فصل هفتم

استفاده از روش‌های فراابتکاری در بدست آوردن ورودی‌های کنترلی و منحنی مسیرهای بهینه[i]

۷-۱ معرفی الگوریتم‌های فراابتکاری

بهینه‌سازی در موارد بسیاری همچون مهندسی، فعالیت‌های اقتصادی و طراحی‌های صنعتی مهم می‌باشد. بدون هیچ اغراقی می‌توان ادعا کرد که بهینه‌سازی در هر جایی لازم و مفید است و این بدلیل محدود بودن منابع، مدت زمان اجرایی و بودجه در کاربردهای واقعی می‌باشد که ما را به نوعی مجبور می‌سازد تا برای استفاده بهینه از منابع ارزشمند خود تحت محدودیت‌های مختلف، راه حل‌هایی پیدا کنیم. لذا واضح است که هر چیزی همانند حداقل‌سازی مصرف سوخت و هزینه یا حداکثرسازی سود، تولید، عملکرد و کارآیی می‌تواند از کاربردهای بهینه‌سازی قلمداد گردد.

در واقع هدف مسئله بهینه‌سازی یا برنامه‌ریزی ریاضی، طراحی مسائل مربوطه با استفاده از ابزارهای ریاضی می‌باشد. از آنجایی که بسیاری از مسائل کاربردی دنیای واقعی، اغلب دارای غیرخطی‌گری بالایی می‌باشند، لذا نیازمند ابزارهای بهینه‌سازی پیچیده‌ای هستند. از این رو، امروزه شبیه سازی های کامپیوتری تبدیل به یک ابزار ضروری برای حل مسائل بهینه‌سازی با الگوریتم‌های جستجوی موثر و متفاوت شده‌اند.

همواره، پشت هرگونه شبیه‌سازی کامپیوتری و روش محاسباتی تعدادی الگوریتم در حال کار وجود دارد. مولفه‌های پایه‌ای هر الگوریتم و فعل و انفعال‌های داخلی آن، روش کاری الگوریتم و در نهایت تاثیر و کارآیی آن را تعیین می‌کنند.

بسیاری از الگوریتم‌های رایج یا کلاسیک، از نوع قطعی[۱] هستند، بعبارتی الگوریتم‌هایی می‌باشند که قادر به یافتن جواب بهینه به صورت دقیق هستند. بعنوان مثال روش سیمپلکس در برنامه‌ریزی خطی، الگوریتمی از نوع قطعی می‌باشد. برخی از الگوریتم‌های بهینه‌سازی قطعی از اطلاعات گرادیان (مقادیر تابع و مشتقات آن) استفاده می‌کنند که به الگوریتم‌های مبتنی بر گرادیان[۲] معروف هستند. بعبارتی، این الگوریتم‌ها روش‌های تکراری می‌باشند که بصورت گسترده از اطلاعات گرادیان تابع در تکرارها استفاده می‌کنند. بعنوان مثال الگوریتم سریع‌ترین سقوط که در فصل گذشته بررسی شد، یک الگوریتم مبتنی بر گرادیان می‌باشد که در مسائل تک وجهی نرم[۳] دارای عملکرد فوق العاده‌ای است، اما در صورت وجود ناپیوستگی در تابع هدف، دیگر عملکرد خوبی نخواهد داشت. لذا، در مواردی از این قبیل، الگوریتم‌های غیر گرادیانی[۴] ترجیح داده می‌شوند. این نوع الگوریتم‌ها از هیچ مشتقی استفاده نکرده و

---

[۱] Deterministic
[۲] Gradient-based
[۳] Smooth unimodal problems
[۴] Nongradient

فقط مبتنی بر مقادیر خود تابع هستند. بعنوان مثال جستجوی الگوی هوک-جیو[۱] از نوع الگوریتم‌های بدون گرادیان[۲] می‌باشد.

در حالت کلی، الگوریتم بهینه‌سازی، یک فرآیند تکراری است که از یک حدس اولیه شروع شده با تولید پاسخ‌های جدید و بهتر x^{t+1} برای مسئله مورد نظر، از پاسخ فعلی x^t در تکرار یا زمان t ادامه یافته و در نهایت و بعد از تعداد خاصی (به اندازه کافی بزرگ) تکرار، احتمالا به سمت پاسخی پایدار، در حالت ایده‌آل پاسخ بهینه مسئله مورد نظر، همگرا می‌گردد.

بعبارتی دیگر، الگوریتم بهینه‌سازی در ابتدا عامل‌های (پاسخ‌های احتمالی) خود را بطور کاملا تصادفی در فضای جستجو پخش می‌کند. سپس، با محاسبه تابع هزینه هرکدام، بهترین عامل (پاسخ) فعلی را مشخص می‌کند. در ادامه، با دو نوع جستجوی همه‌جانبه و محلی، به کاوش فضای جستجو می‌پردازد. در جستجوی همه‌جانبه، با استفاده از بهترین پاسخ موجود و تصادفی‌سازی، به تولید پاسخ‌های نسبتا هدف‌دار پرداخته و در جستجوی محلی به تولید پاسخ‌هایی جدید حول بهترین پاسخ موجود می‌پردازد. و این روند تا برقرار شدن شرایط اتمام الگوریتم ادامه می‌یابد.

در حالت کلی، برای الگوریتم‌های تصادفی[۳]، دو نوع ابتکاری[۴] و فراابتکاری[۵] با اندک تفاوتی وجود دارد. بطور ساده، لفظ ابتکاری به معنی پیدا کردن و یا یافتن با سعی و خطا[۶] می‌باشد. بعبارت دیگر، می‌توان جواب کیفی یک مسئله بهینه‌سازی پیچیده را در یک مدت زمان معقول پیدا کرد، اما هیچ تضمینی مبنی بر دست‌یابی به جواب بهینه وجود ندارد. لذا می‌توان انتظار داشت که این الگوریتم‌ها در بیشتر مواقع و نه در تمامی موارد، کارآیی داشته باشند و این امر زمانیکه بجای جواب بهینه بدنبال جواب قابل قبول با دسترسی آسان هستیم، خوب است. نوع پیشرفته‌تر الگوریتم‌های ابتکاری تحت عنوان الگوریتم-های فراابتکاری معروف هستد که در اینجا فرا به معنی ماورا یا سطح بالاتر می‌باشد. شایان ذکر است که تعاریف واحدی برای ابتکاری و فراابتکاری موجود نیست. در این راستا، برخی هر دو عبارت را بجای یکدیگر استفاده می‌کنند ولی با این وجود، گرایش عمومی تمایل به فراابتکاری نامیدن هر نوع الگوریتم تصادفی با تصادفی‌سازی[۷] و جستجوی محلی دارد.

در حالت کلی الگوریتم‌های فراابتکاری بهتر از الگوریتم‌های ابتکاری ساده عمل می‌کنند. چرا که، تمامی الگوریتم‌های فراابتکاری از مصالحه[۸] خاصی بین جستجوی محلی و تصادفی‌سازی استفاده می‌کنند. تصادفی‌سازی روش مناسبی برای گریز از جستجوی محلی[۹] به جستجوی همه‌جانبه[۱۰] بوده که این نیز نشانگر تمایل الگوریتم‌های فراابتکاری به بهینه‌سازی همه‌جانبه است.

