**دانشگاه آزاد اسلامی**

**جزوه درس:**

# هوش مصنوعی و سیستم‌های کارشناس

کارشناسی ارشد مهندسی برق


تهیه و تنظیم از:

**دکتر داود عزیزیان**

**عضو هیأت علمی دانشگاه آزاد اسلامی واحد ابهر**

d.azizian@abhariau.ac.ir

# پیشگفتار:

جزوهٔ حاضر برای درس هوش مصنوعی و سیستم‌های کارشناس، بگونه‌ای تدوین شده است که ضمن ارائه مطالب بصورت خلاصه و مختصر، در کنار مطالب ارائه شده در کلاس، بتواند کمکی برای فهم و درک بهتر این درس برای دانشجویان کارشناسی ارشد رشتهٔ مهندسی برق باشد.

داود عزیزیان

# مراجع:


۱- **"مروری بر برخی روش‌های بهینه‌سازی هوشمند"، حبیب مطیع قادر.، دانشگاه آزاد اسلامی شبستر.**

۲- **"الگوریتم‌های بهینه‌سازی الهام گرفته از طبیعت"، فرشاد مریخ بیات، انتشارات نص.**

۳- **برخی مقالات و پایان نامه‌های مرتبط با هوش مصنوعی**

۴- **راهنمای جعبه ابزار شبکه‌های عصبی در نرم افزار متلب.**


# ارزیابی و نمره نهایی:

ارزیابی نهایی بر اساس موارد زیر انجام خواهد شد:

- **امتحان پایانترم: ۱۲ نمره**
- **پروژهٔ پایه ( نمونه موجود): ۶ نمره**
- **پروژهٔ پیشرفته (اعمال الگوریتم‌های جدید بر روی موضوعات موجود): ۸ نمره**
- **پروژهٔ نوآورانه (استفاده از الگوریتم‌های ارائه شده در ۳ سال اخیر یا پیشنهاد موضوعات کاملاً نو): ۹ تا ۱۰ نمره**
- **ارائهٔ مقاله یا الگوریتم‌های بدیع و تازه: ۱ تا ۲ نمره** اضافی

# سرفصل مطالب

## فصل اول: مقدمه‌ای بر هوش مصنوعی و روش‌های بهینه‌سازی و جستجو

- روش‌های بهینه‌سازی و جستجو
- روش‌های بهینه‌سازی کمینه جو
- توابع توزیع احتمال

## فصل دوم: الگوریتم‌های تکاملی (الگوریتم ژنتیک)

- تکامل طبیعی و نحوهٔ کارکرد الگوریتم ژنتیک
- روش‌های کدگذاری
- - تابع هدف
- انتخاب و تولید نسل جدید
- آمیزش و ترکیب
- جهش
- همگرایی، شرایط خاتمه و محدودیت‌ها
- انواع الگوریتم ژنتیک

## فصل سوم: الگوریتم‌های مبتنی بر هوش جمعی

- الگوریتم تجمع ذرات
- الگوریتم مورچگان
- الگوریتم زنبور عسل
- الگوریتم باکتری

## فصل چهارم: الگوریتم‌های متفرقه

- الگوریتم شبیه ساز سرد شدن فلزات
- الگوریتم انفجار بزرگ

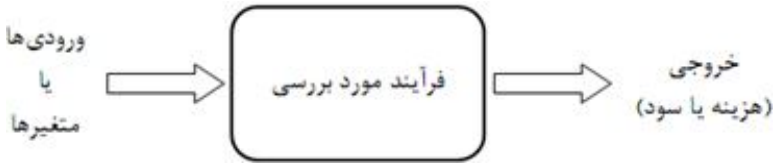
## فصل پنجم: شبکه‌های عصبی مصنوعی

# فصل اول:

# مقدمه‌ای بر هوش مصنوعی و روش‌های بهینه‌سازی

## ۱-۱- روش‌های بهینه‌سازی و جستجو

بهینه‌سازی فرآیندی است که برای بهتر کردن چیزی (فکر ، ایده و طرحی) دنبال می‌شود. بهینه‌سازی ابزاری ریاضی برای یافتن پاسخ بسیاری از مسائل است. هدف در بهینه‌سازی یافتن بهترین جواب برای یک مساله است (از میان بیش از یک جواب برای مساله که دارای ارزش یکسانی نیستند). تعریف بهترین جواب، به نوع مساله و محدودیت‌ها وابسته است. نحوه فرمول بندی مساله نیز بر چگونگی تعریف بهترین جواب تاثیر دارد. برخی از مسایل جواب‌های مشخصی دارند (مسایل ساده): بهترین بازیکن یک رشتهٔ ورزشی، طولانی‌ترین روز سال و پاسخ یک معادله ی دیفرانسیل معمولی درجه اول. اما برخی مسایل دارای جواب های ماکزیمم یا مینیمم (نقاط بهینه یا اکسترمم) متعددی بوده و بهترین جواب معمولاً یک مفهوم نسبی است (مانند بهترین اثر هنری ، زیباترین منظره و گوش نوازترین قطعه ی موسیقی). بهینه‌سازی، فرآیند تغییر دادن ورودی‌ها یا متغیرهای یک دستگاه/ فرآیند (شکل ۱) و یا تابع (تابع هدف "objective function"، تابع هزینه " cost function" و یا تابع برازندگی "fitness function") بنحویست که بهترین نتیجه یا خروجی حاصل آید. هر نوع مساله بهینه‌سازی را می‌توان بصورت مینیمم سازی یا ماکزیمم سازی مقدار یک تابع هزینه مطرح کرد.




**شکل ۱- فرایند بهینه‌سازی**

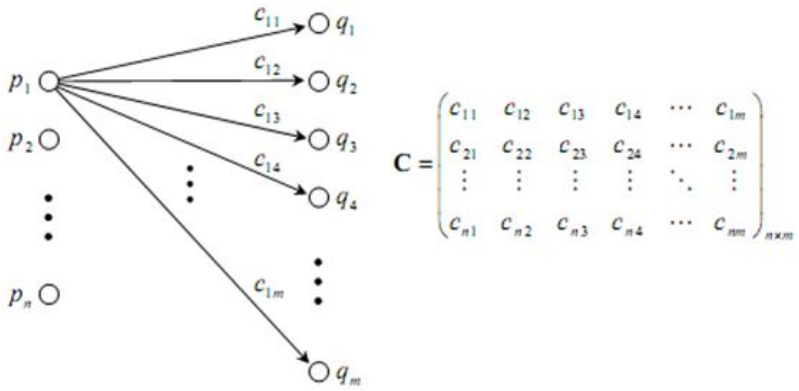
یک مساله بهینه‌سازی از نظر ریاضی به صورت زیر بیان می‌شود:

$$\text{minimize } f(x) \text{ subject to } g_i(x) \leq b_i, \quad i = 1,2,\dots,m \qquad (۱)$$

که در آن $x=(x_1,x_2,\dots,x_n) \in R^n$، ورودی‌ها یا متغیرهای مستقل مساله‌اند و تابع هدف بصورت $f: R^n \to R$ تعریف شده. مجموعهٔ توابع $g_i: R^n \to R$ نیز قیود و محدودیت‌های نامساوی مسئله را بیان می‌کنند. می‌توان زیر مجموعه‌ای از $R^n$ مانند $\Omega$ یافت، بنحوی که قیود نامساوی برای همهٔ اعضای این مجموعه برقرار باشند (اصطلاحاً $\Omega$ را مجموعهٔ شدنی یا feasible می‌نامند).

هر گاه تابع هدف و قیود مساله همگی خطی باشند، یک مساله برنامه ریزی خطی مطرح می‌شود (تابعی خطی است که $f(ax + \beta y) = af(x) + \beta f(y)$). روش‌های مختلفی برای حل مسایل بهینه‌سازی خطی وجود دارند. یکی از مسایل مشهور در زمینهٔ بهینه‌سازی خطی ، مساله حمل و نقل است.

**مساله حمل و نقل :** محصولی در $n$ واحد تولیدی بمقدار $p_1$، $p_2$، ... و $p_n$ تولید و به $m$ مصرف‌کننده بمیزان $q_1$، $q_2$، ... و $q_m$ ارسال می‌شود ($C_{ij}$: هزینهٔ انتقال هر واحد کالا از واحد تولیدی $i$ به مصرف کنندهٔ $j$). برای دستیابی به کمترین هزینهٔ حمل و نقل، هر واحد تولیدی چه میزان از نیاز مصرف کننده‌ها را باید تأمین کنند؟




**شکل ۲- بیان شماتیک مسئلهٔ حمل و نقل**

در حالت کلی مسایل بهینه‌سازی را می‌توان از دیدگاه‌های زیر تقسیم‌بندی کرد:

- **بهینه‌سازی با سعی و خطا (trial and error) / بهینه‌سازی روی تابع:** سعی و خطا فرآیندی است که در آن متغیرهای ورودی تغییر داده می‌شوند، بدون آنکه اطلاع دقیقی از نحوهٔ تأثیر آنها بر خروجی در دست باشد. بسیاری از کشفیات بشر، نتیجهٔ این نوع بهینه‌سازی است. در روش دیگر، ماهیت مسأله بصورت فرمول دقیق در دست بوده و می‌توان از روش‌های ریاضی برای بهینه‌سازی استفاده کرد.
- **بهینه‌سازی تک بعدی / بهینه‌سازی چند بعدی:** در بهینه‌سازی تک بعدی فقط یک متغیر در مساله وجود داشته و این در حالیست که مسایل بهینه‌سازی چند بعدی دارای بیش از یک متغیر هستند.
- **بهینه‌سازی پویا / بهینه‌سازی ایستا:** اگر تابع یا فرآیند تحت بررسی در بهینه‌سازی، تابعی از زمان باشد، بهینه‌سازی را پویا می نامند؛ در غیر اینصورت بهینه‌سازی ایستا خوانده می‌شود. بعنوان

مثال، یافتن بهترین مسیر رانندگی می‌تواند یک مسالهٔ بهینه‌سازی ایستا باشد. اما در عمل عواملی چون میزان ترافیک متغیر در ساعات مختلف، سبب تبدیل آن به یک مسئلهٔ متغیر با زمان می‌شود.

- **بهینه‌سازی گسسته / بهینه‌سازی پیوسته:** اگر ماهیت متغیرهای مساله پیوسته باشد، بهینه سازی را پیوسته و در غیر اینصورت گسسته می‌نامند.
- **بهینه‌سازی مقید / بهینه‌سازی بدون قید:** در برخی مسایل متغیرها نمی‌توانند هر مقداری را اختیار کنند و می‌بایست مقادیر متغیرها یک مجموعه از شرایط را برآورده کنند. این شرایط را قید و مسایل توام با قید را مقید می نامند. هر مساله که قیدی در آن وجود نداشته. بدون قید خوانده می‌شود.
- **بهینه‌سازی مینیمم جو / بهینه‌سازی تصادفی:** در روش بهینه سازی مینیمم جو از یک نقطهٔ مشخص در فضای جستجو شروع کرده و با استفاده از قوانینی بر پایه ریاضیات و روندی مشخص جواب‌های بهتری بدست می‌آورند. در این روش‌ها سرعت همگرایی بسیار بالاست و براحتی در مینیمم یا ماکزیمم‌های محلی گرفتار می‌شوند. در مقابل روش‌های تصادفی، از الگوهای احتمالی برای ایجاد جواب های بهتر استفاده کرده و پیش بینی عملکرد الگوریتم ساده نیست. سرعت همگرایی و نیز احتمال گرفتار شدن در نقاط بهینهٔ محلی در این الگوریتم‌ها در مقایسه با الگوریتم‌های مینیمم جو، کمتر است.

## ۱-۲- روش‌های جستجو و بهینه‌سازی مینیمم (کمینه) جو

در علوم کامپیوتر و ریاضیات، یک الگوریتم جستجو، الگوریتمی است که یک مسأله را بعنوان ورودی گرفته و بعد از ارزیابی راه‌حل‌های ممکن، یک راه‌حل را برای آن مسأله برمی‌گرداند. هنگامی که مسأله‌ای را حل می‌کنیم معمولاً دنبال آن هستیم که بهترین راه‌حل (حل بهینه) از بین حل‌های ممکن را برای مسأله بیابیم. محدوده‌ای که جواب‌های مسأله (از جمله جواب بهینه) در آن قابل قبول می‌باشند، فضای جستجو ( Search Space) نامیده می‌شود؛ بعبارتی مجموعهٔ راه‌حل‌های ممکن برای یک مسأله را فضای جستجو می‌نامند.

در مسائل جستجو بهینه سازی با دو دسته کلی از الگوریتم‌ها مواجه هستیم؛ بعضی الگوریتم‌ها که با عنوان الگوریتم‌های ناآگاهانه شناخته می‌شوند، از روش‌های ساده برای جستجوی فضای نمونه استفاده می‌کنند. یک الگوریتم جستجوی ناآگاهانه به ماهیت مسأله کاری نداشته و می‌تواند بطور عمومی برای محدودهٔ

وسیعی از مسائل طراحی شود. از جمله مشکلات این الگوریتم‌ها، بزرگ بودن فضای جستجو و زمان زیاد جستجو می‌باشد. در حالی که الگوریتم‌های آگاهانه با استفاده روش‌های دانش محور می‌کوشند تا زمان جستجو را کاهش دهند.

**روش تحلیلی:** این روش برای برخی از توابع هزینه، معتبر و بسادگی قابل استفاده است. در بسیاری از مسایل پیچده نیز با اندکی اصلاح، می‌توان مقدار مینیمم یا ماکزیمم را با روش‌های تحلیلی پیدا کرد. نقاط اکسترمم هر تابع تک متغیره، صفرهای مشتق آنند و اگر علامت مشتق دوم تابع در نقطهٔ اکسترممی منفی (مثبت) باشد، آن یک مینیمم (ماکزیمم) محلی می‌باشد (در توابع چند متغیره نیز بجای مشتق از گرادیان تابع استفاده می‌شود). اما برای مسایلی که تابع هدف پیچیده دارند یا دارای تعریف ریاضی روشنی نیستند، روش تحلیلی کارآیی نداشته و روش‌های دیگری (که در ادامه مطرح شده‌اند) مطرح شده‌اند.

**جستجوی خطی:** روش جستجوی خطی ساده‌ترین روش از دستهٔ روش‌های جستجوی لیست است. در این روش که اساس کار بسیاری از روش‌های بهینه‌سازی چند متغیره است، با شروع از نقطه‌ای در فضای جستجو و حرکت در جهتی خاص، نقاط جدید (جواب‌های بهتر) از رابطهٔ زیر بدست می‌آیند:

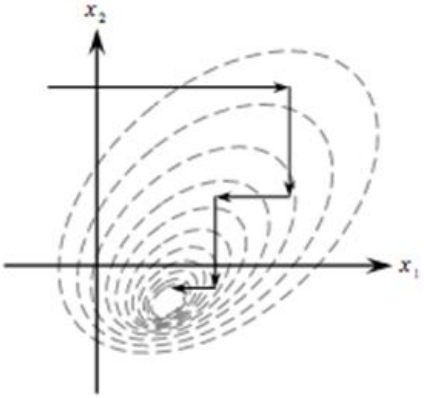
$$x_{k+1} = x_k + a_k d_k \qquad (۲)$$

که $x_k$ جواب یافته شده در مرحلهٔ $k$، $a_k$ اندازهٔ گام بهینه (اسکالر) است. $d_k$ برداری هم بُعد با $x_k$ بوده و جهت و میزان حرکت را تعیین می‌کند. برای یک جستجوی خطی دقیق (exact line search)، $a_k$ می‌تواند از رابطهٔ زیر محاسبه می‌شود ($f$ تابع هزینه‌ایست که باید مینیمم شود):

$$f(x_k + a_k d_k) = \min_a f(x_k + a d_k) \qquad (۳)$$

در یک جستجوی خطی نادقیق و یا قابل قبول، $a_k$ بنحوی انتخاب می‌شود که تابع هزینه در حد قابل قبولی کم ($f(x_{k+1}) - f(x_k)$ منفی و به اندازهٔ کافی کم) شود. از آنجائیکه یافتن مقدار بهینه برای اندازهٔ گام، کاری سخت (محاسبات زیاد) است، در عمل بندرت از روش جستجوی خطی دقیق استفاده می‌شود.

برای تعیین جهت حرکت ($d_k$)، روش‌های متفاوتی وجود دارد. در یک روش، جهت حرکت همواره در راستای مجموعه‌ای از بردارهای پایهٔ متعامد فضای جستجو در نظر گرفته می‌شود. بدین ترتیب که حرکت در یک جهت تا وقتی معتبر است که منجر به کاهش هزینه شود؛ در غیر این صورت، به یکی از جهات دیگر (که عمود بر جهت قبلی است)، تغییر می‌کند (شکل ۳).




**شکل ۳- نمودار یک تابع درجه دو در فضای دو بعدی و مسیر که در بهینه‌سازی خطی**

برای بهبود زمان روش جستجوی خطی می‌توان از روش‌های دیگر جستجوی لیست چون جستجوی دودویی (مناسب برای لیستی با تعداد دادهٔ زیاد)، جستجو با میان‌یابی (مناسب برای داده‌های مرتب شده با تعداد زیاد و توزیع یکنواخت)، الگوریتم گراور و الگوریتم Table Hash استفاده کرد.

**جستجوی درختی:** الگوریتم‌های جستجوی درختی، قلب شیوه‌های جستجو برای داده‌های ساخت یافته‌اند. مبنای اصلی جستجوی درختی، گره‌هایی است که از یک ساختمان داده گرفته شده‌اند. هر عنصر که بخواهد اضافه شود با داده‌های موجود در گره‌های درخت مقایسه و به ساختار درخت اضافه می‌شود. با تغییر ترتیب داده‌ها و قرار دادن آنها در درخت، درخت با شیوه‌های مختلفی جستجو می‌شود.

**جستجوی گراف:** این الگوریتم‌ها توسعه یافتهٔ الگوریتم‌های جستجوی درختی هستند و از آنجمله می‌توان به الگوریتم‌های دیکسترا، اکروسکال، نزدیک‌ترین‌همسایگی و پریم اشاره کرد.

**روش‌های نیوتونی:** دسته‌ای از روش‌های بهینه‌سازی بدون قیدند که از ساده‌ترین آن‌ها روش بیشترین کاهش یا روش گرادیان است. فرض کنید f حول $x_k$ بطور یکنواخت مشتق پذیر بوده و $g_k \triangleq \nabla f(x_k) \neq 0$.

با استفاده از دو جمله بسط تیلور داریم:

$$f(x) \cong f(x_k) + (x - x_k)^T g_k \qquad (۴)$$

طبق الگوی جستجوی خطی $(x - x_k = a_k d_k)$، $d_k$ که در $d_k^T g_k < 0$ صدق کند، می‌تواند بعنوان جهتی کاهشی استفاده شود (زیرا $f(x) < f(x_k)$). با ثابت فرض کردن $a_k$، منفی ترین مقدار $d_k^T g_k$ می‌تواند سریع‌ترین کاهش را بهمراه داشته باشد که این زمانی است که $d_k = -g_k$. بنابراین

$$x_{k+1} = x_k - a_k g_k \qquad (۵)$$

شیوهٔ تعیین $a_k$ مشابه با روش جستجوی خطی می‌باشد. روش گرادیان بسیار به نوع تابع هزینه حساس بوده و فرض مشتق پذیر بودن تابع هزینه، آن را به دستهٔ خاصی از توابع محدود می‌کند.

یکی از نسخه‌های بهبود یافتهٔ روش گرادیان، روش نیوتون است که در آن از مشتق مرتبهٔ دوم تابع هزینه نیز برای تعیین جهت حرکت و کاهش مقدار تابع استفاده می‌شود. در این روش از تقریب درجهٔ دوم تابع $f$ حول $x_k$ بصورت زیر بهره گرفته می‌شود:

$$f(x_{k+1}) = f(x_k + s) \cong f(x_k) + [\nabla f(x_k)]^T s + \frac{1}{2} s^T \nabla^2 f(x_k) s \quad (۶)$$

برای آن که مقدار $f(x_{k+1})$ مینیمم شود، رابطهٔ زیر باید برقرار باشد:

$$x_{k+1} = x_k - [\nabla^2 f(x_k)]^{-1} \nabla f(x_k) = x_k - G_k^{-1} g_k \quad (۷)$$

جهت $G_k^{-1} g_k$، جهتی کاهشی است (جهت کاهشی نیوتون) و در روش نیوتون $a_k$ برابر ماتریس $G_k^{-1}$ است (پیشتر یک اسکالر بود)، که به طور مستقیم از ساختار تابع بدست می‌آید.

ضعف همهٔ روش‌های نیوتونی، استفاده از مشتق می‌باشد، لذا در توابعی که مشتق پذیر نبوده و یا اصلاً تابعی برای مشتق‌گیری وجود ندارد، استفاده از این روش ها ممکن نیست. برای تعدیل مشکلات، روش‌های بهینه‌سازی دیگری چون روش‌های گرادیان مزدوج و شبه نیوتونی (Quasi-Newton) بر پایهٔ روش نیوتونی ارائه شده‌اند.

**جستجوی خصمانه:** در یک بازی مثل شطرنج، یک درخت بازی (شامل تمام حرکات ممکن توسط هر دو بازیکن و نتایج حاصل از ترکیب این حرکات) وجود دارد و ما می‌توانیم این درخت را جستجو کرده و مؤثرترین استراتژی برای بازی را بیابیم. در چنین مواردی از الگوریتم‌های جستجو مثل الگوریتم مینیمم-ماکسیمم (می‌نیمیم مجموعه‌ای از ماکزیمم‌ها)، هرس کردن درخت جستجو و هرس کردن آلفا-بتا استفاده می‌کنند. این الگوریتم‌ها نمونه‌هایی از جستجوی آگاهانه هستند.

**جستجوی آگاهانه و الگوریتم‌های مکاشفه‌ای (Heuristic)**: سیستم‌های پیچیدهٔ اجتماعی، تعداد زیادی از مسائل دارای طبیعت ترکیبی را پیش روی ما قرار می‌دهند: مثلاً در یک مسئلهٔ ترکیبی مسیر کامیون‌های حمل‌ونقل باید تعیین شود، انبارها یا نقاط فروش محصولات باید جایابی شوند، شبکه‌های ارتباطی باید طراحی شوند، کانتینرها باید بارگیری شوند، رابط‌های رادیویی می‌بایست دارای فرکانس مناسب باشند، مواد اولیه چوب، فلز، شیشه و چرم باید به اندازه‌های لازم بریده شوند.

