتیب ۰/۹ و ۱ انتخاب می‌شوند.

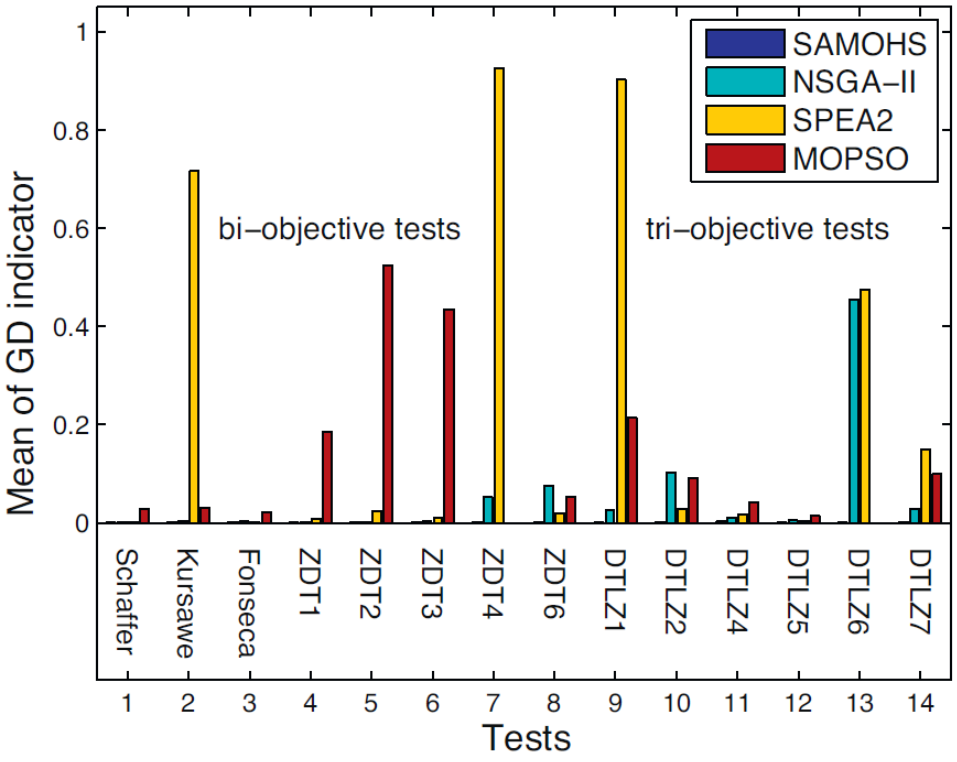
بر طبق مطالب گذشته [۶۶، ۲۳، ۲۵]، یک مقدار کم برای PAR اختصاص داده می‌شود. بنابراین، $PAR_{mean۱}$، $PAR_{std۱}$، $PAR_{mean۲}$ و $PAR_{std۲}$ به ترتیب مقادیر ۰/۲، ۰/۰۲۵، ۰/۱ و ۰/۰۲۵ انتخاب می‌شود. برای پارامتر HMS به منظور مقایسه نسبتاً خوب، مقدار ۱۰۰ در نظر گرفته می‌شود. به علاوه، همان گونه که قبلاً ذکر شد، برای پارامتر K مقادیر $K_{min۱} = ۱/۵$ ، $K_{max۱} = ۲$، $K_{min۲} = ۰/۱$ و $K_{max۲} = ۰/۵$ در نظر گرفته شده است. نسخه حقیقی‌ـ کد NSGA-II توسط دب و همکاران [۱۰] معرفی شده است. مقادیر پارامترهای زیر برای NSGA-II استفاده می‌شود. احتمال تقاطع ۰/۹، احتمال جهش $\frac{۱}{n}$ (n تعداد متغیرهای تصمیم)، شاخص توزیع و جهش برای SBX [۲۷] (شبیه‌سازی متقاطع دودویی) هر دو ۲۰ هستند و اندازه جمعیت ۱۰۰ می‌باشد. پارامترهای پیش فرض استفاده شده در SPEA۲ [۵۴] همانند NSGA-II هستند. در الگوریتم MOPSO [۲۸] اندازه جمعیت ۱۰۰ می‌باشد، اندازه مخزن ۱۰۰، احتمال جهش ۰/۵ و تقسیمات برای شبکه‌ی انطباقی ۳۰ می‌باشند. برای اطمینان از یک مقایسه‌ی عادلانه، ۴ الگوریتم، تحت شرایط یکسانی شامل اندازه جمعیت ۱۰۰، حداکثر تابع ارزیابی ۲۵،۰۰۰ برای مسائل دو هدفه و حداکثر تابع ارزیابی ۵۰،۰۰۰ برای مسائل ۳ هدفه می‌باشد.

بحث و نتایج محاسباتی

در این بخش، مقدار شاخص GD، مقدار شاخص گستردگی و مقدار شاخص HV به‌دست آمده به وسیله الگوریتم‌های SAMOHS، NSGA-II، SPEA۲ و MOPSO در ۳۰ اجرای مستقل به ترتیب در جداول ۴.۴ـ ۲.۴ گزارش شده‌اند [۹]. به علاوه، میانگین مقدار شاخص GD، مقدار شاخص گستردگی و مقدار شاخص HV به‌دست آمده به وسیله‌ی این ۴ الگوریتم در ۳۰ اجرای مستقل به ترتیب در تصاویر ۹.۴ـ ۷.۴ نشان داده شده اند. میانگین و انحراف معیار این سه شاخص برای تجزیه و تحلیل کمی محاسبه شده است. لازم است تأکید شود که نتایج محاسباتی الگوریتم‌های SPEA۲، NSGA-II و MOPSO به طور مستقیم از [۵۰] برای مقایسه عادلانه گرفته شده است. به وضوح می‌توان از جدول ۲.۴ و شکل ۷.۴ دید که SAMOHS مقادیر بهتری برای شاخص GD نسبت به ۳ الگوریتم دیگر در همه مسائل تست دارد. به این معنی که، مرزهای پارتو تقریبی به‌دست آمده با SAMOHS نسبت به مرزهای

[۲۷] Simulated Binary Crossover
[۲۸] MultiObjective Parical Swarm Optimization

پارتو تقریبی به‌دست آمده توسط NSGA-II، SPEA۲ و MOPSO به مرزهای پارتو بهینه نزدیک‌تر است. جدول ۳.۴ و شکل ۸.۴ به وضوح نشان می‌دهند که از بین ۱۴ مسئله تست بررسی شده الگوریتم SAMOHS در ۱۰ مسئله تست، نتایج بهتری برای شاخص گستردگی دارد. به‌طور خاص، در مقایسه با NSGA-II الگوریتم SAMOHS نتایج بسیار بهتری برای همه‌ی مسائل تست شده به جز ZDT۳ دارد. بنابراین، می‌توان ادعا کرد که SAMOHS نتایج بهتری از NSGA-II در رابطه با تنوع مرز پارتو تقریبی دارد. در مقایسه با SPEA۲ الگوریتم SAMOHS برای همه مسائل به جز کورساو [۱۰]، فونسه‌کا و ZDT۳ بهتر است. در مقایسه با الگوریتم MOPSO الگوریتم SAMOHS برای همه مسائل به جز مسائل ZDT۲ و ZDT۳ نتایج بسیار بهتری تولید می‌کند. دلیل اصلی که الگوریتم SAMOHS عملکرد ضعیفی روی مسئله‌ی ZDT۳ نشان می‌دهد این است که مرز پارتوی مسئله ZDT۳ ناپیوسته است. از طرف دیگر، مرز پارتوی ناپیوسته مسئله ZDT۳ متناظر با جواب‌های بهینه پارتو گسسته در فضای متغیر تصمیم می‌باشد. به علاوه، محاسبه‌ی BW به واریانس حافظه هارمونی هر متغیر مربوط است. لذا جواب‌های بهینه پارتو گسسته مسئله ZDT۳ در فضای متغیرهای تصمیم برای تنوع BW مفید نیستند. در نتیجه، عملکرد تنوع الگوریتم SAMOHS روی مسئله ZDT۳ می‌تواند تضعیف شود. از جدول ۴.۴ و شکل ۹.۴ مشاهده می‌شود که الگوریتم SAMOHS بهترین مقادیر HV را

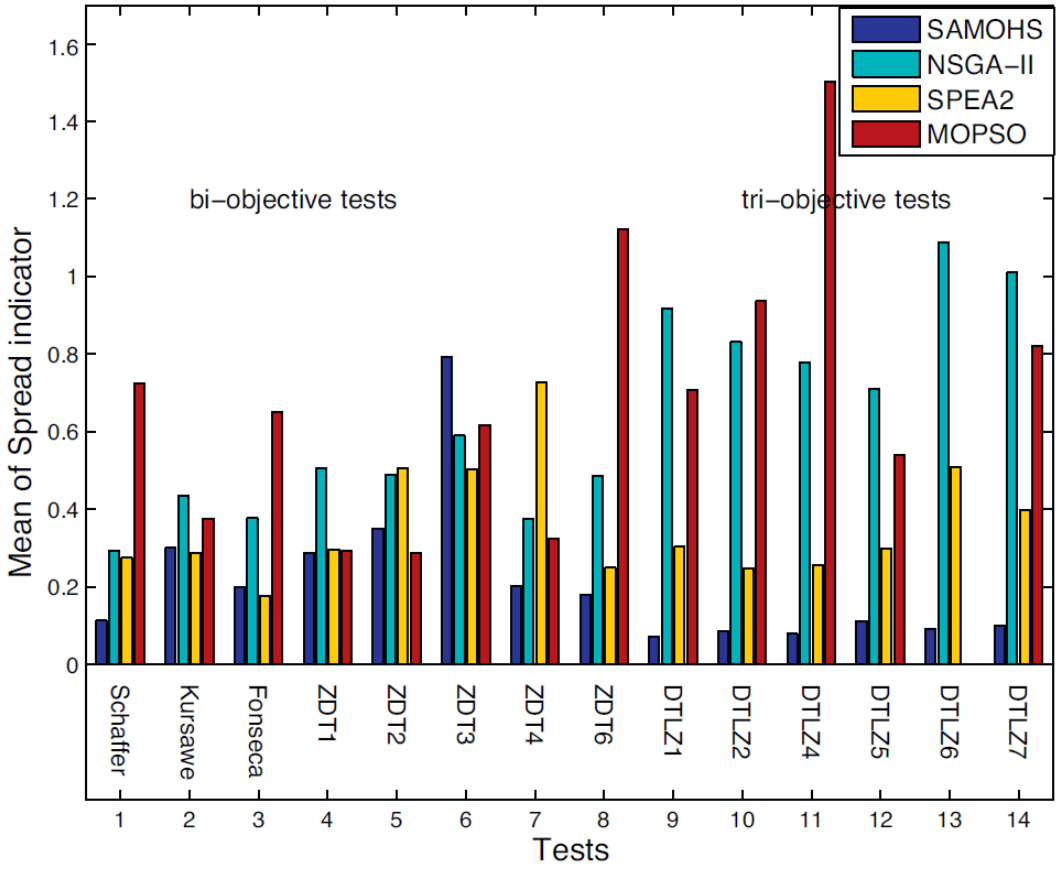


شکل ۷.۴: میانگین شاخص GD با استفاده از الگوریتم‌های SAMOHS، NSGA-II، SPEA۲ و MOPSO.

در بین چهار الگوریتم روی همه مسائل تست شده دارد. به این معنی که، الگوریتم SAMOHS بهتر از الگوریتم‌های NSGA-II، SPEA۲ و MOPSO است هنگامی که همگرایی و تنوع به‌طور هم‌زمان در نظر گرفته می‌شوند؛ زیرا مرزهای پارتو به‌دست آمده با این سه الگوریتم قادر نیستند به مرز پارتوی بهینه برسند، اما الگوریتم SAMOHS پیشنهاد شده یک مقدار نزدیک به یک تولید می‌کند. به این معنی

الگوریتم‌ها		SAMOHS		SPEA۲		NSGA-II		MOPSO	
مسائل تست	میانگین	انحراف معیار	میانگین	انحراف معیار	میانگین	انحراف معیار	میانگین	انحراف معیار	میانگین
شافر	**۳.۷۴۷۹e-۴**	**۱.۴۶۰۴e-۵**	۲.۱۵۷۲e-۳	۲.۰۹۹۹e-۴	۲.۱۲۳۲e-۳	۲.۱۱۳۰e-۴	۲.۹۲۸۵e-۲	۱.۵۵۶۶e-۲	
کورساو	**۱.۶۰۸۴e-۳**	**۲.۰۲۸۵e-۴**	۲.۸۹۷۴e-۳	۲.۳۶۳۷e-۴	۷.۱۵۷۶e-۱	۱.۲۵۲۳e-۲	۳.۰۱۴۷e-۲	۱.۷۴۴۲e-۳	
فونسه‌کا	**۱.۲۲۹۸e-۳**	**۲.۱۷۴۷e-۵**	۲.۵۶۵۶e-۳	۲.۰۰۸۲e-۴	۱.۸۵۷۳e-۳	۱.۰۷۳۱e-۴	۲.۱۴۱۶e-۲	۷.۴۱۸۳e-۳	
ZDT۱	**۲.۵۱۵۷e-۴**	**۵.۸۶۶۲e-۵**	۱.۳۴۳۷e-۳	۱.۴۰۷۸e-۴	۸.۶۱۰۴e-۳	۲.۵۹۷۳e-۳	۱.۸۵۶۴e-۱	۷.۷۴۲۹e-۲	
ZDT۲	**۲.۸۲۸۸e-۴**	**۵.۲۱۲۱e-۵**	۹.۸۱۱۲e-۴	۶.۴۱۳۸e-۴	۲.۴۷۶۶e-۲	۱.۶۰۸۳e-۲	۵.۲۴۲۸e-۱	۲.۹۶۹۹e-۱	
ZDT۳	**۵.۶۱۷۲e-۴**	**۴.۷۶۷۰e-۵**	۲.۴۷۸۳e-۳	۱.۲۷۴۶e-۴	۹.۷۱۶۵e-۳	۵.۲۳۰۵e-۳	۴.۳۴۱۸e-۱	۶.۴۸۸۰e-۲	
ZDT۴	**۸.۱۳۶۴e-۴**	**۲.۱۶۱۲e-۳**	۵.۱۶۳۵e-۲	۱.۳۲۸۱e-۳	۹.۲۵۱۲e-۱	۴.۲۸۲۱e-۱	–	–	
ZDT۶	**۱.۱۵۷۹e-۴**	**۱.۱۰۰e-۵**	۷.۵۸۱۸e-۲	۶.۰۷۹۷e-۳	۱.۹۳۰۹e-۲	۱.۳۹۹۴e-۳	۵.۲۱۳۵e-۲	۲.۴۹۶۳e-۲	
DTLZ۱	**۷.۵۳۸۱e-۴**	**۱.۵۸۸۲e-۴**	۲.۷۱۰۲e-۲	۶.۶۴۵۹e-۲	۹.۰۲۵۹e-۱	۷.۶۸۸۰e-۱	۲.۱۳۷۴e-۱	۱.۵۵۷۷e-۱	
DTLZ۲	**۱.۹۰۶۲e-۳**	**۱.۱۴۴۰e-۳**	۱.۰۱۸۹e-۱	۱.۰۵۵۹e-۱	۲.۸۱۳۶e-۲	۱.۲۵۳۲e-۲	۹.۱۲۶۹e-۱	۲.۶۸۹۶e-۲	
DTLZ۴	**۲.۶۳۴۲e-۳**	**۸.۱۹۰۴e-۴**	۱.۰۶۲۳e-۲	۷.۹۸۳۶e-۲	۱.۷۲۸۴e-۲	۹.۹۱۱۴e-۲	۴.۲۶۴۳e-۲	۱.۳۳۴۷e-۲	
DTLZ۵	**۴.۵۳۹۱e-۴**	**۳.۱۶۲۹e-۵**	۴.۷۷۶۳e-۳	۴.۰۴۳۵e-۴	۴.۲۹۸۲e-۳	۴.۴۱۴۱e-۴	۱.۴۰۹۲e-۲	۸.۷۷۴۷e-۲	
DTLZ۶	**۴.۲۷۰۵e-۴**	**۵.۲۵۶۹e-۵**	۴.۵۴۱۱e-۱	۲.۰۱۶۴e-۱	۴.۷۵۶۴e-۱	۱.۲۱۷۷e-۱	–	–	
DTLZ۷	**۱.۳۳۷۱e-۳**	**۲.۷۳۶۴e-۴**	۲.۸۱۴۷e-۲	۱.۸۱۹۴e-۲	۱.۴۹۱۹e-۱	۶.۹۲۷۸e-۲	۹.۸۷۶۸e-۲	۳.۶۱۳۹e-۳	

جدول ۲.۴: مقایسه نتایج بین الگوریتم SAMOHS و الگوریتم‌های تکاملی چند هدفه براساس شاخص همگرایی GD (مقادیر پررنگ به عنوان بهترین مقادیر هستند و خط تیره، عدم همگرا شدن به مرز پارتوی اصلی را نشان می‌دهد).



