حل مسئله چند هدفه (قسمت سوم)

ستاره روشن

حسین مختاریان

محمدرضا رضائی

چکیده — در این گزارش به بررسی عملکرد چهار الگوریتم تکاملی آخر که در سایت pymoo موجود است، بر روی مسئله یافتن ترتیب کودها با ماکزیمم دسترس پذیری همچنین ماکزیمم بهروری پرداخته خواهد شد. با توجه به اینکه الگوریتم UNSGA-III در پارت اول این پژوهش آورده شده است ما نتایج را برای سه الگوریتم RNSGA-III ، MOEA/D و C-TAEA ارائه می دهیم.

۱. 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.

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

```
samples = np.asarray(df)
```

```
df
```

country	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
...	...	...	...	...	...	...
Mexico	Sodium nitrate	32779.02	1030.91	31748.11	0.00	0.000000
Mexico	Superphosphates above 35%	622825.23	2467787.77	-1844962.54	811919.38	1699.959390
Mexico	Superphosphates, other	0.00	0.00	0.00	21682383.00	45397.574551
Mexico	Urea	21917270.86	303460.75	21613810.11	8931655.72	18700.689238
Mexico	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 ۲۳ ژن می‌باشد. کود در آلل (allel) شماره ۰، 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 ۲۳ دارای کمترین اهمیت است.

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

۲. 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 ۲۳ متغیر(خط ٤)، دو هدف (خط ٥)، بدون قید (خط ٦)، و کران پایین ۰ و کران بالای ۲۳ برای هر متغیر (خط ۷و ۸) است. سپس قسمت evaluate خواهد بود که شامل مینیمم کردن دو هدف f1 و f2 است (مسئله اصلی ماکزیمم کردن این دو که با ضرب یک منفی با مینیمم کردن در واقع ماکزیمم را بدست می‌آوریم.). برای هر کشور چک می‌شود و اهمیت هر کود با توجه به مکانی که قرار دارد با مقادیر قبل جمع می‌شود. باید خاطر نشان کرد که در این کتابخانه اول تمامی جمعیت به این تابع داده می‌شود پس نیاز است آرایه‌ای به اندازه تعداد جمعیت برای f1 و f2 درنظر گرفته بشود (خط۱۱ و ۱۲). در آخر f1 (دسترس پذیری) و f2 (بهروری) در آرایه خط ۲۳ کنار هم به ازای هر عضو جمعیت قرار می‌گیرند.

```
class MyProblem(Problem):

    def __init__(self):
        super().__init__(n_var=23,
                         n_obj=2,
                         n_constr=0,
                         xl=np.array([0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]),
                         xu=np.array([23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23]))

    def _evaluate(self, X, out, *args, **kwargs):
        f1 = np.zeros(X.shape[0])
        f2 = np.zeros(X.shape[0])
        # print(X.shape)
        for p in range (X.shape[0]):
            for i in range (22):
                temp = X[p,i]
                for j in range (115):
                    if samples [j,0] == 4 and samples [j,1] == X[p,i]:
                        f1[p] += -(samples[j,4] * (23 - i))
                        f2[p] += -(samples[j,6] * (23 - i))


        out["F"] = np.column_stack([f1, f2])
        #out["G"] = np.column_stack([0,0])

```

شکل ب. تعریف مسئله

۳. الگوریتم‌ها

در این بخش ابتدا فاز پیاده سازی الگوریتم RNSGA-III معرفی شده (شکل ت). و سپس نتیجه pareto front ارائه خواهد شد (شکل ث). فراخوانی و import را با الگوریتم RNSGA-III آغاز میکنیم نیاز است از Myproblem در بخش قبل یک شی ساخته شود (به نام problem). سپس با استفاده از جدول زیر، مقادیر را در بخش های مختلف الگوریتم به طور متناظر جایگذاری می کنیم. در خط ۱۲ مقادیری به الگوریتم داده شده است که خلاصه وضعیت مقدار پارامترها در جدول زیر معرفی شده است تا از آن در الگوریتم های بعدی بهره بریم.

