انواع فرامکاشفه های تکاملی

انواع فرامکاشفه های تکاملی

- استراتژی تکامل
- تکامل تفاضلی
- الگوریتم ممتیک
- الگوریتم های فرهنگی
- ژنتیک تاگوچی
- الگوریتم هم تکاملی
  - هم تکاملی رقابتی
  - هم تکاملی همکاری
- الگوریتم تکاملی دیپلوییدی
- بهینه سازی تولید مثل غیر جنسی
- سیستم ایمنی مصنوعی

استراتژی تکامل

- فرآیند های زیست شناختی توسط تکامل بهینه می شوند ⟸ باید بتوان تکامل را تبدیل
- تکامل خود یک فرآیند زیست شناختی است

در استراتژی تکامل :

- هم به تکامل فنوتایپی توجه داریم
- هم به تکامل ژنوتایپی
- اما تأکید بیشتر بر تکامل رفتار ژنوتایپی موجودات است.

- هر موجود توسط ① بلوک های سازنده ژنی خود
  ② مجموعه‌ای از پارامترهای استراتژی که رفتار موجودات محیط را مدل می‌کنند ⟸ تع

- ویژگی های ژنی و پارامترهای استراتژی بلاهم تکامل می‌یابند
- در حالی که تکامل ویژگی های ژنی توسط پارامترهای استراتژی ک

کاربرد

- بهینه سازی آزمایش‌هایی برای سلول هیدرودینامیکی
- حل مسائل بهینه سازی تابعی
- طراحی کنترل کننده
- طراحی ترانسفورماتور و سیستم‌های قدرت

استراتژی تکامل عبارتند از:

- استفاده از نمایش اعداد حقیقی (ممیز شناور).
- بکارگیری یک جمعیت با تعداد اعضاء ثابت.
- عملگر جهش در استراتژی تکامل عملگر اصلی است (در برخی از روش‌های مبتنی بر استرا
  تکامل از عملگر بازترکیب نیز استفاده شده است. لیکن استفاده از این عملگر موجب دو
  شدن از ماهیت استراتژی تکامل می‌شود).
- الگوریتم عملگر جهش و پارامترهای آن طی فرآیند استراتژی تکامل، تکامل می‌یابند.

---

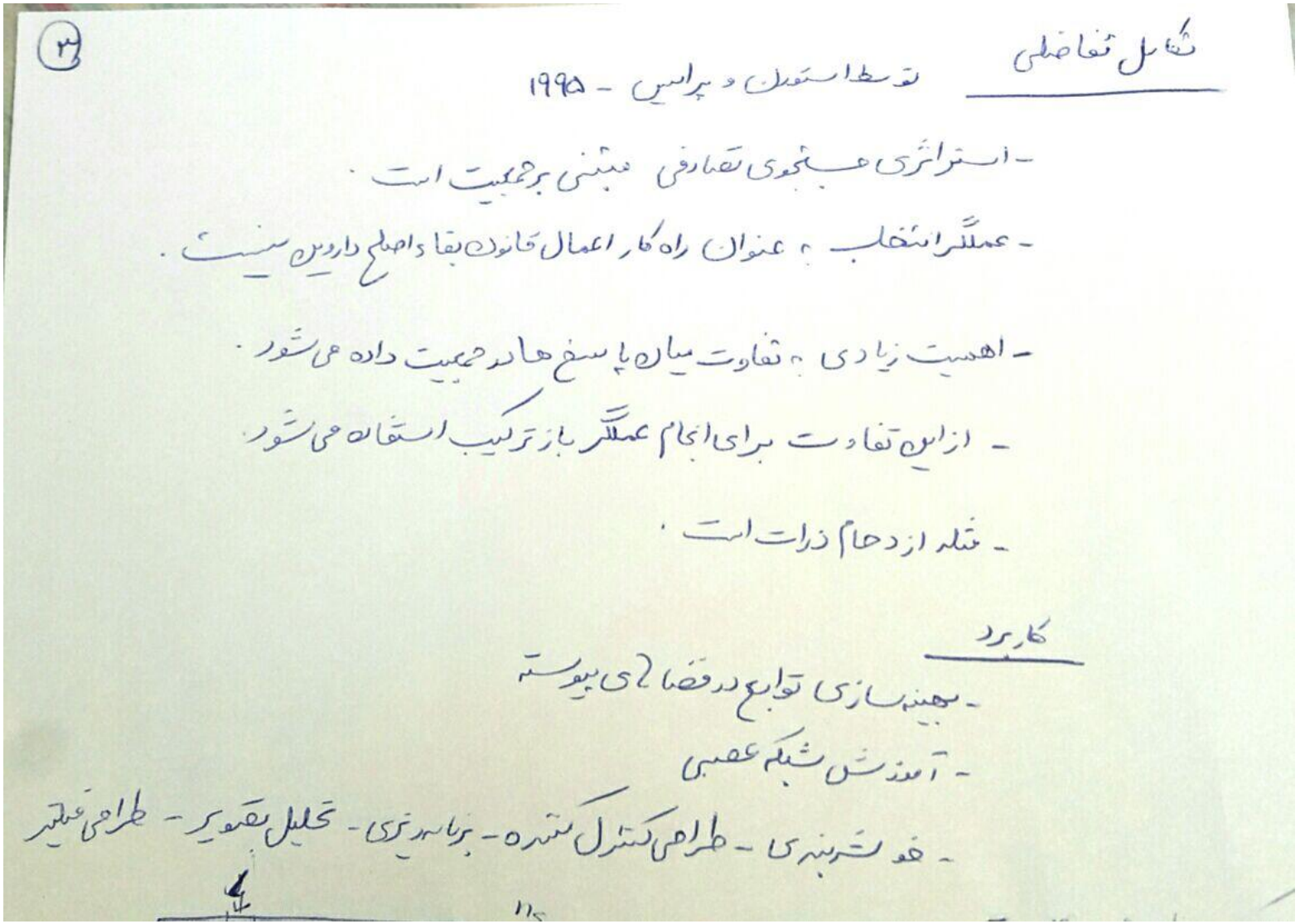
echenberg

```
Function ES(problem) returns a state that is a local optimum
Input: μ, λ, ProblemSize
Output: Sbest

Population ı ← InitializePopulation(μ, ProblemSize);
EvaluatePopulation(Population);
Sbest ← GetBestSolution(Population);
while ¬ StopCondition() do
  Children ← ∅;
  for i = 0 to λ do
    Pi ← GetParent(Population, i);
    Si ← ∅;
    Siproblem ← Mutate(Piproblem, Pistrategy);
    Sistrategy ← Mutate(Pistrategy);
    Children ← Si;
  end
  EvaluatePopulation(Children);
  Population ← Replace(Population, Children, μ);
  Sbest ← GetBestSolution(Population);
end
return Sbest;
```

الگوریتم ۳-۳ شبه کد استراتژی تکامل

- پارامترهای استراتژی، همراه هر کدام از موجودات می‌باشند. این پارامترهای استراتژی خود تطبیق می‌باشند تا بهترین جهت جستجو و بیشترین اندازه گام هر بعد را تعیین کنند.
- نرخ جهش با توجه به برازش پاسخ‌ها به صورت تطبیقی تنظیم می‌شود. برای پاسخ‌های برازنده نرخ جهش کم و برای پاسخ‌های با برازش پایین این نرخ بالا در نظر گرفته می‌شود.
- نرخ جهش بایستی منجر به موفقیت در بهبود برازش پاسخ در ۲۰٪ موارد شود. در صورتیکه درصد موفقیت بیشتر از ۲۰٪ بود نرخ جهش افزایش می‌یابد و در غیر این‌صورت این نرخ کاهش خواهد یافت.
- احتمال جهش با توجه به یک تابع توزیع احتمال (مثلاً گوسین) محاسبه می‌شود.
- منظور از استراتژی تکامل $(\mu/\rho, \lambda)-ES$ آن است که از μ والد λ فرزند تولید می‌شود. در ضمن در مرحله تولید فرزند، از هر ρ والد، یک فرزند تولید می‌شود.
- در جایگزینی انتخاب $(\mu + \lambda)$، از مجموع جمعیت والدین و فرزندان $(\mu+\lambda)$، تعداد μ پاسخ با توجه به یک استراتژی انتخاب، گزینش می‌شوند.
- در جایگزینی انتخاب (μ,λ)، تعداد μ عضو برتر از تعداد λ فرزند، گزینش شده و به نسل بعد منتقل می‌شوند.
- همیشه رابطه $\lambda \geq \mu \geq 1$ برقرار است.

