به نام خدا

تمرین ششم

سارا سادات یونسی - ۹۸۵۳۳۰۵۳

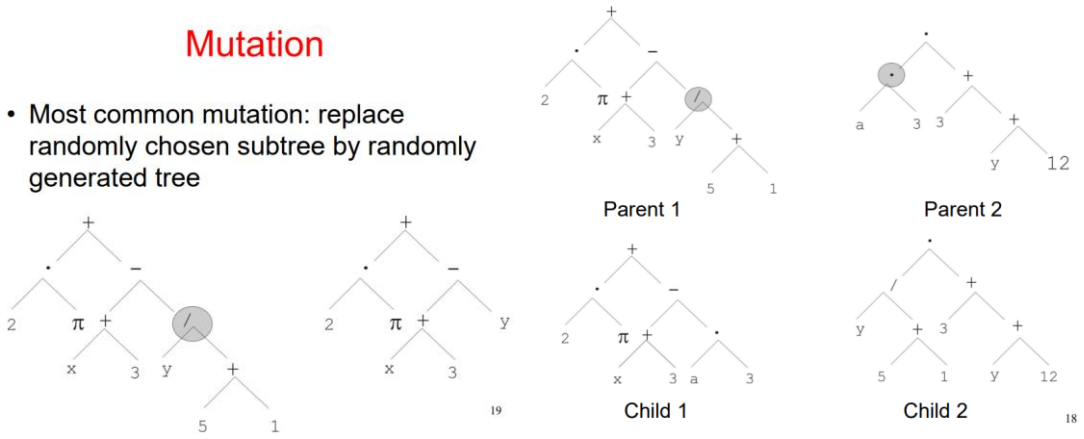
فهرست

سوال ۱

در ابتدا از کاربر مواردی مانند المنت های مورد نیاز کاربر برای مثال دو باکس اسم و دگمه تایید و رنگ مورد نظر کاربر و دیزاین مورد نظر کاربر و محدودیت های طراحی را میگیریم و بعد یک دیزاین یا چند دیزاین مورد نظر با ساختار درخت مشخص می شود که هر گره در درخت می تواند تگ html ای ما باشد.که نحوه ی قرار گیری ان ها را اینکه در چه شاخه ای قرار دارند تعیین می کند.که در چه موقعیتی این تگ ها از هم قرار دارند. یک تابع فیتنس خواهیم داشت که در نظر میگیرد چقدر انتظارات ما از طراحی براورده شده است.در selection مواردی با بالاترین فیتنس انتخاب میشود.

Reproduction : فرزندان ترکیب شده اند با یکدیگر و در crossover دو والد باهم درخت هایشان ترکیب می شود تا دو فرزند را بسازند.

Mutation هم به صورت رندم یک مولفه ی درخت را تغییر می دهیم تا بتوانیم تابع ارزیابی را به بهترین حالت خود برسانیم و نسل های بهتری را تولید کنیم. تا جایی تکرار می کنیم که نسل های خوبی تولید شود و یا به یک تعداد مشخص تکرار برسیم شکل های زیر از جزوه برداشته شده و ساختار های ترکیب و جهش را به خوبی نشان می دهد.



همچنین مراحل ژنتیک پروگرمینگ طبق جزوه :

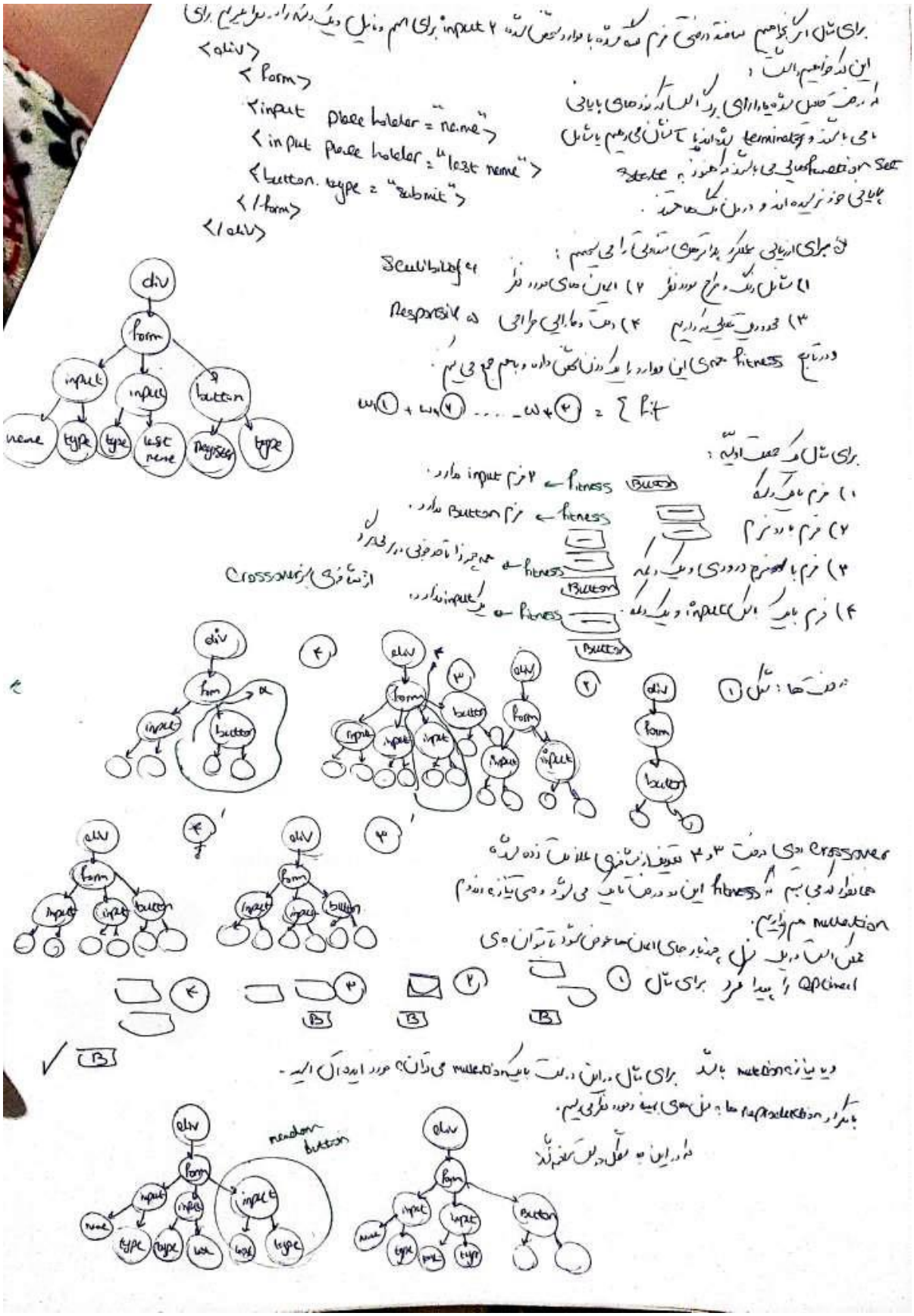
GP Process

1. Initialize population of computer programs
2. Determine fitness of each program
3. Reproduce according to fitness values and reproduction probability
4. Perform crossover of subexpressions
5. Go to step 2 unless termination conditions are met

Example application: symbolic regression

- Given some points in $\mathbf{R}^2$, $(x_1, y_1), \ldots, (x_n, y_n)$
- Find function f(x) s.t. $\forall i = 1, \ldots, n : f(x_i) = y_i$
- Possible GP solution:
  - Representation by F = {+, -, /, sin, cos}, T = $\mathbf{R} \cup \{x\}$
  - Fitness is the error $err(f) = \sum_{i=1}^{n} (f(x_i) - y_i)^2$
  - All operators standard
  - pop.size = 1000, ramped half-half initialisation
  - Termination: n "hits" or 50000 fitness evaluations reached (where "hit" is if $|f(x_i) - y_i| < 0.0001$)

نمونه ای از ساختار درخت: ساختار درختی ممکن است شامل گره‌هایی باشد که تگ‌های HTML، سبک‌های CSS و دیگر عناصر طراحی را نشان می‌دهند، همراه با عملیاتی برای دستکاری و ترکیب این عناصر برای تشکیل یک صفحه وب منسجم. برای مثال، یک ساختار درختی ساده شده ممکن است شبیه ساختار درختی HTML ارائه شده در پاسخ قبلی باشد. عملکرد تناسب اندام: تابع فیتنس ارزیابی می کند که یک صفحه وب ایجاد شده چقدر با ترجیحات طراحی کاربر مطابقت دارد. در این زمینه، تابع تناسب معیارهای طراحی مانند طرح، طرح رنگ، پاسخگویی و زیبایی شناسی کلی را در نظر می گیرد. دلیل انتخاب این تابع فیتنس ، نزدیک بودن صفحه وب تولید شده به مشخصات کاربر است. نمرات بالاتر برای دقت و کارایی در دستیابی به طرح مورد نظر تعلق می گیرد. ۲. عملکرد فیتنس تعریف: تابعی که به هر درخت یک امتیاز عددی اختصاص می‌دهد که کیفیت آن را بر اساس: دقت: HTML تولید شده چقدر با نیازهای کاربر مطابقت دارد. کارایی: کد چقدر مختصر و ساختارمند است. معیارهای اضافی: خوانایی، قابلیت نگهداری، دسترسی، پاسخگویی و غیره. نمونه عملکرد فیتنس= دقت_نمره * امتیاز_کارایی * امتیاز_معیار_اضافی



موارد گفته شده در این مقاله

sgai18garev.pdf (brighton.ac.uk)

Fitness of a design There is no objective way to describe beauty or good design in general, but we can evaluate a design according to classical aesthetic principles (clarity, orderliness and symmetry). Deciding how those features should be weighted requires drawing on fields such as psychology, cognitive science and neuroscience. 3.1 Gestaltung According to [15] the first impression of a design influences all of the following perceptions. A user first views the whole Gestalt (German for "essence or shape of an entity's complete form") of the design and then starts noticing details. The following Gestalt principles were suggested in [16]: • Objects which are similar to each other in shape, size, colour, or texture will be perceived as part of a pattern. • Lines or curves will lead the eye as a path and point attention towards breaking point like the end of the path or a crossing with another one. • Viewers will subconsciously fill blanks and perceive structures as a whole, even if they are not closed. • Simple and minimalistic designs reduce distraction. • Similar elements can create the perception of combined objects, if they are close enough to each other. The eye tends to separate objects into background and foreground and shaping both in contrast to each other. • A composition should provide order and balance, otherwise the viewer will feel unease and will not be able to decipher the content. A fitness function could provide positive scores to simple and minimalist design; to order and balance; to paths that point attention to key elements (e.g. the logo); and perhaps to similarity of objects. 3.2 Page Layout Three aspects of page layout are considered here: symmetry; the golden ratio; and im ages. 4 Symmetry. In [17] it is suggested that male users consider vertical symmetrical de signed web pages more beautiful and appealing than asymmetrical ones. Surprisingly, the assessments by female participants seem not to be influenced by the factor whether a web page is symmetrical designed or not. Altaboli and Yin [18] confirm that the more similar the numbers and sizes of objects in two neighbouring areas are, the more ap pealing the visual aesthetics are perceived. Golden Ratio. An asymmetrical design can be appealing as well, especially if it con forms to the golden ratio. The golden ratio is achieved when the ratio between two quantities is the same as the ratio between the bigger quantity and the sum of both quantities. The ratio is commonly believed to make a design more appealing to the human eye. Therefore, it is used, inter alia, in architecture, art, and music; the ancient Greek sculptor Phidias may have known of this ratio in 447 BC, when he created the sculptures for the Parthenon [19]. The golden ratio can also be found in nature, which might indicate that it offers some advantage in natural selection. Images. Images are a critical part of a website as they help visitors to connect and feel comfortable [20]. They are also much more important than text because few visitors read all the text on any website. Millennials favour websites with a large main picture; the users observed in [21] rated the visual appealing of such pages "significantly higher" compared to websites without such a main picture. Tullis and Tullis [22] showed that visual appeal ratings of e-commerce websites improved as the size of the largest image increased. So a fitness function for page layout could evaluate symmetry; use of the golden ratio; and the size of the largest image. 3.3 Colour When designing a website, it is important that the colour scheme matches the content, because it will determine in which way this content is perceived. The interpretation of colour depends on the social and cultural background of the viewer, some colours also create specific emotions: "Within the psychology of colors, warm colors show excite ment, optimism, and creativity; cool colors symbolize peace, calmness, and harmony" [23]. 5 The foundation of colour theory is the colour wheel as shown in Error! Reference source not found.. Complementary colours are colours which are on opposite sides of the wheel (e.g. yellow – violet). They create a high contrast when used in design. Anal ogous colours create a harmonious

design by combining a colour and its neighbouring colours (e.g. red - red orange and red - red violet). However, since this project aims to create a universal solution, it is not supposed to be optimised for a certain style or scheme. The nature of the colour theme is determined by the colour of the logo, which is given as an input variable. The fitness function will therefore evaluate the use of complementary colours to create a contrast between the main element and the background, and of analogous colours that harmonise with the logo

.