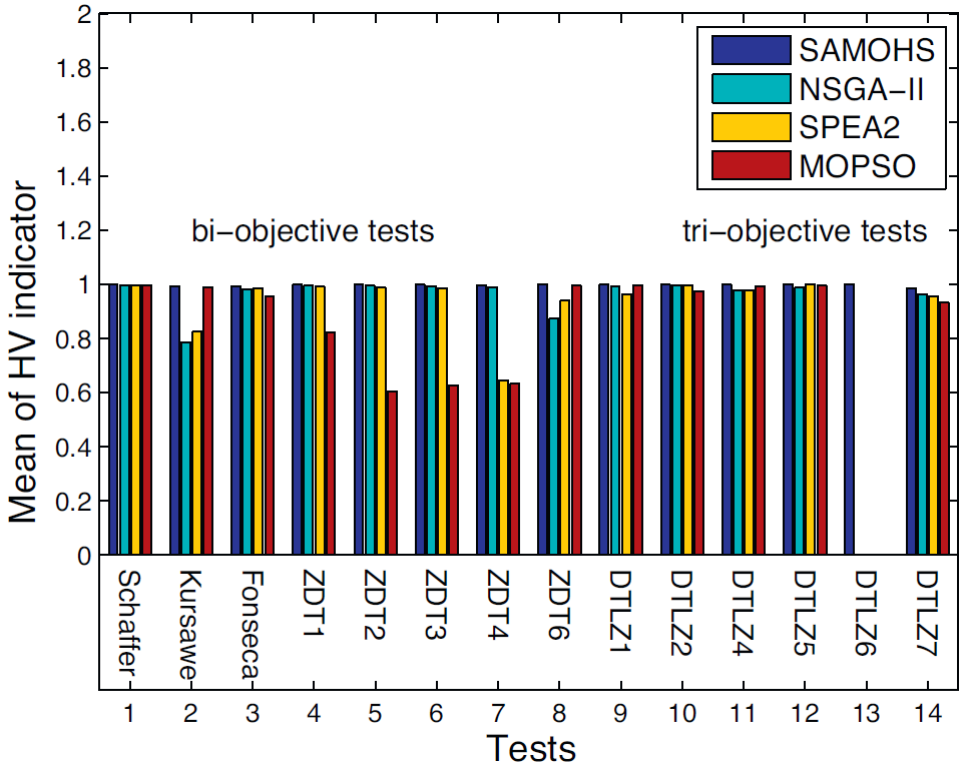
شکل ۸.۴: میانگین شاخص گستردگی با استفاده از الگوریتم‌های SAMOHS، NSGA-II، SPEA۲ و MOPSO.

که الگوریتم SAMOHS قادر به تولید یک مرز پارتو درست و خوب توزیع شده روی مسئله DTLZ۶ است.

به منظور واضح نشان دادن مرز پارتوی تقریبی به‌دست آمده به وسیله الگوریتم SAMOHS ، تصاویر

الگوریتم‌ها		SAMOHS		SPEA۲		NSGA-II		MOPSO	
مسائل تست	میانگین	انحراف معیار	میانگین	انحراف معیار	میانگین	انحراف معیار	میانگین	انحراف معیار
شافر	**۰.۱۱۲۱۷**	**۱.۶۱۲۶e-۲**	۰.۲۹۲۲۸	۲.۱۳۳۹e-۲	۰.۲۷۵۰۳	۲.۵۷۱۱e-۲	۰.۷۲۵۷۲	۱.۳۴۷۶e-۱
کورساو	۰.۲۹۹۸۲	۱.۳۶۷۹e-۲	۰.۴۳۴۰۴	۱.۹۸۱۸e-۲	**۰.۲۸۶۱۱**	**۱.۱۴۷۴e-۲**	۰.۳۷۵۹۵	۱.۶۶۶۴e-۲
فونسه‌کا	۰.۲۰۰۰۱	۱.۷۸۷۰e-۲	۰.۳۷۶۷۲	۲.۵۲۲۲e-۲	**۰.۱۷۶۶۱**	**۱.۱۱۰۰e-۱**	۰.۶۴۹۷۰	۳.۱۲۱۲e-۱
ZDT۱	**۰.۲۸۷۱۸**	**۳.۸۸۵۸e-۲**	۰.۵۰۴۲۹	۳.۹۲۵۱e-۲	۰.۲۹۶۴۴	۱.۰۸۵۰e-۱	۰.۲۹۳۸۱	۱.۶۹۵۶e-۲
ZDT۲	۰.۳۴۹۸۱	۶.۱۱۴۴e-۲	۰.۴۸۷۷۵	۲.۷۶۸۶e-۲	۰.۵۰۵۱۷	۱.۸۳۵۶e-۱	**۰.۲۸۸۰۳**	**۱.۷۵۸۰e-۲**
ZDT۳	۰.۷۹۱۷۰	۱.۷۰۳۶e-۲	۰.۵۹۰۲۵	۳.۰۴۳۹e-۲	**۰.۵۰۳۱۰**	**۹.۷۲۸۳e-۲**	۰.۶۱۷۸۰	۳.۵۰۱۹e-۲
ZDT۴	۰.۲۰۰۸۰	**۴.۲۱۶۸e-۲**	۰.۳۷۵۲۴	۲.۴۴۴۸e-۲	۰.۷۲۷۶۶	۵.۱۵۱۷e-۱	۰.۳۲۳۵۵	۳.۲۹۵۳e-۲
ZDT۶	**۰.۱۷۸۷۸**	**۲.۱۳۷۴e-۲**	۰.۴۸۶۱۱	۳.۶۰۵۴e-۲	۰.۲۴۸۶۱	۴.۹۶۶۷e-۲	۱.۱۲۳۲۶	۱.۷۳۱۱e-۱
DTLZ۱	۰.۰۷۱۰۹	**۹.۹۵۶۱e-۳**	۰.۹۱۸۶۷	۶.۶۱۰۸e-۲	۰.۳۰۲۳۶	۲.۴۹۱۹e-۱	۰.۷۰۸۰۹	۷.۸۸۵۴e-۲
DTLZ۲	۰.۰۸۶۱۸	**۱.۱۹۸۳e-۲**	۰.۸۳۰۹۲	۶.۸۴۲۷e-۲	۰.۲۴۶۷۳	۳.۵۱۸۱e-۲	۰.۹۳۶۲۸	۳.۱۶۸۴e-۱
DTLZ۴	۰.۰۷۸۲۴	**۱.۵۳۴۸e-۲**	۰.۷۷۸۶۴	۷.۸۰۹۹e-۲	۰.۲۵۵۶۸	۷.۵۶۱۶e-۲	۱.۵۰۲۱۲	۳.۳۰۹۶e-۱
DTLZ۵	۰.۱۱۱۶۸	**۸.۹۱۲۵e-۳**	۰.۷۱۰۸۰	۷.۸۰۰۹e-۲	۰.۲۹۸۴۰	۱.۸۳۷۰e-۲	۰.۵۴۰۸۷	۵.۵۸۰۲e-۱
DTLZ۶	۰.۰۹۰۸۴	**۱.۳۲۹۱e-۲**	۱.۰۸۹۴۰	۱.۸۲۵۹e-۱	۰.۵۰۸۱۵	۹.۲۲۱۴e-۲	–	–
DTLZ۷	۰.۰۹۹۷۱	**۱.۹۳۱۲e-۲**	۱.۰۰۹۶۰	۵.۰۵۷۲e-۲	۰.۳۹۷۹۰	۷.۰۳۰۴e-۲	۰.۸۲۱۵۵	۸.۹۷۴۱e-۲

جدول ۳.۴: مقایسه نتایج بین الگوریتم SAMOHS و الگوریتم‌های تکاملی چند هدفه براساس شاخص گستردگی (مقادیر پر رنگ به عنوان بهترین مقادیر هستند و خط تیره عدم همگرا شدن به مرز پارتوی اصلی را نشان می‌دهد).



شکل ۹.۴: میانگین شاخص HV با استفاده از الگوریتم‌های SAMOHS، NSGA-II، SPEA۲ و MOPSO.

۱۲.۴ ـ ۱۰.۴ مرز پارتوی تقریبی از همه مسائل تست با الگوریتم SAMOHS را توضیح می‌دهند. در شکل ۱۰.۴ مرز پارتوی اصلی و تقریبی به‌دست آمده با الگوریتم SAMOHS هر دو در همان شکل نشان داده شده است. مثلاً برای تصاویر ۱۱.۴ و ۱۲.۴ عکس چپ مرز پارتوی اصلی و عکس راست، مرز پارتوی تقریبی که با الگوریتم SAMOHS به‌دست آمده است و توزیع یکنواخت مرز پارتوی اصلی برای همه مسائل تست می‌باشد، را نشان می‌دهد. بنابراین شناخت اینکه الگوریتم SAMOHS قادر به همگرایی

الگوریتم‌ها		SAMOHS		SPEA۲		NSGA-II		MOPSO	
مسائل تست	میانگین	انحراف معیار	میانگین	انحراف معیار	میانگین	انحراف معیار	میانگین	انحراف معیار
شافر	۰.۹۹۹۷۱۵	**۵.۱۳۱۶e-۵**	۰.۹۹۶۸۴۵	۱.۰۳۷۴e-۴	۰.۹۹۶۷۷	۱.۱۲۵۵e-۴	۰.۹۹۵۶۹۱	۲.۰۹۹۵e-۳
کورساو	۰.۹۹۱۵۱۰	**۰.۹۹۱۵۱۰**	۰.۷۸۶۳۷۹	۳.۶۲۵۱e-۴	۰.۸۲۴۱۶۳	۲.۱۹۷۵e-۴	۰.۹۸۸۶۳۱	۱.۰۹۵۳e-۳
فونسه‌کا	۰.۹۹۲۴۱۲	**۱.۶۴۹۰e-۳**	۰.۹۷۹۶۰۴	۱.۰۲۸۶e-۳	۰.۹۸۵۳۵۵	۴.۷۱۸۳e-۴	۰.۹۵۵۲۰۵	۲.۰۷۳۰e-۲
ZDT۱	۰.۹۹۹۰۰۴	**۲.۹۳۵۴e-۴**	۰.۹۹۴۷۰۴	۱.۸۹۱۰e-۴	۰.۹۹۱۱۶۶	۶.۳۳۱۸e-۴	۰.۸۲۰۷۹۴	۷.۰۴۷۳e-۲
ZDT۲	۰.۹۹۷۶۵۹	**۶.۵۳۴۷e-۴**	۰.۹۹۵۵۴۳	۲.۱۹۸۳e-۴	۰.۹۸۶۸۲۸	۲.۷۰۷۱e-۳	۰.۶۰۲۵۵۱	۱.۶۱۶۷e-۱
ZDT۳	۰.۹۹۷۶۵۳	**۵.۷۵۱۳e-۴**	۰.۹۹۲۵۴۸	۹.۸۱۲۶e-۳	۰.۹۸۳۳۹۷	۷.۱۰۶۷e-۳	۰.۶۲۷۳۹۸	۴.۴۹۲۶e-۲
ZDT۴	۰.۹۹۵۸۱۷	**۳.۸۰۵۱e-۳**	۰.۹۸۹۶۸۲	۱.۷۹۷۴e-۳	۰.۶۴۵۹۵۱	۱.۳۵۴۲e-۱	۰.۶۳۳۱۳۲	۲.۱۱۱۸e-۱
ZDT۶	۰.۹۹۹۱۹۹	**۲.۵۹۷۲e-۵**	۰.۸۷۲۰۷۶	۱.۰۱۹۸e-۲	۰.۹۳۸۸۹۷	۴.۳۷۷۶e-۳	۰.۹۹۴۵۴۳	۲.۶۶۹۳e-۲
DTLZ۱	۰.۹۹۸۳۷۴	**۱.۲۵۰۷e-۳**	۰.۹۹۲۲۸۲	۳.۲۶۰۳e-۴	۰.۹۶۳۸۰	۴.۳۵۷۷e-۲	۰.۹۹۵۱۸۰	۴.۷۳۴۴e-۳
DTLZ۲	۰.۹۹۸۲۴۶	**۳.۵۴۳۹e-۴**	۰.۹۹۵۳۲۷	۱.۰۱۹۹e-۳	۰.۹۹۶۱۹	۳.۳۰۱۰e-۴	۰.۹۷۲۶۹۴	۴.۶۱۸۱e-۲
DTLZ۴	۰.۹۹۸۰۰۸	**۲.۱۲۷۳e-۴**	۰.۹۷۵۵۲۵	۳.۱۱۴۵e-۲	۰.۹۷۶۵۲۲	۳.۱۷۷۰e-۲	۰.۹۹۳۰۲۸	۱.۳۷۱۳e-۳
DTLZ۵	۰.۹۹۹۸۷۸	**۲.۷۷۰۹e-۵**	۰.۹۸۷۰۴۰	۱.۰۳۳۳e-۳	۰.۹۹۹۴۶	۱.۳۶۹۷e-۴	۰.۹۹۴۴۰۱	۹.۰۸۲۱e-۳
DTLZ۶	۰.۹۹۹۹۷۱	**۱.۶۵۲۳e-۵**	۰.۰۰e+۰۰	۰.۰۰e+۰۰	۰.۰۰e+۰۰	۰.۰۰e+۰۰	۰.۰۰e+۰۰	۰.۰۰e+۰۰
DTLZ۷	۰.۹۸۵۹۱۴	**۵.۶۹۲۵e-۴**	۰.۹۶۳۳۴۶	۴.۰۲۵۶e-۳	۰.۹۵۶۳۵	۴.۴۵۲۹e-۳	۰.۹۳۱۶۲۳	۵.۷۵۰۲e-۳

جدول ۴.۴: مقایسه نتایج بین SAMOHS و الگوریتم‌های تکاملی چند هدفه براساس شاخص ابر حجم (HV) (مقادیر پررنگ به عنوان بهترین مقادیر هستند).