تکامل تفاضلی



تکامل تفاضلی توسط استفن و پراسیر - ۱۹۹۵

- استراتژی جستجوی تصادفی مبتنی بر جمعیت است.
- عملگر انتخاب به عنوان راه کار اعمال قانون بقا و اصلح داروین است.
- اهمیت زیادی به تفاوت میان پاسخ ها در جمعیت داده می شود.
- از این تفاوت برای انجام عملگر باز ترکیب استفاده می شود.
- فاصله از دهام ذرات است.

کاربرد

- بهینه سازی توابع در فضاهای پیوسته
- آموزش شبکه عصبی
- فوشربندی - طراحی کنترل کننده - برنامه ریزی - تحلیل تصویر - طراحی عکس

تکامل تفاضلي

```
Function DE(problem) returns a state that is a local maximum
Input: Populationsize, Problemsize, Weightingfactor, Crossoverrate
Output: Sbest

Population ← InitializePopulation(Populationsize, Problemsize);
EvaluatePopulation(Population);
Sbest ← GetBestSolution(Population);
while ¬ StopCondition() do
  NewPopulation ← ∅;
  foreach Pi ∈ Population do
    Si ← NewSample(Pi, Population, Problemsize, Weightingfactor, Crossoverrate);
    if Fitness(Si) ≥ Fitness(Pi) then
      NewPopulation ← Si;
    else
      NewPopulation ← Pi;
    end
  end
  Population ← NewPopulation;
  EvaluatePopulation(Population);
  Sbest ← GetBestSolution(Population);
end
return Sbest;
```

الگوریتم ۳-۵ شبه‌کد تکامل تفاضلی

```
Function NewSample() returns a new solution
Input: P0, Population, n_s, β, p_r
Output: S

repeat
  P1 ← RandomMember(Population);
until P1 ≠ P0 ;
repeat
  P2 ← RandomMember(Population);
until P2 ≠ P0 ∨ P2 ≠ P1 ;
repeat
  P3 ← RandomMember(Population);
until P3 ≠ P0 ∨ P3 ≠ P1 ∨ P3 ≠ P2 ;
CutPoint ← RandomPosition(n_s);
S ← 0;
for i to n_s do
  if i ≡ CutPoint ∧ Rand() < p_r then
    Si ← P3i + β × (P1i - P2i);
  else
    Si ← P0i ;
  end
end
return S;
```

الگوریتم ۳-۶ شبه‌کد تابع تولید پاسخ جدید

تکامل تفاضلی

شبه‌کد تکامل تفاضلی در الگوریتم ۳-۵ ارائه شده است. خصوصیات مهم تکامل تفاضلی عبارتند از:

- تکامل تفاضلی به دلیل انجام عملگر بازترکیب بر روی کلیه اعضاء و نه فقط بهترین اعضاء، در مقایسه با سایر الگوریتم‌های تکاملی توانایی پویش بیشتری دارد.
- برخلاف استراتژی‌های تکامل و برنامه‌نویسی تکاملی، در تکامل تفاضلی اندازه‌های گام از یک توزیع شناخته شده از قبل نمونه‌گیری نمی‌شوند.
- اندازه‌های گام، تحت تأثیر تفاوت بین موجودات جمعیت کنونی است.
- میزان فاصله اولیه بین موجودات تحت تأثیر اندازه جمعیت است. هر چقدر که موجودات در یک جمعیت بیشتر باشند، فاصله بین آنها کمتر است.
- اندازه جمعیت دارای تأثیر مستقیم بر روی قابلیت پویش تکامل تفاضلی است. هر چقدر که تعداد موجودات در جمعیت بیشتر باشد، بردار تفاضلی بیشتری در دسترس می‌باشد و جهت‌های بیشتری قابل جستجو هستند. در هر حال باید توجه داشت که، پیچیدگی محاسباتی هر نسل با اندازه جمعیت افزایش می‌یابد.

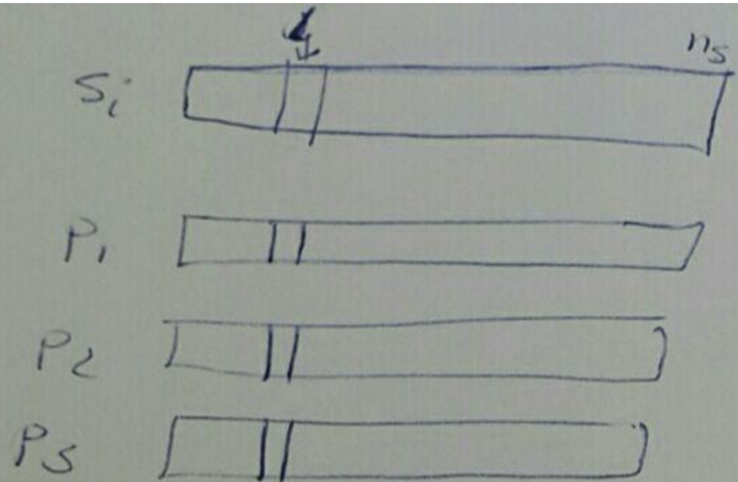
تکامل تفاضلی

- در الگوریتم ۳-۶، هر چقدر که مقدار β کوچک‌تر باشد اندازه‌های گام جهش کوچک‌تر می‌شود و زمان بیشتری برای همگرایی الگوریتم مورد نیاز است. مقدار بزرگ‌تر β قابلیت پویش را افزایش می‌دهد اما ممکن است که باعث شود الگوریتم از پاسخ به اندازه کافی خوب خارج شود. مقدار β باید به اندازه‌ای کوچک باشد که امکان پویش قله‌های تیز (دره‌های تنگ) را فراهم کند و به اندازه‌ای بزرگ باشد که تنوع جمعیتی ایجاد نماید. زمانی که اندازه جمعیت افزایش می‌یابد، بایستی مقدار β افزایش یابد.
- هر چقدر که تعداد موجودات در جمعیت بیشتر باشد، مقدار بردارهای تفاضلی کوچک‌تر می‌شود و موجودات به یکدیگر نزدیکتر خواهند شد. بنابراین، می‌توان از گام‌های کوچک‌تری

تکامل تفاضلی

برای پویش مکان‌های محلی استفاده نمود. تعدد موجودات در جمعیت، نیاز به گام‌های جهش بزرگ را کاهش می‌دهد. نتایج تجربی نشان می‌دهد که مقادیر بزرگ برای n_s و β در الگوریتم ۳-۶، اغلب منجر به همگرایی زودرس می‌شود [۵۷، ۱۷]. مقدار $\beta = 0.5$ معمولاً کارآیی خوبی دارد [۱۳۰، ۲۰، ۲]. توجه شود که همیشه داریم: $\beta \in (0, \infty)$.

