一、参考论文：

R. L. Pinheiro, D. Landa-Silva, and J. Atkin, "**A technique based on trade-off maps to visualise and analyse relationships between objectives in optimisation problems**," Journal of Multi-Criteria Decision Analysis, vol. 24, no. 1-2, pp. 37–56, 2017.

这篇论文提出分析和可视化 MOPs 中各个目标之间关系的四步分析法。
前提需要：得到该多目标问题的近似 Pareto optimal set（注意是归一化之后的 PF）
四步分析的目的：分析问题算例的子集，从而可以为相同问题的其他算例设计更有效的定制算法。以下第二部分介绍具体分析思路，第三部分介绍 Python 代码的相关分析及画图。

二、四步分析法：

第一步：全局成对关系分析

使用 Kendall correlation coefficients 进行全局成对关系分析。

这个值的计算方法如下（在论文的第 3 节有计算方法）：
假设我们的 Pareto optimal set 里面有μ个解，则分析目标i和目标j关系的做法是：计算得到 concordant 对数为μ_c，得到 discordant 对数为μ_d，则 values 等于：

$$\tau = \frac{\mu_c - \mu_d}{\frac{1}{2}\mu(\mu - 1)}$$

values < -0.5 表示 trade-off surface exists（conflicting）
values > 0.5 表示 strongly harmonious correlations，则目标可以合成
其余的 values 值表示两个目标是 independent，但仅表示目标不是全局 dependent，并不意味着没有局部 trade-offs

第二步：目标值范围分析

目标值范围比较大的目标是有意义的目标；
目标值范围比较小的目标是无意义的目标，因为如果一个目标值范围小，则其他目标的改变基本不会影响到这个目标。

解决没有意义的目标，有两种方法：
1. 直接忽略这个目标
2. Cluster the objectives

通过这一步，可以分析得到目标值范围大的目标和目标值范围小的目标。

第三步：Trade-off 区域的分析

Trade-off 区域分析类似于画卡诺图，卡诺图是使用真值表来可视化和简化布尔代数表达式的方法，i个变量有2^i个格子。

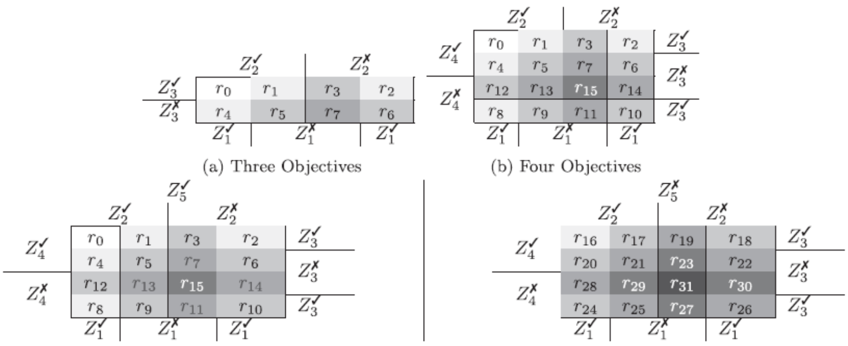
对于每一个目标（i = 1, 2, …m），都定义一个阈值t_i。阈值可以是平均数等。

tion (Z_1 = ✓, Z_2 = ✗, Z_3 = ✓) in a three-objective scenario indicates that for Z_1 and Z_3 the objective values are better than their respective thresholds, while Z_2 presents a value, which is worse than its threshold.

根据阈值，画出 region map，与卡诺图类似：

(replacing 0's and 1's by ✓ and ✗).

这个 region map 会有2^m个区域，每个区域使用 Gray code 编码。较浅的区域比较深的区域有更多目标值表现得好。



(c) Five Objectives

这里面涉及到一个问题，每个目标的阈值怎么分析？具体看论文 Threshold analysis 的描述，主要思想是确定每个目标的最大值和最小值，平均划分为α份，确定（α-1）个值，选一个合适的作为阈值。这里合适的阈值指的是在该阈值下恰好没有一个解属于r_0区域。