در واقع الگوریتم‌های فراابتکاری، یکی از انواع الگوریتم‌های بهینه‌سازی تقریبی بوسیله سعی و خطا هستند که دارای راهکارهای برون رفت از بهینه محلی بوده و قابل استعمال در طیف گسترده‌ای از مسائل می‌باشند که جواب‌های قابل قبولی را در مدت زمان معقولی فراهم می‌کنند. اغلب، پیچیدگی

---

[۱] Hooke-Jeeves
[۲] Gradient-free
[۳] Stochastic
[۴] Heuristic
[۵] Meta-heuristic
[۶] Trial and Error
[۷] Randomization
[۸] Trade off
[۹] Locally Search
[۱۰] Globally Search

مسائل مورد نظر امکان جستجوی پاسخ ها یا ترکیب های ممکن را سلب می‌کند. چاره کار پیدا کردن جواب‌های عملی و خوب در بازه زمانی قابل قبول می‌باشد. این در حالی است که هیچ تضمینی برای پیدا کردن جواب بهینه همه‌جانبه وجود ندارد و حتی نمی‌دانیم یک الگوریتم چگونه عمل خواهد کرد و اصلا در صورت عمل کردن، چرا عمل می‌کند. ایده مورد نظر، داشتن الگوریتمی عملی و موثر است که در بیشتر اوقات عمل کرده و قادر به تولید جواب‌های با کیفیت قابل قبول باشد. انتظار داریم تعدادی از جواب‌های کیفی پیدا شده تقریبا بهینه باشند، هر چند که هیچ تضمینی برای اینگونه بهینگی نیز وجود ندارد.

دو مولفه مهم هر الگوریتم فراابتکاری تنوع[۱]و تمرکز[۲]، یا پایش[۳] و استخراج[۴] می‌باشند. تنوع به معنی تولید جواب‌های گوناگون و به اندازه کافی دور از پاسخ‌های فعلی، جهت پایش فضای جستجو در مقیاس همه‌جانبه است. تمرکز نیز به معنی استفاده از اطلاعات بدست آمده از بهترین جواب فعلی جهت تشدید جستجوی محلی در آن منطقه است.

مزیت تنوع یا پایش، جلوگیری از افتادن در بهینه محلی با استفاده از تصادفی‌سازی است و بعبارتی بهینه همه‌جانبه را قابل دسترس‌تر می‌کند، ولی باعث کاهش نرخ همگرایی و اتلاف تلاش‌های محاسباتی می‌گردد که احتمالا بدلیل دوری پاسخ‌های جدید از بهینه همه‌جانبه می‌باشد. مزیت تمرکز یا استخراج نیز افزایش نرخ همگرایی است، اما این اشکال را دارد که احتمال افتادن در بهینه محلی را افزایش می‌دهد، چرا که پاسخ نهایی به مقدار زیادی به نقطه شروع بستگی دارد.

در حالت کلی، استخراج خیلی زیاد و پایش خیلی کم به معنی سرعت همگرایی بیشتر الگوریتم با احتمال پیدا کردن بهینه همه‌جانبه پایین است. از طرفی دیگر، استخراج خیلی کم و پایش خیلی زیاد ممکن است باعث سرگردانی مسیر جستجو شده و در نتیجه سرعت همگرایی کاهش یابد. لذا، جهت دست‌یابی به عملکرد هرچه بهتر الگوریتم، یک تعادل بهینه بین دو مولفه مذکور لازم می‌باشد. تعادل بهینه نیز به معنی مقدار درستی از پایش و استخراج است که باعث بهینه شدن عملکرد یک الگوریتم می‌گردد.

به هر حال، پیدا کردن راهی برای بدست آوردن چنین تعادلی هنوز مسئله حل نشده‌ای است. در حقیقت هیچ یک از الگوریتم های موجود نمی‌توانند مدعی دست یابی به چنین تعادلی باشند. بعبارتی تعادل بهینه خودش نوعی مسئله فوق بهینه‌سازی[۵] به معنی بهینه‌سازی یک الگوریتم بهینه‌سازی می‌باشد. اینگونه تعادل ممکن است به چندین عامل مانند مکانیزم کاری الگوریتم، مجموعه پارامترهای آن، تنظیم[۶] و کنترل این پارامترها و حتی به مسئله مورد نظر وابسته باشد. علاوه بر این، احتمالا، اینگونه تعادل بطور کلی وجود نداشته باشد یا از مسئله‌ای به مسئله دیگر تغییر کند.

---

[۱] Diversification
[۲] Intensification
[۳] Exploration
[۴] Exploitation
[۵] Hyper optimization
[۶] Tuning

۲-۷ طبقه‌بندی الگوریتم‌های فراابتکاری

در برنامه‌ریزی ریاضی، یک مساله آسان یا مهار شدنی[۱]، مساله‌ای است که راه‌حل آن را بتوان بوسیله یک الگوریتم کامپیوتری، با یک زمان حل (یا تعداد مراحل) بصورت تابعی چند جمله‌ای از مرتبه مسئله بدست آورد. یک الگوریتم، مساله-P[۲] یا مسئله چند جمله‌ای-زمان[۳] نامیده می‌شود هرگاه، شمار مراحل مورد نیاز جهت یافتن پاسخ، بوسیله یک چند جمله‌ای مرتبه n نشان داده شود و حداقل یک الگوریتم برای حل داشته باشد.
از طرفی دیگر، یک مساله سخت یا مهار نشدنی[۴] مساله‌ای است که زمان حل آن بصورت تابع نمایی مرتبه n باشد. اگر راه حل یک مساله چند جمله‌ای، توسط زمان آن تخمین و ارزیابی شود، چند جمله‌ای غیرقطعی (NP)[۵] نامیده می‌شود اما باید توجه داشت که الزاما نمی‌توان هیچ قانون مشخصی برای تخمین زدن راه حل در نظر گرفت. در نتیجه پاسخ تخمینی نمی‌تواند تضمینی برای بهینگی یا حتی نزدیک به بهینگی را داشته باشد. در واقعیت الگوریتم مشخصی برای حل مسائل NP-سخت وجود ندارد و فقط راه حل‌های تقریبی یا ابتکاری امکان‌پذیر می‌باشند.
در حالت کلی، حل مسائل بهینه‌سازی دنیای واقعی بسیار چالش برانگیز بوده و در بسیاری از موارد مجبور به مواجهه با مسائل NP-سخت می‌باشند. بنابراین روش‌های ابتکاری و فراابتکاری در بدست آوردن پاسخ‌های تقریبی، نزدیک به بهینه یا در حد مطلوب، بسیار رضایت‌بخش می‌باشند. در این راستا الگوریتم‌های جدید و متنوعی ارائه شده‌اند. در بین این الگوریتم‌ها، بسیاری همچون الگوریتم بهینه‌سازی ازدحام ذرات[۶]، فاخته[۷] و کرم شتاب[۸] شهرت فراوانی را بخاطر کارآیی‌شان کسب کرده‌اند.

الگوریتم‌های فراابتکاری را می‌توان به طرق مختلفی تقسیم بندی کرد ولی با توجه به کثرت و تنوع این الگوریتم‌ها، انجام سیستماتیک این عمل، کاری چالش برانگیز خواهد بود. مشخصا، طبقه‌بندی این الگوریتم‌ها به معیار و ملاک طبقه‌بندی وابسته است و طبیعتا با تغییر آن، این طبقه‌بندی نیز دستخوش تغییر خواهد شد. به هر حال، ما در این کتاب روی ویژگی منبع الهام این الگوریتم‌ها تمرکز کرده و آن را مبنا و معیار طبقه‌بندی خود قرار داده‌ایم[۱].

از بسیاری جهات، همیشه طبیعت الهام بخش محققین بوده و هست. امروزه نیز بسیاری از الگوریتم‌های جدید الهام گرفته شده از طبیعت[۹] هستند. لذا ضمن تاکید روی منبع الهام و وابسته به جزئیات و زیرمنابعی که در نظر گرفته می‌شود، می‌توان سطوح مختلفی از طبقه‌بندی را ارائه نمود. در اینجا نیز بدلیل سادگی، از منابع سطح بالایی چون زیست، فیزیک/شیمی استفاده خواهد شد.