- در الگوریتم ۳-۶، احتمال تولیدمثل p_r، دارای اثر مستقیم بر تنوع در تکامل تفاضلی است. این پارامتر تعداد عناصری از والد $x_i(t)$، را که تغییر خواهند کرد، کنترل می‌کند. هر چقدر که احتمال تولیدمثل بیشتر باشد، تنوع بیشتری در جمعیت جدید ایجاد می‌شود، که منجر به افزایش قابلیت پویش خواهد شد. افزایش p_r، اغلب منجر به همگرایی سریعتر می‌شود. در حالی که کاهش آن قدرت جستجو را افزایش می‌دهد [۲۰، ۵۷].
- پارامترهای تفاضل تکاملی می‌توانند ثابت نبوده و می‌توانند در فرآیند جستجویی الگوریتم به صورت تطبیقی تغییر نمایند.
- امکان بهره‌برداری از جفت پاسخ‌های بیشتر در هر نسل (به‌جای تنها دو پاسخ P_{1i} و P_{2i} در الگوریتم ۳-۶) برای پویش جهت‌های بیشتر در فضای جستجو وجود دارد.



با ترتیب به اندازه همه جمعیت

```
cutpoint = randomposition(ns)
S ← 0
for i = 1 to ns
    if i = cutpoint ∧ Rand() < Pr
        S_i = P3_i + β × (P1_i − P3_i)
    else
        S_i ← P0_i
```

وقتی اندازه جمعیت افزایش یابد گام هم افزایش می‌یابد

در الگوریتم اصلی

```
for P_i ∈ population
    S_i ← newsample(-- .)
    if fitness(S_i) ≥ fitness(P_i)
        new population ← S_i
    else
        newpopulation ← P_i
```

الگوریتم ممتیک

مثال‌هایی از مم عبارت است از:

- خصیصه‌ها و هنجارهای مثبت و منفی در یک جامعه؛ که ریشه در حقیقتی فرهنگی دارنـد و در انتقال‌شان از مفاهیم ژنی بهره‌ای نمی‌برند.
- مد لباس؛ که آخرین مدهای لباس نتیجه ایده‌های طراحان مد هستند.
- علوم؛ دانشمندان نظریات خود را با یکدیگر در میان می‌گذارند و دانش را توسعه می‌دهند.
- ادبیات؛ به چاپ رسیدن رمان‌های نویسندگان و اشعار شاعران که موجب تقلیدی از یک سبک و یا اثر ادبی می‌شود.
- موسیقی؛ که علاوه بر موسیقیدانان، حتی پرندگان نیز از نوای موسیقی یکدیگر تقلید می‌کنند.

شباهت‌های بین ژن و مم عبارت است از:

- ژن‌ها از کروموزومی به کروموزوم دیگر انتقال می‌یابند (به وسیله تولیدمثل) و مم‌ها نیز از مغزی به مغز دیگر انتقال پیدا می‌کنند (به وسیله تقلید کردن).
- در دوران تکامل ژنی بهترین ژن‌ها و در سال‌های زندگی بهترین مم‌ها باقی می‌مانند.

الگوریتم ممتیک

تفاوت‌های بین ژن‌ها و مم‌ها را می‌توان در موارد زیر دانست:

- مقادیر ژن‌ها از پیش مشخص شده هستند. مثلاً پدر و مادر سفید پوست فرزند سفید پوست خواهند داشت. این در حالی است که مقادیر مم‌ها به هیچ عنوان، تا قبل از زندگی موجود، قابل تخمین نخواهند بود.
- ژن‌ها به طور معمول در طی نسل‌های مختلف ثابت هستند، در حالی که مم‌ها در طول یک دوره زندگی مرتباً در حال تغییرند.
- مم‌ها بهینگی و اصلاح را در نسل حاضر امکان‌پذیر می‌سازند در حالی که این امکان در ژن‌ها در طی نسل‌های طولانی فراهم می‌شود (زیرا معجزه‌ی تکامل تنها با طی یک زمان بسیار طولانی رخ می‌دهد).

و نیز طیف وسیعی
۳-۷ ارائه شده است. خصوصیات مهم الگوریتم ممتیک عبارتند از:

- به منظور افزایش سرعت و کارآیی در الگوریتم ممتیک، جستجوی محلی بر روی همه اعضاء در جمعیت انجام نمی‌شود. به همین دلیل معمولاً جمعیت کوچکی از فرزندان برازنده پس از عملگر جهش به عملگر جستجوی محلی معرفی می‌گردند.
- الگوریتم‌های ممتیک به دو دسته مهم لامارکی[۱] و بالدوینی[۲] تقسیم می‌شوند.
- اگر کروموزوم y نتیجه عملگر جستجوی محلی بر روی کروموزوم x باشد و داشته باشیم Fitness(y)>Fitness(x) (البته با فرض اینکه در حال حل یک مسأله بیشینه‌سازی باشیم) در الگوریتم‌های ممتیک مبتنی بر نظریه لامارک، کروموزوم y جایگزین کروموزوم x می‌شود. این در حالی است که در همین شرایط، در الگوریتم‌های ممتیک بالدوینی، تنها برازش کروموزوم y جایگزین برازش کروموزوم x می‌شود. در واقع در الگوریتم ممتیک مبتنی بر نظریه بالدوین تنها به عملگرهای تولیدمثل اجازه تغییر کروموزوم‌ها داده می‌شود.
- سرعت همگرایی در الگوریتم‌های ممتیک لامارکی بیشتر از الگوریتم‌های ممتیک بالدوینی است. این در حالی است که قابلیت پویش الگوریتم‌های ممتیک بالدوینی بسیار بیشتر از نوع لامارکی است.
- ویژگی خاص الگوریتم‌های ممتیک بالدوینی، عدم تغییر ژنوتایپ کروموزوم‌ها در مرحله زندگی (جستجوی محلی) الگوریتم است. این ویژگی باعث می‌شود که برازش نقاط همسایه بهینه محلی بهبود یابد. این کار، باعث اعمال تغییر در دورنمای برازش در این نقاط می‌شود. با این عمل، بخش‌هایی از برازش که ماهیتی خارپشتی دارند هموار شده و پیمایش فضای جستجو برای الگوریتم ساده‌تر خواهد شد. در بسیاری از مقالات و کتب، به هموارسازی (ساده‌سازی) دورنمای برازش توسط جستجوگر محلی مبتنی بر نظریه بالدوین، اثر بالدوین[۳] گفته می‌شود.
- می‌توان بجای جایگزینی پاسخ جدید در جستجوگر محلی (نظریه لامارک) و یا عدم جایگزینی پاسخ (نظریه بالدوین)، با یک احتمال ρ_l این جایگزینی را انجام داد. با این کار، در وضعیت $\rho_l = 1$، الگوریتم ممتیک لامارکی و در هنگام $\rho_l = 0$، الگوریتم ممتیک بالدوینی حاصل می‌شود.
- الگوریتم‌های ممتیک تطبیقی، بجای استفاده از تنها یک جستجوگر محلی ثابت، از مجموعه‌ای

```
Function MA(problem) returns a state that is a local optimum
Input: Populationsize, Problemsize, Pcrossover, Pmutation, MemePopsize
Output: Sbest

Population ← InitializePopulation(Populationsize, Problemsize);
EvaluatePopulation(Population);
Sbest ← GetBestSolution(Population);
while ¬ StopCondition() do
  Parents ← SelectParents(Population, Populationsize);
  Children ← ∅;
  foreach Parent1, Parent2 ∈ Parents do
    Child1, Child2 ← Crossover(Parent1, Parent2, Pcrossover);
    Children ← Mutate(Child1, Pmutation);
    Children ← Mutate(Child2, Pmutation);
  end
  EvaluatePopulation(Children);

  MemeticPopulation ← SelectMemeticPopulation(Children, MemePopsize);
  foreach Si ∈ MemeticPopulation do
    Si ← LocalSearch(Si);
  end

  Population ← Replace(Population, Children);
  Sbest ← GetBestSolution(Population);
end
return Sbest;
```

الگوریتم ۳-۷ شبه کد الگوریتم ممتیک

الگوريتمهاي فرهنگي

تعریف‌های گوناگونی برای فرهنگ می‌توانیم داشته باشیم، برای مثال:

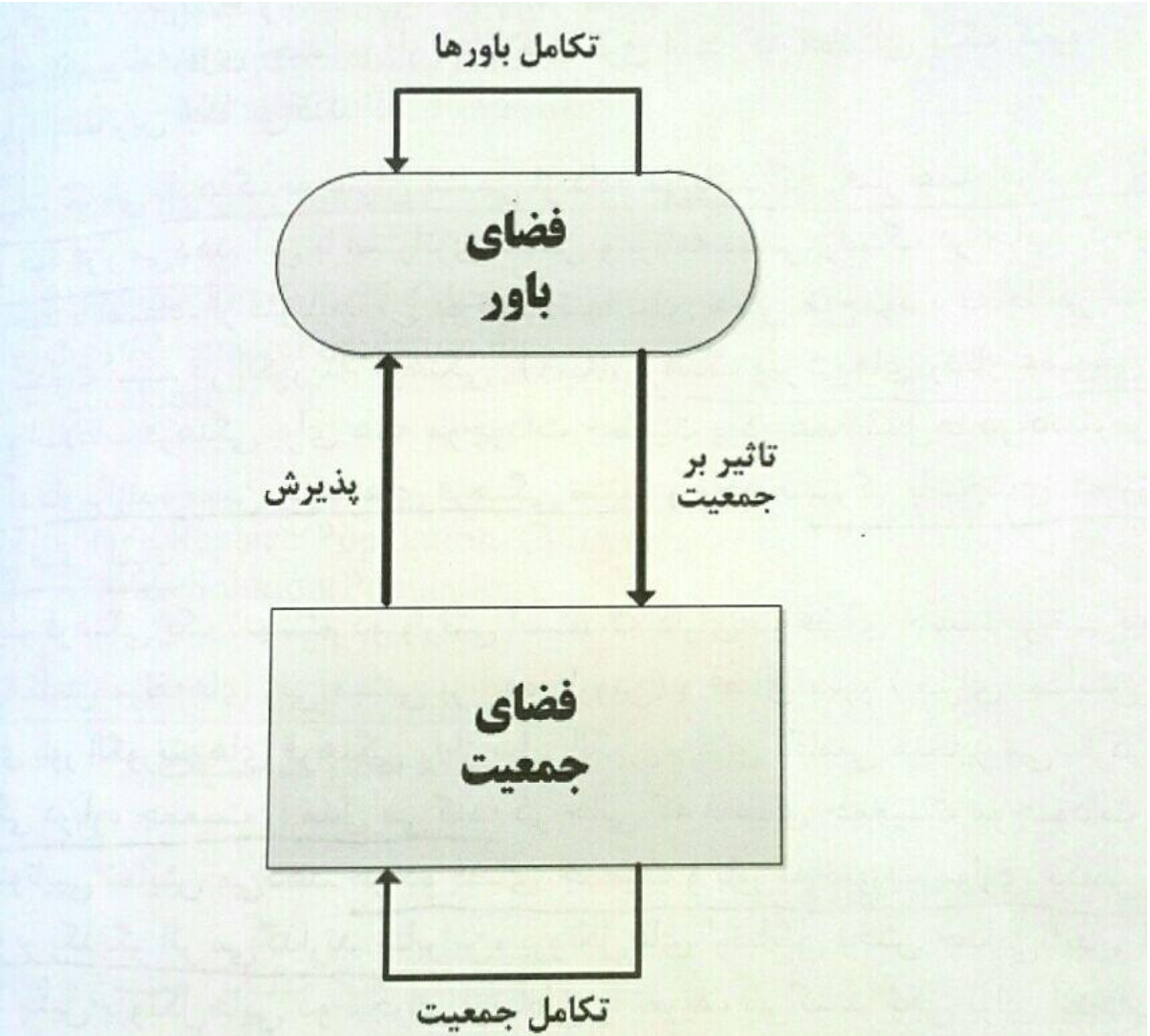
- فرهنگ، سیستمی از پدیده‌های مفهومی و نمادی است که به صورت اجتماعی و تاریخی میان گروه‌های اجتماعی منتقل می‌شود.
- فرهنگ به ذخایری از دانش‌ها، تجارب، باورها، ارزش‌ها، افکار، مفاهیم، رهبران، مذاهب، نقش‌ها، روابط زمانی و فضایی، تدابیر جهان و اشیاء مادی و دارایی‌های بدست آمده از گروهی از مردم در طول یک نسل از طریق تلاش گروهی و فردی اطلاق می‌شود.
- فرهنگ، کلیه رفتارهای یاد گرفته شده گروهی از مردم است که عموماً به عنوان سنت آن مردم تلقی می‌شود و از نسلی به نسل دیگر منتقل می‌شود.
- فرهنگ یک برنامه‌ریزی اجتماعی از تمایلات فکری است که اعضای یک گروه و یا دسته از مردم را از سایرین جدا می‌کند.

الگوريتم هاي فرهنگي

- فرهنگ به عنوان داده میباشد و رفتار همه افراد در یک جمعیت را تحت تاثیر خود قرار می دهد.
- فرهنگ ویژگیهای رفتار عمومی جمعیت را ذخیره می کند.
- وجه مشترک اعضای برازنده مبین پدیده های فرهنگی مثبت و وجه مشترک پاسخ های کم ارزش بیان گر حقایق فرهنگی منفی است.
- الگوریتم فرهنگی دارای دو فضای جستجو است.
- فضای جمعیت: برای نمایش مولفه های ژنی
- فضای باور: که اطلاعات فرهنگی درباره جمعیت را مدل می کند.(استخر مم) و مانند یک انبار دانش عمل می کند.
- هر دو فضا به صورت موازی تکامل می یابندو هر دو بر هم اثر می گذارند.
- مم در فضای باورتعمیم تجارب افراد در فضای جمعیت است.
- مم واحدهای اطلاعاتی منتقل شونده توسط مفاهیم رفتاری است.
- تعمیم تجربه در طول چندین نسل جمع آوری شده و شکل می گیرند و تنها برای یک نسل نیستند.

```
Function CA(problem) returns a state that is a local optimum
Input: Problem_size, Population_num
Output: S_best

Population ← InitializePopulation(Problem_size, Population_num);
Belief Space ← Initialize BeliefSpace (Problem_size, Population_num);

S_best ← GetBestSolution(Population);
while ¬ StopCondition() do
  EvaluatePopulation(Population);
  Children ← ReproduceWithInfluence(Population, Belief Space);
  Population ← Select(Children, Population);

  Belief_candidate ← AcceptBelief(Population);
  UpdateBelief Space(Belief Space, Belief_candidate);

  S_best ← GetBestSolution(Population);
end
return S_best;
```

الگوریتم ۳-۸ شبه کد الگوریتم فرهنگی



شکل ۳-۱ ارتباط فضاهای باور و جمعیت در الگوریتم‌های فرهنگی

عملگرهای تولید مثل از باورها برای کنترل تغییرات موجودات استفاده می کنند.

