Supplementary Material

Enabling empirical SPI target derivation using *a priori* soundscape rankings and multi-objective optimisation

Andrew Mitchell[a,*]

[a] *University College London, Bartlett School of Sustainable Construction, London,*

Abstract

This document provides detailed technical information supporting the multi-objective optimization method presented in Section IV.B.1 of *Soundscape Perception Indices (SPI): Developing context-dependent single value scores of multidimensional soundscape perceptual quality.* The optimization method enables systematic derivation of SPI target distributions from empirical soundscape quality rankings. The method is based on the optimization of a set of objective functions that are designed to capture the most important aspects of soundscape perception. The optimization is performed using a genetic algorithm, which is a stochastic optimization method that is well-suited to the multi-objective optimization problem of deriving SPI targets.

Table of contents

1 Code setup **2**

2 Role of *a priori* rankings in target definition **2**

2.1 Example using park soundscapes from the ISD 3

3 Optimisation Task Formulation **4**

3.1 SPI Targets 4

3.2 Objective Functions 5

3.3 NSGA-II Problem Definition in `pymoo` 6

3.4 Park Quality optimization 8

4 Optimization result **11**

4.1 Selecting the best solution 11

4.2 Resulting Target Distribution 12

*Corresponding author
Email address: `a.j.mitchell@ucl.ac.uk` (Andrew Mitchell)

1. Code setup

This analysis uses the following Python (v3.11.8) packages:

- `pymoo` v0.6.1.3 (Blank and Deb, 2020)
- `soundscapy` v0.7.4 (Mitchell, 2024; Mitchell et al., 2023)
- `KS2D` (Taillon)
- `rpy2` v3.5.12 https://rpy2.github.io

And the following R (v4.4.1) package for MSN fitting and sampling:

- `sn` v2.1.1 (Azzalini, 2023)

```
# Import libraries

import warnings

import numpy as np
import pandas as pd
import soundscapy as sspy
from soundscapy.surveys.survey_utils import LANGUAGE_ANGLES, PAQ_IDS

import optimize_target as ot # import scripts
from MultiSkewNorm import MultiSkewNorm

warnings.filterwarnings("ignore")
```

2. Role of *a priori* rankings in target definition

The core challenge in developing a reference SPI target is determining what constitutes an "ideal" soundscape perception distribution for a given context. While we can directly specify MSN parameters to create bespoke targets based on theoretical expectations or design goals, developing empirically-grounded reference targets requires a more systematic approach.

The *a priori* ranking serves as a bridge between existing knowledge about soundscape quality and the mathematical framework of the SPI. By starting with a ranking of soundscapes whose relative quality has been assessed through some external measure (in this demonstration, mean SSS01 scores, though other metrics could be used), we can use optimization techniques to derive MSN parameters that:

1. When used as an SPI target, produce scores that result in the same ranking order
2. Generate high SPI scores for the highly-ranked soundscapes
3. Define a distribution in the circumplex space that captures the perceptual characteristics common to high-quality soundscapes in this context

This approach allows us to work backwards from known good (and poor) examples to define what the target distribution should look like. For instance, if we know that location A has a better soundscape than location B for our purposes, the optimal target distribution should result in location A receiving a higher SPI score than location B.

2.1. Example using park soundscapes from the ISD

To demonstrate the method of deriving an SPI target from empirical data, we used a subset of locations from the International Soundscape Database (ISD) (Mitchell et al., 2021) that were classified as parks or park-like spaces.

```
# Load latest ISD dataset

data = sspy.isd.load()
data, excl_data = sspy.isd.validate(data)
data = data.query("Language != 'cmn'")

# Exclude RegentsParkJapan outliers (the below code resulted in the `excl_id` list)
# excl_id = list(data.query(
    # "LocationID == 'RegentsParkJapan'"
    # ).query("ISOEventful > 0.72 | ISOEventful < -0.5").index)
# Excluded RegentsParkFields outliers
# excl_id = excl_id + list(data.query(
    # "LocationID == 'RegentsParkFields' and ISOPleasant < 0").index) # Helicopters
excl_id = [652, 706, 548, 550, 551, 553, 569, 580, 609, 618, 623, 636, 643]
data.drop(excl_id, inplace=True)

# Calculate ISOPleasant and ISOEventful
# Here we use the adjusted angles from Aletta et al. (2024) for each language included.
for i, row in data.iterrows():
    lang = row["Language"]
    angles = LANGUAGE_ANGLES[lang]
    iso_pl, iso_ev = (
        sspy.surveys.processing._adj_iso_pl(row[PAQ_IDS], angles, scale=4),
        sspy.surveys.processing._adj_iso_ev(row[PAQ_IDS], angles, scale=4),
    )
    data.loc[i, "ISOPleasant"] = iso_pl
    data.loc[i, "ISOEventful"] = iso_ev
```

