حل مسئله چند هدفه

ستاره روشن

چکیده — در این گزارش به بررسی عملکرد چهار الگوریتم تکاملی بر روی مسئله یافتن ترتیب کودها با ماکزیمم دسترس پذیری همچنین ماکزیمم بهروری.

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.

```
 = 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 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 f_1 و f_2 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 f_1 و f_2 درنظر گرفته بشود (خط۱۱ و ۱۲). در آخر f_1 (دسترس پذیری) و f_2 (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

این قسمت بسیار ساده و صریح است. تنها نیاز است الگوریتم فراخوانی و import شود (شکل ت). با الگوریتم NSGA۲ آغاز میکنیم نیاز است از Myproblem در بخش قبل یک شی ساخته شود (خط ۱). در خط دوم مقادیری به الگوریتم داده به ترتیب، اندازه جمعیت (برابر ۱۰۰)، نوع representation (در اینجا permutation که اول بصورت رندوم)، سپس crossover (در اینجا تغییر order)، سپس جهش (در این جا برعکس کردن).

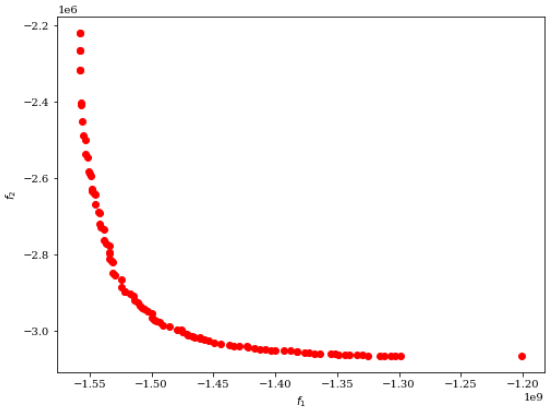
جدول أ. پارامترهای مورد استفاده

Representation	Permutation
crossover	Order
mutation	Inverse
Population size	۱۰۰
Initial population	Random
Termination	۵۰۰ generation

```python
problem = MyProblem()
algorithm = NSGA2(pop_size=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", 500), seed = 1, verbose = False)
plot = Scatter()
plot.add(problem.pareto_front(), plot_type="line", color="black", alpha=0.7)
plot.add(res.F, color="red")
plot.show()
```

شکل ت. فراخوانی الگوریتم NSGA۲

نتایج در خط ٦ ذخیره می‌شود. خطوط ۷، ۸، ۹ و ۱۰ نمودار pareto front را برای ما رسم خواهند کرد (شکل ث).

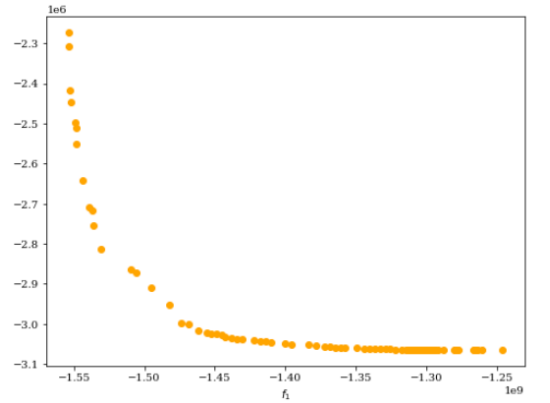


شکل ث. نمودار pareto front برای NSGA۲ (محور عمودی بهروری و محور افقی دسترس پذیری)

حال به بررسی الگوریتم RSNGA۲ می‌پردازیم. پارامترها شبیه به جدول أ، استفاده خواهد شد تنها اینجا ref points با مقادیر قابل مشاهده در خط ۳ و همچنین مقدار اپسیلون به پیشنهاد خود مقاله ۰.۰۱ انتخاب شده. نتیجه pareto front در شکل ح مشاهده می‌کنید.

