Studying the Performance of the Jellyfish Search Optimiser for the Application of Projection Pursuit

H. Sherry Zhang[a,*], Dianne Cook[b], Nicolas Langrené[c], Jessica Wai Yin Leung[b]

[a] *University of Texas at Austin, Department of Statistics and Data Sciences, Austin, United States, 78751*

[b] *Monash University, Department of Econometrics and Business Statistics, Melbourne, Australia, 3800*

[c] *BNU-HKBU United International College, Department of Mathematical Sciences, Zhuhai, China, 519087*

Abstract

Projection pursuit (PP) is a dimension reduction technique that identifies low-dimensional projections in high-dimensional data by optimising a criteria function known as the PP index. The optimisation in PP can be non-smooth and requires identifying optima with a small "squint angle", detectable only from close proximity. To address these challenges, this study investigates the performance of a recent swarm-based algorithm, Jellyfish Search Optimiser (JSO), for optimising PP indexes. The performance of JSO is evaluated across various hyper-parameter settings and compared with existing optimisers. Additionally, this work proposes novel methods to quantify two properties of the PP index – smoothness and squintability – that capture the complexities inherent in PP optimisation problems. These two metrics are evaluated with JSO hyper-parameters to determine their effect on JSO success rate. The JSO algorithm has been implemented in the `tourr` package, while calculations for smoothness and squintability are available in the `ferrn` package.

Keywords: projection pursuit, Jellyfish Search Optimiser (JSO), Optimisation

README:

- British English ("American or British usage is accepted, but not a mixture of these")*
- No "we" - always third person
- Check if the affiliation information is correct
- word use:
  - Use jellyfish search optimiser, or JS optimiser, not jellyfish optimiser (my bad for leading the way)
  - "hyper-parameter" rather than "hyperparameter"

1. Introduction [Nicolas and Jessica]

Projection Pursuit (PP) is a high-dimensional data visualisation approach that involves computing informative linear projections of data by optimising a variety of objective functions, namely *projection pursuit index (PPI)* (Hall [1], Cook et al. [2], Lee and Cook [3], Loperfido [4], Loperfido [5]). Given high-dimensional

*Corresponding author

Email addresses: `huize.zhang@austin.utexas.edu` (H. Sherry Zhang), `dicook@monash.edu` (Dianne Cook), `nicolaslangrene@uic.edu.cn` (Nicolas Langrené), `Jessica.Leung@monash.edu` (Jessica Wai Yin Leung)

data $X \in \mathbf{R}^n \times p$ and PPI $f(\cdot)$, PP finds the orthonormal projection matrix $A \in \mathbf{R}^{p \times d}$ by solving the following optimisation problem:

$$\max_{A} \quad f(XA) \quad \text{subject to} \quad A'A = I$$

These indices $f(\cdot)$ are characterised by the informativeness or "interestingness" of a projection and are often non-linear and non-convex. As such, an effective and efficient optimisation procedure is essential to enable sufficient exploration of the data landscape during the visualisation process and to arrive at a global optimal viewpoint of the data.

Significant efforts have been devoted to enhancing the functionality of PP and facilitating a better experience in the visualisation process. Cook et al. [6] introduced the PP guided tour, which enabled interactive visualisation of the optimisation to visually explore high-dimensional data. It is implemented in the R [7] package `tourr`[8]. Zhang et al. [9] highlighted potential problems of the optimisation routine implemented. While improving the quality of the optimisation solutions in the tour is essential, it is also important to be able to watch the projected data as the optimisation progresses. As such, it is pertinent to integrate the guided tour with a global optimisation algorithm that is efficient in finding the global optimal and enables viewing of the projected data during the exploration process.

The artificial Jellyfish Search Optimiser (JSO) [10] is a swarm-based metaheuristic designed to solve global optimisation problems. Inspired by the search behaviour of jellyfish in the ocean, JSO is one of the latest swarm intelligence algorithms [11], which was shown to have stronger search ability and faster convergence with fewer parameters compared to classic optimisation methods [10]-[12]. These practical properties makes JSO a strong candidate in integration with the PP guided tour. As such, it is of interest to explore the potential of JSO in enhancing the PP guided tour and examine the characteristics and advantages of such implementation.

The primary goal of this study is to investigate the performance of JSO in the context of the PP guided tour. A series of simulation experiments using well-recognised data sets and PPIs are conducted to yield insights regarding the behaviour of JSO under different settings and the sensitivity of hyper-parameter in the optimisation. Second, to observe the performance of JSO with different types of PPI, a collection of metrics is introduced to capture specific properties of the index including squintability and smoothness as introduced in Laa and Cook [13]. To the best of the authors' knowledge, this is the first attempt to quantitatively measure squintability and smoothness. Finally, the relationship between the JSO performance, hyper-parameters tuning and various properties of PPI is analysed to provide helpful guidance for practitioners that are using the guided tour for high dimensional visualisation.

The rest of this paper is structured as follows. Section 2 introduces the background of the PP guided tour and reviews existing methods in the literature. Section 3 describes the details of JSO and introduces metrics that measure different properties of a PPI. Section 4 visualises the behaviour of the JSO in terms of describes a simulation study on the performance of the JSO using well-known data sets and index functions. Section 5 displays two sets of simulation experiments comparing the JSO with the search-better optimiser and studies the impact of different PPI properties on the optimisation performance. Section 6 summarises the work and provides suggestions for future directions.