روند کلی و ساختار درخت در این ابزار به شرح زیر است: • این ابزار با جمعیتی از درختان تصادفی شروع می شود که نشان دهنده طرح های مختلف صفحات وب هستند .هر درخت شامل گره هایی است که با عناصر HTML مانند برچسب ها، ویژگی ها و متن مطابقت دارند .به عنوان مثال، درختی که یک صفحه وب ساده را با یک عنوان و یک پاراگراف نشان می دهد، می تواند به شکل زیر باشد:

```
<html>
<head>
<title>My Web Page</title>
<head/>
<body>
<p>Hello, world!</p>
<body/>
<html/>
```

این ابزار هر درخت را با توجه به یک تابع تناسب ارزیابی می‌کند که اندازه‌گیری می‌کند طراحی صفحه وب تا چه حد نیازها و ترجیحات کاربر را برآورده می‌کند .تابع تناسب اندام می تواند بر اساس معیارهای مختلفی مانند طرح بندی، زیبایی شناسی، عملکرد، قابلیت استفاده، دسترسی و غیره باشد. به عنوان مثال، یک تابع تناسب اندام که به صفحات وب که عنوان واضح، فونت قابل خواندن و طراحی پاسخگو دارند پاداش می دهد .مثل این

```
:def fitness(tree)
score = 0
check if the tree has a title tag #
:if tree.has_tag("title")
score += 10
check if the tree has a body tag #
:if tree.has_tag("body")
score += 10
check if the tree has a font-family attribute #
:if tree.has_attribute("font-family")
score += 10
check if the tree has a meta tag with a viewport attribute #
```

:if tree.has_tag("meta") and tree.has_attribute("viewport")
score += 10
return the score #
return score

این ابزار بهترین درختان را از بین جمعیت بر اساس ارزش تناسب اندامشان انتخاب می کند و از آنها برای ایجاد نسل جدیدی از درختان با اعمال عملیات ژنتیکی مانند متقاطع و جهش استفاده می کند .عملیات متقاطع شامل تعویض قسمت‌های مشخص شده از جفت‌های انتخاب شده (والدین) برای تولید فرزندان جدید و متفاوتی است که بخشی از نسل جدید برنامه‌ها می‌شوند .عملیات جهش شامل تغییر یک گره تصادفی در یک درخت به یک گره متفاوت است .به عنوان مثال، یک متقاطع بین دو درخت می تواند شبیه این باشد.

:Parent 1

<html>
<head>
<title>My Web Page</title>
<head/>
<body>
Hello, world!
<body/>
<html/>

:Parent 2

<html>
<head>
<title>Another Web Page</title>
<head/>
<body>
<h1>Welcome!</h1>
This is a web page.
<body/>
<html/>

:Offspring 1

<html>
<head>
<title>My Web Page</title>
<head/>
<body>
<h1>Welcome!</h1>
This is a web page.
<body/>
<html/>

:Offspring 2

<html>
<head>
<title>Another Web Page</title>
<head/>
<body>
Hello, world!
<body/>
<html/>

- The tool repeats this process for a given number of generations or until a satisfactory solution is found. The tool returns the best tree from the final generation as the output web page design.

:A possible fitness function for this tool is

```
:def fitness(tree)
    score = 0
    check if the tree has a form tag #
    :if tree.has_tag("form")
        score += 10
    check if the tree has two input tags with name and surname attributes #
    :if tree.count_tags("input", {"name": "name"}) == 1 and tree.count_tags("input", {"name": "surname"}) == 1
        score += 20
    check if the tree has a button tag with type and value attributes #
    :if tree.has_tag("button", {"type": "submit", "value": "Register"})
        score += 10
    check if the tree has a style tag with some CSS rules #
    :if tree.has_tag("style")
        score += 10
    return the score #
    return score
```

دلیل انتخاب این عملکرد تناسب اندام این است که به طرح های صفحه وب که دارای عناصر مورد نیاز برای فرم ثبت نام هستند، مانند فیلدهای ورودی، یک دکمه و برخی سبک ها، پاداش می دهد .تابع تناسب اندام نیز وزن های مختلفی را بر اساس اهمیت و سختی عناصر به آنها اختصاص می دهد .به عنوان مثال، داشتن دو فیلد ورودی با ویژگی های نام و نام خانوادگی مهمتر و دشوارتر از داشتن برچسب سبک است، بنابراین امتیاز بالاتری می گیرد .برای انجام این ابزار به صورت دستی در مثال زیر، می‌توانیم این مراحل را دنبال کنیم: • مرحله ۱: یک جمعیت تصادفی از درختان ایجاد کنید که طرح‌های صفحه وب را نشان می‌دهند .برای سادگی، می توانیم فرض کنیم که اندازه جمعیت ۴ و عمق درخت ۳ است. می توانیم از عناصر HTML زیر به عنوان گره های ممکن برای درخت ها استفاده کنیم<html> :، <head>، <title>، <style>، <body>، <form>، <input>، <button>، ، <h1>، <h2>، <h3>، ، .همچنین می‌توانیم از برخی ویژگی‌ها و متن‌ها برای گره‌ها استفاده کنیم، مانند نام، نوع، مقدار، رنگ، خانواده فونت، و غیره.

:Tree 1

```
<html>
<head>
<title>Register</title>
<head/>
<body>
<form>
<"input name="name" type="text>
<"input name="surname" type="text>
<"button type="submit" value="Register>
<form/>
<body/>
<html/>
```

:Tree 2

```
<html>
<head>
<title>Sign Up</title>
<style>
body {
;color: blue
;font-family: Arial
}
<style/>
<head/>
<body>
<h1>Welcome!</h1>
<p>Please fill in the form below.</p>
<form>
<"input name="name" type="text>
<"input name="surname" type="text>
<"button type="submit" value="Sign Up>
<form/>
<body/>
<html/>
```

:Tree 3

```
<html>
<head>
<title>Join Us</title>
<head/>
<body>
<div>
<h2>Join Us</h2>
<p>Enter your details below.</p>
```

<form>
<"input name="name" type="text>
<"input name="surname" type="text>
<"button type="submit" value="Join Us>
<form/>

<body/>
<html/>

:Tree 4
<html>
<head>
<title>Create Account</title>
<head/>
<body>

<h3>Create Account</h3>
<form>
<"input name="name" type="text>
<"input name="surname" type="text>
<"button type="submit" value="Create Account>
<form/>

<body/>
<html/>

- •مرحله ۲: هر درخت را با توجه به تابع تناسب تعریف شده در سوال ۲ ارزیابی کنید. مقادیر تناسب درختان عبارتند از:

Tree 1: 40
Tree 2: 50
Tree 3: 40
Tree 4: 40

•مرحله ۳: بهترین درختان را از بین جمعیت با توجه به ارزش تناسب آنها انتخاب کنید و با اعمال عملیات متقاطع و جهش، از آنها برای ایجاد نسل جدیدی از درختان استفاده کنید .برای سادگی، می توانیم فرض کنیم که نرخ نخبه گرایی ۰,۲۵، نرخ متقاطع ۰,۵ و نرخ جهش ۰,۱ است .یک نسل جدید درختان ممکن است به شکل زیر باشد:

Tree 1: 50 (elitist)
<html>
<head>
<title>Sign Up</title>

<style>
body {
;color: blue
;font-family: Arial
}
<style/>
<head/>
<body>
<h1>Welcome!</h1>
Please fill in the form below.
<form>
<"input name="name" type="text>
<"input name="surname" type="text>
<"button type="submit" value="Sign Up>
<form/>
<body/>
<html/>

Tree 2: 40 (crossover)
<html>
<head>
<title>Register</title>
<head/>
<body>

<h2>Join Us</h2>
Enter your details below.
<form>
<"input name="name" type="text>
<"input name="surname" type="text>
<"button type="submit" value="Join Us>
<form/>

<body/>
<html/>

Tree 3: 40 (crossover)
<html>
<head>
<title>Join Us</title>
<head/>
<body>
<form>
<"input name="name" type="text>
<"input name="surname" type="text>

button<>

- مرحله ۴: برخی از درختان نسل جدید را با چرخاندن برخی از بیت ها به طور تصادفی بر اساس میزان جهش، جهش دهید .یک نسل جهش یافته از درختان می تواند به شکل زیر باشد:

Tree 1: 50 (elitist)

```
<html>
<head>
<title>Sign Up</title>
<style>
body {
color: blue;
font-family: Arial;
}
</style>
</head>
<body>
<h1>Welcome!</h1>
<p>Please fill in the form below.</p>
<form>
<input name="name" type="text">
<input name="surname" type="text">
<button type="submit" value="Sign Up">
</form>
</body>
</html>
```

Tree 2: 40 (mutated)

```
<html>
<head>
<title>Register</title>
```

```
</head>
<body>
<div>
<h2>Join Us</h2>
<p>Enter your details below.</p>
<form>
<input name="name" type="text">
<input name="surname" type="text">
<button type="submit" value="Register">
</form>
</div>
</body>
</html>
```

Tree 3: 40 (mutated)

```
<html>
<head>
<title>Join Us</title>
</head>
<body>
<form>
<input name="name" type="text">
<input name="surname" type="text">
<button type="submit" value="Join Us">
</form>
</body>
</html>
```

Tree 4: 40 (crossover)

```
<html>
<head>
```

```
<title>Create Account</title>
</head>
<body>
<span>
<h3>Create Account</h3>
<form>
<input name="name" type="text">
<input name="surname" type="text">
<button type="submit" value="Create Account">
</form>
</span>
</body>
</html>
```

• مرحله ۵: مراحل ۲ تا ۴ را تکرار کنید تا راه حل رضایت بخش پیدا شود یا به حداکثر تعداد نسل برسد. این ابزار بهترین درخت را از نسل نهایی به عنوان طراحی صفحه وب خروجی برمی گرداند. به عنوان مثال، پس از ۱۰ نسل، ابزار می تواند درخت زیر را به عنوان بهترین راه حل بازگرداند

```
<html>
<head>
<title>Sign Up</title>
<style>
body {
color: blue;
font-family: Arial;
}
form {
margin: 10px;
padding: 10px;
border: 1px solid black;
}
```

```
input {
display: block;
margin: 5px;
}
button {
display: block;
margin: 5px;
background-color: green;
color: white;
}
</style>
</head>
<body>
<h1>Welcome!</h1>
<p>Please fill in the form below.</p>
<form>
<input name="name" type="text" placeholder="Name">
<input name="surname" type="text" placeholder="Surname">
<button type="submit" value="Sign Up">Sign Up</button>
</form>
</body>
</html>
```

این درخت دارای مقدار تناسب ۶۰ است که حداکثر امتیاز ممکن برای تابع تناسب تعریف شده در سوال ۲ است. این طراحی صفحه وب دارای تمام عناصر مورد نیاز برای یک فرم ثبت نام، مانند فیلدهای ورودی، یک دکمه، و مقداری استایل است. همچنین دارای برخی ویژگی‌های اضافی مانند مکان‌ها، حاشیه‌ها، لایه‌ها، حاشیه‌ها و رنگ‌های پس‌زمینه است. این طراحی صفحه وب به خوبی نیازها و ترجیحات کاربر را برآورده می کند.