```python
from pymoo.algorithms.rnsga2 import RNSGA2

ref_energy = get_reference_directions("energy", 2, 100, seed=1)
algorithm_energy = RNSGA2(pop_size=100, sampling=get_sampling("perm_random"),
                  crossover=get_crossover("perm_ox"),
                  mutation=get_mutation("perm_inv"),
                  eliminate_duplicates=True,
                  ref_points = ref_energy, epsilon=0.01,
                  normalization='front')
res_RNSGA2_energy = minimize (problem, algorithm_energy, Termination = ("n_gen", 200),
                              seed = 1, verbose = False, save_history=True)
plot = Scatter()
plot.add(problem.pareto_front(), plot_type="line", color="black", alpha=0.7)
plot.add(res_RNSGA2_energy.F, color="Orange")
plot.show()
```

شکل ج. الگوریتم RNSGA۲



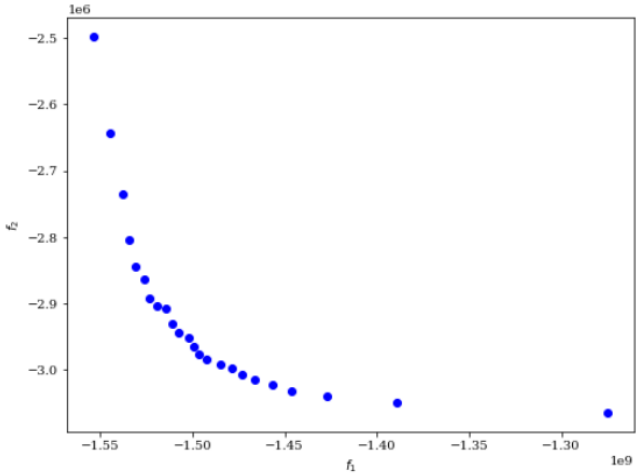
شکل ح. نمودار pareto front برای RSNGA۲ (محور عمودی بهروری و محور افقی دسترس پذیری)

الگوریتم بعدی NSGA۳ است. اینجا باید به ۱۰۰ reference direction داشته باشیم (خط ۲) و بقیه پارامترها مشابه جدول أ، است.

```python
from pymoo.algorithms.nsga3 import NSGA3
ref_energy = get_reference_directions("energy", 2, 100, seed=1)
algorithm_energy = NSGA3(pop_size=100, sampling=get_sampling("perm_random"),
                  crossover=get_crossover("perm_ox"),
                  mutation=get_mutation("perm_inv"),
                  eliminate_duplicates=True,
                  ref_dirs=ref_energy,
                  normalization='front')
res_NSGA3_energy = minimize (problem, algorithm_energy, Termination = ("n_gen", 200),
                             seed = 1, verbose = False, save_history=True)
plot = Scatter()
plot.add(problem.pareto_front(), plot_type="line", color="black", alpha=0.7)
plot.add(res_NSGA3_energy.F, color="lightblue")
plot.show()
```

شکل خ. فراخوانی الگوریتم NSGA۳

در شکل زیر pareto front این الگوریتم دیده می‌شود.



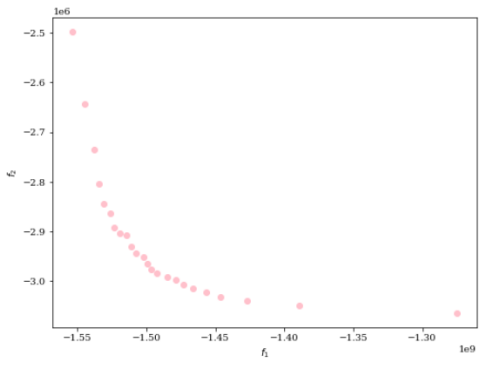
شکل د. نمودار pareto front برای NSGA۳ (محور عمودی بهروری و محور افقی دسترس پذیری)

در آخر به الگوریتم UNSGA۳ می‌پردازیم. پارامترها شبیه به جدول أ، استفاده شده‌اند. و reference direction نیز در خط ۲ تعریف شده است.