بعبارت کلی، منبع الهام اصلی طبیعت می‌باشد. فلذا، تقریبا تمام الگوریتم‌ها را می‌توان بعنوان برگرفته شده از طبیعت در نظر گرفت. از طرفی دیگر، اکثر الگوریتم‌های برگرفته شده از طبیعت براساس

---

[۱] Tractable
[۲] P-problem
[۳] Polynomial-time
[۴] Intractable problem
[۵] Nondeterministic Polynomial
[۶] Particle Swarm Optimization (PSO)
[۷] Cuckoo Search (CS)
[۸] Firefly Algorithm (FA)
[۹] Nature-Inspired (NI)

مشخصات موفقیت‌آمیز سیستم زیستی شناختی می‌باشند. لذا، بزرگترین شاخه الگوریتم‌های برگرفته شده از طبیعت، برگرفته شده از زیست[۱] می‌باشد. در بین الگوریتم‌های برگرفته شده از زیست نیز نوع خاصی از الگوریتم‌ها وجود دارند که از هوش جمعی الهام گرفته شده‌اند و می‌توان آنها را الگوریتم‌های بر مبنای هوش جمعی[۲] نامید. این نوع الگوریتم‌ها نیز جز الگوریتم‌های پر کاربرد می‌باشند، بعنوان مثال: بهینه‌سازی ازدحام ذرات، جستجوی فاخته، الگوریتم خفاش[۳] و الگوریتم کرم شب تاب.

مشخصات، تمامی الگوریتم‌های فراابتکاری براساس سیستم‌های زیست شناختی نمی‌باشند. چراکه، خیلی از الگوریتم‌های برگرفته شده از طبیعت، با الهام از سیستم‌های فیزیکی/شیمی ایجاد شده‌اند. حتی ممکن است برخی براساس موسیقی ایجاد شده باشند.

براساس مباحث فوق، تمامی الگوریتم‌ها را می‌توان به چهار قسمت کلی تقسیم بندی کرد: بر پایه هوش جمعی (SI)، برگرفته شده از زیست شناسی (غیر SI)، بر پایه فیزیک/شیمی و سایر الگوریتم‌ها. در ادامه بطور مختصر به توضیح هر یک از این قسمت‌ها پرداخته و تنها روی الگوریتم‌های مهم و جدید تمرکز خواهد شد.

شایان ذکر است طبقه‌بندی ارائه شده یکتا نیست و این بدلیل وجود الگوریتم‌هایی است که همزمان می‌توانند در طبقه‌بندی‌های مختلفی قرار بگیرند. بعبارتی ساده‌تر، عمل طبقه‌بندی خیلی به نقطه تاکید و دیدگاه شخص تقسیم کننده وابسته می‌باشد. بعنوان مثال، اگر تاکید روی مسیر جستجو باشد، در این صورت الگوریتم‌ها را می‌توان به دو کلاس بر پایه مسیر[۴] و بر پایه جمعیت[۵] تقسیم کرد. در این راستا، تبرید شبیه سازی[۶] شده مثال خوبی برای الگوریتم‌های بر پایه مسیر است در حالی که بهینه‌سازی ازدحام ذرات و کرم شب تاب نمونه خوبی از الگوریتم‌های بر پایه جمعیت هستند. در حالیکه، اگر تاکید روی برهم کنش عامل‌ها[۷] باشد، الگوریتم‌ها را می‌توان به دو کلاس بر پایه جذابیت[۸] و بر پایه غیر جذابیت[۹] تقسیم‌بندی کرد. کرم شب تاب مثال خوبی برای کلاس الگوریتم‌های بر پایه جذابیت است چراکه، این الگوریتم از جاذبه نور و جذابیت شب تاب‌ها استفاده می‌کند در حالیکه الگوریتم ژنتیک در کلاس غیر جذاب‌ها قرار می‌گیرد، چراکه که هیچ گونه جذابیت آشکاری در آن استفاده نشده است. از طرفی دیگر، اگر تاکید روی معادلات به روز کننده الگوریتم‌ها باشد، طبقه بندی به دو کلاس بر پایه معادله[۱۰] و بر پایه قانون[۱۱] می‌انجامد. بعنوان مثال، بهینه‌سازی ازدحام ذرات، الگوریتم خفاش و جستجوی فاخته بخاطر استفاده روشن از معادلات به روز کننده، نمونه‌های خوبی برای کلاس بر پایه معادله هستند، درحالیکه، الگوریتم ژنتیک معادلات مشخصی برای جهش[۱۲] و تقاطع[۱۳] ندارد. هرچند که این

---

[۱] Biology-Inspired (BI)
[۲] Swarm-Intelligence (SI)
[۳] Bat Algorithm-BA
[۴] Trajectory-Based
[۵] Population-Based
[۶] Simulated Annealing-SA
[۷] Agents
[۸] Attraction-Based
[۹] Non-Attraction-Based
[۱۰] Equation-Based
[۱۱] Rule-Based
[۱۲] Mutation
[۱۳] Crossover

طبقه‌بندی یکتا نیست و بعنوان مثال، کرم شب تاب از سه قانون روشن استفاده می‌کند که قابلیت تبدیل به یک معادله غیرخطی واحد را دارند. همانطور که پیش‌تر نیز بیان شد، عمل طبقه بندی به انگیزه و دیدگاه شخص طبقه‌بندی کننده بستگی دارد و ما نیز با تاکید روی منبع الهام الگوریتم‌ها، طبقه‌بندی زیر را ارائه کرده‌ایم.

۱-۲-۷ الگوریتم‌های بر پایه هوش جمعی (SI)

هوش جمعی (SI) بر رفتار جمعی چندعامل دارای برهم کنش تاکید دارد که از چندین قانون ساده پیروی می‌کنند. با آنکه هر عامل را به تنهایی می‌توان بدون هوش در نظر گرفت، ولی ممکن است کل سیستم چندعاملی، رفتار خود سازماندهی نشان دهد که به نوعی بیانگر رفتار هوش جمعی است. بسیاری از الگوریتم‌های ارائه شده از همین سیستم‌های هوش جمعی موجود در طبیعت نشات گرفته‌اند.

تمام الگوریتم‌های SI، از خاصیت چندعاملی برگرفته شده از رفتار گروهی حشراتی چون مورچه‌ها و زنبورها و نیز حیواناتی چون دسته پرندگان و ماهی‌ها است. بعنوان مثال، الگوریتم بهینه‌سازی انبوه ذرات از رفتار گروهی ماهی‌ها و پرندگان استفاده می‌کند در حالیکه الگوریتم کرم شب تاب از رفتار تابشی کرم شب تاب‌های جمعی نشات گرفته شده است. جستجوی فاخته بر مبنای خاصیت پارازیت اندازی جوجه برخی از گونه‌های فاخته بوده و الگوریتم خفاش براساس استفاده از سونار ریزخفاش‌ها هنگام شکار است. بهینه‌سازی کلونی مورچگان از برهم کنش حشرات گروهی چون مورچه‌ها استفاده می‌کند در صورتیکه کلاس خاصی از الگوریتم‌های زنبور عسل بر پایه رفتار غذایابی زنبورهای عسل می‌باشد.

به دلایل زیادی، الگوریتم‌های SI جزو الگوریتم‌های شناخته شده و پر کاربرد می‌باشند. یکی از دلایل آن، این است که معمولا الگویتم‌های SI اطلاعات را بین عامل‌ها پخش می‌کنند فلذا خود سازماندهی[۱]، هم‌تکاملی[۲] و یادگیری[۳] در طی تکرارها باعث کارآیی بالای این الگوریتم‌ها می‌گردد. دلیل دیگر نیز در موازی‌سازی[۴] ساده چندین عامل برای عملی کردن هرچه بیشتر بهینه‌سازی‌های ابعاد وسیع از دیدگاه اجرایی است.