第四步：多目标散点图分析

所有目标值都要进行归一化，以选定目标作为 x 轴，剩下的目标作为 y 轴，画散点图。作为 x 轴的目标一般是目标范围比较大的目标。

三、分析实例：

依据论文的描述，我用 Python 实现了这个四步分析方法。所有代码以及输入（数据）输出位于 VARO-master 文件夹下。

要使用相关代码，最好先配置 Anaconda+Jupyter Notebook 环境。Windows 下参考文章：

https://blog.csdn.net/Xiao13Yu14/article/details/82192575

配置好环境之后打开 main.ipynb，进行分析。

以下的分析以 Multiobjective Multiple Neighborhood Search Algorithms for Multiobjective Fleet Size and Mix Location-Routing Problem With Time Windows 这篇文章的数据为例，描述如何得到这篇文章 supplementle file 中的图。

分析的前提需要：
1. 把所有的算例的 pf 结果文件放在 VARO-master/result 文件夹下。
2. VARO-master 下有 kendall 文件夹，用于存放 kendall 分析结果。
3. VARO-master 下有 range 文件夹，用于存放目标值范围分析的结果。
3. VARO-master 下有 threshold 文件夹，用于存放阈值分析的结果。

第一步：全局成对关系分析

1. 修改 pf 文件所在的路径为你的 pf 文件对应的存放路径：

```
In  [10]: # 第一步：全局成对关系分析（分析kendall相关系数）
          # instance_name表示pf文件所在路径
          instance_name = []
          instance_name.append('result/as/MOEAD/MOEAD_GLS/test50-0-0-0-0.d0.tw0.D4/PF/pf')
```

2. kendall 分析的结果会存放在 VARO-master/kendall 文件夹下，为了得到 csv 文件的名字，需要根据你的存放路径修改以下内容：

```
class KCC:
    def __init__(self, instance_name, symmetric):
        self.instance_name = instance_name
        self.symmetric = symmetric
        self.pf_file_name = instance_name
        # Read the pf file
        self.pf = np.loadtxt(self.pf_file_name)
        print (self.pf.shape)
        self.SOL_NUM = self.pf.shape[0]
        self.OBJ_NUM = self.pf.shape[1]
        self.nor_pf_file_name = instance_name
        self.nor_pf = np.loadtxt(self.nor_pf_file_name)
        ##################################
        # 根据对称还是非对称，提取算例名
        if self.symmetric is False:
            self.csv_name = self.instance_name.replace('result/as/MOEAD/MOEAD_GLS/', '').replace('/PF/pf', '')
        else:
            self.csv_name = self.instance_name.replace('result/s/MOEAD/MOEAD_GLS/', '').replace('/PF/pf', '')
        ##################################
```

这个实现的结果是，以下算例输出的 kendall 相关的 csv 文件名为 **RC101_100-kendall.csv**

```
instance_name.append('result/s/MOEAD/MOEAD_GLS/RC101_100/PF/pf')
```

3. 运行以下代码段即可得到结果。

```python
for i in range(86):
    if i < 30: # 一共有30个非对称算例
        kcc = KCC(instance_name[i], False)
    else:
        kcc = KCC(instance_name[i], True)
    kcc.analyze_KCC_2()

# kendall文件下会输出算例名-kendall.csv文件
```

3. 打开 **VARO-master/kendall/RC101_100-kendall.csv**，obj 表示哪两个目标，res 列表示 kendall 相关性的值。

	A	B	C	D	E
1	obj	res	zero	up	down
2	\$f_{1}-f_{2}\$	-0.04032269	0	0.5	-0.5
3	\$f_{1}-f_{3}\$	-0.160648825	0	0.5	-0.5
4	\$f_{1}-f_{4}\$	-0.128643088	0	0.5	-0.5
5	\$f_{1}-f_{5}\$	0.063043891	0	0.5	-0.5
6	\$f_{1}-f_{6}\$	-0.085214062	0	0.5	-0.5
7	\$f_{2}-f_{3}\$	-0.328712349	0	0.5	-0.5
8	\$f_{2}-f_{4}\$	-0.260208472	0	0.5	-0.5
9	\$f_{2}-f_{5}\$	-0.084893025	0	0.5	-0.5
10	\$f_{2}-f_{6}\$	-0.0208792	0	0.5	-0.5
11	\$f_{3}-f_{4}\$	0.496809364	0	0.5	-0.5
12	\$f_{3}-f_{5}\$	-0.425685258	0	0.5	-0.5
13	\$f_{3}-f_{6}\$	-0.052607271	0	0.5	-0.5
14	\$f_{4}-f_{5}\$	-0.366844857	0	0.5	-0.5
15	\$f_{4}-f_{6}\$	-0.013397345	0	0.5	-0.5
16	\$f_{5}-f_{6}\$	-0.026203702	0	0.5	-0.5