2. Projection pursuit, index functions and optimisation [Di and Sherry]

A tour on high-dimensional data is constructed by geodesically interpolating between pairs of planes. Any plane is described by an orthonormal basis, A_t, where t represents time in the sequence. The term "geodesic" refers to maintaining the orthonormality constraint so that each view shown is correctly a projection of the data. The PP guided tour operates by geodesically interpolating to target planes (projections) which have high PP index values, as provided by the optimiser. The geodesic interpolation means that the viewer sees a

continuous sequence of projections of the data, so they can watch patterns of interest forming as the function is optimised. There are five optimisation methods implemented in the `tourr` package:

- `search_geodesic()`: provides a pseudo-derivative optimisation. It searches locally for the best direction, based on differencing the index values for very close projections. Then it follows the direction along the geodesic path between planes, stopping when the next index value fails to increase.
- `search_better()`: is a brute-force optimisation searching randomly for projections with higher index values.
- `search_better_random()`: is essentially simulated annealing [14] where the search space is reduced as the optimisation progresses.
- `search_posse()`: implements the algorithm described in Posse [15].
- `search_polish()`: is a very localised search, to take tiny steps to get closer to the local maximum.

There are several PP index functions available: `holes()` and `cmass()` [2]; `lda_pp()` [16]; `pda_pp()` [3]; `dcor2d()` and `splines2d()` [17]; `norm_bin()` and `norm_kol()` [18]; `slice_index()` [19]. Most are relatively simply defined, for any projection dimension, and implemented because they are relatively easy to optimise. A goal is to be able to incorporate more complex PP indexes, for example based on scagnostics (Wilkinson et al. [20], Wilkinson and Wills [21]).

An initial investigation of PP indexes, and the potential for scagnostics is described in Laa and Cook [13]. To be useful here an optimiser needs to be able to handle functions which are not very smooth. In addition, because data structures might be relatively fine, the optimiser needs to be able to find maxima that occur with a small squint angle, that can only be seen from very close by. One last aspect that is useful is for an optimiser to return local maxima in addition to global because data can contain many different and interesting features.

3. The jellyfish optimiser and properties of PP indexes [Nicolas and Jessica]

JSO mimics the natural movements of jellyfish, which include passive and active motions driven by ocean currents and their swimming patterns, respectively. In the context of optimization, these movements are abstracted to explore the search space in a way that balances exploration (searching new areas) and exploitation (focusing on promising areas). The algorithm aims to find the optimal solution by adapting the behaviour of jellyfish to navigate towards the best solution over iterations [10].

To solve the optimisation problem embedded in the PP guided tour, a starting projection, an index function and the maximum number of iterations are provided as input. Then, the current projection is evaluated by the index function. The projection is then moved in a direction determined by a random factor, influenced by how far along we are in the optimization process. Occasionally, completely new directions may be taken like a jellyfish might with ocean currents. A new projection is accepted if it is an improvement compared to the current one, rejected otherwise. This process continues and iteratively improves the projection, until the pre-specified maximum number of trials is reached.

Algorithm: Jellyfish Optimizer Pseudo Code

Input: `current_projections`, `index_function`, `trial_id`, `max_trial`
Output: `optimized_projection`
Initialize `current_best` as the projection with the best index value from `current_projections`, and `current_idx` as the array of index function values for each projection in `current_projections`
for each trial_id in 1 to max_tries **do**
    Calculate c_t based on the current_idx and max_trial
    if c_t is greater than or equal to 0.5 **then**
        Define trend based on the current_best and current_projections
        Update each projection towards the trend using a random factor and orthonor-

malisation
else
if a random number is greater than $1 - c_t$ **then**
Slightly adjust each projection with a small random factor (passive)
else
For each projection, compare with a random jelly and adjust towards or away from it (active)
Update the orientation of each projection to maintain consistency
Evaluate the new projections using the index function
if any new projection is worse than the current, revert to the `current_projections` for that case
Determine the projection with the best index value as the new current_best
if trial_id $\geq$ max_trial, print the last best projection **exit**
return the set of projections with the updated current_best as the optimized_projection

The JSO implementation involves several key parameters that control its search process in optimization problems. These parameters are designed to guide the exploration and exploitation phases of the algorithm. While the specific implementation details can vary depending on the version of the algorithm or its application, we focus on two main parameters that are most relevant to our application: the number of jellyfish and the maximum number of trials.

Laa and Cook [13] has proposed five criteria for assessing projection pursuit indexes (smoothness, squintability, flexibility, rotation invariance, and speed). Since not all the properties affects the execution of the optimisation, here we consider the three relevant properties (smoothness, squintability, and speed), and propose three metrics to evaluate these three properties.

3.1. Smoothness

If we evaluate the index function at some random points (like the random initialization of the jellyfish optimizer), then we can interpret these random index values as a random field, indexed by a space parameter: the random projection angle. This analogy suggests to use this random training sample to fit a spatial model, a simple one being a (spatial) Gaussian process.