مرز پارتوی اصلی برای همه مسائل تست است، آسان می‌باشد.

۲.۳.۴ مقایسه بین الگوریتم SAMOHS و الگوریتم‌های جست‌وجوی هارمونی چند هدفه

برای نشان دادن اثر استراتژی خود انطباق پیشنهادی، الگوریتم SAMOHS با MOHS۱ و MOHS۲ ایجاد شده توسط ریکارت و همکاران[۲۹] [۴۳] که در فصل قبل توضیح داده شدند، مقایسه می‌شود. الگوریتم‌های MOHS۱ و MOHS۲ هر دو الگوریتم جست‌وجوی هارمونی را برای حل مسائل بهینه‌سازی چندهدفه به کار می‌گیرند.

مسائل تست

از آنجایی‌که MOHS۱ و MOHS۲ تنها روی توابع دو هدفه در [۴۳] ارزیابی می‌شوند و این دو الگوریتم به مجموعه ای از پارامترها بر طبق مسئله‌ی خاص نیاز دارند، مسائل تست پیش فرض در [۴۳] و سه مسئله‌ی دو هدفه که قبلاً ذکر شده‌اند، انتخاب می‌شوند. بنابراین مسائل تست شامل ZDT۱-ZDT۶ و مسائل (۸.۴)، (۹.۴) و (۱۰.۴) می‌باشند. چون ZDT۵ یک مسئله با متغیرهای دوتایی است و الگوریتم SAMOHS متغیرهای اعشاری استفاده می‌کند، مسئله ZDT۵ بررسی نمی‌شود.

تنظیمات محاسباتی

پارامترهای کنترلی برای SAMOHS پیشنهادی، پیش از این معرفی شدند. برای اطمینان از مقایسه عادلانه، MOHS۱ و MOHS۲ از پارامترهای کنترلی پیش فرض در فصل قبل استفاده می‌کنند.

[۲۹] Ricart et al.

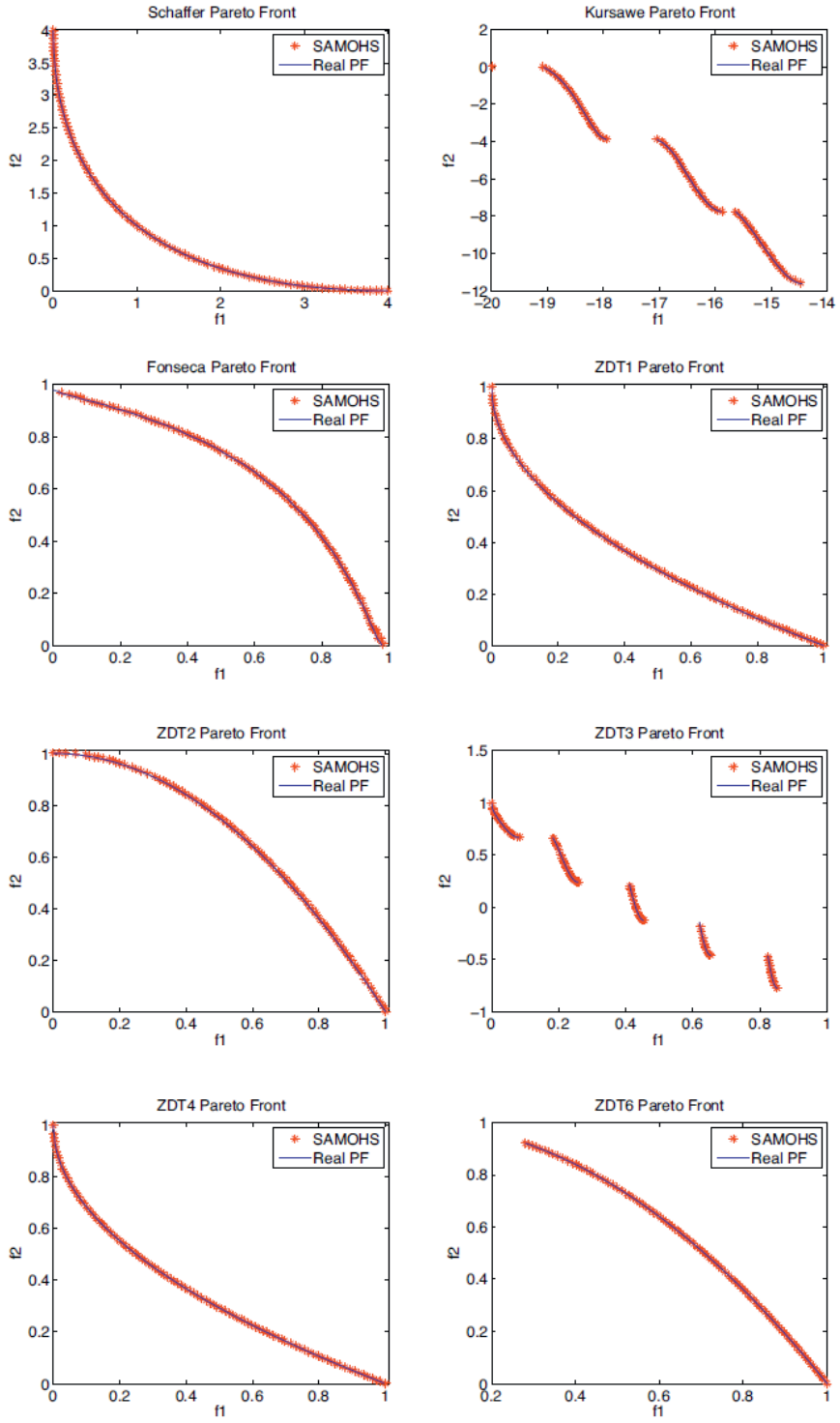
پارامترهای الگوریتم‌های SAMOHS، MOHS۱ و MOHS۲ در جدول ۵.۴ ارائه شده اند که Δ_x دامنه‌ی مقادیر فضای جست‌وجو است. لازم نیست BW قرار داده شود؛ زیرا در الگوریتم SAMOHS به‌طور انطباقی به روز می‌شود. به علاوه، $PAR_{mean۱}$، $PAR_{std۱}$، $PAR_{mean۲}$ و $PAR_{std۲}$ به ترتیب، ۰.۲، ۰.۰۲۵، ۰.۱ و ۰.۰۲۵ قرار داده می‌شوند. جدول ۵.۴ به‌طور واضح نشان می‌دهد که HMS و حداکثر تابع ارزیابی برای سه الگوریتم به ترتیب ۱۰۰ و ۲۵،۰۰۰ قرار داده می‌شوند. بنابراین مقایسه بین الگوریتم SAMOHS و دو الگوریتم دیگر، عادلانه است.

	$HMCR$	PAR	BW	HMS	ارزیابی تابع ماکزیمم
$SAMOHS$	$HMCR_{min}$ =۰.۹ $HMCR_{max}$ =۱.۰	(۵.۴) or (۱.۴)	معادلات (۱.۴)، (۷.۴) یا(۶.۴)،(۷.۴)	۱۰۰	۲۵،۰۰۰
$MOHS۱$	۰.۹۵	PAR =۰.۱	۰.۰۱×Δ_x	۱۰۰	۲۵،۰۰۰
$MOHS۲$	۰.۹۵	PAR =۰.۱	۰.۰۱×Δ_x	۱۰۰	۲۵،۰۰۰

جدول ۵.۴: پارامترهای محاسباتی برای الگوریتم‌های SAMOHS، MOHS۱ و MOHS۲.

نتایج تجربی و تحلیل‌ها

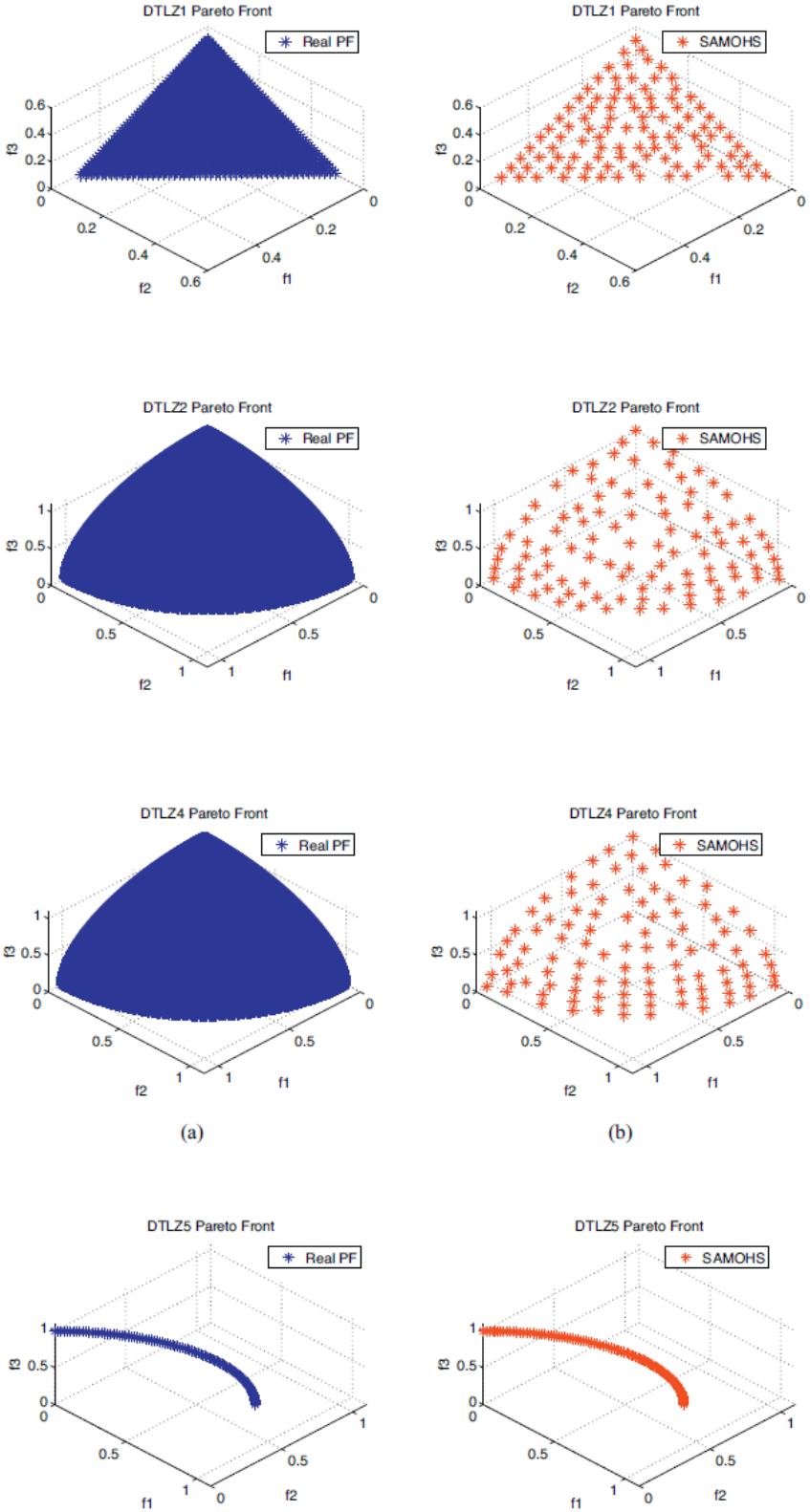
سه شاخص که قبلاً معرفی شده بودند را با استفاده از الگوریتم‌های MOHS۱، SAMOHS و MOHS۲ در سی اجرای مستقل در جداول ۸.۴ـ ۶.۴ آورده‌ایم. به علاوه، میانگین مقادیر شاخص‌های به‌دست آمده، با این سه الگوریتم در ۳۰ اجرای مستقل در تصاویر ۱۵.۴ـ۱۳.۴ آورده شده است. از جدول ۶.۴ و شکل ۱۵.۴، مشاهده می‌شود که الگوریتم SAMOHS بهترین اجرا را در بین سه الگوریتم از نظر همگرایی دارد. به این معنی که، مرز پارتوی تقریبی به‌دست آمده با الگوریتم SAMOHS نسبت به MOHS۱ و MOHS۲ نزدیک‌تر به مرز پارتوی واقعی است. طبق نتایج به‌دست آمده از جدول ۷.۴ و شکل ۱۴.۴، برای شاخص گستردگی از بین ۸ مسئله تست شده الگوریتم SAMOHS در ۶ مسئله نتایج بهتری دارد. در مقایسه با MOHS۱، الگوریتم SAMOHS نتایج بهتری برای همه مسائل تست به جز ZDT۳ به‌دست می‌آورد. برای ZDT۳ نتایج به‌دست آمده با SAMOHS و MOHS۱ تقریباً مشابه‌اند و الگوریتم SAMOHS انحراف معیار استاندارد کمتری نسبت به MOHS۱ دارد. به این معنی که، الگوریتم SAMOHS پایداری بهتری از MOHS۱ دارد. همچنین در مقایسه با MOHS۲، الگوریتم SAMOHS نتایج بهتری برای همه مسائل تست به جز ZDT۳ و ZDT۶ دارد. نتایج به‌دست آمده با SAMOHS و MOHS۲ روی ZDT۳ و ZDT۶ مشابه‌اند. بنابراین می‌توان ادعا کرد که در مورد تنوع (گوناگونی) مرز پارتوی تقریبی، الگوریتم SAMOHS نسبت به مسائل MOHS۱ و MOHS۲ بهتر عمل می‌کند. جدول ۸.۴ و شکل ۱۵.۴ به‌طور بدیهی نشان می‌دهند که الگوریتم SAMOHS پیشنهاد شده بهترین نتایج شاخص HV را برای همه مسائل تست به جز مسئله ZDT۶ تولید می‌کند. برای مسئله ZDT۶، با کمک الگوریتم MOHS۲ بهترین نتایج به‌دست می‌آید و نتایج SAMOHS خیلی نزدیک به نتایج MOHS۲ است. از این رو، می‌توان ادعا کرد که الگوریتم SAMOHS وقتی که با الگوریتم‌های MOHS۱ و MOHS۲ از نظر همگرایی و تنوع مقایسه می‌شود، بهتر عمل می‌کند. به منظور نشان دادن مرز پارتوی به‌دست آمده، نتایج تجربی این سه الگوریتم برای مسائل شافر و ZDT۴ در تصاویر ۱۶.۴ و ۱۷.۴ به ترتیب به تصویر کشیده شده است. از شکل ۱۶.۴ مشاهده می‌شود



شکل ۱۰.۴: طرح‌هایی از جواب غیر تسلطی در فضای هدف مسائل شافر، فونسه‌کا، کورساو، ZDT۱، ZDT۲، ZDT۳، ZDT۴، ZDT۶ .

