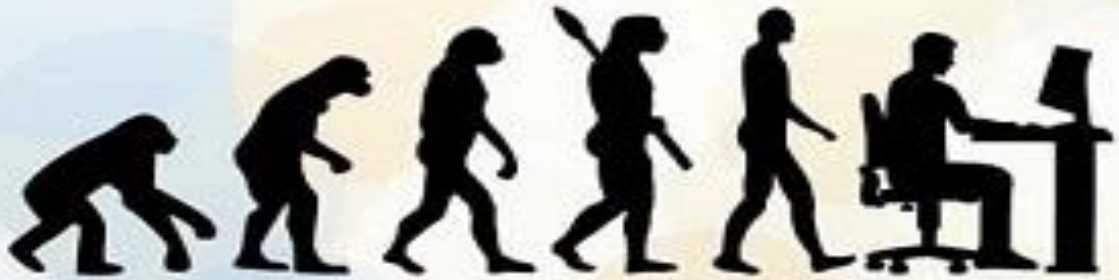
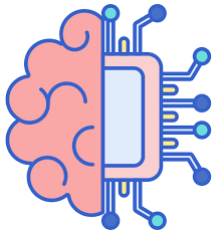
هوش محاسباتی

Computational Intelligence

میلاد سلطانی





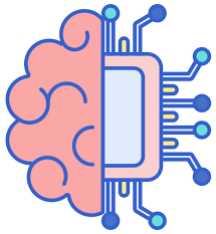
فهرست مطالب فصل چهارم

- مقدمه
- محاسبات تکاملی
- نظریه تکامل داروین
- مؤلفه های اصلی الگوریتم تکاملی
- انتخاب طبیعی، فنوتایپ و ژنوتایپ
- ساختار و مراحل اجرای الگوریتم تکاملی
- نمونه الگوریتم های تکاملی و بهینه سازی
- الگوریتم ژنتیک

فصل چهارم

محاسبات تکاملی

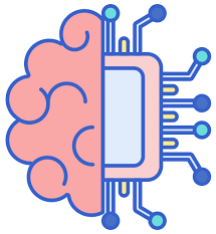
Evolutionary Computing



تعریف محاسبات تکاملی (Evolutionary Computing)

- محاسبات تکاملی (EC) یک زیر شاخه از هوش مصنوعی و علوم کامپیوتر است که از اصول تکامل طبیعی برای حل مسائل پیچیده استفاده می‌کند. این روش‌ها الگوریتم‌هایی هستند که با الهام از فرآیندهای زیستی، مانند انتخاب طبیعی، تولید مثل و جهش به یافتن راه حل های بهینه یا نزدیک به بهینه برای مسائل مختلف می‌پردازند.
  - الگوریتم‌های محاسبات تکاملی بر پایه نظریه تکامل داروین (Darwin's Theory of Evolution) جهت پیاده‌سازی برنامه‌های کامپیوتری شکل گرفته‌اند.

- محاسبات تکاملی معمولاً برای مسائل بهینه‌سازی (Optimization) و جستجو (Search) استفاده می‌شود؛ بخصوص در مواردی که فضای جستجو بزرگ، پیچیده یا غیرخطی است و روش‌های کلاسیک کارایی کافی ندارند.



محاسبات تکاملی

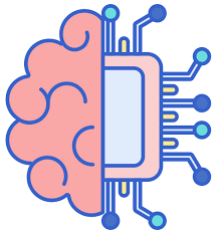
- الگوریتم‌های محاسبات تکاملی خانواده‌ای از روش‌های حل مسأله هستند که:
  - مبتنی بر جمعیت (Population-based) و آزمون و خطا (Trial and Error) بوده
  - از بهینه‌سازی تصادفی (Stochastic Optimization) یا بهینه‌سازی فرا اکتشافی (Meta-Heuristic) جهت همگرایی به جواب بهینه سراسری یا تقریبی از جواب بهینه استفاده می‌کنند.

- بر اساس مکانیزم تعریف شده در فرآیند تکامل، جمعیت متشکل از جواب‌های کاندید به نحوی تکامل پیدا می‌کنند که با مرور زمان و گذر نسل‌های متوالی، برازندگی (Fitness) آنها افزایش پیدا کند.

تاریخچه محاسبات تکاملی (دهه ۱۹۵۰)

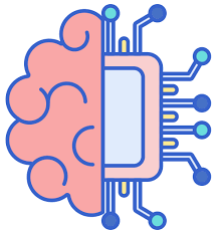
ایده‌های اولیه

* محاسبات تکاملی از ایده‌های داروینیسم الهام گرفت. اولین مفاهیم:
  - در اواخر دهه ۱۹۵۰، جان هالند (John Holland) ایده استفاده از اصول تکامل برای حل مسائل بهینه‌سازی را مطرح کرد. او بعدها در دهه ۱۹۷۰، الگوریتم ژنتیک (Genetic Algorithm) را توسعه داد.
  - مفاهیم اولیه‌ی الگوریتم‌های تکاملی (Evolution Strategies) در دهه ۱۹۵۰ در آلمان معرفی شد.



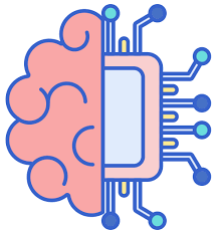
تاریخچه محاسبات تکاملی (دهه ۱۹۶۰)

- **توسعه رسمی الگوریتم‌ها**
  - الگوریتم ژنتیک (GA)
    - در این زمان، هالند و همکارانش در دانشگاه میشیگان تحقیقاتی را در مورد استفاده از الگوریتم‌های تکاملی انجام دادند.
  - استراتژی‌های تکاملی (ES)
    - این الگوریتم‌ها توسط رایشِنبِرگ و هانس-پاول شُوایتزِر در آلمان توسعه یافتند.
    - این روش بیشتر بر بهینه‌سازی پارامترها در مسائل مهندسی تمرکز داشت.



تاریخچه محاسبات تکاملی (دهه ۱۹۸۰–۱۹۷۰)

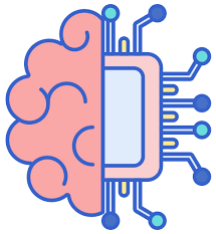
- **دهه ۱۹۷۰ – برنامه‌نویسی تکاملی (Evolutionary Programming)**
  - لارنس فوگل (Lawrence Fogel)
    - مفهوم برنامه‌نویسی تکاملی را معرفی کرد. این روش بر مدلسازی فرآیندهای یادگیری در سیستم‌های خودکار تأکید داشت.
- **دهه ۱۹۸۰ – گسترش الگوریتم‌ها**
  - توسعه نرم‌افزارهای اولیه:
    - الگوریتم‌های تکاملی برای حل مسائل پیچیده در مهندسی، علوم زیستی و اقتصاد مورد استفاده قرار گرفتند.
  - مفهوم الگوریتم‌های ازدحامی:
    - ایده‌های اولیه در مورد الگوریتم‌های جمعی مانند PSO در این دوره شکل گرفتند.



تاریخچه محاسبات تکاملی (دهه ۱۹۹۰)

تلفیق و گسترش کاربردها

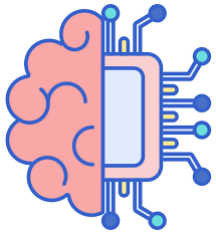
* الگوریتم‌های چندهدفه:
  - مفاهیم بهینه‌سازی چندهدفه، مانند NSGA-II، در این دوره توسعه یافتند.
* الگوریتم‌های ترکیبی:
  - محاسبات تکاملی با روش‌های دیگر مانند شبکه‌های عصبی و منطق فازی ترکیب شد.
* کتاب‌های مرجع:
  - هالند و سایر پژوهشگران کتاب‌هایی منتشر کردند که استانداردهای علمی محاسبات تکاملی را تعریف کردند.



تاریخچه محاسبات تکاملی (دهه ۲۰۰۰)

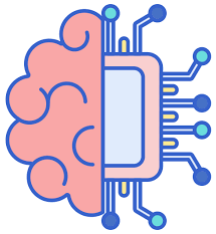
کاربردها و پیشرفت‌های مدرن

* کاربردهای گسترده:
  - محاسبات تکاملی در بیوانفورماتیک، طراحی مهندسی، بازی‌ها، و مسائل مالی
* الگوریتم‌های مدرن‌تر:
  - الگوریتم کلونی مورچه‌ها (ACO)
  - الگوریتم ازدحام ذرات (PSO)
* رایانش موازی و توزیع‌شده:
  - استفاده از سخت‌افزارهای پیشرفته برای تسریع اجرای الگوریتم‌های تکاملی.