تئوری پیچیدگی به ما می‌گوید که مسائل ترکیبی اغلب چندجمله‌ای نبوده ودر عمل به اندازه‌ای بزرگند که بدست آوردن جواب بهینهٔ آنها بسیادگی میسر نیست. در یک مسئلهٔ بهینه‌سازی واقعی می‌بایست به جواب‌های تخمینی با کیفیت و زمان قابل پذیرش بسنده کرد. اغلب روش‌های کلاسیک ریاضیات، نقطه بهینهٔ محلی (Local Optimal) را بعنوان نقطه بهینه انتخاب می‌کنند و هر یک از این روش‌ها تنها برای مسأله خاصی کاربرد دارند. الگوریتم‌هایی وجود دارند که می‌توانند یافتن جواب‌های خوب در فاصله مشخصی از جواب بهینه را تضمین کنند (الگوریتم‌های تقریبی) و الگوریتم‌های نیز هستند که تضمین می‌دهند با احتمال بالا جواب نزدیک به بهینه تولید کنند (الگوریتم‌های احتمالی). جدای از این دو دسته، می‌توان الگوریتم‌هایی را پذیرفت که هیچ تضمینی در ارائه جواب نداشته اما بطور متوسط بهترین تقابل کیفیت و زمان حل را برای مسائل ترکیبی به همراه دارند (الگوریتم‌های مکاشفه‌ای).

الگوریتم‌های مکاشفه‌ای عبارتند از معیارها، روشها یا اصولی برای تصمیم‌گیری بین چندین خط‌مشی و انتخاب اثربخش‌ترین آنها برای دستیابی به اهداف موردنظر. در این الگوریتم‌ها جواب بهینه نتیجهٔ اعتدال بین نیاز به ساخت معیارهای ساده و در عین حال توانایی تمایز درست بین انتخاب‌های خوب و بد می‌باشد.

یک الگوریتم مکاشفه‌ای می‌تواند حسابی سرانگشتی برای هدایت یک دسته از اقدامات باشد. برای مثال، یک روش مشهور برای انتخاب طالبی رسیده عبارتست از فشار دادن محل اتصال به ریشه از یک طالبی نامزد و سپس بو کردن آن محل؛ اگر بوی آن محل مانند بوی داخل طالبی باشد، آن طالبی به احتمال زیاد رسیده است. این قاعده سرانگشتی نه تضمین می‌کند که تنها طالبی‌های رسیده به عنوان نامزد انتخاب شوند و نه تضمین می‌کند که طالبی‌های رسیده آزمایش‌شده، تشخیص داده شوند (بهرحال این روش، اثربخش‌ترین روش است).

در مثالی دیگر یک استاد شطرنج را در نظر بگیرید که با انتخاب بین چندین حرکت ممکن روبرو شده است. وی ممکن است تصمیم بگیرد که یک حرکت اثربخش‌ترین خواهد بود، زیرا موقعیتی فراهم می‌کند که بنظر می‌رسد بهتر از موقعیت‌های حاصل از حرکت‌های دیگر باشد. این واقعیت که اساتید بزرگ شطرنج همواره پیروز نخواهند بود، نشان دهنده این است که الگوریتم‌های مکاشفه‌ای انتخاب اثربخش‌ترین حرکت را تضمین نمی‌کنند.

بیشتر مسائل پیچیده نیازمند ارزیابی تعداد انبوهی از حالت‌های ممکن برای تعیین یک جواب دقیق می‌باشند و زمان لازم برای یافتن یک جواب دقیق اغلب در یک زمان محدود میسر نیست. الگوریتم‌های

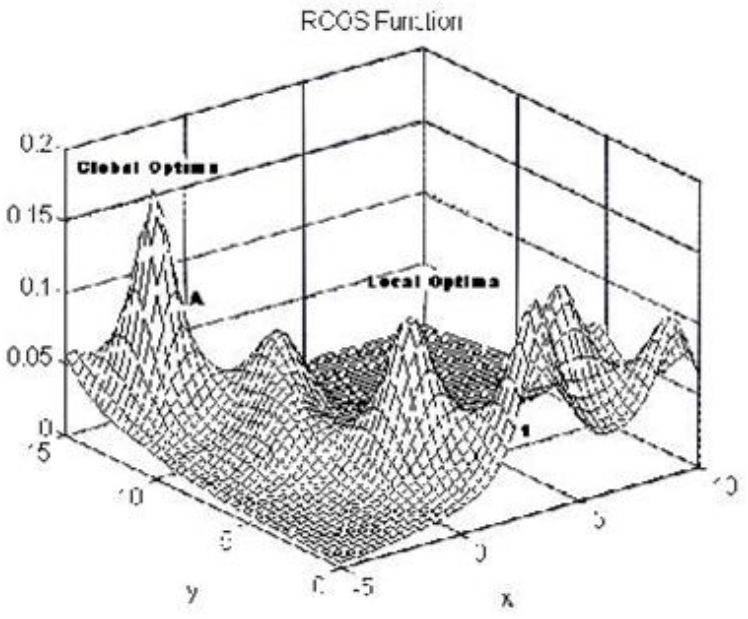
مکاشفه‌ای با ارزیابی‌های کمتر که جوابهایی در محدودهٔ قابل قبول ارایه می‌نمایند، در حل چنین مسائلی اثربخش خواهند بود.

در حالت کلی الگوریتم‌های مکاشفه‌ای به سه دسته زیر تقسیم می‌شوند:

- روش‌هایی که بر ویژگی‌های ساختاری مسأله و جواب متمرکز شده و با کمک آنها الگوریتم‌های جستجوی محلی تعریف می‌کنند.
- روش‌هایی که بر هدایت مکاشفه‌ای یک الگوریتم جستجوی محلی متمرکز و شرایطی فراهم می‌آورند که آن الگوریتم بتواند بر فرار از بهینه محلی غلبه کند.
- روش‌هایی که بر ترکیب یک مفهوم مکاشفه‌ای با انواعی از برنامه‌ریزی ریاضی متمرکز می‌شوند.

مکاشفه‌ای‌های نوع اول می‌توانند خیلی خوب عمل کنند، اما ممکن است در بهینه‌های محلی گیر کرده و هیچ شانسی برای فرار از آنها نداشته باشند. برای تعدیل این مشکل، الگوریتم‌های فرا مکاشفه‌ای مطرح شده‌اند که تقابل بین ایجاد تنوع جستجو (وقتی جستجو به سمت مناطق نامناسب رود) و تشدید جستجو (برای دستیابی به بهترین جواب در فضای جستجو) را مدیریت می‌کنند. از بین آنها می‌توان به الگوریتم‌های هوشمندی چون ژنتیک، بهینه‌سازی مورچه، شبکه‌های عصبی مصنوعی و ... اشاره کرد.

نکته: منحنی شکل زیر را در نظر بگیرید که دارای دو نقطه ماکزیمم است (تنها یکی از آنها ماکزیمم سرتاسری است). درصورت استفاده از روش‌های بهینه‌سازی سنتی و ریاضی، مجبوریم تا در یک بازه بسیار کوچک مقدار ماکزیمم تابع را بیابیم.




**شکل ۴- نقاط بهینهٔ محلی و سرتاسری**

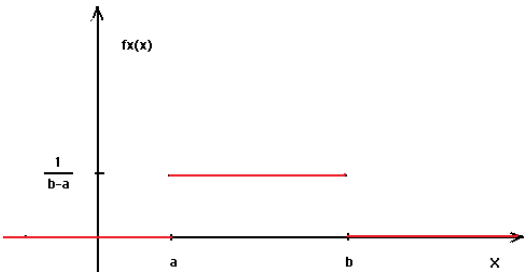
بدیهی است اگر از نقطه ۱ شروع کنیم تنها به یک ماکزیمم محلی دست یافته و الگوریتم متوقف خواهد شد. اما در روش‌های هوشمند، بدلیل خصلت تصادفی آنها، حتی اگر از نقطه ۱ شروع کنیم باز ممکن است در میانهٔ راه نقطه A به صورت تصادفی انتخاب شود. بنابراین شانس دست‌یابی به نقطهٔ بهینه سرتاسری (Global Optimal) وجود خواهد داشت. ضمناً روش‌های ریاضی اغلب منجر به یک دستورالعمل خاص برای حل هر مسأله می‌شوند، در حالیکه روش‌های هوشمند قابلیت بکارگیری در حل هر مسأله‌ای را دارند.

## ۱-۳- انواع توابع توزیع احتمال در متغیرهای تصادفی

با توجه به کاربرد گستردهٔ متغیرهای تصادفی در الگوریتم‌های هوشمند جستجو و بهینه سازی، در ادامه سه نوع توزیع متداول از توابع احتمال تصادفی مرور شده است.

* متغیر تصادفی یکنواخت: متغیر تصادفی X دارای توزیع یکنواخت در فاصله (a,b) است، هرگاه دارای تابع چگالی احتمالی بفرم زیر باشد (شکل ۵):

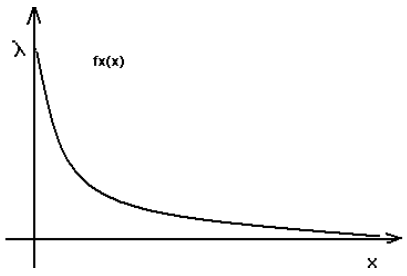
$$f_X(x)=\begin{cases}\dfrac{1}{b-a} & a<x<b\\ 0 & o.w\end{cases} \qquad (۸)$$




**شکل ۵- تابع چگالی احتمالی یکنواخت**

* متغیر تصادفی نمایی: متغیر تصادفی دارای توزیع نمایی است، اگر دارای تابع چگالی احتمالی زیر باشد:

$$f_X(x)=\lambda e^{-\lambda x} \qquad x>0 \qquad (۹)$$

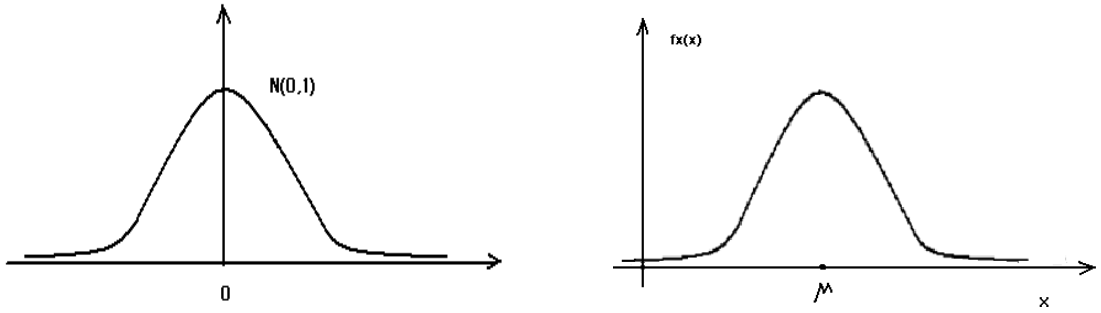



**شکل ۶- تابع چگالی احتمالی نمایی**

* متغیر تصادفی نرمال: متغیر تصادفی X دارای توزیع نرمال با میانگین $\mu$ و واریانس $\sigma^2$ است، هرگاه دارای تابع چگالی احتمالی بفرم زیر باشد (شکل۷-الف).

$$f_X(x) = \frac{1}{\sqrt{2\sigma^2}} e^{\frac{-1}{2\sigma^2}(x-\mu)^2} \qquad -\infty < x < +\infty, \quad \mu \in \Re, \quad \sigma^2 > 0 \qquad (۱۰)$$

اگر متغیر تصادفی دلخواه X دارای توزیع نرمال با میانگین $\mu$ و واریانس $\sigma^2$ باشد، آنگاه متغیر تصادفی $Z = \frac{X-\mu}{\sigma}$ دارای توزیع نرمال با میانگین ۰ و واریانس ۱ است که بدان نرمال استاندارد گویند (شکل۷-ب).




(ب) (الف)

**شکل ۷- الف) تابع چگالی احتمالی نرمال (کوشی) و ب) فرم استاندارد آن**

خواص توزیع نرمال را می‌توان بصورت زیر خلاصه کرد:

- منحنی نرمال تنها دارای یک نقطه ماکزیمم در نقطه $x = \mu$است.
- منحنی نرمال دارای دو نقطه عطف در نقاط $x = \mu - \sigma$ و $x = \mu + \sigma$ است.
- منحنی نسبت به خط $x = \mu$ متقارن است، یعنی $f_X(x-a) = f_X(x+a)$
- مساحت زیر منحنی نرمال برابر ۱ است.

# فصل دوم:

# الگوریتم‌های تکاملی (الگوریتم ژنتیک)

## ۲-۱- تکامل طبیعی (قانون انتخاب طبیعی داروین)

اولین کسی که توانست قوانین حاکم بر انتقال صفات ارثی را شناسایی کند، کشیشی اتریشی به نام «گریگور مندل» بود. اما متأسفانه جامعهٔ علمی آن دوران به دیدگاه‌ها و کشفیات او اهمیت چندانی نداد و نتایج کارهای «مندل» بدست فراموشی سپرده شد. در سال ۱۹۰۰ میلادی کشف مجدد قوانین «مندل»، توسط «درویس»، «شرماک» و «کورنز» باعث شد که نظریات او مورد توجه و قبول قرار گرفته و «مندل» به عنوان پدر علم ژنتیک شناخته شود. در سال ۱۹۵۳ با کشف ساختمان جایگاه ژن‌ها از سوی «جیمز واتسون» و «فرانسیس کریک»، رشته‌ای جدید در علم زیست‌شناسی بوجود آمد که زیست‌شناسی مولکولی نام گرفت. در خلال اوایل دههٔ هفتاد، دو رشته زیست‌شناسی مولکولی و ژنتیک ادغام شدند و رشته‌ای به نام «مهندسی‌ژنتیک» را بوجود آوردند که طی اندک زمانی توانست رشته‌های مختلفی اعم از پزشکی، صنعت و کشاورزی را تحت‌الشعاع خود قرار دهد.

هنگامی که لغت تنازع بقا به کار می‌رود اغلب بار ارزشی منفی آن به ذهن می‌آید. شاید همزمان قانون جنگل به ذهن برسد و حکم بقای قوی‌ترها! البته همیشه هم قوی‌ترین‌ها برنده نبوده‌اند. مثلاً دایناسورها با وجود جثه عظیم و قوی‌تر بودن در طی روندی کاملاً طبیعی بازیِ بقا و ادامه نسل را واگذار کردند در حالی که موجوداتی بسیار ضعیف‌تر از آنها حیات خویش را ادامه دادند. ظاهراً طبیعت، بهترین‌ها را تنها بر اساس هیکل انتخاب نمی‌کند! در واقع درست‌تر آنست که بگوییم طبیعت مناسب‌ترین‌ها را انتخاب می‌کند نه بهترین‌ها.

قانون انتخاب طبیعی (Natural Selection) بدین صورت است که تنها گونه‌هایی از یک جمعیت ادامه نسل می‌دهند که بهترین خصوصیات را داشته باشند و آنهایی که این خصوصیات را نداشته باشند به تدریج و در طی زمان از بین می‌روند. برای نمونه در میان یک گله گرگ با افراد دارای ژن‌های گوناگون، گرگی احتمال بقای بالاتر را دارد که قوی‌تر از دیگران است، چراکه هم امکان بیشتری برای جفت‌گیری و گسترش ژن خود

را دارد و هم بیش از گرگ‌های دیگر به شکار دست یافته و زنده می‌ماند. در مثالی دیگر، فرض کنید گونهٔ خاصی از افراد، هوش بیشتری از بقیه افراد یک جامعه دارند. در شرایط طبیعی، این افراد پیشرفت و رفاه نسبتاً بالاتری خواهند داشت و این رفاه، خود باعث طول عمر بیشتر و باروری بهتر خواهد بود. حال اگر این خصوصیّت (هوش) ارثی باشد، بالطبع بدلیل زاد و ولد بیشتر، تعداد افراد باهوش در نسل بعد جامعه بیشتر خواهد بود. اگر همین روند را ادامه دهید خواهید دید که در طی نسل‌های متوالی دائماً جامعهٔ نمونهٔ ما باهوش و باهوش‌تر می‌شود. بدین ترتیب یک مکانیزم ساده طبیعی توانسته است در طی چند نسل عملاً افراد کم‌هوش را از جامعه حذف کند، علاوه بر اینکه میزان هوش متوسط جامعه نیز دائماً در حال افزایش است. بدین ترتیب می‌توان دید که طبیعت با بهره‌گیری از یک روش بسیار ساده (حذف تدریجی گونه‌های نامناسب و در عین حال تکثیر بالاتر گونه‌های بهینه)، توانسته است دائماً هر نسل را از لحاظ خصوصیّات مختلف ارتقاء بخشد. علاوه براین، در هر نسل جدید بعضی از خصوصیات به صورتی کاملاً تصادفی تغییر یافته و سپس در صورتی که این خصوصیت تصادفی شرایط طبیعت را ارضاء کند حفظ می‌شود در غیر این‌صورت به شکل اتوماتیک از چرخه طبیعت حذف می‌گردد. در واقع می‌توان تکامل طبیعی را بدین صورت خلاصه کرد: جستجوی کورکورانه (Blind Search) + بقای قوی‌تر.

در دهه ۷۰ میلادی دانشمندی از دانشگاه میشیگان به نام «جان هلند» ایده استفاده از الگوریتم ژنتیک را در بهینه‌سازی‌های مهندسی مطرح کرد. ایده اساسی این الگوریتم انتقال خصوصیات موروثی توسط ژن‌هاست. فرض کنید مجموعه خصوصیات انسان توسط کروموزوم‌های او به نسل بعدی منتقل می‌شوند. هر ژن در این کروموزوم‌ها نماینده یک خصوصیت است. حال اگر این کروموزوم به تمامی، به نسل بعد انتقال یابد، تمامی خصوصیات نسل بعدی شبیه به خصوصیات نسل قبل خواهد بود. اما در عمل دو اتفاق برای کروموزوم‌ها می‌افتد: (۱) در تولید مثل، ابتدا ترکیب اتفاق افتاده و ژن‌های والدین برای ایجاد کروموزوم‌های جدید ترکیب می‌شوند. (۲) سپس جنین تشکیل شده دچار جهش شده و بعضی ژن‌ها بصورت کاملاً تصادفی تغییر می‌کنند (مثلاً ژن رنگ چشم می‌تواند بصورت تصادفی باعث شود تا در نسل بعدی یک نفر دارای چشمان سبز باشد، در حالی که تمامی نسل قبل دارای چشم قهوه‌ای بوده‌اند). این همان چیزیست که باعث می‌شود تا فرزند تعدادی از خصوصیات پدر و نیز تعدادی از خصوصیات مادر را به ارث ببرد. میزان شایستگی (Fitness) موجود زنده نیز بواسطه بقای آن اندازه گیری می‌شود.

## ۲-۲- کلیات و مفاهیم مورد استفاده در الگوریتم ژنتیک

الگوریتم‌های ژنتیک یکی از اعضای خانوادهٔ مدل‌های محاسباتی الهام گرفته شده از روند تکامل‌اند. این الگوریتم‌ها راه‌حل‌های بالقوهٔ یک مسأله را در قالب کروموزوم‌های ساده‌ای کد کرده و سپس عملگرهای ترکیبی را بر روی این ساختارها اعمال می‌کنند. الگوریتم ژنتیک اغلب بعنوان روشی مبتنی بر جستجوی تصادفی برای بهینه‌سازی و تخمین پارامتر یکار می‌رود. اساس این الگوریتم قانون تکامل داروین است که می‌گوید: موجودات ضعیف‌تر از بین رفته و موجودات قوی‌تر باقی می‌مانند. در واقع، قانون انتخاب طبیعی برای بقا می‌گوید که هر چه امکان تطبیق موجود بیشتر باشد بقای موجود امکان‌پذیرتر است و احتمال تولید مثل بیشتری، برایش وجود دارد.

اولین و مهمترین نقطه قوت این الگوریتم‌ها این است که الگوریتم‌های ژنتیک ذاتاً موازیند. اکثر الگوریتم‌های دیگر موازی نبوده و فقط می‌توانند فضای مسأله مورد نظر را در یک جهت (در یک لحظه) جستجو کنند. اگر راه‌حل پیدا شده یک جواب بهینه محلی باشد، باید تمام کارهایی انجام شده را کنار گذاشته و دوباره از اول شروع کرد. از آنجایی که الگوریتم ژنتیک چندین نقطه شروع دارد، در یک لحظه می‌تواند فضای مسأله را از چند جهت مختلف جستجو کند. بدلیل موازی بودن و این که چندین رشته در یک لحظه مورد ارزیابی قرار می‌گیرند، الگوریتم‌های ژنتیک برای مسائلی که فضای جستجوی بزرگی دارند، بسیار مفیدند. از آنجایی که تعداد کمی از مسائل دنیای واقعی خطی هستند، الگوریتم ژنتیک بعنوان یک الگوریتم موازی، در حل مسائل واقعی بسیار کارآمدتر از روش‌های سنتی است. یکی از نقاط قوت الگوریتم‌های ژنتیک (شاید در ابتدا یک کمبود بنظر برسد)، عدم اطلاع الگوریتم از ماهیت مسائلی است که آنها را حل می‌کند. این الگوریتم تغییرات تصادفی در راه‌حل‌های کاندیدا می‌دهد و از تابع برازش برای سنجش پیشرفت تغییرات بهره می‌گیرد.

یکی دیگر از مزایای الگوریتم این است که می‌تواند چندین پارامتر را همزمان تغییر دهد. بسیاری از مسائل واقعی نمی‌توانند محدود به یک ویژگی شوند (تا آن ویژگی بهبود یابد) و باید چند جانبه در نظر گرفته شوند. ممکن است برای یک مسأله چندین راه حل پیدا شود (هر یک با یک معیار خاص، جواب کطلوب است).