4. 运行以下代码段可以得到分析图，这个分析图位于 **VARO-master/Step-1-All.eps**

```python
# Step 1: kendall画图
num = 4
instance_name = []
##################################
# 以下instance_name为kendall.csv文件
instance_name.append('kendall/test150-0-0-0-0.d0.tw4.D6-kendall.csv')
instance_name.append('kendall/test250-0-0-0-0.d0.tw4.D6-kendall.csv')
instance_name.append('kendall/C204_100-kendall.csv')
instance_name.append('kendall/R101_100-kendall.csv')
##################################
linestyle_list = ['-', '--', '-.', ':']
marker_list = ['^', '^', 's', 's']
l_list = []
##################################
# 以下为图上的图例名
labels_list = ['150-4-6', '250-4-6', 'C204', 'R101']
##################################
color_list = ['b', 'g', 'r', 'y', 'm', 'c', 'k', 'lime', 'orange', 'greenyellow', 'darkmagenta', 'moccasin']
plt.figure(figsize=(8, 4))
for i in range(num):
    data = pd.read_csv(instance_name[i])
    x = data.iloc[:, 0]
    y = data.iloc[:, 1]
    l, = plt.plot(x, y, color_list[i], linestyle=linestyle_list[i%4], marker = marker_list[i%4], linewidth=1)
    l_list.append(l)
    zero = data.iloc[:, 2]
    up = data.iloc[:, 3]
    down = data.iloc[:, 4]
    l, = plt.plot(x, zero, 'black', linewidth=1)
    lup = plt.plot(x, up, 'lightgray', linewidth=1)
    ldown = plt.plot(x, down, 'lightgray', linewidth=1)
plt.ylim(-1, 1)
plt.xlim('$f_{1}-f_{2}$', '$f_{5}-f_{6}$')
plt.xticks(rotation=45)
plt.legend(handles=l_list, labels=labels_list, loc='best')
plt.savefig('Step-1-All.eps', format='eps', dpi=1000)
plt.show()
```

5. **VARO-master/Step-1-All.eps** 如下：

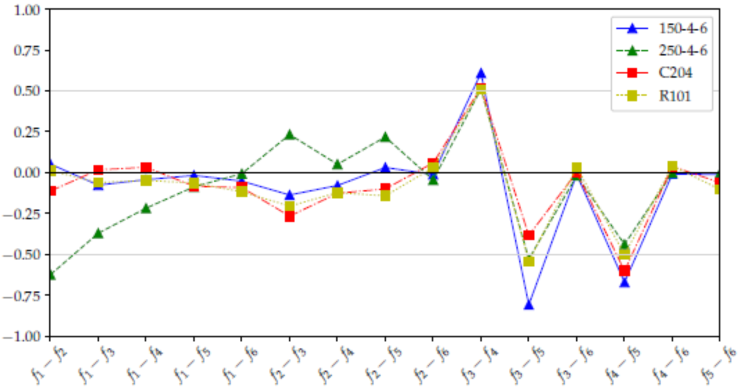


Fig. S1. Pairwise correlation values (y-axis) for each pair of objectives (x-axis) for four selected instances. The results for each instance are shown in different colors and linestyles. The square marker represents real-world instances and the triangle marker represents traditional instances. It shows the global pairwise relationships using the Kendall correlation method. Three pairs of objectives ($f_3 - f_4$, $f_3 - f_5$, $f_4 - f_5$) have either high harmonious (value > 0.5) or conflicting (value < -0.5) relationships. These strong relationships between objectives indicate that both real-world instances and traditional instances provide interesting multiobjective challenges. Basically, real world instances show stronger dependency relationships than traditional instances according to the pairwise correlation values.