How can we define a measure of smoothness from this? The distribution of a Gaussian process is fully determined by its mean function and covariance function. The way the covariance function is defined is where smoothness comes into play: if an index is very smooth, then two close projection angles should produce close index values (strong correlation); by contrast, if an index is not smooth, then two close projection angles might give very different index values (fast decay of correlations with respect to distance between angles).

Popular covariance functions are parametric positive semi-definite functions, some of which have a parameter to capture the smoothness of the Gaussian field. In particular, consider the Matérn class of covariance functions, defined by

$$K(u) := \frac{(\sqrt{2\nu}u)^\nu}{\Gamma(\nu)2^{\nu-1}}\mathcal{K}_\nu(\sqrt{2\nu}u)$$

where $\nu > 0$ is the smoothness parameter and where $\mathcal{K}_\nu$ is the modified Bessel function. The Matérn covariance function can be expressed analytically when ν is a half-integer, the most popular values in the literature being 1/2, 3/2 and 5/2 . The parameter ν, called smoothness parameter, controls the decay of the covariance function. As such, it is an appropriate measure of smoothness of a random field.

In our context, we suggest to use this parameter as a measure of the smoothness of the index function by fitting a Gaussian process prior with Matérn covariance on a dataset generated by random evaluations of

the index function, as in the initial stage of the jellyfish random search. There exist several R packages, such as GpGp or ExaGeoStatR, to fit the hyperparameters of a GP covariance function on data. In this project, we make use of the GpGp package.

The fitted value $\nu > 0$ can be interpreted as follows: the higher ν, the smoother the index function.

3.2. Squintability

From the literature, it is commonly understood that a large squint angle implies that the objective function value is close to optimal even when we are not very close to the perfect view to see the structure. A small squint angle means that index function value improves substantially only when we are very close to the perfect view. As such, low squintability implies rapid improvement in the index value when near the perfect view.

In this study, we propose two metrics to capture the notion of squintability.

[We generate random points that is beyond 1.5 projection distance and interpolate. Then we fit a kernel or use nonlinear least squares.]

First, parametric model.

[Nicolas's pdf]

Second, we consider the product of the largest absolute magnitude of rate of change of f and the corresponding projection angle as a second measure of squintability. Since f is decreasing, the rate of change of f is negative and thus $|\min_x f(' x)|$ gives the absolute magnitude of the most negative rate of change.

[Nicolas's pdf]

TODO: add equation reference to main text line 705

To the best of our knowledge, this is the first attempt to measure the notion of squintability.

3.3. Speed

The speed of optimizing an index function can be measured empirically by recording the duration of the optimisation. Alternatively, one can also gauge the speed in terms of computational complexity of evaluating the index function (for instance, in big O notation, with respect to the sample size) of computing the index function.

4. Visualisation of jellyfish optimiser

The data structure proposed in Zhang et al. [9] is implemented for JS optimisers and additional information such as index function used, data dimension, and jellyfish hyper-parameters can be included as additional columns. An example data collected from finding the sine-wave structure in 6D data (`d = 6`) using a spline index function(`splines2d`) is printed below:

```
Rows: 10
Columns: 12
$ index     <chr> "splines2d", "splines2d", "splines2d", "splines2d", "splines~
$ d         <dbl> 6, 6, 6, 6, 6, 6, 6, 6, 6, 6
$ n_jellies <dbl> 50, 50, 50, 50, 50, 50, 50, 50, 50, 50
$ max_tries <dbl> 50, 50, 50, 50, 50, 50, 50, 50, 50, 50
$ sim       <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
$ seed      <int> 3462, 3462, 3462, 3462, 3462, 3462, 3462, 3462, 3462, 3462
$ basis     <list> <<matrix[6 x 2]>>, <<matrix[6 x 2]>>, <<matrix[6 x 2]>>, <<m~
$ index_val <dbl> 0.036664532, 0.005121912, 0.058683104, 0.056527991, 0.015734~
$ method    <chr> "search_jellyfish", "search_jellyfish", "search_jellyfish", ~
```

```
$ tries     <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
$ loop      <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
$ time      <drtn> 4.362133 mins, 4.362133 mins, 4.362133 mins, 4.362133 mins, ~
```

The jellyfish hyper-parameters used are 50 jellyfishes (`n_jellies`) and a maximum of 50 tries (`max_tries`). The full data contains 50 simulations, recorded by the variable `sim`, with different seeds (`seed`). Recorded in columns `basis`, `index_val`, `tries`, `loop` and `time`, are the sampled projection basis, calculated index value, iteration ID, jellyfish ID, and time taken to find the optimum, respectively. The basis column records every visited basis before comparing with the current (last) basis. If a visited basis has a smaller index value than the current (last) one, the JS optimiser keeps the current basis for the next iteration in the search but still records the visited inferior basis. This data can be used to derive numerical summaries and plots:

- **Success rate**: The success rate is defined as the proportion of simulations that achieve a final index value within 0.05 of the best index value found among all 50 simulations. In the dataset above, the optimal index value is 0.999. Out of the 50 simulations, 44 achieved an index value within [0.949, 0.999], resulting in a success rate of 0.88. Figure 1 shows the best projection found in each simulation, sorted by the index value. Most of the projections show a clear sine-wave structure, while the last six show only a straight line with clusters at the end. When local optima exist in the projection pursuit problem, plotting the projections found in each simulation or by each jellyfish can help identify these local optima.
- **Projection distance**: The projection distance between projection bases and the theoretical best basis (or the empirical best basis if the theoretical best is unknown) is calculated as the Frobenius norm, which measures how close a visited basis is to the best basis. Plotting this projection distance against the index value for each basis visited reveals how the index value changes as the basis approaches the optimal basis, helping to identify the squintability of the index function. Figure 2 shows the index value against the projection distance for the given example and its binned version. As the projection distance decreases, the index value initially increases with an increasing slope and then levels off. The index value shows greater variation at medium projection distances compared to small and large distances.