پاسخ این سوال را در قسمت اول با چت بات و در قسمت های بعد با مقاله و در اخر به صورت دستی حل کردم.

سوال ۲

مراحل الگوریتم :

۱. ایجاد جمعیت :
۲. حساب کردن fitness هر کروموزم در جمعیت
۳. ساختن یک pool
۴. تولید فرزندان به روش crossover
۵. تکرار مراحل بالا در نسل بعدی

•ساختار کروموزوم :

یک رشته باینری به طول ۹ بیت، بیت اول بیت علامت، Code Grayدر واقع از Code Gray Signed استفاده کردیم.

sign	۴ بیت صحیح	۴ بیت اعشار

•محاسبهfitness

ابتدا مقدار واقعی کرومزوم را به د ست میآوریم. آن را در معادله میگذاریم. مقدار fitness را به صورت زیر حساب میکنیم.

$$fitness(x) = \frac{1}{|f(x)| + 1}$$

•حفظ نخبگان نسل :در هر نسل ۲۰ درصد افراد که بیشترین fitness را دارند به صورت کامل در نسل بعدی وجود خواهند داشت .

•ایجاد :pool با استفاده از روش wheel roulette یک pool ایجاد میکنیم سپس روی آن crossover و mutationانجام میدهیم .

•نحوه :crossover از روش point one استفاده میکنیم. یعنی یک خط در نظر میگیریم قسمت سمت چپ از والد اول و قسمت سمت راست از والد ۲ گرفته میشود.

• نحوه :mutation با توجه به نرخ جهش بیت ها میتوانند Complement شوند.

توضیحات کد :

- **POPULATION_SIZE: تعداد جمعیت اولیه که شامل تعدادی کروموزوم (حلهای ممکن) است.**
- **ELITIST_RATE: نرخ نخبگی که نشان میدهد چه تعداد از بهترین کروموزومها در هر نسل حفظ میشوند.**
- **GENERATIONS: تعداد نسلهایی که الگوریتم اجرا میشود.**
- **MUTATION_RATE: نرخ جهش که نشان میدهد چه تعداد از کروموزومها در هر نسل تغییرات تصادفی میکنند.**
- **FUNC: تابع چندجملهای که ریشههای آن را میخواهیم پیدا کنیم. در این کد، تابع $$۱۸۶ - x^3 - 7.22x^2 + 15.5x 13.2$$ است. وقتی می خواهیم توابع مختلف را تست کنیم در این قیمت میگذاریم.**
- **model: یک شیء از کلاس Root_GA که الگوریتم ژنتیک را پیادهسازی میکند.**
- **model.set_function: یک متد از کلاس Root_GA که تابع مورد نظر، محدودهی جستجو و دقت را تنظیم میکند. در این کد، محدودهی جستجو بین -۹ و ۹ و دقت ۹ رقم اعشار است.**

```
POPULATION_SIZE = 100
ELITIST_RATE = 0.2
GENERATIONS = 10
MUTATION_RATE = 0.1
FUNC = lambda x: 186 * (x)**3 -7.22 *(x)**2 +15.5*(x)-13.2
# 4*(x)**3 -5*(x)**2 + (x)-1
#  (x)**2 -8 *(x)+4
# 2*(x)-4
model = Root_GA(GENERATIONS, POPULATION_SIZE, ELITIST_RATE, MUTATION_RATE)
model.set_function(FUNC, -9, 9, 9)
```

• تابع digit_to_gray یک رقم اعشاری را به لیستی از چهار بیت در کد خاکستری تبدیل می کند. کد خاکستری راهی برای نمایش اعداد است که در هر مرحله فقط یک بیت تغییر می کند. این روش برای جلوگیری از خطاهای جهش در الگوریتم ژنتیک مفید است.

• تابع to_gray یک عدد اعشاری را به لیستی از نه بیت در کد خاکستری تبدیل می کند. این تابع ابتدا علامت عدد را به صورت یک بیت ذخیره می کند. سپس با استفاده از تابع digit_to_gray قسمت های صحیح و کسری عدد را تبدیل کرده و به لیست اضافه می کند.

• تابع gray_to_digit لیستی از چهار بیت در کد خاکستری را به یک رقم اعشاری تبدیل می کند. این تابع برعکس تابع digit_to_gray را انجام می دهد.

• تابع to_float لیستی از نه بیت در کد خاکستری را به یک عدد اعشاری تبدیل می کند. این تابع برعکس تابع to_gray را انجام می دهد. این تابع ابتدا علامت عدد را از بیت اول مشخص می کند. سپس اجزای صحیح و کسری عدد را با استفاده از تابع gray_to_digit تبدیل کرده و آنها را به صورت رشته ای به هم متصل می کند. در نهایت رشته را به عدد اعشاری تبدیل می کند.

• کلاس Root_GA یک شی از الگوریتم ژنتیک برای یافتن ریشه های یک تابع چند جمله ای ایجاد می کند. این کلاس شامل متغیرها و متدهای زیر است:

• جمعیت متغیر فهرستی از کروموزوم ها (راه حل های ممکن) است که در هر نسل تولید می شود. هر کروموزوم لیستی از نه بیت در کد خاکستری است که یک عدد اعشاری را نشان می دهد.

• متغیر جمعیت_size تعداد جمعیت اولیه را مشخص می کند. این مقدار به عنوان پارامتر ورودی هنگام ایجاد یک شی از کلاس Root_GA داده می شود.

• متغیر elitist_rate میزان نخبه گرایی را مشخص می کند. این مقدار نشان می دهد که در هر نسل چه تعداد از بهترین کروموزوم ها حفظ شده و به نسل بعدی منتقل می شوند. این مقدار به عنوان پارامتر ورودی هنگام ایجاد یک شی از کلاس Root_GA داده می شود.

• متغیر نسل ها تعداد نسل هایی را که الگوریتم اجرا می کند مشخص می کند. این مقدار به عنوان پارامتر ورودی هنگام ایجاد یک شی از کلاس Root_GA داده می شود.

• متغیر mutation_rate میزان جهش را مشخص می کند. این مقدار نشان می دهد که چه تعداد از کروموزوم ها به طور تصادفی در هر نسل تغییر می کنند. این مقدار به عنوان پارامتر ورودی هنگام ایجاد یک شی از کلاس Root_GA داده می شود.

• متد set_function تابع چند جمله ای را تنظیم می کند که می خواهیم ریشه های آن، محدوده جستجو و طول کروموزوم را پیدا کنیم. این متد هنگام ایجاد یک شی از کلاس Root_GA فراخوانی می شود. این روش شامل پارامترهای زیر است:

• پارامتر func تابع چند جمله ای را به عنوان تابع لامبدا می گیرد. برای مثال، تابع $۱۸۶ + x^3 - 7.22x^2 - 15.5x - 13.2$$ را می توان به صورت لامبدا x نوشت: ۱۸۶ * (x)**3 - 7.22 * (x)**2 + 15.5 * (x - 13.2).

• پارامتر min_root حداقل مقدار ممکن را برای ریشه ها مشخص می کند. برای مثال، اگر محدوده جستجو بین -۹ و ۹ باشد، این پارامتر برابر با -۹ است.

• پارامتر max_root حداکثر مقدار ممکن را برای ریشه ها مشخص می کند. به عنوان مثال، اگر محدوده جستجو بین -۹ و ۹ باشد، این پارامتر برابر با ۹ است.

• پارامتر chromosome_length طول کروموزوم را مشخص می کند. این مقدار برابر با تعداد بیت های کد خاکستری است که یک عدد اعشاری را نشان می دهد. در این کد روی نه ثابت شده است.

• روش create_chromosome با تولید یک عدد اعشاری تصادفی بین -۹ و ۹ و تبدیل آن به کد خاکستری با استفاده از تابع to_gray، کروموزوم ایجاد می کند. • متد create_population با فراخوانی متد create_chromosome برای زمان جمعیت_size و افزودن کروموزوم ها به جمعیت لیست، جمعیت اولیه را ایجاد می کند. • روش فیتنس ارزش تناسب یک کروموزوم را محاسبه می کند. ارزش تناسب معیاری است که نشان می دهد کروموزوم چقدر به ریشه تابع نزدیک است. مقدار تناسب با تبدیل کروموزوم به عدد اعشاری با استفاده از تابع to_float، ارزیابی تابع تابع در آن عدد و گرفتن معکوس قدر مطلق به اضافه یک محاسبه می‌شود. هر چه مقدار تناسب بیشتر باشد، کروموزوم به ریشه نزدیکتر است.

• روش ranked_population فهرست مرتب شده ای از کروموزوم های جمعیت را بر اساس مقادیر تناسب آنها به ترتیب نزولی برمی گرداند. • روش create_pool مجموعه ای از کروموزوم ها را ایجاد می کند که برای نسل بعدی انتخاب می شوند. اندازه استخر برابر با اندازه جمعیت است. استخر از دو بخش تشکیل شده است: بخش نخبه و بخش باقی مانده. قسمت الیتیست بهترین کروموزوم ها را با توجه به میزان الیتیسم دارد. قسمت باقیمانده شامل کروموزوم هایی است که به طور تصادفی بر اساس مقادیر تناسب آنها انتخاب می شوند. هر چه ارزش تناسب اندام بالاتر باشد، شانس انتخاب شدن بیشتر است

•روش متقاطع یک کروموزوم فرزند را با ترکیب دو کروموزوم والد ایجاد می کند .کروموزوم فرزند از دو قسمت تشکیل شده است: قسمت اول از کروموزوم والد اول تا نقطه متقاطع تصادفی کپی می شود و قسمت دوم از کروموزوم والد دوم از نقطه متقاطع تا انتها کپی می شود.

• روش جهش یک کروموزوم جهش یافته را با چرخاندن برخی از بیت های کروموزوم به طور تصادفی با توجه به نرخ جهش ایجاد می کند .هر چه میزان جهش بالاتر باشد، احتمال تغییر بیت بیشتر است .