الگوریتم فرهنگي

شبه‌کد الگوریتم فرهنگی در الگوریتم ۳-۸ ارائه شده است. خصوصیات مهم الگوریتم فرهنگی عبارتند از:

- برای انتخاب گروهی از موجودات برای تطبیق‌پذیری مجموعه باورها از تابع AcceptBelief استفاده می‌شود و برای اعمال تأثیر باورها بر روی همه موجودات در فضای جمعیت از تابع ReproduceWithInfluence بهره‌برداری می‌گردد (الگوریتم ۳-۸).
- انواع مختلفی از الگوریتم‌های فرهنگی، توسعه یافته است که در ساختمان داده‌های مورد استفاده در مدل کردن فضای باور با یکدیگر متفاوت هستند.
- فضای باور شامل مجموعه‌های باور است. مجموعه باور شامل تعدادی مؤلفه دانش می‌باشد تا الگوهای رفتاری موجودات در فضای جمعیت را نشان دهد. نوع مؤلفه‌های دانش و ساختمان داده‌های مورد استفاده برای نمایش دانش بستگی به مسأله مورد بررسی دارد.
- مؤلفه دانش موقعیتی[1]: این مؤلفه، شامل بهترین راه‌حل‌های پیدا شده در فرآیند تکاملی و یا وجه مشترک پاسخ‌های برازنده می‌باشد.
- مؤلفه‌های دانش هنجار[2]: این مؤلفه، استانداردهایی برای رفتار موجودات فراهم می‌کند که برای هدایت تنظیمات جهشی (مانند گام جهش، احتمال جهش و غیره) در بخش تکامل جمعیت الگوریتم استفاده می‌شود. در مورد مسأله بهینه‌سازی تابع، مؤلفه دانش هنجار مجموعه‌ای از فواصل را نگهداری می‌کند، که هر کدام مربوط به بخش‌های متفاوتی از دورنمای برازش هستند

الگوریتم فرهنگي

- با توجه به نیاز مسأله، در مجموعه باور امکان استفاده از مؤلفه‌های دانش دیگری نیز وجود دارد. به عنوان مثال، در مسائلی که ممکن است دورنمای برازش طی زمان تغییر کند (مسائل بهینه‌سازی با دورنمای برازش پویا)، از یک مؤلفه دانش با نام مؤلفه دانش سابقه استفاده می‌شود. این مؤلفه، اطلاعاتی درباره دنباله تغییرات محیط را نگه می‌دارد. از این مؤلفه برای پیش‌بینی تغییرات آینده در فضای جستجوی مسأله جهت اخذ بهترین تصمیم استفاده می‌شود (هر پاسخ یا موجود در جمعیت یک تصمیم است).
- نوع مؤلفه‌های دانش و روشی که دانش نمایش داده می‌شود، بر روی توابع تأثیر و پذیرش، تأثیر می‌گذارد.
- تابع پذیرش (AcceptBelief در الگوریتم ۳-۸)، تعیین می‌کند که کدامیک از موجودات از

جمعیت کنونی، برای شکل‌گیری باورها، استفاده خواهند شد. روش‌های ایستا، از رتبه‌بندی مطلق مبتنی بر مقدار برازش استفاده می‌کنند و n٪ از موجودات برتر را انتخاب می‌کنند. در اینجا می‌توان از هر کدام از روش‌های انتخاب در الگوریتم‌های تکاملی استفاده نمود؛ به عنوان مثال انتخاب مسابقه‌ای یا انتخاب چرخ رولت، که در آنها تعداد موجودات انتخاب شده برابر با موجودات اولیه می‌باشد. روش‌های پویا تعداد ثابتی موجود برای تنظیم فضای باور ندارند و تعداد موجودات ممکن است که از نسلی به نسلی دیگر تغییر کند. در اینجا می‌توان از رتبه‌بندی نسبی استفاده کرد که به عنوان مثال در آن موجوداتی که دارای کارآیی بیشتر از مقدار میانگین هستند انتخاب شوند [۱۸].

- تعداد موجوداتی که برای تنظیم فضای باور مورد استفاده قرار می‌گیرند در ابتدا بزرگ می‌باشد و در طول زمان به صورت نمایی کاهش می‌یابد.
- روش‌های تطبیق‌پذیر از اطلاعات فضای جستجو و فرآیند تکاملی الگوریتم برای تنظیم تعداد موجوداتی که باید انتخاب شوند، استفاده می‌کنند. رینولدز و چانگ[1] [۱۰۷]، یک تابع پذیرش فازی را برای تعیین تعداد موجودات، که مبتنی بر شماره نسل و نرخ موفقیت موجودات است، پیشنهاد نموده‌اند.
- هنگامی که الگوریتم در تنظیم مؤلفه دانش هنجار، گام‌های جهش را کوچک‌تر می‌کند، یک شیوه محافظه‌کارانه بکار می‌رود. بدین ترتیب قابلیت پویش در الگوریتم تضعیف شده و انتفاع در آن تقویت می‌گردد.
- باورها برای تنظیم موجودات در فضای جمعیت برای تطبیق بیشتر با عقاید عمومی استفاده می‌شوند. تنظیمات از طریق تابع تأثیر (ReproduceWithInfluence در الگوریتم ۳-۸) تحقق می‌پذیرد. برای درک این فرآیند، فرض کنید که EP به عنوان فرآیند جستجو در فضای جمعیت مورد استفاده قرار می‌گیرد. الگوریتم حاصله CAEP نامیده می‌شود. فضای باور برای تعیین اندازه‌های گام جهش در EP و جهت تغییرات (یعنی اندازه‌های گام اضافه یا کم شود) استفاده می‌شود.

ژنتیک تاگوچی

- یکی از مشکلات ژنتیک استاندارد عدم ثابت بودن نتایج در اجراهای مختلف آن است.
- نتایج پایدار نیست
- راه حل آن ممتیک و روش تاگوچی است
- درتاگوچی سعی می شود که تاثیر هر کدام از ژنها در نتیجه برازش نهایی یک کروموزوم سنجیده شود.
- مقدار مناسب یک ژن از میان مقادیر ژنهای دو یا چند کروموزوم پایه پیدا می شود.
- ار مفهوم آرایه متعامد برای تولید مجموعه ای از کروموزومهای پایه استفاده میشود.

ژنتیک تاگوچي

Experiment number	Factors						
	A	B	C	D	E	F	G
	Column number						
	1	2	3	4	5	6	7
1	1	1	1	1	1	1	1
2	1	1	1	2	2	2	2
3	1	2	2	1	1	2	2
4	1	2	2	2	2	1	1
5	2	1	2	1	2	1	2
6	2	1	2	2	1	2	1
7	2	2	1	1	2	2	1
8	2	2	1	2	1	1	2

شکل ۳-۲ آرایه متعامد دو بعدی با ۷ ورودی

Experiment number	Column number														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2
3	1	1	1	2	2	2	2	1	1	1	1	2	2	2	2
4	1	1	1	2	2	2	2	2	2	2	2	1	1	1	1
5	1	2	2	1	1	2	2	1	1	2	2	1	1	2	2
6	1	2	2	1	1	2	2	2	2	1	1	2	2	1	1
7	1	2	2	2	2	1	1	1	1	2	2	2	2	1	1
8	1	2	2	2	2	1	1	2	2	1	1	1	1	2	2
9	2	1	2	1	2	1	2	1	2	1	2	2	1	1	2
10	2	1	2	1	2	1	2	2	1	2	1	1	2	2	1
11	2	1	2	2	1	2	1	1	2	1	2	2	1	2	1
12	2	1	2	2	1	2	1	2	1	2	1	1	2	1	2
13	2	2	1	1	2	2	1	1	2	2	1	1	2	2	1
14	2	2	1	1	2	2	1	2	1	1	2	2	1	1	2
15	2	2	1	2	1	1	2	1	2	2	1	2	1	1	2
16	2	2	1	2	1	1	2	2	1	1	2	1	2	2	1

شکل ۳-۳ آرایه متعامد دو بعدی با ۱۵ ورودی

Experiment number	Column number			
	1	2	3	4
1	1	1	1	1
2	1	2	2	2
3	1	3	3	3
4	2	1	2	3
5	2	2	3	1
6	2	3	1	2
7	3	1	3	2
8	3	2	1	3
9	3	3	2	1

شکل ۳-۴ آرایه متعامد سه بعدی با ۴ ورودی

```
Function TGA(problem) returns a state that is a local optimum
Input: Populationsize, Problemsize, Pcrossover, Pmutation
Output: Sbest

Population ← InitializePopulation(Populationsize, Problemsize);
EvaluatePopulation(Population);
Sbest ← GetBestSolution(Population);
while ¬ StopCondition() do
  Parents ← SelectParents(Population, Populationsize);
  Children ← ∅;
  foreach Parent1, Parent2 ∈ Parents do
    Child1, Child2 ← Crossover(Parent1, Parent2, Pcrossover);

    Child3 ← Taguchi-Method(Child1, Child2);

    Children ← Mutate(Child3, Pmutation);
  end
  EvaluatePopulation(Children);
  Population ← Replace(Population, Children);
  Sbest ← GetBestSolution(Population);
end
return Sbest;
```

الگوریتم ۳-۹ شبه کد الگوریتم ژنتیک تاگوچی

Chromosome c_1 : 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0.