5. Application [Di and Sherry]

The JS optimiser provides a new swarm-based search strategy to optimise projection pursuit problems. Two examples are provided in this section to demonstrate its performance. The first example compares its performance with the search-better optimiser and explores how it behaves with different combinations of hyper-parameters. The second example studies the effect of optimisation properties defined in Section 3, along with jellyfish hyper-parameters, on the outcome of the optimisation.

5.1. Performance of the JS optimiser in pipe-finding problems

The performance of the JS optimiser is investigated in two folds: 1) compare it with a commonly used optimiser: the search-better optimiser [9, 13], and 2) examine its success rate under different hyper-parameters (number of jellyfishes and maximum number of tries). The projection pursuit problem used is finding the pipe shape using the holes index, investigated by Laa and Cook [13].

Fifty simulations are conducted with both JS and the search-better optimiser, in four dimensions ($d = 6, 8, 10, 12$). The JS optimiser uses 100 jellyfishes and 100 maximum number of tries, while the search-better optimiser allows a maximum of 1000 samples at each iteration before the algorithm terminates. Figure 3 presents the final projections found by the two optimisers, broken down by 10th quantile, faceted by the data dimension. In the 6-dimensional data scenario, the JS optimiser consistently identifies a clear pipe shape. The search-better optimiser also finds the pipe shape but with a wide rim, suggesting a further polish search may be required. With increasing dimensions, the JS optimiser may not always identify the pipe shape due to random sampling, but it still presents a pipe shape in over 50% of cases. When compared

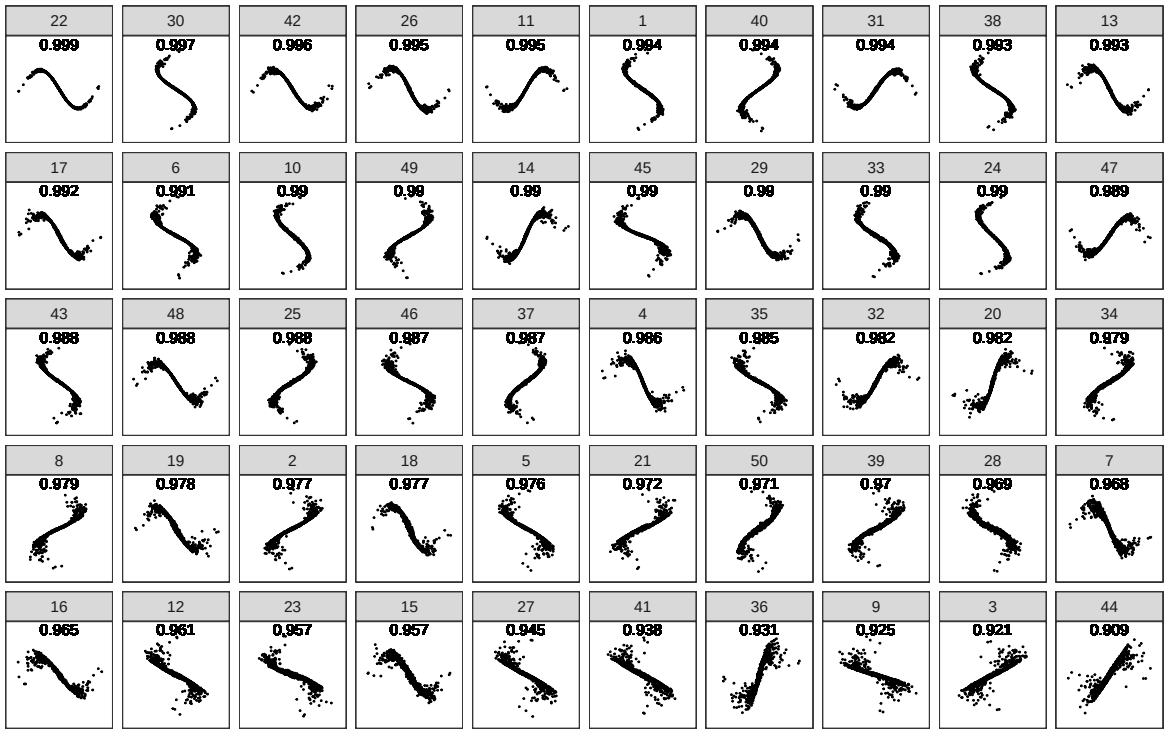

Figure 1: Best projections found by the JS opitmiser in each simulation, sorted by the index value. The projection pursuit problem is to detect the 6D sine-wave structure using the spline index. Most simulations have detected a clear sine-wave pattern, while the last six show a straight line with clustering at the ends.. In this simulation, the success rate is 0.88 (44/50).