第二步：目标值范围分析

1. 运行以下代码段可以得到结果。

```
# 第二步：目标值范围分析
# 要求这里的pf是没有归一化的
# range文件夹下输出-rang.csv文件
ins = 'result/as/MOEAD/MOEAD_GLS/test150-0-0-0-0.d0.tw4.D6/PF/pf'
kcc = KCC(ins, False)
kcc.analyze_obj_range()
ins = 'result/as/MOEAD/MOEAD_GLS/test250-0-0-0-0.d0.tw4.D6/PF/pf'
kcc = KCC(ins, False)
kcc.analyze_obj_range()
ins = 'result/s/MOEAD/MOEAD_GLS/C204_100/PF/pf'
kcc = KCC(ins, True)
kcc.analyze_obj_range()
ins = 'result/s/MOEAD/MOEAD_GLS/R101_100/PF/pf'
kcc = KCC(ins, True)
kcc.analyze_obj_range()
```

2. 打开 **VARO-master/range/R101_100-range.csv** 可以得到每个目标值的范围，min 表示最小值，max 表示最大值，average 表示平均值。（应该是不归一化的结果，以下仅供参考）

	A	B	C	D
1	obj	min	max	average
2	f_1	0	1	0.198355
3	f_2	0	1	0.399031
4	f_3	0	1	0.345884
5	f_4	0	1	0.38408
6	f_5	0	1	0.339906
7	f_6	0	1	0.124441

3. 修改相应的算例名称，可以得到这个算例的目标范围分析图 **VARO-master/ test150-0-0-0-0.d0.tw4.D6-step2.eps**：

```
# Step 2: Plot
#################################
# 需要画哪个算例就改instance_name为算例名
instance_name = 'test150-0-0-0-0.d0.tw4.D6'
#################################
data = pd.read_csv('range/' + instance_name + '-range.csv')
x = data.iloc[:, 0]
OBJ_NUM = data.shape[0]
average_pct = data.iloc[:, 3] / data.iloc[:, 2] * 100
min_pct = data.iloc[:, 1] / data.iloc[:, 2] * 100
max_pct = data.iloc[:, 2] / data.iloc[:, 2] * 100
print (data.iloc[:, 1])
fmt = '%.2f\%%' # %需要转义符
yticks = mtick.FormatStrFormatter(fmt)
plt.gca().yaxis.set_major_formatter(yticks)
plt.plot(x, average_pct, 'bo')
for i in range(OBJ_NUM):
    plt.plot([x[i], x[i]], [min_pct[i], max_pct[i]])
plt.title(instance_name)
# plt.savefig(instance_name + '-step2.png')
plt.savefig(instance_name + '-step2.eps', format='eps', dpi=1000)
plt.show()
```

4. 相关分析图如下：

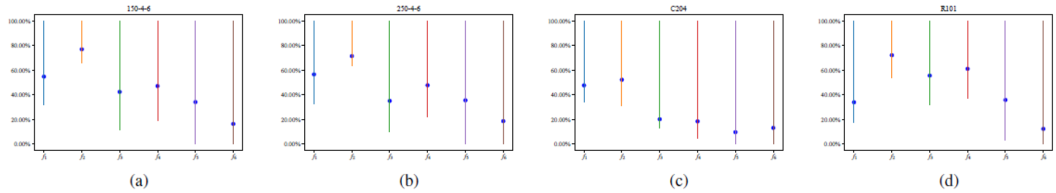


Fig. S2. Results for the objective ranges analysis for four selected instances. (a) Instance 150-4-6. (b) Instance 250-4-6. (c) Instance C204. (d) Instance R101. The y-axis presents the minimum, maximum and average value of each objective as a percentage of the overall maximum value found for the respective objective. Longer lines indicate larger ranges. Almost all objectives (except f_2) have large ranges (over 60%). It indicates that the selected four instances have conflicting objectives. Although there are solutions with good values for a given objective, at least one other objective has a poor value.