۲-۲-۷الگوریتم‌های الهام گرفته شده از زیست شناختی (غیر SI)

الگوریتم‌های هوش جمعی SI متعلق به کلاس بزرگتری از الگوریتم‌ها هستند که BI یا برگرفته شده از زیست شناختی نامیده می‌شوند. بعبارتی می‌توان گفت که، برخی از الگوریتم‌های BI مستقیما از خواص رفتار جمعی استفاده نمی‌کنند و به این علت آنها را غیر SI می‌نامند. بعنوان مثال الگوریتم ژنتیک جز الگوریتم‌های BI بوده ولی جز الگوریتم SI نمی‌باشد. در حقیقت، الگوریتم‌های BI اکثر الگوریتم‌های برگرفته شده از طبیعت را شامل می‌شوند. لذا طبق دیدگاه تئوری مجموعه‌ها می‌توان

---

[۱] Self-Organization

[۲] Co-Evolution

[۳] Learning

[۴] Parallelized

گفت که الگوریتم‌های SI زیرمجموعه الگوریتم‌های BI بوده و خود این الگوریتم‌ها نیز زیرمجموعه الگوریتم‌های NI می‌باشند.

۷-۲-۳ الگوریتم‌های بر پایه فیزیک/شیمی

الگوریتم‌هایی هستند که زیرمجموعه الگوریتم‌های NI بوده ولی زیرمجموعه الگوریتم‌های BI نمی-باشند. بعبارتی منبع الهام آنها بر پایه تقلید از برخی خواص فیزیکی/شیمیایی طبیعت همانند بارهای الکتریکی، جاذبه زمین و سیستم رودخانه‌ها می‌باشد. از آنجایی که سیستم‌های طبیعی مختلفی به این کلاس تعلق دارند، می‌توان آن را به زیر شاخه‌های دیگری نیز تقسیم کرد که در اینجا ضرورت چندانی ندارد.

۷-۲-۴ سایر الگوریتم‌ها

منبع الهام برخی از الگوریتم‌ها نیز از طبیعت گرفته نشده است و از منابع دیگری چون جامعه، عواطف و غیره بر گرفته شده است. الگوریتم رقابت استعماری یکی از این الگوریتم‌ها است.

جدول ۷-۱ لیستی از الگوریتم‌های فراابتکاری [۱]

سایر	بر پایه فیزیک/شیمی	زیست شناختی-غیر SI	هوش جمعی-SI
الگوریتم رقابت استعماری	تبرید شبیه‌سازی شده	الگوریتم جهش ترکیبی قورباغه	الگوریتم خفاش
الگوریتم قهرمانی لیگ	جستجوی هارمونی	بهینه‌سازی زیست جغرافی	بهینه‌سازی کلونی مورچگان
بهینه‌سازی احساسی جمعی	چکه آب‌های هوشمند	تکامل تفاضلی	بهینه‌سازی ازدحام ذرات
تکامل دستوری	سیاه چاله	مکان‌یابی دلفین	غذایابی باکتری
الگوریتم جستجوی تکاملی	انفجار بزرگ، انقباض بزرگ	بهینه‌سازی فکر بکر	الگوریتم زنبورهای عسل
بهینه‌سازی ردگیری معکوس	جستجوی کهکشان	الگوریتم برگرفته از انسان	جستجوی فاخته
جستجوی تعاونی مصنوعی	جستجوی گرانشی	الگوریتم گرده افشانی گل	الگوریتم کرم شب‌تاب
بهینه‌سازی جامعه آشفته	دینامیک شکل‌گیری رودخانه	مدل ابرهای جو	ازدحام ماهی
	جستجوی تخلیه تصادفی	تکامل زنبور ملکه	ازدحام گربه
	بهینه‌سازی نیروی مرکزی	جستجوی دسته ماهی‌ها	جستجوی گرگ
	بهینه‌سازی مارپیچی	حرکت قزل‌آلای بزرگ	جستجوی میمون

در ادامه این فصل، پنج الگوریتم شناخته شده از طبقات مختلف جدول فوق و نیز الگوریتم ژنتیک معرفی شده و برای تعیین ورودی‌های کنترلی مدار باز و منحنی مسیرهای بهینه مسئله ۲-۲-۶، بعنوان

مسئله مرجع، مورد بحث و بررسی قرار خواهند گرفت. این الگوریتم‌ها به ترتیب عبارتند از: الگوریتم خفاش و نسخه جدید آن، الگوریتم جهش ترکیبی قورباغه، الگوریتم تبرید شبیه‌سازی شده، الگوریتم رقابت استعماری، الگوریتم بهینه‌سازی ازدحام ذرات و الگوریتم ژنتیک.

۷-۳ الگوریتم خفاش [۲]

در این بخش به دنبال معرفی الگوریتم فراابتکاری جدیدی تحت عنوان الگوریتم خفاش[۱] (BA) هستیم که بر اساس ویژگی مکان‌یابی اکو[۲] خفاش‌ها می‌باشد. این الگوریتم از روش تنظیم فرکانسی برای افزایش تنوع راه حل‌ها در جمعیت استفاده می‌کند. از طرفی دیگر، این الگوریتم از زوم کردن خودکار[۳] ناشی از تقلید تغییرات نرخ انتشار پالس و بلندی صدای خفاش‌ها هنگام شکار، برای متعادل‌سازی پایش و استخراج طی فرآیند جستجو استفاده می‌کند.
توانایی مکان‌یابی اکو ریز خفاش‌ها سحرآمیز است. این خفاش‌ها می‌توانند در تاریکی کامل طعمه هایشان را پیدا کرده و انواع مختلف حشرات را تشخیص دهند. در ابتدا الگوریتم خفاش را از طریق ساده‌سازی رفتار مکان‌یابی اکو خفاش‌ها فرمول‌بندی کرده و سپس روال کار آن توضیح داده خواهد شد.

۷-۳-۱ مکان‌یابی اکو خفاش‌ها

۷-۳-۱-۱ رفتار ریزخفاش‌ها

خفاش‌ها حیوانات شگفت‌انگیزی هستند. آنها تنها پستانداران دارای بال بوده و همچنین توانایی بی-نظیری به نام مکان‌یابی اکو دارند. تخمین زده می‌شود که حدودا هزار گونه‌ی متفاوت از این حیوان وجود دارد که بیش از ٪۲۰ همه انواع پستانداران بشمار می‌آید. حدود اندازه‌ی آنها از ریزخفاش (حدود ۱.۵ تا ۲ گرم) تا خفاش‌های غول‌پیکر با طول بال‌های حدود ۲ متر و وزنی بیش از حدود Kg ۱ است. اکثر خفاش‌ها از مکان‌یابی اکو به یک درجه معینی بهره می‌برند بعبارتی دیگر، از این ویژگی در همه جا استفاده نمی‌کنند. در این بین، ریزخفاش‌ها گونه متفاوتی از خفاش‌ها هستند که از خاصیت مکان‌یابی اکو خود در همه جا استفاده می‌کنند.
ریزخفاش‌ها از یک نوع سونار[۴] (ردیاب صوتی) به نام مکان‌یابی اکو برای پیدا کردن شکار، ممانعت از برخورد با موانع و تعیین شکاف‌های نشیمن‌گاه خود در تاریکی استفاده می‌کنند. این خفاش‌ها یک پالس صوتی بسیار بلند منتشر کرده و به پژواک برگشتی از اشیای مجاور گوش می‌دهند. ویژگی‌های پالس‌های آنها با هم متفاوت می‌باشند بطوریکه می‌تواند به استراتژی‌های شکار گونه‌های مختلف آنها مرتبط باشد. بیشتر خفاش‌ها از سیگنالهای کوتاه و مدوله فرکانسی در حدود یک اکتاو[۵] برای تجسس منطقه استفاده می‌کنند، در حالی که بقیه آنها از سیگنالهای فرکانس ثابت برای مکان‌یابی اکو استفاده می‌کنند. تغییرات پهنای باند سیگنالهای این حیوانات به گونه‌های آنها وابسته است و اغلب با استفاده از هارمونیک‌های بیشتر، افزایش می‌یابد.