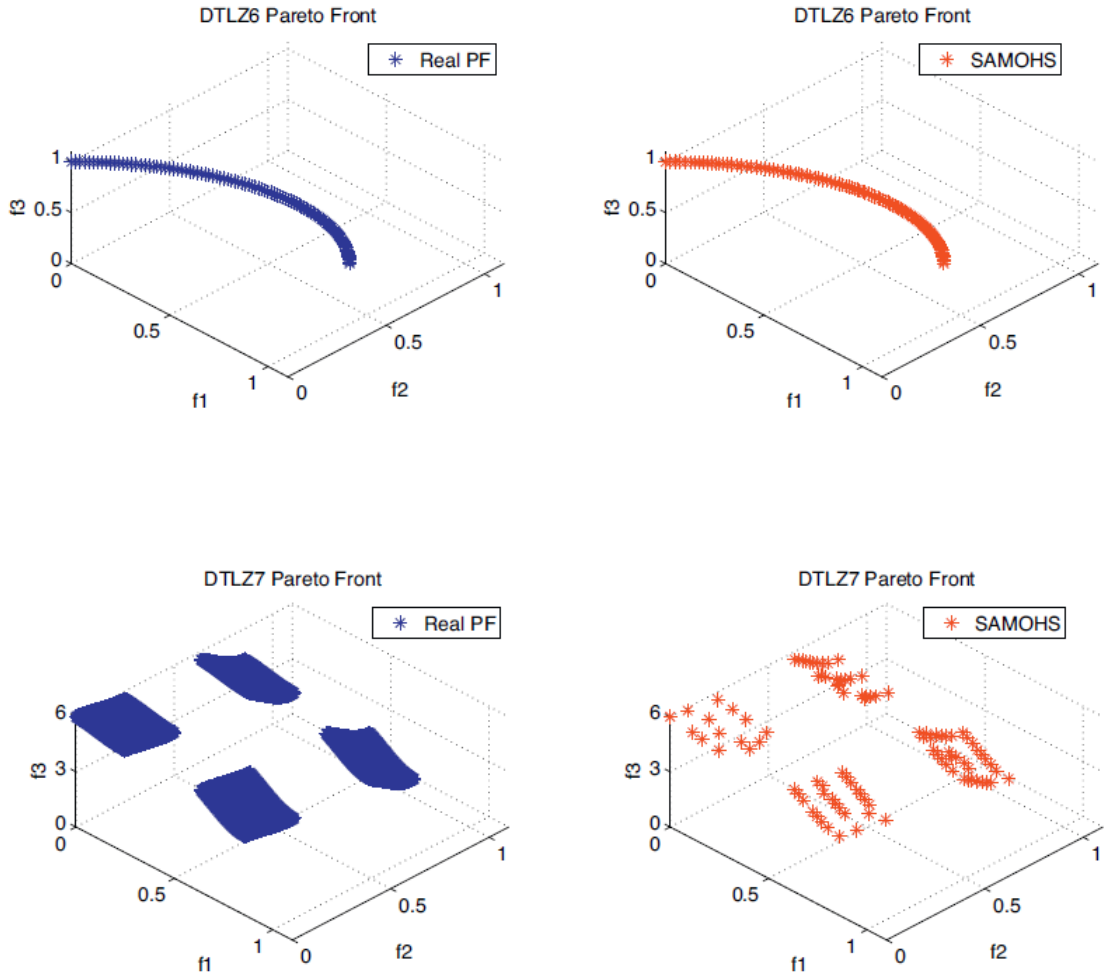
که MOHS۱ و MOHS۲ قادر به همگرایی به مرز پارتوی دقیق برای مسئله شافر هستند در حالی که آنها (به خصوص MOHS۱) قادر به تولید یک مرز پارتوی با توزیع خوب نیستند.

در حالی که الگوریتم SAMOHS یک مرز پارتوی دقیق و با توزیع خوب به‌دست می‌آورد. اجرای بهتر الگوریتم SAMOHS را می‌توان به این واقعیت نسبت داد که فاصله پهنای باند خود انطباق پیشنهاد



شکل ۱۱.۴: طرح‌هایی از جواب‌های غیر تسلطی در فضای هدف مسائل DTLZ۱، DTLZ۲، DTLZ۴ و DTLZ۵.

شده در الگوریتم SAMOHS قادر به افزایش توانایی‌های جست‌وجوی سراسری و محلی است. از شکل ۱۷.۴ مشاهده می‌شود که الگوریتم‌های MOHS۱ و MOHS۲ برای مسئله ZDT۴ قادر به همگرایی

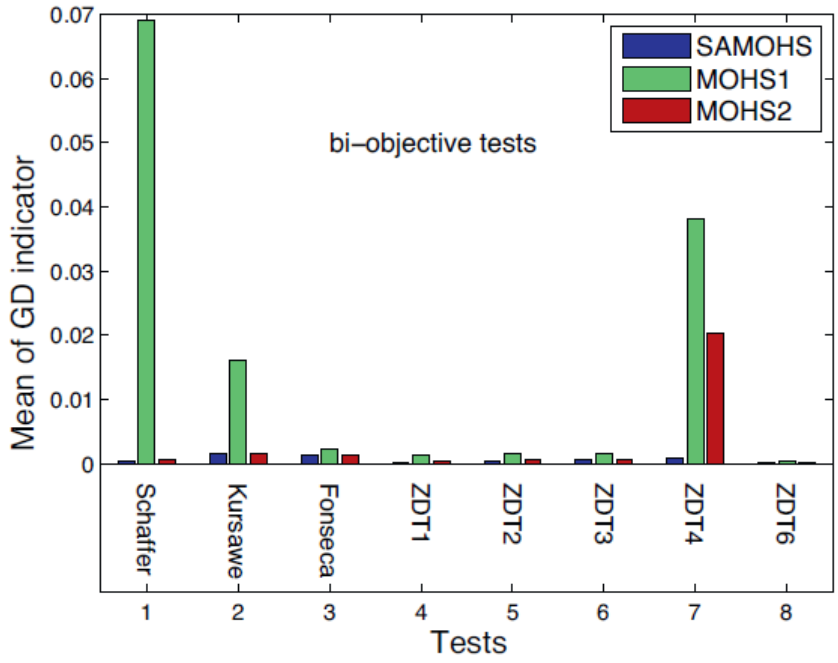


شکل ۱۲.۴: طرح‌هایی از جواب‌های غیرتسلطی در فضای هدف DTLZ۶، DTLZ۷.

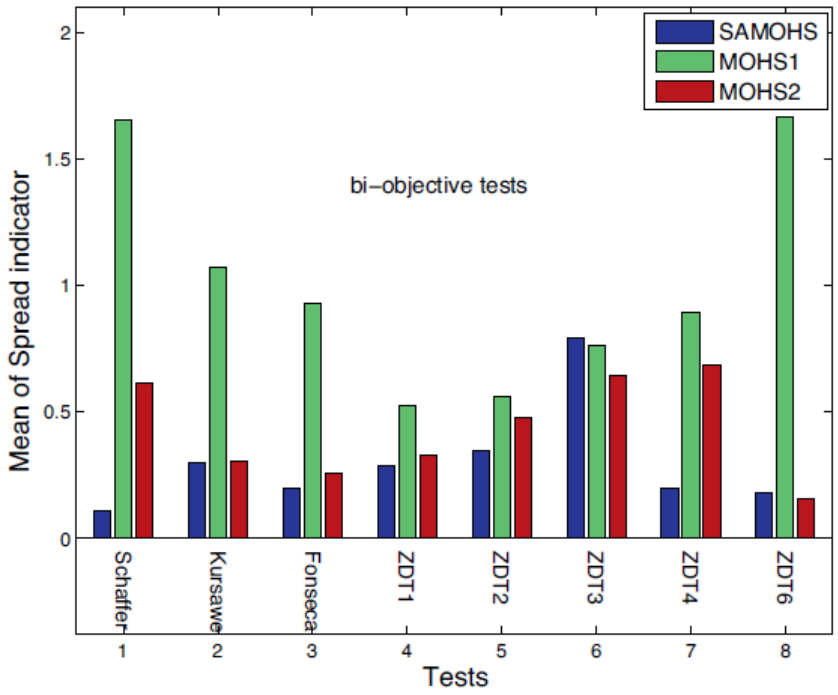
MOHS۲		MOHS۱		SAMOHS		الگوریتم‌ها
انحراف معیار	میانگین	انحراف معیار	میانگین	انحراف معیار	میانگین	مسائل تست
۲.۰۶۲۵e-۴	۵.۵۵۸۴e-۴	۱.۰۸۶۴e-۱	۶.۹۱۲۲e-۲	**۱.۴۶۰۴e-۵**	**۳.۷۴۷۹e-۴**	شافر
۳.۴۶۹۷e-۴	۱.۶۳۲۴e-۳	۱.۱۱۴۳e-۲	۱.۶۱۹۶e-۲	**۲.۰۲۸۵e-۴**	**۱.۶۰۸۴e-۳**	کورساو
۴.۵۱۴۰e-۵	۱.۲۹۳۰e-۳	۵.۲۱۱۱e-۴	۲.۳۶۲۷e-۳	**۲.۱۷۴۷e-۵**	**۱.۲۲۹۸e-۳**	فونسه‌کا
۱.۸۴۲۴e-۴	۴.۹۳۶۰e-۴	۳.۲۴۲۸e-۴	۱.۳۰۲۵e-۳	**۵.۸۶۶۲e-۵**	**۲.۵۱۵۷e-۴**	ZDT۱
۲.۵۳۴۹e-۴	۵.۴۲۰۹e-۴	۶.۷۳۶۳e-۴	۱.۵۲۳۷e-۳	**۵.۲۱۲۱e-۵**	**۲.۸۲۸۸e-۴**	ZDT۲
۹.۶۳۸۹e-۵	۷.۱۶۳۱e-۴	۳.۶۶۵۰e-۴	۱.۵۴۴۱e-۳	**۴.۷۶۷۰e-۵**	**۵.۶۱۷۲e-۴**	ZDT۳
۱.۳۵۳۲e-۲	۲.۰۱۸۶e-۲	۲.۶۰۸۲e-۲	۳.۸۲۴۸e-۲	**۲.۱۶۱۲e-۳**	**۸.۱۳۶۴e-۴**	ZDT۴
۱.۰۴۳۳e-۵	۱.۱۸۵۸e-۴	۱.۳۳۸۸e-۳	۳.۴۶۰۱e-۴	**۱.۱۱۰۰e-۵**	**۱.۱۵۷۹e-۴**	ZDT۶

جدول ۶.۴: نتایج مقایسه بین الگوریتم SAMOHS و الگوریتم‌های جست‌وجوی هارمونی چند هدفه براساس شاخص GD (بهترین مقادیر پر رنگ هستند).

به مرز دقیق پارتو نیستند در حالی که الگوریتم SAMOHS پیشنهاد شده قادر به تولید یک مرز پارتو دقیق و با توزیع خوب می‌شود. اجرای بهتر الگوریتم SAMOHS را می‌توان به این واقعیت نسبت داد



شکل ۱۳.۴: میانگین شاخص GD با استفاده از الگوریتم‌های SAMOHS، MOHS۱ و MOHS۲.



شکل ۱۴.۴: میانگین شاخص گستردگی با استفاده از الگوریتم‌های SAMOHS، MOHS۱ و MOHS۲.

که تنظیم پارامتر خود انطباق براساس واریانس حافظه هارمونی برای پارامترهای HMCR و PAR می‌تواند تعادل بهتری بین تنوع و تشدید برقرار کند.

طبق نتایج تجربی فوق و تحلیل‌ها، می‌توان نتیجه گرفت که الگوریتم SAMOHS پیشنهاد شده نتایج بهتری نسبت به الگوریتم‌های MOHS۱ و MOHS۲ از نظر همگرایی و تنوع دارد. به علاوه، عملکرد الگوریتم SAMOHS با استفاده از مکانیزم خود انطباقی پیشنهاد شده نتایج بهتری می‌دهد. این نتایج

الگوریتم‌ها		SAMOHS		MOHS۱		MOHS۲
مسائل تست	میانگین	انحراف معیار	میانگین	انحراف معیار	میانگین	انحراف معیار
شافر	**۰.۱۱۲۱۷**	**۱.۶۱۲۶e-۲**	۱.۶۵۷۷۶	۱.۱۹۴۳e-۱	۰.۶۱۶۷۴	۷.۲۵۶۸e-۲
کورساو	**۰.۲۹۹۸۲**	**۱.۳۶۷۹e-۲**	۱.۰۷۲۱۶	۱.۱۵۷۶e-۱	۰.۳۰۷۲۸	۱.۱۸۵۵e-۲
فونسه‌کا	**۰.۲۰۰۰۱**	**۱.۷۸۷۰e-۲**	۰.۹۳۱۵۵	۹.۱۱۲۱e-۲	۰.۲۶۰۳۸	۳.۶۴۸۷e-۲
ZDT۱	**۰.۲۸۷۱۸**	**۳.۸۸۵۸e-۲**	۰.۵۲۴۴۹	۳.۷۱۳۱e-۲	۰.۳۲۷۵۴	۵.۷۰۱۷e-۲
ZDT۲	**۰.۳۴۹۸۱**	**۶.۱۱۴۴e-۲**	۰.۵۶۱۱۳	۴.۰۰۶۴e-۲	۰.۴۸۰۶۱	۶.۲۱۴۹e-۲
ZDT۳	۰.۷۹۱۷۰	۱.۷۰۳۶e-۲	۰.۷۶۱۹۲	۱.۹۰۷۷e-۲	**۰.۶۴۴۰۹**	**۱.۵۹۴۷e-۲**
ZDT۴	**۰.۲۰۰۸۰**	**۴.۲۱۶۸e-۲**	۰.۸۹۳۶۲	۱.۰۶۶۷e-۱	۰.۶۸۸۹۳	۲.۳۴۵۲e-۱
ZDT۶	۰.۱۷۸۷۸	۲.۱۳۷۴e-۲	۱.۶۶۴۸۷	۷.۰۴۷۳e-۲	**۰.۱۵۵۶۶**	**۱.۴۴۷۹e-۲**

جدول ۷.۴: نتایج مقایسه بین الگوریتم SAMOHS و الگوریتم‌های جست‌وجوی هارمونی چند هدفه براساس شاخص Δ (بهترین مقادیر پر رنگ هستند).