جدول أ. پارامترهای مورد استفاده

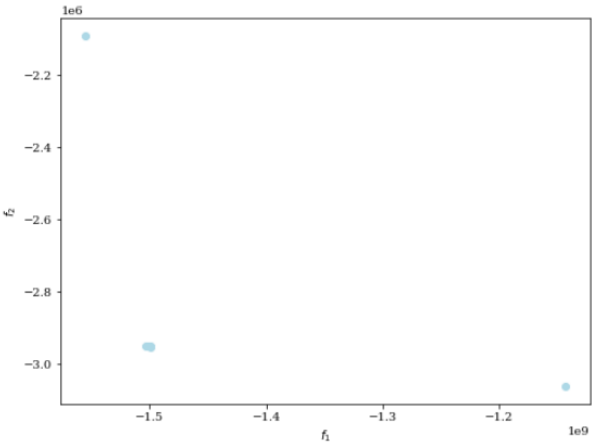
Representation	Permutation
crossover	Order
mutation	Inverse

Population size	100
Initial population	Random
Termination	500 generation

```python
from pymoo.algorithms.rnsga3 import RNSGA3

# Define reference points
ref_points = get_reference_directions("das-dennis", 2, n_partitions=12)

# Get Algorithm
algorithm = RNSGA3(ref_points = ref_points, pop_per_ref_point = 100,
                   sampling=get_sampling("perm_random"),
                   crossover=get_crossover("perm_ox"),
                   mutation=get_mutation("perm_inv"),
                   eliminate_duplicates=True)
res =  minimize (problem, algorithm, Termination = ("n_gen", 200), seed = 1, verbose = False, save_history=True)

reference_directions = res.algorithm.survival.ref_dirs

plt = Scatter()
plt.add(problem.pareto_front(), plot_type="line", color="black", alpha=0.7)
plt.add(res.F, color="lightblue")
#plt.add(ref_points)
plt.show()
```

شکل ت. فراخوانی الگوریتم RNSGA-III

نتایج در خط ۱۲ ذخیره می‌شود. خط آخر کد نیز نتیجه pareto front را به ما نشان می دهد.(شکل ث). اینجا ref points را به پیشنهاد مقاله می توان به صورت [۰,۸ ۰,۵] [۰,٤ , ۰,۳] در نظر گرفت



شکل ث. نتیجه pareto front الگوریتم RNSGA-III

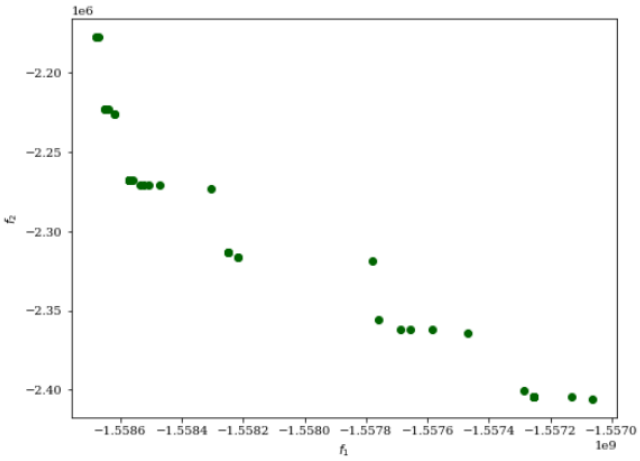
(محور عمودی f2 ومحور افقی f1)

الگوریتم بعدی MOEAD است. اینجا باید به اندازه اهداف reference direction داشته باشیم (خط ۲) و بقیه پارامترها مشابه جدول أ، است.

```python
from pymoo.algorithms.moead import MOEAD
algorithm = MOEAD(get_reference_directions("das-dennis", 2, n_partitions=99),
                  n_neighbors=20,
                  decomposition="pbi",
                  prob_neighbor_mating=0.7,pop_size = 100,seed=1,sampling=get_sampling("perm_random"),
                  crossover=get_crossover("perm_ox"),mutation=get_mutation("perm_inv"),eliminate_duplicates=True)

res_MOEAD = minimize(problem, algorithm, termination=('n_gen', 200), verbose = False, save_history=True)
plt = Scatter()
plt.add(problem.pareto_front(), plot_type="line", color="black", alpha=0.7)
plt.add(res_MOEAD.F, color="darkgreen")
#plt.add(ref_points)
plt.show()
```

شکل ج. فراخوانی الگوریتم MOEAD

در شکل زیر pareto front این الگوریتم دیده می‌شود.