---

[۱] Bat algorithm
[۲] Echolocation
[۳] Automatic zooming
[۴] Sonar
[۵] Octave

۲-۱-۳-۷ آواشناسی مکان‌یابی اکو

اگرچه دوام هر پالس خفاش در حدود چندین هزارم ثانیه است (حدود ۸ تا ۱۰ میلی ثانیه)، اما یک فرکانس ثابت در ناحیه‌ی ۲۵ KHz تا ۱۵۰ KHz دارد. علیرغم اینکه برخی از گونه‌ها می‌توانند فرکانس‌های بالاتر از ۱۵۰ KHz را ساطع کنند، دامنه فرکانسی بیشتر گونه‌ها در ناحیه بین ۲۵ KHz تا ۱۰۰ KHz می‌باشد. هر انفجار فراصوتی حدودا ۵ تا ۲۰ میلی ثانیه طول می‌کشد و ریز خفاش‌ها در هر ثانیه حدود ۱۰ تا ۲۰ صدا مانند انفجار صوتی را منتشر می‌کنند. به هنگام شکار طعمه و در نزدیکی صید، سرعت انتشار پالس می‌تواند تا حدود ۲۰۰ پالس بر ثانیه افزایش یابد. این چنین انفجار صوتی کوتاه نشانگر توانایی خارق‌العاده پردازش سیگنال خفاش‌ها می‌باشد. در واقع، مطالعات نشان می‌دهد که زمان مجتمع‌سازی گوش خفاش در حدود ۳۰۰ تا ۴۰۰ میکروثانیه است.

از آنجا که سرعت صوت در هوا حدودا $v=$ ۳۴۰m/s است، طول موج λ انفجار صدای فراصوتی با فرکانس ثابت f از رابطه زیر به دست می‌آید.

$$\lambda = \frac{\upsilon}{f} \qquad (۷.۳-۱)$$

که برای بازه فرکانسی نمونه ۲۵ KHz تا ۱۵۰ KHz، در بازه ۲mm تا ۱۴mm است. طول موج این چنینی مرتبط با اندازه شکار می‌باشد.

پالس ساطع شده بطور شگفت‌انگیزی می‌تواند به بلندی ۱۱۰ db باشد که خوشبختانه در ناحیه‌ی فراصوت می‌باشد. بلندی صدا از بلندترین حالت در هنگام جستجوی طعمه تا مقداری آرام‌تر به هنگام نزدیک شدن به طعمه، تغییر می‌کند. بر مبنای فرکانس‌های حقیقی، محدوده حرکت این پالس‌های کوتاه بطور معمول چند متر است. بنابراین ریزخفاش‌ها می‌توانند مسیر خود را برای عدم برخورد با موانعی به کوچکی قطر موی انسان نیز کنترل کنند.

مطالعات نشان می‌دهد که ریزخفاش‌ها از تاخیر زمانی انتشار و ردیابی اکو، اختلاف زمانی میان دو گوششان و تغییرات بلندی اکوها برای بالا بردن توان ایجاد سناریوی سه بعدی محیط، استفاده می‌کنند. آنها می‌توانند فاصله و جهت هدف، نوع شکار و حتی سرعت حرکت شکاری به کوچکی یک حشره را تعیین کنند. در واقع مطالعات بیانگر آن هستند که خفاش‌ها با استفاده از تغییرات اثر دوپلر بوجود آمده از سرعت بال زدن حشرات، قادر به تمییز اهداف خود می‌باشند.

این رفتار مکان‌یابی اکو ریزخفاش‌ها می‌تواند به نحوی فرموله شود که در صورت اعمال به تابع هدف، آن را بهینه کند و این امر فرمول‌بندی یک الگوریتم بهینه‌سازی جدید را امکان پذیر می‌سازد. در ادامه، نمای اجمالی فرموله کردن الگوریتم خفاش (BA) را بیان کرده و سپس به پیاده‌سازی آن پرداخته خواهد شد.

۲-۳-۷ ساختار الگوریتم خفاش

اگر برخی از مشخصات مکان‌یابی اکو ریز خفاش‌ها ساده‌سازی شود، می‌توان الگوریتم‌های گوناگون الهام گرفته شده از خفاش را توسعه داد. در این راستا، قوانین تقریبی و ایده‌آل‌سازی شده‌ی زیر به کار برده شده اند:

۱. همه‌ی خفاش‌ها برای تشخیص فاصله و تمییز تفاوت بین غذا و موانع از مکان‌یابی اکو استفاده می‌کنند.
۲. خفاش‌ها بطور تصادفی با سرعت v_i، فرکانس ثابت f_{min} ،طول موج متغیر λ و بلندی صدای A_i در مکان x_i در جستجوی شکار پرواز می‌کنند. همچنین بر مبنای میزان نزدیکی به اهداف خود، می‌توانند به طور خودکار، طول موج (یا فرکانس) پالس‌های منتشر شده و سرعت انتشارشان $r\epsilon[۰و۱]$ را تنظیم نمایند.
۳. اگر چه بلندی صدا از راه‌های زیادی می‌تواند تغییر کند اما فرض می‌کنیم که بلندی صدا از یک مقدار بزرگ (مثبت) A_0 به یک مقدار ثابت مینیمم A_{min} تغییر می‌کند.

ساده‌سازی دیگر این است که برای تخمین تاخیر زمانی و توپولوژی سه بعدی از ردیابی اشعه استفاده نمی‌شود، اگرچه ممکن است این خصیصه برای هندسه محاسباتی[۱] مناسب باشد، اما ما از آن استفاده نمی‌کنیم زیرا در موارد چند بعدی دارای محاسبات بسیار گسترده‌ای است. علاوه بر این فرضیات ساده کننده، از تقریب‌های زیر نیز برای سادگی بهره می‌بریم.

در حالت کلی فرکانس fدر محدوده $[f_{min}$ و $f_{max}]$ منطبق بر محدوده طول موج $[\lambda_{min}, \lambda_{max}]$ است. برای مثال محدوده فرکانسی KHz [۲۰ , ۵۰۰] با محدوده طول موجی mm [۰.۷ , ۱۷] مطابق است. برای یک مساله مشخص می‌توانیم از هر طول موجی برای سهولت کار استفاده کنیم. در پیاده سازی واقعی می‌توانیم محدوده را از طریق تنظیم طول موج‌ها (یا فرکانس‌ها) تنظیم کنیم و محدوده قابل پایش (یا بزرگترین طول موج) باید متناظر با اندازه دامنه دلخواه انتخاب شود و سپس درجه صدا تا محدوده‌های کوچک‌تر کاهش یابد.

علاوه بر این ما مجبور به استفاده از خود طول موج‌ها نیستیم و در عوض می‌توانیم فرکانس را در حالی که طول موج λ ثابت است، تغییر دهیم.

برای ساده سازی فرض می کنیم که $f \in[۰, f_{max}]$ و می‌دانیم فرکانس‌های بالاتر، طول موج‌های کوتاه تری دارند و در نتیجه فاصله کمتری را می‌پیمایند. محدوده معمول برای خفاش‌ها در حدود چند متر می‌باشد. سرعت انتشار پالس به سادگی می‌تواند در محدوده $[۰و۱]$ قرار بگیرد، بطوریکه صفر به معنای این است که هیچ پالسی وجود ندارد و یک به معنای بیشترین سرعت انتشار پالس است.