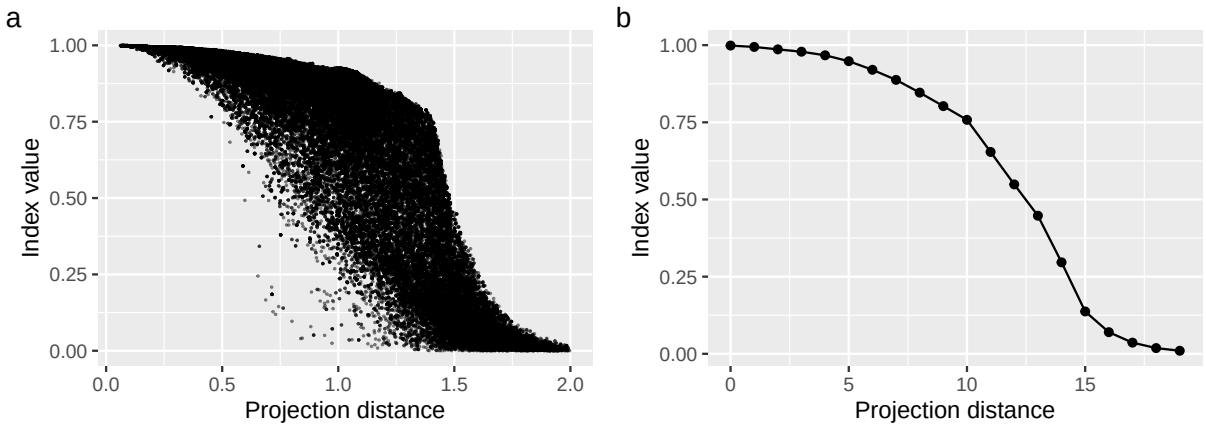

Figure 2: Projection distance plotted against the index value for detecting the 6D sine-wave structure using the spline index function. a) The index values start around 0 with low variance at large projection distances. As the projection distance decreases, the index values increase with a larger variance. Towards the minimum projection distance, the index values increase and cluster around 1, indicating the optimiser is approaching to the optimum. b) Projection distances are binned with a bin width of 0.1, and index values are averaged over each bin. The relationship between the two variables forms a sigmoid curve.