```python
from pymoo.algorithms.unsga3 import UNSGA3
ref_energy = get_reference_directions("energy", 2, 100, seed=1)
algorithm = UNSGA3(pop_size=100, sampling=get_sampling("perm_random"),
                  crossover=get_crossover("perm_ox"),
                  mutation=get_mutation("perm_inv"),
                  eliminate_duplicates=True,
                  ref_dirs=ref_energy,
                  normalization='front')
res_UNSGA3_energy = minimize (problem, algorithm, Termination = ("n_gen", 200),
                              seed = 1, verbose = False, save_history=True)
plot = Scatter()
plot.add(problem.pareto_front(), plot_type="line", color="black", alpha=0.7)
plot.add(res_UNSGA3_energy.F, color="pink")
plot.show()
```

شکل ذ. الگوریتم UNSGA۳

و pareto front در شکل زیر آمده است.

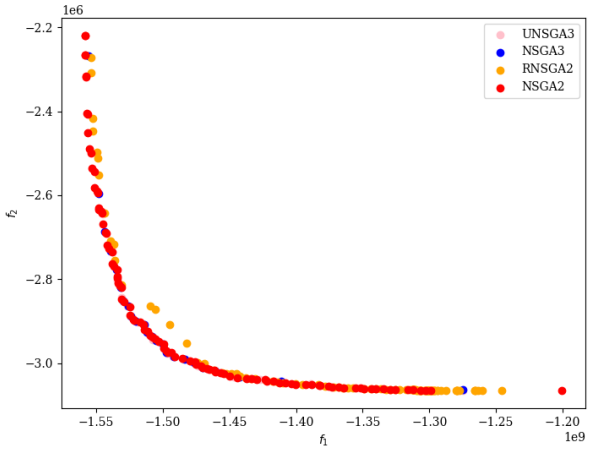


شکل ر. نمودار pareto front برای RSNGA۲ (محور عمودی f۲ و محور افقی f۱)

لازم به ذکر است در قسمت ۵پیوست مقایسه reference pointها آمده است.

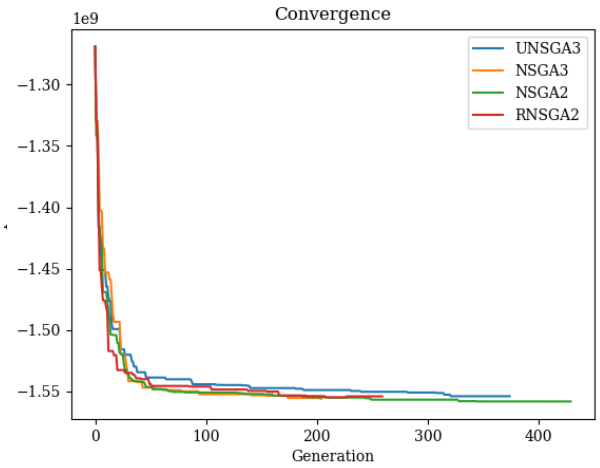
۴. نتیجه‌گیری

در این قسمت با مقایسه pareto front الگوریتم‌ها متوجه می‌توان شد که تمامی الگوریتم‌ها تقریبا یک pareto front داشتند بطور کل RNSGA۲ عملکرد بدتری داشت.



شکل ز. مقایسه pareto front الگوریتم‌ها. (محور عمودی f۲ و محور افقی f۱ همچنین رنگ نتایج مربوط به هر الگوریتم در راهنمای بالای شکل مشخص شده است.)

همچنین از لحاظ همگرایی الگوریتم‌ها بصورت شکل زیر می‌باشند. که همگرایی RNSGA۲ از همه بهتر است.



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

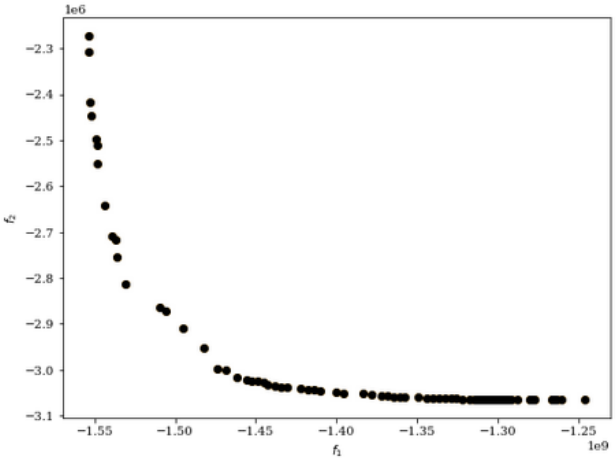
۵. پیوست

در این بخش به بررسی reference point ها می‌پردازیم. از دو نوع reference point استفاده می‌کنیم یکی das-dennis و دیگری energy که دومی را خود کتابخانه pymoo توصیه کرده بود در تمامی نتایج در قسمت‌های قبل از energy استفاده شده است.

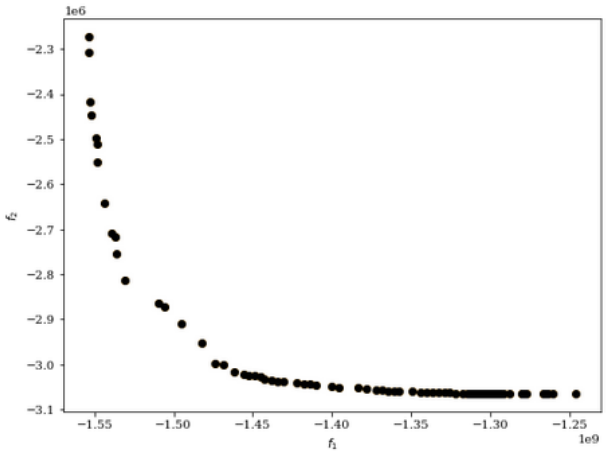
```python
from pymoo.algorithms.rnsga2 import RNSGA2
from pymoo.factory import get_visualization, get_reference_directions
from pymoo.util.ref_dirs.energy import RieszEnergyReferenceDirectionFactory