تاریخچه محاسبات تکاملی (دهه ۲۰۱۰)

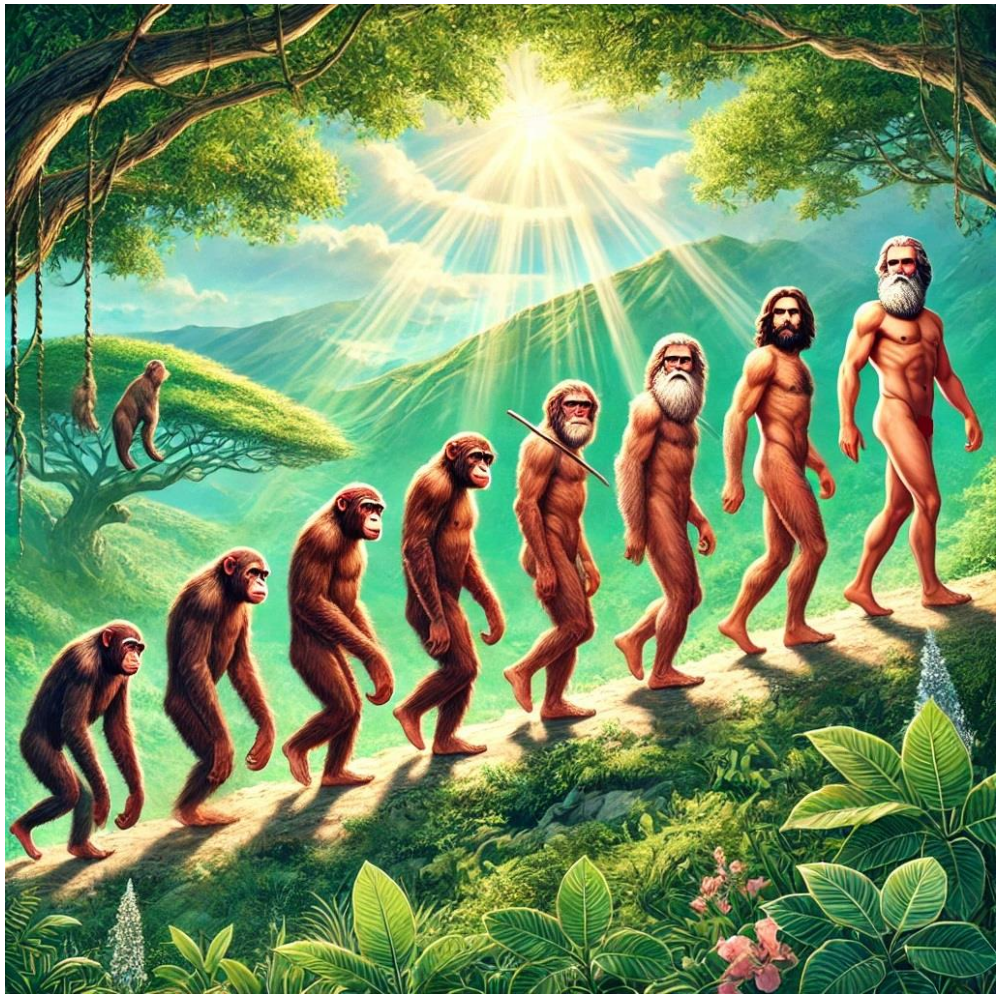
- توسعه الگوریتم‌های پیچیده‌تر:
  - استراتژی‌های تکامل (Evolution Strategies - ES) پیشرفته و بهینه‌سازی چندهدفه
- کاربردهای گسترده در صنایع:
  - در صنایع مختلف از جمله خودروسازی، رباتیک، طراحی مدارهای الکترونیکی و بهینه‌سازی منابع
- ادغام با یادگیری ماشین و هوش مصنوعی:
  - یادگیری تقویتی (Reinforcement Learning) و یادگیری عمیق (Deep Learning)
- پیشرفت‌های الگوریتمی:
  - الگوریتم جدید NSGA-II (Non-dominated Sorting Genetic Algorithm)
  - الگوریتم جدید CMA-ES (Covariance Matrix Adaptation Evolution Strategy)



تاریخچه محاسبات تکاملی (دهه ۲۰۲۰)

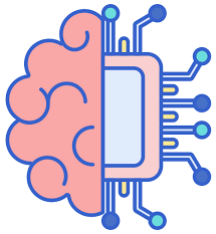
- توسعه ترکیب محاسبات تکاملی با یادگیری ماشین:
  - تنظیم بهینه پارامترهای مدل‌های یادگیری عمیق، کاربرد در یادگیری تقویتی و یافتن سیاست‌های بهینه
- الگوریتم‌های پیشرفته‌تر برای جستجوی پارامترها:
  - CMA-ES به عنوان یک الگوریتم پرکاربرد برای مسائل بهینه‌سازی در فضاهای جستجو بزرگتر و پیچیده‌تر
- محاسبات تکاملی در زمینه بهینه‌سازی چندهدفه و هوش مصنوعی:
  - بهینه‌سازی چندهدفه در طراحی محصولات صنعتی و آزمایش‌های علمی
  - در رایانش ابری و محاسبات موازی برای حل مسائل بزرگ و پیچیده نیز رشد یافت.
- پیشرفت در کاربردهای زیستی و پزشکی:
  - شبیه‌سازی‌های زیستی و پزشکی مانند مدل‌سازی دارویی و بهینه‌سازی درمان‌ها
- گسترش در هوش مصنوعی مولد:
  - محاسبات تکاملی در مدل‌های مولد مانند Generative Adversarial Networks (GANs)

تکامل طبیعی: مبنای بیولوژیکی محاسبات تکاملی



- تکامل طبیعی (Natural Evolution) فرآیندی زیستی است که در آن موجودات زنده با گذر زمان تغییر می‌کنند تا خود را با محیط‌شان سازگارتر کنند.

- این فرآیند توسط چارلز داروین در نظریه تکامل طبیعی شرح داده شد و اصول آن اساس الگوریتم‌های محاسبات تکاملی را تشکیل می‌دهند.



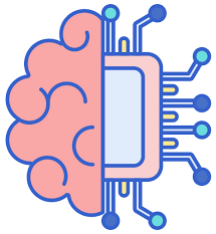
چهار فرضیه اصلی نظریه تکامل داروین

نمونه‌های موجود در یک جمعیت یا گونه‌های جاندار، تفاوت‌های معناداری با یکدیگر دارند.

در طول فرآیند تکامل و در گذار از هر نسل به نسل بعدی، برخی از ویژگی‌های موجودات زنده به فرزندان منتقل می‌شوند.

در هر نسل (تکرار) از فرآیند تکامل، تعداد نمونه‌های فرزند ایجاد شده از تعداد نمونه‌هایی که زنده خواهند ماند، بیشتر خواهد بود.

بقاء و تولید مثل نمونه‌ها به صورت تصادفی انجام نمی‌شود.



اصول کلیدی تکامل طبیعی (۱)

۱) انتخاب طبیعی (Natural Selection)

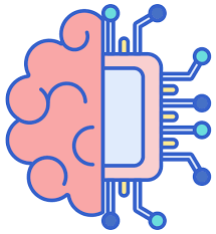
* موجوداتی که بهتر با محیط سازگارند، شانس بیشتری برای بقا و تولید مثل دارند.
* موجوداتی که ویژگی‌های مطلوب‌تری دارند، ژن‌های خود را به نسل بعد منتقل می‌کنند.

۲) وراثت (Inheritance)

* ویژگی‌های یک موجود زنده از طریق ژن‌ها (DNA) به نسل‌های بعد منتقل می‌شود.
* ژن‌ها اطلاعات بیولوژیکی را رمزگذاری کرده و صفات موجودات را تعیین می‌کنند.
* وراثت تضمین می‌کند که ویژگی‌های برتر می‌توانند در جمعیت باقی بمانند.

۳) جهش (Mutation)

* تغییرات تصادفی در ژن‌ها است که باعث ایجاد تنوع ژنتیکی شده و مثبت، منفی یا خنثی است.
* یکی از عوامل اصلی در تولید ویژگی‌های جدید و سازگاری بهتر موجودات با محیط است.



اصول کلیدی تکامل طبیعی (۲)

۴) تنوع ژنتیکی (Genetic Diversity)

* به مجموعه‌ای از ترکیبات مختلف ژن‌ها در یک جمعیت اشاره دارد.
  - تنوع بالا احتمال یافتن ویژگی‌های مطلوب برای بقا را افزایش داده و در تنوع ژنتیکی کم، احتمال انقراض بیشتر است.

۵) انتخاب جنسی (Sexual Selection)

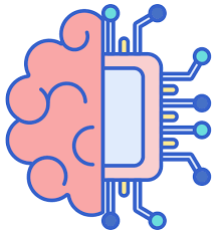
* فرآیندی که در آن موجودات ویژگی‌هایی را توسعه می‌دهند که برای جفت‌گیری جذاب‌تر باشد.
  - این ویژگی‌ها ممکن است به طور مستقیم با بقا مرتبط نباشند (مانند پرهای رنگین در طاووس).

۶) رقابت (Competition)

* موجودات باید برای دسترسی به منابع محدود موجود در محیط (مانند غذا) رقابت کنند.

۷) سازگاری (Adaptation)

* فرآیندی که یک جمعیت ویژگی‌هایی را که برای بقا و تولید مثل مفید هستند، توسعه می‌دهد.
  - این ویژگی‌ها ممکن است شامل تغییرات فیزیکی، رفتاری یا زیستی باشند



نقش اصول تکامل طبیعی در محاسبات تکاملی

انتخاب طبیعی: عملگر انتخاب (Selection Operator)

راه‌حل های بهتر (برازنده‌تر) شانس بیشتری برای انتخاب و انتقال به نسل بعد دارند.

وراثت: انتقال ویژگی‌ها (Inheritance of Traits)

راه‌حل ها یا موجودات (کاندیداها) ویژگی‌های خود را به نسل های بعد منتقل می‌کنند.