بر اساس این تقریب‌ها و ساده‌سازی‌ها، مراحل اساسی الگوریتم خفاش می‌تواند به صورت فلوچارت زیر خلاصه شود. واضح است که الگوریتم خفاش با مقدار دهی اولیه تعداد خفاش‌ها، موقعیت هر یک از آنها (جواب اولیه)، سرعت انتشار پالس، بلندی صدا و فرکانس شکار، شروع می‌شود. در هر تکرار، هر یک از خفاش‌ها از جواب اولیه خود به سمت بهترین جواب همه‌جانبه پرواز می‌کنند. اگر هرکدام از خفاش‌ها جواب بهتری پیدا کنند، سرعت انتشار پالس و بلندی صدا به روز می‌گردد. در طی تکرار پرواز نیز بهترین جواب موجود به روز می‌گردد. این فرآیند تا برآورده شدن شرط اتمام الگوریتم، بطور پیوسته‌ای تکرار می‌شود و در نهایت بهترین جواب بدست آمده بعنوان بهترین جواب نهایی منظور می‌گردد.

---

[۱] Computational geometry

۱-۲-۳-۷ حرکت خفاش‌های مجازی[۱]

طبیعتا در شبیه‌سازی‌ها از خفاش‌های مجازی استفاده می‌کنیم. بنابر این باید قوانینی را برای چگونگی به روز کردن موقعیت x_i و سرعت v_i در فضای جست و جوی d-بعدی تعریف کنیم. پاسخ جدید ${x_i}^t$ و سرعت جدید ${v_i}^t$ در مرحله زمانی t *و برای یک تکرار* با روابط زیر مشخص می‌شوند:

$$f_i = f_{\min} + (f_{\max} - f_{\min})\beta$$
$$v_i^t = v_i^{t-1} + (x_i^{t-1} - x^*) f_i \qquad (۷.۳-۲)$$
$$x_i^t = x_i^{t-1} + v_i^t.1$$

که β یک بردار تصادفی متشکل از درایه‌هایی با توزیع یکنواخت در بازه [۱و۰] است. در اینجا x^* بهترین موقعیت (پاسخ) مطلق جاری بوده که بعد از مقایسه همه پاسخ‌ها در میان همه n خفاش واقع می‌شود.

از آنجائیکه حاصل ضرب $\lambda_i f_i$ نمو سرعت است، بنابراین بر اساس نوع مسئله، می‌توان برای تنظیم تغییرات سرعت از f_i (یاλ_i) حین ثابت بودن عامل دیگر λ_i (یا f_i) استفاده کرد. ما نیز در پیاده‌سازی خود بر اساس دامنه مساله مورد علاقه، از $f_{min} = ۰$ و $f_{max} = ۱۰۰$ استفاده خواهیم کرد. در ابتدا، به هر خفاش یک فرکانس تصادفی با توزیع یکنواخت در فاصله $[f_{min}$ و$f_{max}]$ اختصاص داده می‌شود. به این دلیل است که می‌توان الگوریتم خفاش را بعنوان الگوریتم تنظیم فرکانسی جهت تامین ترکیب متعادلی از پایش و استخراج، بحساب آورد. اساسا، بلندی صدا و سرعت انتشار پالس مکانیزمی برای کنترل و زوم کردن خودکار درون ناحیه پاسخ‌های محتمل، فراهم می‌کند.

برای قسمت جستجوی محلی، یک بار یک پاسخ از بین بهترین پاسخ‌های موجود انتخاب می‌شود و سپس پاسخ جدید برای هر خفاش با استفاده از گام تصادفی به صورت محلی تولید می‌شود:

$$x_{new} = x_{old} + \epsilon A^t \qquad (۷.۳-۳)$$

که$\epsilon \in [-۱,۱]$ یک عدد تصادفی است، در حالیکه $A^t = < A_i^t >$ متوسط بلندی صدا برای همه خفاش‌ها در این مرحله زمانی می‌باشد.

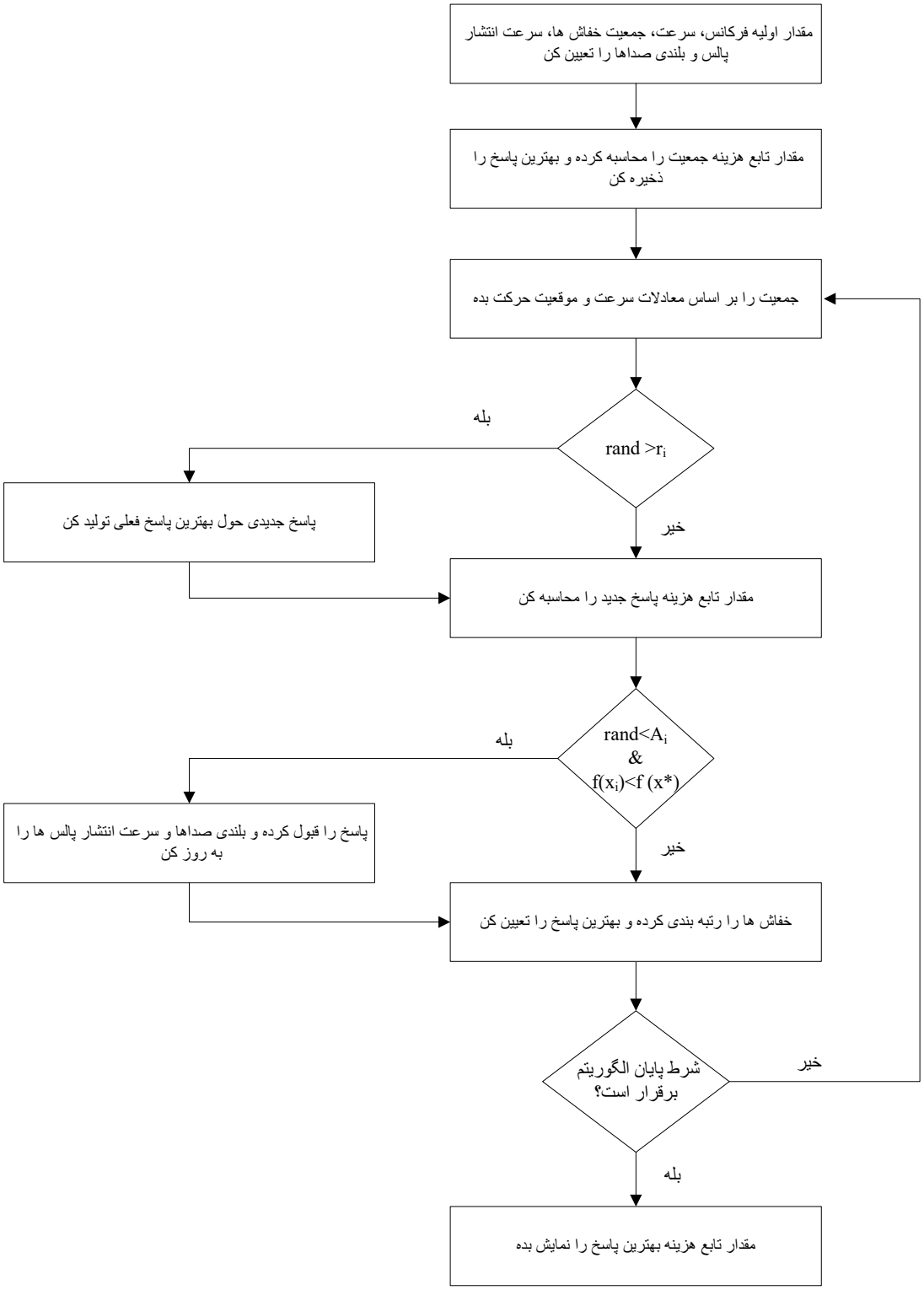
به روز رسانی سرعت و موقعیت خفاش‌ها اندکی شبیه روش استاندارد بهینه‌سازی ازدحام ذرات است که f_i سرعت و محدوده حرکت ذرات را کنترل می‌کند. تا حدودی، BA می‌تواند بعنوان ترکیب متعادلی از بهینه‌سازی استاندارد ازدحام ذرات و جست و جوی محلی متمرکز در نظر گرفته شود که با بلندی صدا و سرعت انتشار پالس کنترل می‌شود.