•روش crossover_pool با اعمال عملیات متقاطع در مجموعه کروموزوم ها، جمعیت جدیدی ایجاد می کند . اندازه جمعیت جدید برابر با اندازه استخر است .جمعیت جدید از دو بخش تشکیل شده است: بخش نخبه گرا و بخش باقی مانده .قسمت الیتیست بهترین کروموزوم ها را با توجه به میزان الیتیسم دارد .قسمت باقیمانده شامل کروموزوم های فرزند است که با انتخاب تصادفی دو کروموزوم والد از استخر و اعمال روش متقاطع بر روی آنها ایجاد می شود.

• روش mutate_pool یک جمعیت جهش یافته را با اعمال عملیات جهش در مجموعه کروموزوم ها ایجاد می کند .اندازه جمعیت جهش یافته برابر با اندازه استخر است .جمعیت جهش یافته شامل کروموزوم های جهش یافته است که با اعمال روش جهش یافته برای هر کروموزوم موجود در استخر ایجاد می شود.

• روش set_population جمعیت را روی یک جمعیت جدید تنظیم می کند .این روش برای به روز رسانی جمعیت پس از هر نسل استفاده می شود .

•روش next_generation با اعمال مراحل الگوریتم ژنتیک، نسل بعدی جمعیت را ایجاد می کند .این روش ابتدا جمعیت را بر اساس ارزش تناسب اندام آنها رتبه بندی می کند .سپس با فراخوانی متد create_pool مجموعه ای از کروموزوم ها را ایجاد می کند .سپس با فراخوانی متد crossover_pool یک جمعیت جدید ایجاد می کند .سپس با فراخوانی متد mutate_pool یک جمعیت جهش یافته ایجاد می کند .سپس با فراخوانی متدset_population ، جمعیت را روی جمعیت جهش یافته تنظیم می کند .این روش بهترین کروموزوم و میانگین ارزش تناسب جمعیت را قبل از ایجاد نسل بعدی برمی گرداند .

•روش اجرا الگوریتم ژنتیک را برای تعداد معینی از نسل ها اجرا می کند .این متد ابتدا جمعیت اولیه را با فراخوانی متد create_population ایجاد می کند .سپس فهرستی از بهترین کروموزوم ها و فهرستی از مقادیر متوسط تناسب اندام برای هر نسل ایجاد می کند .سپس بر روی تعداد نسل‌ها تکرار می‌شود و روش next_generationرا برای هر نسل فراخوانی می‌کند .بهترین کروموزوم و میانگین ارزش تناسب اندام را به لیست های مربوطه اضافه می کند .همچنین گزارش هر نسل را با فراخوانی متد print_log چاپ می کند .پس

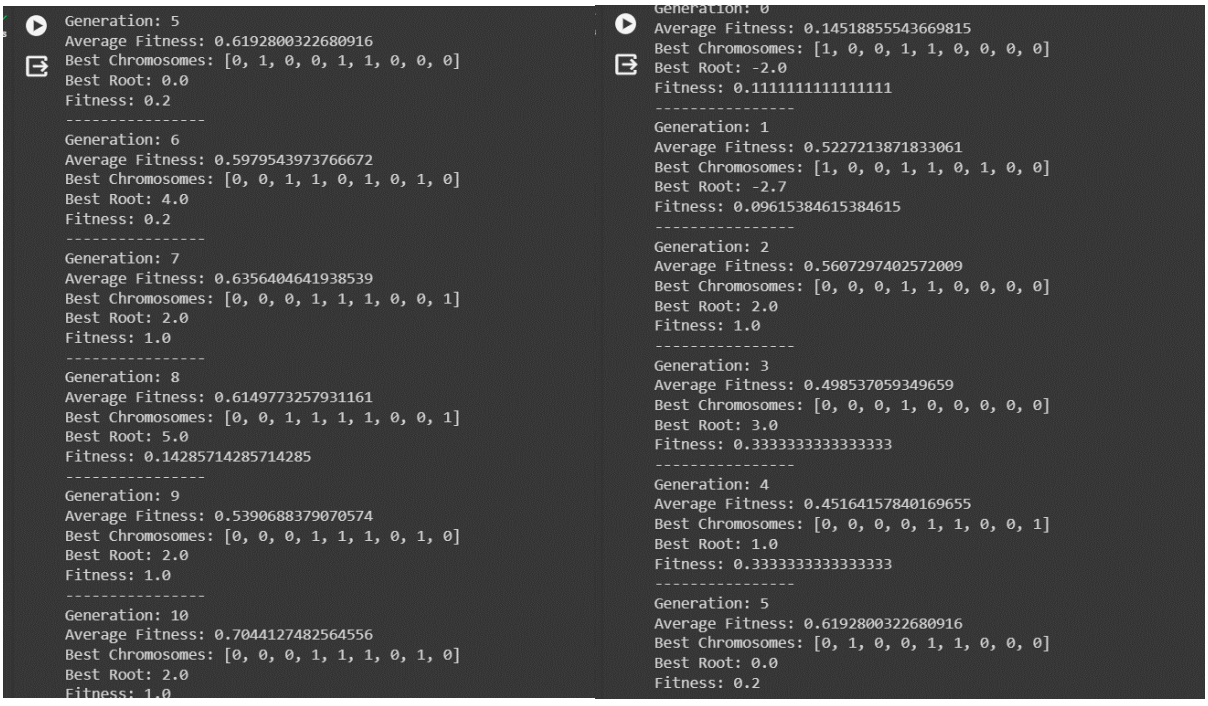
از آخرین نسل، جمعیت را بر اساس ارزش تناسب اندام آنها رتبه بندی می کند .بهترین کروموزوم و میانگین ارزش تناسب اندام را به لیست های مربوطه اضافه می کند .همچنین با فراخوانی متدprint_log ، لاگ آخرین نسل را چاپ می کند .این روش لیستی از بهترین کروموزوم ها و لیست مقادیر متوسط تناسب اندام را برای همه نسل ها برمی گرداند.

• روش print_log گزارش یک نسل را چاپ می کند .این روش تعداد تولید، میانگین ارزش تناسب، بهترین کروموزوم، بهترین ریشه و بهترین ارزش تناسب را به عنوان پارامترهای ورودی می گیرد .آنها را به صورت فرمت شده چاپ می کند .از این روش برای نشان دادن پیشرفت الگوریتم ژنتیک استفاده می شود.

برای معادله اول

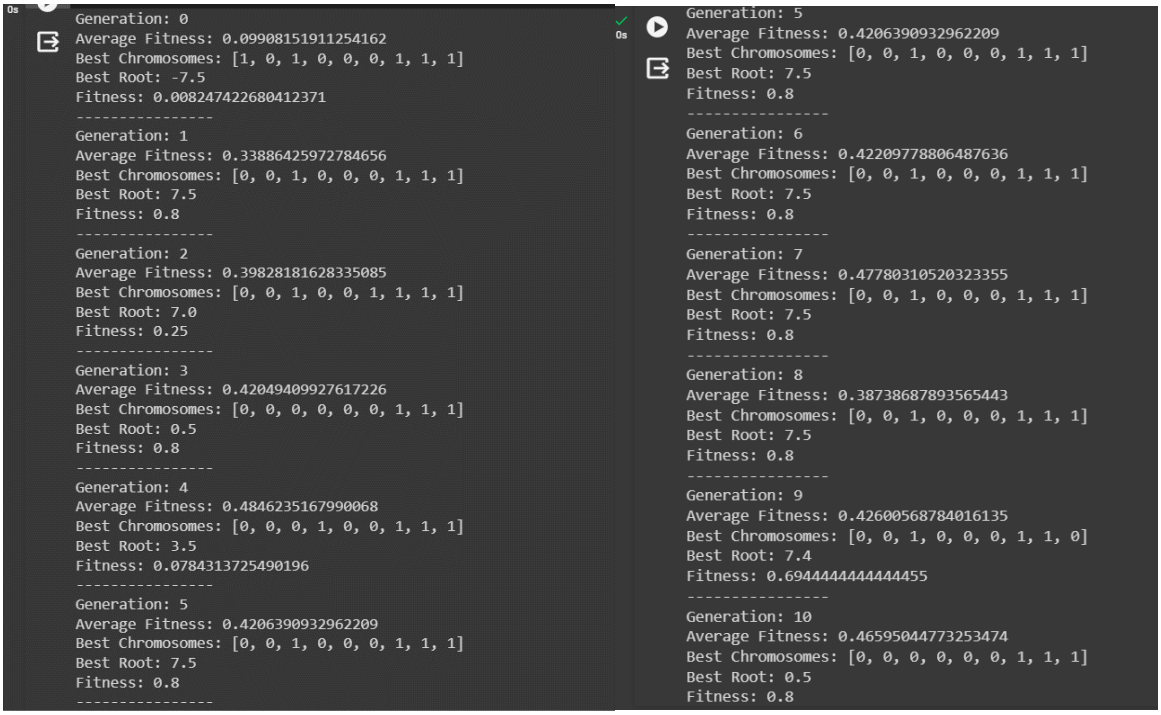
در هر نسل مقدار فیتنس بهترین کروموزوم بهترین ریشه را میبینیم در هر معادله نسلی که نزدیک ترین مقدار فیتنس به ۱ را دارد بهترین جواب را باز می گرداند.

برای مثال همانطور که واضح است جواب x=2 برمیگردد.برای این معادله



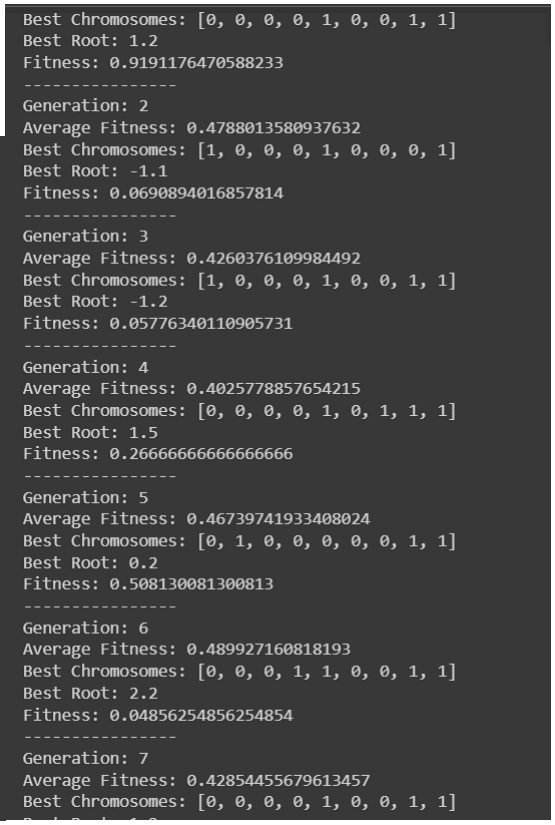
معادله دوم

جواب را تایک رقم اعشار به خوبی محاسبه کرده.