الگوریتم ژنتیک بدلیل تقلید نمودن از طبیعت، دارای چند اختلاف اساسی با روش‌های جستجوی مرسوم است:

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

- الگوریتم ژنتیک، جهت جستجو انتخاب تصادفی انجام داده و به اطلاعات مشتق نیاز نداشته و فقط نیاز به اطلاعاتی در مورد کیفیت حل‌های ایجاد شده به وسیلهٔ هر مجموعه از متغیرها دارد. در صورتیکه در بعضی روش‌های بهینه‌سازی نیاز به شناخت کامل از ساختمان مسأله و متغیرها وجود دارد. از آنجاکه الگوریتم ژنتیک به چنین اطلاعاتی از مسأله نیاز ندارد، نسبت به بیشتر روش‌ها قابل انعطاف‌تر است.
- الگوریتم ژنتیک از قوانین احتمالی پیروی می‌کند و نه از قوانین قطعی. این الگوریتم مناسب‌ترین رشته‌ها را از میان اطلاعات تصادفی سازماندهی شده انتخاب می‌کند. با وجود تصادفی بودن، این الگوریتم در زمره الگوریتم‌های تصادفی ساده نبوده و بطور کارآمدی به اکتشاف اطلاعات گذشته در فضای جستجو می‌پردازد. مکانیزم تصادفی که روی انتخاب و حذف رشته‌ها عمل می‌کند، بگونه‌ایست که رشته‌هایی که عدد برازندگی بیشتری دارند، احتمال بیشتری برای ترکیب و تولید رشته‌های جدید داشته و در مرحله جایگزینی نسبت به دیگر رشته‌ها مقاوم‌ترند. بنابراین جمعیت رشته‌ها در رقابتی بر اساس تابع هدف، بهبود و متوسط برازندگی جمعیت افزایش می‌یابد.
- الگوریتم ژنتیک در هر تکرار چند نقطه از فضای جستجو را در نظر گرفته و بدام افتادن در ماکزیمم محلی را کاهش می‌دهد.
- قادر به بهینه‌سازی مسائل با تعداد متغیرهای زیاد است.
- توابع هزینه بسیار پیچیده نیز از این طریق قابل بهینه‌سازی بوده و احتمال بدام افتادن الگوریتم در اکسترمم‌های محلی زیاد نیست.
- الگوریتم توانایی کار کردن با داده‌های تجربی را علاوه بر توابع تحلیلی دارد.

در الگوریتم ژنتیک، مجموعهٔ متغیرهای طراحی توسط رشته‌هایی با طول ثابت (یا متغیر) کدکذاری می‌شود که آنها را کروموزوم یا فرد (Individual) می‌نامند. هر رشته یا کروموزوم یک نقطهٔ پاسخ در فضای جستجو را نشان می‌دهد. به ساختمان رشته‌ها یعنی مجموعه‌ای از پارامترها که توسط یک کروموزوم خاص نمایش داده می‌شود ژنوتیپ (Genotype) و به مقدار رمزگشایی آن فنوتیپ (Phenotype) می‌گویند. هر مرحلهٔ تکرار نیز نسل و مجموعه‌هایی از پاسخ‌ها در هر نسل را جمعیت می‌نامند.

الگوریتم ژنتیک با تولید نسل (Seeding) آغاز و جمعیت اولیه (Initial Population) را بطور انتخابی یا تصادفی ایجاد می‌کند. از آنجا که الگوریتم برای هدایت عملیات جستجو بطرف نقطهٔ بهینه، از روش‌های

آماری استفاده می‌کند، جمعیت موجود به تناسب برازندگی افراد آن، برای نسل بعد انتخاب می‌شود. سپس عملگرهای ژنتیکی (انتخاب، ترکیب و جهش) اعمال و جمعیت جدید ایجاد می‌شود و چرخه ادامه می‌یابد. معمولاً جمعیت جدید برازندگی بیشتری داشته و از نسلی به نسل دیگر بهبود می‌یابد. زمانی جستجو خاتمه می‌یابد که به حداکثر نسل ممکن رسیده و یا همگرایی (ارضاء معیارهای توقف) حاصل شده باشد.

جدول ۱، الگوریتم ژنتیک را با سیستم‌های طبیعی مقایسه نموده است.

**جدول ۱- مقایسه الگوریتم ژنتیک با سیستم‌های طبیعی**

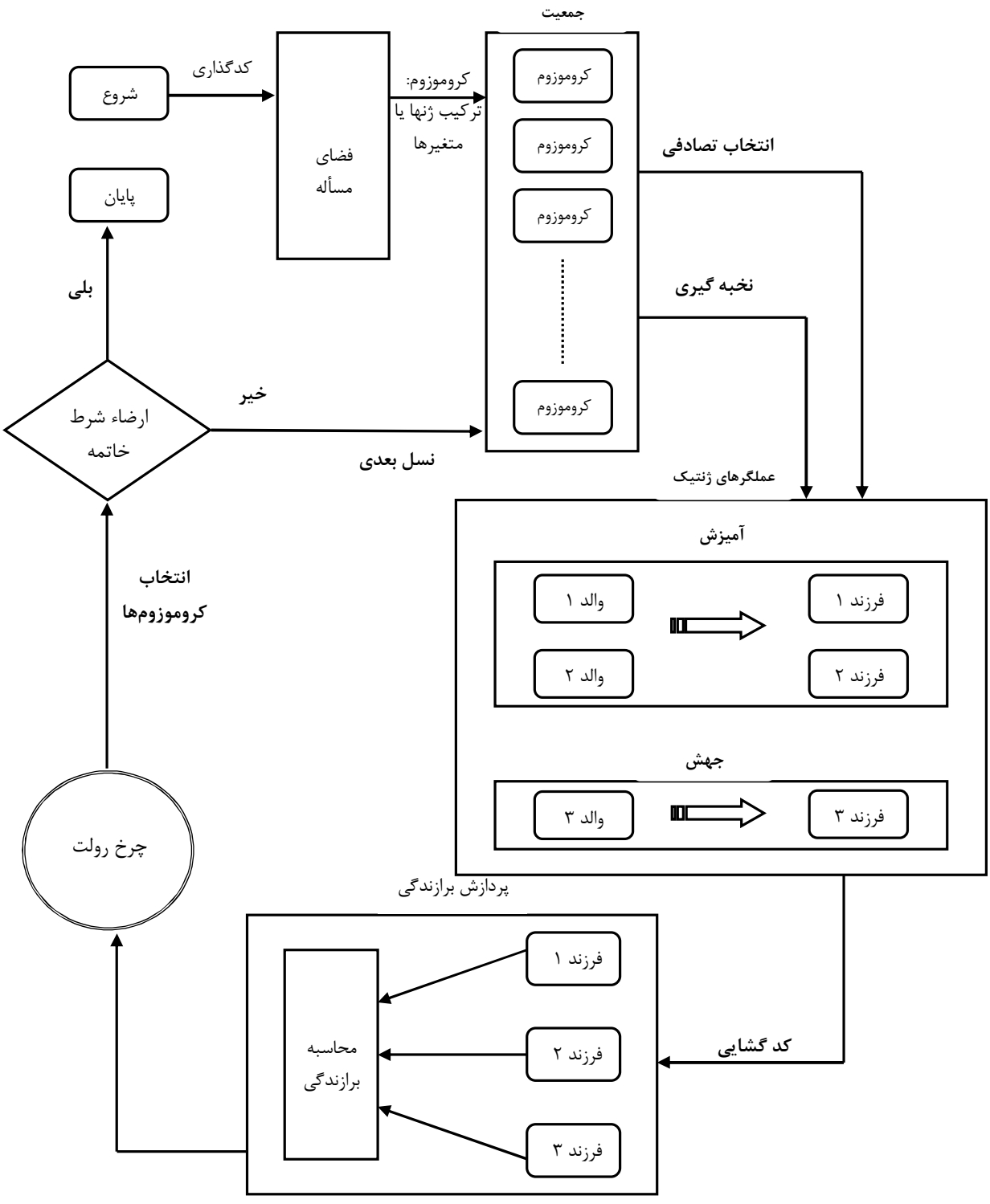
| | سیستم‌های طبیعی | الگوریتم ژنتیک |
|---|---|---|
| **کروموزوم:** | بسته‌های ژنی هستند که اطلاعات وراثتی را از نسلی به نسل دیگر عیناً انتقال می‌یابند. | پاسخ‌های ممکن مسأله به صورت رشته‌های عددی رمزگذاری شده است. |
| **ارزیابی:** | کروموزوم‌ها توسط شرایط محیطی که جمعیت در آن قرار دارد ارزیابی می‌شوند. | مساله به صورت یک رابطه ریاضی در آمده و کروموزوم‌ها توسط تابع برازش ارزیابی می‌شوند. |
| **تکثیر:** | معیار بقای موجود زنده و تکثیر آن، سازش با محیط است. | متناسب با مقدار تابع برازش، رشته‌های جمعیت جدید انتخاب می‌شوند. |
| **آمیزش:** | والدین ژن‌هایشان را در زایش فرزندان مبادله می‌کنند. | رشته‌های جمعیت به صورت دو به دو ترکیب می‌شوند و ژن‌ها یا متغیرهایشان را تعویض می‌کنند. |
| **جهش:** | تغییر ناگهانی و تصادفی ژن‌های فرزندان در طول زنجیره DNA | یک بیت یا خانه از رشته عددی به صورت تصادفی انتخاب و بصورت تصادفی دچار تغییر می‌گردد. |
| **تکامل:** | ایجاد نسل‌های جدید و تکامل موجودات | تکرار مراحل الگوریتم و بهبود برازش رشته‌ها |

## عملگرهای الگوریتم ژنتیک

- کدگذاری (**Coding**) و کدگشایی (**Decoding**): این‌ها شاید مشکلترین مراحل حل مسأله باشند. الگوریتم ژنتیک بجای کار بر روی پارامترها یا متغیرهای مسأله، با شکل کد شدهٔ آنها سروکار دارد.
- ارزیابی (**Evaluation**): برازندگی هر کروموزوم با معیار تابع هدف (تابعی که باید بهینه شود) ارزیابی می‌شود. این تابع هر رشته را با یک مقدار عددی ارزیابی می‌کند (هرچه بیشتر، کیفیت جواب بالاتر).
- آمیزش یا ترکیب (**Crossover**): مهمترین فرآیند در الگوریتم ژنتیک است که در آن نسل قدیمی کروموزوم‌ها با یکدیگر مخلوط و نسل تازه‌ای بوجود می‌آورند. جفت‌هایی که در قسمت انتخاب بعنوان والد برگزیده شده‌اند، ژن‌هایشان را با هم مبادله و اعضای جدید را تولید می‌کنند. ترکیب باعث از بین رفتن تنوع ژنتیکی جمعیت می‌شود (زیرا اجازه ترکیب ژن‌های خوب را می‌دهد).
- جهش (**Mutation**): بعد از ایجاد یک عضو در جمعیت جدید، هر ژن آن با احتمالی مشخص، جهش می‌یابد. در جهش ممکن است ژنی از مجموعه ژن‌های جمعیت حذف یا بدآن اضافه شود. جهش یک ژن به معنای تغییر آن ژن متناسب با روش کدگذاری است.

## ۲-۳- نحوهٔ کارکرد الگوریتم ژنتیک

در حالت کلی الگوریتم ژنتیک چرخه‌ای تکراری را طی می‌کند (شکل ۱): ابتدا جمعیتی اولیه از افراد بطور اتفاقی و بدون در نظرگرفتن معیار خاصی تولید می‌شود.

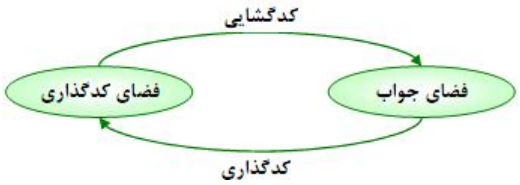



**شکل ۱- فلوچارت الگوریتم ژنتیک**

برای تمامی کروموزوم‌های (افراد) نسل صفر، مقدار برازش با توجه به تابع برازش (بخش ۲-۸) تعیین می‌شود. سپس با کمک مکانیزم‌های انتخاب (بخش ۲-۹)، زیرمجموعه‌ای از جمعیت اولیه انتخاب و روی آنها عملیات ترکیب و جهش اعمال می‌شود. در نهایت فرزندان با افراد جمعیت اولیه (نسل پیشین) از لحاظ مقدار برازش مقایسه می‌شوند؛ در هرحال افرادی باقی خواهند ماند که بیشترین مقدار برازش را داشته باشند. با درنظر گرفتن نسل جدید بعنوان جمعیت اولیه برای مرحلهٔ بعد، مراحل تکرار می‌شوند.

## ۲-۴- روش‌های کدگذاری (کدینگ)

الگوریتم ژنتیک بجای کار بر روی متغیرهای مسأله، با شکل کد شده آنها سروکار دارد. یکی از ویژگی‌های اصلی الگوریتم ژنتیک کار متناوب آن بر روی فضای کدینگ و فضای جواب است (شکل ۱).




**شکل ۱- ارتباط فضای کد گذاری و فضای جواب**

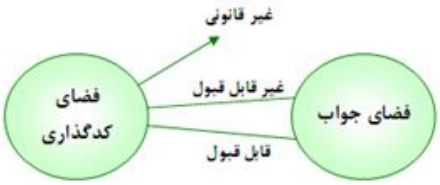
اعمال ژنتیکی بر روی فضای کدینگ (یا کروموزوم‌ها) اعمال می‌شوند، در حالی که انتخاب و ارزیابی بر روی فضای جواب عمل می‌نماید. در کد گذاری سه موضوع بسیار مهم مطرح است که عبارتند از :

- قابل قبول بودن (feasible) کروموزوم
- قانونی بودن (legality) کروموزوم
- یکانی بودن رابطه

قابل قبول بودن یک کروموزوم به این مفهوم است که آیا کدگشایی آن در ناحیه قابل قبول (جزئی از فضای جواب مساله) قرار دارد یا خیر؟ غیر قابل قبول بودن یک کروموزوم ناشی از طبیعت مسایل بهینه سازی با محدودیت می باشد. به طور معمول در مسایل بهینه سازی فضای قابل قبول به وسیله یک سری معادلات/ نامعادلات مشخص می‌شود. در چنین مواردی روش‌های جریمه به منظور جلوگیری از ایجاد کروموزوم‌های غیر قابل قبول پیشنهاد شده اند. در مسایل بهینه سازی با محدودیت به طور معمول جواب در مرز بین فضای قابل قبول و فضای غیر قابل قبول قرار دارد؛ در نتیجه روش‌های جریمه، کمک می‌کنند که الگوریتم ژنتیک از هر دو طرف به سمت جواب حرکت کند.

قانونی بودن یک کروموزوم به این مفهوم است که آیا کروموزوم در اثر کدگشایی در فضای جواب قرار خواهد گرفت یا خیر؟ غیر قانونی بودن کروموزوم ناشی از طبیعت روش‌های کدگذاری می باشد. در چنین مواردی

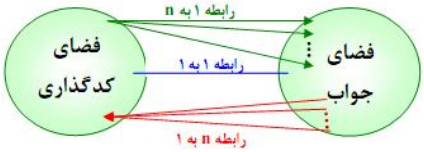
به طور معمول از روش های ترمیم به منظور تبدیل کروموزوم های غیر قانونی به کروموزوم های قانونی استفاده می کنند. این موارد در شکل ۲ نیز نشان داده شده است.




**شکل ۲- قانونی و قابل قبول بودن جواب‌هادر کدگشایی**

مساله سوم، بحث یکانی رابطه بین فضای کدگذاری و فضای جواب است که می‌تواند به یکی از سه صورت زیر باشد (شکل ۳):

- رابطه یک به یک (بهترین)
- رابطه n به یک
- رابطه n به n (بدترین)




**شکل ۳- یکانی بودن در ارتباط فضاهای جواب و کدگذاری**

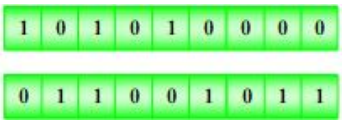
باتوجه به ماهیت مسأله، روش‌های کدگذاری متفاوتی استفاده می‌شوند:

- کدگذاری باینری (Binary Coding)
- کدگذاری جایگشتی (Permutation Coding)
- کدگذاری مقدار (Value Coding)
- کدگذاری درختی (Tree Coding)

نمایش مناسب رشته‌ها به ویژگی‌های فضای جستجو بستگی دارد، ولی معمولاً بصورت رشته‌های باینری نمایش داده می‌شوند.

## کدگذاری باینری (دودویی):

یکی از روش‌های کدینگ، کدگذاری دودویی (تبدیل کروموزوم یا جواب مسأله به رشته‌ای از اعداد باینری) است که در آن اعضای جمعیت به رشته‌هایی از ۰ و ۱ تبدیل می‌شوند. تعداد بیت‌هایی که برای کد گذاری متغیرها استفاده می‌شود، به دقت مورد نظر برای جواب‌ها، محدوده تغییر پارامترها و رابطه بین متغیرها وابسته است. در این نوع کدگذاری، هر بیت معادل یک ژن می‌باشد. این کدگذاری که کدینگ استاندارد در الگوریتم‌های ژنتیک است، ساده‌ترین و بهترین تبدیل برای عملگرهای ژنتیک می‌باشد. از آنجا که در این کدگذاری طول کروموزوم‌ها، می‌تواند بسیار بزرگ می‌شود، در مسائل پیچیده چندان مناسب نیست.




**شکل ۴- نمونه‌ای از کدگذاری باینری**

برای حل یک مسأله بهینه‌سازی بکمک الگوریتم ژنتیک، ابتدا متغیرهای مستقل مسأله و دامنهٔ تغییرات آنها معین می‌شود. در حالت کلی متغیرها می‌توانند بصورت پیوسته یا گسسته نمایش داده شوند. بعنوان مثال فرض کنید الگوریتم می‌خواهد ماکزیمم تابعی سه متغیره را پیدا کند؛ هر پاسخ ممکن شامل سه عدد X، Y و Z می‌باشد. با فرض اینکه جستجو در محدودهٔ اعداد صحیح مثبت ۰ تا ۲۵۵ انجام شود، طول هر عدد در تبدیل باینری حداکثر ۸ بیت می‌باشد. اگر هر کروموزوم را بصورت XYZ در نظر بگیریم، برای پوشش دادن تمام پاسخ‌های ممکن، لازم است طول کروموزوم ۲۴=۸×۳ بیت باشد. در همین مسأله، اگر لازم باشد اعداد منفی نیز جستجو شوند، باید یک بیت علامت به ابتدای هر رشته اضافه شود (۰ به معنی عدد مثبت و ۱به معنی عدد منفی). توجه شود که برای اعداد اعشاری نیز می‌بایست چنین تمهیداتی انجام شود.

معمولاً در مسائل طراحی، استفاده از متغیرهای گسسته اجتناب ناپذیر است (جنس مواد و ابعاد استاندارد آنها گسسته‌اند). مثلاً اگر متغیری دارای ۸ مقدار باشد، برای بیان تمام حالات به یک زیر رشته سه خانه و اگر دارای۱۰ مقدار باشد، به یک زیر رشته حداقل ۴ خانه نیاز است. ولی زیر رشته‌ای با ۴ خانه توانایی ۲۴ مقدار متفاوت را دارد. برای مثال فرض کنید در یک مسأله طراحی جنس قطعه مورد بررسی به عنوان متغیر بهینه‌سازی از میان جنس‌های آهن، چدن و برنج قابل انتخاب باشد؛ این متغیر را می‌توان بصورت زیر کد گذاری کرد: آهن=۰۰، برنج=۰۱، چدن=۱۰، آهن=۱۱. همانگونه که نشان داده شده، برای فضای چهارم بصورت تصادفی یکی از سه جنس قابل قبول انتخاب شده است. پس از اینکه تمام متغیرهای طراحی بصورت زیر دسته‌هایی در نظر گرفته شدند، با کنار هم قرار دادن این زیر رشته‌ها، به یک رشته دودوئی از اعداد خواهیم رسید. این رشته از اعداد (همان کروموزوم) معرف یک طرح می‌باشد.

یکی از بخش‌های مهم الگوریتم ژنتیک، رمز گشایی است که در مرحلهٔ ارزیابی انجام می‌شود. در روند اجرای الگوریتم ژنتیک لازم است رشته‌ها به متغیرها تبدیل شده و با قرار دادن مقدار متغیر در تابع هدف، برازش آن رشته بدست آید. برای باز گرداندن هر رشته به فضای واقعی (فضای متغییرها)، باید تعداد بیت‌های مربوط به تک تک متغیرها، نوع متغیرها (پیوسته یا گسسته) و محل هر متغیر در رشته مشخص باشند.

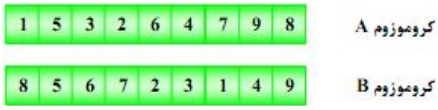
برای هر متغیر پیوسته، ابتدا زیر رشته مربوطه مشخص شده و سپس مقدار واقعی (دهدهی) آن بدست می‌آید. اگر متغیر X در فاصله ۰ تا $2^n-1$ (n: تعداد بیت‌ههای اختصاص یافته به متغیر) نگاشت شده باشد، مقدار دهدهی آن از $X_{decimal}=X_{bainary}\times(X_{max}-X_{min})/(2^n-1)+X_{min}$ بدست می‌آید. در متغیرها ناپیوسته،

بجای رمز کردن متغیرها، اندیس‌های مربوط به آنها رمز می‌شوند. بعبارتی مقدار عددی زیر رشته متناظر با آن در واقع نمایانگر اندیس درایه‌ای از بردار کدگذاری است (مقدار آن درایه همان مقدار متغیر است).

## کدگذاری جایگشتی (جهشی):

این نوع کدگذاری می‌تواند در مسایل ترتیبی نظیر مساله فروشنده دوره گرد یا مسأله زمانبندی کارها بکار رود. در این روش، کروموزوم‌ها به صورت رشته‌ای از اعداد طبیعی نشان داده می‌شوند که هرکدام از این اعداد، مربوط به پارامتر ویژه‌ای در فضای حل مسأله است. ترتیب قرارگیری این اعداد مهم بوده و طول رشته دقیقاً با تعداد پارامترهای تعریف شده در مسأله برابر است.

مثال: مسأله «فروشنده دوره گرد»- تعدادی شهر داریم که فاصله میان آنها معلوم است؛ با شروع از یک شهر و ختم به همان شهر می‌بایست از تمام شهرها فقط و فقط یکبار عبور نموده و کمترین مسافت طی شود.