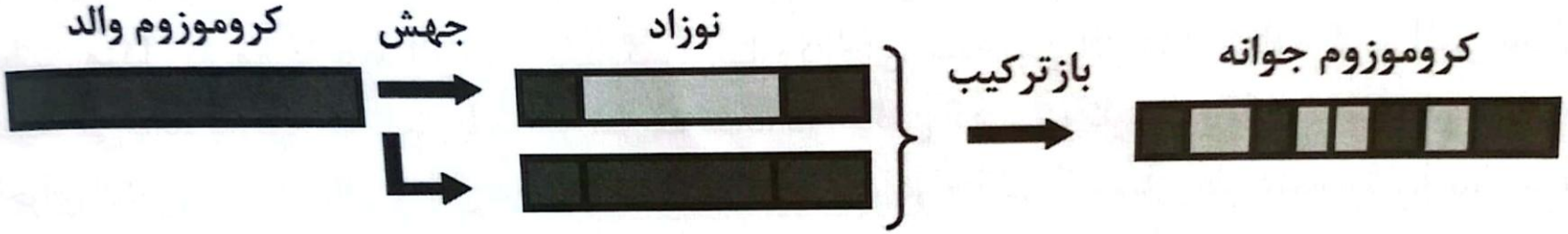
Chromosome c_2 : 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0.

Experiment Number (i)	Factors (f)							Function value	η_i
	A	B	C	D	E	F	G		
	Column number								
	1	2	3	4	5	6	7		
1	1.0	1.0	1.0	1.0	0.0	0.0	0.0	4.0	1/4.0
2	1.0	1.0	1.0	0.0	1.0	1.0	1.0	6.0	1/6.0
3	1.0	0.0	0.0	1.0	0.0	1.0	1.0	4.0	1/4.0
4	1.0	0.0	0.0	0.0	1.0	0.0	0.0	2.0	1/2.0
5	0.0	1.0	0.0	1.0	1.0	0.0	1.0	4.0	1/4.0
6	0.0	1.0	0.0	0.0	0.0	1.0	0.0	2.0	1/2.0
7	0.0	0.0	1.0	1.0	1.0	1.0	0.0	4.0	1/4.0
8	0.0	0.0	1.0	0.0	0.0	0.0	1.0	2.0	1/2.0
E_{f1}	1.167*	1.167*	1.167	1	1.5	1.5	1.5		
E_{f2}	1.5*	1.5*	1.5	1.667	1.167	1.167	1.167		
Optimal level	2*	2*	2	2	1	1	1		
Optimal chromosome	0.0	0.0	0.0	0.0	0.0	0.0	0.0		

شکل ۳-۵ چگونگی تولید فرزند سوم از دو فرزند محصول عملگر بازترکیب

بهینه سازی تولید مثل غیر جنسی
Asexual reproduction optimization(ARO)

- ابتدا یک عضو تکی (والد) به صورت تصادفی تولید می شود.
- ارزیابی می شود
- یک زیر رشته در والد به صورت تصادفی انتخاب می شود.(g) و مورد جهش قرار می گیرد. (نوزاد)
- سپس کروموزوم والد و نوزاد تحت بازترکیب یکنواخت قرار می گیرند.با احتمال Pc (جوانه)
- جوانه ارزیابی شده و در صورتیکه برازش آن از کروموزوم والد بهتر باشد جایگزین آن می شود.



شکل ۳-۷ مثالی برای تبیین عملکرد عملگرهای تولیدمثل در بهینه‌سازی تولیدمثل غیرجنسی

بهینه‌سازی تولیدمثل غیرجنسی توسط فراست[1] و همکاران [۱۵۱] پیشنهاد شده است (الگوریتم ۳-۱۳). خصوصیات مهم بهینه‌سازی تولیدمثل غیرجنسی عبارتند از:

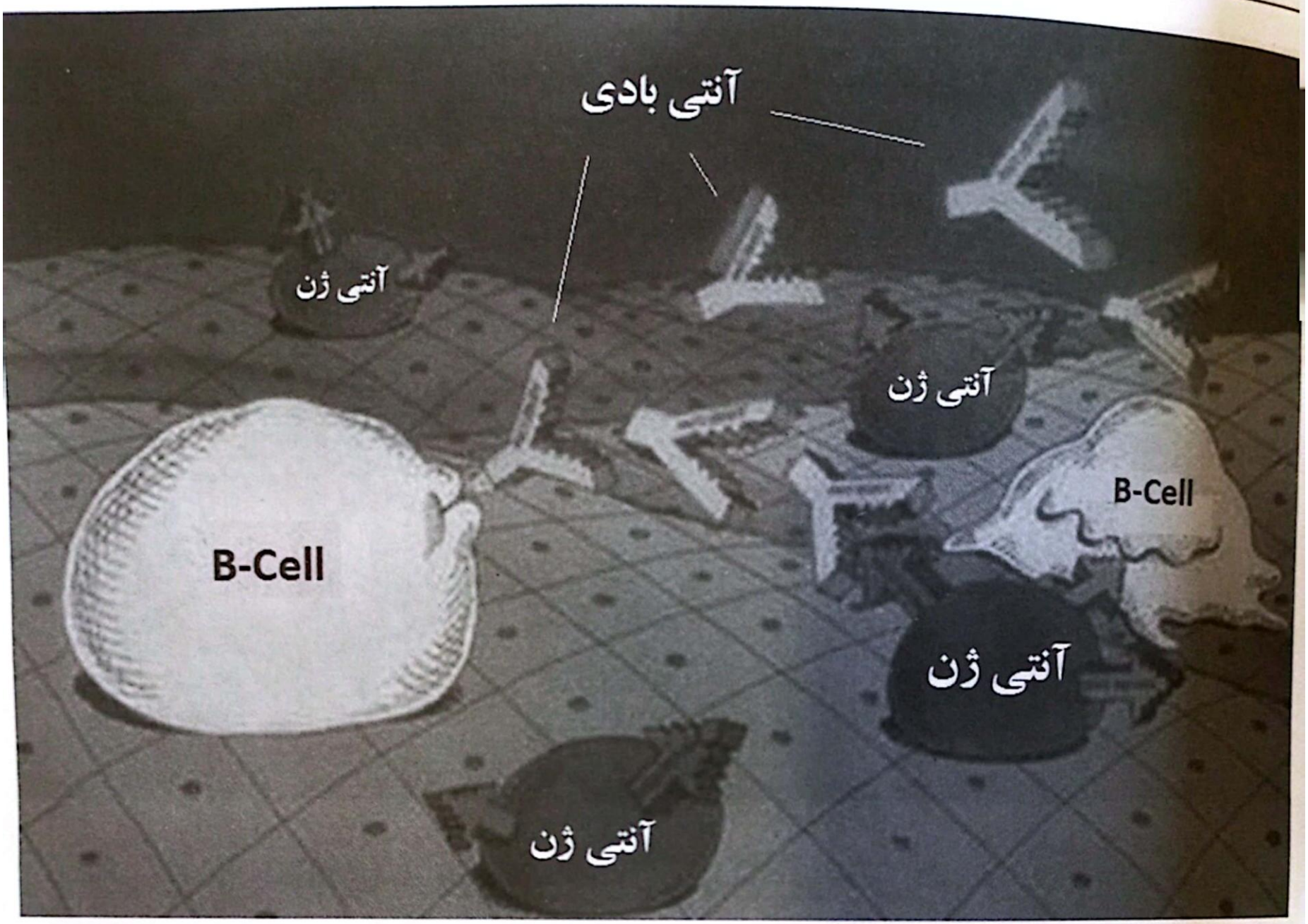
- ❖ استفاده از نمایش رشته بیتی.
- ❖ طول ثابت و یکسان برای هر کروموزوم.
- ❖ بکارگیری یک عضو تکی بجای جمعیتی از اعضاء.
- ❖ در بهینه‌سازی تولیدمثل غیرجنسی، دیدگاهی حریصانه برای حرکت در دورنمای برازش مسأله وجود دارد. با توجه به این دیدگاه، فرزند تولید شده تنها زمانی جایگزین والد خود می‌شود که از او برازنده‌تر باشد.
- ❖ بکارگیری عملگر بازترکیب یکنواخت.
- ❖ استفاده از جهش معکوس‌سازی بیت.
- ❖ تنظیم احتمال بازترکیب P_c به صورت کاملاً تصادفی.
- ❖ برای طول زیررشته تصادفی مورد جهش (g) در کروموزوم والد همواره این رابطه برقرار است: $g \sim Uniform[1,L]$.
- ❖ ویژگی‌های مثبت بهینه‌سازی تولیدمثل غیرجنسی عبارت است از: سرعت بالا و عدم وجود هر گونه پارامتر در این الگوریتم.
- ❖ ویژگی منفی بهینه‌سازی تولیدمثل غیرجنسی رفتار شدیداً تصادفی این الگوریتم است که متأسفانه ماهیت این روش را به الگوریتم جستجوی تصادفی (الگوریتم ۱-۲ در فصل اول) بسیار نزدیک کرده است.

```
Function ARO(problem) returns a state that is a local maximum
Input: Problemsize
Output: Sbest

Parent ← InitializeParent(Problemsize);
EvaluateSolution(Parent);
while ¬ StopCondition() do
    Larva ← Mutate(Parent);
    Bud ← Crossover(Parent, Larva);
    EvaluateSolution (Bud);
    if Fitness(Bud) > Fitness(Parent) then
      Parent ← Bud;
    end
end
Sbest ← Parent;
return Sbest;
```