The following locations were identified as parks and included:

```
# Separate out parks

parks = [
    "RegentsParkFields",
    "RegentsParkJapan",
    "Noorderplantsoen",
    "StPaulsCross",
    "MiradorSanNicolas",
    "RussellSq",
    "Noorderplantsoen",
    "MonumentoGaribaldi",
    "CampoPrincipe",
]

park_data = data.query("LocationID in @parks")
```

An initial ranking of these locations was created based on their mean SSS01 scores (overall soundscape quality rating) from the survey responses in the ISD:

```
# Creating a somewhat arbitrary ranking of parks
rank_on = "sss01"
park_quality = pd.DataFrame(
    park_data.groupby("LocationID")[rank_on].mean().sort_values(ascending=False)
)
park_quality["Rank"] = range(1, len(park_quality) + 1)
park_quality.head(10)
```

LocationID	sss01	Rank
RegentsParkJapan	4.617978	1
RegentsParkFields	4.467290	2
CampoPrincipe	4.345455	3
MonumentoGaribaldi	4.156250	4
RussellSq	4.020548	5
MiradorSanNicolas	3.964286	6
StPaulsCross	3.803030	7
Noorderplantsoen	2.412371	8

It's important to note that, as stated in the main paper, this ranking is used primarily to demonstrate the methodology of deriving a target from a pre-existing ranking. While based on real survey data, this particular ranking should not be considered a definitive assessment of these spaces' soundscape quality, due to the mono-dimensional nature of the SSS01 question. To develop a true reference target, the ranking would need to be arrived at through more rigorous empirical methods such as paired-choice comparisons or other experimental protocols. The purpose of using this ranking is twofold:

1. To demonstrate that the SPI framework can incorporate existing knowledge or preferences about soundscape quality into the target definition process
2. To show how multi-objective optimization can be used to derive MSN parameters that produce an SPI scoring system aligned with predetermined quality assessments

3. Optimisation Task Formulation

3.1. SPI Targets

To set up the optimisation task, we first need to express the parameter space and any constraints. The SPI target is a set of parameters that define the distribution of soundscape perception in a given soundscape. The target is defined as a multivariate skew-normal (MSN) distribution with the following parameters:

$$Y \sim MSN(\xi, \Omega, \alpha) \tag{1}$$

where:

$$\xi = (\xi_x, \xi_y), -1 \leq \xi \leq 1 \tag{2}$$

is the location parameter, which defines the mean of the distribution in the x and y dimensions. The location parameter is constrained to lie within the range $-1 \leq \xi \leq 1$ to ensure that the target distribution is within the range of possible soundscape perceptions (i.e. within the circumplex).

$$\Omega = \begin{pmatrix} var(x) & cov(x,y) \\ cov(y,x) & var(y) \end{pmatrix} \tag{3}$$

and

$$0 \leq var() \leq 1; -1 \leq cov() \leq 1 \tag{4}$$

is the covariance matrix, which defines the shape of the distribution. The covariance matrix must be symmetric ($cov(x,y) = cov(y,x)$) and positive definite to ensure that the distribution is well-defined. These requirements arise from the mathematical properties needed for a valid probability distribution. The variance and covariance parameters are constrained within realistic ranges based on observed soundscape distributions in the ISD.

$$\alpha = (\alpha_x, \alpha_y), -50 \leq \alpha \leq 50 \tag{5}$$

is the skewness parameter, which defines the skewness of the distribution in the x and y dimensions.