جهش: عملگر جهش (Mutation Operator)

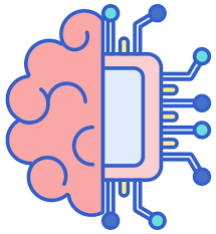
ایجاد تغییرات تصادفی در راه‌حل ها جهت افزایش تنوع. (جلوگیری از گیر افتادن در بهینه محلی)

ترکیب ژنتیکی: عملگر ترکیب (Crossover Operator)

دو راه‌حل ترکیب می‌شوند تا ویژگی‌های مفید هر دو را به نسل بعد منتقل کنند.

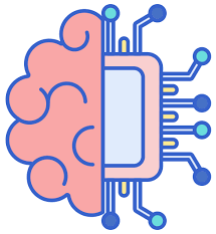
تنوع ژنتیکی: تنوع جمعیت اولیه (Population Diversity)

ایجاد یک جمعیت اولیه متنوع در الگوریتم‌های تکاملی، مشابه تنوع ژنتیکی در طبیعت است.



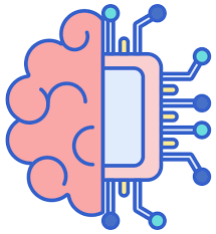
نقش محاسبات تکاملی در هوش مصنوعی

- حل مسائل بهینه‌سازی پیچیده
  - بسیاری از مسائل در هوش مصنوعی، مانند طراحی شبکه‌های عصبی از نوع بهینه‌سازی هستند.
- الهام از طبیعت و سیستم‌های زیستی
  - شبکه‌های عصبی مصنوعی از ساختار مغز الهام گرفته‌اند، محاسبات تکاملی نیز از فرآیندهای زیستی مانند تکامل و انتخاب طبیعی الهام می‌گیرد.
- حل مسائل غیرخطی و غیر قطعی
  - در مسائل پیچیده که معادلات دقیق یا قطعی وجود ندارد، محاسبات تکاملی یک ابزار مناسب برای یافتن راه‌حل های نزدیک به بهینه است.



ارتباط محاسبات تکاملی با یادگیری ماشین

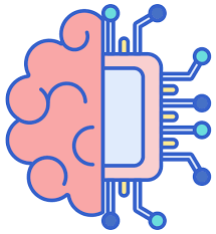
- بهینه‌سازی هایپر پارامترها
  - اهمیت زیاد تنظیم هایپر پارامترها (مانند تعداد لایه‌ها در شبکه‌های عصبی) در یادگیری ماشین
- طراحی معماری مدل‌ها
  - استفاده محاسبات تکاملی در شبکه‌های عصبی مصنوعی برای طراحی خودکار معماری شبکه‌ها
- ترکیب یادگیری و تکامل
  - NeuroEvolution of Augmenting Topologies (NEAT): محاسبات تکاملی برای تکامل شبکه‌های عصبی
  - Evolutionary Reinforcement Learning: محاسبات تکاملی برای بهبود الگوریتم‌های یادگیری تقویتی
- انتخاب ویژگی (Feature Selection)
  - چالش مهم در یادگیری ماشین: انتخاب ویژگی‌های مؤثر از میان داده‌های موجود
- بهینه‌سازی تابع هدف
  - بسیاری از مسائل در یادگیری ماشین، بهینه‌سازی تابع هدف (مانند کاهش خطا یا افزایش دقت) هستند.



مزایا و محدودیت‌های محاسبات تکاملی

- **مزایا:**
  - **استقلال از مشتقات:** محاسبات تکاملی نیازی به محاسبه مشتقات ندارد
  - **تعامل با داده‌های نویزی یا ناقص**
  - **قابلیت جستجوی جهانی:** این روش‌ها برخلاف تکنیک‌های گرادیان نزولی، می‌توانند از بهینه‌های محلی عبور کرده و به بهینه‌های جهانی برسند.

- **محدودیت‌ها:**
  - **هزینه محاسباتی بالا:** اجرای الگوریتم‌های تکاملی به منابع محاسباتی زیادی نیاز دارد.
  - **سرعت پایین:** محاسبات تکاملی معمولاً از روش‌های کلاسیک یادگیری ماشین کندتر هستند.
  - **نیاز به تنظیم دقیق پارامترها:** عملکرد بهینه نیازمند انتخاب مناسب پارامترها مانند نرخ جهش و اندازه جمعیت است.



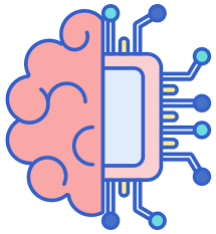
چهار مؤلفه اصلی یک الگوریتم تکاملی

۱ **جمعیت** که مجموعه نمونه‌های موجود در جمعیت را مدل‌سازی می‌کند

۲ **عملگرهای تکاملی** که سبب تغییر کدهای ژنتیکی نمونه‌های موجود در جمعیت و افزایش تنوع در جواب‌های کاندید تولید شده می‌شود.

۳ **برازندگی** که کیفیت جواب‌های کاندید تولید شده را می‌سنجد.

۴ **انتخاب** که اصل بقای برازنده‌ترین‌ها را در الگوریتم‌های محاسبات تکاملی شبیه‌سازی می‌کند.



مفاهیم مهم در الگوریتم‌های تکاملی

کروموزوم (Chromosome)

- اطلاعات ژنی یک موجود در کروموزوم ذخیره می‌شود. هر کروموزوم از DNA تشکیل می‌شود. در الگوریتم های تکاملی منظور از کروموزوم، یک رشته، گراف، درخت یا دنباله‌ای از صفر و یک‌ها می‌باشد که معادل یک پاسخ برای حل یک مسئله می‌باشند.

ژن (Gene)

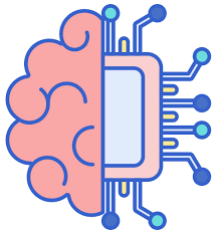
- کروموزوم به چند قسمت به نام ژن (تشکیل دهنده خصوصیات گونه‌ها یا افراد) تقسیم می‌شود. در یک الگوریتم تکاملی، هر پاسخ چند بخش دارد که به هر کدام از بخش‌ها یک مشخصه یا ویژگی می گویند. مشخصه یا ویژگی در هر پاسخ معادل ژن در کروموزوم است.

آلل (Allele)

- هر ژن مجموعه‌ای از مقادیر مجاز را می‌تواند اختیار کند. به هر مقدار یک آلل می گویند. مثلاً ژن مربوط به رنگ چشم، دارای آلل‌های سیاه، خاکستری، قهوه‌ای، آبی، سبز و عسلی است. در الگوریتم های تکاملی مقادیر مجاز برای مشخصه‌های هر پاسخ معادل مفهوم آلل است.

برازش (Reproduction)

- شایستگی یک موجود در یک جمعیت، برازش آن موجود نامیده می‌شود. برازش هر پاسخ در جمعیت، با توجه به شایستگی فنوتایپ آن پاسخ تعیین می‌گردد. در یک الگوریتم تکاملی، با استفاده از یک یا چند تابع، برازش هر پاسخ محاسبه می‌شود. تابع برازش، تنها ارتباط یک الگوریتم تکاملی با مسئله مورد تحلیل می‌باشد.



مفاهیم مهم در الگوریتم‌های تکاملی

فنوتایپ (Phenotype)

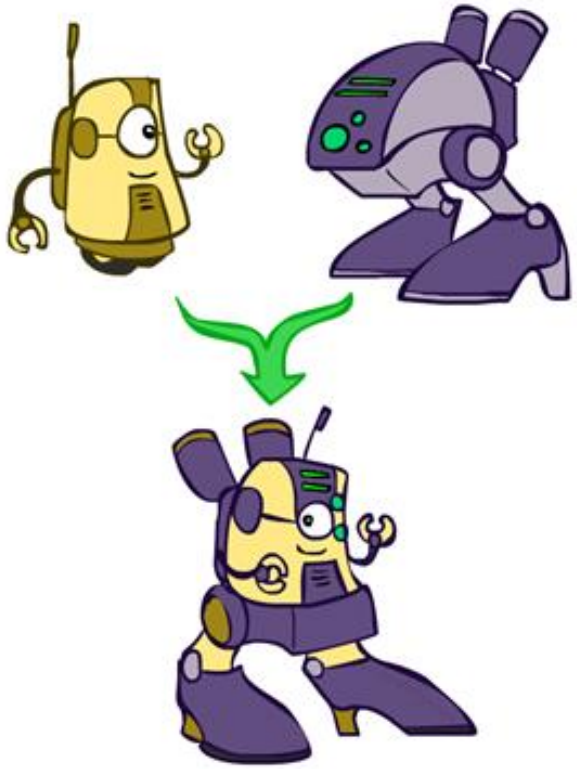
- موجودات از بدن و مجموعه‌ای از رفتارها تشکیل شده‌اند. به این ویژگی‌ها، فنوتایپ آن موجود زنده گفته می‌شود. به عبارت دیگر، شکل ظاهری و رفتاری، رنگ مو، چشم و پوست، همگی بخش از فنوتایپ موجودات زنده هستند.