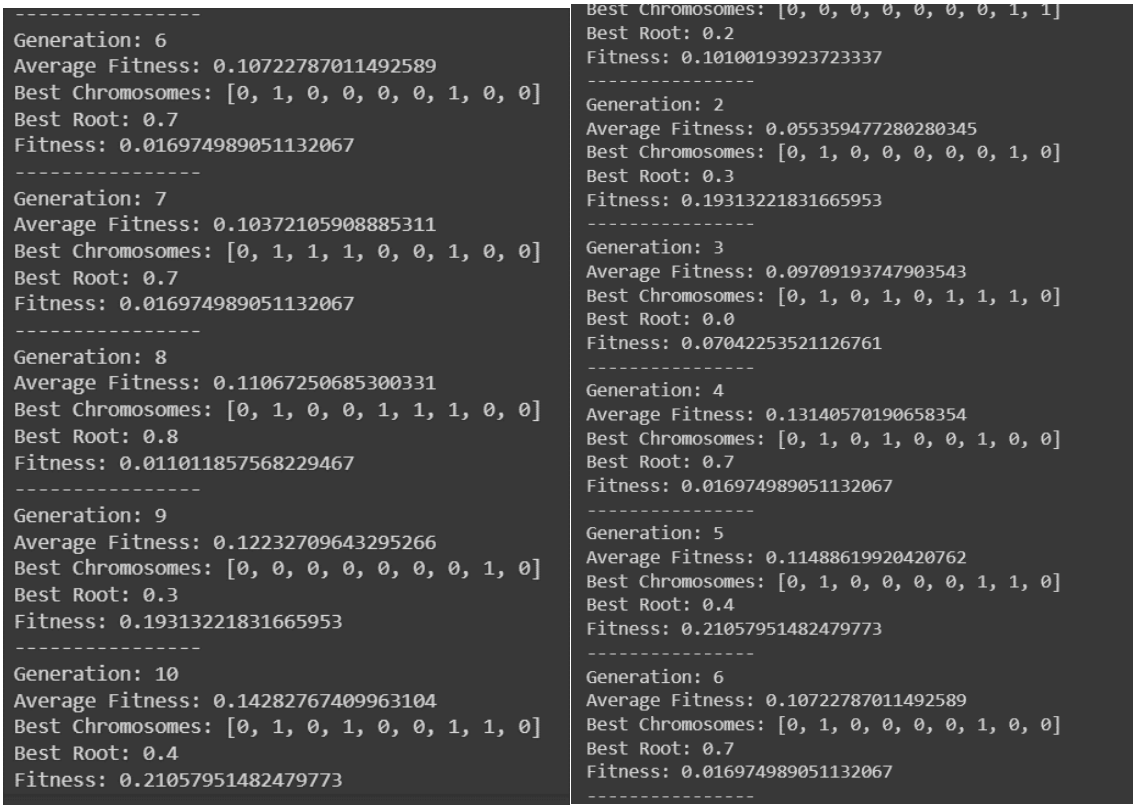
معادله سوم

جواب را تایک رقم اعشار به خوبی محاسبه کرده.



معادله چهارم

جواب را تایک رقم اعشار به خوبی محاسبه کرده.



سوال ۳

ابتدا مراحل الگوریتم ژنتیک را بررسی می کنیم : (موارد گفته شده را طبق جزوه نوشتم)

۱. ابتدا باید پارامتر های خود را رمزگذاری کنیم

Often encoded as binary strings
•Any finite alphabet can be used •
Typically, population member string is of fixed length

ساختار ژنوم: یک راه ممکن برای نشان دادن یک ژنوم برای این مشکل استفاده از جایگشت اعداد ۱ تا ۳۶ است که با ترتیب پر کردن مربع از چپ به راست و بالا به پایین مطابقت دارد .به عنوان مثال، ژنوم [۱، ۲، ۳، ٤، ٥، ٦، ۷، ۸، ۹، ۱۰، ...، ۳٦] نشان دهنده مربع است: 1 2 3 4 5 6 7 8 9 10 11 12 13

14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36

۲. مرحله تولید جمعیت

اندازه جمعیت بزرگتر ممکن است تنوع و اکتشاف فضای جستجو را افزایش دهد، .اندازه جمعیت: این تعداد ژنوم در هر نسل است مقدار ممکن برای این پارامتر ۱۰۰ است .اما هزینه محاسباتی را نیز افزایش دهد

Reproduction is used to form new population of n individuals Select members of current population Use stochastic process based on fitnesse

•Start with moderate sized population

•50-500 is often a good starting place for a GA

۳. ارزیابی برازندگی

1. “Bunching” of fitness values near top of scale, thereby lowering fitness differentials

2. Fitness values often are nearly constant

باید با توجه به سیاست و تابع انتخاب شده ان را ماکس یا صفر کنیم

۴. مرحله , selection

•Reproduction often done by normalizing fitnesses and generating n random numbers between 0 and 1

•Baker’s method: Use roulette wheel with n pointers spaced 1/n apart; use normalized fitness; spin wheel once.

•Generation gap approach: Replace x percent that have worst fitness values (x is defined as the generation gap) use with large populations

• “Elitist strategy” ensures that individual with highest fitness is copied into next generation (most GAs use this)

نرخ نخبه گرایی: این نسبت بهترین ژنوم ها از نسل فعلی است که بدون هیچ تغییری به نسل بعدی کپی می شود .این تضمین می کند که کیفیت جمعیت در طول زمان کاهش نمی یابد . مقدار ممکن برای این پارامتر ۰,۱ است

•روش انتخاب: این روش انتخاب ژنوم از جمعیت است که برای ایجاد نسل بعدی استفاده می شود .یک روش رایج انتخاب، انتخاب مسابقات است که به صورت تصادفی زیرمجموعه ای از ژنوم ها را از بین جمعیت انتخاب می کند و بهترین را از بین آنها انتخاب می کند .اندازه زیر مجموعه را اندازه مسابقات می گویند .یک مقدار ممکن برای این پارامتر ۲ است

۵. Reproduction

• “Tournament selection” All methods above rely on global population statistics – Could be a bottleneck esp. on parallel machines – Relies on presence of external fitness function which might not exist: e.g. evolving game players

• Informal Procedure: – Pick k members at random then select the best of these – Repeat to select more individuals

۶. Crossover

•Two-point crossover, with p(c) of 60-80% is common

•Often start with relatively high crossover rate, and reduce it during the run

Crossover combines inversion and recombination:

Parent1 (3 5 7 2 1 6 4 8)
Parent2 (2 5 7 6 8 1 3 4)

Child (5 8 7 2 1 6 3 4)

(1) Copy a randomly selected portion of Parent1 to Child
(2) Fill the blanks in Child with those numbers in Parent2 from left to right, as long as there are no duplication in Child.

This operator is called the *Order1* crossover. 66

نرخ متقاطع بالاتر ممکن است کاوش در فضای .نرخ متقاطع: این احتمال به کارگیری روش متقاطع برای یک جفت ژنوم است روش • .مقدار ممکن برای این پارامتر ۰,۸ است .جستجو را افزایش دهد، اما همچنین راه حل های خوب موجود را مختل کند یک روش جهش رایج برای ژنوم های جایگشتی، جهش مبادله ای است .جهش: این روش اصلاح ژنوم برای تولید ژنوم جدید است به عنوان مثال، اعمال جهش مبادله در ژنوم [۱، ۲، ۳، ٤، ٥، ٦] می .که موقعیت های دو ژن تصادفی را در ژنوم عوض می کند نرخ جهش بالاتر .نرخ جهش: این احتمال به کارگیری روش جهش در ژنوم است • .[، ۲، ٦، ٤، ٥، 13] تواند فرزندان را تولید کند مقدار ممکن برای .ممکن است تنوع و اکتشاف فضای جستجو را افزایش دهد، اما همچنین راه حل های خوب موجود را مختل کند این پارامتر ۰,۱ است

Mutation .۷

•Stochastically flipping bits often with p(m) ~ .001
•If real-valued parameters used, mutation can assign any value in parameters allowed range
•Probability of mutation usually held constant or increased during run •Can increase mutation rate when fitness variablility drops below some threshold

Mutation involves swapping two numbers of the list:

* *

Before: (5 8 7 2 1 6 3 4)

After: (5 8 6 2 1 7 3 4)

نرخ جهش بالاتر ممکن است تنوع و اکتشاف فضای جستجو را .نرخ جهش: این احتمال به کارگیری روش جهش در ژنوم است .مقدار ممکن برای این پارامتر ۰,۱ است .افزایش دهد، اما همچنین راه حل های خوب موجود را مختل کند

Terminate .۸

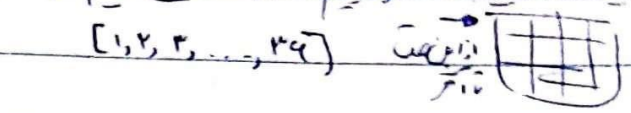
یک شرط پایان متداول این است که حداکثر تعداد .شرط پایان: این معیاری است که تعیین می کند چه زمانی الگوریتم متوقف شود .یک مقدار ممکن برای این پارامتر ۱۰۰۰ نسل یا مقدار تناسب ۱۲ است .نسل یا مقدار تناسب هدف را مشخص کنید

٩. Selecting the Number of Generations

- Often a trial and error process
- Optimum value (if known) a function of the problem
- Multiple runs often done; overall best solution selected
- To achieve the desired results, 500 to 5000 generations are required

برای این مسئله از ...

از یک تا ۳۶ دیگ آزمایشی یک عدد تصادفی برمی‌گردانیم. از روی همان عدد برنج می‌ریزیم سپس داخل آن را به مزارع می‌دهیم:



تابع fitness:

تابع‌های برازندگی توسط algorithm یا مسئله می‌توانند تابع ارزیابی برای این الگوریتم باشند. هر کروموزوم چه‌قدر خوب یا بد است در اینجا برای ما تعداد برخورد ملکه‌ها در هر کروموزوم یا همان تعداد زوج ملکه‌های همین می‌شود.

۱) $fitness = \frac{1}{\text{مجموع تعداد برخورد ملکه‌ها}} = \frac{1}{\text{مجموع تعداد زوج ملکه‌های across}}$

۲) fitness = ۱۲ − (تعداد ملکه‌ها در هر سطر از تعداد زوج ملکه‌ها در آن سطر + تعداد ملکه‌ها در هر ستون از تعداد زوج ملکه‌ها در آن ستون)

fitness score هر جدول را برای هر کروموزوم پیدا می‌کنیم. سپس جدول‌ها را بر اساس fitness score به صورت نزولی مرتب می‌کنیم و جدول‌هایی که بیشترین fitness score را دارند انتخاب می‌کنیم.

می‌توان این تابع را طوری تعریف کرد که مقدار بیشتر یا کمتر، بهتر یا بدتر باشد.

پارامترها:

۱) size: تعداد کروموزوم‌های اولیه جدول

۲) Selection: روش چرخ رولت

۳) Crossover: با احتمال ۸۰٪ تا ۹۰٪ دو کروموزوم انتخاب شده crossover point انجام می‌دهیم و فرزندان را به جمعیت اضافه می‌کنیم.

۴) Mutation: یک عدد را با احتمال کم انتخاب می‌کنیم و در دو سطر با هم دیگر swap می‌کنند. مثل crossover و Mutation مربوط به الگوریتم ژنتیک است.

با احتمال ۱٪ این عمل را انجام می‌دهیم. جدول‌ها را بر اساس fitness بهتری برمی‌گردانیم.

۵) تعداد نسل‌ها را با تعداد iteration برابر می‌کنیم. دقت کنید که ... مثال را انجام دهیم.