3.2. Objective Functions

Our optimization problem requires carefully chosen objective functions that can effectively translate an ordinal ranking of soundscape quality into meaningful MSN parameters. Two competing objectives are defined to ensure the resulting target distribution is both valid and useful:

1. Rank Correlation Objective:

$$f_1 = r(ranks_{quality}, ranks_{target}) \tag{6}$$

   where r is the Spearman rank correlation coefficient. This objective ensures the derived target preserves the original quality ordering of the soundscapes. A high rank correlation indicates that when the target is used to calculate SPI scores for each location, those scores produce a similar ranking to our *a priori* assessment.

2. Weighted SPI Objective:

$$f_2 = \sum_{i=1}^{m} \frac{1}{rank_i} \cdot SPI_i \tag{7}$$

   where m is the number of locations and $rank_i$ is the *a priori* rank of location i. This objective addresses several important aspects:

   - It ensures the target produces meaningfully scaled scores, not just correct rankings
   - The weighting ($\frac{1}{rank_i}$) prioritizes high SPI scores for locations ranked as high quality
   - It prevents solutions that achieve the correct ranking but with compressed or arbitrary score ranges
   - It helps anchor the target distribution in regions of the circumplex space associated with positive soundscape experiences for this context.

The two objectives work together to resolve key challenges in target derivation:

- Rank correlation alone could produce valid but impractical targets (e.g., targets that correctly rank soundscapes but give very low scores to all locations)
- Weighted scores alone might maximize scores without preserving the relative quality relationships
- Together, they ensure the target both discriminates between soundscape quality levels and produces scores that reflect absolute quality judgments

In `pymoo`, each objective function is supposed to be minimized. Therefore, in the code implementation these objectives are negated to convert them into minimization problems. For each step in the algorithm with a given trial set of parameters, a target distribution will be produced, the SPI for each test location assessed according to the protocol described in the full paper, and the resulting set of SPI scores and ranking will be scored using the objective functions.

3.3. NSGA-II Problem Definition in `pymoo`

```
import pathos
from pymoo.core.callback import Callback
from pymoo.core.problem import ElementwiseProblem, StarmapParallelization
from pymoo.visualization.scatter import Scatter
from pyrecorder.recorder import Recorder
from pyrecorder.writers.streamer import Streamer
from pyrecorder.writers.video import Video
from pymoo.decomposition.asf import ASF


class MyProblem(ElementwiseProblem):
    def __init__(self, data, ranking, **kwargs):
        super().__init__(
            n_var=7,
            n_obj=2,
            n_constr=0,
            xl=np.array([-1, -1, 0, 0, -1, -50, -50]),
            xu=np.array([1, 1, 0.5, 0.5, 1, 50, 50]),
            n_eq_constr=1,
            elementwise_evaluation=True,
            **kwargs,
        )

        self.data = data
        self.ranking = ranking

    def _evaluate(self, X, out, *args, **kwargs):
        # Check if the matrix is positive definite
        h = 1 - int(
            np.all(np.linalg.eigvals(np.array([[X[2], X[4]], [X[4], X[3]]])) > 0)
        )
        out["H"] = h
        if h != 0:
            out["F"] = np.column_stack([0, 0])
            return
        else:
```

```
            tgt = MultiSkewNorm()
            tgt.define_dp(
                np.array([X[0], X[1]]),
                np.array([[X[2], X[4]], [X[4], X[3]]]),
                np.array([X[5], X[6]]),
            )
            tgt.sample()
            r, wspi, spi_ranks, target = ot.target_success(tgt, self.ranking, self.data)

            f1 = -r[0]
            f2 = -wspi / 100

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


class VideoCallback(Callback):
    def __init__(self) -> None:
        super().__init__()
        self.rec = Recorder(Streamer(sleep=0.1))

    def notify(self, algorithm):
        sc = Scatter(
            title="Gen %s" % algorithm.n_gen,
            labels=["spearman", "WSPI"],
        )
        sc.add(algorithm.pop.get("F"))
        sc.do()
        self.rec.record()
```

The optimization problem described above presents significant computational challenges. A naive approach might involve systematically sampling the parameter space through a grid search, evaluating both objective functions at each point. However, with seven parameters to optimize and the need for fine granularity to capture optimal solutions, the search space becomes prohibitively large. Additionally, the requirement that the covariance matrix must be positive definite creates irregular boundaries in the parameter space that make systematic searching impractical.