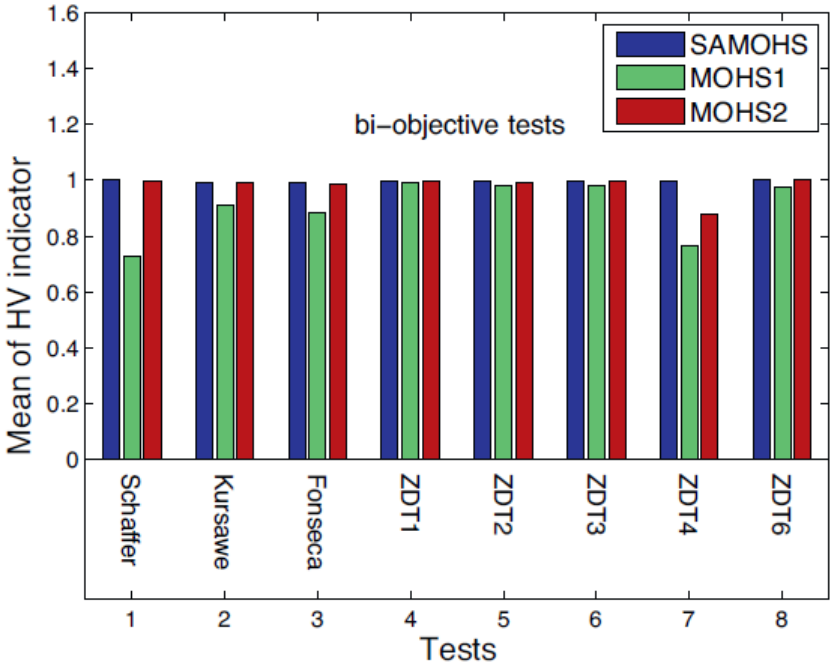
الگوریتم‌ها		SAMOHS		MOHS۱		MOHS۲
مسائل تست	میانگین	انحراف معیار	میانگین	انحراف معیار	میانگین	انحراف معیار
شافر	**۰.۹۹۹۷۱۵**	**۵.۱۳۱۶e-۵**	۰.۷۲۴۶۱۲	۲.۰۰۷۲e-۱	۰.۹۹۷۰۷۴	۹.۲۸۶۱e-۴
کورساو	**۰.۹۹۱۵۱۰**	**۶.۰۱۸۹e-۴**	۰.۹۱۰۹۱۶	۳.۷۷۸۵e-۲	۰.۹۹۱۴۹۶	۴.۲۳۱۹e-۴
فونسه‌کا	**۰.۹۹۲۴۱۲**	**۱.۶۴۹۰e-۳**	۰.۸۸۱۲۰۸	۴.۲۸۳۳e-۲	۰.۹۸۶۴۹۴	۳.۶۷۳۶e-۳
ZDT۱	**۰.۹۹۹۰۰۴**	**۲.۹۳۵۴e-۴**	۰.۹۸۹۴۸۷	۲.۸۵۰۴e-۳	۰.۹۹۶۵۵۱	۱.۷۱۹۲e-۳
ZDT۲	**۰.۹۹۷۶۵۹**	**۶.۵۳۴۷e-۴**	۰.۹۸۲۱۳۵	۶.۶۷۷۲e-۳	۰.۹۹۲۵۹۶	۳.۸۵۴۶e-۳
ZDT۳	**۰.۹۹۷۶۵۳**	**۵.۷۵۱۳e-۴**	۰.۹۸۱۲۹۹	۴.۱۹۲۸e-۳	۰.۹۹۵۵۹۷	۱.۵۶۳۸e-۳
ZDT۴	**۰.۹۹۵۸۱۷**	**۳.۸۰۵۱e-۳**	۰.۷۶۲۱۹۵	۱.۵۷۲۷e-۱	۰.۸۷۹۰۳۱	۶.۲۴۶۹e-۲
ZDT۶	۰.۹۹۹۱۹۹	۲.۵۹۷۲e-۵	۰.۹۷۷۷۶۷	۱.۰۹۳۸e-۲	**۰.۹۹۹۲۲۷**	**۱.۰۶۷۴e-۵**

جدول ۸.۴: نتایج مقایسه‌ی بین الگوریتم SAMOHS و الگوریتم‌های چند هدفه جست‌وجوی هارمونی براساس شاخص HV (بهترین مقادیر پر رنگ هستند).

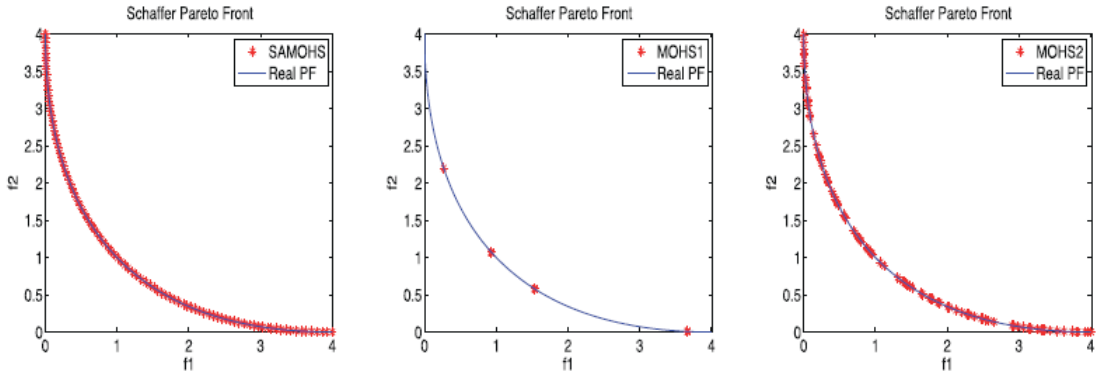
تجربی، اثر و مکانیزم خود انطباق پیشنهاد شده را نشان می‌دهد.

۴.۴ طراحی لوله‌های گرمایی ماهواره

از آنجایی که لوله‌های گرمایی معمولاً برای سیستم خنک کننده ماهواره‌ای کاربرد دارند، طراحی لوله‌های گرمایی بسیار مفید است و مطالعات زیادی [۴۹، ۲۹، ۱۳] روی این مسئله متمرکز شده‌اند. گیم و هانگبو [۶۷] الگوریتم هارمونی را برای حل این مسئله استفاده کرده‌اند اما آنها این مسئله چند هدفه را به یک مسئله بهینه‌سازی تک هدفه تبدیل کردند؛ به این معنی که تعداد اجرای بسیاری برای به‌دست آوردن مجموعه بهینه پارتو نیاز است. به همین دلیل، مسئله طراحی لوله‌های گرمایی ماهواره با استفاده از الگوریتم‌های SAMOHS، MOHS۱ و MOHS۲ در ادامه حل شده است.



شکل ۱۵.۴: میانگین شاخص HV با استفاده از الگوریتم‌های SAMOHS، MOHS۱ و MOHS۲.

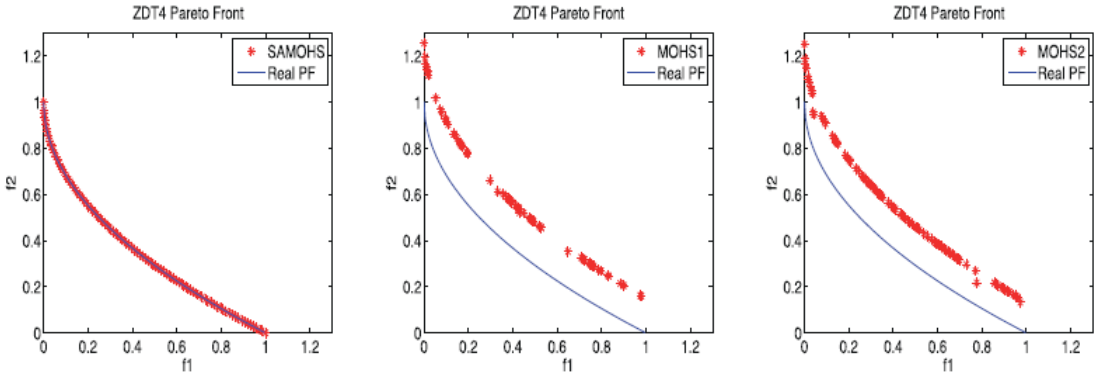


(a) SAMOHS (b) MOHS1 (c) MOHS2

شکل ۱۶.۴: مرزهای پارتوی به‌دست آمده برای مسئله شافر با الگوریتم‌های SAMOHS، MOHS۱ و MOHS۲.

پارامتر طراحی	حد پایین	حد بالا
طول برجستگی بال (L_f)	۱۰.۰ mm	۲۵.۴ mm
طول برش از ناحیه متصل چسبیده (L_c)	۱.۵ mm	۲.۵ mm
ضخامت بال (t_f)	۱.۰ mm	۱.۷ mm
ضخامت چسبندگی (t_b)	۰.۱۲ mm	۰.۲۲ mm
دمای عملیات (T_op)	$-۲۰.۰°C$	$۶۰.۰°C$

جدول ۹.۴: محدوده طراحی پارامتر.



(a) SAMOHS (b) MOHS1 (c) MOHS2

شکل ۱۷.۴: مرزهای پارتوی به‌دست آمده برای مسئله ZDT۴ با الگوریتم‌های SAMOHS، MOHS۱ و MOHS۲.

فرمول‌بندی مسئله

مدل لوله‌های گرمایی در مرجع [۶۷] معرفی شده است. هدف اصلی این مسئله، ماکزیمم کردن هدایت حرارتی و مینیمم ساختن جرم کل است. این مسئله پنج پارامتر طراحی شده دارد و جزئیات محدوده‌ی متغیرها در جدول ۹.۴ ارائه شده است. در واقع این مسئله، یک مسئله بهینه‌سازی دو هدفه است که در آن تابع هدف مربوط به ساختن هدایت حرارتی، ماکزیمم‌سازی است و می‌توان آن را به‌صورت زیر، مدل‌سازی ریاضی کرد.

$$
\begin{aligned}
G &= f(L_f, L_c, t_f, t_b, T_{op}) \\
&= ۰٫۳۷۴۵۳۷۸ - ۰٫۹۳۵۲۹۰۹ * t_b + ۱٫۰۱۶۱۲ * {t_b}^۲ + ۰٫۰۲۳۲۴۱۲۸ * L_c \\
&- ۰٫۰۰۷۲۰۹۹۳ * {L_c}^۲ + ۰٫۰۰۱۸۳۸۳۷۹ * L_f - ۰٫۰۰۰۰۵۳۷۹۷۰۷ * {L_f}^۲ \\
&+ ۰٫۰۲۴۴۷۳۹۱ * t_f + ۰٫۰۰۲۳۰۴۵۸۳ * {t_f}^۲ - ۰٫۰۰۰۶۴۸۳۴۱۱ * T_{op} \\
&- ۰٫۰۰۰۰۰۰۹۲۳۲۹۷۱ * {T_{op}}^۲ - ۰٫۰۲۲۵۹۷۰۲ * t_b * L_c \\
&- ۰٫۰۰۴۷۳۵۶۵۲ * t_b * {L_c}^۲ + ۰٫۱۱۰۲۴۴۲ * {t_b}^۲ * L_c - ۰٫۰۰۹۷۰۲۵۳۳ \\
&* {t_b}^۲ * {L_c}^۲ + ۰٫۰۰۵۳۸۲۲۱۱ * t_b * L_f - ۰٫۰۰۰۰۹۵۴۰۴۸۴ * t_b * {L_f}^۲ \\
&* ۰٫۰۰۵۱۵۰۴۸ * {t_b}^۲ * L_f - ۰٫۰۰۰۱۲۳۲۵۲۴ * {t_b}^۲ * {L_f}^۲ + ۰٫۲۹۷۲۵۸۹ \\
&* t_b * t_f - ۰٫۱۰۵۲۹۳۵ * t_b * {t_f}^۲ - ۰٫۵۴۲۲۲۶۲ * {t_b}^۲ * t_f - ۰٫۱۸۲۹۶۸۷ \\
&* {t_b}^۲ * {t_f}^۲
\end{aligned}
\tag{۱۸.۴}
$$

جرم کل مینیمم شده و به‌صورت زیر ارائه می‌شود.

$$
\begin{aligned}
M &= f(L_f, L_c, t_f, t_b) \\
&= (۱۳۱۳٫۸۷۷ - ۷۵۵ * L_c + ۱۱٫۰ * {L_c}^۲ + ۱٫۴۰۲۵۹۷ * L_f) \\
&- (۱٫۲۷۸۳۱۴e - ۱۵) * {L_f}^۲ + ۶۲٫۳۸۷۷۶ * t_f - ۶٫۱۲۲۴۴۹ * {t_f}^۲ \\
&- ۳۸۰٫۸ * t_b + ۱۱۲۰ * {t_b}^۲) * ۲۱
\end{aligned}
\tag{۱۹.۴}
$$

بهترین جواب توافقی

برای یک مسئله کاربردی، یک جواب دقیق از مجموعه بهینه پارتو به‌دست آمده، نیاز است. چون هر جواب همه توابع هدف را تا درجه‌ای برآورده می‌کند، رویکرد عضویت فازی [۱۶] برای انتخاب بهترین جواب توافقی در این پایان‌نامه استفاده شده است. در ابتدا، هر تابع هدف برای هر جواب در مجموعه غیر تسلطی به‌دست آمده، توسط یک تابع عضویت به‌صورت زیر تعریف می‌شود:

$$\mu_k^i = \begin{cases} ۱; f_k < {f_k}^{min} \\ \frac{{f_k}^{max} - f_k}{f_k - {f_k}^{min}}; {f_k}^{min} < f_k < {f_k}^{max} \\ ۰; f_k > {f_k}^{max} \end{cases} \quad (۲۰.۴)$$

که ${f_k}^{min}$ و ${f_k}^{max}$ به ترتیب مقدار مینیمم و ماکزیمم kـامین تابع هدف در مجموعه غیر تسلطی تقریبی هستند و i نشان دهنده i ـامین جواب است.
سپس، تابع عضویت نرمال‌شده μ^i برای هر جواب i ام به صورت زیر محاسبه می‌شود:

$$\mu^i = \frac{\sum_{k=۱}^{M} \mu_k^i}{\sum_{i=۱}^{N} \sum_{k=۱}^{M} \mu_k^i} \quad (۲۱.۴)$$

که M تعداد توابع هدف را نشان می‌دهد و N تعداد جواب‌های غیرتسلطی است. سرانجام، جواب i ی که مقدار ماکزیمم μ^i را دارد، به عنوان بهترین جواب توافقی انتخاب می‌شود.

تنظیم پارامتر الگوریتم‌ها

برای تست سازگاری الگوریتم SAMOHS، تمام پارامترهایی که در بخش‌های قبل ذکر شدند (به جز ماکزیمم تعداد محاسبه) مورد استفاده قرار می‌گیرند. برای الگوریتم‌های MOHS۱ و MOHS۲ نیز به همین شکل است. برای مقایسه عادلانه همه الگوریتم‌ها، ماکزیمم تعداد محاسبه توابع ۵۰،۰۰۰ در نظر گرفته می‌شود.