مراحل:

۱) یک کروموزوم‌های ۳۵ تایی از اعداد یک تا ۳۵ می‌سازیم ۲- تابع fitness را حساب می‌کنیم ۳- از روی تابع fitness بالاترها را انتخاب می‌کنیم

۴- Crossover می‌کنیم و فرزندان جدیدی می‌سازیم ۵- جهش تصادفی انجام می‌دهیم

مراحل ۲-۵ را آنقدر ادامه می‌دهیم تا به جواب برسیم یا تعداد تکرار کافی شود

در مرحله‌ی نهایی: بهترین راه حل بدست آمده یعنی بالاترین تابع ارزیابی را که به بهترین تابع fitness رسیده است را انتخاب می‌کنیم.

		fitness
کروموزوم ۱)	[۱, ۲, ... , ۳۵, ۳۶]	۰/۵
کروموزوم ۲)	[۳۱, ۳۵, ۲۳, ... , ۷, ۸]	۰/۷
کروموزوم ۳)	[۱۹, ۱۶, ... , ۲۷, ۳۲]	۰/۳
کروموزوم ۴)	[۱۵, ۱۴, ... ۳۶]	۰/۱

انتخاب کروموزوم ۱ و ۲ بالاترین fitness

Percent 1 ۱, ۲, ۳, ۴, ۵, ۶, ۷, ۸ ... ۳۶

Percent 2 ۳۱, ۳۵, ۲۳, ۲۲, ۲۱, ... , ۶, ۷, ۸

Child لـ با توجه به هر دو ۳۵ اعداد را به طور تصادفی انتخاب می‌کنیم.

mutation

Child: ۳۱, ۳۵, ۱۷, ۹, ۱۵, ... , ۳۶, ۳۵

۳۱, ۳۵, ۳۶, ۹, ۱۵, ... , ۷, ۳۶

یک رویکرد درون مقاله

عملکرد تناسب امتیاز کل تخلف را به حداقل برسانید :برای هر سطر و ستون، تفاوت بین تعداد اعداد فرد و زوج را بشمارید .مقادیر مطلق این تفاوت ها را اضافه کنید تا امتیاز تخلف را بدست آورید .یک راه حل کامل با شانس و زوج مساوی در هر سطر و ستون نمره ۰ خواهد داشت.

مولفه های :اندازه جمعیت: ۱۰۰ نفر (جایگشت)

نوع انتخاب: انتخاب تورنمنت با) k=2 انتخاب ۲ نفر به صورت تصادفی، برترین ها را انتخاب کنید(

احتمال متقاطع: ۰,۸

احتمال جهش: ۰,۰۵

نخبه گرایی: ۱۰ درصد از افراد برتر هر نسل را حفظ کنید

معیارهای توقف :حداکثر نسل: ۲۰۰ بدون بهبود در تناسب اندام برای ۵۰ نسل الگوریتم :جمعیت را با جایگشت های تصادفی ۱ تا ۳۶ مقداردهی کنید .تناسب اندام هر فرد را با استفاده از امتیاز تخلف محاسبه کنید .با استفاده از انتخاب مسابقات، دو نفر والدین را انتخاب کنید .انجام متقاطع با احتمال کامپیوتر :به طور تصادفی یک نقطه متقاطع را انتخاب کنید .دم ژنوم والدین را عوض کنید تا دو فرزند ایجاد کنید .هر یک از فرزندان را با احتمال Pmجهش دهید :به طور تصادفی دو عدد را در ژنوم عوض کنید .تناسب اندام فرزندان را ارزیابی کنید .افراد کم تناسب جمعیت را با نسل جدید جایگزین کنید 10 .درصد از افراد قوی نسل قبلی (نخبه گرایی) را حفظ کنید . مراحل ۳-۸ را تکرار کنید تا معیارهای توقف برآورده شوند .ژنوم نهایی بهترین فرد نشان دهنده راه حل معمای مربع زوج و فرد است

chatgptیک رویکرد دیگر تکمیلی توسط

برای حل این مشکل با استفاده از الگوریتم ژنتیک، باید ساختار ژنوم، تابع تناسب و پارامترها را تعریف کنیم

.در اینجا نحوه برخورد ما با آن آمده است :ساختار ژنوم :ما می توانیم یک راه حل بالقوه را به عنوان یک شبکه ۶ x6نشان دهیم که هر عنصر دارای یک عدد منحصر به فرد از ۱ تا ۳۶ است .به عنوان مثال، یک ژنوم می تواند به صورت لیستی از ۳۶ عدد منحصر به فرد نشان داده شود1] :. ۲، ۳، ۴، ۵، ۶، ۷، ۸، ۹، ۱۰، ۱۱، ۱۲، ۱۳، ۱۴، ۱۵، ۱۶، ۱۷، ۱۸، ۱۹، ۲۰، ۲۱، ۲۲، ۲۳، ۲۴، ۲۵ ، ۲۶، ۲۷، ۲۸، ۲۹، ۳۰، ۳۱، ۳۲، ۳۳، ۳۴، ۳۵، ۳۶]

عملکرد تناسب اندام :تابع تناسب ارزیابی خواهد کرد که ترتیب خاصی از اعداد تا چه حد شرایط داشتن تعداد مساوی از اعداد فرد و زوج را در هر سطر و ستون برآورده می کند

.تابع تناسب اندام را می توانیم به صورت زیر تعریف کنیم :برای هر سطر و ستون، تعداد اعداد زوج و فرد را بشمارید .تفاوت مطلق بین تعداد اعداد فرد و زوج در هر سطر و ستون را محاسبه کنید .تمام تفاوت های مطلق را خلاصه کنید .هر چه مجموع کمتر باشد، تناسب راه حل بهتر است .مولفه های :ما باید پارامترهایی مانند اندازه جمعیت، نرخ جهش، نرخ متقاطع و تعداد نسل ها را تعریف کنیم

.در اینجا یک طرح کلی از الگوریتم ژنتیک برای حل این مشکل آورده شده است:

:مقداردهی اولیه :یک جمعیت اولیه از راه حل های بالقوه (ژنوم) ایجاد کنید .

ارزیابی سازگاری :تناسب هر ژنوم را با استفاده از تابع تناسب ارزیابی کنید

.انتخاب :بر اساس تناسب اندام، ژنوم ها را برای نسل بعدی انتخاب کنید

.متقاطع :با ترکیب جفت ژنوم های انتخاب شده، ژنوم های جدید ایجاد کنید

.جهش :ایجاد تغییرات تصادفی در برخی از ژنوم ها برای حفظ تنوع ژنتیکی

.تکرار :مراحل ۲-۵ را برای تعداد معینی از نسل ها تکرار کنید

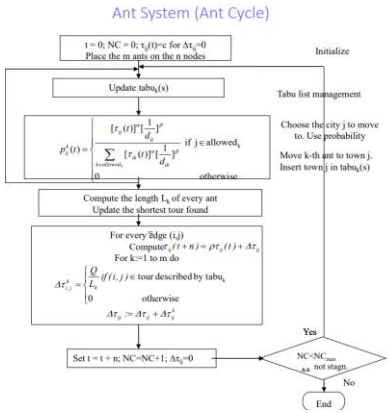
.با تکرار این مراحل، الگوریتم ژنتیک امیدوار است به سمت راه حلی همگرا شود که محدودیت های داده شده را برآورده کند .به خاطر داشته باشید که جزئیات پیاده سازی و تنظیم دقیق پارامترها به چارچوب الگوریتم ژنتیک خاص یا زبان برنامه نویسی مورد استفاده بستگی دارد.

سوال ۴

این سوال را با الگوریتم کلونی مورچگان بررسی می کنیم :

طبق اسلاید

نحوه پیاده سازی الگوریتم:



Algorithm

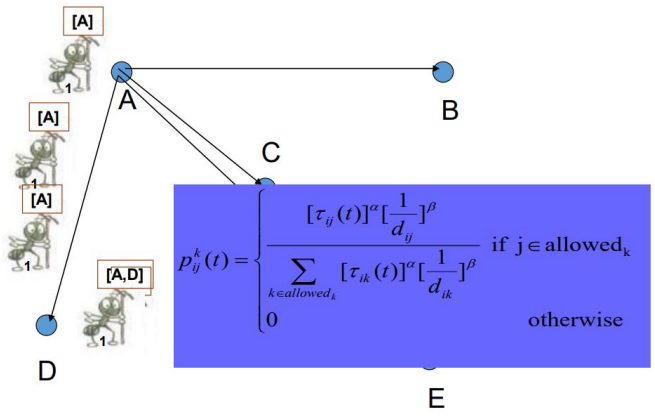
- While (termination not satisfied)
  - create ants
    - Starting point depends on problem constraints
    - Initial pheromone is > 0, but very small
  - Find solutions
  - Pheromone evaporation
  - Daemon activities {optional}

End of First Run

Save Best Tour (Sequence and length)

All ants die

New ants are born



• Ants do not know the global structure of the problem - discover the network

• Limited ability to sense local environment - can only “see” adjacent nodes of immediate neighborhood.

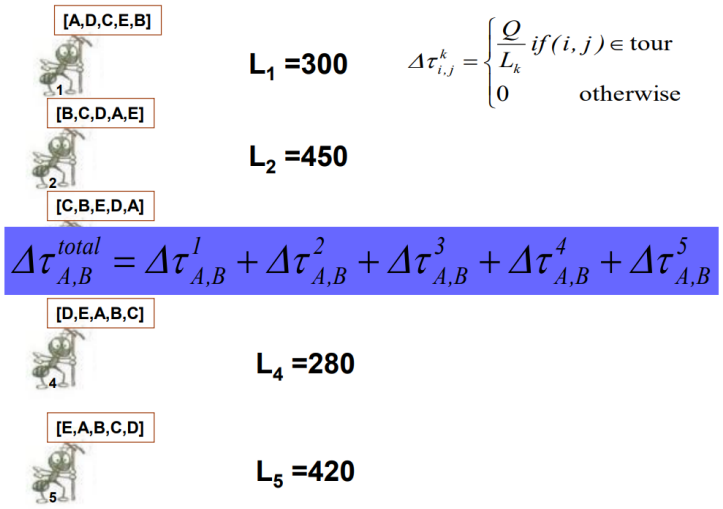
• Each ant chooses an action based on variable probability

• random choice

• pheromone mediated

• Local Updating is used to avoid very strong pheromone edges and hence increase exploration (and hopefully avoid locally optimal solutions).

• The Global Updating function gives the shortest path higher reinforcement by increasing the amount of pheromone on the edges of the shortest path.