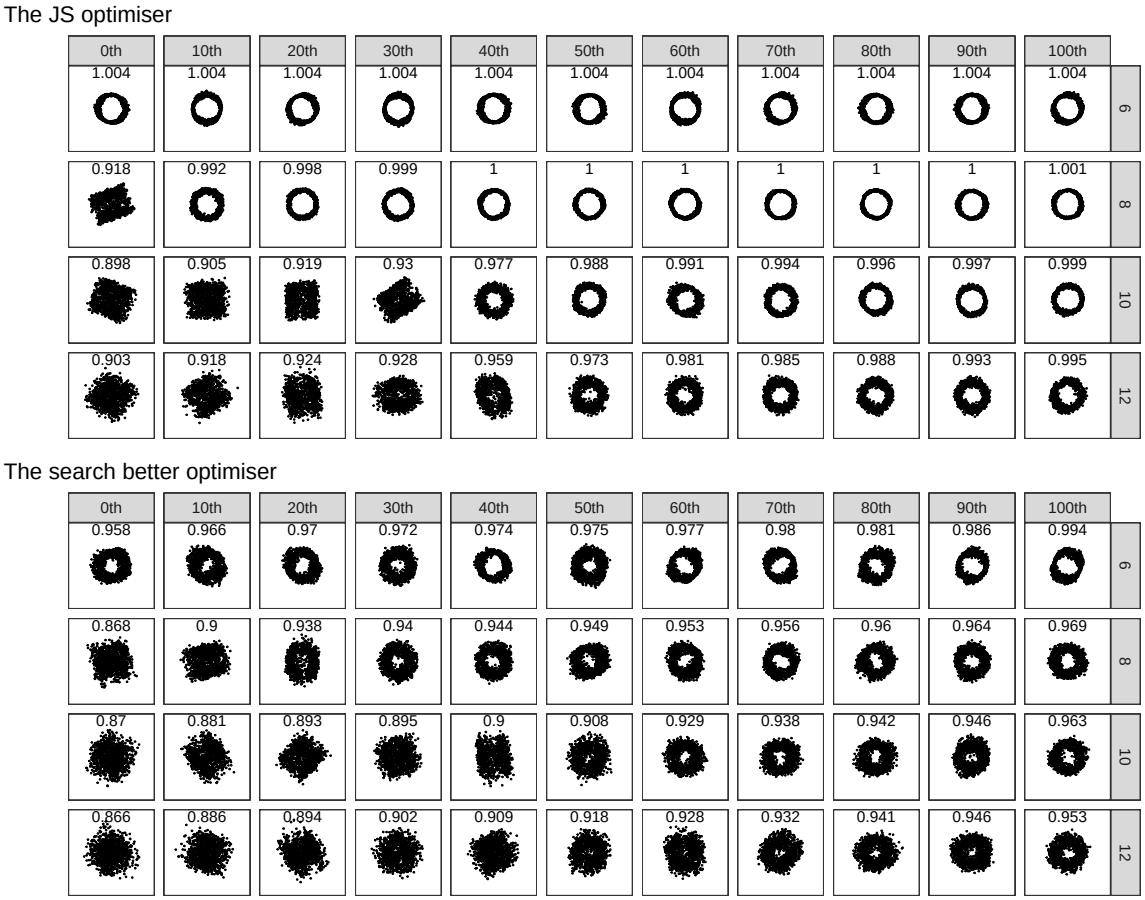

Figure 3: Projections found by the jellyfish and search better optimisers at each 10th quantile across 50 simulations. The projection pursuit problem is to find the pipe shape using the holes index in the 6, 8, 10, and 12-dimensional spaces. The JS optimiser uses 100 jellyfishes and a maximum number of tries of 100. The search better optimiser uses a maximum of 1000 tries in each step of random sampling step before the algorithm terminates. In the 6-D data space, the JS optimiser always finds a clear pipe shape while the search better optimiser also finds the pipe shape but with a wide rim. At higher data dimensions, the JS optimiser finds a higher index value and a clearer pipe shape across all the quantiles than the search better optimiser.

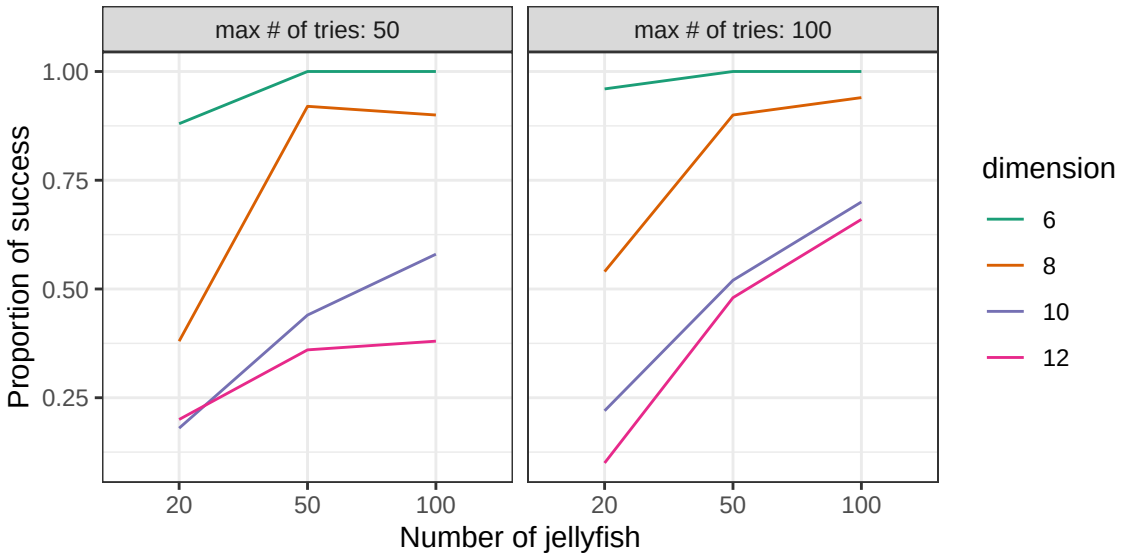

Figure 4: Proportion of simulations reaches near-optimal index values in the pipe-finding problem using the holes index. The proportion is calculated based on the number of simulations, out of 50, that achieve an index value within 0.05 of the best-performing simulation. As the dimensionality increases, the proportion of simulations reaching the optimal index value increases.

to the search-better optimiser, the JS optimiser reaches higher index values and clearer pipe shapes across all quantiles.

In the second experiment, fifty simulations are conducted at each hyper-parameter combination to analyse its effects on the success rate of the JS optimiser. The hyper-parameters tested include: 1) 20, 50, and 100 jellyfishes, and 2) 50 and 100 maximum number of tries. In each simulation, the success rate is calculated as defined in Section 4. Figure 4 presents the proportion of success for each hyper-parameter combinations. As the number of jellyfishes and maximum tries increase, the success rate also increases. For simpler problems (6 dimensions), small parameter values (20 jellyfishes and a maximum number of tries of 50) can already result in high success rate, while for higher-dimensional problems (8, 10, and 12 dimensions), a combination of 100 jellyfishes and 100 maximum number of tries is necessary for at least half of the jellyfishes to find the optimal projection.

5.2. Factors affecting the JS optimiser success rate: optimisation properties and jellyfish hyper-parameters

The optimisation properties, including smoothness, squintability, and speed, offer numerical metrics to characterise the complexity of a projection pursuit optimisation problem. This example investigates how these metrics, along with the jellyfish search hyper-parameters (the number of jellyfishes and the maximum number of tries), affect the success of the JS optimiser. Simulations are conducted to obtain the success rate across various projection pursuit problems, characterised by shape-to-find, data dimension, and index function used, as well as different hyper-parameter combinations. Smoothness and squintability are calculated for each projection pursuit problem as outlined in Section 3. A generalised linear model is used to construct the relationship between the success rate, jellyfish hyper-parameters and optimisation properties.

In addition to the pipe-finding problem, a new shape, sine wave, is investigated in 6D and 8D spaces with six indices considered: `dcor2d_2`, `loess2d`, `MIC`, `TIC`, `spline`, and `stringy`. Combining with two jellyfish hyper-parameters, a total of 52 cases is produced, comprising of 24 pipe-finding cases and 28 sine-wave finding cases. For each case, the JS optimiser is run fifty times and the success rate is calculated as in Section 5.1.