第三步：Trade-off 区域的分析

阈值分析

1. 运行以下代码段进行阈值分析：（注意在使用之前按照 region 的格子个数，修改 region 函数的内容）

```
for i in range(self.SOL_NUM):
    my_code = ''
    for j in range(self.OBJ_NUM):
        if self.nor_pf[i][j] <= threshold: # better: 0
            my_code += '0'
        else:
            my_code += '1' # worse
    region_dict[my_code] = region_dict[my_code] + 1
sol_num = []
for i in range(len(region_list)):
    sol_num.append(region_dict[region_list[i]] / self.SOL_NUM)
region_list = np.reshape(region_list, (8, 8))
sol_num = np.reshape(sol_num, (8, 8))
for i in range(8):
    if i % 2 == 1:
        region_list[i] = region_list[i][::-1]
        sol_num[i] = sol_num[i][::-1]
# print (region_list)
# print (sol_num)
data = pd.DataFrame(sol_num)
data.to_csv(self.instance_name + '-' + str(threshold) + '-3.csv', inde
gray_list = []
for i in range(8):
    for j in range(8):
        count = 0
        for k in range(self.OBJ_NUM):
            if region_list[i][j][k] == '0':
                count = count + 1
        gray_list.append(count)
gray_list = np.reshape(gray_list, (8, 8))
data = pd.DataFrame(gray_list)
data.to_csv('region_map.csv', header=False, index=False)
return region_dict['000000']
```

修改完之后运行代码：

```
for i in range(86):
    if i < 30:
        kcc = KCC(instance_name[i], False)
    else:
        kcc = KCC(instance_name[i], True)
    kcc.analyze_threshold()
```

2. 再运行以下代码段得到阈值图：（要注意根据具体情况修改这两个数值，total_instance_num 表示总的算例个数，as_num 表示非对称算例个数）

```
# Step 3: Threshold analysis
# 待选阈值的list
x = [0.0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.

threshold_len = len(x)
as_list = [0] * threshold_len
s_list = [0] * threshold_len
#################################
total_instance_num = 86
as_num = 30
################
for i in range(total_instance_num):
    if i < as_num:
        csv_name = instance_name[i].replace('result/as/MOEAD/MOEAD_GLS/', '').
        data = pd.read_csv('threshold/' + csv_name + '-threshold.csv')
        y = data.iloc[:, 1]
        for j in range(threshold_len):
            if y[j] != 0:
                as_list[j] = as_list[j] + 1
    else:
        csv_name = instance_name[i].replace('result/s/MOEAD/MOEAD_GLS/', '').r
        data = pd.read_csv('threshold/' + csv_name + '-threshold.csv')
        y = data.iloc[:, 1]
        for j in range(threshold_len):
            if y[j] != 0:
                s_list[j] = s_list[j] + 1
print (as_list)
print (s_list)
l1, = plt.plot(x, as_list, 'b-', linestyle='-', linewidth=1)
l2, = plt.plot(x, s_list, 'g-', linestyle='--', linewidth=1)
plt.xlim(0, 1)
plt.ylim(0, 60)
plt.ylabel('\# Of Instances')
plt.xlabel('Threshold')
plt.legend(handles=[l1, l2], labels=['real-world instances', 'traditional inst
# plt.savefig('step3.png')
plt.savefig('step3.eps', format='eps', dpi=1000)
plt.show()
```

3. 得到阈值图 **VARO-master/ step3.eps**，根据这个图得到 real-world instances 的阈值是 0.1，traditional instances 的阈值是 0.05。

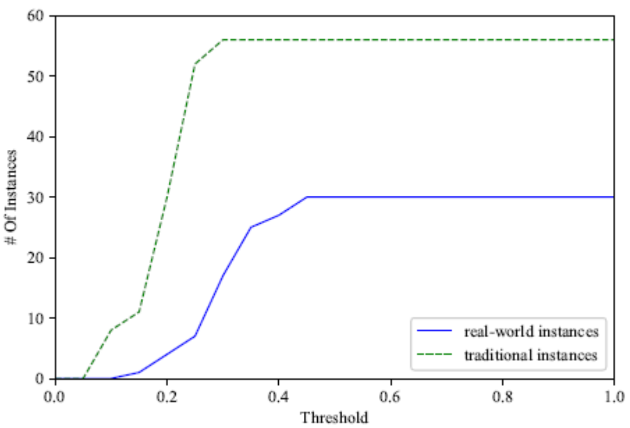


Fig. S3. Threshold analysis for 30 real-world instances and 56 traditional instances, which shows the number of instances with solutions in r_0 when the threshold increases (normalized values). Region r_0 represents solutions with good values in all objectives. There are no solutions with good values for all objectives.