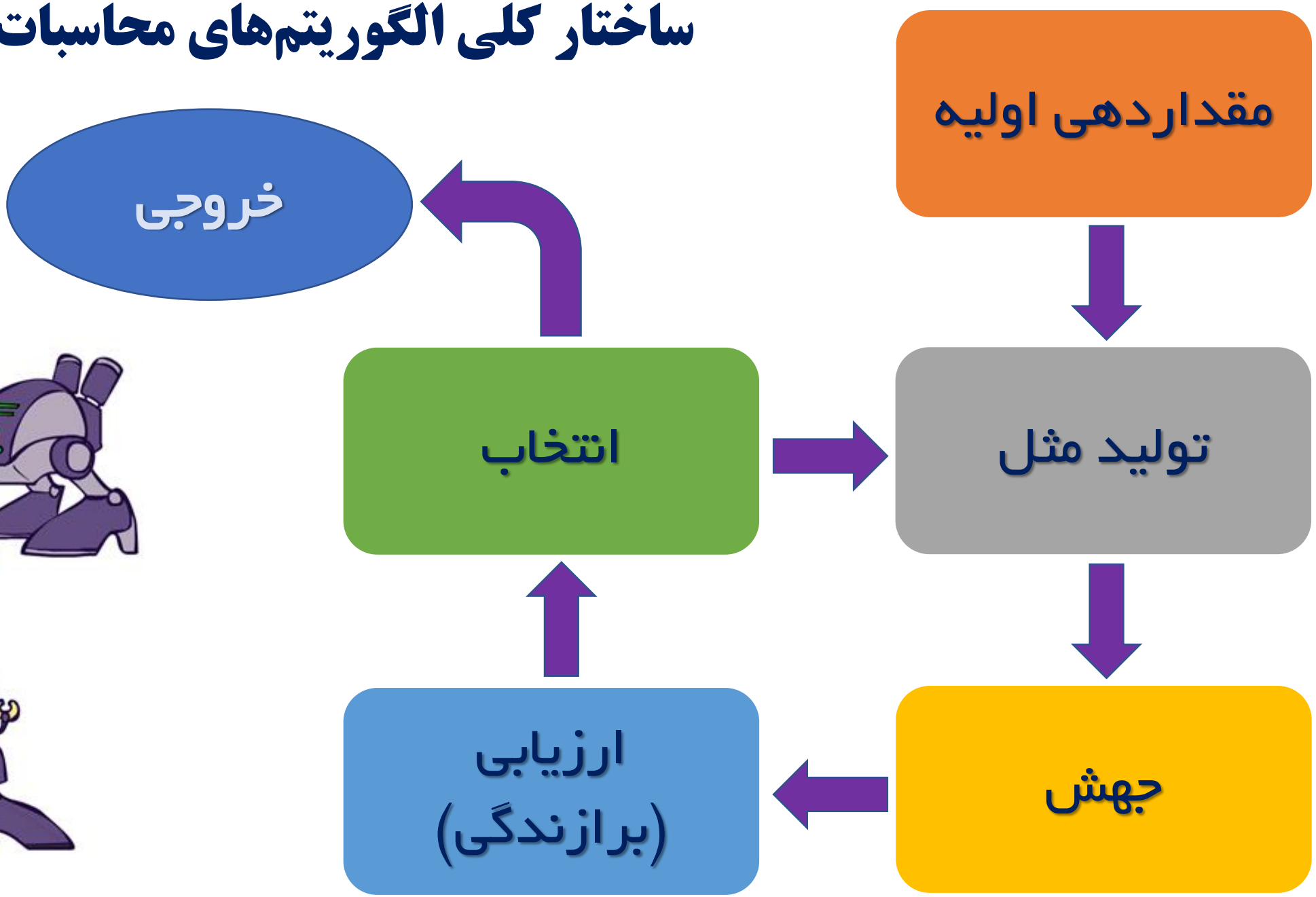
ژنوتایپ (Genotype)

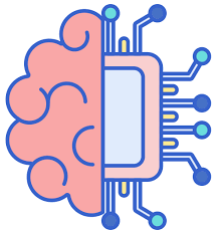
- شکل ظاهری موجود توسط مجموعه‌ای از دستورات کدبندی شده (Encoded) در سلول‌های آن مشخص می‌شود. به این مجموعه دستورالعمل کدبندی شده، ژنوتایپ گفته می‌شود. ژنوتایپ به اطلاعاتی اطلاق می‌شود که در DNA موجودات زنده کدبندی شده است.

در الگوریتم‌های محاسبات تکاملی به جمعیت جواب‌های کاندید یک مسأله بهینه‌سازی که به سمت جواب بهینه همگرا می‌شوند، فنوتایپ گفته می‌شود. هر کدام از جواب‌های کاندید، مجموعه‌ای از ویژگی‌های مختص به خود دارند که از طریق اعمال عملگرهای تکاملی تغییر پیدا می‌کند؛ به این مجموعه از ویژگی‌ها ژنوتایپ گفته می‌شود.

ساختار کلی الگوریتم‌های محاسبات تکاملی

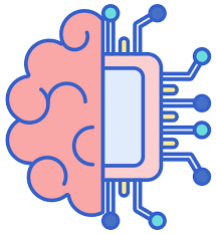






نمونه های الگوریتم های محاسبات تکاملی

- الگوریتم ژنتیک
- الگوریتم کلونی مورچگان
- الگوریتم کرم شب تاب
- الگوریتم بهینه سازی فاخته
- الگوریتم کلونی زنبور عسل
- الگوریتم بهینه سازی گرگ خاکستری
- الگوریتم جهش قورباغه
- الگوریتم بهینه سازی ازدحام ماهیان مصنوعی
- الگوریتم بهینه سازی علف هرز مهاجم
- الگوریتم جستجوی ممنوع Tabu
- الگوریتم بهینه سازی مبتنی بر جغرافیای زیستی
- الگوریتم تکاملی تفاضلی
- الگوریتم شاهین هریس
- الگوریتم کلونی پنگوئن های امپراتور
- الگوریتم شعله پروانه
- الگوریتم بهینه سازی وال ها یا نهنگ
- الگوریتم رقابت استعماری
- الگوریتم بهینه سازی ازدحام ذرات
- الگوریتم بهینه سازی مبتنی بر آموزش و یادگیری
- الگوریتم فرهنگی
- الگوریتم شبیه سازی تبرید
- الگوریتم جستجوی هارمونی



اصطلاحات طبیعت و معادل کامپیوتری آنها

طبیعت	کامپیوتر
جمعیت	مجموعه ای از راه حل ها
فرد	راه حلی برای یک مسئله
برازندگی	کیفیت یک راه حل
کروموزوم	رمزگذاری برای یک راه حل
ژن	بخشی از رمزگذاری برای یک راه حل
تولید مثل	ترکیب

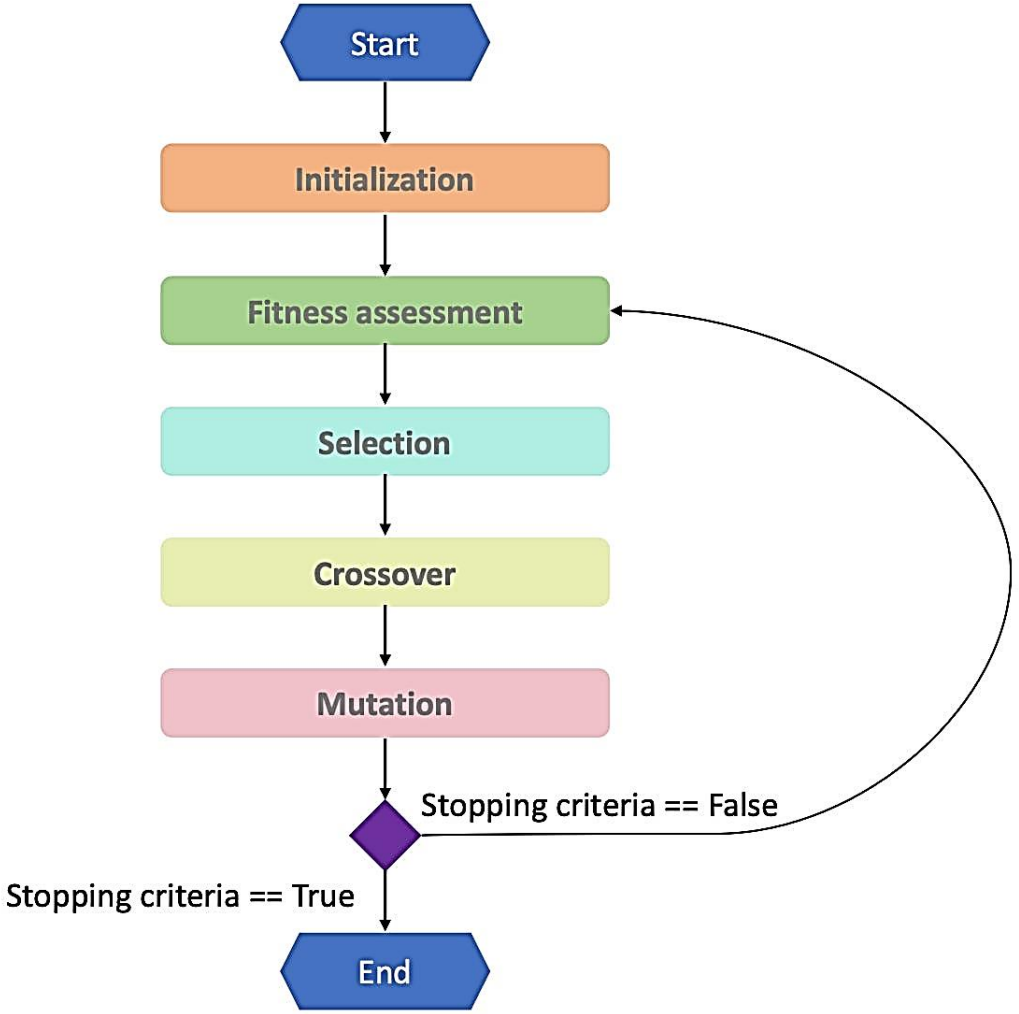
Genetic Algorithm

الگوریتم ژنتیک (Genetic Algorithm)

- الگوریتم ژنتیک (GA) یکی از مشهورترین روش‌های محاسبات تکاملی است که از فرآیند تکامل زیستی الهام گرفته است.