شکل ح. نمودار pareto front برای MOEAD

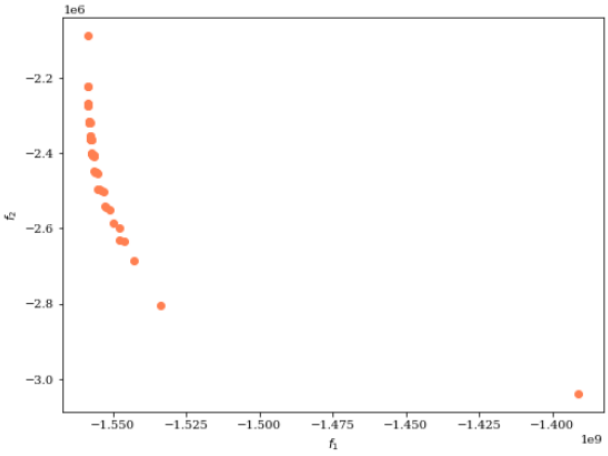
(محور عمودی f2 و محور افقی f1)

در آخر به الگوریتم CTAEA می‌پردازیم. پارامترها شبیه به جدول أ، استفاده شده‌اند. و reference direction هم در خط ۲ تعریف شده است.

```python
from pymoo.algorithms.ctaea import CTAEA

ref_dirs = get_reference_directions("das-dennis", 2, n_partitions=100)

algorithm = CTAEA(sampling=get_sampling("perm_random"),
                  crossover=get_crossover("perm_ox"),
                  mutation=get_mutation("perm_inv"),
                  eliminate_duplicates=True,
                  ref_dirs=ref_dirs)
res_CTAEA = minimize(problem, algorithm, termination=('n_gen', 200), verbose = False, save_history=True)
plt = Scatter()
plt.add(problem.pareto_front(), plot_type="line", color="black", alpha=0.7)
plt.add(res_CTAEA.F, color="coral")
#plt.add(ref_points)
plt.show()
```

شکل خ. الگوریتم CTAEA

و pareto front در شکل زیر آمده است.

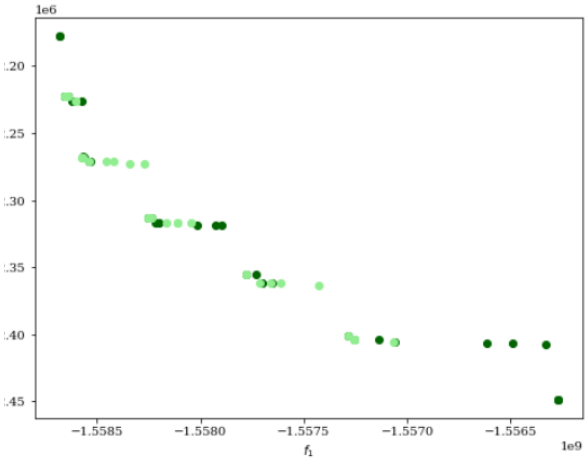


شکل د. نمودار pareto front برای CTAEA

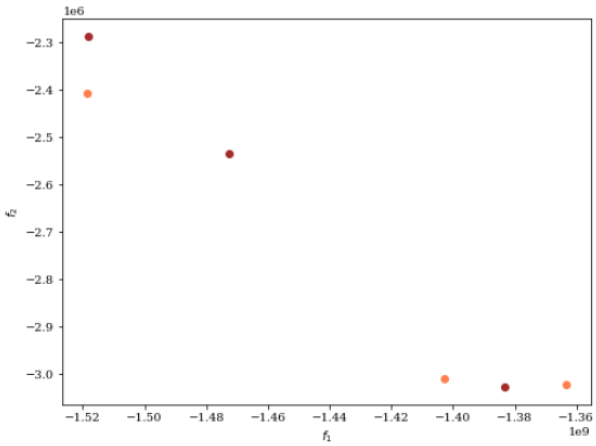
(محور عمودی f2 و محور افقی f1)