画 Trade-off 图

1. 以下分析针对 6 个目标的问题来写，如目标数目不是 6，要对代码进行修改，具体为圈出的红色部分。

```python
def region(self, threshold):
    a = GrayCode()
    region_list = a.getGray(6)
    region_dict = {}
    for i in range(len(region_list)):
        region_dict[region_list[i]] = 0
    for i in range(self.SOL_NUM):
        my_code = ''
        for j in range(self.OBJ_NUM):
            if self.nor_pf[i][j] <= threshold: # better: 0
                my_code += '0'
            else:
                my_code += '1' # worse
        region_dict[my_code] = region_dict[my_code] + 1
    sol_num = []
    for i in range(len(region_list)):
        sol_num.append(region_dict[region_list[i]] / self.SOL_NUM)
    region_list = np.reshape(region_list, (8, 8))
    sol_num = np.reshape(sol_num, (8, 8))
    for i in range(8):
        if i % 2 == 1:
            region_list[i] = region_list[i][::-1]
            sol_num[i] = sol_num[i][::-1]
    # print (region_list)
    # print (sol_num)
    data = pd.DataFrame(sol_num)
    data.to_csv(self.instance_name + '-' + str(threshold) + '-3.csv', index=False)
    gray_list = []
    for i in range(8):
        for j in range(8):
            count = 0
            for k in range(6):
                if region_list[i][j][k] == '0':
                    count = count + 1
            gray_list.append(count)
    gray_list = np.reshape(gray_list, (8, 8))
    data = pd.DataFrame(gray_list)
    data.to_csv('region_map.csv', header=False, index=False)
    return region_dict['000000']
```

2. 目标为 6 的 trade-off 图应该是 8 行 8 列共 64 个格子的，因为$2^6 = 64$

0.00%	0.00%	0.01%	0.01%	0.07%	0.06%	0.00%	0.00%
0.05%	0.00%	0.01%	0.02%	2.37%	1.90%	0.12%	0.87%
1.19%	0.00%	0.38%	0.46%	13.79%	9.63%	1.75%	7.78%
0.60%	0.00%	0.62%	0.76%	1.62%	1.41%	0.01%	1.10%
3.89%	0.00%	1.49%	1.72%	2.30%	1.95%	0.06%	2.76%
1.93%	0.00%	0.34%	0.48%	9.34%	6.16%	1.55%	3.98%
0.19%	0.00%	0.10%	0.13%	6.37%	5.20%	0.68%	0.88%
0.03%	0.00%	0.28%	0.27%	0.66%	0.66%	0.00%	0.04%

3. 如果之前的几步都做好了，会发现 **VARO-master/region_map.csv** 这个文件。每个格子的数字表示有几个目标比阈值好。

	A	B	C	D	E	F	G	H
1	6	5	4	5	4	3	4	5
2	5	4	3	4	3	2	3	4
3	4	3	2	3	2	1	2	3
4	5	4	3	4	3	2	3	4
5	4	3	2	3	2	1	2	3
6	3	2	1	2	1	0	1	2
7	4	3	2	3	2	1	2	3
8	5	4	3	4	3	2	3	4

4. 填充颜色：6 个目标应该有 7 种不同的灰色，把灰色等分，填充上去，其中数字 6 的格子应该是白色，数字 0 的格子应该是最深色。
5. 加标号：在填充完颜色的图的周围加标号，表示某个目标是好于阈值还是差于阈值，将这个图存为 region.xlsx
6. 运行代码段，生成 distribution map 和 frequency map

```
as_num = 30
#################
for i in range(1, as_num):
    print (instance_name[i])
    data = pd.read_csv(instance_name[i])
    data_sum = data_sum + data
data_sum = data_sum / as_num
print (data_sum)
data = pd.DataFrame(data_sum)
data.to_csv('as-distribution.csv', header=False, index=False)

frequency = np.zeros((8, 8))
for i in range(as_num):
    data = pd.read_csv(instance_name[i])
    # print (data)
    for c in range(8):
        column = data.iloc[:, c]
        # print (column)
        for r in range(8):
            if column[r] != 0:
                frequency[r][c] = frequency[r][c] + 1
print (frequency)
frequency = frequency / as_num
data = pd.DataFrame(frequency)
data.to_csv('as-frequency.csv', header=False, index=False)
```