۲-۲-۳-۷ بلندی صدا[۲] و سرعت انتشار پالس[۳]

همچنانکه تکرارها به پیش می‌روند، به جهت فراهم آوردن مکانیزمی موثر برای کنترل پایش و استخراج و گذر از مرحله پایش به مرحله استخراج، بلندی صدا A_i و سرعت انتشار پالس r_i باید به روز شوند. بدلیل کاهش رایج بلندی صدا حین پیدا کردن شکار توسط خفاش، ضمن افزایش سرعت انتشار پالس،

---

[۱] Virtual [۲] Loudness [۳] Pulse emission rate



شکل ۱-۷ نمای الگوریتم خفاش

بلندی صدا می‌تواند هر مقدار مناسبی انتخاب شود. بعنوان مثال می‌توان از A_o=۱۰۰ و $A_{min} = ۱$ استفاده کرد، در حالیکه برای سادگی می‌توان A_o=۱ و $A_{min} = ۰$ در نظر گرفت. فرض $A_{min} = ۰$ به این معنی است که یک خفاش فقط شکار را یافته و به طور موقت انتشار صدا را متوقف می‌کند.

$$A_i^{t+1} = \alpha A_i^t, \quad r_i^{t+1} = r_i^0[1-e^{-yt}] \qquad (۷.۳-۴)$$

که α و y ثابت هستند. در واقع α شبیه فاکتور خنک‌سازی[۱] یک جدول سردکُنی در الگوریتم تبرید شبیه سازی شده است. برای هر $0 < \alpha < 1$ و $y > 0$ داریم:

$$A_i^t \to 0 \,, r_i^t \to r_i^0 \,, \text{ while } t \to \infty \qquad (۷.۳-۵)$$

در حالت ساده، می‌توان $y = \alpha$ در نظر گرفت که ما نیز در شبیه‌سازی خود از $y = \alpha = 0.9$ استفاده کرده‌ایم. انتخاب پارامترها نیازمند انجام برخی آزمایشات است. در ابتدا، هر خفاش باید مقادیر متفاوتی از بلندی صدا و سرعت انتشار پالس داشته باشد که این امر می‌تواند با تصادفی‌سازی به دست آید. برای مثال، بلندی صدای اولیه A_i^o می‌تواند در بازه [۱و۲] قرار گیرد در حالی که سرعت انتشار اولیه r_i^o می‌تواند در حدود صفر باشد یا در صورت استفاده از معادله فوق، هر مقداری مانند $r_i^o \in$ [۰و۱] در نظر گرفته شود. بلندی صدا و نرخ انتشار تنها در صورتی که پاسخ‌های جدید بهبود یابند، به روز رسانی می‌شوند که به معنای حرکت خفاش‌ها به سمت پاسخ بهینه است.

۳-۳-۷ پیاده‌سازی الگوریتم روی مثال مخزن هم زده شده پیوسته راکتور شیمیایی

پیاده سازی الگوریتم خفاش براساس شبه کد داده شده، در هر زبان برنامه نویسی نسبتا سر راست است. برای سادگی آن را در متلب و روی مسئله ۲-۶.۲ مربوط به مخزن هم زده شده پیوسته راکتور شیمیایی پیاده سازی کرده که نتایج به دست آمده به شرح زیر می‌باشد:

در این مثال معادلات حالت مخزن با شرایط اولیه برابر $x(0) = [0.05 \;\; 0]^T$ به صورت زیر می باشد که در آن $x_1(t) = T(t)$ (انحراف از دمای حالت ماندگار)، $x_2(t) = C(t)$ (انحراف از غلظت در حالت ماندگار) و $u(t)$ ورودی کنترلی نرمالیزه شده است که تاثیر جریان مایع خنک کننده در واکنش شیمیایی را نشان می‌دهد. دو معادله زیر همان معادلات حالت ۳۲-۶.۳ می‌باشد

$$\dot{x}_1 = -2\left[x_1(t)+0.25\right]+\left[x_2(t)+0.5\right]\exp\left[\frac{25x_1(t)}{x_1(t)+2}\right]-\left[x_1(t)+0.25\right]u(t) \qquad (۷.۳-۶)$$

$$\dot{x}_2 = 0.5-x_2(t)-\left[x_2(t)+0.5\right]\exp\left[\frac{25x_1(t)}{x_1(t)+2}\right] \qquad (۷.۳-۷)$$

تابع معیاری که باید حداقل شود همان معادله ۳۳-۶.۲ بوده که به صورت زیر می‌باشد

$$J = \int_0^{0.78} \left[x^2{}_1(t)+x^2{}_2(t)+Ru^2(t)\right]dt \qquad (۷.۳-۸)$$

همانگونه که ملاحظه می‌شود در این مسئله با یک سیستم دینامیکی مواجه هستیم و این در حالی است که الگوریتم‌های فراابتکاری همچون الگوریتم خفاش برای حل مسائل استاتیکی ارائه شده‌اند. بعبارتی دیگر، این الگوریتم‌ها مستقل از گرادیان بوده و مفهوم مشتق متغیرهای حالت سیستم‌های دینامیکی را در بر نمی‌گیرند. از طرفی دیگر، در این الگوریتم‌ها مقدار ثابتی بعنوان ورودی در نظر گرفته شده و به تبع آن مقدار ثابتی بعنوان خروجی به دست می‌آید. اما همانطور که می‌دانیم، ورودیهای سیستم‌های دینامیکی لزوما مقدار ثابتی نبوده و می‌تواند تابعی از زمان باشند. راه حل موجود برای مواجهه با چنین

---

[۱] Cooling factor

مشکلاتی، در درجه اول گسسته‌سازی سیستم و در درجه دوم ایجاد برداری متشکل از مقادیر حاصل از گسسته‌سازی ورودی تابع زمان سیستم دینامیکی و استفاده از آن بعنوان ورودی ثابت مرسوم، در تکرارها است. بعبارتی دیگر، همانند روال استفاده شده در فصل ششم برای الگوریتم سریع‌ترین سقوط، ابتدا معادلات حالت سیستم و تابع هزینه‌ی مورد نظر را با گام $\Delta t = ۰,۰۱$ گسسته سازی می‌شود. لذا، در این حالت ورودی کنترلی (موقعیت هر خفاش یا متغیرهای الگوریتم) از حالت پیوسته در زمان به برداری متشکل از ۷۸ قسمت (۷۸ بُعد) تبدیل شده است ($t_f = N.\Delta t$) و در هر بار تکرار، حالت‌های سیستم با استفاده از چنین ورودی‌های کنترلی تصادفی محاسبه و در نهایت مقدار تابع هزینه بدست می‌آید.
بطور کلی، در گام اول سیگنال کنترلی را به فواصل زمانی ۰.۰۱ ثانیه گسسته می‌کنیم و برای هر خفاش یک سیگنال کنترلی گسسته در بازه (t_f و ۰) به صورت تصادفی اختصاص می‌دهیم (مکان اولیه خفاش‌ها را به صورت تصادفی تعیین می‌کنیم). لذا در این قسمت ما به ازای موقعیت خفاش i-ام یک بردار به شکل زیر داریم:

$$u_i = \left[u_i(0) \quad u_i(0.01) \quad \dots \quad u_i(t_f) \right]$$

بعبارتی دیگر، در ابتدا عامل‌های خود را بطور تصادفی در فضای جست و جو پخش می‌کنیم. در گام بعدی، دینامیک سیستم را نیز به فواصل زمانی ۰.۰۱ ثانیه گسسته می‌کنیم. برای هر خفاش با استفاده از رابطه اخیر مقدار حالت‌هایش[۱] را نیز می‌یابیم. بعبارتی، می‌توان با استفاده از $u_i(۰)$ و $x_i(۰)$ خفاش i-ام، $x_i(۰.۰۱)$ آن را محاسبه کرد. با ادامه این روند نیز می‌توان $x_i(t_f)$ را بدست آورد که در نهایت بردار متغیرهای هر خفاش بصورت زیر بدست می‌آید:

$$x_i = [x_i(۰) \quad x_i(۰.۰۱) \quad \dots \quad x_i(t_f)]$$

حال نوبت گسسته سازی تابع هزینه است که در نهایت به فرم زیر تبدیل می‌شود

$$J = \int_0^t g(x,u)dt + \int_t^{2t} g(x,u)dt + \dots + \int_{(N-1)t}^{Nt} g(x,u)dt \xrightarrow{\Delta t << 1} \sum_{k=0}^{N-1} g\left(x(k),u(k)\right) \qquad (۷.۳-۸)$$

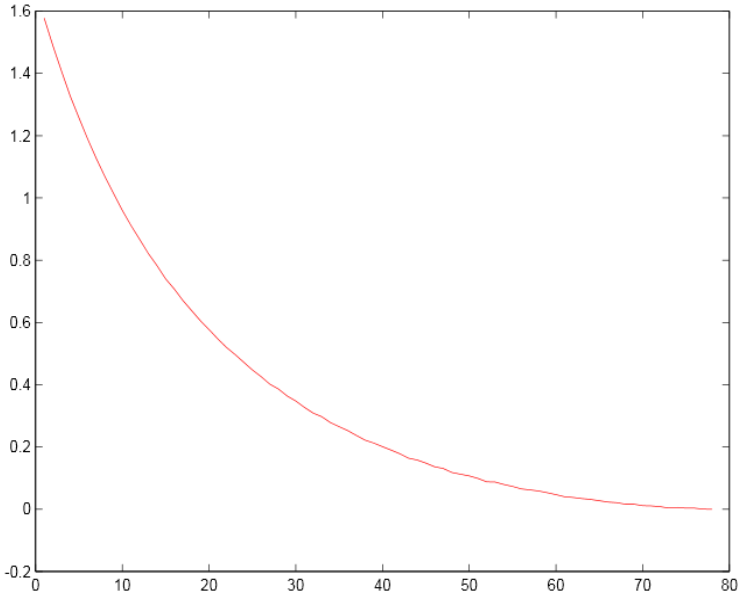
با این عمل و با استفاده از روابط فوق، مقدار هزینه را برای هر خفاش پیدا کرده و با هزینه سایر خفاش‌ها مقایسه می‌کنیم. در نتیجه، موقعیت خفاشی که کمترین هزینه را داراست (خفاشی که بیشترین غذا را پیدا کرده) بعنوان بهترین موقعیت موجود، هدف سایر خفاش‌ها قرار می‌گیرد. بعبارت دیگر سیگنال کنترلی این خفاش هدف می‌شود و بقیه خفاش‌ها تلاش می‌کنند تا سیگنال کنترلی خود را به آن نزدیک کنند (بعبارت دیگر، عامل‌ها شروع به جست و جو در فضای جست و جو می‌کنند). الگوریتم این عمل را بصورت ایجاد یک سری انحرافات تصادفی در پاسخ‌ها محقق می‌سازد که منجر به یک پاسخ جدید در همسایگی نقطه‌ی قبلی می‌گردد. این فرآیند تکراری بصورت مکان‌یابی اکو ریزخفاش‌ها ادامه می‌یابد تا اینکه در آخر همه خفاش‌ها به یک سیگنال کنترلی نزدیک به بهینه برسند.
رویه کلی شبیه‌سازی عملکرد الگوریتم بدین صورت است که ابتدا مقادیر اولیه الگوریتم را تعریف می‌کنیم، لذا در این مرحله ما به ترتیب از ۲۰، ۳۰، ۴۰و ۵۰ خفاش با ۷۸ بعد (تعداد نقاط گسسته سازی)

---

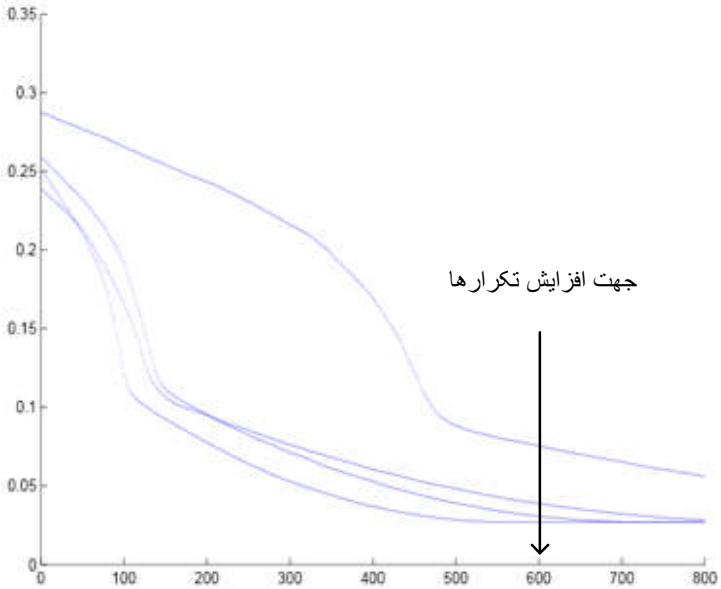
[۱] State

در ۸۰۰ بار تکرار با بلندی صدای ۰.۵ و سرعت انتشار پالس صفر، $f_{min}=0$، $f_{max}=2$، سرعت و فرکانس اولیه صفر استفاده کردیم. در ادامه باید محدوده‌های پاسخ را تعیین کرد که در این مسئله هیچ محدودیتی روی پاسخ‌ها (ورودی‌های کنترلی) نداشتیم. در آخر پاسخ‌های الگوریتم (ورودی‌های کنترلی) را بصورت تصادفی مقداردهی کرده و از این میان بهترین پاسخ و مقدار تابع هزینه آن را مشخص و به ترتیب در متغیرهای $best$ و f_{min} ذخیره می‌نماییم. در نهایت برنامه مذکور اجرا شده و نتایج آن بصورت زیر حاصل می‌شود.

همانگونه که ملاحظه می‌گردد با ورودی کنترلی بدست آمده حداقل مقدار تابع هزینه ۰.۰۲۷۳ حاصل شده است. نکته قابل توجه از شکل حاصله از تغییرات مقدار تابع هزینه در طی اجرای الگوریتم، این است که با افزایش تعداد خفاش‌ها سرعت همگرایی اولیه الگوریتم افزایش می‌یابد که این نیز به بهای افزایش محاسبات و در نتیجه افزایش مدت زمان اجرای سیستم تمام می‌شود.



شکل الف۲-۷ نمودار تغییرات ورودی کنترلی برحسب زمان



شکل ب۲-۷ نمودار تغییرات مقدار تابع هزینه به ازای ۲۰، ۳۰، ۴۰ و ۵۰ خفاش برحسب تکرارها