نحوه ی انتخاب راه ها که با فرمون رابطه مستقیم دارد

- Find solutions
  - Transition probability:

$$P_{ij}(t) = \frac{\tau_{ij}(t)^{\alpha}\left(\frac{1}{d_{ij}}\right)^{\beta}}{\sum_{j \in \text{allowed nodes}} \tau_{ij}(t)^{\alpha}\left(\frac{1}{d_{ij}}\right)^{\beta}}$$

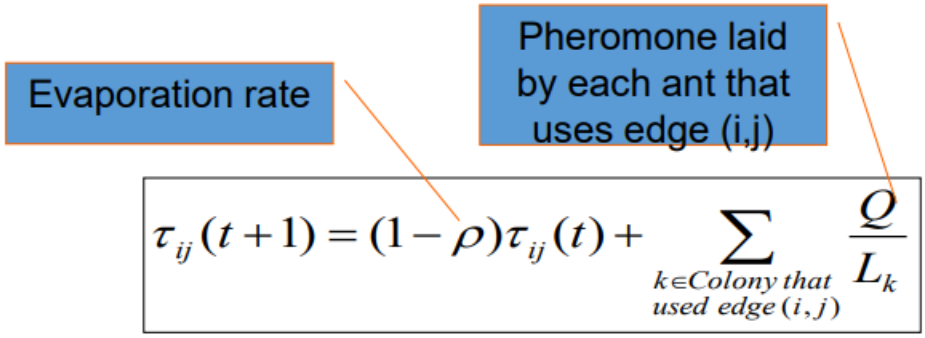
Quantity of pheromone

Heuristic distance

α,β constants

نحوه ی اپدیت فرمون ها که با نرخ تبخیر و فرمون های موجود در راه ها ارتباط دارد و بعد از یک دوره با توجه به مورچه ها و اطلاعات ان ها از اطلاعات محلی و اپدیت ان ها بدست می اید:

- Pheromone update



۲) همواره بهتر است که تولید ماتریس‌ها را به الگوریتم‌های زیست‌سنجی بسپاریم

S = تابع هیوریستیک (که در اینجا α_1 و α_2 برای سلول‌های روستا + درصد سلول‌های اروپا)

α_1 و α_2 وزن‌هایی که ما تعیین کرده‌ایم

distance = $\frac{1}{S + \beta}$ $\quad \beta = 0.02$

برای اینکه شباهت همسایگی بیشتر شود

احتمال انتخاب یکی از این خروجی‌ها به این بستگی دارد که فاصله‌های آن‌ها تا مبدأ چقدر باشد

و تابع چگالی احتمال هم دارد

$$p(t) = \frac{\tau_{ij}^{\alpha} \left(\frac{1}{d_{ij}}\right)^{\beta}}{\sum \tau_{ij}^{\alpha} \left(\frac{1}{d_{ij}}\right)^{\beta}}$$

زمان

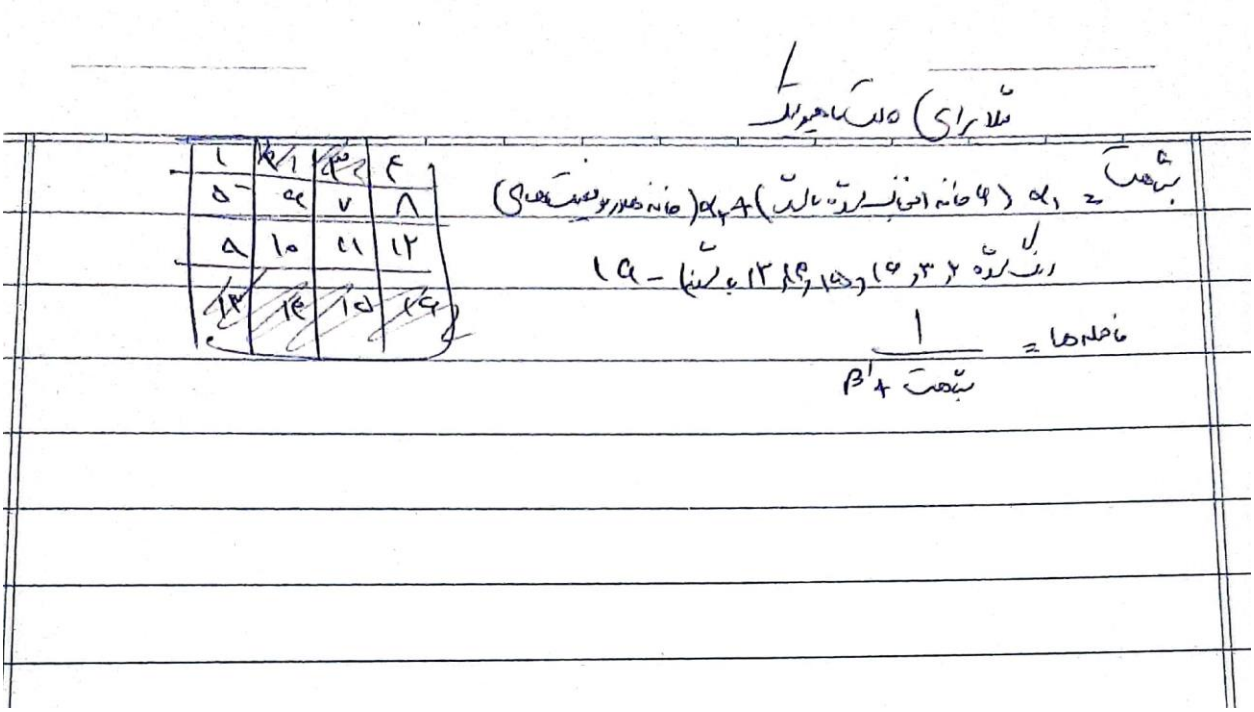
درج هم آپدیت می‌شود بعد از هر دوره که با نرخ تبخیر تغییر احتمالی هم دارد و فاصله‌ی هر node و Q زمان موجود در هر یک می‌باشد.

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum Q/L_k$$

بهترین جواب را آن نمونه‌ای است که بیشترین احتمال را از بین تعداد کمی مورد آن انتخاب می‌کنیم

پس از چند مرحله سال‌های نهایی جواب از خروجی‌ها بیشتر می‌آید و درجه‌ها از آن بیشتر تبعیت می‌کند و الگوریتم را تا جایی تکرار می‌کنیم که همگرا شود الگوریتم بعد چند مرحله از لحاظ درصد تغییری نکند استفاده می‌شود تا جواب نهایی را بدانیم همچنین از اقلیت‌ها هم برای همگرایی هم با احتمال کمتر استفاده می‌شود. آن‌هم بعد از چند مرحله تکرار بیشتر اینکه بیشتر مورچه‌ها از یک مسیر عبور کنند و از آن مسیر استفاده کنند و با احتمال کمتر از مسیرهای دیگر هم عبور کنند.

اقلیت‌ها بیشتر برای اکتشاف هستند



است که یک الگوریتم الهام گرفته شده از زیستی است که رفتار مورچه های واقعی را Ant Colony Optimization مخفف ACO را می توان با استفاده از مورچه ها برای جستجوی پیکسل ACO .برای یافتن راه حل های بهینه برای مسائل پیچیده تقلید می کند مورچه ها با قرار دادن دنباله های .هایی که به یک الگو یا حالت خاص در تصاویر تعلق دارند، برای تشخیص تصویر اعمال کرد مورچه ها تمایل .فرمون روی پیکسل هایی که بازدید می کنند با یکدیگر ارتباط برقرار می کنند که نشان دهنده کیفیت محلول است دارند پیکسل هایی با غلظت فرمون بالاتر را دنبال کنند، که منجر به یک حلقه بازخورد مثبت می شود که بهترین راه حل ها را :هستند، باید اجزای زیر را تعریف کنیم s برای تشخیص تصاویری که در حالت ACO برای استفاده از الگوریتم .تقویت می کند

- ساختار راه حل: یک راه ممکن برای ارائه راه حل برای این مشکل استفاده از یک ماتریس باینری هم اندازه تصویر است که در 4xبه عنوان مثال، یک راه حل برای یک تصویر ٤ .تعلق دارد یا خیر s آن هر عنصر نشان می دهد که آیا پیکسل مربوطه به حالت :می تواند به شکل زیر باشد

این راه حل به این معنی است که پیکسل ها در ردیف ها و ستون های دوم و سوم بخشی از حالت S هستند، در حالی که بقیه نیستند. • تابع تناسب: یک راه ممکن برای تعریف تابع تناسب برای این مشکل این است که تعداد پیکسل هایی که با حالت S مطابقت دارند را در راه حل شمارش کنیم و تعداد پیکسل هایی را که با حالت S مطابقت ندارند کم کنیم. هرچه ارزش تناسب اندام بالاتر باشد، راه حل بهتر است. به عنوان مثال، مقدار تناسب راه حل بالا 6=6-12 خواهد بود.

• پارامترهای الگوریتم: برخی از پارامترهایی که باید برای الگوریتم مشخص کنیم عبارتند از:

• تعداد مورچه ها: این تعداد عواملی است که تصویر را کاوش می کنند و راه حل ها را می سازند. تعداد بیشتر مورچه ها ممکن است تنوع و اکتشاف فضای جستجو را افزایش دهد، اما هزینه محاسباتی را نیز افزایش می دهد. مقدار ممکن برای این پارامتر ۱۶ است.

• سطح اولیه فرمون: مقدار فرمونی است که در ابتدای الگوریتم به هر پیکسل اختصاص می یابد. سطح فرمون اولیه بالاتر ممکن است مورچه ها را تشویق کند تا پیکسل های بیشتری را کشف کنند، اما همچنین از تمایز بین پیکسل های خوب و بد می کاهد. مقدار ممکن برای این پارامتر ۰,۱ است.

• نرخ تبخیر فرمون: این کسری از فرمون است که پس از هر تکرار الگوریتم از هر پیکسل از بین می رود. نرخ تبخیر فرمون بالاتر ممکن است از گیر افتادن مورچه ها در بهینه محلی جلوگیری کند، اما اثر بازخورد مثبت را نیز کاهش می دهد. مقدار ممکن برای این پارامتر ۰,۲ است.

• قانون به روز رسانی فرمون: این فرمولی است که تعیین می کند پس از هر تکرار الگوریتم، مورچه ها چه مقدار فرومون به هر پیکسل اضافه می کنند. یک قانون رایج به‌روزرسانی فرمون این است که مقدار ثابتی از فرمون را به پیکسل‌هایی که بخشی از بهترین راه‌حل یافت شده تاکنون هستند، اضافه می‌کنیم و بقیه را بدون تغییر می‌گذاریم. مقدار ممکن برای این پارامتر ۰,۵ است.

• قانون حرکت مورچه ها: این فرمولی است که تعیین می کند مورچه ها چگونه پیکسل بعدی را برای بازدید بر اساس پیکسل فعلی و سطوح فرمون انتخاب می کنند. یک قانون متداول حرکت مورچه ها استفاده از یک تابع احتمالی است که پیکسل هایی با غلظت فرمون بالاتر و ارزش تناسب بالاتر را ترجیح می دهد، اما همچنین امکان کاوش در پیکسل های با مقادیر کمتر را فراهم می کند. مقدار ممکن برای این پارامتر ۰,۸ است.

• شرط پایان: این معیاری است که تعیین می کند چه زمانی الگوریتم متوقف شود. یک شرط پایان متداول این است که حداکثر تعداد تکرار یا مقدار تناسب هدف را مشخص کنید. یک مقدار ممکن برای این پارامتر ۱۰۰ تکرار یا مقدار تناسب ۱۶ است.

پارامترهای الگوریتم کلونی مورچگان تنظیم شده و ردهای فرومون مقداردهی اولیه می‌شوند.

تا زمانی که شرط توقف ارضا نشده باشد:

مرحله اول یا مرحله «تولید جواب‌های کاندید (Construct Ant Solution) «را شروع کن.

مرحله دوم یا مرحله «جستجوی محلی جواب‌ها (Local Search) «را شروع کن. در این مرحله، از جواب‌های بهینه محلی استفاده می‌شود تا مشخص شود کدام یک از فرومون‌ها باید به روز رسانی شوند. این مرحله اختیاری است و در برخی از پیاده‌سازی‌های انجام شده از الگوریتم کلونی مورچگان وجود ندارد.

مرحله سوم یا مرحله «به روز رسانی فرومون (Pheromone Update) «را انجام بده.

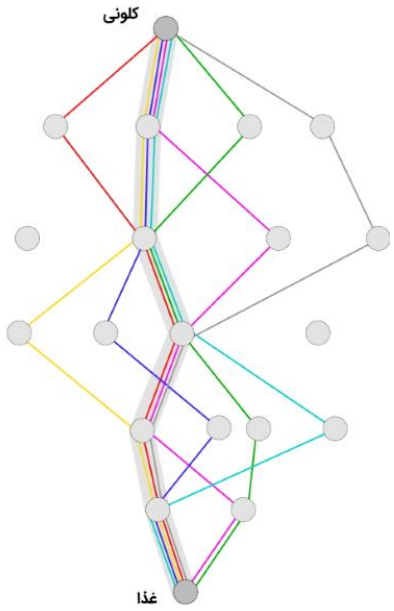
در صورتی که شرط توقف ارضا شده باشد، اجرای الگوریتم را متوقف کن؛ در غیر این صورت، مراحل را مجددا انجام بده.