- این الگوریتم برای حل مسائل پیچیده و بهینه‌سازی طراحی شده است و مبتنی بر استفاده از جمعیتی از راه‌حل های ممکن است که با استفاده از عملگرهای تکاملی (مانند انتخاب، ترکیب، و جهش) بهبود می‌یابند.

مراحل اصلی در ساختار کلی الگوریتم ژنتیک



۱) ایجاد جمعیت اولیه (Initialization)

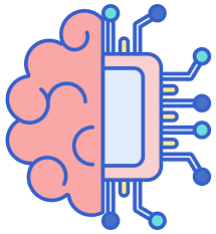
۲) ارزیابی (Evaluation)

۳) انتخاب (Selection)

۴) تولید نسل جدید (Reproduction)

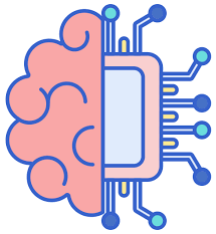
۵) جایگزینی (Replacement)

۶) تکرار و خاتمه (Iteration and Termination)



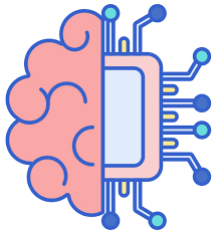
ایجاد جمعیت اولیه و ارزیابی

- جمعیت اولیه شامل مجموعه‌ای از راه‌حل‌های ممکن (افراد یا کروموزوم‌ها) است که به‌طور تصادفی یا با روش‌های خاص تولید می‌شود.
- هر فرد در جمعیت به صورت یک بردار یا رشته نمایش داده می‌شود که نشان‌دهنده یک راه‌حل برای مسئله است.
  - مثال: در مسائل بهینه‌سازی، هر کروموزوم می‌تواند یک نقطه در فضای جستجو باشد.
- برازندگی (Fitness) هر کروموزوم با استفاده از یک تابع هدف ارزیابی می‌شود.
- تابع هدف معیاری است که نشان می‌دهد هر راه‌حل چقدر بهینه یا مناسب است.
  - مثال: در مسئله کوله‌پشتی، برازندگی می‌تواند مجموع ارزش آیتم‌های انتخابی باشد.



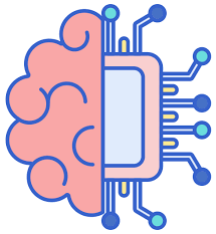
انتخاب

- کروموزوم‌های بهتر (با برازندگی بالاتر) شانس بیشتری برای انتقال ژن‌های خود به نسل بعد دارند.
- روش‌های رایج برای انتخاب:
  - **چرخ رولت (Roulette Wheel):** احتمال انتخاب هر فرد متناسب با برازندگی او است.
  - **تورنمنت:** گروهی از افراد به‌صورت تصادفی انتخاب شده و بهترین فرد در گروه انتخاب می‌شود.
  - **انتخاب بر اساس رتبه (Rank-Based Selection):** افراد بر اساس رتبه برازندگی مرتب شده و انتخاب می‌شوند.



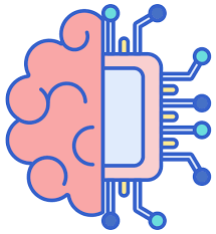
تولید نسل جدید

- از کروموزوم‌های انتخاب‌شده برای تولید نسل جدید استفاده می‌شود. این مرحله شامل عملگرهای زیر است:
  - * **ترکیب یا تقاطع (Crossover):** **ترکیب** ژن‌های دو کروموزوم والد جهت تولید یک یا چند فرزند
    - **تک‌نقطه‌ای (Single-Point):** کروموزوم در یک نقطه شکسته شده و بخش‌هایی از دو والد ترکیب می‌شوند.
    - **چندنقطه‌ای (Multi-Point):** کروموزوم در چند نقطه شکسته و دوباره ترکیب می‌شود.
    - **یکنواخت (Uniform Crossover):** ژن‌های والدین به‌صورت تصادفی انتخاب و ترکیب می‌شوند.
  - * **جهش (Mutation):** برخی ژن‌ها بطور تصادفی تغییر می‌کنند تا تنوع ژنتیکی در جمعیت حفظ شود.
    - این عملگر از گیر افتادن الگوریتم در بهینه‌های محلی جلوگیری می‌کند.
    - مثال: در یک رشته دودویی، مقدار "0" ممکن است به "1" تغییر کند.



جایگزینی، تکرار و خاتمه

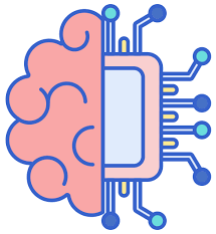
- نسل جدید کروموزوم‌ها جایگزین نسل قبلی می‌شوند.
- استراتژی‌های جایگزینی:
  - **کامل (Full Replacement):** تمام جمعیت قدیمی با نسل جدید جایگزین می‌شود.
  - **جزئی (Elitism):** بهترین کروموزوم‌های نسل قبلی به نسل جدید منتقل می‌شوند تا تضمین شود که راه‌حل‌های خوب از بین نمی‌روند.
- تمام مراحل تکرار می‌شوند تا یکی از شرایط خاتمه برقرار شود:
  - تعداد معینی از نسل‌ها تولید شده باشد.
  - مقدار برازندگی به یک حد مطلوب رسیده باشد.
  - بهبود معناداری در برازندگی مشاهده نشود.



نمایش یا کدگذاری (Encoding)

- راه‌حل های مسئله به شکل کروموزوم‌ها نمایش داده می‌شوند و نوع نمایش مستقیماً بر عملکرد الگوریتم ژنتیک و توانایی آن برای یافتن راه‌حل‌های بهینه تأثیر دارد.

- کروموزوم‌ها معمولاً به صورت رشته‌هایی از مقادیر نمایش داده می‌شوند که هر بخش از آن (ژن) نشان‌دهنده یک ویژگی یا پارامتر در راه‌حل است.
  - نمایش باینری (Binary Encoding)
  - نمایش عددی یا حقیقی (Real-Valued Encoding)
  - نمایش جایگشتی (Permutation Encoding)
  - نمایش نمادی یا گسسته (Symbolic or Discrete Encoding)
  - نمایش چندگانه یا ترکیبی (Hybrid Encoding)



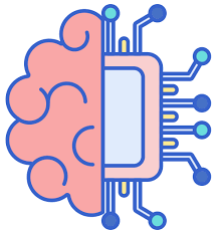
نمایش باینری (Binary Encoding)

- کروموزوم‌ها به صورت رشته‌هایی دودویی از ژن ها (هر ژن = یک بیت) نمایش داده شده و می‌تواند نشان‌دهنده ویژگی‌های مختلفی مانند وجود یا عدم وجود یک آیتم باشد.

* **مثال:** در مسئله کوله‌پشتی یک کروموزوم = [0, 1, 1, 0, 1]
  - مقدار 1 یعنی آیتم در کوله‌پشتی انتخاب شده است و مقدار 0 یعنی آیتم انتخاب نشده است.

* **مزایا:**
  - ساده و آسان برای پیاده‌سازی.
  - عملگرهای ژنتیکی مانند تقاطع و جهش برای رشته‌های باینری به راحتی قابل اجرا هستند.

* **معایب:**
  - نمایش نامناسب برای مسائل عددی یا پیوسته.
  - ممکن است در مسائل پیچیده با طول زیاد، کارایی کمتری داشته باشد.



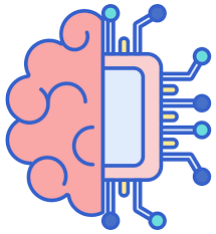
نمایش عددی یا حقیقی (Real-Valued Encoding)

- کروموزوم به صورت آرایه‌ای از اعداد نمایش داده شده که برای مسائل عددی و پیوسته (Continuous) مناسب است.

* **مثال:** در بهینه‌سازی تابع یک کروموزوم = [0/8, 4/0, -1/3, 2/5]
  - هر عدد نشان‌دهنده مقدار یک متغیر است.

* **مزایا:**
  - مناسب برای مسائل عددی و پیوسته.
  - دقت بالاتری در یافتن مقادیر بهینه دارد.

* **معایب:**
  - نیاز به تعریف عملگرهای ژنتیکی پیچیده‌تر مانند ترکیب عددی و جهش مقیاسی.



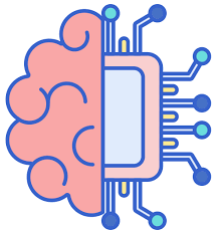
نمایش جایگشتی (Permutation Encoding)

- کروموزوم‌ها به صورت یک آرایه یا لیست از اعداد یا نمادها نمایش داده می‌شوند که ترتیب آن‌ها اهمیت دارد.
- برای مسائل مربوط به ترتیب یا توالی استفاده می‌شود.