We therefore employ the Non-dominated Sorting Genetic Algorithm II (NSGA-II), which uses principles from evolutionary computation to search the parameter space efficiently. The algorithm maintains a population of potential solutions, where each solution represents a complete set of MSN parameters (, Ω,). In our implementation, we use a population of 150 solutions, initialized randomly within the parameter constraints defined in Section 3.1.

```
from pymoo.algorithms.moo.nsga2 import NSGA2
from pymoo.operators.crossover.sbx import SBX
from pymoo.operators.mutation.pm import PM
from pymoo.operators.sampling.rnd import FloatRandomSampling
from pymoo.optimize import minimize
from pymoo.termination.default import DefaultMultiObjectiveTermination

algorithm = NSGA2(
    pop_size=150,
    sampling=FloatRandomSampling(),
```

```
    crossover=SBX(),
    mutation=PM(),
    eliminate_duplicates=True,
    # callback=VideoCallback()
)

termination = DefaultMultiObjectiveTermination(n_max_gen=100)
```

The algorithm proceeds iteratively, with each iteration (or generation) involving four main steps:

First, the algorithm evaluates both objective functions (rank correlation and weighted scores) for each solution in the current population. Since we have multiple objectives, there is rarely a single "best" solution. Instead, solutions are ranked based on dominance - solution A dominates solution B if it performs at least as well on both objectives and better on at least one. Solutions that are not dominated by any other solutions form the first front, those only dominated by solutions in the first front form the second front, and so on.

Second, to maintain diversity in the population, the algorithm calculates a "crowding distance" for each solution. This distance measures how close a solution is to its neighbors in terms of objective function values. Solutions that are more isolated (have larger crowding distances) are preferred to prevent the population from clustering too tightly around local optima.

Third, new solutions are generated through crossover and mutation operations. Crossover combines parameters from two parent solutions to create offspring solutions, while mutation introduces small random changes to parameter values. These operations are controlled to ensure new solutions remain within the valid parameter ranges and covariance matrix constraints.

Finally, the algorithm selects solutions to form the next generation's population. Solutions are chosen primarily based on their front ranking (lower/better fronts are preferred), and within the same front, solutions with larger crowding distances are preferred. This selection process ensures both convergence toward better solutions and maintenance of diversity in the population.

The algorithm runs for 100 generations, producing a set of solutions known as the Pareto front - solutions representing different trade-offs between our two objectives. From this front, we select a final solution using an Augmented Scalarizing Function with weights [0.48, 0.52], indicating a slight preference for the weighted score objective while maintaining strong rank correlation.