**شکل ۵- مثالی از کدگذاری جایگشتی (جهشی)**

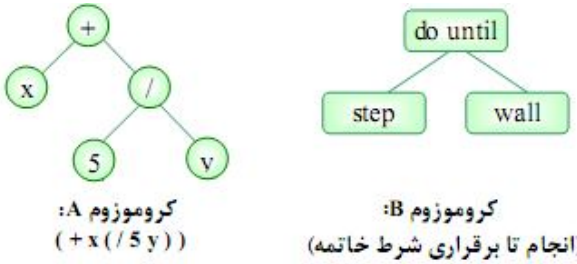
در این مسأله با جایگشت‌های مختلفی از مجموعه راه‌حل‌ها رو به رو هستیم. نکته‌ای که باعث شده تا کدگذاری باینری روش مناسبی برای این مسأله نباشد، این است که حتما باید برش میان دو والد به نحوی صورت بگیرد که هیچ عنصری تکراری وجود نداشته باشد.

## کد گذاری ارزشی (مقدار):

در این نوع روش کدگذاری، کروموزوم‌ها می‌توانند هر نوع داده مرتبط با مسأله را در خود جای دهند. این داده‌ها ممکن است از نوع اعداد حقیقی، عبارات منطقی، دستورات جهت‌یابی، رشته‌های حرفی و... باشند.

## کدگذاری درختی:

این روش که توسط «جان کوزا» توسعه یافت، بیشتر برای برنامه‌نویسی تکاملی توسط نرم‌افزار «لیسپ» مورد استفاده قرار می‌گیرد (بدلیل آنکه برنامه های آن به این فرم نمایش داده می شوند). در کد گذاری درختی، هر کروموزوم یک درخت از اشیایی نظیر توابع یا دستورها در زبان برنامه نویسی می باشد (شکل ۶).




**شکل ۶- دو مثال از کدگذاری درختی**

## ۲-۵- جمعیت

مفهوم جمعیت در الگوریتم ژنتیک شبیه به چیزی است که در زندگی طبیعی وجود دارد. برای هر مسأله مجموعه انتخاب‌هایی بعنوان پاسخ (درست یا غلط) وجود دارند که به آنها پاسخ‌های ممکن یا شدنی (فضای جستجو) گویند. مثلاً اگر مسأله یافتن ماکزیمم یک تابع در مجموعه اعداد صحیح باشد، مجموعه تمام اعداد صحیح می‌توانند بعنوان فضای جستجوی مسأله در نظر گرفته شوند. در الگوریتم ژنتیک پس از تعیین سیستم کدینگ و مشخص شدن روش تبدیل هر جواب به کروموزوم، بعنوان اولین مرحله لازم است مجموعه‌ای از جواب‌های ممکن (کروموزوم‌ها) بعنوان جمعیت اولیه ایجاد شود. اعضای این مجموعه عموماً بصورت تصادفی انتخاب می‌شوند (معمولاً از قیدهایی برای جلوگیری از پراکندگی بیش از حد جمعیت استفاده می‌شود). اما گاهی اوقات برای بالا بردن سرعت و کیفیت الگوریتم از روش‌های ابتکاری نیز برای تولید جمعیت اولیه استفاده می‌گردد. تعداد اعضای جمعیت به نوع مسأله بستگی دارد؛ در واقع با تغییر تعداد اعضا می‌توان دقت جواب‌ها و سرعت همگرایی جستجو را بهبود بخشید. اندازهٔ جمعیت اولیه به سایز رشتهٔ کد شده نیز وابسته است؛ بعنوان مثال جمعیت در یک مسأله با کروموزوم‌های ۳۲ بیتی باید بیشتر از حالتی با کروموزوم‌های ۱۶ بیتی است.

اگر تعداد کروموزوم‌ها بسیار کم باشد، الگوریتم ژنتیک امکان انجام عمل ترکیب کمتری خواهد داشت و تنها بخشی کوچک از فضای جستجو کشف خواهد شد (ممکن است جستجو برای رسیدن به حل بهینه موفقیت‌آمیز نبوده یا مستلزم صرف زمان زیادی باشد). از طرفی دیگر، اگر تعداد کروموزوم‌ها بسیار زیاد باشد، روند الگوریتم ژنتیک کند خواهد بود. بررسی‌ها نشان داده اند که بدلیل برخی محدودیت‌ها، استفاده از جمعیت زیاد ثمر بخش نخواهد بود. در عمل تعداد اعضای جمعیت بصورت تجربی بدست می‌آید.

## ۲-۶- تابع هدف

یعنی، تابع هدف، شاخصی از نحوهٔ عملکرد افراد در فضای مسأله بوده و هدف ما را در بهینه سازی مشخص می‌کند. در الگوریتم ژنتیک و الگوریتم‌های مشابه، عموماً تابع برازندگی یا برازش (fitness function) بجای تابع هدف معرفی می‌شود. یک مشکل چگونگی نوشتن تابع برازش است که منجر به بهترین راه حل برای مسأله شود. اگر این تابع درست انتخاب نشود، ممکن است راه‌حلی برای مسأله پیدا نشده و یا مسأله‌ای دیگر به اشتباه حل شود.

تابع برازندگی از اعمال تبدیل مناسب بر روی تابع هدف یعنی تابعی که قرار است بهینه شود بدست می‌آید. این تابع هر رشته را با یک مقدار عددی ارزیابی می‌کند که کیفیت آن را مشخص می‌نماید. هر چه کیفیت رشته جواب بالاتر باشد مقدار برازندگی جواب (مقدار محاسبه شده توسط تابع برازش) بیشتر است. بسته به اینکه مسأله مورد نظر بیشینه‌سازی یا کمینه‌سازی باشد برازندگی بیشتر مترادف با بیشینه یا کمینه بودن تابع هدف خواهد بود. از آنجایی که الگوریتم ژنتیک بدنبال بیشینه تابع است، در مسائل بیشینه‌سازی، تابع برازندگی برابر با تابع هدف انتخاب می‌شود؛ با اینحال مسائل کمینه‌سازی می‌بایست به بیشینه‌سازی تبدیل شوند. برای تبدیل مسائل کمینه‌سازی به بیشینه‌سازی روش‌های مختلفی وجود دارد ($\varphi_i$: مقدار تابع هدف و $f_i$: مقدار تابع برازش برای فرد iام می‌باشند):

* روش اول- ساده‌ترین راه، کم کردن $\varphi_i$ از یک مقدار ثابت $C$ است (به ازای تمامی جواب‌های ممکن بزرگتر باید $C$ از $\varphi_i$ باشد): $f_i = C - \varphi_i$

نکته: اگر نتوان بزرگترین مقدار تابع هدف را حدس زد، در هر نسل $C= \varphi_{max+} \varphi_{max}$ در نظر گرفته می‌شود.

* روش دوم- راه دیگر استفاده از تابع نمائی است: $f_i = e^{-\varphi_i}$

* روش سوم- روشی دیگر نیز معکوس کردن تابع نمائی است: $f_i = \frac{1}{\varphi_i}$

## مسائل بهینه‌سازی با قید:

معمولاً مسائل بهینه‌سازی دارای قیدهایی هستند که هنگام حل مسأله باید ارضاء شوند. در صورتیکه قیدها کم یا ساده باشند و احتمال ارضاء شدن آنها به خودی خود زیاد باشد، می‌توان در هر نسل افرادی را که قیدها را ارضاء نمی‌کنند با افراد جدید (به طور تصادفی) جایگزین کرد. ولی در شرایط پیچیده قیدها می‌توانند بصورت تابع جریمه در تابع هدف منظور شوند ($\varphi_i^{new} = w_i P_i + \varphi_i$). علامت و مقدار $P_i$ باید به گونه‌ای باشد که موجب شود افرادی از هر نسل که کمتر قیدها را ارضاء می‌کنند، در نهایت از برازندگی کمتری بر خوردار باشند. دقت شود که ضریب وزن $w_i$ میزان تأثیر قیود در تابع هدف را مشخص می‌کنند و مقدار آن بصورت تجربی برگزیده می‌شود. برای دستیابی به یک مسئله بیشینه‌سازی، تابع برازش می‌تواند با جایگزینی $\varphi_i^{\text{new}}$ بجای $\varphi_i$و به یکی از سه روش بالا محاسبه شود.

## ۲-۷- انتخاب (Selection) و تولید (Reproduction) نسل جدید

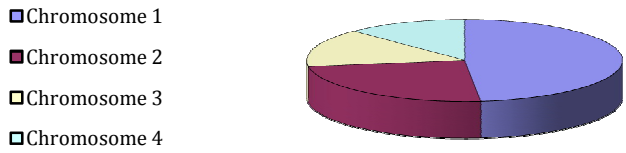
عملگر انتخاب رابط بین دو نسل است و بعضی از اعضای نسل کنونی را برای تولید نسل جدید انتخاب می‌کند. پس از اعمال عملگرهای ژنتیک، می‌بایست جمعیتی جدید از رشته‌ها برای دور بعدی الگوریتم تعیین شود. این مجموعه انتخابی از کلیه کروموزوم‌های فرزند ( حاصل از عملگرهای ترکیب و جهش) و تعدادی از کروموزوم‌های مرحله قبل خواهد بود. این مرحله را تولید یا تکثیر گویند. بعد از انتخاب اعضای نسل جدید، مجدداً عملگرهای ژنتیک بر روی اعضای برگزیده اعمال می‌شوند. معیار انتخاب اعضاء، ارزش تطابق آنهاست، بااینحال روند انتخاب حالتی تصادفی دارد.

شاید انتخاب مستقیم و ترتیبی (بدین شکل که بهترین اعضاء انتخاب شوند) در نگاه اول روش مناسبی به نظر برسد. اما در الگوریتم ژنتیک ما با ژن‌ها روبه‌رو هستیم، یک عضو با تطابق پایین اگرچه در نسل خودش عضو مناسبی نمی‌باشد اما ممکن است شامل ژن‌های خوب باشد و اگر شانس انتخاب شدنش صفر باشد، این ژن‌های خوب نمی‌توانند به نسل‌های بعد منتقل شوند. پس روش انتخاب باید به گونه‌ای باشد که به این عضو نیز شانس انتخاب شدن داده شود. راه‌حل مناسب، طراحی روش انتخاب به گونه‌ای است که احتمال انتخاب شدن اعضای با تطابق بالاتر بیشتر باشد. انتخاب باید به گونه‌ای صورت بگیرد که تا جایی که ممکن است هر نسل جدید نسبت به نسل قبلی‌اش میانگین برازش بهتری داشته باشد. روش‌های متداول انتخاب عبارتند از:

- انتخاب چرخ رولت
- انتخاب ترتیبی
- انتخاب بولتزمن
- انتخاب حالت پایدار
- نخبه سالاری
- انتخاب رقابتی
- انتخاب قطع سر
- انتخاب قطعی بریندل
- انتخاب جایگزینی نسلی اصلاح شده
- انتخاب مسابقه

### ❋ انتخاب چرخ رولت:

انتخاب چرخ رولت که اولین بار توسط هولند پیشنهاد شد، یکی از مناسب‌ترین انتخاب‌های تصادفی بوده که از ایدهٔ احتمال انتخاب اقتباس شده است. روش پیاده‌سازی چرخ رولت به این شکل است که دایره‌ای در نظر گرفته و آن را متناسب با تعداد کروموزوم‌ها بصورت شکل ۷ تقسیم می‌کنند.؛ هر بخش متناظر با برازندگی کروموزومی می‌باشد. با چرخاندن چرخ، در هر بخش که متوقف شد، کروموزوم متناظر با آن انتخاب می‌گردد. در ادامه چرخ رولت به دو روش پیاده می‌شود تا تعیین کند کدام اعضاء شانس باز تولید را دارند.




**شکل ۷- بازه‌های متناظر با هر کروموزوم در چرخ رولت**

احتمال انتخاب و بقای هر کروموزوم، براساس برازندگی آن محاسبه می‌شود ($f_k$ برازندگی کروموزوم $k$):

$$P_k = \frac{f_k}{\sum_{i=1}^{n} f_i} \qquad (۱)$$

روش اول: کروموزوم‌ها را براساس $P_k$ مرتب کرده و احتمال تجمعی انتخاب را بصورت زیر بدست می‌آوریم:

$$q_k = \sum_{i=1}^{k} P_i \qquad (۲)$$

با توجه به رابطهٔ (۲)، هر عضو به نسبت تطابقش، تعدادی از بخش‌های چرخ رولت را به خود اختصاص می‌دهد.. برای پیاده‌سازی چرخ رولت، یک عدد تصادفی بین ۰ تا ۱ تولید و اگر عدد در بازه $q_{k-1}$ تا $q_k$ قرار داشت، کروموزوم $k$ام انتخاب می‌شود. روند انتخاب تا جایی تکرار می‌شود که به اندازه کافی عضو برای تشکیل نسل بعد وجود داشته باشد.

روش دوم: برداری m عنصری مانند V در نظر بگیرید، که هر عضو به نسبت برازش ($f_i$) یا احتمال انتخابش ($P_k$)، در بردار $V$ تکرار می‌شود؛ هر چه $f_i$ بیشتر باشد، عضو مکان‌های بیشتری را به خود اختصاص می‌دهد. بعد از تشکیل این بردار، یک مقدار تصادفی $1 \le r \le m$ انتخاب می‌شود؛ این مقدار به مکانی در بردار اشاره می‌کند که آن مکان خود معرف عضوی از اعضای جمعیت است.

بعنوان مثال اگر جمعیتی ۴ عنصری 4 داشته باشیم و تطابق یا برازش اعضای آن $f_1 = 10, f_2 = 10, f_3 = 15, f_4 = 25$ باشند، مقدار مجموع تطابق‌ها عبارت است از $f_i = 60$. بردار $V$ را می‌توان ۶۰ عنصری در نظر گرفته و به عضو ۱، ده مکان، به عضو ۲، ده مکان، به عضو ۳، پانزده مکان و به عضو ۴، بیست و پنج مکان اختصاص داد: $V = \{1,..,1,2,..,2,3,..,3,4,..,4\}$. سپس r بین ۱ تا ۶۰ و به تصادف انتخاب می‌شود. فرض کنید r=32، در اینصورت $V[32] = 3$ و عضو ۳ انتخاب می شود.

### ❋ انتخاب قطعی

در این روش که توسط برینـدل معرفی شده، احتمال انتخاب برای هر کروموزوم طبق رابطهٔ (۱) و تعداد مورد انتظار برای آن بصورت زیر محاسبه می‌شود (n: تعداد اعضای جمعیت و $f_{mean}$ برازش متوسط جمعیت).

$$e_k = P_k \times n = \frac{f_k}{\sum_{i=1}^{n} f_i} \times n = \frac{f_k}{\frac{1}{n}\sum_{i=1}^{n} f_i} = \frac{f_k}{f_{mean}} \qquad (۳)$$

بعد از مرتب سازی کروموزوم‌ها بر اساس برازش یا $e_k$، هر کروموزوم به تعداد $e_k$ (قسمت صحیح) کپی شده و نهایتاً به تعداد لازم جهت تکمیل جمعیت، بترتیب از بالای لیست برداشته می‌شود.

* انتخاب حالت پایدار (**Steady State Selection**):

در اکثر الگوریتم‌های ژنتیک که در مقالات ارائه شده‌اند، جمعیت جدید به طور کامل توسط فرزندان بوجود می‌آید و این فرزندان جایگزین والدین خود می‌شوند. در بعضی روش‌ها به برخی از اعضای جمعیت قدیمی، اجازه حضور در جمعیت جدید، داده می‌شود. انتخاب حالت پایدار یکی از این روش‌هاست. در این روش، فقط تعداد اندکی از اعضای جمعیت کنونی، با اعضای جدید جایگزین می‌شوند، به عبارت دیگر، بدترین اعضا با فرزندانی که از بهترین اعضا بوجود آمده‌اند تعویض می‌شوند اما بافت کلّی جمعیت، چندان تغییر نمی‌کند.

* انتخاب نخبه گرایی (**Elitist Selection**):

ایدهٔ نخبه سالاری یا نخبه گرایی، ویژگی تازه‌ای به پروسهٔ انتخاب اضافه می‌کند، در نخبه سالاری، بهترین عضو هر جمعیت، زنده می‌ماند و در جمعیت بعد حضور دارد، به عبارت دیگر عضوی که بالاترین تطابق را دارد به طور خودکار به جمعیت جدید منتقل می‌شود. این روش ابتدا در سال ۱۹۷۵ توسط کندی جونز معرفی شد. اعمال نخبه سالاری در الگوریتم ژنتیک، معمولاً باعث بهبود کارایی آن می‌شود، با اینحال در برخی موارد ممکن است سبب هم شکل شدن جمعیت‌ها و در نتیجه عدم همگرایی مسئله شود.

* انتخاب رقابتی (**Rivalry Selection**):

این روش تعدادی از اعضای جمعیت را به تصادف انتخاب کرده و اگر شرطی خاص برقرار باشد، بهترین یا تعدادی از بهترین‌های آنها را به عنوان والد بر می‌گزیند؛ در غیر اینصورت بدترین عضو یا تعدای از بدترین‌ها، در تشکیل جمعیت آینده به عنوان والد در نظر گرفته می‌شوند. شکل استاندارد این روش، رقابت دوتایی یا باینری است و بشکل زیر توصیف می‌شود:

- دو عضو به تصادف انتخاب می‌شوند.
- مقدار $r$ بصورت تصادفی بین ۰ و ۱ تعین می‌شود.
- پارامتر $0 \leq K \leq 1$ توسط کاربر تعیین می‌شود (مثلاً $K$=۰/۷۵).

- اگر $r < k$ برترین و اگر $r \leq K$ بدترین این دو عضو، به عنوان والد انتخاب می‌شود.

نکته: دو عضو انتخاب شده برای رقابت، به جمعیت بر می‌گردند و می‌توانند دوباره در رقابت شرکت کنند.

نکته: روش انتخاب رقابتی می‌تواند به صورت رقابت n تایی نیز انجام شود.

* انتخاب قطع سر:

در این روش که توسط گلدنبرگ معرفی شده، $T$(یک عدد کوچکتر از صد) درصد برتر کروموزوم‌های مرتب شده بر مبنای برازندگی، انتخاب می‌شوند. سپس از هر یک از آنها (100/T) کپی به نسل بعد انتقال داده می‌شود. مثلاً به ازای تعداد ۲۰ کروموزوم و $T$=20، ۲۰٪ (۴ تای) اول لیست کروموزوم‌های مرتب شده، انتخاب و از هر کدام (۱۰۰/۲۰) یا ۵ کپی ایجاد می‌کنیم.

* انتخاب جایگزینی نسلی اصلاح شده:

در این روش که توسط وایتلی ارائه شده، ابتدا کروموزوم‌های جمعیت براساس مقدار برازش مرتب شده و سپس فرزندان تولیدی جایگزین همان تعداد کروموزوم از انتهای لیست والدین می‌شوند.

* انتخاب مسابقه (Tournament Selection):

این روش که توسط گلدبرگ ارائه شده، برای کاهش احتمال هم گرایی زودرس پیشنهاد شده است. به ازای هر فرد جمعیت در هر نسل، مجموعه‌ای (مجموعه مسابقه که تعداد اعضای آن اندازه مسابقه گفته می‌شود. اندازه مسابقه معمولاً برابر ۲ فرض می‌شود) تصادفی تولید کرده و بهترین آن‌ها بعنوان یک عضو از نسل جدید انتخاب می‌شود.

در روش دیگری که توسط وزل ارائه شده، بجای تولید تصادفی مجموعه مسابقه، اعضای آن با کمک چرخ رولت ایجاد و بهترین عضو آن بعنوان بخشی از نسل جدید در نظر گرفته می‌شود.

* روش رتبه بندی:

زمانی که مقادیر برازندگی اختلاف زیادی دارند، روش های انتخاب قبلی دارای مشکلاتی خواهد بود. برای مثال اگر شایستگی بهترین کروموزوم در روش های قبلی ۹۰٪ باشد در این صورت کروموزوم‌های دیگر شانس خیلی کمتری برای انتخاب خواهند داشت. روش رتبه‌بندی، در ابتدا جمعیت ژنتیکی را دسته بندی کرده و به هر کروموزوم یک رتبه برازندگی در دسته بندی می‌دهد (بدترین حالت رتبه ۱، بعد ۲،... و بهترین

رتبه برابر n). در این صورت تمامی کروموزوم‌ها شانسی برای انتخاب خواهند داشت. البته این ممکن است روش به همگرایی کند منجر شود (زیرا بهترین کروموزوم‌ها اختلاف زیادی با هم نخواهند داشت).

## ۲-۸- آمیزش، ترکیب یا جابجایی (Crossover)

در طبیعت بقای نسل یکی از مهمترین فاکتورهاست و تنها عملگر ممکن برای این امر آمیزش است. مهمترین عملگر در الگوریتم‌های ژنتیک، عملگر آمیزش یا **ترکیب** است. آمیزش یا ترکیب، فرآیندی است که در آن نسل قدیمی کروموزوم‌ها با یکدیگر مخلوط و ترکیب می‌شوند تا نسل تازه‌ای از کروموزوم‌ها بوجود بیاید. جفت‌هایی که در قسمت انتخاب، به عنوان والد در نظر گرفته شدند، در این قسمت ژن‌هایشان را با هم مبادله می‌کنند و هر کدام از کروموزوم‌ها خصوصیاتی از خود را به فرزندان انتقال می‌دهند. بدیهی است کروموزوم‌هایی که دارای برازندگی بیشتری هستند می‌بایست شانس بیشتری برای آمیزش داشته باشند.

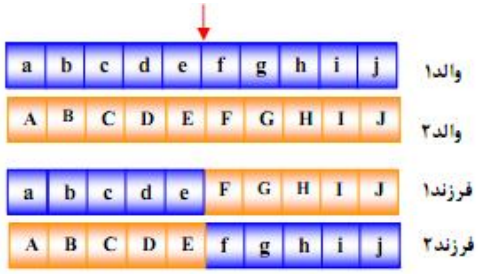
والدینی با برازش یا تطابق بالا باعث بوجود آمدن فرزندانی با تطابق بیشتر می‌شوند. از آنجا که ترکیب اجازه می‌دهد که ژن‌های خوب یکدیگر را بیابند، این عملگر در الگوریتم ژنتیک سبب از بین رفتن تنوع ژنتیکی جمعیت می‌گردد.