ref_dennis = get_reference_directions("das-dennis", 2, n_points=5, seed=1)
ref_energy = get_reference_directions("energy", 2, 8, seed=1)

algorithm_energy = RNSGA2(pop_size=100, sampling=get_sampling("perm_random"),
                  crossover=get_crossover("perm_ox"),
                  mutation=get_mutation("perm_inv"),
                  eliminate_duplicates=True,
                  ref_points = ref_energy, epsilon=0.01,
                  normalization='front')
res_RNSGA2_energy = minimize (problem, algorithm_energy, Termination = ("n_gen", 200), seed = 1, verbose = F
algorithm_dennis = RNSGA2(pop_size=100, sampling=get_sampling("perm_random"),
                  crossover=get_crossover("perm_ox"),
                  mutation=get_mutation("perm_inv"),
                  eliminate_duplicates=True,
                  ref_points = ref_dennis, epsilon=0.01,
                  normalization='front')


res_RNSGA2_dennis = minimize (problem, algorithm_dennis, Termination = ("n_gen", 200), seed = 1, verbose = F

plot = Scatter()
plot.add(problem.pareto_front(), plot_type="line", color="black", alpha=0.7)
plot.add(res_RNSGA2_energy.F, color="Orange")
plot.add(res_RNSGA2_dennis.F, color="black")
plot.show()
```

شکل ش. به عنوان نمونه نحوه result گرفتن برای مقایسه به این صورت است از هردو روش به تفکیک نتیجه گرفته شده و روی یک نمودار رسم میکنیم.

در ادامه مشاهده می‌شود که برای energy رنگ نارنجی و برای dennis رنگ مشکی انتخاب شده است و چون رنگ مشکی بعدتر رسم شده و دقیقا نقاط یکی هستند بنابراین در شکل تنها مشکی قابل مشاهده است. الگوریتم RNSGA۲ تفاوتی میان این دو روش نیست.



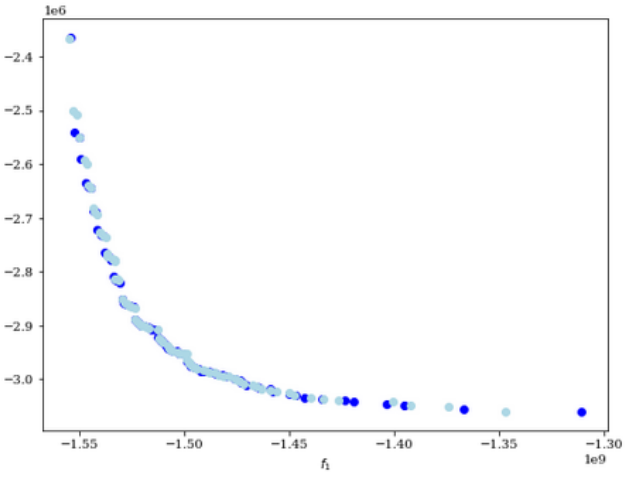
شکل ص. pareto front برای RNSGA۲ که با ۵ پارتیشن برای dennis و ۸ برای energy. و محورهای افقی و عمودی نیز به ترتیب f۲و f۱ را نشان می‌دهند.