الگوریتم ۳-۱۳ شبه‌کد بهینه‌سازی تولیدمثل غیرجنسی

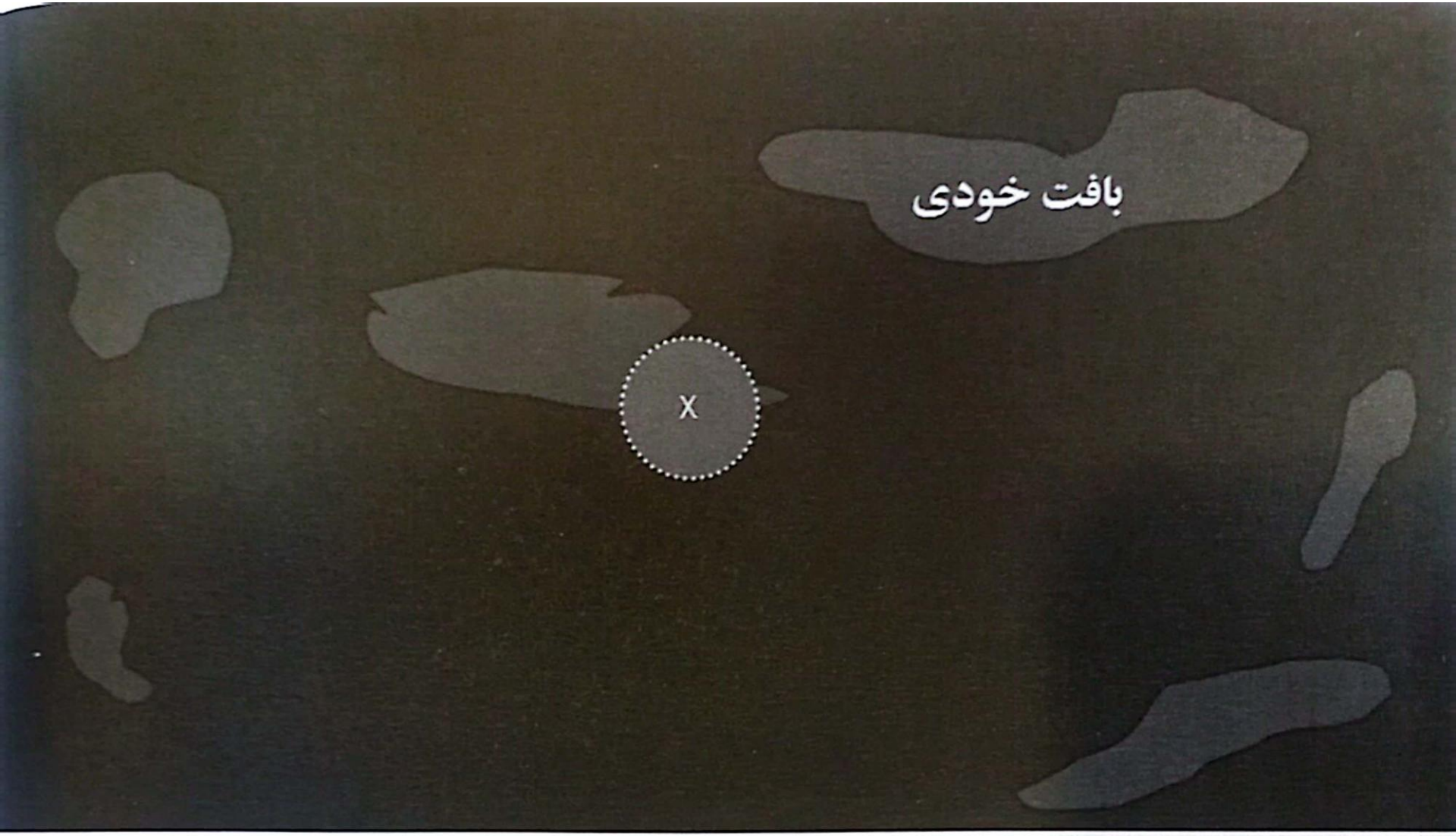
سیستم ایمنی مصنوعی

- هدف ایمنی طبیعی انسان تمایز بین بافت خودی و غیر خودی (آنتی ژن) است.
- پاسخ ایمنی بدن در مقابل آنتی ژن ، آنتی بادی است.
- تشخیص آنتی ژن به عهده گلبول سفیدی به نام لنفوسیت است.
- لنفوسیت B-cll , T-cell
- آنتی بادی ها دارای گیرنده خاصی برای آنتی ژنها هستند.و در زمان تشخیص آنتی ژنها تکثیر کلونی در B-cell اتفاق می افتد.و با T-cell تقویت می شود.
- اگر آنتی بادی بافت خودی را به عنوان آنتی ژن تشخیص دهد نابود می شود.
- در تکثیر کلونی سلول پلاسما و سلول حافظه تشکیل می شود.