ترکیب با این امید صورت می‌گیرد که کروموزوم‌های جدید، حاوی بخش‌های مناسب کروموزوم‌های قدیمی باشند و در نتیجه نسل بهتری تولید شود، با این حال خوب است که برخی از قسمت‌های نسل قدیم برای نسل بعدی باقی بمانند. بنابراین لازم نیست که ترکیب در تمام والدین هر نسل اتفاق بیفتد، در واقع ممکن است برخی والدین بدون عمل ترکیب به نسل‌های جدید تبدیل شوند. برای تعیین رخ دادن یا ندادن ترکیب از پارامتری به نام احتمال ترکیب ($P_C$) استفاده می شود که مقداری بین ۰ تا ۱ دارد (معمولاً بین ۰/۵ تا ۰/۸ در نظر گرفته می‌شود). اگر احتمال ترکیب ۰٪ باشد (ترکیبی صورت نگیرد)، نسل جدید دقیقاً همان نسل قدیم است و اگر این احتمال ۱۰۰٪ باشد، همه فرزندان در نتیجه ترکیب به وجود آمده‌اند.

بدلیل اهمیت این مرحله، بسیاری از تحقیقات دربارهٔ الگوریتم ژنتیک بر روش‌های ترکیب و تحلیل آنها اختصاص یافته است؛ در ادامه متداولترین روش‌های ترکیب توضیح داده می‌شوند:

* ترکیب تک نقطه‌ای:

اگر تعداد ژن‌ها در کروموزوم‌ها m باشد، ترکیب تک نقطه‌ای، دو کروموزوم را با انتخاب تصادفی موقعیتی مانند p ($p \leq m$)، ترکیب می‌کند. از دو کروموزوم والد، دو فرزند به صورت زیر بوجود می‌آید:

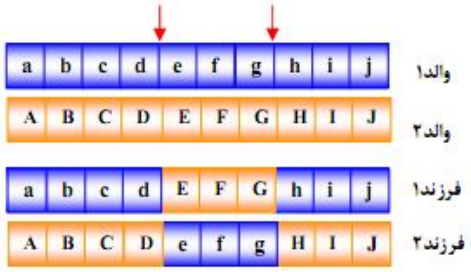



**شکل ۸- ترکیب تک نقطه‌ای**

یک فرزند با کپی کردن ژن‌های۱ تا p-۱ از کروموزوم والد اول و ژن‌هایp تا m از کروموزوم دوم، ساخته می‌شود و فرزند دیگر بطور مشابه، با کپی کردن ژن‌های ۱ تا p-۱ از والد دوم و ژن‌های p تا m از والد اول، بوجود می‌آید. در این نوع ترکیب از دو والد، دو فرزند بوجود می‌آید. لازم به ذکر است که اگرp=۱، یا p=m، آنگاه دو والد بدون تغییر وارد جمعیت بعدی می‌شوند.

## * ترکیب دو نقطه‌ای:

در این ترکیب دو موقعیت $p_1$ و $p_2$ (موقعیت‌های ترکیب) بطور تصادفی بین ۱ تا m انتخاب می‌شوند. فرزند اول، ژن‌های ۱ تا $p_1-1$ را از والد اول، ژن‌های $p_1$ تا $p_2-1$ را از والد دوم و ژن‌های $p_2$ تا m را نیز از والد اول، به ارث می‌برد. فرزند دوم، ژن‌های ۱ تا $p_1-1$ را از والد دوم، ژن‌های $p_1$ تا $p_2-1$ را از والد اول و ژن‌های $p_2$ تا m را مجدداً از والد دوم، بدست می‌آورد (شکل ۹). در این روش ترکیب نیز، از یک جفت دو فرزند بوجود می‌آید، اما احتمال اینکه والدها بدون تغییر به جمعیت بعد منتقل شوند، کمتر است.




**شکل ۹- ترکیب دو نقطه‌ای**

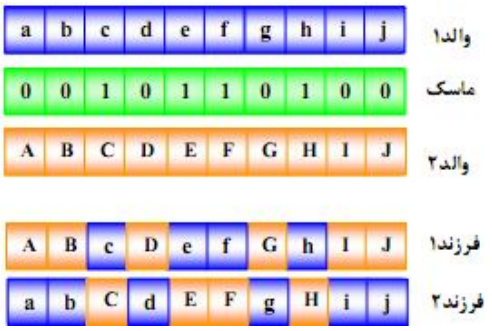
## * ترکیب n نقطه‌ای:

در این روش n نقطه تصادفی در کروموزوم‌های والد انتخاب و ژن های بین نقاط انتخاب شده بصورت یک در میان با هم عوض می‌شوند. هر چه تعداد نقاط انتخاب شده زیاد باشد، کارآیی نیز افزایش پیدا می‌کند. تعداد نقاط جابه جایی بهینه که مورد نظر قبول عموم قرار گرفته است، بطور معمول نصف طول کروموزوم می‌باشد. معمولاً کاربرهای الگوریتم‌های ژنتیک تعداد نقاط شکست در عمل ترکیب را یک یا دو نقطه انتخاب می‌کنند.

### * ترکیب یکنواخت (Monotonous):

در این ترکیب، هر ژن کروموزوم جدید وابسته به موقعیتش بصورت تصادفی از یکی از دو والد انتخاب می‌شود؛ مثلاً ژن اول از والد اول، ژن دوم از والد دوم، ژن سوم از والد اول. برخلاف ترکیب‌های پیشین، در این حالت می‌توان فقط یک فرزند تولید کرد. یک الگوی بیان کننده عمل جابه‌جایی (ماسکی با طول m) بصورت تصادفی ایجاد می‌شود. چنانچه در شکل ۱۰ دیده می‌شود، مقدار هر بیت این ماسک، والد مرجع انتقال به آن بیت از فرزند را تعیین می‌کند. در این شکل مشاهده می‌شود که اگر بیت مربوط در الگوی جابه‌جایی یا ماسک ۱ باشد، آن بیت در فرزند اول (دوم) برابر با مقدار بیت متناظر در والد اول (دوم) و اگر بیت ماسک ۰ باشد، مقدار بیت مربوطه در این فرزند برابر با مقدار بیت متناظر در والد دوم (اول) است.

ترکیب‌های یک و چند نقطه‌ای در جایی اثر می‌کنند که کروموزوم فقط در آن نقاط می‌تواند جدا شود.




**شکل ۱۰- ترکیب یکنواخت**

اما عملگر جابه‌جایی یکنواخت پتانسیل جابجا شوندگی را به تمام نقاط یک کروموزوم به صورت یکنواخت نسبت می‌دهد. به این معنی که احتمال جابجا شدن کروموزوم در هر نقطه برابر خواهد بود. بنابراین جمعیت جدیدی که با ترکیب یکنواخت بوجود می‌آید، دارای تنوع ژنتیکی بیشتری است؛ بهمین دلیل این نوع ترکیب در جمعیت‌هایی که اعضای کمی دارند اثر بهتری دارد تا جمعیت‌هایی که تعداد اعضای زیادی دارند. در جمعیت‌های کوچک، ممکن است به تنوع ژنتیکی نیاز باشد تا همگرایی سریعتر شود؛ اما در جمعیت‌های بزرگ، معمولاً تنوع ژنتیکی لازم فراهم است. هنگامی که از عملگر ترکیب یکنواخت در مقادیر حقیقی استفاده شود به آن "ترکیب‌بندی منفصل" نیز گفته می‌شود.

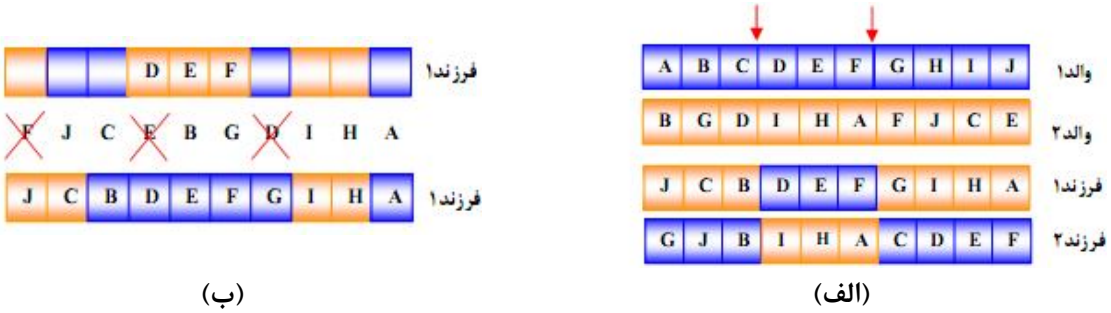
### * ترکیب حسابی:

اگر A و B دو عضو انتخابی از جمعیت فعلی باشند بعنوان والد باشند، از آنها دو فرزند a و b به صورت زیر بوجود می‌آید (پارامتر $\delta$ مقداری در بازه $[0,1]$ بوده و در هر ترکیب می‌تواند مقدار متفاوتی داشته باشد):

$$a = \delta A + (1-\delta)B \quad \text{و} \quad b = \delta B + (1-\delta)A \qquad (۴)$$

## ❋ ترکیب مرتب شده (Ordered Crossover):

در این روش که توسط دیویس معرفی گردیده، ترتیب نسبی ژن ها در کروموزوم ها را تا حد ممکن حفظ می‌کند. دو فرزند بوسیله انتخاب دو نقطه برش در والد بدست می‌آیند: عناصر بین دو نقطه، در فرزندان کپی و باقی ژن‌های فرزندان با عناصر استفاده نشده از والد دیگر با شروع از نقطه برش دوم به بعد، پر می‌شوند.




**شکل ۱۱- الف) ترکیب مرتب شده و ب) نحوه پر کردن عناصر در آن**

بعنوان مثال در شکل ۱۱، برای ایجاد فرزند اول، عناصر بین نقاط برش از والد اول کپی می شوند. سپس با شروع از نقطه برش دوم والد دوم و حذف عناصری که قبلاً در فرزند اول ظاهر شده‌اند، باقی عناصر فرزند پر می شوند (فرزند دوم نیز به همین ترتیب ایجاد می شود).

## ❋ ترکیب چرخشی:

در این روش که توسط الیور اسمیت و هولند معرفی شده، یک چرخه بین ژن‌های کروموزوم‌های والد ایجاد می‌گردد. ابتدا اولین ژن را بعنوان نقطه یا ژن مرجع در نظر گرفته و محتوای آن به کروموزوم دوم منتقل می‌شود. محتوای کروموزوم دوم در ژن یا نقطه مرجع را در کروموزوم اول یافته و مکان آن بعنوان مرجع جدید در نظر گرفته می‌شود. با در نظر گرفتن نقطه یا ژن مرجع جدید، این عمل تا جایی که چرخه کامل شود (تا جایی که مجدداً به نقطه شروع در کروموزوم اول برسد)، ادامه می‌یابد. در نهایت برای تکمیل ژن‌های باقیمانده فرزند اول، از عناصر متناظر کروموزوم دوم استفاده می‌شود.

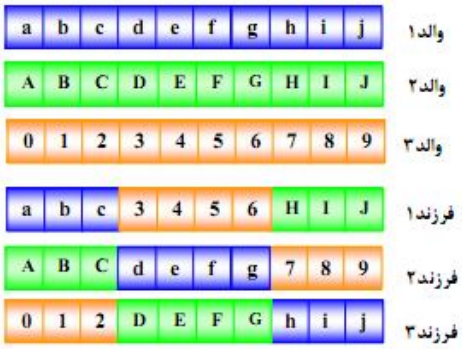
مثال شکل ۱۲ را در نظر بگیرید: برای ایجاد فرزند اول، با شروع از اولین عنصر والد اول، J دیده می شود که آنرا به H در والد دوم منتقل می‌کنیم؛ در ادامه H والد اول به G والد دوم، G در والد اول به I والد دوم، I در والد اول به J در والد دوم نگاشت می‌شوند و چرخه (با رسیدن به اولین عنصر یعنی J) کامل می‌شود. عناصری از چرخه که در والد اول هستند، در فرزند اول قرار داده می شود و باقی مکان‌های فرزند اول با عناصر متناظر در والد دوم پر می‌شوند (فرزند دوم نیز به همین ترتیب ایجاد می شود).


(ب) (الف)

**شکل ۱۲- الف) ترکیب چرخشی و ب) نحوه پر کردن عناصر در آن**

## * ترکیب مورب یا قطری (Diagonal Crossover):

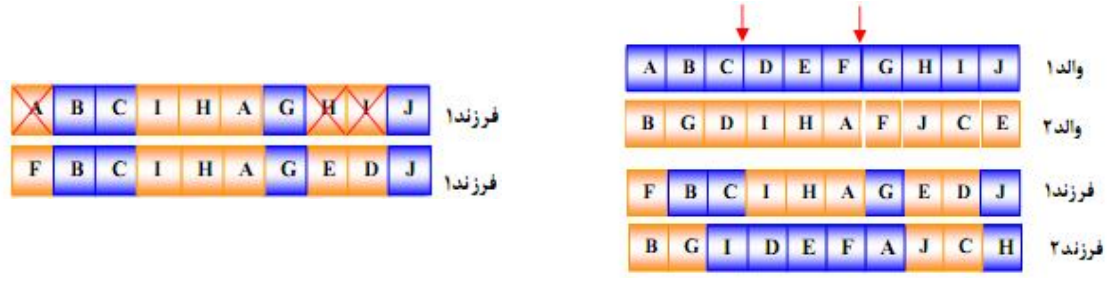
در این روش که تعمیم یافته روش تک نقطه‌ایست، برای n والد، n-1 نقطه جابه‌جایی تعیین و ژنها بین نقاط جابه‌جایی کروموزوم‌های والدین در امتداد قطر برداشته و n کروموزوم فرزند ایجاد می‌شوند (شکل ۱۳).




**شکل ۱۳- ترکیب مورب**

## * ترکیب نگاشت جزئی (Partially mappped crossover):

این روش که توسط گلدبرگ و لینگل معرفی شده، در حقیقت حالت خاصی از ترکیب دو نقطه‌ایست. در این روش دو عدد تصادفی بعنوان نقاط برش انتخاب و بخش بین این نقاط در دو کروموزوم تعویض می‌شود.




(ب) (الف)

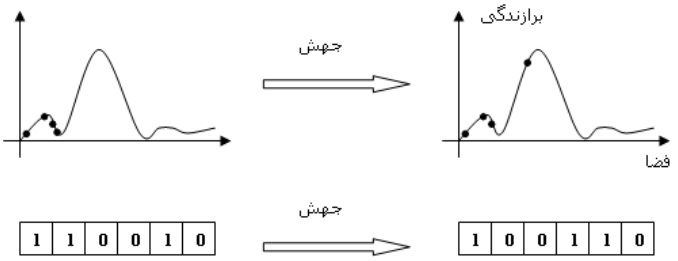
**شکل ۱۴- الف) ترکیب نگاشت جزئی و ب) نحوه پر کردن عناصر در آن**

بعد در والد اول از ابتدای کروموزوم شروع نموده هر عددی را که در قسمت مابین کروموزوم جدید نباشد عیناً نوشته و برای تکراری‌ها جای خالی (حفره) قرار داده می‌شود. به فرض اگر حفره در فرزند اول باشد، ابتدا

موقعیت ژنی از والد دوم را که دارای مقدار مساوی با مقدار حفره است پیدا می شود. سپس از والد اول مقدار ژنی را که در موقعیت یکسان با ژن پیدا شده قرار دارد بعنوان مقدار حفره در نظر گرفته می‌شود (شکل ۱۴).

## ۲-۹- جهش (Mutation)

در طبیعت برخی عوامل محیطی سبب بوجود آمدن تغییرات غیرقابل پیش‌بینی در کروموزوم‌ها می‌شوند. از آنجایی که الگوریتم‌های مبتنی بر ژنتیک از قانون تکامل پیروی می‌کنند، در آنها عملگر جهش با احتمال کم اعمال می‌شود. جهش باعث جستجو در فضاهای دست نخورده مسأله می‌شود. مهمترین وظیفه جهش اجتناب از همگرایی مسئله به بهینه‌های محلی است (شکل ۱۵). به تعبیری دیگر جهش را می‌توان مشابه شروع مجدد تصادفی الگوریتم تپّه نوردی ***(Hill Climbing)*** هنگام گیر افتادن در فلات دانست.




**شکل ۱۵- تأثیر جهش بر فرار از بهینه محلی**

در الگوریتم ژنتیک نیز بعد از اینکه یک عضو در جمعیت جدید به وجود آمد، هر ژن آن با احتمالی مشخص جهش می‌یابد. احتمال جهش($P_m$) مقداریست که توسط کاربر تعیین می‌شود. در جهش ممکن است ژنی از مجموعه ژن‌های جمعیت حذف و یا ژنی که تا بحال وجود نداشته است به آن اضافه شود. جهش یک ژن به معنای تغییر آن ژن است و وابسته به نوع کدگذاری، روش‌های متفاوت جهش استفاده می‌شود. اگر احتمال تغییر ۱۰۰٪ باشد، یعنی همه کروموزوم‌های تغییر کرده‌اند و اگر ۰٪ باشد، هیچ تغییری نکرده‌اند. در هر حال $P_m$ بین ۰ تا ۱ است؛ نرخ بالای جهش، الگوریتم را کاملا تصادفی کرده و باعث تنوع و پراکندگی ژنتیکی در جمعیت می‌شود (این پراکنندگی ممکن است همگرایی را به تأخیر بیندازد). در الگوریتم استاندارد ژنتیک بنا بدلایلی مقدار این پارامتر بسیار کوچک (مثلاً $P_m$=0.01 یا $P_m$=0.001) انتخاب می‌شود. با اینحال برخی مراجع مقدار آنرا برابر با $P_m$=1/m پیشنهاد کرده‌اند. پیشنهاد می‌شود برای جمعیت‌های بزرگ یا در نسل‌های آخر از $P_m$ های کوچکتر و برای جمعیت‌های کوچک یا در نسل‌های ابتدایی از $P_m$های بزرگتر استفاده شود. برای فرزندی که از مرحله ترکیب بوجود آمده، مقداری تصادفی بین ۰ تا ۱ برای هر ژن تولید شده و اگر این مقدار از $P_m$ کمتر باشد، ژن جهش یافته و در غیراینصورت ژن بی‌تغییر می‌ماند.

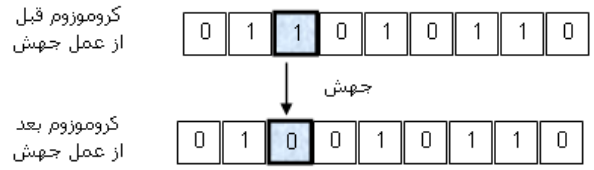
در ادامه چند روش جهت پیاده‌سازی جهش بیان می‌شود:

### ۲-۹-۱- جهش برای متغیرهای کد شده باینری

در الگوریتم ژنتیک با کدگذاری باینری، عملگر جهش تنها می‌تواند با تولید تصادفی یکی از اعداد ۰ و۱ و جایگزینی آن بجای بیت مورد نظر صورت گیرد. بااینحال دو روش دیگر نیز در ادامه ارائه شده‌اند.

#### * وارونه سازی بیت:

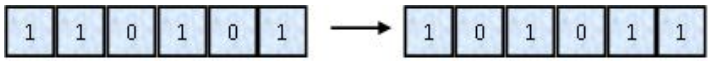
در این روش، عمل جهش دودوئی در یک بیت را می‌توان با متمم ساختن آن بیت انجام داد (اگر بیت ۰ باشد به ۱ و اگر ۱ باشد به ۰ تبدیل می‌شود). مثالی از این نوع جهش در شکل ۱۶ نشان داده شده است.




**شکل ۱۶- جهش در کروموزوم‌های باینری به روش وارونه سازی بیت**

#### * وارونه سازی کروموزوم:

این عمل در طبیعت بسیار رخ می‌دهد ولی در الگوریتم‌های ژنتیک بدلیل تخریب زیاد بندرت استفاده می‌شود. در این روش، ترتیب قرارگیری بیت‌های کروموزوم معکوس می‌شود (شکل ۱۷).




**شکل ۱۷- جهش در کروموزوم‌های باینری به روش وارونه سازی کروموزوم**

### ۲-۹-۲- جهش برای متغیرهای کد شده حقیقی

در کدگذاری حقیقی، عملگر جهش باعث تولید تصادفی یک مقدار جدید در یک موقعیت خاص کروموزوم می‌شود تا تغییرات تصادفی در جمعیت سبب بررسی نواحی بیشتری از فضای کاوش شده و از همگرایی بی‌موقع (محلی) الگوریتم جلوگیری می‌شود.

#### * جهش تصادفی یا یکنواخت:

در این حالت، یک ژن از کروموزوم به طور تصادفی انتخاب شده و مقدار آن با یک مقدار جدید تصادفی (از محدوده مجاز آن ژن) جایگزین می‌شود.

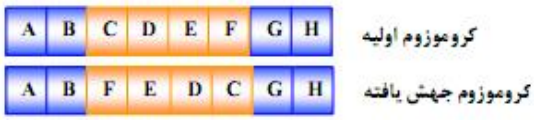

**شکل ۱۸- جهش تصادفی در کروموزوم‌های حقیقی**

❋ جهش مرزی:

در این عملگر یکی از ژن‌های کروموزوم بطور تصادفی با حد پایین یا بالای محدوده آن ژن جایگزین می‌شود.

❋ وارونه سازی:

از این نوع جهش مخصوصاً در الگوریتم‌هایی استفاده می‌شود که کد گذاری بر اساس مقدار باشد. البته دیده شد که در کدگذاری باینری هم می‌توان این جهش را بکار برد. در این حالت دو ژن از کروموزوم به طور تصادفی انتخاب شده و ترتیب ژن ها در بین این دو ژن معکوس می شود (شکل ۱۹).




**شکل ۱۹- جهش در کروموزوم‌های حقیقی به روش وارونه سازی**

❋ تغییر مقدار:

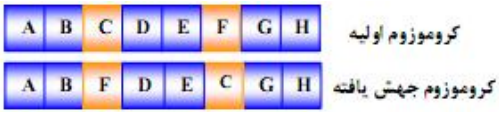
این نوع جهش را نمی‌توان برای کدگذاری باینری یا کدگذاری‌های مشابه بکار برد. در این جهش به ژنی که شرایط جهش را دارد، مقداری تصادفی (یا از پیش تعیین شده) اضافه یا کم می‌شود. مقداری که به ژن افزوده یا از آن کاسته می‌شود، وابسته به محدوده مقدار ژن است. بدیهی است مقدارهای بزرگ پراکندگی ژنتیکی را افزایش می‌دهند. در شکل ۲۰ ژن‌های دوم و سوم کروموزوم جهش یافته‌اند. از این شکل پیداست که مقدار انتخابی برای افزودن یا کاستن می‌تواند از ژنی به ژن دیگر متفاوت باشد.