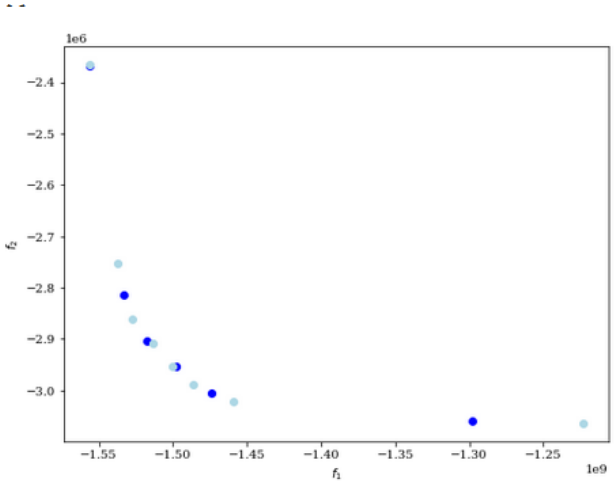
شکل ض. pareto front برای RNSGA۲ که با ۹۰ پارتیشن برای dennis و ۹۰ برای energy. و محورهای افقی و عمودی نیز به ترتیب f۲و f۱ را نشان می‌دهند.

با توجه به اشکال بالا نه تعداد پارتیشن ها و نه نوع reference point گرفتن روی pareto front تاثیر نداشته است.

در NSGA۳ اما با کم شدن تعداد پارتیشن تعداد نقاط کم می‌شد. می‌توان گفت تعداد نقاط بیشتر بهتر است چون set جوابهای قابل قبول ما بیشتر است اما دو روش تفاوتی چشمگیری باهم ندارند.

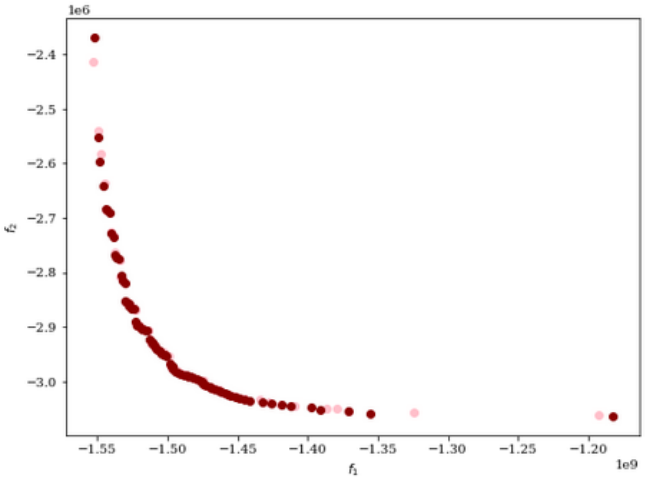


شکل ط. pareto front برای NSGA۳ که با ۹۰ پارتیشن برای dennis و ۹۰ برای energy. و محورهای افقی و عمودی نیز به ترتیب f۲و f۱ را نشان می‌دهند. آبی پر رنگ نشان دهنده dennis و کم رنگ نشان دهنده energy است.

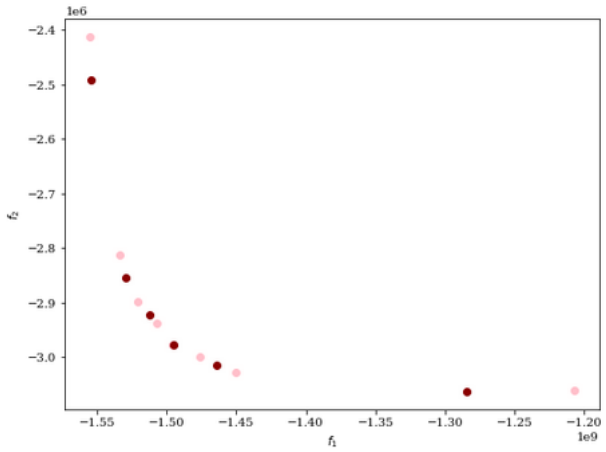


شکل ظ. pareto front برای NSGA۳ که با ۵ پارتیشن برای dennis و ۸ برای energy. و محورهای افقی و عمودی نیز به ترتیب f۲و f۱ را نشان می‌دهند. آبی پر رنگ نشان دهنده dennis و کم رنگ نشان دهنده energy است.

در UNSGA۳ نیز به همین منوال است. یعنی تعداد اعضای pareto front ما وابسته به تعداد پارتیشن‌ها است و اما دو روش dennis و energy تفاوت چشمگیری با یکدیگر ندارند. اما در اینجا به نظر کمی dennis که با رنگ قهوه‌ای نمایش داده شده بهتر است.



شکل ع. pareto front برای UNSGA۳ که با ۹۰ پارتیشن برای dennis و ۹۰ برای energy. و محورهای افقی و عمودی نیز به ترتیب f۲و f۱ را نشان می‌دهند. قهوه‌ای نشان دهنده dennis و صورتی نشان دهنده energy است.



شکل غ. pareto front برای UNSGA۳ که با ۵ پارتیشن برای dennis و ۸ برای energy. و محورهای افقی و عمودی نیز به ترتیب f۲و f۱ را نشان می‌دهند. قهوه‌ای نشان دهنده dennis و صورتی نشان دهنده energy است.

در حالت کلی تفاوت چشمگیری بین دو روش مشاهده نمی‌شد. در نتایج همانطور که قبلا ذکر شده است از روش energy و با تعداد نقاط ۹۰ استفاده شده است در فضای ۲ بعدی.