7. 把数据填充到 excel 图里面，例如：打开 as-distribution.csv，内容如下：

	A	B	C	D	E	F	G	H
1	0	0	5.28E-05	0.000127	0.000709	0.000605	0	0
2	0.00046	0	5.92E-05	0.000159	0.023738	0.018984	0.001213	0.0087
3	0.011947	0	0.003787	0.004562	0.137864	0.096343	0.017522	0.077783
4	0.006042	0	0.006176	0.007642	0.016156	0.01405	7.71E-05	0.011022
5	0.038863	0	0.01488	0.017164	0.023039	0.019499	0.000575	0.027615
6	0.019334	0	0.003374	0.004825	0.093436	0.061578	0.015519	0.039774
7	0.001883	0	0.000993	0.00134	0.063653	0.051988	0.00677	0.008788
8	0.00028	0	0.002754	0.002736	0.006569	0.006573	0	0.000414

把这些数据复制到刚才的 region.xlsx 中，注意保留小数点，得到 region map 相关的图。
8. 效果如下：

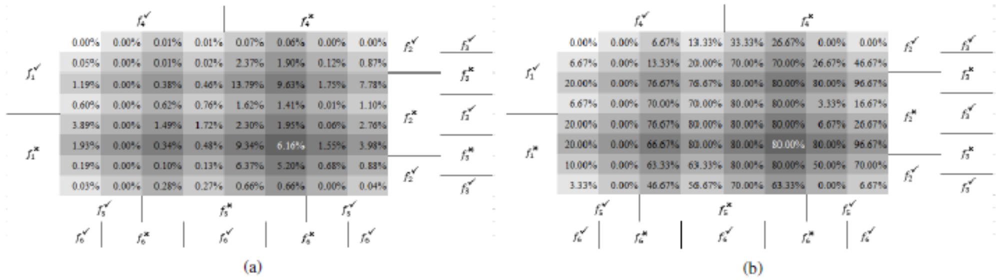


Fig. S4. Region maps for 30 real-world instances. (a) Overall distribution of solutions. (b) Frequency of instances. The solid lines between labels around the map divide it into different regions. $f_i^{\checkmark}$ represents good value in f_i while $f_i^{\times}$ represents bad value in f_i. The cells of the map follow the Gray code (replacing 0's and 1's by ✓and ✗). Each cell in the map represents a region r_k using a binary encoding such that the least significant digit represents objective f_1 and the most significant digit is objective f_m. For example, the solution with good values in all objectives, i.e., ($f_1^{\checkmark}$, $f_2^{\checkmark}$, $f_3^{\checkmark}$, $f_4^{\checkmark}$, $f_5^{\checkmark}$, $f_6^{\checkmark}$), falls into region r_0 in the upper left corner of the map because $0_{10} = 000000_2$. The percentage of solutions in each region was computed for each instance. The distribution map shows the average percentage of the solutions in each region. The frequency map shows the percentage of instances that contain at least one solution in a region. The range threshold was set to the minimum value such that there are no solutions in r_0 (i.e. 0.1 for real-world instances and 0.05 for traditional instances). There are no solutions with good values in all objectives. As for real-world instances, there are more regions without solutions than traditional instances, representing trade-offs for the decision maker. Moreover, the frequency of solutions is higher in most regions for traditional instances, meaning that the fitness landscapes of instances in traditional instances are more alike than those in real-world instances.