**(1.29, 5.68, 2.86, 4.11, 5.55) ➡ (1.29, 5.68, 2.73, 4.22, 5.55)**

**شکل ۲۰- جهش در کروموزوم‌های حقیقی به روش تغییر مقدار**

❋ روش تعویض (Swap mutation):

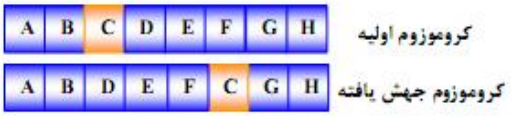
دو ژن یا دو بلوک از ژن ها در کروموزوم بطور تصادفی انتخاب و موقعیت آن ها عوض می شود (شکل ۲۱).




**شکل ۲۱- جهش در کروموزوم‌های حقیقی به روش تعویض**

* روش درجی (**Insertion mutation**):

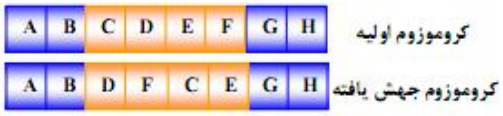
یک ژن یا یک بلوک از ژن ها در کروموزوم بطور تصادفی انتخاب و در یک نقطه تصادفی دیگر درج می‌شود.




**شکل ۲۲- جهش در کروموزوم‌های حقیقی به روش درجی**

* روش در هم آمیخته (**Scramble mutation**):

یک بلوک از ژن ها بطور تصادفی انتخاب و دوباره بصورت تصادفی مرتب می شوند (شکل ۲۳)




**شکل ۲۳- جهش در کروموزوم‌های حقیقی به روش درهم آمیخته**

## ۲-۱۰- ترمیم (Repair) و نخبه گزینی (Elitism)

ممکن است پس از انجام اعمال عملگرهای جهش و ترکیب، راه حل های نامعتبر ایجاد شوند. برای رفع این مشکل از عملگر ترمیم استفاده می‌شود تا راه حل ایجاد شده معتبر گردد. این عملگر بسته به مشخصات مسئله و پارامترهای آن توسط برنامه نویس مشخص می‌گردد.

ضمناً برای این که راه حل‌های بهتر از بین نروند، کروموزومی که در نسل جدید دارای پایین‌ترین مقدار برازندگی باشد حذف و کروموزومی که در نسل قبلی دارای بالاترین برازندگی است، جایگزین آن می‌شود.

## ۲-۱۱- همگرایی و شرایط خاتمه در الگوریتم ژنتیک

سوال مهمی که می تواند مطرح شود این است که آیا الگوریتم ژنتیک همواره به سمت بهینه مطلق همگرا می شود؟ تحقیقات رونکلف در سال ۱۹۹۴، همگرایی الگوریتم ژنتیک را تحت شرایط خاص به اثبات می‌رساند. در این تحقیقات، نشان داده شده است که همگرایی بسمت بهینه مطلق یک خاصیت ذاتی الگوریتم ژنتیک نمی‌باشد، ولی با رعایت شرایط خاصی امکان پذیر است. در این تحقیق نشان داده شد که چنانچه در هر مرحله تولید از الگوریتم ژنتیک، بهترین جواب‌ها نگاه داشته شده و به مرحله بعد وارد شوند، الگوریتم می‌تواند بسمت بهینه مطلق همگرا شود. عوامل متعددی در همگرایی الگوریتم ژنتیک و سرعت آن

موثرند: جمعیت اولیه، مقادیر احتمال جهش و ترکیب، چگونگی انجام عمل ترکیب و جهش، تابع برازندگی و چگونگی انتخاب جمعیت بعدی و ... . چگونگی تأثیر این عوامل در سرعت همگرایی، به مسأله مورد نظر بستگی داشته و از طریق آزمایش بدست می‌آید.

برای اینکه تشخیص خاتمه اجرای الگوریتم نیز می‌توان از شیوه‌ها و معیارهای متفاوتی استفاده کرد:

- اگر تعداد تکرار یا تعداد نسل‌های الگوریتم به حد مشخصی برسد.
- اگر جواب نهایی مورد انتظار با دقت قابل قبول بدست آید.
- اگر با پیشروی الگوریتم هیچ نوع بهبودی مشاهده نشود (خواه الگوریتم جواب دلخواه را پیدا کرده و خواه در مینیمم محلی گیر کرده باشد) و بیشترین درجه برازش نسل‌ها حاصل شده باشد.
- اگر مقدار میانگین برازندگی (یا تابع هدف) نسل به مقدار خاصی همگرا شده باشد.
- فاصله برازندگی بهترین فرد جمعیت از متوسط برازندگی جمعیت از میزان مشخصی کوچکتر شود.
- بازرسی چشمی انجام گرفته و الگوریتم بصورت دستی متوقف شود.
- ترکیبی از معیارهای بالا برقرار باشد.

## ۲-۱۲- انواع الگوریتم‌های ژنتیک

الگوریتم ژنتیک (GA) که نمونه اولیه آن توسط هولند در سال ۱۹۷۵ ارائه شد، تکامل طبیعی را در سطح ژن و کروموزوم شبیه‌سازی می‌کنند. عملکرد غالب در تولید نسل جدید، پیوند کروموزوم هاست، گرچه جهش در ژن‌ها نیز به عنوان یک عملکرد ثانوی به کار می‌رود. انواع بسیاری برای این الگوریتم شناخته شده است، که در ادامه به تعدادی از آنها اشاره می‌شود:

- الگوریتم ژنتیک سری یا ترتیبی (***Sequential GA***): الگوریتم ژنتیک سری همان الگوریتم ژنتیک معمولی است که در مقابل نوع موازی، سری نام گرفته است. الگوریتم ژنتیک در هر تکرار محاسباتی (نسل) روی جمعیتی از رشته‌ها عمل می‌کند.
- الگوریتم ژنتیک موازی (***Parallel GA***): تا کنون دو مدل اصلی در الگوریتم ژنتیک موازی مطرح گشته است: مدل جزیره‌ای (Island model) و مدل همسایگی (Find grained model). بطور مثال در مدل جزیره‌ای چندین زیرجمعیت مجزا مطابق با الگوریتم ژنتیک معمولی تکامل می‌یابد و هر از چند گاهی زیر جمعیت‌های همسایه، بهترین کروموزوم یکدیگر را معاوضه می‌کنند.

- الگوریتم ژنتیک آشفته (***Messy GA***)
- الگوریتم ژنتیک هیبرید (***Hybrid GA***)
- الگوریتم ژنتیک خودسازمان (***Adaptive GA***)
- الگوریتم ژنتیک زایشی (Generational GA)
- الگوریتم ژنتیک حالت دائمی (Steady State GA)

نکته: اگر یک کروموزوم فاصله اش با سایر کروموزوم های نسل خودش زیاد باشد (خیلی بهتر از بقیه باشد) و خیلی زود پیدا شود، ممکن است راه حل را به سوی جواب بهینه محلی سوق دهد. این اتفاق به طور معمول در جمعیت‌های با تعداد کمتر اتفاق می‌افتد.

## ۲-۱۳- استراتژی برخورد با محدودیت‌های مسئله

بحث دیگری که در اجرای الگوریتم ژنتیک وجود دارد چگونگی برخورد با محدودیت‌های مسأله می‌باشد؛ زیرا عملگرهای ژنتیک مورد استفاده در الگوریتم می‌توانند باعث تولید کروموزوم‌های غیرموجه شوند. در ادامه چند تکنیک معمول جهت مواجهه با محدودیت‌ها ارائه شده‌اند.

### ❋ استراتژی اصلاح عملگرهای ژنتیک

یک روش برای جلوگیری از تولید کروموزوم غیرموجّه، تعریف عملگرهای ژنتیکی بگونه‌ایست که پس از عمل بر روی کروموزم‌ها، کروموزوم تولید شده نیز موجّه باشد. البته پیدا کردن عملگری که دارای شرط فوق باشد، بسیار دشوار بوده و برای مسائل مختلف، متفاوت است.

### ❋ استراتژی رَدّی

در این روش، هرکروموزوم پس از تولید از نظر موجّه بودن تست و در صورت غیرموجّه بودن حذف می‌شود.

### ❋ استراتژی اصلاحی

در این روش به جای اینکه کروموزوم غیرموجّه حذف گردد، تبدیل به یک کروموزوم موجّه می‌شود. این روش نیز مانند روش اول به نوع مسأله وابسته بوده و یافتن فرآیند اصلاح گاهی بسیار پیچیده می‌باشد.

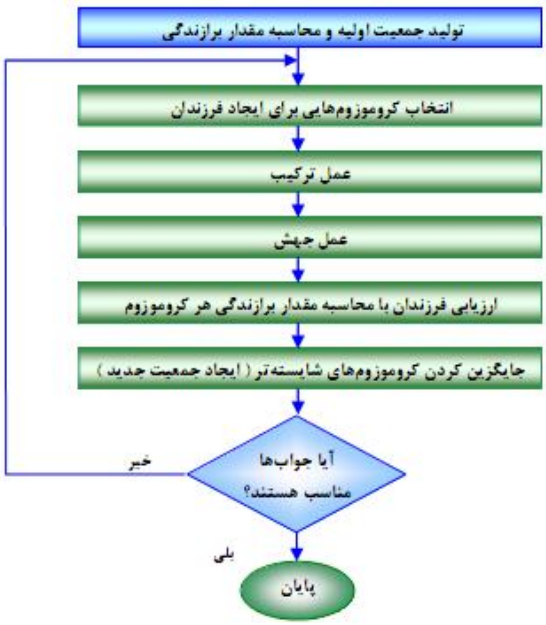
* استراتژی جریمه‌ای

در این روش بر خلاف سه روش قبل (که از ورود جواب‌های غیرموجّه جلوگیری می‌کردند)، جواب غیرموجّه با احتمال کم امکان حضور می‌یابند. سه روش فوق دارای این عیب بودند که به هیچ نقطه‌ای بیرون از فضای موجّه توجّه نمی‌کردند، اما در بعضی مسائل بهینه‌سازی، جواب‌های غیرموجّه درصد زیادی از جمعیت را اشغال می‌کنند. در چنین شرایطی اگر جستجو فقط در ناحیه موجّه انجام گیرد، شاید یافتن جواب موجّه خیلی وقت‌گیر و مشکل باشد و یا مسئله در نقاط بهینه محلی بدام افتد.

استراتژی جریمه‌ای از متداولترین تکنیک‌های مورد استفاده برای سر و کار داشتن با جواب‌های غیرموجّه است. در این روش، محدودیت‌های مسأله بصورت مشخص در نظر گرفته نمی‌شوند و برای هر تخلف از محدودیت‌ها، یک ضریب جریمه در تابع هدف اختصاص داده می‌شود. مسأله اصلی، چگونگی انتخاب یک مقدار مناسب برای ضریب جریمه است. نکته‌ای که در روش جریمه وجود دارد، این است که در این روش یک جواب غیرموجّه بسادگی حذف نمی‌شود، زیرا ممکن است در ژنهای آن اطلاعات مفیدی وجود داشته باشد که با اندکی تغییر به جواب بهینه تبدیل شود.

## ۲-۱۴- فلوچارت اجرای الگوریتم ژنتیک

برای پیاده‌سازی الگوریتم ژنتیک، ابتدا باید متغیرهای مسئله تعیین و بنحو مناسبی کدگذاری شوند. ضمناً براساس هدف مسأله، یک تابع برازش برای متغیرها (کروموزوم‌ها) تعریف می‌گردد.




**شکل ۲۴- فلوچارت مراحل اجرای الگوریتم ژنتیک**

در ابتدای اجرای الگوریتم، یک جمعیت اولیه دلخواه بطور تصادفی انتخاب و میزان برازندگی هر یک از کروموزوم‌های جمعیت اولیه توسط تابع برازش محاسبه می شود. در ادامه اجرای الگوریتم، مراحل زیر مطابق با شکل ۲۴، بصورت تکراری پی گرفته می‌شوند:

مرحله ۱ (تولید یا تکثیر): در این مرحله تعداد مناسبی از زوج کروموزوم‌ها براساس میزان برازندگی آنها برای استفاده در مراحل بعدی انتخاب می‌شوند. کروموزوم‌هایی که دارای مقدار برازندگی بالایی هستند، ممکن است چندین بار در مرحله تولید انتخاب شوند، در حالی که کروموزوم‌های با مقدار برازندگی کم ممکن است هیچگاه انتخاب نگردند (بخش ۲-۷).

مرحله ۲ (ترکیب یا آمیزش): در این مرحله عمل گر ترکیب با احتمال $P_c$ بر روی کروموزوم‌های والد عمل کرده و با ترکیب آنها، کروموزوم‌های جدیدی ( فرزندان) را تولید می‌کند (بخش ۲-۸).
انتخاب اینکه در هر لحظه کدام دو زوج کروموزوم بعنوان والدین تعیین شوند، می‌تواند بر اساس استراژی‌های مختلفی انجام شود. از جمله این انتخاب‌ها می‌توان موارد زیر را پیشنهاد کرد:

- ترکیب دو والد پشت سر هم
- انتخاب تصادفی یک زوج والد از بین کروموزوم‌ها و ترکیب آنها
- ترکیب بخشی از کروموزوم‌ها با بهترین عضو جمعیت همان نسل (با تعریف احتمال ترکیب بهترین)
- ترکیب بخشی از کروموزوم‌ها با بهترین فرد یافته شده تاکنون (با تعریف احتمال ترکیب بهترین)

البته دو راه حل آخر ممکن است تحت شرایطی باعث هم شکل شدن جمعیت و همگرایی بیش از حد سریع (به بهینه محلی) شوند.

مرحله ۳ (جهش): عمل جهش با احتمال $P_c$ بر روی کروموزوم‌های حاصل از عمل ترکیب انجام می‌شود (بخش ۲-۹).

مرحله ۴ (ارزیابی): بمنظور ارزیابی فرزندان، مقدار برازندگی کروموزوم‌های جدید محاسبه می‌گردد.

مرحله ۵: در این مرحله جمعیت جدید برای ورود به مرحله بعد الگوریتم، انتخاب می‌گردد. این کار با مقایسه مقدار برازندگی کروموزوم‌ها انجام می‌شود. روش‌های مختلفی برای ایجاد جمعیت جدید وجود دارد که بطور نمونه می‌توان دو روش زیر را نام برد:

- تمام اعضای جمعیت جدید از میان فرزندان انتخاب می‌شوند.

- افراد جمعیت جدید از میان شایسته‌ترین افراد دو نسل (والدین و فرزندان) انتخاب می‌شوند.

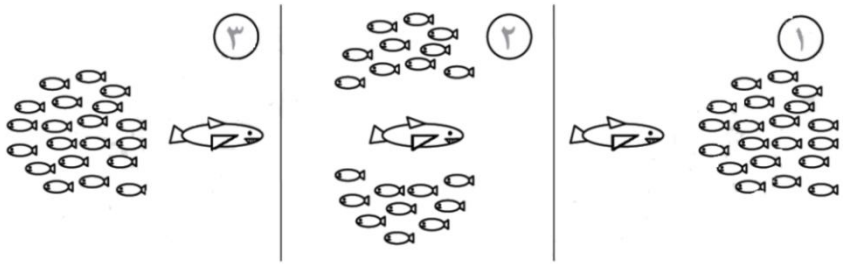
حذف همه کروموزوم‌های جمعیت مرحله قبل و انتخاب جمعیت جدید از میان فرزندان، ممکن است بسیاری از جواب‌های مناسب را که در میان جمعیت مرحله قبل وجود دارد، حذف نماید. بنابراین پیشنهاد می شود که چنانچه از روش اول برای انتخاب جمعیت جدید استفاده می شود، در هر مرحله، بهترین جواب (ها) ذخیره شوند (مانع از دست رفتن اطلاعات در مرحله انتخاب می شود). البته راه حل دوم ممکن است باعث هم شکل شدن جمعیت و همگرایی بیش از حد سریع (به بهینه محلی) شود.

مرحله ۶: بعد از طی مراحل، چنانچه شرایط خاتمه الگوریتم فراهم باشد، الگوریتم پایان پذیرفته و در غیر این صورت جمعیت موجود بعنوان جمعیت اولیه برای مرحله بعد مورد استفاده قرار می‌گیرد (بخش ۲-۱۱).

# فصل سوم:

# الگوریتم‌های مبتنی بر هوش جمعی

برای برخی از حیوانات که بصورت گروهی زندگی می‌کنند (از جمله دسته‌های ماهی)، رفتارهای پیچیده‌ای به هنگام حرکت مشاهده می‌شود. این در حالیست که هر کدام از اعضای جمع به اطلاعات محدودی دسترسی داشته و فقط از موقعیت تعداد اندکی از همسایگانشان خبر دارند. بعنوان مثال، دسته‌ای از ماهی‌ها برای دفع خطر یک شکارچی، در ابتدا به دو قسمت تقسیم و سپس از نو ساخته می‌شوند؛ البته در هر حال، نزدیکی و فشردگی کل جمعیت از طرف همهٔ ماهی‌ها کنترل می‌گردد.




شکل ۱- دفع خطر شکارچی توسط گروهی از ماهی‌ها

در چنین مجموعه‌ای، هر کدام از حیوانات فقط از چند قانون ساده تبعیت کرده و رفتارهای پیچیده‌ای که در کل جمعیت قابل مشاهده‌اند، چیزی جز ترکیب این قوانین ساده نیستند. هر ماهی در یک دسته، از موقعیت، جهت حرکت و سرعت ماهی‌های اطرافش خبر دارد و با استفاده از این اطلاعات و پیروی از چند قانون ساده، خود را با جمعیت تطبیق می‌دهد. بعنوان مثال موارد زیر از جمله قوانین ساده‌ای‌اند که هر ماهی در یک دسته رعایت می‌کند:

- همواره سعی داشته باش که بطور نسبی در نزدیکی سایرین حرکت کنی.
- سعی کن در جهتی که بقیه حرکت می‌کنند، حرکت کنی.تقریباً با همان سرعتی که دیگران حرکت می‌کنند، حرکت کن.

حاصل تبعیت از چنین قوانینی، حرکت‌های پیچیده‌ای است که در مجموع بوسیله دستهٔ ماهی‌ها انجام می‌گیرد. این رفتارها و رفتارهای حشرات، همگی با اصلی بنام خود سازمان‌دهی (Self - organization) شناخته می‌شوند. خود سازمان‌دهی فرآیندیست که در آن الگوی کلی یک سیستم پیچیده، فقط در اثر تعداد

زیادی از تعاملات مرتبه پایین و درونی شکل می‌گیرد؛ بعلاوه، قوانینی که بر تعامل بین عناصر تشکیل دهندهٔ سیستم حاکمند، فقط از اطلاعات محلی استفاده کرده و هیچ وقت از اطلاعات سراسری بهره نمی‌برند.

برخی از الگوهای موجود در طبیعت را در نظر بگیرید: طرز چینش و تقسیم‌بندی طرح پوست زرافه با چند قانون ساده قابل بیان است. مارپیچی که دانه‌های گل آفتاب گردان را تشکیل می‌دهد، با سری فیبوناچی قابل توصیف است. عملکرد جمعی پرندگان و ماهی‌ها نیز با وجود تمام پیچیدگی‌هایشان، از ترکیب و تکرار چند قانون ساده ساخته شده‌اند. تفاوت بین این نمونه‌ها در سکون سیستم آنهاست؛ در دو نمونه دوم قوانینی برای حرکت نیز وجود دارند، اما در دو نمونه اول، فقط قوانینی برای طرز چینش قابل توصیف هستند.

با وجود آن که فقط ۲ درصد از گونه‌های حشرات دارای زندگی اجتماعی هستند، اما بیش از ۵۰ درصد تودهٔ زیستی حشرات را تشکیل می‌دهند. این میزان در برخی جاها، مانند جنگل‌های بارانی آمازون به بیش از ۷۵ درصد می‌رسد. منظور از زندگی اجتماعی، تجمع تعداد زیادی از یک گونهٔ خاص در قالب یک مجموعه یا کلونی و تعامل آنها با همدیگر است. همه مورچه‌ها، موریانه‌ها و همچنین برخی از گونه‌های زنبورها در قالب کلونی زندگی می‌کنند. اجتماع حشرات می‌توانند مسایلی را با همکاری یکدیگر حل و فصل نمایند که هیچ یک از اعضای آن اجتماع به تنهایی قادر به حل آنها نیست. بیشتر این مسایل بصورت مسایل بهینه‌سازی قابل بیان هستند. بعنوان مثال، تلاش حشرات برای یافتن کوتاهترین مسیر در هنگام جستجو برای غذا، تخصیص مناسب نیروهای کاری برای انجام کارها مختلف و همچنین طبقه‌بندی محل‌های حاوی تخم‌ها و نوزادان از جمله مسایل بهینه‌سازی‌اند که حشرات اجتماعی با همکاری یکدیگر آنها را حل می‌کنند. این مسایل همگی دارای معادلی در بین مسایل بهینه‌سازی روزمره هستند.

هر تلاشی که برای حل یک مسأله بهینه‌سازی صورت می‌گیرد، باعث بوجود آمدن اطلاعاتی در مورد آن مسأله می‌گردد. در هر اجتماع از حشرات، نوع خاصی از ارتباط بین حشرات وجود دارد. این ارتباط در گونه‌های مختلف می‌تواند بصورت مستقیم یا غیرمستقیم باشد. بعنوان مثال، هنگامی که یک زنبور عسل یک منبع غذایی جدید پیدا می‌کند، با اجرای یک رقص ویژه، جهت و فاصلهٔ محل منبع غذایی را به سایر زنبورها اطلاع می‌دهد (ارتباط بسیار مستقیم). برای آنکه زنبوری از پیام مورد نظر اطلاع یابد، می‌بایست رقص زنبور را بطور مستقیم مشاهده و آن را تفسیر کند. ارتباط و تماس فیزیکی، نوع دیگری از ارتباط‌های مستقیم میان حشرات اجتماعی است. ارتباط غیرمستقیم نیاز به مهارت بیشتری داشته و ارتباط حشره باید محیط اطراف را بنحوی تغییر دهد که سایر همنوعانش از تغییر محیط آگاه شده و پیام حشره را دریافت کنند.