الگوریتم کلونی مورچگان از دو بخش تشکیل شده است. در بخش اول، مقادیر پارامترهای مسأله تنظیم و متغیرهای جمعیت اولیه مقداردهی می‌شوند. بخش دو شامل یک حلقه تکرار است که سه مرحله الگوریتم کلونی مورچگان را اجرا می‌کند. در هر حلقه از الگوریتم کلونی مورچگان، جواب‌های کاندید توسط تمامی مورچه‌های مصنوعی ساخته می‌شوند. جواب‌های تولید شده، از طریق یک الگوریتم جستجوی محلی بهبود می‌یابند. در مرحله آخر، فرومون‌ها به روز رسانی می‌شوند.

شبه کد الگوریتم کلونی مورچگان در ادامه آمده است.

۱. پارامترهای الگوریتم کلونی مورچگان تنظیم شده و ردهای فرومون مقداردهی اولیه می‌شوند.
۲. تا زمانی که شرط توقف ارضا نشده باشد:
   - مرحله اول یا مرحله «تولید جواب‌های کاندید (Construct Ant Solution) «را شروع کن.
   - مرحله دوم یا مرحله «جستجوی محلی جواب‌ها (Local Search) «را شروع کن. در این مرحله، از جواب‌های بهینه محلی استفاده می‌شود تا مشخص شود کدام یک از فرومون‌ها باید به روز رسانی شوند. این مرحله اختیاری است و در برخی از پیاده‌سازی‌های انجام شده از الگوریتم کلونی مورچگان وجود ندارد.
   - مرحله سوم یا مرحله «به روز رسانی فرومون (Pheromone Update) «را انجام بده.
۳. در صورتی که شرط توقف ارضا شده باشد، اجرای الگوریتم را متوقف کن؛ در غیر این صورت، مراحل را مجددا انجام بده.

الگوریتم کلونی مورچگان از دو بخش تشکیل شده است. در بخش اول، مقادیر پارامترهای مسأله تنظیم و متغیرهای جمعیت اولیه مقداردهی می‌شوند. بخش دو شامل یک حلقه تکرار است که سه مرحله الگوریتم کلونی مورچگان را اجرا می‌کند. در هر حلقه از الگوریتم کلونی مورچگان، جواب‌های کاندید توسط تمامی مورچه‌های مصنوعی ساخته می‌شوند. جواب‌های تولید شده، از طریق یک الگوریتم جستجوی محلی بهبود می‌یابند. در مرحله آخر، فرومون‌ها به روز رسانی می‌شوند



مرحله اول: «تولید جواب‌های کاندید(Construct Ant Solution) »

فرآیند تولید جواب‌های کاندید را می‌توان به عنوان یک مسیر در گراف ساختاری $G_C = (V, E)$ در نظر گرفت. به عبارت دیگر، منظور از گسترش جواب بهینه، مشخص کردن مسیرهای حرکتی ممکن برای مورچه مصنوعی در گراف ساختار مدل فرومون است. از طریق چنین روشی، مناطق همسایگی جواب کاندید جزئی جستجو می‌شود تا بهترین مسیر در جهت جواب بهینه سراسری مشخص شود. مسیرهای مجاز در گراف G_C، به طور ضمنی توسط مکانیزم تولید جواب تعریف می‌شوند. مکانیزم تولید جواب، مجموعه همسایگان امکان‌پذیر $N(s^p) \subseteq C$ را نسبت به هر کدام از جواب‌های جزئی و به طور مجزا تعریف می‌کند.

در هر مرحله از تولید جواب‌های کاندید، نحوه انتخاب مؤلفه‌های مجموعه همسایگان امکان‌پذیر برای گسترش جواب کاندید جزئی، به صورت کاملا احتمالی انجام می‌شود. قوانین لازم برای انتخاب یک مؤلفه از مجموعه همسایگان امکان‌پذیر، در پیاده‌سازی‌های مختلف از الگوریتم کلونی مورچگان، متفاوت از هم هستند. با این حال، شناخته شده‌ترین قانون، مربوط به الگوریتم «سیستم مورچگان» (Ant System) است:

$$p(c_{ij} \mid s^p) = \frac{\tau_{ij}^{\alpha} . \eta(c_{ij})^{\beta}}{\sum_{c_{il} \in N(s^p)} \tau_{il}^{\alpha} . \eta(c_{il})^{\beta}}, \ \forall c_{ij} \in N(s^p)$$

در این رابطه، τ_{ij} مقادیر فرومون متناظر با مؤلفه c_{ij} و $\eta(.)$، تابعی است که در هر مرحله از تولید جواب کاندید، یک مقدار به اصطلاح «اکتشافی» (Heuristic) را به هر [مؤلفه] جواب کاندید $c_{ij} \in N(s^p)$ اختصاص می‌دهد. مقادیر اکتشافی تولید شده توسط تابع $\eta(.)$، «اطلاعات اکتشافی» (Heuristic Information) نامیده می‌شوند. همچنین، پارامترهای α و β، پارامترهایی با مقادیر مثبت هستند که مقدارشان، میزان اهمیت نسبی (وزن) اطلاعات فرومون (مقادیر متغیرهای یک جواب کاندید) و اطلاعات اکتشافی، در تولید مقدار احتمالی رابطه بالا را مشخص می‌کنند. این رابطه، تعمیمی از مدل ریاضی ارائه شده در بخش‌های قبل، برای توصیف رفتار بهینه مورچگان

مرحله دوم: «جستجوی محلی جواب‌ها(Local Search) »

سته به پیاده‌سازی انجام شده از الگوریتم کلونی مورچگان، پس از تولید جواب‌های کاندید و پیش از اینکه فرومون‌ها به روز رسانی شوند، ممکن است فرآیندهای اضافی برای تضمین عملکرد بهینه الگوریتم ضروری باشند. بنابراین، این مرحله، اختیاری است. طبیعت این فرآیندها، ازدحامی است. یعنی توسط فقط یک مورچه مصنوعی قابل انجام نیست. به این دسته از فرآیندها، «عملیات کمکی یا پس‌زمینه (Daemon Actions) «گفته می‌شود.

شایع‌ترین عملیات پس‌زمینه در الگوریتم‌های مبتنی بر الگوریتم کلونی مورچگان، به‌کارگیری جستجوی محلی در جواب‌های کاندید تولید شده است. به‌عنوان نمونه، از جواب بهینه شده محلی برای تصمیم‌گیری در مورد به روز رسانی مقادیر فرومون‌ها استفاده می‌شود.

مرحله سوم: «به روز رسانی فرومون(Pheromone Update) »

هدف مرحله به روز رسانی فرومون، افزایش مقادیر فرومون متناظر با جواب‌های کاندید خوب و بهینه و کاهش مقادیر فرومون متناظر با جواب‌های بد است. چنین کاری، از طریق دو فرآیند عمده انجام می‌شود:

۱. کاهش مقادیر فرومون متناظر با تمامی جواب‌های کاندید از طریق فرآیند «تبخیر فرومون» (Pheromone Evaporation)

۲. افزایش مقادیر فرومون متناظر با جواب‌های کاندید عضو مجموعه «جواب‌های خوب» یا S_{upd}

این دو فرآیند از طریق رابطه زیر کنترل می‌شوند. به رابطه زیر، «قانون به روز رسانی فرومون» گفته می‌شود.

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho \sum_{s \in S_{upd} | c_{ij} \in s} F(s)$$

بخش اول این رابطه، فرآیند تبخیر فرومون (کاهش مقدار فرومون تمامی جواب‌های کاندید) را کنترل می‌کند. بخش دوم آن، تنها مقادیر فرومون متناظر با جواب‌های کاندید عضو مجموعه «جواب‌های خوب» یا S_{upd} را افزایش می‌دهد. در این رابطه، S_{upd} مجموعه جواب‌های کاندیدی را شامل می‌شود که برازندگی بالایی دارند؛ یعنی به جواب بهینه سراسری نزدیک‌تر هستند. پارامتر $\rho \in (0, 1]$، «نرخ تبخیر» (Evaporation Rate) نام دارد و $F : S \rightarrow R_0^+$، «تابع برازندگی» (Fitness Function) نام دارد.

به بیان ساده، این فرآیند منجر به افزایش مقدار فرومون متناظر با مورچه‌هایی می‌شود که در بهترین مسیرهای موجود به سمت جواب بهینه قرار دارند (به جواب بهینه نزدیک‌ترند) و دارای برازندگی بالاتری (هزینه کمتر یا سود بیشتر) هستند. در نتیجه، مورچه‌های دیگر نیز به این مسیر همگرا می‌شوند.

وجود پارامتر تبخیر فرومون، برای جلوگیری از «همگرایی سریع و نابالغ» (Rapid and Premature Convergence) الگوریتم کلونی مورچگان ضروری است. پارامتر تبخیر، نوعی مکانیزم «فراموشی» (Forgetting) در فرآیند بهینه‌سازی فراهم می‌کند و سبب تاکید بیشتر بر جستجو و کاوش مناطق جدید، در فضای جستجوی الگوریتم‌های پیاده‌سازی شده مبتنی بر الگوریتم کلونی مورچگان می‌شود.

\

تبخیر رد فرومون پس از چند تکرار از روش بهینه‌سازی کلونی مورچگان

یشتر تفاوت موجود میان پیاده‌سازی‌های گوناگون انجام شده از الگوریتم کلونی مورچگان، به قانون به روز رسانی فرومون‌ها و نحوه شخص کردن جواب‌های کاندید عضو مجموعه «جواب‌های خوب» یا S_{upd} مرتبط است. یکی از مهم‌ترین بخش‌های مرحله به روز رسانی رومون‌ها، انتخاب اعضای مجموعه «جواب‌های خوب» S_{upd} است.

ر بیشتر الگوریتم‌های تکاملی مبتنی بر الگوریتم کلونی مورچگان، S_{upd} زیرمجموعه‌ای از $S_{iter} \cup \{s_{bs}\}$ تعریف می‌شود. در این رابطه، S_{iter} مجموعه جواب‌های کاندید تولیده شده در تکرار کنونی است، و s_{bs} مجموعه بهترین جواب‌های کاندید پیدا شده از اولین تکرار لگوریتم تاکنون است.

ه عنوان نمونه، در قانون به روز رسانی فرومون الگوریتم سیستم مورچگان، مجموعه جواب‌های کاندید عضو مجموعه «جواب‌های خوب» ز طریق رابطه $S_{upd} \leftarrow S_{iter}$ مشخص می‌شوند.

ر صورتی که در قانون به روز رسانی فرومون‌ها، تاکید بیشتری روی استفاده از جواب‌های خوب هر نسل برای به روز رسانی باشد، سرعت ستیابی به جواب‌های خوب بهینه افزایش می‌یابد. با این حال، احتمال همگرایی نابالغ افزایش پیدا می‌کند. معمولا توصیه می‌شود که در هنگام پیاده‌سازی قانون به روز رسانی فرومون، مکانیزم‌هایی طراحی شوند تا از همگرایی سریع و نابالغ الگوریتم جلوگیری شود.