* **مثال:** در مسئله فروشنده دوره‌گرد (TSP) یک کروموزوم = [5, 2, 4, 1, 3]
  - مسیر فروشنده: شهر ۳ ← شهر ۱ ← شهر ۴ ← شهر ۲ ← شهر ۵.

* **مزایا:**
  - مناسب برای مسائل ترتیب یا توالی مانند مسیریابی و زمان‌بندی.

* **معایب:**
  - نیاز به تعریف عملگرهای ترکیب و جهش ویژه که ترتیب عناصر را حفظ کنند.



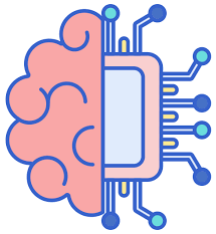
نمایش نمادی یا گسسته (Symbolic or Discrete Encoding)

- کروموزوم‌ها از نمادها یا مقادیر گسسته استفاده کرده و برای مسائلی که مقادیر گسسته و محدود دارند، مناسب است.

* **مثال:** در طراحی مدار یک کروموزوم = ['A', 'B', 'C', 'D']
  - هر نماد نشان‌دهنده یک قطعه در مدار است.

* **مزایا:**
  - انعطاف‌پذیری بالا برای مسائل خاص.

* **معایب:**
  - تعریف عملگرهای ژنتیکی ممکن است پیچیده‌تر باشد.



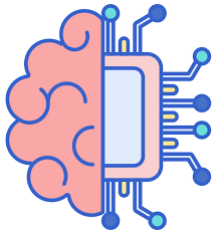
نمایش چندگانه یا ترکیبی (Hybrid Encoding)

- گاهی اوقات، یک مسئله پیچیده تر نیازمند استفاده از ترکیبی از روش‌های قبل است که شامل داده‌های مختلف (عددی، باینری، و غیره) هستند.

* **مثال:** در طراحی سیستم تولید یک کروموزوم = [3, -1/3, 2/5, 1, 0, 1]
  - بخش باینری: انتخاب قطعات.
  - بخش عددی: تنظیم مقادیر پارامترها.

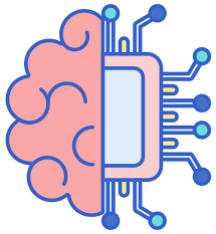
* **مزایا:**
  - انعطاف‌پذیر و قابل استفاده برای مسائل پیچیده.

* **معایب:**
  - نیازمند تعریف دقیق عملگرها برای بخش‌های مختلف.



معیارهای انتخاب روش نمایش

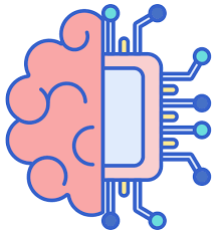
- نوع مسئله:
  - مسائل عددی: نمایش عددی یا حقیقی.
  - مسائل ترتیب: نمایش جایگشتی.
  - مسائل دودویی: نمایش باینری.
- پیچیدگی مسئله:
  - مسائل ساده: روش‌های استاندارد (باینری یا عددی).
  - مسائل ترکیبی: نمایش ترکیبی.
- سهولت استفاده:
  - روش‌هایی که عملگرهای ساده‌تری دارند (مانند باینری) در مسائل ابتدایی ترجیح داده می‌شوند.



تابع برازندگی (Fitness Function)

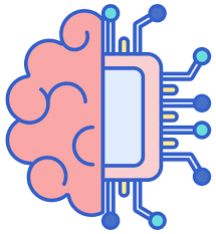
- تابع برازندگی مهمترین بخش الگوریتم ژنتیک است که به هر کروموزوم در جمعیت یک مقدار برازندگی اختصاص می‌دهد. این مقدار نشان دهنده کیفیت یا مطلوبیت آن کروموزوم به عنوان یک راه‌حل برای مسئله است.

- کاربردهای تابع برازندگی
  - **ارزیابی کروموزوم‌ها:** مشخص می‌کند که کدام کروموزوم‌ها در جمعیت بهتر هستند.
  - **راهنمای انتخاب:** کروموزوم هایی با برازندگی بالاتر شانس بیشتری برای انتخاب و تولید مثل دارند.
  - **جهت دهی به جستجو:** باعث می‌شود الگوریتم به سمت نواحی بهینه در فضای جستجو حرکت کند.



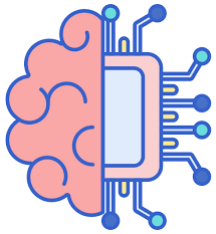
انواع تابع برازندگی

- **تابع مجموع یا خطی (Linear Fitness Function)**
  - برای مسائلی که هدف ما بیشینه یا کمینه کردن یک مقدار خطی است.
  - برازندگی هر کروموزوم برابر با مقدار مستقیم تابع هدف است.
- مثال: بهینه‌سازی تابع $f(x) = 2x+3$
  - کروموزوم‌ها: $x_1=1$, $x_2=2$, $x_3=3$
  - برازندگی:
    - $f(x_1) = 5$, $f(x_2) = 7$, $f(x_3) = 9$
- کاربرد:
  - مسائل ساده مانند بیشینه سازی خطی



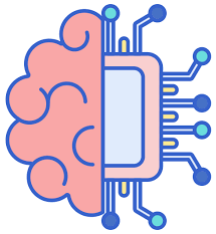
انواع تابع برازندگی

- **تابع غیرخطی یا پیچیده (Non-Linear Fitness Function)**
  - برای مسائل غیرخطی و پیچیده استفاده می‌شود.
- مثال: بهینه‌سازی تابع $f(x) = x^2 - 4x + 3$
  - کروموزوم‌ها: $x_1 = 1 , x_2 = 2 , x_3 = 3$
  - برازندگی:
    - $f(x_1) = 0 , f(x_2) = -1 , f(x_3) = 0$
- کاربرد:
  - مسائل مهندسی و ریاضی پیچیده.



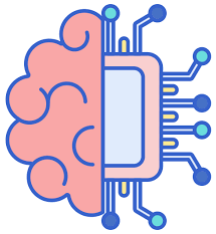
انواع تابع برازندگی

- **تابع چند هدفه (Multi-Objective Fitness Function)**
  - برای مسائل با چندین هدف بهینه‌سازی، مانند کاهش هزینه و افزایش کیفیت به‌طور همزمان.
  - معمولاً یک ترکیب وزنی از اهداف مختلف است.
- مثال: بهینه‌سازی
  - $f_1(x) = x^2$ (هدف اول) و $f_2(x) = 10-x$ (هدف دوم)
- تابع برازندگی:
  - $F(x) = w_1 f_1(x)+w_2 f_2(x)$
  - $w_1 = 0.6 , w_2 = 0.4$
- کاربرد:
  - مسائل بهینه‌سازی چند هدفه، مانند طراحی خودرو (سرعت و مصرف سوخت).



انواع تابع برازندگی

- **تابع مقید (Constrained Fitness Function)**
  - شامل قیدهای اضافی است که باید رعایت شوند.
  - به کروموزوم هایی که قیدها را نقض می‌کنند، برازندگی کمتری داده می‌شود.
- مثال: بهینه‌سازی تابع $f(x) = x^2$ با قید $x>0$
  - کروموزوم‌ها: $x_1 = -2$, $x_2 = 1$, $x_3 = 3$
  - برازندگی:
    - $f(x_1) = \emptyset$ (Invalid) , $f(x_2) = 1$, $f(x_3) = 9$
- کاربرد:
  - مسائل مهندسی که شامل محدودیت‌های فیزیکی یا عملی هستند.

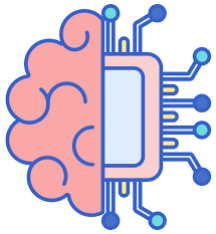


انواع تابع برازندگی

- **تابع ترتیبی (Rank-Based Fitness Function)**
  - به جای مقدار واقعی تابع هدف، رتبه کروموزوم‌ها در جمعیت برای برازندگی استفاده می‌شود.

- مثال:
  - جمعیت: $[f(x_1) = 10 , f(x_2) = 20 , f(x_3) = 15]$
  - رتبه‌ها: $f_2 > f_3 > f_1$
  - برازندگی: به ترتیب رتبه از چپ به راست $2 > 3 > 1$

- کاربرد:
  - مسائل با تفاوت‌های کوچک در برازندگی که مقایسه مستقیم دشوار است.

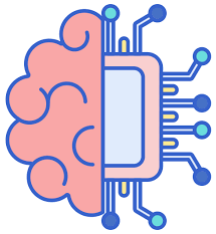


انواع تابع برازندگی

- **تابع مبتنی بر احتمال (Probability-Based Fitness Function)**
  - برازندگی بصورت احتمال رخداد یا موفقیت یک کروموزوم تعیین می‌شود.

- مثال:
  - در بازی شطرنج، برازندگی یک استراتژی ممکن است بر اساس احتمال برنده شدن محاسبه شود.

- کاربرد:
  - مسائل با عدم قطعیت یا شبیه‌سازی.

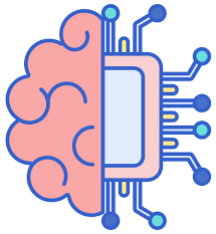


فرآیند انتخاب (Selection) در الگوریتم ژنتیک

- این مرحله از الگوریتم ژنتیک تعیین می‌کند که کدام کروموزوم‌ها (راه‌حل ها) از جمعیت فعلی برای تولید نسل بعدی انتخاب شوند. هدف انتخاب، اطمینان از این است که کروموزوم های با برازندگی بالاتر شانس بیشتری برای بقا و تولید مثل داشته باشند.

