- **نحوه تولید جمعیت اولیه:**

0	1	2	3	...	31	32	33
V_1	V_2	V_3	V_4	...	V_{31}	V_{32}	V_{33}

همانطور که در کروموزم بالا دیده میشود نحوه تولید جمعیت براساس یک آرایه 34 تایی میباشد که خانه i-ام نشان دهنده راس i-ام میباشد و اعداد قرار گرفته در خانه i-ام نشان دهنده راسی است که راس i-ام به آن وصل میباشد. از طریق این کروموزم میتوان خوشه ها را پیدا کرد.

این کروموزم را از طریق تابع habitat_safty () میسازیم به طوری که مقادیر قرار گرفته در هر خانه معتبر باشد. یعنی حتما یالی بین آن راس ها باشد. **جمعیت اولیه را 20 انتخاب میکنیم.**

```
def habitat_safty():
    habitae=[]
    for i in range(20):
        temp_gene=[]
        for j in range(number_of_node):
            nonzeroind_index=np.nonzero(adjaceny_matrix[j,:])
            conncted_node_j=random.choice(nonzeroind_index[0])
            temp_gene.append(conncted_node_j)

        habitae.append(temp_gene)

    return habitae
```

- **تابع fetness:**

تابع def habitae_safty(habitat) جمعیت کوکوهارا به عنوان ورودی میگیرد و برازندگی و خوشه بندی آنها به عنوان خروجی میدهد.

- **def hatch_egg(habitat) :**

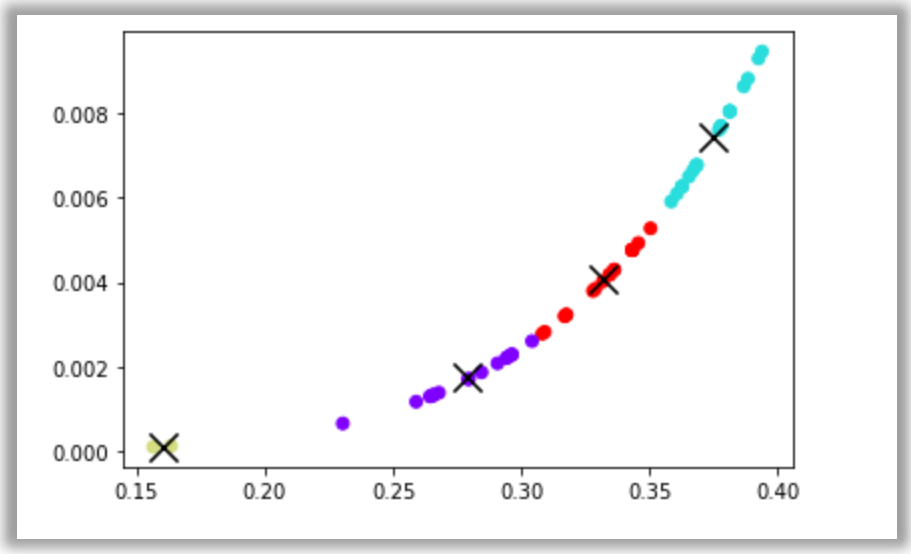
در این مرحله ابتدا تعداد تخم ها را برای هر کوکو به صورت تصادفی انتخاب میکنیم (برای هر کوکو بین 5 تا 10 تخم به صورت تصادفی انتخاب میکنیم). از آنجایی که این مسئله گسسته است و شعاع یا فاصله تخم گذاری برای فضاهای گسسته بی معنی و تعریف نشده است شعاع تخم گذاری برای این مسئله به صورت یک مثال توضیح میدهیم. فرض کنیم کوکو مربوطه 7 تخم گذاشته است و قرار است این تخم ها را در مکان های مختلفی قرار دهیم. در ابتدا پارامتر $\alpha = 1$ تعریف میکنیم. این پارامتر عدد صحیح مثبتی است و با توجه به مسائل مختلف متفاوت است. برای تخم اول یکی از ژن های کروموزم کوکو را ($\alpha \times 1$) به تصادف انتخاب میکنیم (از بین ژن های 0 تا 34) و آن را به تصادف تغییر میدهیم. کروموزم حاصل شده مربوط به تخم اول میباشد. به همین ترتیب با توجه به شماره تخم تعداد ژن های $\alpha \times n$ برای تغییر انتخاب میشوند تا کروموزم مربوط به تخم حاصل شود. مثلا برای تخم هفتم به تصادف $\alpha \times 7$ ژن را تغییر میدهیم. بعد از آنکه مرحله تخم گذاری کامل شد با استفاده از **roulette_selection** بهترین تخم های گذاشته شده از نظر تابع برازندگی انتخاب میشوند. **با این گونه تخم گذاری در واقع تخم هایی در محدوده مکانی از کوکو گذاشته میشود.**