3.4. Park Quality optimization

```
# initialize the thread pool and create the runner
mp = pathos.helpers.mp
n_process = 12
pool = mp.Pool(n_process)
runner = StarmapParallelization(pool.starmap)

# Initialize the NSGA problem
park_problem = MyProblem(
    data=park_data, ranking=park_quality.sort_index()["Rank"], elementwise_runner=runner
)

# Run the optimization
park_res = minimize(
    park_problem, algorithm, termination, seed=42, save_history=True, verbose=True
```

```
)

pool.close()

park_F = park_res.F
park_X = park_res.X
```

```
=============================================================================================
n_gen  |  n_eval  | n_nds  |     cv_min    |     cv_avg    |       eps      |   indicator
=============================================================================================
     1 |      150 |      2 |  0.000000E+00 |  0.7599240000 |              - |             -
     2 |      300 |      4 |  0.000000E+00 |  0.3532980000 |  0.2307692308 |         ideal
     3 |      450 |      5 |  0.000000E+00 |  0.000000E+00 |  0.2103253984 |             f
     4 |      600 |      6 |  0.000000E+00 |  0.000000E+00 |  0.0675906442 |         ideal
     5 |      750 |      7 |  0.000000E+00 |  0.000000E+00 |  0.4865944937 |         nadir
     6 |      900 |      8 |  0.000000E+00 |  0.000000E+00 |  0.0400000000 |         ideal
     7 |     1050 |      8 |  0.000000E+00 |  0.000000E+00 |  1.3605273566 |         nadir
     8 |     1200 |      8 |  0.000000E+00 |  0.000000E+00 |  0.3479604701 |         ideal
     9 |     1350 |      7 |  0.000000E+00 |  0.000000E+00 |  0.0532169713 |             f
    10 |     1500 |      8 |  0.000000E+00 |  0.000000E+00 |  0.0283981598 |         nadir
    11 |     1650 |     10 |  0.000000E+00 |  0.000000E+00 |  0.0536853296 |         ideal
    12 |     1800 |     10 |  0.000000E+00 |  0.000000E+00 |  0.0389273969 |             f
    13 |     1950 |     11 |  0.000000E+00 |  0.000000E+00 |  0.0152857034 |             f
    14 |     2100 |     13 |  0.000000E+00 |  0.000000E+00 |  0.0220277675 |         ideal
    15 |     2250 |     14 |  0.000000E+00 |  0.000000E+00 |  0.0112857632 |         ideal
    16 |     2400 |     15 |  0.000000E+00 |  0.000000E+00 |  0.2329789760 |         nadir
    17 |     2550 |     13 |  0.000000E+00 |  0.000000E+00 |  0.0298152224 |         ideal
    18 |     2700 |     14 |  0.000000E+00 |  0.000000E+00 |  0.1541574361 |         nadir
    19 |     2850 |     13 |  0.000000E+00 |  0.000000E+00 |  0.0314136126 |         ideal
    20 |     3000 |     13 |  0.000000E+00 |  0.000000E+00 |  0.0221547289 |             f
    21 |     3150 |     14 |  0.000000E+00 |  0.000000E+00 |  0.0042640672 |             f
    22 |     3300 |     14 |  0.000000E+00 |  0.000000E+00 |  0.0221583788 |             f
    23 |     3450 |     14 |  0.000000E+00 |  0.000000E+00 |  0.0030074562 |             f
    24 |     3600 |     13 |  0.000000E+00 |  0.000000E+00 |  0.0038190783 |         ideal
    25 |     3750 |     13 |  0.000000E+00 |  0.000000E+00 |  0.0032096043 |             f
    26 |     3900 |     13 |  0.000000E+00 |  0.000000E+00 |  0.0029877594 |             f
    27 |     4050 |     12 |  0.000000E+00 |  0.000000E+00 |  0.0031231890 |             f
    28 |     4200 |     12 |  0.000000E+00 |  0.000000E+00 |  0.0087298000 |         ideal
    29 |     4350 |     12 |  0.000000E+00 |  0.000000E+00 |  0.0033031039 |             f
    30 |     4500 |     13 |  0.000000E+00 |  0.000000E+00 |  0.0100999852 |             f
    31 |     4650 |     13 |  0.000000E+00 |  0.000000E+00 |  0.0018602710 |             f
    32 |     4800 |     13 |  0.000000E+00 |  0.000000E+00 |  0.0035010475 |             f
    33 |     4950 |     12 |  0.000000E+00 |  0.000000E+00 |  0.0149444800 |         ideal
    34 |     5100 |     10 |  0.000000E+00 |  0.000000E+00 |  0.0056905039 |             f
    35 |     5250 |     10 |  0.000000E+00 |  0.000000E+00 |  0.000000E+00 |             f
    36 |     5400 |     10 |  0.000000E+00 |  0.000000E+00 |  0.000000E+00 |             f
    37 |     5550 |     10 |  0.000000E+00 |  0.000000E+00 |  0.000000E+00 |             f
    38 |     5700 |     10 |  0.000000E+00 |  0.000000E+00 |  0.0006039017 |             f
    39 |     5850 |     10 |  0.000000E+00 |  0.000000E+00 |  0.0006039017 |             f
    40 |     6000 |     10 |  0.000000E+00 |  0.000000E+00 |  0.0006039017 |             f
    41 |     6150 |     10 |  0.000000E+00 |  0.000000E+00 |  0.0006039017 |             f
```