- هدف فرآیند انتخاب:

* **تکثیر کروموزوم های قوی‌تر:** کروموزوم هایی که برازندگی بیشتری دارند، احتمال بیشتری برای انتقال ژن‌های خود به نسل بعد دارند.
* **حفظ تنوع جمعیت:** تنوع جمعیت برای جلوگیری از همگرایی زودرس ضروری است.
* **ایجاد فشار انتخابی:** کمک می‌کند که جمعیت به سمت مناطق بهینه در فضای جستجو حرکت کند.



روش‌های انتخاب

- **انتخاب چرخ رولت (Roulette Wheel Selection)**
  - احتمال انتخاب هر کروموزوم متناسب با مقدار برازندگی آن است.
  - کروموزوم های با برازندگی بیشتر، شانس بیشتری برای انتخاب دارند.

- **مزایا:**
  - ساده و قابل درک.

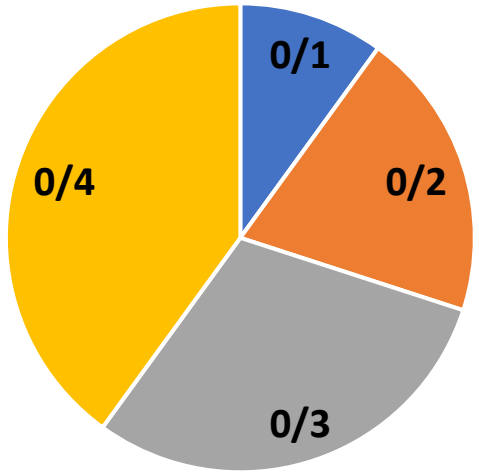
- **معایب:**
  - در صورت تفاوت زیاد در برازندگی، کروموزوم های ضعیف شانس بسیار کمی خواهند داشت.

مثال:

جمعیت:

- کروموزوم‌ها: C_1 , C_2 , C_3 , C_4
- برازندگی: $f(C_1) = 10 , f(C_2) = 20 , f(C_3) = 30 , f(C_4) = 40$
- مجموع برازندگی: $10 + 20 + 30 + 40 = 100$
- احتمال انتخاب:

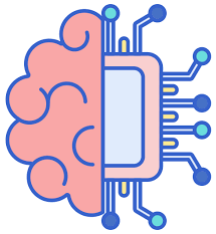
- $P(C_1) = \frac{10}{100} = 0.1 \longmapsto [0, 0.1]$
- $P(C_2) = \frac{20}{100} = 0.2 \longmapsto [0.1, 0.3]$
- $P(C_3) = \frac{30}{100} = 0.3 \longmapsto [0.3, 0.6]$
- $P(C_4) = \frac{40}{100} = 0.4 \longmapsto [0.6, 1]$



مراحل:

۱. یک عدد تصادفی در بازه [0 , 1] تولید می‌شود.

۲. براساس این عدد و احتمال کروموزوم‌ها، کروموزوم انتخاب می‌شود.



روش‌های انتخاب

- **انتخاب تورنمنت (Tournament Selection)**
  - انتخاب بهترین کروموزوم در بین گروهی از کروموزوم های (معمولاً بیشتر از ۲) تصادفی انتخاب شده
- مثال:
  - جمعیت و برازندگی کروموزوم‌ها: $f(C_1) = 10$, $f(C_2) = 20$, $f(C_3) = 30$, $f(C_4) = 40$
  - C_2 و C_3 بطور تصادفی انتخاب می‌شوند. چون $f(C_3)>f(C_2)$ است پس C_3 انتخاب می‌شود.
- مزایا:
  - امکان کنترل فشار انتخابی از طریق اندازه گروه تورنمنت.
  - ساده و مؤثر.
- معایب:
  - ممکن است به کروموزوم های ضعیف شانس کمی داده شود.

مثال:

فرض کنید جمعیت اولیه شامل ۶ کروموزوم با شایستگی های زیر است:

$f(C_1) = 10 , f(C_2) = 25 , f(C_3) = 15 , f(C_4) = 30 , f(C_5) = 20 , f(C_6) = 5$

مرحله ۱: تعیین اندازه تورنمنت (k): فرض کنید اندازه تورنمنت $k=3$ است. در هر تورنمنت، ۳ کروموزوم بصورت تصادفی انتخاب می‌شوند.

مرحله ۲: تشکیل تورنمنت و انتخاب برنده: برای هر انتخاب، مراحل زیر را انجام می‌دهیم:

- **تورنمنت ۱:** کروموزوم‌های C_4 , C_3 , C_2 بصورت تصادفی انتخاب می‌شوند. شایستگی آنها:

$f(C_2) = 25 , f(C_3) = 15 , f(C_4) = 30$

برنده تورنمنت: C_4 (بالاترین شایستگی).

- **تورنمنت ۲:** کروموزوم‌های C_6 , C_5 , C_1 بصورت تصادفی انتخاب می‌شوند. شایستگی آنها:

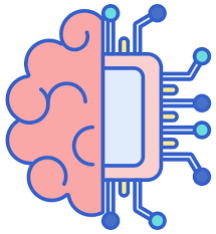
$f(C_1) = 10 , f(C_5) = 20 , f(C_6) = 5$

برنده تورنمنت: C_5 (بالاترین شایستگی).

- **تورنمنت ۳:** کروموزوم‌های C_4 , C_3 , C_6 بصورت تصادفی انتخاب می‌شوند. شایستگی آنها:

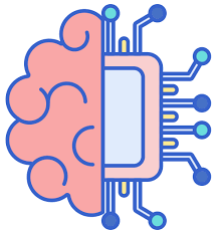
$f(C_3) = 15 , f(C_4) = 30 , f(C_6) = 5$

برنده تورنمنت: C_4 (بالاترین شایستگی).



روش‌های انتخاب

- **انتخاب قطعی (Elitism Selection)**
  - بهترین کروموزوم‌ها بدون تغییر مستقیماً به نسل بعد منتقل می‌شوند.
  - این روش تضمین می‌کند که بهترین راه‌حل های موجود از بین نمی‌روند.
- مثال:
  - اگر C_4 بهترین کروموزوم باشد، مستقیماً به نسل بعد منتقل می‌شود.
- مزایا:
  - حفظ بهترین راه‌حل ها.
- معایب:
  - ممکن است تنوع جمعیت کاهش یابد.



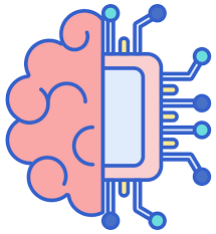
روش‌های انتخاب

- **انتخاب براساس رتبه (Rank-Based Selection)**
  - کروموزوم‌ها براساس برازندگی مرتب می‌شوند و شانس انتخاب آنها براساس رتبه آنها تعیین می‌شود.
- مثال:
  - جمعیت و رتبه‌ها: $R(C_4) = 1 , R(C_3) = 2 , R(C_2) = 3 , R(C_1) = 4$
  - احتمال انتخاب: $\frac{1}{R}$
- مزایا:
  - تفاوت زیاد در برازندگی را متعادل می‌کند.
- معایب:
  - ممکن است کروموزوم های ضعیف‌تر همچنان شانس زیادی داشته باشند.

فرآیند تولید مثل (Reproduction) در الگوریتم ژنتیک

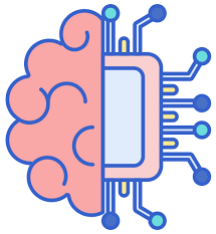
- در این فرآیند کروموزوم‌های انتخاب شده در مرحله **انتخاب** از طریق اعمال دو عملیات اصلی، یعنی **ترکیب (Crossover)** و **جهش (Mutation)** به تولید نسل جدید کمک می‌کنند.

- هدف این مرحله، تولید کروموزوم‌های جدید است که ویژگی‌های مطلوب کروموزوم‌های والد را به ارث ببرند و در عین حال، تنوع ژنتیکی جمعیت حفظ شود.



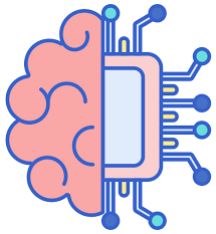
ترکیب (Crossover)

- فرآیندی است که در آن دو والد (کروموزوم) با یکدیگر ادغام شده و فرزندان جدیدی ایجاد می‌شوند.
- ترکیب باعث می‌شود که اطلاعات ژنتیکی از والدین به نسل جدید منتقل شود.
- نقش ترکیب:
  - **ایجاد تنوع:** با مخلوط کردن ژن‌های والدین، ترکیب‌های جدیدی ایجاد می‌شوند.
  - **انتقال اطلاعات مفید:** صفات خوب از والدین به فرزندان منتقل می‌شود.
  - **جهت‌دهی به بهینه‌سازی:** فرزندان معمولاً بهتر از والدین هستند، زیرا اطلاعات مفید هر دو والد را دارند.