- **def clustring_cuckoo(habitat):**

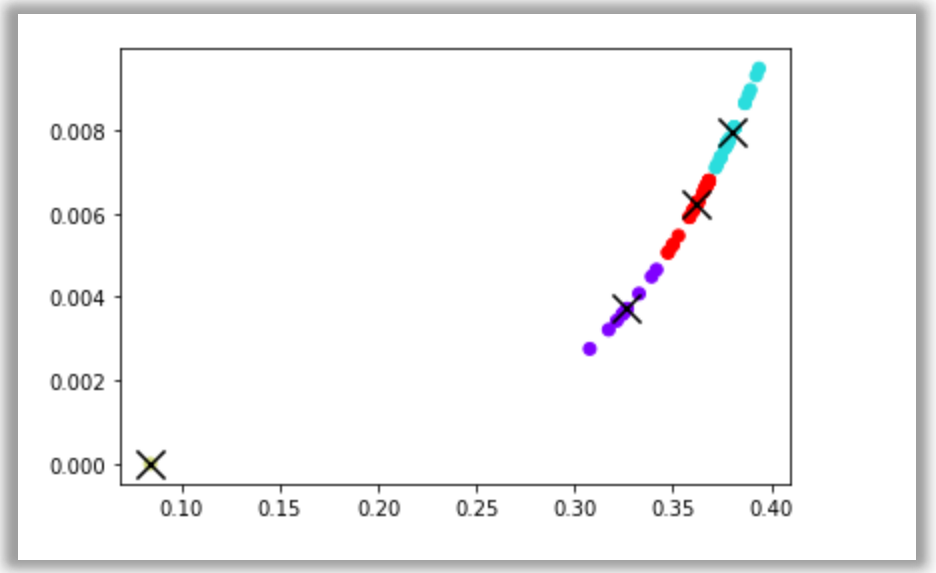
در این مرحله کوکو ها و تخم هایی که به جوجه تبدیل شده اند باید با استفاده از **مقدار برازندگی خوشه بندی** شوند. برای خوشه بندی ابتدا تابع برازندگی را با استفاده از نگاشت $k(x) = x^5$ **به دو بعد گسترش میدهیم**(نگاشت $k(x) = x^5$ به دلخواه انتخاب شده است) و سپس با استفاده از الگوریتم **k_mean** به **4** خوشه دسته بندی میشوند. بعد از خوشه بندی شدن بهترین خوشه از نظر برازندگی انتخاب میشود. یعنی میانگین تابع براندگی در هر خوشه مقایسه میشود تا بهترین خوشه مشخص شود. بعد از مشخص شدن بهترین زیستگاه(بهترین خوشه) باید بقیه کوکو های از زیستگاه های دیگر (از خوشه های دیگر) به نزدیکی بهترین زیستگاه مهاجرت کنند. برای این منظور کوکو هایی که در بهترین زیستگاه نیستند را انتخاب میکنیم و سپس کوکویی به صورت تصادفی از بهترین زیستگاه انتخاب میکنیم به این صورت که باید به صورت رندوم بین 20 تا 30 ژن از آن را انتخاب میکنیم و به کوکویی که در زیستگاه دیگر است اختصاص میدهیم. بعد از این مراحل مشخص میشود که سایر خوشه ها به بهترین خوشه نزدیک میشوند.

در نمودار زیر خوشه بندی کوکو ها **قبل از مهاجرت** به بهترین زیستگاه آمده است. و همان طور که مشخص است بهترین زیستگاه مربوط به خوشه آبی میباشد که برازندگی آن از کمی بیشتر از 0.35 شروع میشود.

پایین زیسگاه **بعد از** نشان داده مشخص است کوکو ها به زیستگاه شده اند.



در شکل کوکو ها **مهاجرت** میشود که بیشتر بهترین نزدیک



- **def kill_cuckoo(habitat):** :