Smoothness and squintability are computed for each case, following the procedures outlined in Section 3.1 and Section 3.2. To calculate smoothness, three hundred random bases are simulated. Index values and projection distance (to the optimal basis) are calculated for each random basis before fitting them into a Gaussian process model. For squintability, fifty random bases are sampled and interpolated to the optimal basis with a step size of 0.005. For these interpolated bases, index values and projection distances to the optimal basis are calculated and averaged with a bin width of 0.005. The squintability measure is then calculated from fitting a 4-parameter scaled logistic function of index value against projection distance, estimated by non-linear least squares.

Table 1 presents the parameters estimated from the Gaussian process (variance, range, smooth, and nugget) and the scaled logistic function (θ_1 to θ_4) for each case. The column "smooth" is used as the smoothness measure and the column "squint" is calculated as [TODO: equation reference] as squintability. The low squint value of MIC and TIC index is due to its convex shape of the index value against the projection distance, as shown in Figure 5. Comparing to other concave shapes, the maximum first derivative happens at a smaller projection distance (θ_2), requiring the optimiser to get closer to the optimal to see a significant change in the index value, hence a small squintability measure.

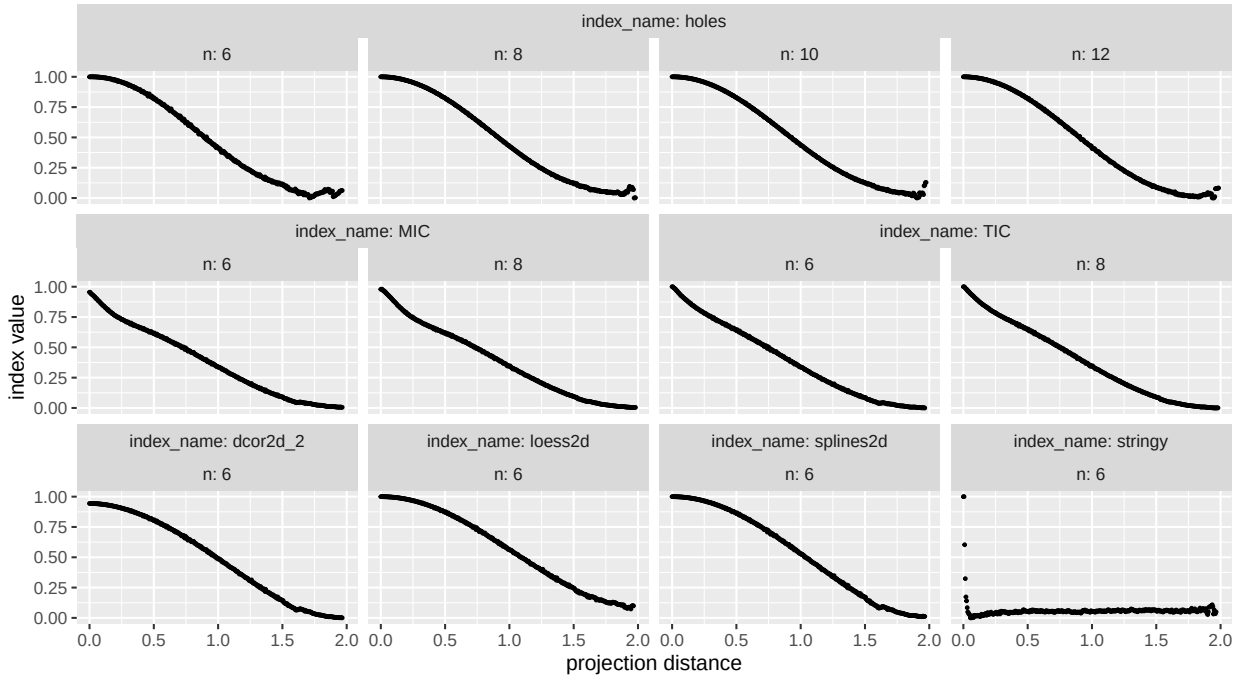

Figure 5: Index values versus projection distance for the 12 pipe/sine-wave finding problem, after the binning procedure for calculating the squintability measure. The index values, averaged at bin width of 0.005, are scaled from 0 to 1 for comparison (holes, TIC, and stringy). The `MIC` and `TIC` index curves are convex while others are concave. The stringy curve shows an instantaneous jump to the optimum when aproaching the best basis.