نتایج

بهترین جواب طراحی لوله‌های گرمایی ماهواره با استفاده از این سه الگوریتم، در جدول ۱۰.۴ ارائه شده است. در این جدول، BCS نشان دهنده بهترین جواب توافقی برای این مسئله بهینه‌سازی [۶۷] است. از جدول ۱۰.۴ مشاهده می‌شود که الگوریتم SAMOHS پیشنهاد شده بهترین مقدار هدایت حرارتی را برابر ۰٫۳۸۰۸$(\frac{W}{K})$ و بهترین مقدار جرم کل را برابر ۲۵٫۸۶۷۸(kg) نشان می‌دهد. بنابراین می‌توان ادعا کرد که الگوریتم SAMOHS پیشنهاد شده از نظر جست‌وجوی جواب‌های رأسی بهتر از MOHS۱ و MOHS۲ اجرا می‌شود. BFGS[۳۰] که یکی از شیوه‌های کلاسیک ریاضی است (روشی در محاسبات عددی بهینه‌سازی برای برنامه‌سازی غیرخطی بدون قید است که این روش تقریبی برای روش بهینه سازی نیوتون است)، مقدار ماکزیمم هدایت حرارتی ۰٫۳۸۰۸ را پیدا می‌کند و مقدار مینیمم جرم کل

[۳۰] Broyden Fletcher Goldfarb Shanno

(kg) ۲۵٫۸۶۸۰ را برای مسئله بهینه‌سازی تک هدفه پیدا می‌کند. از جدول ۱۰.۴ مقدار مینیمم جرم کل به‌دست آمده با الگوریتم SAMOHS به‌دست می‌آید که کوچکتر از BFGS است؛ در حالی‌که ماکزیمم هدایت حرارتی به‌دست آمده با الگوریتم SAMOHS و BFGS برابر است. گیم و هانگبو [۶۷] یک جواب توافقی به‌دست آوردند که در آن، مقدار هدایت حرارتی برابر $(\frac{W}{K})$ ۰.۳۸۱۰ و مقدار جرم کل برابر (kg) ۲۶٫۷۰۴ می‌باشد. واضح است که بهترین جواب توافقی با SAMOHS به‌دست می‌آید و جواب به‌دست آمده با [۶۷] تسلطی نیست. به این معنی که بهترین جواب توافقی به‌دست آمده با SAMOHS نیز موجود و مؤثر است.

طبق نتایج فوق، می‌توان نتیجه گرفت که الگوریتم SAMOHS برای حل این مسئله مهندسی مؤثر و قوی است و بهترین جواب توافقی براساس عضویت فازی با ارزش و معنی‌دار است. علاوه بر این الگوریتم SAMOHS وقتی با الگوریتم‌های MOHS۱ و MOHS۲ برای حل مسائل مهندسی مقایسه می‌شود، بهتر عمل می‌کند.

الگوریتم‌ها	SAMOHS		MOHS۱			MOHS۲		
طراحی پارامترها	بهترین مقدار هدایت	بهترین جرم	BCS	بهترین مقدار هدایت	بهترین جرم	BCS	بهترین مقدار هدایت	بهترین جرم	BCS
L_f	۱۹.۱۲۶۲	۱۰.۰۰۰۰	۱۰.۸۲۳۹	۱۲.۹۵۹۸	۱۰.۰۲۱۰	۱۰.۰۲۱۰	۱۸.۵۳۶۷	۱۰.۵۲۵۵	۱۲.۰۱۵۳
L_c	۱.۵۰۰۰	۲.۵۰۰۰	۲.۴۵۶۲	۱.۷۹۷۰	۲.۴۵۸۰	۲.۴۳۱۳	۱.۶۱۲۱	۲.۴۷۶۷	۲.۴۱۸۴
t_f	۱.۷۰۰۰	۱.۰۰۰۰	۱.۰۰۰۰	۱.۶۶۹۵	۱.۰۰۸۶	۱.۰۰۷۷	۱.۶۷۹۷	۱.۰۰۰۹	۱.۰۰۰۰
t_b	۰.۱۲۰۰	۰.۱۷۱۳	۰.۱۲۰۰	۰.۱۲۰۱	۰.۱۷۳۳	۰.۱۲۰۱	۰.۱۲۰۰	۰.۱۶۰۰	۰.۱۲۰۰
T_{op}	−۲۰.۰۰۰۰	−۲۰.۰۰۰۰	−۲۰.۰۰۰۰	−۱۹.۹۶۱۶	−۱۸.۵۹۲۷	−۱۹.۸۸۷۵	−۲۰.۰۰۰۰	−۲۰.۰۰۰۰	−۲۰.۰۰۰۰
مقدار هدایت حرارتی $(\frac{W}{K})$	۰.۳۸۰۸	۰.۳۲۱۵	۰.۳۵۵۳	۰.۳۷۶۷	۰.۳۲۰۴	۰.۳۵۴۸	۰.۳۸۰۳	۰.۳۲۹۴	۰.۳۵۷۱
جرم کل (Kg)	۲۷.۵۳۱۰	۲۵.۸۶۷۸	۲۵.۹۷۰۲	۲۷.۰۷۷۷	۲۵.۸۹۶۲	۲۵.۹۶۵۷	۲۷.۳۹۸۷	۲۵.۸۹۶۷	۲۶.۰۲۲۶

جدول ۱۰.۴: بهترین جواب از لوله‌ی گرمایی ماهواره

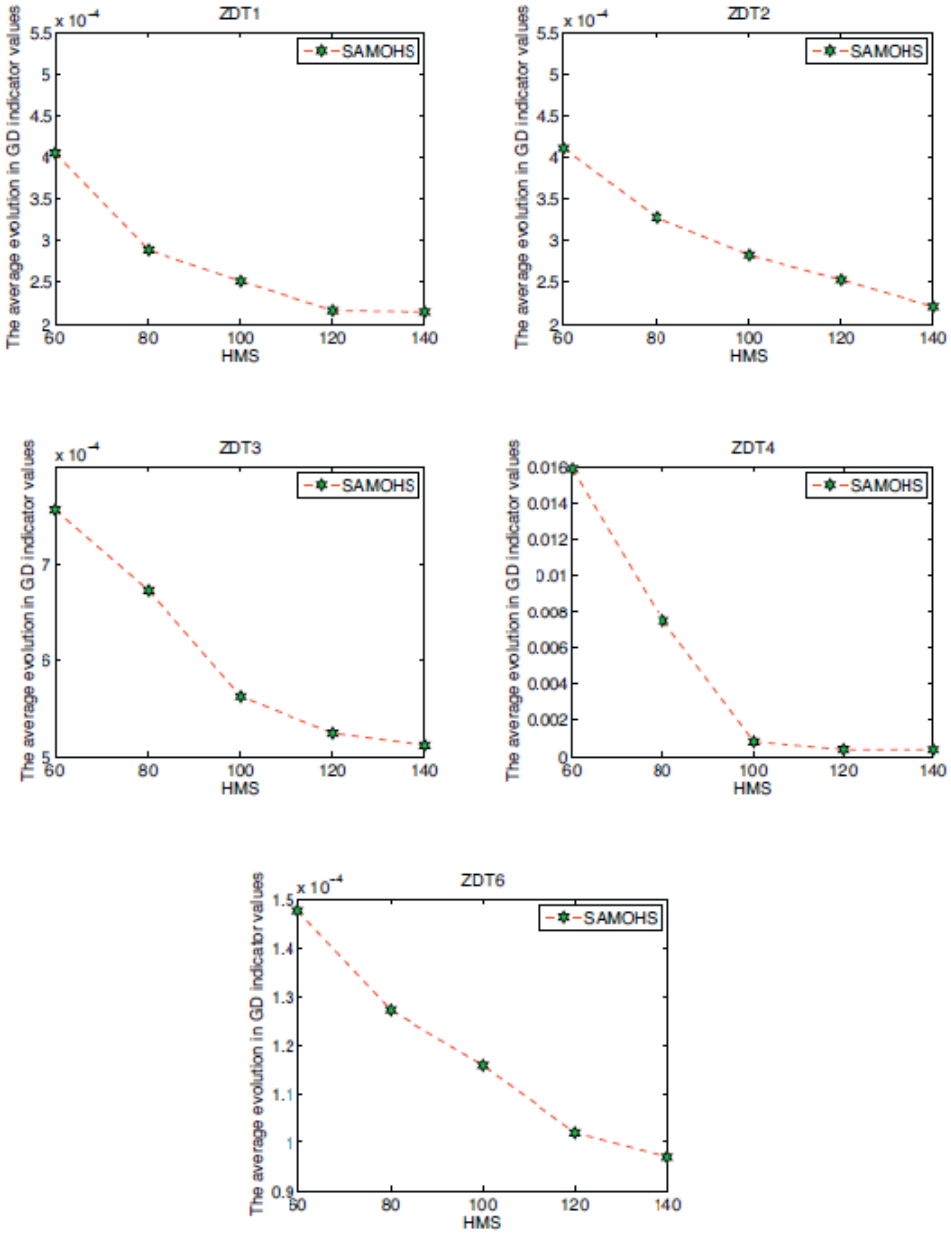
۱.۴.۴ اندازه حافظه هارمونی متفاوت برای الگوریتم SAMOHS

اجرای الگوریتم SAMOHS با استفاده از اندازه حافظه هارمونی در اینجا بررسی می‌شود. مقادیر ۶۰، ۸۰، ۱۰۰، ۱۲۰ و ۱۴۰ برای اندازه حافظه هارمونی برای حل مسائل ZDT امتحان می‌شوند. پارامترهای کنترلی که برای الگوریتم SAMOHS استفاده می‌شوند مشابه پارامترهای موجود در جدول ۵.۴ هستند، به جز اندازه حافظه هارمونی و تعداد ماکزیمم تکرار T که در اینجا T برابر با ۲۵۰ است. چون که جواب‌های غیر تسلطی از حافظه هارمونی استخراج می‌شوند و الگوریتم SAMOHS آرشیو خارجی استفاده نمی‌کند، تعداد جواب‌های غیر تسلطی به‌دست آمده با SAMOHS کمتر یا مساوی با حافظه هارمونی هستند. به این معنی که، تعداد جواب‌های غیر تسلطی متفاوت می‌شود هنگامی که از اندازه‌های متفاوتی برای حافظه هارمونی استفاده کنیم. بنابراین، مقایسه تنوع جواب‌های غیر تسلطی برای اندازه‌های متفاوت حافظه‌هارمونی، نامناسب است. در نتیجه، تنها همگرایی که در نظر گرفته می‌شود با استفاده از شاخص GD است. نتایج تجربی به‌دست آمده با الگوریتم SAMOHS با استفاده از اندازه‌های متفاوت حافظه هارمونی در ۳۰ اجرا در جدول ۱۱.۴ ارائه شده‌اند. ارزیابی میانگین مقادیر شاخص GD نسبت به اندازه‌های مختلف حافظه هارمونی به وسیله الگوریتم SAMOHS در شکل ۱۸.۴ شرح داده شده است. با افزایش اندازه حافظه هارمونی مقادیر شاخص GD بهتر می‌شود. هر چند زمان تعداد اجرا با الگوریتم SAMOHS طولانی‌تر می‌شود. بنابراین مناسب است که اندازه حافظه

هارمونی بین ۱۲۰ ـ ۸۰ انتخاب شود.

پارامترها	HMS = ۶۰		HMS = ۸۰		HMS = ۱۰۰		HMS = ۱۲۰		HMS = ۱۴۰	
مسائل تست	میانگین	SD	میانگین	SD	میانگین	SD	میانگین	SD	میانگین	SD
ZDT۱	۴.۰۵۳۶e-۴	۱.۰۵۴۱e-۴	۲.۸۸۹۵e-۴	۴.۵۵۹۹e-۵	۲.۵۱۵۷e-۴	۵.۸۶۶۲e-۵	۲.۱۶۲۵e-۴	۴.۰۰۲۷e-۵	**۲.۱۴۶۳e-۴**	**۲.۱۴۶۳e-۴**
ZDT۲	۴.۱۱۰۰e-۴	۹.۶۱۵۸e-۵	۳.۲۸۰۶e-۴	۶.۰۷۶۴e-۵	۲.۸۲۸۸e-۴	۵.۲۱۲۱e-۵	۲.۵۳۲۲e-۴	۳.۹۸۵۲e-۵	**۲.۲۰۸۹e-۴**	**۲.۴۸۷۳e-۵**
ZDT۳	۷.۵۵۵۴e-۴	۶.۰۰۴۲e-۵	۶.۷۲۶۳e-۴	۳.۸۳۳۲e-۵	۵.۶۱۷۲e-۴	۴.۷۶۷۰e-۵	۵.۲۴۰۵e-۴	۳.۴۳۲۰e-۵	**۵.۱۱۵۹e-۴**	**۴.۳۰۲۶e-۵**
ZDT۴	۱.۵۸۷۴e-۲	۱.۳۴۸۶e-۲	۷.۴۶۰۹e-۳	۶.۶۵۳۸e-۳	۸.۱۳۶۴e-۴	۲.۱۶۱۲e-۳	۳.۸۱۹۲e-۴	۱.۶۰۱۲e-۴	**۳.۷۴۷۰e-۴**	**۱.۹۳۶۱e-۴**
ZDT۶	۱.۴۷۶۰e-۴	۱.۲۳۹۴e-۵	۱.۲۷۱۷e-۴	۱.۱۵۹۵e-۵	۱.۱۵۷۹e-۴	۱.۱۰۰e-۵	۱.۰۱۹۲e-۴	۷.۹۷۵۲e-۶	**۹.۶۹۱۸e-۵**	**۸.۰۴۶۲e-۶**

جدول ۱۱.۴: مقادیر شاخص GD به‌دست آمده با الگوریتم SAMOHS برای اندازه‌های متفاوت حافظه هارمونی.



شکل ۱۸.۴: ارزیابی میانگین مقدار شاخص GD در اندازه حافظه هارمونی متفاوت برای الگوریتم SAMOHS

فصل ۵

نتیجه‌گیری

الگوریتم جست‌وجوی هارمونی به دلیل کاربردی بودن برای مسائل بهینه‌سازی پیوسته و گسسته، محاسبات ریاضیاتی کم، پارامترهای کم و اجرای آسان به یکی از پر کاربردترین الگوریتم‌های بهینه‌سازی در مسائل مختلف تبدیل شده است و می‌توان آن را در مسائل مختلف مهندسی با تغییر در پارامترها و عملگرها تطبیق نمود. این الگوریتم توجه زیادی به خود معطوف کرده به طوری که تا کنون در مسائل بهینه‌سازی عملی نظیر بهینه‌سازی ساختاری، طراحی بهینه شبکه‌های توزیع آب، مسائل مسیریابی، طراحی اسکلت‌های فلزی، مدل‌های انتقال انرژی، مدل‌سازی حفظ محیط زیست و غیره به کار رفته است. از ویژگی‌های دیگر الگوریتم جست‌وجوی هارمونی این است که در مدت زمان مناسبی فضاهای حل با محدوده عملکرد بهتری را شناسایی می‌کند. این ویژگی در صورتی که مسئله مورد مطالعه از بهینگی محلی برخوردار باشد دچار مشکل می‌شود و در بهینگی محلی متوقف شده و نمی‌تواند به بهینگی سراسری برسد. دلیل این مشکل، عدم کارایی مناسب الگوریتم در اجرای جست‌وجوی محلی در مسائل بهینه‌سازی گسسته است. به منظور رفع این مشکل و تطبیق روش حل با مسئله، محققان با تغییر در پارامترهای الگوریتم انواع مختلفی از این الگوریتم ارائه کردند تا دقت حل و نرخ همگرایی افزایش یابد. پارامترهای PAR و HMCR دو پارامتری هستند که نقش اساسی در تنظیم صحیح بردارهای حل جدید دارند. در فصل سوم، دو طرح برای تحلیل مسائل چندهدفه با استفاده از جست‌وجوی هارمونی پیشنهاد شد که نتایج محاسباتی نشان می‌دهند که این الگوریتم‌ها در مقایسه با الگوریتم NSGA-II که در حال حاضر به عنوان یکی از رایج‌ترین الگوریتم‌های تکاملی بهینه‌سازی چندهدفه می‌باشد، قابل رقابت هستند.