42	6300	10	0.000000E+00	0.000000E+00	0.0017727437	f
43	6450	12	0.000000E+00	0.000000E+00	0.0195162366	f
44	6600	12	0.000000E+00	0.000000E+00	0.000000E+00	f
45	6750	12	0.000000E+00	0.000000E+00	0.000000E+00	f
46	6900	12	0.000000E+00	0.000000E+00	0.000000E+00	f
47	7050	12	0.000000E+00	0.000000E+00	0.000000E+00	f
48	7200	12	0.000000E+00	0.000000E+00	0.0042393714	f
49	7350	12	0.000000E+00	0.000000E+00	0.0020779414	f
50	7500	12	0.000000E+00	0.000000E+00	0.0020779414	f
51	7650	12	0.000000E+00	0.000000E+00	0.0029476155	f
52	7800	12	0.000000E+00	0.000000E+00	0.000000E+00	f
53	7950	12	0.000000E+00	0.000000E+00	0.0527040487	nadir
54	8100	11	0.000000E+00	0.000000E+00	0.0022303428	f
55	8250	11	0.000000E+00	0.000000E+00	0.0042397805	f
56	8400	11	0.000000E+00	0.000000E+00	0.000000E+00	f
57	8550	12	0.000000E+00	0.000000E+00	0.0032736003	f
58	8700	12	0.000000E+00	0.000000E+00	0.000000E+00	f
59	8850	12	0.000000E+00	0.000000E+00	0.000000E+00	f
60	9000	12	0.000000E+00	0.000000E+00	0.000000E+00	f
61	9150	12	0.000000E+00	0.000000E+00	0.000000E+00	f
62	9300	12	0.000000E+00	0.000000E+00	0.000000E+00	f
63	9450	12	0.000000E+00	0.000000E+00	0.0010929922	f
64	9600	12	0.000000E+00	0.000000E+00	0.0010929922	f
65	9750	12	0.000000E+00	0.000000E+00	0.0010929922	f
66	9900	12	0.000000E+00	0.000000E+00	0.0010929922	f
67	10050	12	0.000000E+00	0.000000E+00	0.0010929922	f
68	10200	12	0.000000E+00	0.000000E+00	0.0010929922	f
69	10350	13	0.000000E+00	0.000000E+00	0.0127464015	ideal
70	10500	13	0.000000E+00	0.000000E+00	0.000000E+00	f
71	10650	13	0.000000E+00	0.000000E+00	0.000000E+00	f
72	10800	13	0.000000E+00	0.000000E+00	0.0000889336	f
73	10950	13	0.000000E+00	0.000000E+00	0.0000889336	f
74	11100	13	0.000000E+00	0.000000E+00	0.0000889336	f
75	11250	13	0.000000E+00	0.000000E+00	0.0001067203	f
76	11400	13	0.000000E+00	0.000000E+00	0.0001067203	f
77	11550	13	0.000000E+00	0.000000E+00	0.0002112172	f
78	11700	13	0.000000E+00	0.000000E+00	0.0002112172	f
79	11850	13	0.000000E+00	0.000000E+00	0.0002112172	f
80	12000	13	0.000000E+00	0.000000E+00	0.0002112172	f
81	12150	13	0.000000E+00	0.000000E+00	0.0002112172	f
82	12300	13	0.000000E+00	0.000000E+00	0.0002112172	f
83	12450	13	0.000000E+00	0.000000E+00	0.0002112172	f
84	12600	13	0.000000E+00	0.000000E+00	0.0002112172	f
85	12750	13	0.000000E+00	0.000000E+00	0.0002112172	f
86	12900	13	0.000000E+00	0.000000E+00	0.0002112172	f
87	13050	13	0.000000E+00	0.000000E+00	0.0002112172	f
88	13200	13	0.000000E+00	0.000000E+00	0.0067505547	f
89	13350	13	0.000000E+00	0.000000E+00	0.000000E+00	f
90	13500	13	0.000000E+00	0.000000E+00	0.000000E+00	f
91	13650	13	0.000000E+00	0.000000E+00	0.0014340535	f
92	13800	13	0.000000E+00	0.000000E+00	0.0027146967	f
93	13950	13	0.000000E+00	0.000000E+00	0.000000E+00	f

```
 94 |    14100 |     13 |  0.000000E+00 |  0.000000E+00 |  0.000000E+00 |             f
 95 |    14250 |     13 |  0.000000E+00 |  0.000000E+00 |  0.000000E+00 |             f
 96 |    14400 |     13 |  0.000000E+00 |  0.000000E+00 |  0.000000E+00 |             f
 97 |    14550 |     13 |  0.000000E+00 |  0.000000E+00 |  0.000000E+00 |             f
 98 |    14700 |     13 |  0.000000E+00 |  0.000000E+00 |  0.0030192941 |             f
 99 |    14850 |     13 |  0.000000E+00 |  0.000000E+00 |  0.000000E+00 |             f
100 |    15000 |     13 |  0.000000E+00 |  0.000000E+00 |  0.000000E+00 |             f
```