## ۳-۱- الگوریتم بهینه‌سازی انبوه ذرات یا PSO (Particle Swarm Optimization)

جیمز کندی روانشناس اجتماعی و راسل سی ابرهارت مهندس برق، صاحبان اصلی ایدهٔ PSO می‌باشند. در این الگوریتم، تعدادی از موجودات (ذره‌ها) در فضای جستجو پخش شده‌اند. هر ذره مقدار تابع هدف را در موقعیتی از فضا که در آن قرار دارد، محاسبه می‌کند. سپس با استفاده از ترکیب اطلاعات محل فعلی‌اش و بهترین محلی که تاکنون در آن بوده و نیز اطلاعات یک یا چند ذره از بهترین ذرات موجود در جمعیت، جهتی را برای حرکت انتخاب می‌کند. همهٔ ذرات با انتخاب جهتی، حرکت کرده و یک مرحله از الگوریتم به پایان می‌رسد. این مراحل تا زمانیکه جواب مورد نظر بدست آید، تکرار می‌شوند. در واقع انبوه ذرات که مقدار کمینهٔ یک تابع را جستجو می‌کنند، همانند دسته‌ای از پرندگان جوینده غذا، عمل می‌کنند. هر ذره (مثلاً $i$اُم) در الگوریتم PSO از سه بردار $d$ بُعدی ($d$ بُعد فضای جستجو) تشکیل شده است:

- $x^i$ موقعیت فعلی ذره
- $v^i$ سرعت حرکت ذره
- $x^{i,best}$ بهترین موقعیتی که ذره تا به حال تجربه کرده است.

در هر مرحله‌ای که الگوریتم تکرار می‌شود، $x^i$ به عنوان یک جواب برای مسأله محاسبه می‌شود. اگر این موقعیت بهتر از جواب‌های پیشین باشد در $x^{i,best}$ ذخیره می‌شود. $f^i$ (مقدار برازش$x^i$) و $f^{i,best}$ (مقدار برازش$x^{i,best}$) نیز از عناصر تشکیل دهندهٔ ذره بحساب می‌آیند. در هر تکرار$x^i$ و$v^i$ جدیدی بدست می‌آیند و هدف از اجرای الگوریتم، بهتر کردن $x^{i,best}$ است.

الگوریتم PSO چیزی فراتر از یک مجموعهٔ ذرات است. هیچ کدام از ذرات قدرت حل هیچ مسأله‌ای را ندارند، بلکه هنگامی می‌توان به حل مسأله امیدوار بود که آنها با همدیگر ارتباط و تعامل داشته باشند. در واقع برای انبوه ذرات، حل مسأله یک مفهوم اجتماعی است که از رفتار تک‌تک ذرات و تعامل میان آنها بوجود می‌آید. بهترین موقعیتی که بوسیله همهٔ ذرات پیدا شده، با $x^{gbest}$ نشان داده می‌شود که از مقایسهٔ مقادیر $f^{i,best}$ همهٔ ذرات انتخاب می‌شود. برازش $x^{gbest}$ بصورت $f^{gbest}$ نشان داده می‌شود.

اگر تعداد ذرات موجود در جمعیت، $n$ باشد، آن گاه می‌توان روابط زیر را بیان کرد:

$$x^{i,best}(t) = arg \min_{\tau \le t} f\left(x^i(\tau)\right) = arg\, min\{f\left(x^i(t)\right), f\left(x^{i,best}(t-1)\right)\} \quad (۱)$$

$$f^{i,best}(t) = f\left(x^{i,best}(t)\right) = \min_{\tau \le t} f^i(\tau) = min\{f^i(t), f^{i,best}(t-1)\} \quad (۲)$$

$$x^{gbest}(t) = arg \min_{i=1,\dots,n} f\left(x^{i,best}(t)\right) \quad (۳)$$

$$f^{gbest}(t) = f(x^{gbest}(t)) = \min_{i=1,\dots,n} f^{i,best}(t) \quad (۴)$$

در ابتدای الگوریتم، ذرات با موقعیت‌ها و سرعت‌های تصادفی ایجاد می‌شوند. درطی اجرای الگوریتم، موقعیت و سرعت هر ذره در مرحله $t+1$اُم از الگوریتم، از روی اطلاعات مرحلهٔ قبلی ساخته می‌شوند. روابطی که سرعت و موقعیت ذرات را تغییر می‌دهند، عبارتند از (اندیس j، مولفهٔ jاُم از بردار را بیان می‌کند):

$$v_j^i(t+1) = wv_j^i(t) + c_1 r_1 \left(x_j^{i,best}(t) - x_j^i(t)\right) + c_2 r_2 \left(x_j^{gbest}(t) - x_j^i(t)\right) \quad (۵)$$

$$x_j^i(t+1) = x_j^i(t) + v_j^i(t+1) \quad (۶)$$

در این روابط، $w$ ضریب اینرسی، $r_1$ و $r_2$ اعدادی تصادفی در بازهٔ $[0,1]$ با توزیع یکنواخت و $c_1$ و $c_2$ ضرایب یادگیری مربوط به تجارب شخصی هر ذره و کل جمعیت هستند. $r_1$ و $r_2$ باعث می‌شوند که نوعی گوناگونی در جواب‌ها به وجود بیاید و به این نحو جستجوی کاملی روی فضا انجام پذیرد. از معادلهٔ (۵) می‌توان دید که هر ذره به هنگام حرکت، (۱) جهت حرکت قبلی خود، (۲) بهترین موقعیتی که در آن قرار داشته و یا (۳) بهترین موقعیتی که بوسیله کل جمعیت تجربه شده، را در نظر می‌گیرد. برخی مواقع، روابط (۵) و (۶) بصورت زیر برای همهٔ مؤلفه‌ها بیان می‌شود:

$$v^i(t+1) = wv^i(t) + F^i(t) \quad (۷)$$

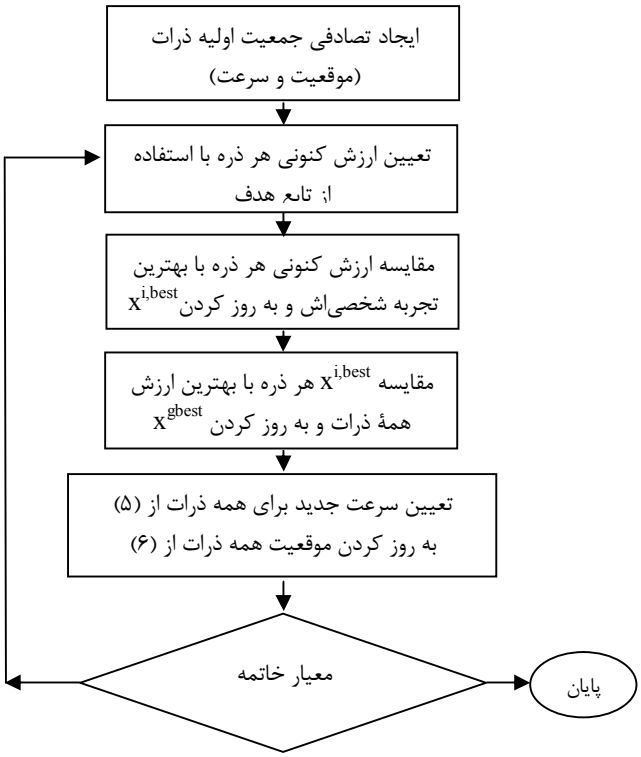
$$F^i(t) = c_1 R_1 \dot{\times} \left(x^{i,best}(t) - x^i(t)\right) + c_2 R_2 \dot{\times} \left(x^{gbest}(t) - x^i(t)\right)$$

$$x^i(t+1) = x^i(t) + v^i(t+1) \quad (۸)$$

که در آن، $R_1$ و $R_2$ دو بردار (هم اندازه با بعد فضای جستجو) از اعداد تصادفی مستقل با توزیع یکنواختند. $F^i(t)$ در (۷)، معادل با یک نیروی خارجی و $\dot{\times}$ نشان دهندهٔ عمل ضرب عضو به عضو دو ماتریس است. بمنظور محدود کردن میزان حرکت هر ذره، مقدار مولفه‌های سرعت ذرات در بازهٔ $[-v_{max}, v_{max}]$ در نظر گرفته می‌شود. شکل ۲ فلوچارت مراحل الگوریتم PSO را که در بالا توضیح داده شد، نشان می‌دهد.

### ۳-۱-۱- پارامترهای PSO

ضریب اینرسی w بر روی همگرایی الگوریتم PSO تأثیر مستقیم دارد. در واقع می‌توان بواسطهٔ این ضریب، تأثیر سرعت‌های گذشته را بر سرعت‌های زمان حال کنترل نمود. می‌توان برای برقراری موازنهٔ بهتر میان جستجوی سراسری و جستجوی محلی مقدار w را تغییر داد. مقدار زیاد برای w باعث می‌شود که ذرات موجود در الگوریتم، به جستجوی مناطق جدیدتر روی آورده و یک جستجوی سراسری را انجام دهند.




شکل ۲- الگوریتم بهینه‌سازی انبوه ذرات

در مقابل یک مقدار کم برای w باعث می‌شود که ذرات در منطقهٔ محدودی مانده و یک جستجوی محلی انجام دهند. جستجوی محلی برای دقیق‌تر کردن جواب‌های فعلی مناسب است و جستجوی سراسری برای یافتن جواب‌های بهتری که احتمالاً در نواحی ناشناخته از فضای جستجو موجودند، بکار می‌رود.

تغییرات سرعت ذره، یا شتابی که به ذره وارد می‌شود، به این صورت قابل محاسبه است:

$$\Delta v^i(t+1) = v^i(t+1) - v^i(t) = F^i(t) - (1-w)v^i(t) \qquad (۹)$$

در واقع $(1-w)$ بعنوان یک ضریب اصطکاک عمل می‌کند (w ضریب سیالی محیطی است که ذره در آن حرکت می‌کند). اگر معادلهٔ (۷) بصورت معادلهٔ فضای حالت یک سیستم گسسته فرض شود، $w \geq 1$ باعث ناپایداری سیستم ذرات می‌شود. از طرفی به ازای مقادیر کمتر w، سرعت همگرایی سامانه بیشتر خواهد شد. یک مقدار مناسب برای w، باعث ایجاد تعادل بین جستجوی‌های محلی و سراسری می‌شود و نیز کاهش تعداد تکرارهای لازم برای همگرایی به یک جواب مناسب، می‌شود. در الگوریتم PSO ابتدایی، مقدار w ثابت در نظر گرفته می‌شد. اما نتایج عملی حاکی از آنند که بهتر است مقدار w در مراحل ابتدایی، یک مقدار بزرگ در نظر گرفته شود تا یک جستجوی کامل و سراسری از فضای جستجو صورت گیرد؛ سپس در طی مراحل اجرای الگوریتم، مقدار w بتدریج کاهش داده می‌شود تا الگوریتم به مرز همگرایی نزدیک شده و جواب‌های دقیق‌تری بدست آید. مفهوم w را بصورت ضریب سیالی محیط بیاد آورید: مقدار بیشتر w، بمعنی راحت‌تر بودن حرکت در محیط (محیط دارای گران‌روی کمتر) است؛ با کمتر شدن مقدار w، حرکت ذرات

در محیط سخت‌تر (گران‌روی محیط بیشتر) و امکان همگرایی ذرات بسمت نقاط بهینه بیشتر می‌شود. معمولاً w=۱/۲ و کاهش دادن تدریجی آن برای بیشتر مسایل مناسب است. همچنین انتخاب w بصورت تصادفی (توزیع یکنواخت) در بازهٔ [۰/۵ تا ۱] در برخی موارد نتایج خوبی بهمراه داشته است. عملکرد ضریب اینرسی در الگوریتم PSO همانند عملکرد دما در روش شبیه‌ساز سرد کردن فلزات در فصل دوم است؛ دماهای بالاتر در این الگوریتم باعث آزادی عمل بیشتر اتم‌ها (جواب‌های مسأله) شده و بتدریج که از دما کاسته می‌شود، تغییرات کمتری از طرف الگوریتم پذیرفته می‌شود.

توجه شود که اگر الگوریتم بدون در نظر گرفتن محدودیت‌های سرعت اجرا شود، در چند تکرار، سرعت ذرات بشدت افزایش یافته و به مقادیر غیر قابل قبول می‌رسد. کندی نشان داد که برای ذرات تک-بُعدی که بصورت غیر تصادفی حرکت می‌کنند، اگر مقدار $c_1 + c_2$ بین ۰ تا ۴ باشد، مسیرهایی که ذرات طی می‌کنند، قابل قبول‌ترند. کلرک و کندی نشان دادند که راه‌های زیادی برای تعیین مقادیر ضرایب یادگیری وجود دارد؛ یکی از ساده‌ترین روش‌های تعیین مقادیر مناسب برای ضرایب یادگیری، توسط رابطهٔ زیر پیشنهاد می‌شود:

$$w=x \qquad c_1 = x\emptyset_1 \qquad c_2 = x\emptyset_2 \qquad (۱۰)$$

که در آن، $\emptyset_1$ و $\emptyset_2$ اعدادی مثبت بوده و بنحوی انتخاب می‌شوند که $\emptyset = \emptyset_1 + \emptyset_2 \geq 4$ باشد. x نیز از روی رابطه زیر تعیین می‌شود:

$$x = \frac{2}{\emptyset - 2 + \sqrt{\emptyset^2 - 4\emptyset}} \qquad (۱۱)$$

کلرک مقادیر $\emptyset_1 = \emptyset_2$ را پیشنهاد می‌کند ($w \simeq 0.7298$، $c_1 = c_2 \simeq 1.4962$). با استفاده از روابط فوق، ذرات بدون نیاز به کمیت محدود کننده $v_{max}$ همگرا می‌شوند.

### ۳-۱-۲- الگوی بهینه محلی

در کل می‌توان دو الگوی مختلف برای PSO پیشنهاد داد: الگوی بهینه سراسری و الگوی بهینه محلی. الگوی بهینه سراسری، چیزیست که تاکنون در این نوشتار به بررسی آن پرداخته شد. در استخراج الگوی بهینه محلی، الگوریتم PSO را می‌توان بشکل مجموعه‌ای از بردارها تصور کرد که در فضایی متشکل از تجارب خصوصی هر ذره و برخی از ذرات دیگر، نوسان می‌کند. در حالت کلی، هر ذره با عده‌ای دیگر از ذرات دارای ارتباط است که بطور معمول این ارتباط دو طرفه می‌باشد (بنام رابطه همسایگی یا مجاورت شناخته می‌شود). مجموعهٔ ذراتی که با یک ذره دارای ارتباط همسایگی‌اند، بنام مجموعهٔ همسایگی شناخته می‌شوند

(مجموعهٔ همسایگی یک ذره، شامل خود آن ذره نیز می‌شود). برای ذرهٔ iام، بهترین موقعیتی که بوسیله همسایگانش تجربه شده، بصورت $x^{i,nbest}$ نمایش داده می‌شود. حال اگر در روابط (۵) و (۷) به جای $x^{gbest}$ از $x^{i,nbest}$ استفاده شود، الگوی بهینه محلی بدست می‌آید.

در الگوی سراسری، یک ذره در جمع وجود دارد که بعنوان جاذب، سایر ذرات را بخود جذب کرده و احتمالاً همهٔ ذرات به محل بهترین ذره، همگرا می‌شوند. اما در الگوی بهینه محلی، چندین جاذب در جمع وجود داشته و هر ذره فقط بسمت بهترین همسایهٔ خود جذب می‌شود. اگر دامنهٔ تعریف همسایگی به همهٔ جمع توسعه یابد، آنگاه الگوی بهینه محلی با الگوی بهینه سراسری معادل خواهد بود. برای مسایل ساده که جواب یکتا دارند، الگوی بهینه سراسری همیشه توصیه می‌شود. با اینحال برای توابع پیچیده، الگوی بهینه محلی با تعریف همسایگی متغیر مناسب است. در حالت کلی روابط همسایگی را می‌توان به دو نوع اصلی تقسیم کرد:

- همسایگی مبتنی بر فاصله
- همسایگی مبتنی بر ارتباط‌های اجتماعی

در همسایگی مبتنی بر فاصله، به طور معمول فاصلهٔ هندسی یا اقلیدسی دو ذره در فضای جستجو، مشخص می‌کند که آیا دو ذره با هم همسایه‌اند یا خیر. در همسایگی مبتنی بر روابط اجتماعی، همسایگی فقط به معنی نزدیکی و یا دوری فیزیکی نمی‌باشد. در این نوع همسایگی، الگوی خاصی تعیین کننده همسایگی دو ذره می‌باشد. اشکال مختلفی از همسایگی‌های ممکن، بصورت گراف‌هایی در شکل ۳ نمایش داده شده‌اند. اگر دو ذره همسایه باشند، یالی از گراف آن دو را به هم وصل می‌کند. اگر همسایگی بصورت سراسری تعریف شود، آنگاه مطابق شکل ۳-الف همهٔ رأس‌های گراف همسایگی به همدیگر متصل هستند (گراف کامل). همسایگی فون نیومن، نوع خاصی از همسایگی است که در آن هر ذره با یک ذره در هر کدام از جهت‌های بالا، پایین، چپ و راست همسایه است (شکل ۳-ب). شکل ۳-پ همین همسایگی را بصورت دو بُعدی نشان می‌دهد. شکل ۳-ت نیز همسایگی موسوم به هرمی را نشان می‌دهد. یکی از معمول‌ترین تعاریف مورد استفاده برای همسایگی، تعریف همسایگی حلقوی است (شکل ۳-ث).در همسایگی ستاره‌ای نیز همهٔ ذرات به ذره‌ای بنام رهبر متصلند که اطلاعات سایرین بوسیله او پردازش می‌شود (شکل ۳-ج).

(الف) (ب) (پ) (ت) (ث) (ج)

شکل ۳- اشکال مختلف همسایگی، الف) گراف کامل، ب) فون نیومن دو بعدی، پ) فون نیومن سه بعدی، ت)

### ۳-۱-۳- الگوریتم PSO با اطلاعات کامل (FIPS)

مندس نحوهٔ ارتباط ذرات با همسایگان‌شان را تغییر داده و روابط جدیدی را برای حرکت ذرات در فضای جستجو بوجود آورد و آنرا الگوریتم PSO با اطلاعات کامل (FIPS) نامید. در این الگوریتم، هر ذره از اطلاعات همهٔ همسایگانش برای تعیین جهت حرکت بهره می‌گیرد. روابط مربوط به FIPS عبارتند از:

$$v^i(t+1) = wv^i(t) + \frac{c}{n_i}\sum_{j\in N_i} R_j \otimes (x^{j,best}(t) - x^i(t)) \quad (۱۲)$$

$$x^i(t+1) = x^i(t) + v^i([t+1]) \quad (۱۳)$$

که در آن $N_i$ مجموعهٔ همسایه‌ها و $n_i$ تعداد همسایه‌های ذرهٔ iام هستند. $R_j$ بردار اعداد تصادفی (با توزیع یکنواخت) در بازهٔ [0,1] است. اگر در روابط بالا، علاوه بر خود ذره تنها اطلاعات بهترین ذره بکار گرفته شود، آنگاه FIPS به PSO تبدیل می‌شود. با پارامترهای مناسب، FIPS جواب‌های بهتری در تعداد تکرار کمتر نسبت به PSO بدست می‌آورد؛ البته عملکردش بشدت به نحوه تعریف همسایگی وابسته است.

### ۳-۱-۴- الگوریتم PSO دودویی

در این نسخه از PSO که به وسیله کندی و ابرهارت در سال ۱۹۹۷ معرفی گردید، تغییری کوچک بر روی الگوریتم PSO اولیه انجام گرفته است، تا بتوان برای بهینه کردن کمیت‌های گسسته نیز از آن استفاده کرد. در این الگوریتم، سرعت بعنوان یک مقدار آستانهٔ احتمالی استفاده شده است که معین می‌کند که مولفه‌ای از بردار موقعیت ۰ یا ۱ باشد؛ بعبارت بهتر اگر σ عددی تصادفی (با توزیع یکنواخت) در بازهٔ [0,1] باشد، مقدار بیت jام از بردار (دودویی) موقعیت ذرهٔ iام، بصورت زیر تعیین می‌گردد:

$$x_j^i(t) = \begin{cases} 1 & ,\sigma < F\left(x_j^i(t-1), v_j^i[t], x_j^{i,best}(t-1), x_j^{gbest}(t-1)\right) \\ 0 & , \quad otherwise \end{cases} \quad (۱۴)$$

در این روابط، F می‌تواند هر تابعی باشد که مقداری بین [۰ و ۱] تولید می‌کند. یک انتخاب بسیار ساده برای F، تابع سیگموید است؛ در اینصورت رابطهٔ (۱۴) را می‌توان بصورت زیر نوشت:

$$x_j^i(t) = \begin{cases} 1 & ,\sigma < s\left(v_j^i(t)\right) \\ 0 & ,otherwise \end{cases} \quad (۱۵)$$

S(.) تابع سیگموید: $s$

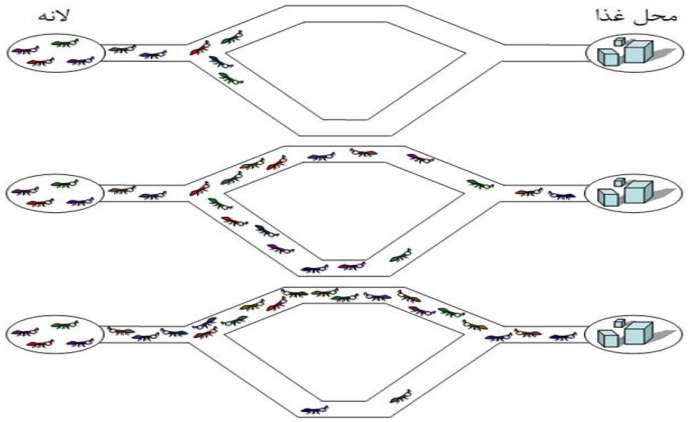
در PSO دودویی، سرعت ذرات همانند الگوریتم PSO استاندارد تغییر داده می‌شوند.

این الگوریتم، در بسیاری موارد بسیار قدرتمندتر و سریع‌تر از الگوریتم‌های ژنتیک دودویی عمل می‌کند.