به منظور بهبود عملکرد الگوریتم جست‌وجوی هارمونی برای حل مسائل بهینه‌سازی چند هدفه، یک

جست‌وجوی هارمونی خود انطباق براساس واریانس حافظه هارمونی در فصل چهارم پیشنهاد شد. برای نشان دادن کارایی الگوریتم SAMOHS این الگوریتم با مسائل معیاری زیادی تست شده است. نتایج به‌دست آمده از مقایسه الگوریتم‌های MOHS۱ و MOHS۲ با الگوریتم SAMOHS پیشنهادشده نشان دادکه این الگوریتم نتایج بهتری از نظر همگرایی و تنوع دارد و به علاوه عملکرد الگوریتم SAMOHS پیشنهادشده با استفاده از مکانیزم خود انطباقی پیشنهادی نتایج بهتری می‌دهد. نتایج محاسباتی نشان می‌دهند که الگوریتم SAMOHS قابل رقابت با الگوریتم‌های تکاملی چندهدفه دیگر می‌باشد و الگوریتم SAMOHS عملکرد بهتری از الگوریتم‌های تکاملی چند هدفه دیگر دارد. همچنین می‌توان نتیجه گرفت که الگوریتم SAMOHS پیشنهاد شده، توانایی تولید جواب‌های بهینه پارتو با توزیع خوب و درست را دارد.

با بررسی این نتایج امید بخش، عرصه بعدی کار پیشنهاد می‌شود:

(۱) مطالعه عملکرد الگوریتم‌های پیشنهاد شده در فصل سوم با استفاده از بستر آزمایش پیشنهاد شده در CEC ۲۰۰۹ [۵۳]؛

(۲) تحلیل موقتی رفتار الگوریتم‌ها در حل مسائل بررسی‌شده، برای دست‌یابی به نتایج استاندارد با توجه به زمان اجرا؛

(۳) به کارگیری طرح‌های جدید برای تحلیل مسائل کلاسیک در بهینه‌سازی ترکیبی، برای مثال مسئله فروشنده دوره‌گرد (TSP) [۵۸].

(۴) تأیید اعتبار طرح‌های الگوریتم‌های جست‌وجوی هارمونی چند هدفه با مقایسه آنها با روش‌های دیگر برای مثال بهینه‌سازی ازدحام ذره (PSO) [۳۲].

(۵) بهبود بیشتر عملکرد تنوع الگوریتم SAMOHS در حل مسائل بهینه‌سازی چند هدفه با مرز پارتوی گسسته.

(۶) بررسی اجرای الگوریتم SAMOHS با استفاده از توزیع متفاوت برای تنظیم پارامتر خود انطباق مانند توزیع کوشی.

(۷) کاربرد وسیع الگوریتم SAMOHS برای حل بیشتر مسائل مهندسی کاربردی.

پیوست آ

ضمیمه

```
clc
clear
close all
format shortG

%% Parameters Setting

nvar=2;                         % Number Of Variable
lb=-pi*ones(1,nvar);     % Lower Bound
ub=pi*ones(1,nvar);     % Upper Bound

maxiter=100;          % Maximum Of Iteration

npop=20;               % Harmony Memory Size

HMCR=0.95;           % Harmony Memory Consideration Rate
```

```
PAR=0.7;              % Piterch Adjustment Rate

BW=1;           % Bandwidth

BW_RF=0.95;      % BandWidth Reduction Factor

%% Initial Population
tic
emp.x=[];
emp.fit=[];

pop=repmat(emp,npop,1);

for i=1:npop
    pop(i).x=unifrnd(lb,ub);
    pop(i).fit=fitness(pop(i).x);
end

% Sorting
[~, ind]=sort([pop.fit]);
pop=pop(ind);

gpop=pop(1);  % Global Solutuion


%%  Main Loop

BEST=zeros(maxiter,1);
MEAN=zeros(maxiter,1);

for iter=1:maxiter

    newpop=pop;
```

```
    for i=1:npop

        % Consideration
        J=rand(1,nvar)<HMCR;
        J=find(J==1);

        for j=J
            k=randi([1 npop]);
            newpop(i).x(j)=pop(k).x(j);

% if rand<PAR
% d=BW*unifrnd(-1,1);
% newpop(i).x(j)=newpop(i).x(j)+d;
% end

        end


        % Piterch Adjustment
         J=rand(1,nvar)<PAR;
         J=find(J==1);
         d=BW*unifrnd(-1,1,size(J));
         newpop(i).x(J)=newpop(i).x(J)+d;

        % Check Bound
        newpop(i).x=CB(newpop(i).x,lb,ub);

        newpop(i).fit=fitness(newpop(i).x);

    end

    % Merge
    [pop]=[pop;newpop];
```

```
    % Sorting
    [~, ind]=sort([pop.fit]);
    pop=pop(ind);
    pop=pop(1:npop);

    gpop=pop(1);

    BW=BW*BW_RF;


    BEST(iter)=gpop.fit;
    MEAN(iter)=mean([pop.fit]);

    disp(['Iter ' num2str(iter) ' BEST = ' num2str(BEST(iter))]);
end

%% Results

disp(' ')
disp([ ' BEST solution = '  num2str(gpop.x)]);
disp([ ' BEST fitness = '  num2str(gpop.fit)]);
disp([ ' Time = '  num2str(toc)]);
figure
x=-10:0.1:10;
y=-10:0.1:10;
[X,Y]=meshgrid(x,y);
Z=log(1+(1-X.^2+100*(Y-X.^2).^2));
Z=real(Z);
surfc(X,Y,Z)
xlabel('x');
ylabel('y');
zlabel('z');

% semilogy(BEST,'r');
```

```
% hold on
% semilogy(MEAN,'b');
% xlabel('Iteration');
% ylabel('Fitness');
% legend('BEST','MEAN');
```

Iter 1 BEST = -9963.0197
Iter 2 BEST = -10052.1695
Iter 3 BEST = -10052.1695
Iter 4 BEST = -10053.348
Iter 5 BEST = -10053.348
Iter 6 BEST = -10053.348
Iter 7 BEST = -10053.348
Iter 8 BEST = -10053.348
Iter 9 BEST = -10053.348
Iter 10 BEST = -10053.348
Iter 11 BEST = -10053.348
Iter 12 BEST = -10053.348
Iter 13 BEST = -10053.348
Iter 14 BEST = -10053.348
Iter 15 BEST = -10053.348
Iter 16 BEST = -10053.348
Iter 17 BEST = -10053.348
Iter 18 BEST = -10053.348
Iter 19 BEST = -10053.348
Iter 20 BEST = -10053.348
Iter 21 BEST = -10053.348
Iter 22 BEST = -10053.348
Iter 23 BEST = -10053.348
Iter 24 BEST = -10053.348
Iter 25 BEST = -10053.348
Iter 26 BEST = -10053.348
Iter 27 BEST = -10053.348
Iter 28 BEST = -10053.348
Iter 29 BEST = -10053.348

Iter 30 BEST = -10053.348
Iter 31 BEST = -10053.348
Iter 32 BEST = -10053.348
Iter 33 BEST = -10053.348
Iter 34 BEST = -10053.348
Iter 35 BEST = -10053.348
Iter 36 BEST = -10053.348
Iter 37 BEST = -10053.348
Iter 38 BEST = -10053.348
Iter 39 BEST = -10053.348
Iter 40 BEST = -10053.348
Iter 41 BEST = -10053.348
Iter 42 BEST = -10053.348
Iter 43 BEST = -10053.348
Iter 44 BEST = -10053.348
Iter 45 BEST = -10053.348
Iter 46 BEST = -10053.348
Iter 47 BEST = -10053.348
Iter 48 BEST = -10053.348
Iter 49 BEST = -10053.348
Iter 50 BEST = -10053.348
Iter 51 BEST = -10053.348
Iter 52 BEST = -10053.348
Iter 53 BEST = -10053.348
Iter 54 BEST = -10053.348
Iter 55 BEST = -10053.348
Iter 56 BEST = -10053.348
Iter 57 BEST = -10053.348
Iter 58 BEST = -10053.348
Iter 59 BEST = -10053.348
Iter 60 BEST = -10053.348
Iter 61 BEST = -10053.348
Iter 62 BEST = -10053.348
Iter 63 BEST = -10053.348

Iter 64 BEST = -10053.348
Iter 65 BEST = -10053.348
Iter 66 BEST = -10053.348
Iter 67 BEST = -10053.348
Iter 68 BEST = -10053.348
Iter 69 BEST = -10053.348
Iter 70 BEST = -10053.348
Iter 71 BEST = -10053.348
Iter 72 BEST = -10053.348
Iter 73 BEST = -10053.348
Iter 74 BEST = -10053.348
Iter 75 BEST = -10053.348
Iter 76 BEST = -10053.348
Iter 77 BEST = -10053.348
Iter 78 BEST = -10053.348
Iter 79 BEST = -10053.348
Iter 80 BEST = -10053.348
Iter 81 BEST = -10053.348
Iter 82 BEST = -10053.348
Iter 83 BEST = -10053.348
Iter 84 BEST = -10053.348
Iter 85 BEST = -10053.348
Iter 86 BEST = -10053.348
Iter 87 BEST = -10053.348
Iter 88 BEST = -10053.348
Iter 89 BEST = -10053.348
Iter 90 BEST = -10053.348
Iter 91 BEST = -10053.348
Iter 92 BEST = -10053.348
Iter 93 BEST = -10053.348
Iter 94 BEST = -10053.348
Iter 95 BEST = -10053.348
Iter 96 BEST = -10053.348
Iter 97 BEST = -10053.348

Iter 98 BEST = -10053.348

Iter 99 BEST = -10053.348

Iter 100 BEST = -10053.348

BEST solution = 3.1416 -3.1416

BEST fitness = -10053.348

Time = 1.8779

مراجع

[۱] رستگار ا. صحرائیان ر، ’’توسعه روش جستجوي هماهنگي در حل مسائل بهینه‌سازي: مطالعه موردي در زمان بندي توليد ماشين هاي موازی’’، (۱۳۹۲)، **نشریه پژوهش های مهندسی صنایع در سیستم های تولید**، شماره اول، ص ۶۱.

[۲] عالم تبریز ا، زندیه م، محمد رحیمی ع، (۱۳۹۲)، **’’الگوریتم‌های فراابتکاری در بهینه‌سازی ترکیبی’’** چاپ سوم، انتشارات صفار، صفحات ۱۸-۱۶.

[۳] فتاحی پ، (۱۳۹۰)، **’’ الگوریتم‌های فراابتکاری’’** چاپ دوم، انتشارات دانشگاه بوعلی سینا، صفحات ۱۸۳-۱۰۲.

[۴] یقینی م ، اخوان کاظم‌زاده م، (۱۳۹۰)، **’’ الگوریتم‌های بهینه‌سازی فراابتکاری’’** چاپ اول، انتشارات جهاد دانشگاهی واحد صنعتی امیر کبیر، صفحات ۲۱۶-۱۰.

[۵] نصری م، نصر اصفهانی ح، (۱۳۹۰)، **’’ الگوریتم فراابتکاری جست‌وجوی هارمونی’’** چاپ اول، انتشارات ناقوس، صفحات ۲۲۲-۱۸.

[6] Ali M. M. and Torn A. and Viitanen S. (2002), ”A direct search variant of the simulated annealing algorithm for optimization involving continuous variables”. **Computers Operations Research**, 29(1), (pp. 87-102).

[7] Cheng Y.M. and Li L. and Chi S.C. (2007), ”Performance studies on six heuristic global optimization methods in the location of critical slip surface”. **Computers and Geotechnics** 34, (pp. 462-484).

[8] Coello C.A.C and Pulido G.T. and Lechuga M.S. (2004), ”Handling multiple objectives with particle swarm optimization”, **IEEE Transactions on evolutionary computation** 8.3, (pp. 256-279).

[9] Dai X. and Yuan X. and Zhang Z. (2015), ”A self-adaptive multi-objective harmony search algorithm based on harmony memory variance”, **Applied Soft Computing**, 35, (pp. 541-557).

[10] Deb K. and Pratap A. and Agarwal S. and Meyarivan T.A.M.T. (2002), ”A fast and elitist multiobjective genetic algorithm: NSGA-II”, **IEEE Transactions on Evolutionary Computation**, 6(2). (pp. 182-197).

[11] Deb K. and Thiele L. and Laumanns M. and Zitzler E. (2001), ”Scalable Test Problems for Evolutionary Multi-Objective Optimization” Zurich. Switzerland, Tech. Rep. 112.

[12] Deb K. (2010), ”Multiobjective NSGA-II code in C”, Accessed July 2010. URL http://www.iitk.ac.in/kangal/codes/nsga2/nsga2-gnuplot-v1.1.5-64bit.tar.gz

[13] De Sousa F.L. and Vlassov V. and Ramos F.M. (2004), ”Generalized extremal optimization: An application in heat pipe design”, **Applied Mathematical Modelling**, 28(10), (pp. 911-931).

[14] Degertekin S.O. (2008), ”Harmony search algorithm for optimum design of steel frame structures: a comparative study with other optimization methods”, **Structural Engineering and Mechanics**, 29(4), (pp. 391-410).

[15] Degertekin S.O. (2008), ” Optimum design of steel frames using harmony search algorithm”, **Structural and Multidisciplinary Optimization** 36.4, (pp. 393-401).

[16] Dhillon J. and Parti S.C. and Kothari D.P. (1993), ”Stochastic economic emission load dispatch”, **Electric Power Systems Research**, 26(3), (pp. 179-186).

[17] Erdal F. and Saka M.P. (2008), ”Effect of beam spacing in the harmony search based optimum design of grillages”, **Asian Journal of Civil Engineering**, 9(3), (pp. 215-228).

[18] Gao X.Z. and Wang X. and Ovaska S.J. (2008), ”Modified harmony search methods for uni-modal and multi-modal optimization”. In Hybrid Intelligent Systems. HIS’08. **Eighth International Conference on**. (pp. 65-72). IEEE.

[19] Geman S. and Geman D. (1984), ”Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images”, **IEEE Transactions on Pattern Analysis and Machine Intelligence**, (6). (pp. 721-741).

[20] Geem Z.W. (2006), ”Optimal cost design of water distribution networks using harmony search”, **Engineering Optimization**, 38(03). (pp. 259-277).

[21] Geem Z.W. (2009), ”Multiobjective optimization of time-cost trade-off using harmony search”, **Journal of Construction Engineering and Management**, 136.6. (pp. 711-716).

[22] Geem Z.W. and Kim J.H and Loganathan G. (2001), ”A new heuristic optimization algorithm: Harmony search”, **Simulation** 76. (pp. 60–68).

[23] Geem Z.W. (2010), ”State-of-the-art in the structure of harmony search algorithm”, in: Recent Advances in Harmony Search Algorithm, **Studies in Computational Intelligence**, 270, Springer. (pp. 1–10).

[24] Geem Z.W. and Lee K.S. and Park Y. (2005), ”Application of harmony search to vehicle routing”, **American Journal of Applied Sciences**, 2(12), 1552-1557.

[25] Geem Z.W. (2006), ”Optimal cost design of water distribution networks using harmony search”, **Engineering Optimization**, 38(03), (pp. 259-277).

[26] Glover F. (1986), ”Future paths for integer programming and links to artificial intelligence”, **Computers operations research**, 13(5), (pp. 533-549).