شکل ۳-۸ تولید آنتی‌بادی توسط B-Cell ها برای تشخیص آنتی‌ژن‌ها

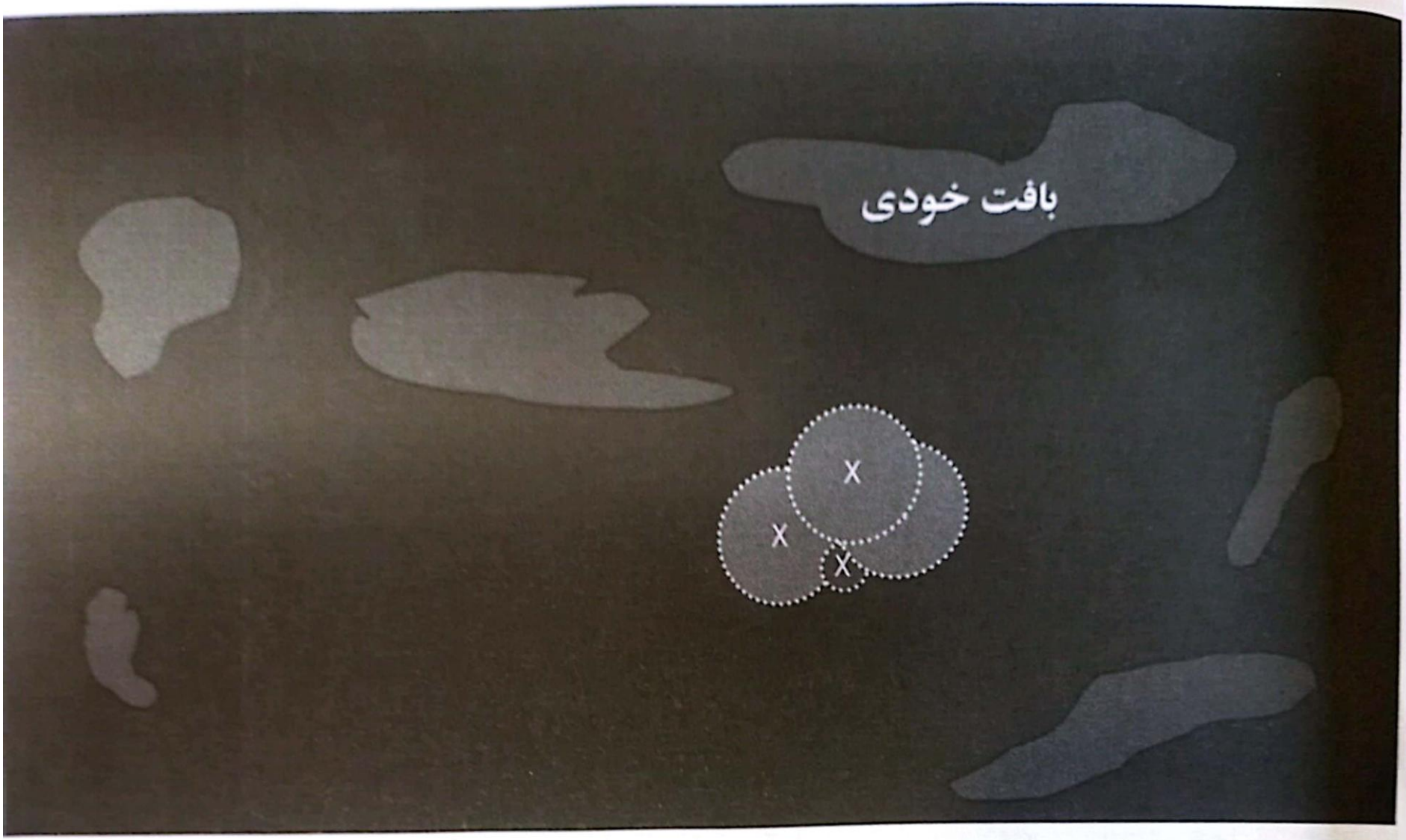
شکل ۳-۹ چگونگی تشخیص و انهدام آنتی‌ژن توسط آنتی‌بادی (جستجو برای تشخیص)



شکل ۳-۱۰ چگونگی تشخیص و انهدام آنتی‌ژن توسط آنتی‌بادی (تشخیص اشتباه بافت خودی)

شکل ۳-۱۱ چگونگی تشخیص و انهدام آنتی‌ژن توسط آنتی‌بادی (نابود شدن آنتی‌بادی غیرشایسته)

شکل ۳-۱۲ چگونگی تشخیص و انهدام آنتی‌ژن توسط آنتی‌بادی (تشخیص آنتی‌ژن)



شکل ۳-۱۳ چگونگی تشخیص و انهدام آنتی‌ژن توسط آنتی‌بادی (انهدام آنتی‌ژن با تکثیر آنتی‌بادی)

سیستم ایمنی مصنوعی

- برگرفته از فرآیند تکثیر سلولی پس از تشخیص عامل خارجی در سیستم ایمنی طبیعی می باشد.
- مرحله اول: گروهی از بهترین سلولها یعنی سلولهایی که آنتی بادی مربوط به آنها بیشترین تطابق با آنتی ژنها را داشته باشدانتخاب می شوند.
- مرحله دوم : سلولهای انتخاب شده تحت عمل کلون قرار می گیرند.و با یک نرخ تکثیر مورد تکثیر واقع می شوند.
- مرحله سوم:سلولهای تکثیر شده تحت عمل فراجهش قرار می گیرند

- استفاده از نمایش رشته بیتی.
- طول ثابت و یکسان برای هر سلول (هر عضو جمعیت).
- بکارگیری یک جمعیت با تعداد اعضاء ثابت. البته در سیستم ایمنی مصنوعی می‌توان با تعریف طول عمر برای سلول‌های جمعیت، از یک جمعیت با تعداد اعضاء متغیر نیز بهره برد.
- در الگوریتم ۳-۱۴، $Selection_{size}$ نشان‌گر تعداد سلول‌هایی است که برای انجام عملیات تکثیر، از جمعیت دور جاری در حلقه while، انتخاب می‌شوند (مرحله اول در انتخاب کلونی). بخاطر

وجود همین مرحله در سیستم ایمنی مصنوعی، ماهیتی مشابه انتخاب داروینی در این فرامکاشفه وجود دارد. به همین دلیل، در کتاب حاضر سیستم ایمنی مصنوعی در این فصل و به عنوان یک روش تکاملی مورد بحث و بررسی قرار گرفته است.

- برای تعیین عدد تکثیر سلولی برای یک سلول ایمنی انتخاب شده در تابع Clone (الگوریتم ۳-۱۴)، از رابطه $N_c = round(\beta.N.R)$ استفاده می‌شود در این رابطه، N_c بیانگر تعداد کپی‌های سلول، β مبین نرخ تکثیر ($\text{Clone}_{\text{rate}}$ در الگوریتم ۳-۱۴)، N نشان‌گر تعداد سلول‌های جمعیت ($\text{Population}_{\text{size}}$ در الگوریتم ۳-۱۴) و R بیانگر برازش مبتنی بر رتبه سلول مورد تکثیر می‌باشد.
- استفاده از جهش معکوس‌سازی بیت در عملگر فراجهش (عملگر جهش در AIS عملگر اصلی است).
- برخلاف الگوریتم‌های تکاملی که معمولاً احتمال جهش مقداری ثابت می‌باشد، احتمال فراجهش (P_{hm}) در سیستم ایمنی مصنوعی مقداری متغیر داشته و اندازه آن بستگی به برازش سلول ایمنی (عضو جمعیت) دارد. برای محاسبه احتمال فراجهش داریم: $P_{hm} = \exp(-P_m . f)$، در این رابطه P_m نرخ جهش بوده و f برازش سلول ایمنی مورد جهش می‌باشد.

```
Function AIS(problem) returns a state that is a local optimum
Input: Population_size, Selection_size, Problem_size, Clone_rate, P_mutation
Output: S_best

Population ← InitializePopulation(Population_size, Problem_size);
EvaluatePopulation(Population);
S_best ← GetBestSolution(Population);
while ¬ StopCondition() do
  Population_select ← Select(Population, Selection_size);
  Population_clones ← ∅;
  foreach p_i ∈ Population_select do
    Population_clones ← Clone(p_i, Clone_rate);
  end
  foreach p_i ∈ Population_clones do
    Population_new ← Hypermutate(p_i, P_mutation);
  end
  EvaluatePopulation(Population_new);
  Population ← Replace(Population, Population_new);
  S_best ← GetBestSolution(Population);
end
return S_best;
```

الگوریتم ۳-۱۴ شبه‌کد سیستم ایمنی مصنوعی