٤. بررسی تأثیر Reference Points

در این پژوهش ما دو نوع Refrence point را مورد آزمایش و بررسی قرار دادیم. یکی به صورت دست نویس و یکی هم به صورت انرژی.(نقاط سبز روشن متعلق به روش انرژی هستند و نقاط سبز تیره متعلق به روش دست نویس هستند.) که با توجه به شکل های زیر می بینیم این دو روش تفاوت چندانی با هم نداشته اند. در الگوریتم CTAEA ، ما Reference point ها را یک مرتبه ۱۰۰ و ۱۰۰ و یک بار هم به صورت ۳ و ۳ در نظر گرفتیم که نقاط نارنجی (دست نویس) عملکرد بهتری نسبت به انرژی داشتند. در الگوریتم RNSGA-III جفت pareto front ها روی هم افتاده است و تفاوتی نداشته اند. در این الگوریتم تعداد R.P را ۱۰ در نظر گرفته ایم که متناسب با انتخاب از جمعیت است.(که به علت زمان اجرای بالا الگوریتم RNSGA-III ما تعداد R.P را برای این الگوریتم ۱۰ و برای سایر الگوریتم ها در محدود انتخابی ۹۰ تا ۱۰۰ در نظر گرفتیم) که در مجموع روش انرژی عملکرد بهتری را به ما نشان داد.



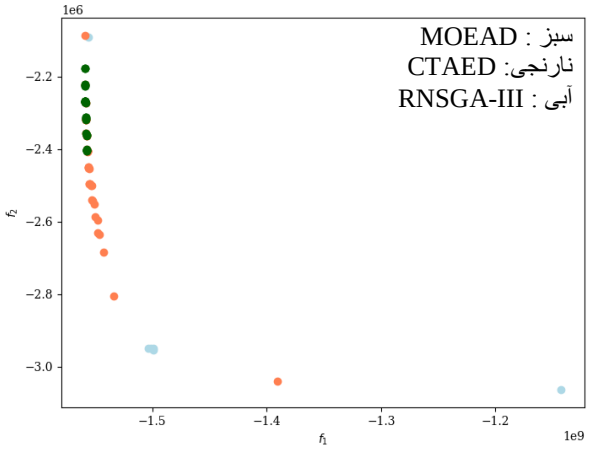
شکل ر. مقایسه pareto front در دو روش انرژی و دست نویس به ازای ۱۰۰ و ۱۰۰ (سبز روشن روش انرژی و سبز تیره روش دست نویس)



شکل ز. مقایسه pareto front در دو روش انرژی و دست نویس به ازای ۳ و ۳ (نارنجی روش دست نویس و قرمز روش انرژی)

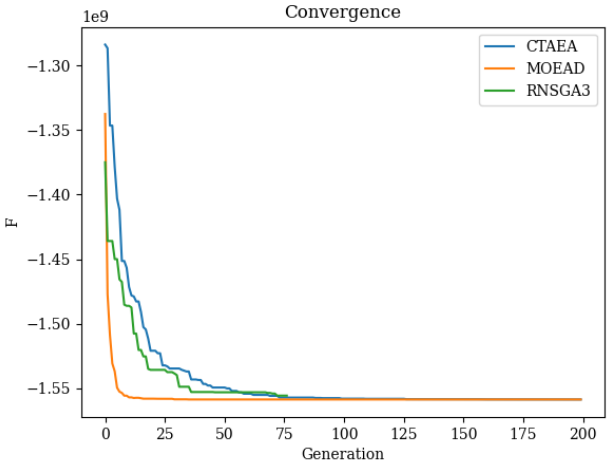
۵. نتیجه گیری

در این قسمت با مقایسه pareto front الگوریتم‌ها متوجه می‌توان شد که تمامی الگوریتم‌ها تقریبا pareto front شبیه به هم داشتند. (شکل د)



شکل س. مقایسه pareto front الگوریتم‌ها.
(محور عمودی f2 و محور افقی f1)

از لحاظ همگرایی الگوریتم‌ها بصورت شکل زیر می‌باشند. که همگرایی MOEAD از همه بهتر است.



شکل ش. مقایسه همگرایی الگوریتم‌ها. محور افقی نمایانگر تعداد نسل‌ها و نمودار عمودی نمایانگر هدف است.