[27] Goldberg D.E. and Deb K. (1991), ”A comparative analysis of selection schemes used in genetic algorithms”, **Foundations of genetic algorithms**, 1, (pp. 69-93).

[28] Haimes Y.Y. and Ladson L.S. and Wismer D.A. (1971), ”Bicriterion formulation of problems of integrated system identification and system optimization”, **IEEE Transactions on Systems Man and Cybernetics**, (3), 296.

[29] Jeong M.J. and Kobayashi T. and Yoshimura S. (2005), ”Extraction of design characteristics of multiobjective optimization–its application to design of artificial satellite heat pipe”, **In International Conference on Evolutionary Multi-Criterion Optimization**. (pp. 561-575). Springer Berlin Heidelberg.

[30] Johnson D.C and Aragon C.R and McGeoch L.A and Schevon C. (1989), ”Optimization by simulated annealing: An experimental evalution”, **Part I. Graph partitioning. Operations Reesearch**, 37. (pp. 865-892).

[31] Derrac J. and Garcia S. and Molina D. and Herrera F. (2011), ”A practical tutorial on the use of non-parametric statistical tests as a methodology for comparing evolutionary andswarm intelligence algorithms”, **Swarm Evol. Comput**. 1 (1). (pp. 3–18).

[32] Kennedy J. and Eberhart R.C. (1942), ”Quot; particle swarm optimization quot”, In Proc. **IEEE Intl. Conf. on Neural Networks**, pp. IV (Vol. 1948, pp. 1995).

[33] Kirkpatrick S. and Gelatt C.D. and Vecchi M.P. (1983), ”Optimization by simulated annealing”, **Science**, 220(4598). (pp 671-680).

[34] Knowles J.D. and Corne D.W. (2000), ”M-PAES: A memetic algorithm for multiobjective optimization”, **Evolutionary Computation**, 2000. Proceedings of the 2000 Congress on. Vol. 1. IEEE.

[35] Li L. and Chi S. and Cheng Y. and Lin G. (2008), ”Improved genetic algorithm and its application to determination of critical slip surface with arbitrary shape in soil slope”, **Frontiers of Architecture and Civil Engineering in China**, 2(2). (pp. 145-150).

[36] Lima J.Q. and Baran B. (2006), ”Optimization de enjambre de particulas aplicada al problema del cajero viajante bi-objetivo”, **Inteligencia Artificial: Revista Iberoamericana de Inteligencia Artificial** 10.32. (pp. 67-76).

[37] Pavelski L.M. and Almeida C.P. and Gonc R.A. (2012), ”Harmony search for multiobjective optimization”, in: 2012 **Brazilian Symposium on Neural Networks**. (pp. 220–225).

[38] Mahdavi M. and Fesanghary M. and Damangir E. (2007), ”An improved harmony search algorithm for solving optimization problems”, **Applied Mathematics and Computation**, 188(2). (pp. 1567-1579).

[39] Mahdavi M. and Chehreghani M.H. and Abolhassani H. and Forsati R. (2008), ”Novel meta-heuristic algorithms for clustering web documents”, **Applied Mathematics and Computation**, 201(1). (pp. 441-451).

[40] Metropolis N. and Rosenbluth A.W. and Rosenbluth M.N. and Teller A.H. and Teller E. (1953), ”Equation of state calculations by fast computing machines”, **The Journal of Chemical Physics**, 21(6). (pp. 1087-1092).

[41] Reinaldo M.J. and DePuy G.W. and Whitehouse G.E. (2006), ”Metaheuristics: A Solution Methodology for Optimization Problems”, **Handbook of Industrial and Optimization Problems, Handbook of Industrial and Systems Engineering, AB Badiru (Ed.)**

[42] Pan Q.K. and Suganthan P.N. and Tasgetiren M.F. and Liang J.J. (2010), ”A self-adaptive global best harmony search algorithm for continuous optimization problems”, **Applied Mathematics and Computation**, 216(3). (pp. 830-848).

[43] Ricart J. and Huttemann G. and Lima J. and Baran B. (2011), ”Multiobjective harmony search algorithm proposals”, **Electronic Notes in Theoretical Computer Science**, 281. (pp. 51-67).

[44] Romeo F. and Sangiovanni V.A. and Huang M. (1986), ”An efficient general cooling schedule for simulated annealing”, **Proceeding of IEEE International Conference on Computer aided design**.

[45] Saab Y. and Rao V. (1991), ”Combinational optimization by stochastic evolution”, **IEEE Transactions on Computer-Aided Design**, 10. (pp. 525-535).

[46] Garca S. and Molina D. and Lozano M and Herrera F. (2009), ”A study on the use of non-parametrictests for analyzing the evolutionary algorithm behaviour” A Case Study on the CEC’2005 Special Session on Real Parameter Optimization, **Journal of Heuristics**. 15 (6). (pp. 617–644).

[47] Sivasubramani S. and Swarup K.S. (2011), ”Multi-objective harmony search algorithm for optimal power flow problem”, **International Journal of Electrical Power Energy Systems** 33.3. (pp. 745-752).

[48] Veldhuizen V. and David A and Lamont G.B. (2000), ”Multiobjective evolutionary algorithms: Analyzing the state-of-the-art”, **Evolutionary Computation**, 8(2). (pp. 125-147).

[49] Rajesh V.G. and Ravindran K.P. (1997), ”Optimum heat pipe design: A nonlinear programming approach”, **International Communications in Heat and Mass Transfer**. Commun. Heat Mass Transf. 24 (3). (pp. 371–380).

[50] Wang, Y.N and Wu L.H and Yuan X.F. (2010), ”Multi-objective self-adaptive differential evolution with elitist archive and crowding entropy-based diversity measure”, **Soft Computing** 14.3. (pp. 193-209).

[51] Wang Y. and Li H.X. and Huang T. and Li L. (2014), ”Differential evolution based on covariance matrix learning and bimodal distribution parameter setting”, **Applied Soft Computing**, 18. (pp. 232-247).

[52] Wang Y.N. and Wu L.H. and Yuan X.F. (2010), ”Multi-objective self-adaptive differential evolution with elitist archive and crowding entropy-based diversity measure”, **Soft Comput**. 14 (3), (pp. 193–209).

[53] Zhang Q. and Zhou A. and Zhao S. and Suganthan P.N. and Liu W. and Tiwari S. (2008), ”Multiobjective optimization test instances for the CEC 2009 special session and competition”, **University of Essex, Colchester, UK and Nanyang technological University, Singapore, special session on performance assessment of multi-objective optimization algorithms, technical report**, pp 264.

[54] Zitzler E. and Laumanns M. and Thiele L. (2001), ”SPEA2: Improving the strength Pareto evolutionary algorithm”.

[55] Zitzler E. and Thiele L. (1999), ”Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach” **IEEE transactions on Evolutionary Computation**, 3(4), (pp. 257-271).

[56] Zitzler E. and Deb K. and Thiele L. (2000), ”Comparison of multiobjective evolutionary algorithms: empirical results”, **Evolutionary Computation** 8, (pp. 173–195).

[57] Ackley, D.H. and Littman M.L. (1994). ”**Altruism in the evolution of communication**” In Artificial Life IV, (pp. 40-48).

[58] Applegate, D.L. and Bixby R.E. and Chvatal V. and Cook W.J. (2011). ”**The traveling salesman problem: a computational study**” Princeton university press.

[59] Corne D.W. and Jerram N.R. and Knowles J.D. and Oates M.J. (2001), ”**PESA-II: Region-based selection in evolutionary multiobjective optimization**”, Proceedings of the Genetic and Evolutionary Computation Conference (GECCO).

[60] Coello C.A.C.C. (2001), ”**A short tutorial on evolutionary multiobjective optimization**”. In International Conference on Evolutionary Multi-Criterion Optimization, (pp. 21-40).

[61] Dorigo M. and Di Caro G. (1999), ”**Ant colony optimization: A new metaheuristic, evolutionary computation**” In CEC 99. Proceedings of the 1999 Congress on(Vol. 2).

[62] Eiben A.E. and James S.E. (2003) "**Introduction to evolutionary computing**" Vol. 53. Heidelberg: Springer.

[63] Enrique, A.L. and Cuadra L. and Gil-Pita. R. (2009), "**Sound classification in hearing aids by the harmony search algorithm**" Music-Inspired Harmony Search Algorithm. Springer Berlin Heidelberg, (pp. 173-188).

[64] Ehrgott M. (2005), "**Multicriteria Optimization, second ed**", Springer Science , Business Media.

[65] Fonseca C.M. and Fleming P.J. (1993), "**Genetic Algorithms for Multiobjective Optimization: Formulation Discussion and Generalization**" In Icga(Vol. 93, No. July, pp. 416-423).

[66] Gao X.Z. and Wang X. and Ovaska S.J. (2009), "**Harmony search methods for multi-modal and constrained optimization**" Music-inspired harmony search algorithm. Springer Berlin Heidelberg. (pp. 39-51).

[67] Geem Z.W. and Hwangbo H. (2006), "**Application of harmony search to multi-objective optimization for satellite heat pipe design**", In Proceedings of(pp. 1-3).

[68] Madaune-Tort M. (2003), "**VII Jornadas Zaragoza-Pau de Matemática Aplicada y estadística**". Jaca (Huesca), 17-18 de septiembre de 2001.

[69] Mukhopadhyay A. and Roy A. and Das S. and Das S. and Abraham A. (2008). "**Population-variance and explorative power of harmony search**" an analysis. In Digital Information Management. ICDIM 2008. Third International Conference on. (pp. 775-781). IEEE.

[70] Talbi, El-Ghazali. "**Metaheuristics: From Design to Implementation**", John Willey and sons (2009).

[71] K. Miettinen. (1999). "**Nonlinear Multiobjective Optimization**", Springer US, p. 298.

[72] Veldhuizen V. and David A and Lamont G.B (2000), "**On measuring multiobjective evolutionary algorithm performance**" In Evolutionary Computation, 2000. Proceedings of the 2000 Congress on. (pp. 204-211). IEEE.

[73] Xu H. and Gao X.Z. and Wang T. and Xue K. (1991), ”**Harmony search optimization algorithm: application to a reconfigurable mobile robot prototype**”. In Recent Advances in Harmony Search Algorithm. (pp. 11-22). Springer Berlin Heidelberg.

[74] Yang X.S. (2009), ”**Harmony Search as a Metaheuristic Algorithm, in: Music-Inspired Harmony Search Algorithm**”. Studies in Computational Intelligence 191, Springer. (pp. 1–14).

[75] Zhou A. and Jin Y. and Zhang Q. and Sendhoff and B. Tsang E. (2006), ”**Combining model-based and genetics-based offspring generation for multi-objective optimization using a convergence criterion**”. In 2006 IEEE International Conference on Evolutionary Computation. (pp. 892-899). IEEE.

واژه‌نامه فارسی به انگلیسی

ابرحجم ..... Hypervolume
اکتشاف ..... Exploration
استخراج ..... Explotitation
انتخاب ..... Selection
انحراف معیار ..... Standard deviation
انطباقی ..... Adaptive
اقلیدسی ..... Euclidean
بازنویسی ..... Rewrite
برازندگی ..... Fitness
بررسی ..... Considering
به‌روز رسانی ..... Updating
بهینه‌سازی ..... Optimization
پارتو ..... Pareto
پهنای باند ..... Band width
پیکربندی ..... Configuration
پویا ..... Dynamic
تکرار ..... Iteration
تقاطع ..... Crossover
تسلط ..... Dominance
تصادفی ..... Random
تنوع ..... Diversification
تنظیم ..... Setting
تشدید ..... Intensification
جهش ..... Mutation
چندهدفه ..... Multiobjective
خود انطباق ..... Self-Adaptive
سراسری ..... Global

فراابتکاری ..... Metaheuristic
فضا ..... Space
قطعه ..... Part
کارا ..... Efficient
کروموزوم ..... Chromosom
گستردگی ..... Extent
عملکرد ..... Performance
رتبه‌بندی ..... Ranking
گام ..... Pitch
مبادله کردن ..... Trade-off
مسئله ..... Problem
مرتب‌سازی ..... Sorting
نامقید ..... Unconstraint
نرخ ..... Rate
همگرایی ..... Convergence

واژه‌نامه انگلیسی به فارسی

ضخامت چسبندگی ..... Adhesive thickness
تنظیم ..... Adjustment
هوش مصنوعی ..... Artificial intelligence
تعادل ..... Balance
مقایسه ..... Comparison
تحدب ..... Convexity
طول برش از ناحیه متصل چسبیده ..... Cutting length of adhesive attached area
جاری ..... Current
متغیر تصمیم ..... Decision variable
گسسته ..... Discrete
پراکندگی ..... Dispersion
توزیع ..... Distribution
کارا ..... Efficient
تکاملی ..... Evolutionary
آزمایشگاهی ..... Experimental
شدنی ..... Feasible
مرز ..... Front
عضویت فازی ..... Fuzzy membership
مقداردهی اولیه ..... Initialization
بداهه‌گویی ..... Improvisation
حافظه ..... Memory
غیرمغلوب ..... Nondominated
مرز بهینه ..... Optimal front
تسلط پارتو ..... Pareto Dominance
تابع جریمه ..... Penalty Function
حد پایین ..... Lower limit
حد بالا ..... Upper limit

طول برجستگی یال ..... Length of flange fin
معیار ..... Test
گرمایشی ..... Thermal
ضخامت بال ..... Thickness of fin
کوتاه کردن ..... Truncating
یکنواخت ..... Uniform
دمای عملیات ..... Operation temperature
تخطی ..... Violation
روش مجموع وزین ..... Weighted sum method

Aabstract

Harmony Search (HS) algorithm is a new meta-heuristic method that inspired from musical process. This algorithm proposed as a metaheuristic method for solving single-objective problem. It became clear that can be used successfully for scientific issues. In this thesis, at first two Harmony Search proposals (MOHS1, MOHS2) are suggesed for solving multiobjective optimization problems. Subsequently, to prove the effectiveness of the proposed scheme, ZDT functions are used as test functions and the algorithm results are compared with results obtained with Nondominated Sorting Genetic Algorithm II (NSGA-II). Although harmony search algorithm for solving optimization problems shows many advantages, its parameters must be defined by users based on the experience and characteristics of the problem.This causes great difficulties for novice users. Subsequently, In order to overcome this difficulty, a Self-Adaptive MultiObjective Harmony Search (SAMOHS) algorithm based on harmony memory variance is proposed in this thesis. For solving multiobjective optimization problems, the proposed algorithm a nondominated sorting and truncating procedure are utilized to update the Harmony Memory. Subsequently, results the proposed SAMOHS algorithm are compared with other multi-objective evolutionary algorithms (SPEA2, MOPSO, NSGA-II), too are compared with two Harmony Search proposals.

keywords: Harmony search; Multi-objective optimization; Pareto dominance; Pareto Optimality; Self-adaptive parameter setting.

Shahrood University of Technology

Faculty of mathematical science

MSc Thesis in: Operations Research

Harmony Search Algorithm for solving Multi-Objective Optimization Problems

By: Zahra Pourgharavi

Supervisors

Dr. Mehrdad Ghaznavi
Dr. Maryam Ghorani

January 2017