تنظیم پارامتر

ستاره روشن

محمدرضا رضائی

حسین مختاریان

چکیده — در این گزارش به بهینه کردن پارامترها در حل مسئله چینش کودها بر اساس دسترس پذیری و بهروری می‌پردازیم. نبوغ این کار استفاده و ترکیب دو کتابخانه در parameter tuning است.

1. پیش پردازش‌ها

در ابتدا دیتاستی را که در task چهارم بدست آورده بودیم فراخوانی می‌کنیم. سپس در آرایه قرار داده و روی اسامی کودها و کشورها factorize انجام می‌دهیم.

```
= pd.read_excel(r"C:\Users\mcf\OneDrive\Desktop\Availability (1).xlsx")
mples = np.asarray(df)
```

untry	Fertilizers	Import&Product	Export	Availability	Use	x
Brazil	Ammonia, anhydrous	24144572.05	399156.28	23745415.77	2571549.00	5931.614778
Brazil	Ammonium nitrate (AN)	21546416.92	214625.65	21331791.27	13821890.37	31882.001541
Brazil	Ammonium sulphate	30988444.53	53840.25	30934604.28	22805149.59	52603.066217
Brazil	Calcium ammonium nitrate (CAN) and other mixtu...	2020987.02	10844.03	2010142.99	1164236.84	2685.464848
Brazil	Diammonium phosphate (DAP)	6948831.81	148854.13	6799977.68	4846157.53	11178.297447
...	...	...	...	...	...	...
lexico	Sodium nitrate	32779.02	1030.91	31748.11	0.00	0.000000
lexico	Superphosphates above 35%	622825.23	2467787.77	-1844962.54	811919.38	1699.959390
lexico	Superphosphates, other	0.00	0.00	0.00	21682383.00	45397.574551
lexico	Urea	21917270.86	303460.75	21613810.11	8931655.72	18700.689238
lexico	Urea and ammonium nitrate solutions (UAN)	1028338.92	46263.77	982075.15	305737.70	640.139510

شکل أ فراخوانی دیتاست.

در گام بعد هر کرومزوم بصورت زیر خواهد بود که شامل 23 ژن می‌باشد. کود در آلل (allel) شماره 0، کودی با بیشترین اهمیت از نظر دسترس پذیری و بهروری خواهد بود. اگر کشوری شامل کودی نباشد در محاسبه fitness کود محاسبه نخواهد شد. کرومزوم زیر کرومزومی فرضی است که در آن کود شماره یک دارای بیشترین اهمیت و کود شماره 23 دارای کمترین اهمیت است.

کود شماره یک	کود شماره دو	کود شماره سه	کود شماره چهار	...	کود شماره بیست و دو	کود شماره بیست و سه

همچنین در بحث بهینه سازی کرومزوم ما به صورت زیر می‌باشد.

Mutation probability	Crossover probability	Tournament size	Population size	Selection method	Crossover method	Mutation method

2. تعریف مسئله

در این بخش ما به تعریف مسئله خواهیم پرداخت. در pymoo نیاز است که تعداد متغیرها، اهداف و حد بالا و پایین متغیرها تعریف شود. در این قسمت با استفاده از تابع super این کار انجام شده که برای تمام الگوریتم‌های ثابت است در این مسئله خاص. مسئله ما دارای 7 متغیر(خط 4)، یک هدف (خط 5)، بدون قید (خط 6)، و کران‌های پایین و بالا برای هر متغیر (خط 7و 8) است. سپس قسمت evaluate خواهد بود که شامل مینیمم کردن هدف است (مسئله اصلی ماکزیمم کردن است که با ضرب یک منفی با مینیمم کردن در واقع ماکزیمم را بدست می‌آوریم. در خود حل مسئله ما در طول 200 نسلی که داریم best fitnessها را محاسبه می‌کنیم و در result قرار می‌دهیم. سپس به ازای هر عضو جمعیت باید mean best fitness این result بدست آید، در خط 17، و بعد در out آن را قرار می‌دهیم.

```
1  class MyProblem (Problem):
2
3      def __init__(self):
4          super().__init__(n_var=7,
5                           n_obj=1,
6                           n_constr=0,
7                           xl=np.array([0,0,1,100,0,0,0]),
8                           xu=np.array([10,10,100,400,4,2,1]))
9
10     def _evaluate(self, X, out, *args, **kwargs):
11         #print (X.shape)
12         mbf = np.zeros(X.shape[0])
13         for i in range (X.shape[0]):
14
15             result, win = main (Selectlist[int(X[i,4])], CXlist[int(X[i,5])], mut[int(X[i,6])],
16                                 X[i,0]/10, X[i,1]/10, int(X[i,2]), int(X[i,3]))
17             mbf[i] = -(np.mean(result, axis = None))
18
19         #plt.plot(-mbf)
20         #plt.show()
21         out["F"] = np.column_stack([mbf])
22         #out["G"] = np.column_stack([0,0])
23 problem = MyProblem()
```

شکل ب. تعریف مسئله

۳. الگوریتم

این قسمت بسیار ساده و صریح است. تنها نیاز است الگوریتم فراخوانی و import شود (شکل ت). با الگوریتم GA بهینه سازی را انجام می‌دهیم. از Myproblem در بخش قبل شکل ب شی ساختیم که در minimize از آن استفاده شده. در خط دوم مقادیری به الگوریتم داده به ترتیب، اندازه جمعیت (برابر 100)، نوع representation (در اینجا Integer که اول بصورت رندوم)، سپس crossover (در اینجا تغییر one-point)، از عملگر جهش در این بخش استفاده نشده.