4. Optimization result

4.1. Selecting the best solution

The optimization produces a set of non-dominated solutions forming a Pareto front, where each point represents a different set of MSN parameters. However, selecting a single solution from this front requires careful consideration of the relative scales of our objective functions. The rank correlation objective (f) typically ranges from -1 to 1, while the weighted SPI score objective (f) can range from 0 to 100. This difference in scales means we cannot directly compare or combine these objectives without normalization.

To address this scale disparity, we first approximate the boundaries of our objective space using the best and worst values found for each objective during the optimization process. These boundary points (called the ideal and nadir points) allow us to normalize both objectives to a common scale ranging from 0 to 1. The normalized front maintains the same trade-off relationships between solutions but allows for fair weighting in the final selection process.

To select a single solution from the Pareto front, we can use various methods such as:

1. Choosing the solution with the highest f_1 value that meets a minimum threshold for f_2.
2. Selecting the solution that minimises the Euclidean distance from the ideal point $(\min -f_1, \min -f_2)$.
3. Using an additional preference function to weight the relative importance of f_1 and f_2.

For this demonstration, we opt for the second method, selecting the solution closest to the ideal point in the normalised objective space. This approach provides a balance between ranking consistency and high SPI scores for top-ranked soundscapes. Once the optimal solution is selected, we can sample from the MSN distribution and plot the derived target distribution, shown in Figure~1(b).

The solution yields the following MSN parameters:

```python
from pymoo.decomposition.asf import ASF

# Get the approximated ideal and nadir points
approx_ideal = park_res.F.min(axis=0)
approx_nadir = park_res.F.max(axis=0)

# Normalize the obtained front
nF = (park_res.F - approx_ideal) / (approx_nadir - approx_ideal)

weights = np.array([0.4, 0.5])
decomp = ASF()

park_I = decomp(nF, weights).argmin()
# print("Best regarding decomposition: Point %s - %s" % (park_I, park_res.F[park_I]))

print(park_tgt.summary())
```

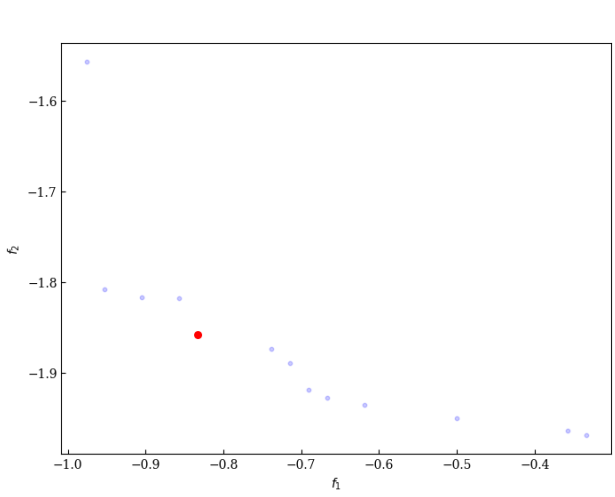
```
Fitted from direct parameters.
Direct Parameters:
xi:    [0.694 0.406]
omega: [[0.157 0.04 ]
 [0.04  0.255]]
alpha: [  5.054 -37.671]


None
None
```