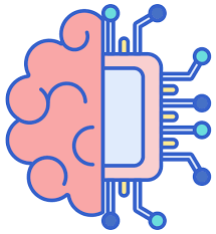
انواع روش‌های ترکیب

- **ترکیب تک نقطه‌ای (Single-Point Crossover)**
  - یک نقطه برش تصادفی انتخاب می‌شود.
  - قسمت‌های چپ و راست کروموزوم‌ها بین والدین جابجا می‌شود.
- مثال: نقطه برش = ۳
- والدین:
  - P_1=[1, 0, 1, |0, 1, 1]
  - P_2=[0, 1, 0, |1, 0, 0]
- فرزندان:
  - CH_1=[1, 0, 1, 1, 0, 0]
  - CH_2=[0, 1, 0, 0, 1, 1]



انواع روش‌های ترکیب

- **ترکیب دو نقطه‌ای (Two-Point Crossover)**
  - دو نقطه برش تصادفی انتخاب می‌شود.
  - بخش میانی کروموزوم‌ها جابجا می‌شود.
- مثال: نقاط برش: ۲ و ۵
- والدین:
  - P_1=[1, 0,|1, 0, 1,|1]
  - P_2=[0, 1,|0, 1, 0,|0]
- فرزندان:
  - CH_1=[1, 0, 0, 1, 0, 1]
  - CH_2=[0, 1, 1, 0, 1, 0]

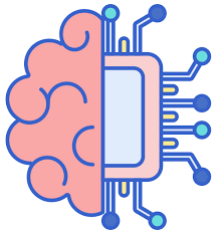


انواع روش‌های ترکیب

- **ترکیب یکنواخت (Uniform Crossover)**
  - هر ژن از والدین به‌طور تصادفی انتخاب می‌شود.

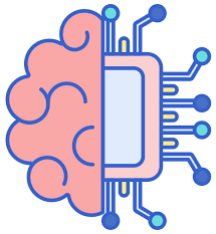
- مثال
- والدین:
  - P_1=[1, 0, 1, 0, 1, 1]
  - P_2=[0, 1, 0, 1, 0, 0]

- انتخاب تصادفی: [1, 1, 1, 1, 0, 1]



انواع روش‌های ترکیب

- **ترکیب تکه‌ای (Segment-Based Crossover)**
  - یک تکه خاص از یک والد به‌طور کامل به فرزند منتقل می‌شود.
- مثال:
- والدین:
  - $P_1=[1,0,1,0,1,1]$
  - $P_2=[0,1,0,1,0,0]$
- بخش انتخابی از $P_1=[1,0]$ و از $P_2=[1,0,0]$



جهش (Mutation)

- فرآیندی است که تغییرات تصادفی در کروموزوم‌ها ایجاد می‌کند. این تغییرات معمولاً کوچک هستند اما نقش مهمی در حفظ تنوع و جلوگیری از همگرایی زودرس دارند.

- نقش جهش
  - **حفظ تنوع:** با ایجاد تغییرات تصادفی، از یکنواخت شدن جمعیت جلوگیری می‌کند.
  - **اکتشاف فضای جستجو:** جهش می‌تواند کروموزوم‌ها را به نواحی جدیدی در فضای جستجو ببرد.
  - **اصلاح همگرایی زودرس:** اگر الگوریتم به یک بهینه محلی برسد، جهش می‌تواند آن را از این وضعیت خارج کند.

انواع روش‌های جهش

- **جهش معکوس (Bit Flip Mutation)**
  - یک بیت تصادفی در کروموزوم تغییر می‌کند.
- مثال:
  - کروموزوم: [1, 0, 1, 0, 1]
  - فرزند: [1, 0, **0**, 0, 1]

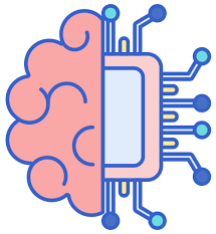
- **جهش تبادلی (Swap Mutation)**
  - دو ژن تصادفی در کروموزوم جابجا می‌شوند.
- مثال:
  - کروموزوم: [1, 0, 1, 0, 1]
  - فرزند: [1, 0, **0**, **1**, 1]

انواع روش‌های جهش

- **جهش درج (Insertion Mutation)**
  - یک ژن انتخاب و در موقعیت دیگری وارد می‌شود.

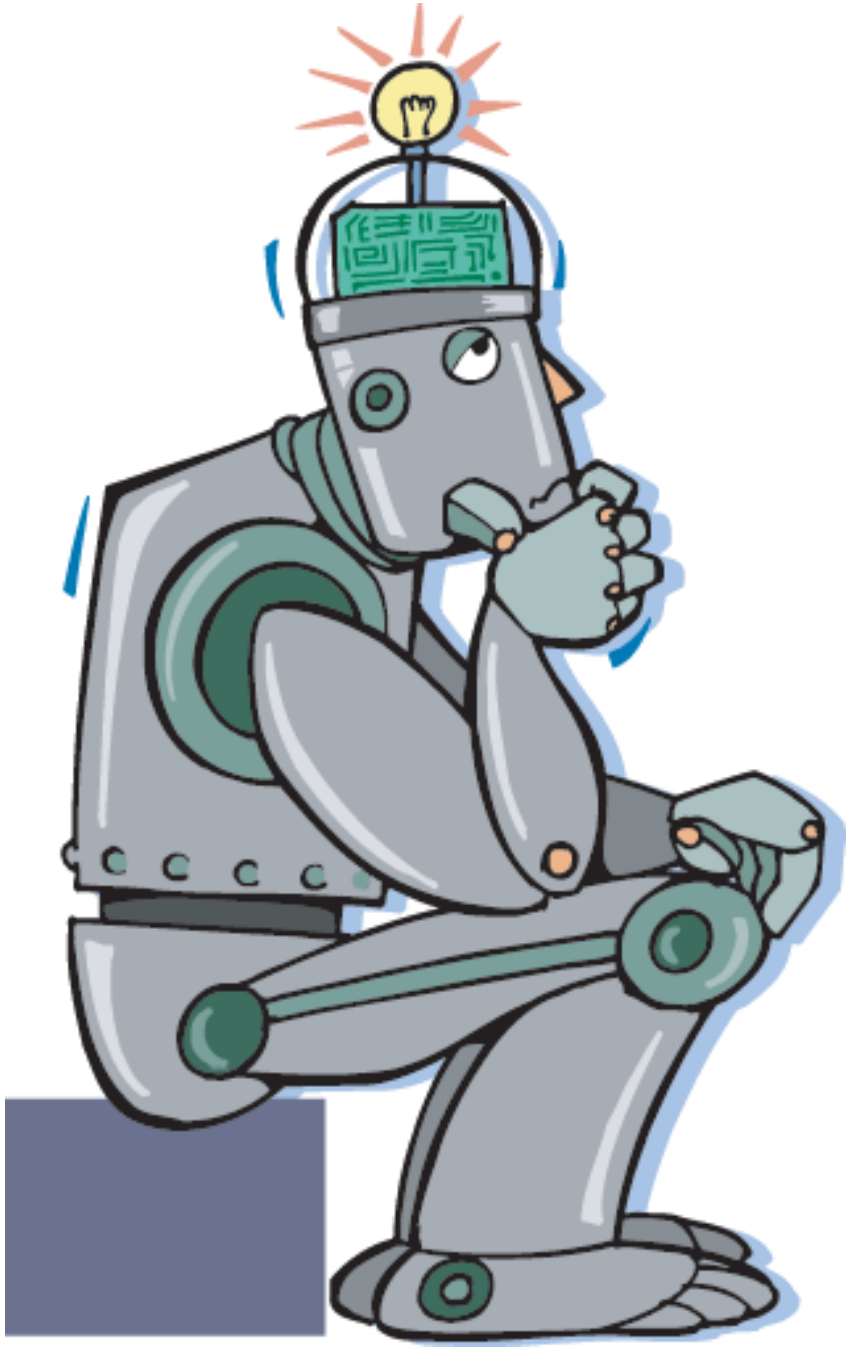
- مثال:
  - کروموزوم: [1, 0, 1, 0, 0]
  - فرزند: [1, 0, **0**, 0, **1**]

- **جهش ترکیبی (Combination Mutation)**
  - ترکیبی از جهش های مختلف در یک کروموزوم اعمال می‌شود.



یک الگوریتم ژنتیک ساده

- با جمعیتی از کروموزوم‌های n بیتی که به‌طور تصادفی تولید می‌شوند (راه‌حل‌های نامزد برای یک مسئله) شروع کنید.
- تابع برازندگی f(x) هر کروموزوم x در جمعیت را محاسبه کنید.
- مراحل زیر را تکرار کنید تا n فرزند ایجاد شود:
  - یک جفت کروموزوم والد از جمعیت فعلی انتخاب کنید. (احتمال انتخاب تابعی از برازندگی) انتخاب با جایگزینی انجام می شود، به این معنی که همان کروموزوم را می توان بیش از یک بار برای تبدیل شدن به والد انتخاب کرد.
  - با احتمال p_c (احتمال ترکیب) یا (نرخ ترکیب)، جفت را در یک نقطه به طور تصادفی انتخاب شده ترکیب کنید تا دو فرزند تشکیل شود. اگر تلاقی صورت نگرفت، دو فرزند کپی والدین مربوطه تشکیل دهید.
  - دو فرزند را در هر مکان با احتمال p_m (احتمال جهش یا نرخ جهش) جهش دهید و کروموزوم های حاصل را در جمعیت جدید قرار دهید. اگر n فرد باشد، یک عضو جدید جمعیت را می توان به طور تصادفی کنار گذاشت.
- جمعیت فعلی را با جمعیت جدید جایگزین کنید.
- به مرحله ۲ بروید.



سؤال؟