说明：

Distribution map：对于某种类型里面的每个算例，distribution map 会计算每个 region 的 solution 数目得到 percentage，然后计算所有算例的 percentage 平均值，填充到对应的格子。

Frequency map：假设一共有 n 个算例，对于特定的某个 region，某个算例只要有解在这个 region，则+1，再除以 n，填充到这个 region 对应的格子。

第四步：多目标散点图分析

1. 运行代码段：

```
ins = 'result/as/MOEAD/MOEAD_GLS/test150-0-0-0-0.d0.tw4.D6/PF/pf'
kcc = KCC(ins, False)
kcc.scatter_plot()
ins = 'result/as/MOEAD/MOEAD_GLS/test250-0-0-0-0.d0.tw4.D6/PF/pf'
kcc = KCC(ins, False)
kcc.scatter_plot()
ins = 'result/s/MOEAD/MOEAD_GLS/C204_100/PF/pf'
kcc = KCC(ins, True)
kcc.scatter_plot()
ins = 'result/s/MOEAD/MOEAD_GLS/R101_100/PF/pf'
kcc = KCC(ins, True)
kcc.scatter_plot()
```

2. 得到图片 **VARO-master/test150-0-0-0-0.d0.tw4.D6-scatter.eps** 等，效果如下：

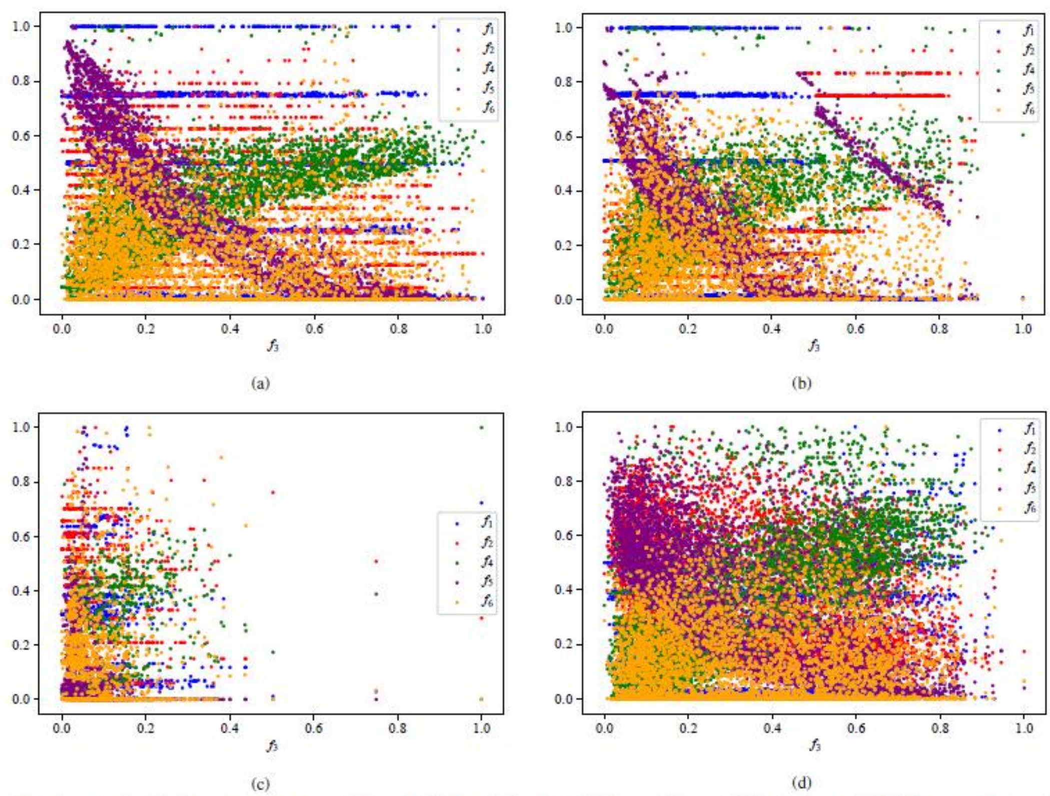


Fig. S6. Scatter plots for four selected instances. (a) Instance 150-4-6. (b) Instance 250-4-6. (c) Instance C204. (d) Instance R101. These scatter plots show the relationship between objective f_3 and each of the other five objectives. The horizontal axis is supposed to be the x-axis and the vertical axis is supposed to be the y-axis. The x-axis shows the value of objective f_3 and the y-axis shows the values of each of the other objectives in different colors. It shows that $f_3 - f_4$ has high harmonious relationships and $f_3 - f_5$ has high conflicting relationships in real-world instances. However, these relationships are not obvious in traditional instances.