4.2. Resulting Target Distribution

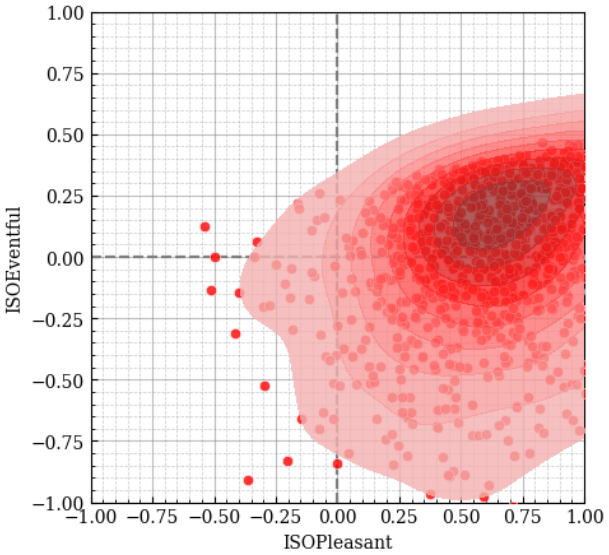
Figure 1 (a) shows the Pareto front obtained from the optimization process, where each point represents a different set of MSN parameters and their corresponding objective function values. The x-axis shows the negative rank correlation (-f) and the y-axis shows the negative weighted SPI score (-f). The selected solution using the ASF with equal weights is highlighted in red.

```
import matplotlib.pyplot as plt

park_X = park_res.X[park_I]
park_tgt = MultiSkewNorm()
park_tgt.define_dp(
    np.array([park_X[0], park_X[1]]),
    np.array([[park_X[2], park_X[4]], [park_X[4], park_X[3]]]),
    np.array([park_X[5], park_X[6]]),
)
park_tgt.sample()

# print(park_tgt.summary())

plot = Scatter()
plot.add(park_res.F, color="blue", alpha=0.2, s=10)
plot.add(park_res.F[park_I], color="red", s=30)
plot.do()
# plot.apply(lambda ax: ax.arrow(0, 0, 0.5, 0.5, color='black',
#                                head_width=0.01, head_length=0.01, alpha=0.4))
plot.show()
plt.show()



# park_tgt.sspy_plot()
df = pd.DataFrame(park_tgt.sample_data, columns=["ISOPleasant", "ISOEventful"])
sspy.plotting.density_plot(
    df, color='red', title=None
)
plt.show()
```

The multi-objective optimization approach presented here demonstrates a systematic method for deriving SPI target distributions from empirical rankings. By simultaneously optimizing for rank correlation and



(a) Multi-objective optimization Pareto front. The selected solution is indicated in red.



(b) SCM distribution of the derived target distribution.

Figure 1: NSGA-II optimization to learn the MSN parameters which produce the Park ranking.

weighted scores, the method produces targets that both preserve ordinal relationships and generate meaningful absolute scores. The park soundscape example shows how this approach can translate qualitative understanding of soundscape quality into quantitative assessment criteria.

References

Azzalini, A.A., 2023. The R package `sn`: The skew-normal and related distributions such as the skew-t and the SUN (version 2.1.1). Università degli Studi di Padova, Italia. URL: https://cran.r-project.org/package=sn. home page: http://azzalini.stat.unipd.it/SN/.

Blank, J., Deb, K., 2020. pymoo: Multi-objective optimization in python. IEEE Access 8, 89497–89509.

Mitchell, A., 2024. Soundscapy: A python package for soundscape assessment and analysis, in: INTER-NOISE and NOISE-CON Congress and Conference Proceedings, pp. 4029–4039. doi:10.3397/in_2024_3404.

Mitchell, A., Aletta, F., Oberman, T., Kang, J., 2023. How do we define soundscape?, in: Forum Acusticum 2023, Turin. arXiv:https://appfa2023.silsystem.solutions/atti/000359.pdf.

Mitchell, A., Oberman, T., Aletta, F., Erfanian, M., Kachlicka, M., Lionello, M., Kang, J., 2021. The International Soundscape Database: An integrated multimedia database of urban soundscape surveys – questionnaires with acoustical and contextual information. doi:10.5281/zenodo.5578572.

Taillon, G., . 2DKS. https://github.com/Gabinou/2DKS. URL: https://github.com/Gabinou/2DKS.