## ۳-۲- الگوریتم بهینه‌سازی مورچه‌ها یا ACO (Ant Colony Optimization)

یکی از جالب‌ترین و کارآمدترین الگوهای بهینه‌سازی طبیعی، جستجوی غذا توسط مورچه‌هاست. ارتباط بین مورچه‌های یک کلونی، توسط ریزش ماده‌ای شیمیایی به نام فرومون بر روی مسیر حرکت است. به این نحو که مورچه‌ها طی حرکت، دنباله‌ای از فرومون را ترسیم کرده و همواره مشتاق دنبال کردن مسیرهای با فرومون بیشتر هستند. تغییر محیط بمنظور ایجاد تغییرات در رفتار (از طریق ارتباط بوجود آمده)، به اصل استیگمرجی (از ترکیب دو واژهٔ یونانی استیگما بمعنی اشاره و ارگون به معنی کار) معروف است. این اصطلاح برای اولین بار در سال ۱۹۵۹ بوسیله زیست شناس فرانسوی پیر پول گراسه برای توضیح رفتار موریانه‌ها بکار برده شد. استیگمرجی پایهٔ اصلی بسیاری از حرکت‌های حشرات بخصوص مورچه‌هاست. در جوامع مورچه‌ای، یک مورچهٔ خاص بنام ملکه فقط مسئولیت تخم‌گذاری را دارد، درحالیکه سایر مورچه‌ها عملکرد کاملاً متفاوتی دارند. مورچه‌های یک مجموعه، خود- ترتیب بوده و رفتارهای پیچیدهٔ مجموعه ناشی از رفتارهای سادهٔ تک‌تک مورچه‌هاست که بصورت خود- ترتیب انجام می‌دهند. خواص جمعی و فردی مورچه‌ها بنحویند که بر روی مسایل مختلف کارآیی مناسبی دارند؛ بخصوص، اگر در بازهٔ زمانی خاص، برخی از مورچه‌ها عملکرد مناسبی نداشته باشند، با اینحال عملکرد کلی مجموعه مناسب خواهد بود.

مورچهٔ آرژانتینی نابینا قادر نیست کوتاه‌ترین مسیر را تشخیص دهد. اما آزمایش نشان می‌دهد که توده‌ای از مورچه‌ها با شروع از محل لانه، پس از مدتی کوتاه‌ترین مسیر بین لانه و غذا را بر می‌گزینند (شکل ۴).




شکل ۴- رفتار مورچه‌های آرژانتینی

الگوریتم بهینه‌سازی کُلونی مورچه‌ها (به اختصار الگوریتم مورچه‌ها)، از رفتار مورچه‌های طبیعی که در مجموعه‌های بزرگ در کنار هم زندگی می‌کنند، الهام گرفته شده است. الگوریتم مورچه‌ها، یک مثال بارز از هوش جمعی است که در آن عامل‌هایی که قابلیت چندان بالایی ندارند، در کنار و با همکاری هم می‌توانند نتایج بسیار خوبی بدست آورند. این الگوریتم برای حل محدودهٔ وسیعی از مسایل بهینه‌سازی بکار می‌رود.

### ۳-۲-۱- الگوریتم ساده شده مورچه‌ها

در این بخش یک الگوریتم ساده و یک الگوی کلی از الگوریتم مورچه‌ها ارائه می‌شود. در این الگوریتم، هدف اصلی هر مورچه، یافتن کوتاه‌ترین مسیر بین دو رأس از گرافی است که مسأله مورد بررسی بر روی آن تعریف شده است. فرض کنید G یک گراف معمولی باشد. با استفاده از الگوریتم ساده شده مورچه‌ها، می‌توان کوتاه‌ترین مسیر موجود بین رأس مبدأ s تا رأس مقصد d را پیدا کرد (طول مسیر=تعداد یال‌های طی شده). یالی که رأس i را به رأس j متصل می‌کند، با $L_{ij}$ نشان داده شده و برای آن کمیتی بنام دنبالهٔ فرمونی (میزان فرومون) $\tau_{ij}$ در نظر گرفته می‌شود. میزان فرومون بوسیله مورچه‌ها خوانده و یا تغییر داده می‌شود. مقدار و غلظت فرومون موجود بر روی یک یال، بعنوان معیاری برای مطلوب بودن آن یال برای انتخاب توسط مورچه‌هاست. در ابتدای الگوریتم، همه یال‌ها دارای مقداری مساوی از فرومون ($\tau_0$) هستند. هر مورچه روندی قدم به قدم را برای ایجاد مسیرهای حرکتی بکار می‌گیرد. اطلاعات محلی که در رأس گراف و در یال‌های خارج شونده از آن نگهداری می‌شود، برای انتخاب تصادفی مقصد بعدی استفاده می‌شود. هنگامی که مورچهٔ k در رأس i قرار دارد، احتمال آن که رأس j را بعنوان مقصد بعدی انتخاب کند، برابرست با:

$$P_{i\to j}^{k} = P_{ij}^{k} = \begin{cases} \frac{\tau_{ij}}{\sum \tau_{im}} & j \in N_i \\ 0 & j \notin N_i \end{cases} \qquad (۱۶)$$

که در آن، $N_i$ مجموعه رأس‌هایی از گراف است که در همسایگی رأس iام قرار دارند (به رأس iام متصلند).

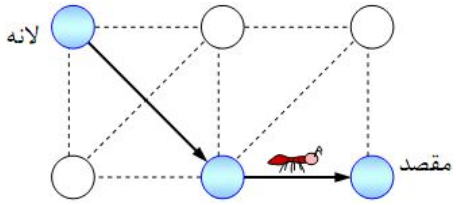
هنگامی که مورچه‌ای از یالی عبور می‌کند، مقداری فرومون نیز در آن یال می‌ریزد. این مقدار در الگوریتم ساده شده، ثابت و برابر $\Delta\tau$ در نظر گرفته می‌شود. اگر مورچه‌ای در زمان t از روی یال بین رأس‌های i و j عبور کند، مقدار فرومون آن یال را بصورت زیر تغییر خواهد داد:

$$\tau_{ij}(t) \leftarrow \tau_{ij}(t) + \Delta\tau \qquad (۱۷)$$

مورچه‌ها با این روش رغبت مورچه‌های دیگر را برای انتخاب مسیری که خود طی کرده‌اند، بیشتر کرده و احتمال انتخاب آن را در آینده افزایش می‌دهند. برای جلوگیری از همگرا شدن مورچه‌ها به یک مسیر غیر بهینه، می‌بایست جستجوی کافی در بین مسیرهای ممکن انجام شود. برای این منظور، همانند فرومون‌های واقعی، فرومون روی یال‌های گراف تبخیر می‌شود. مقدار فرومون یال‌ها بصورت خودکار کاهش یافته تا علاقهٔ مورچه‌ها برای جستجوی مسیرهای جدید بیشتر شود. فرآیند تبخیر بطور معمول بصورت یک تابع کاهشی ساده در نظر گرفته می‌شود. یک شیوه برای کاهش فرومون، استفاده از تابع نمایی کاهشی است؛ بدین ترتیب که مقدار فرومون در هر مرحله، در یک عدد مثبت و کمتر از یک ضرب می‌شود ($e^{-P}\approx 1-P$):

$$\tau \leftarrow (1-P)\tau \quad ,P \in [0,1] \qquad (۱۸)$$

که P عددی موسوم به ضریب تبخیر است. در نمونه‌های پیچیده، مقدار P در مراحل ابتدایی اجرای الگوریتم مقدار زیادی است که بمرور کم می‌شود. اعمال این الگوریتم ساده بر گراف زیر، عملکرد خوب آنرا نشان می‌دهد. با اینحال برای مسأئل پیچیده، رفتار الگوریتم ناپایدار و بشدت به مقادیر پارامترها حساس می‌باشد.




شکل ۵- مسیر بهینه‌ای که بوسیلهٔ مورچه‌ها در یک گراف ساده پیدا شده است

### ۳-۲-۲ الگوریتم مورچه‌ها

الگوریتم ساده شده مورچه‌ها، بدلیل سادگی بیش از حدش دارای محدودیت‌هایی بوده و برای برخورد با مسائل پیچیده مناسب نیست. برای حل مسائل چند هدفه، مورچه‌ها باید دارای نوعی حافظه باشند. در این بخش، الگوریتم کامل معرفی می‌شود که علاوه بر داشتن خواص ابتدایی الگوریتم ساده، برای برخورد با مسایل پیچیده، بهینه شده است (محدودیت‌های الگوریتم ساده در آن وجود ندارند). بعنوان مثال، مقدار فرومونی که هر مورچه روی یال‌ها می‌ریزد، متناسب با کیفیت جواب‌هایی است که یافته است. لذا اطلاعاتی که بوسیله فرومون‌ها بوجود می‌آیند، بمراتب کارآتر بوده و در جهتدهی مناسب جستجوی مورچه‌ها تأثیر بهتری دارد. مسایل گسسته‌ای که الگوریتم مورچه برای حل آنها مناسب است، دارای خواص زیر هستند:

- یک مجموعه متناهی از عناصر بصورت $C = \{c_1, c_2, \dots, c_{N_c}\}$ داده شده باشد.
- مجموعه متناهی از ارتباط‌های ممکن بین اعضای C بصورت $L = \{l_{c_i c_j} | (c_i, c_j) \in \tilde{C} \subseteq C \times C\}$ تعریف شده باشد (رابطهٔ $|L| \leq N_C^2$ همواره برقرار است؛ $|L|$ طول یا تعداد اعضای دنباله $L$ می‌باشد).
- برای هر ارتباط $l_{cicj}$، یک تابع هزینه بصورت $I_{cicj} = J(l_{cicj}, t)$ تعریف شده باشد.
- مجموعه متناهی از قیود $\Omega=\Omega(C,L,t)$ بر روی اعضای مجموعه‌های C و L تعریف شده باشد.
- حالت‌های مسأله بصورت دنباله‌هایی از اعضای C یا L تعریف شده باشند (مثلاً دنباله $s_k = \langle c_1, c_5, c_7, c_{10} \rangle$، می‌تواند یک حالت باشد). اگر S مجموعه تمام حالات قابل تعریف باشد، حالاتی که قیود $\Omega(C,L,t)$ را برآورده می‌سازند (حالات شدنی)، با $\overline{S}$ نمایش داده می‌شوند..
- یک ساختار همسایگی بصورت واقعی قابل تعریف باشد. حالت $s_2$ همسایه یا مجاور حالت $s_1$ شناخته می‌شود اگر $s_1$ و $s_2$ هر دو در مجموعه حالات S بوده و بتوان با یک حرکت حالت $s_1$ را بحالت $s_2$

تبدیل نمود. بعبارت دیگر اگر $c_1$ آخرین عضو دنبالهٔ $s_1$ باشد، می‌باید عضوی از C بنام $c_2$ وجود داشته باشد، بنحوی که مسیری بین $c_1$ و $c_2$ وجود داشته ($I_{cicj} \in L$) و همچنین $s_2 \equiv \langle s_1, c_2 \rangle$ باشد.

- پاسخ مسأله (منظور پاسخ بهینه نیست) بصورت عضوی از $\overline{S}$ قابل تعریف باشد. یک پاسخ، چند بعدی خوانده می‌شود، اگر بصورت مجموعه‌ای از دنباله‌های متمایز روی اعضای C تعریف شده باشد.
- متناظر با هر پاسخ $\Psi$، هزینه‌ای $J_\Psi(L,t)$ (هزینهٔ تمام ارتباط‌های سازندهٔ $\Psi$) تعریف شده باشد.

فرض کنید $G=(C,L)$ گراف مسأله بهینه‌سازی گسسته‌ای باشد که بصورت بالا تعریف شده‌است (اعضای C گره /رأس و اعضای مجموعه L ارتباط/یال خوانده می‌شوند). پاسخ‌های مسأله بهینه‌سازی بصورت مسیرهای شدنی گراف قابل بیان‌اند. الگوریتم مورچه می‌تواند برای یافتن پاسخ‌هایی که قیود $\Omega$ را برآورده سازند، بکار برده رود. در الگوریتم مورچه‌ها، یک کلونی از مورچه‌ها، در کنار هم مسأله بهینه‌سازی گسسته‌ای را (با تعریف آن بصورت گراف) حل می‌کنند. اطلاعاتی که مورچه‌ها درهنگام جستجو بدست می‌آورند، در قالب مقدار فرومون $\tau_{ij}$ روی یال $l_{ij}=l_{cicj}$ ذخیره می‌شود (در واقع فرومون حافظه‌ای بلندمدت برای ذخیرهٔ اطلاعات مورچه‌هاست). بسته به مسأله، دنباله‌های فرومونی، می‌توانند روی همه یا تعداد خاصی از یال‌های گراف مسأله ایجاد شوند. در برخی مسایل، اطلاعاتی اولیه با نام ارزش ذهنی یا *Heuristic value* ($\eta_{ij}$)، برای هر یال ($l_{ij}$) تعریف می‌شود؛ این اطلاعات توسط طراح مسأله برای استفادهٔ مورچه‌ها در اختیار قرار می‌گیرد.

کلونی مورچه‌ها دارای خواص عمومی زیر است:

- هر مورچه برای یافتن یک راه حل برای مسأله، بحد کافی تواناست (احتمالاً جواب خوبی نیز خواهد یافت)، اما راه حل‌های دارای کیفیت مطلوب، تنها با همکاری و تعامل بین مورچه‌ها قابل دستیابی‌اند.
- هر مورچه فقط از اطلاعات فردی خود و یا اطلاعات محلی گرهی که در آن قرار دارد بهره می‌برد.
- مورچه‌ها بصورت غیرمستقیم، از طریق تغییر در مقدار فرومون مسیرها، با یکدیگر در ارتباط‌اند.

همچنین مورچه‌های موجود در کلونی نیز دارای خواص زیرند:

- یک مورچه بدنبال راه حلی می‌گردد که دارای کمترین هزینه باشد: $J = min_\Psi J_\Psi(L,t)$
- مورچه k دارای حافظه‌ای شخصی $M_k$ است که مسیر عبوری‌اش را در آن ذخیره می‌کند. حافظه برای ایجاد راه‌حل‌های مجاز، ارزیابی راه‌حل‌های پیدا شده و بازگشت معکوس مسیر استفاده می‌شود.
- مورچه k که در حالت $s_r = \langle s_{r-1}, i \rangle$ قرار دارد، می‌تواند به گرهی مانند j برود که عضو مجموعه است: $N_i^k \triangleq \{j | (j \in N_i), (\langle s_r, j \rangle \in \tilde{S})\}$. $N_i$ مجموعه گره‌های همسایگی گره i است: $N_i \triangleq \{j | J_{ij} \in L\}$.

- مورچهٔ kام از یک حالت اولیه $s_o^k$ شروع به کار می‌کند.
- برای مورچهٔ kام، مجموعه شرایط خاتمهٔ مخصوصی بصورت $e^k$ تعریف شده است.
- مورچه‌ها از حالت اولیهٔ خودشان شروع بحرکت کرده و در هر حرکت به یکی از حالت‌های مجاور (که شدنی و مجاز باشد) می‌روند. این کار باعث بوجود آمدن یک راه حل شدنی برای مسأله می‌شود. حرکت یک مورچه تا جایی ادامه می‌یابد که حداقل یکی از شرایط خاتمهٔ $e^k$ برآورده شود.
- مورچهٔ kام که در گره iام قرار دارد، می‌تواند به یکی از گره‌هایی که در $N_i^k$ قرار دارد، حرکت کند. انتخاب مقصد، با توجه به یک قانون احتمالی مشخص انجام می‌گیرد.
- قانون احتمالی مورچه‌ها تابعی از قیود مسأله، حافظهٔ هر مورچه و اطلاعات محلی ذخیره شده در هر گره است. اطلاعات محلی هر گره از ترکیب اطلاعات فرومونی $\tau_{ij}$ و اطلاعات ذهنی $\eta_{ij}$ بوجود می‌آید. این اطلاعات بصورت سازمان یافته در جدولی موسوم به جدول مسیریابی ذخیره می‌گردند.
- هر مورچه، در هنگام حرکت از گره i بسمت گره j، مقدار فرومون $\tau_{ij}$ روی یال $J_{ij}$ را تغییر می‌دهد. این کار، تجدید گام به گام و آنی فرومون نامیده می‌شود.
- اگر مورچه‌ای مسیری را که یافته، رو به عقب برگردد و فرومون یال‌های تشکیل دهندهٔ آن را عوض کند، این عمل تجدید تأخیری فرومون نامیده می‌شود.

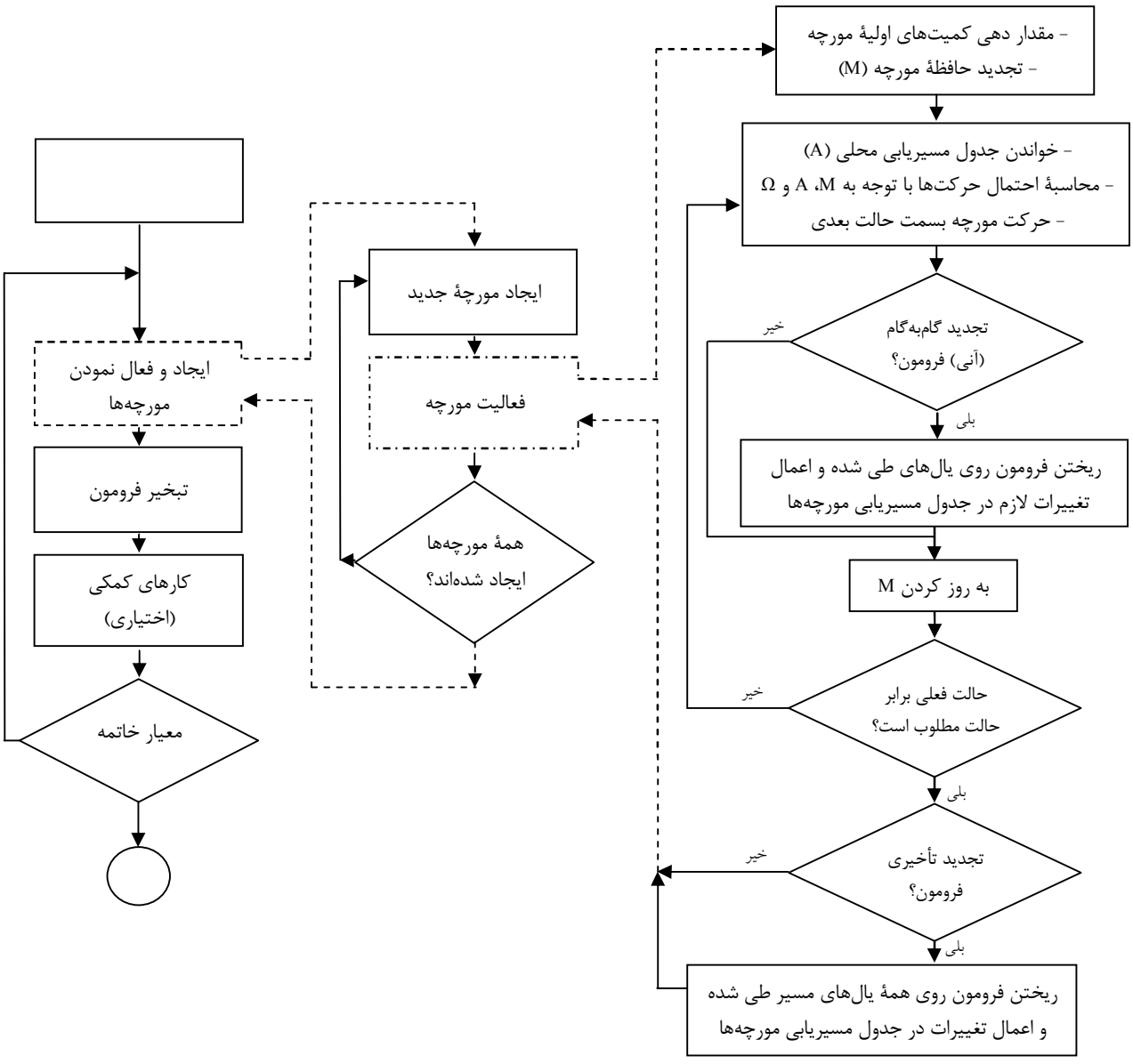
به زبان ساده‌تر، فعالیت مورچه‌ها را می‌توان بصورت زیر جمع‌بندی کرد:

کلونی مورچه‌ها بصورت همزمان اما مستقل روی گراف مسأله حرکت می‌کنند. نحوهٔ حرکت مورچه‌ها بوسیله یک قانون تصادفی (حاصل از اطلاعات محلی ذخیره شده در رأس‌ها) مشخص می‌شود. مورچه‌ها با حرکت‌های خود، راه‌حل‌هایی را برای مسأله بهینه‌سازی ساخته و ارزیابی می‌کنند. مورچه کیفیت جواب خود را با ریزش فرومون روی مسیرش، به سایرین خبر داده و جستجوی مورچه‌های بعدی را جهت خواهد داد.

در کنار فعالیت جمعی مورچه‌ها برای یافتن جواب بهینه، الگوریتم مورچه‌ها نیز می‌تواند شامل دو عمل اضافی دیگر باشد: تبخیر فرومون و کارهای کمکی. تبخیر فرومون باعث می‌شود که میزان فرومون یال‌های گراف با سپری شدن زمان بصورت خودکار کمتر شود. این عمل باعث می‌شود که از همگرایی سریع الگوریتم به یک جواب نامناسب، جلوگیری بعمل آید. در واقع تبخیر فرومون، فراموش‌کاری مورچه‌ها را پیاده کرده و باعث می‌شود که آنها نواحی جدیدی را برای یافتن جواب‌های بهینه‌تر جستجو کنند. کارهای کمکی، کارهای متمرکزی هستند که بوسیله هیچ یک از مورچه‌ها قابل انجام نمی‌باشند. بعنوان مثال، استفاده از یک

الگوریتم بهینه‌سازی محلی دیگر برای بهتر کردن نسبی نتیجه، از جمله کارهای کمکی است. مثال دیگر، ریزش مقداری فرومون اضافی بر روی کوتاه‌ترین مسیر یافته شده بوسیله مورچه‌هاست. تغییر مقدار فرومون یال‌ها، بوسیله کارهای کمکی را تجدید فرومون غیر آنی یا خارج از خط می‌نامند.

شکل ۶، فهرست عملیاتی را در یک مرحله از الگوریتم مورچه‌ها انجام می‌شود، نشان داده است.




شکل ۶- فلوچارت مراحل اجرای الگوریتم مورچه‌ها (در یک مرحله از اجرای آن)