جدول أ. پارامترهای مورد استفاده

Representation	Integer

Crossover	One point
Population size	50
Initial population	Random
Termination	20 generation

```
from pymoo.algorithms.so_genetic_algorithm import GA
from pymoo.optimize import minimize

algorithm = GA(pop_size=20, eliminate_duplicates=True, save_history=True,sampling=get_sampling("int_random"))

res = minimize(problem, algorithm, seed=1, verbose=True,termination = ('n_gen', 20), callback=MyCallback())

print("Best solution found: \nX = %s\nF = %s" % (res.X, res.F))
```

شکل ت. فراخوانی الگوریتم GA

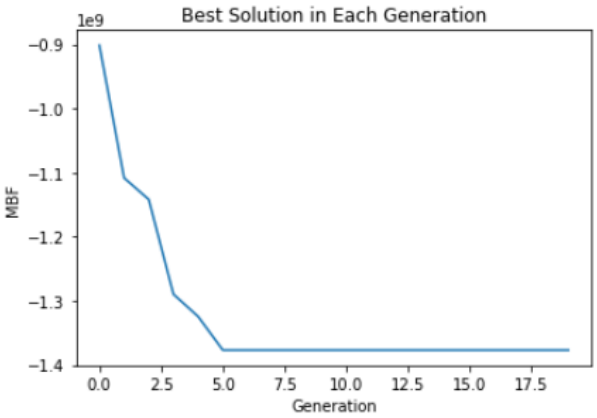
```
from pymoo.model.callback import Callback
class MyCallback(Callback):

    def __init__(self) -> None:
        super().__init__()
        self.data["best"] = []

    def notify(self, algorithm):
        self.data["best"].append(algorithm.pop.get("F").min())
```

شکل ث. تابع call back

به منظور استفاده از قابلیت نمایش نتایج از callback به همراه save history استفاده شده است. در هر generation بهترین داده گرفته می‌شود در زیر نمایش داده شده است.



شکل ج. نمودار بهترین راه حل در هر نسل. محور عمودی نمایانگر MBF و محور افقی نمایانگر نسل است.

شکل زیر نمایانگر هر نسل و optimum fitness و average fitness است و اگر verbose = true قرار گیرد این جدول توسط خود الگوریتم ساخته می‌شود (سرعت به شدت پایین می‌آید در نتیجه توصیه نمی‌شود.).

```
=============================================
n_gen |  n_eval |     fopt     |     favg
=============================================
    1 |      20 | -9.01663E+08 | -4.51298E+08
    2 |      40 | -1.10844E+09 | -6.87582E+08
    3 |      60 | -1.14206E+09 | -8.90938E+08
    4 |      80 | -1.28983E+09 | -1.08024E+09
    5 |     100 | -1.32457E+09 | -1.21352E+09
    6 |     120 | -1.37688E+09 | -1.28763E+09
    7 |     140 | -1.37688E+09 | -1.32217E+09
    8 |     160 | -1.37688E+09 | -1.33616E+09
    9 |     180 | -1.37688E+09 | -1.34225E+09
   10 |     200 | -1.37688E+09 | -1.34895E+09
   11 |     220 | -1.37688E+09 | -1.35011E+09
   12 |     240 | -1.37688E+09 | -1.35242E+09
   13 |     260 | -1.37688E+09 | -1.35433E+09
   14 |     280 | -1.37688E+09 | -1.35731E+09
   15 |     300 | -1.37688E+09 | -1.36003E+09
   16 |     320 | -1.37688E+09 | -1.36092E+09
   17 |     340 | -1.37688E+09 | -1.36246E+09
   18 |     360 | -1.37688E+09 | -1.36340E+09
   19 |     380 | -1.37688E+09 | -1.36397E+09
   20 |     400 | -1.37688E+09 | -1.36476E+09
```

شکل ح جدول بدست آمده از روند اجرای الگوریتم. به ترتیب n_gen نشان دهنده شماره نسل است، n_eval نشان دهنده تعداد evaluation ها است، fopt فیتنس بهینه و favg میانگین فیتنس‌ها در هر نسل می‌باشد.

همچنین بهترین پاسخ به شرح زیر است با فیتنس: -1.37687922e+09

Mutation probability	Crossover probability	Tournament size	Population size	Selection method	Crossover method	Mutation method
0.9	0.8	26	316	Tournament	pmx	Shuffle

زمانی که با این مقادیر الگوریتم را اجرا کردیم به ترتیب زیر از کودها رسیدیم 15 با ارزش ترین و 14 بی ارزش ترین است:

15, 13, 20, 21, 6, 19, 2, 0, 1, 10, 4, 5, 3, 8, 17, 12, 11, 16, 18, 7, 9, 14