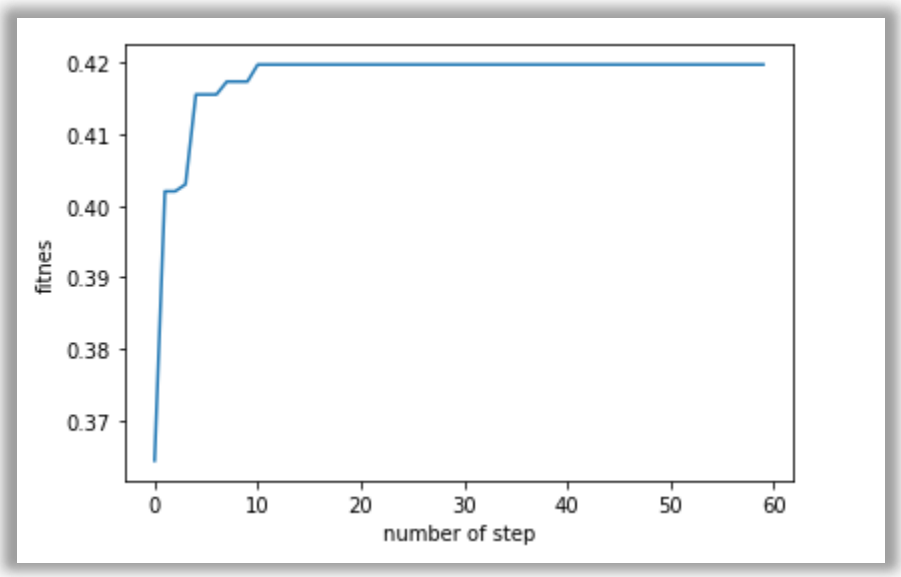
در این مرحله برخی از کوکو ها که زیستگاه مناسبی از نظر تابع برازندگی ندارند را حذف میکنیم.

اگر تعداد کوکو ها بیشتر از **75** شود با استفاده **از roulette_selection یک دهم** از آن جمعیت را انتخاب میکنیم و بقیه کوکو ها را منقرض میکنیم.

- **شرط توقف:**

شرط توقف را اینطور در نظر میگیریم که اگر تابع برازندگی بعد از 50 تکرار تغییری نکرد الگوریتم ما متوقف شود

.



همانطور که در شکل بالا دیده میشود روند تغییرات تابع fetnes از تکرار دهم به بعد ثابت میشود.

- **نتیجه**:

در نهایت بعد از چند تکرار از مراحل الگوریتم به مقادیر زیر میرسیم.

```
i=np.argmax(fetnes)
cluster=cluster[i]
print('maximum fetnes Q is:', max(fetnes))
```

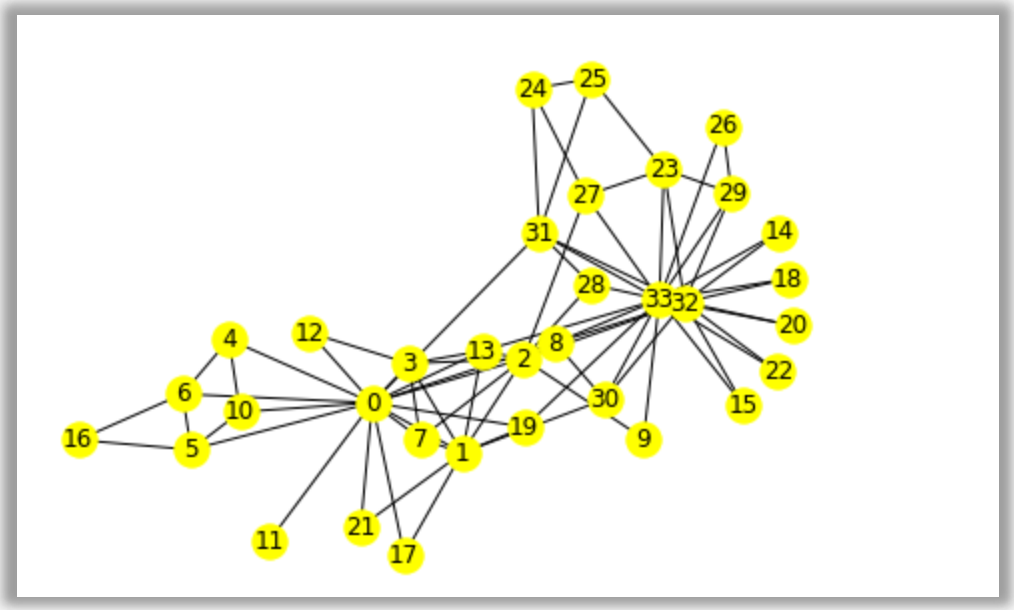
```
maximum fetnes Q is: 0.41978961209730387
```

```
cluster
```

```
[[31, 25, 28, 23, 27, 24],
 [29, 26, 32, 14, 15, 18, 20, 22, 33, 9, 30, 8],
 [7, 0, 11, 12, 17, 3, 2, 13, 1, 19, 21],
 [5, 6, 16, 4, 10]]
```

➢ **توجه:** **در خوشه بندی و همینطور در گراف راس ها از صفرنام گذاری شده اند.**

- **گراف قبل از خوشه بندی:**



- **گراف بعد از خوشه بندی:**