Table 2 combines all the variables calculated above (the JS optimiser success rate, jellyfish hyper-parameters, and optimisation properties) into one table. Three preprocessing steps are applied to the data: 1) scaling the jellyfish hyper-parameters by a factor of 10 for interpretation, 2) creating a new binary variable `long_time` to indicate cases with an average run time over 30 seconds, and 3) encoding the success rate for the stringy case as 0, since none of the 50 simulations identified the sine-wave shape. A generalised linear model with a binomial family and a logit link function is used to fit the data and Table 3 presents the model outputs. The model suggests that the success rate of the JS optimiser is positively associated with the two jellyfish hyper-parameters, as well as with smoothness and squintability. However, being flagged with long runtime and an increase of data dimension will reduce the success rate. The

Table 1: Parameters estimated from the Gaussian process (including variance, range, smoothness, and nugget) and scaled logistic function (θ_1 to θ_4) for the pipe-finding and sine-wave finding problems. The column "smooth" and "squint" represent the smoothness and squintability measures.

	shape	index	d	variance	range	smooth	nugget	θ_1	θ_2	θ_3	θ_4	squint
1	pipe	holes	6	0.002	0.371	2.355	0.221	1.015	0.860	3.368	0.018	0.763
2	pipe	holes	8	0.000	0.178	2.189	0.820	1.014	0.869	3.264	0.029	0.740
3	pipe	holes	10	0.000	0.107	2.192	3.027	1.016	0.885	3.151	0.022	0.737
4	pipe	holes	12	0.000	0.148	2.295	1.581	1.011	0.878	3.345	0.004	0.779
5	sine	MIC	6	0.016	0.092	2.457	0.083	0.894	0.571	1.623	-0.024	0.314
6	sine	MIC	8	0.016	0.081	2.636	0.084	0.932	0.328	1.314	-0.030	0.193
7	sine	TIC	6	0.124	0.110	2.444	0.086	0.951	0.536	1.719	-0.027	0.330
8	sine	TIC	8	0.125	0.099	2.475	0.085	0.949	0.564	1.723	-0.028	0.342
9	sine	dcor2d	6	0.034	0.167	2.663	0.114	0.954	1.039	2.742	-0.019	0.737
10	sine	loess2d	6	0.079	0.336	1.988	0.309	1.016	1.039	2.648	0.080	0.689
11	sine	splines2d	6	0.042	0.242	2.537	0.102	1.014	1.051	2.730	-0.009	0.780
12	sine	stringy	6	0.000	0.007	1.536	38.166	1.045	0.011	254.734	0.053	0.739

variable `squintability` and `dimension` are significant, suggesting their importance relative to jellyfish hyper-parameters in the optimisation success.

Table 2: The first 7 rows of the datasets processed for modelling.

Index	D	Smoothness	Squintability	Num. of jellyfish	Max. Num. of tries	Prop. of success	Time
MIC	6	2.457	0.314	20	50	0.12	2.479 secs
MIC	6	2.457	0.314	20	100	0.24	8.950 secs
MIC	6	2.457	0.314	50	50	0.52	5.651 secs
MIC	6	2.457	0.314	50	100	0.64	13.223 secs
MIC	6	2.457	0.314	100	50	0.76	19.453 secs
MIC	8	2.636	0.193	20	50	0.08	2.566 secs
MIC	8	2.636	0.193	20	100	0.08	4.960 secs

Table 3: Model estimates of proportion of jellyfish success on optimisation properties and jellyfish hyper-parameters from the generalised linear model with a binomial family and a logit link function. The variable smoothness, squintability, number of jellyfish and maximum number of tries are positively associated with the JS optimiser success rate while data dimension and being flagged as long runtime are negatively associated with the success rate. The variable squintability and dimension are significant, suggesting their importance relative to jellyfish hyper-parameters in the optimisation success.

term	estimate	std.error	statistic	p.value
Intercept	-4.523	5.332	-0.848	0.396
Smoothness	1.195	1.915	0.624	0.533
Squintability	8.241	2.737	3.011	0.003
dimension (d)	-0.628	0.255	-2.462	0.014
long time	-0.874	1.286	-0.680	0.497
number of jellyfish	0.216	0.127	1.701	0.089
maximum number of tries	0.113	0.150	0.752	0.452

6. Conclusion [Di and Sherry]

References

[1] P. Hall, On polynomial-based projection indices for exploratory projection pursuit, The Annals of Statistics 17 (1989) 589–605. URL: https://doi.org/10.1214/aos/1176347127.

[2] D. Cook, A. Buja, J. Cabrera, Projection pursuit indexes based on orthonormal function expansions, Journal of Computational and Graphical Statistics 2 (1993) 225–250. URL: https://doi.org/10.2307/1390644.

[3] E.-K. Lee, D. Cook, A projection pursuit index for large p small n data, Statistics and Computing 20 (2010) 381–392. URL: https://doi.org/10.1007/s11222-009-9131-1.

[4] N. Loperfido, Skewness-based projection pursuit: A computational approach, Computational Statistics and Data Analysis 120 (2018) 42–57. doi:https://doi.org/10.1016/j.csda.2017.11.001.
[5] N. Loperfido, Kurtosis-based projection pursuit for outlier detection in financial time series, The European Journal of Finance 26 (2020) 142–164. doi:https://doi.org/10.1080/1351847X.2019.1647864.
[6] D. Cook, A. Buja, J. Cabrera, C. Hurley, Grand tour and projection pursuit, Journal of Computational and Graphical Statistics 4 (1995) 155–172. URL: https://doi.org/10.1080/10618600.1995.10474674.
[7] R Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, 2023. URL: https://www.R-project.org/.
[8] H. Wickham, D. Cook, H. Hofmann, A. Buja, tourr: An R package for exploring multivariate data with projections, Journal of Statistical Software 40 (2011) 1–18. URL: http://doi.org/10.18637/jss.v040.i02.
[9] H. S. Zhang, D. Cook, U. Laa, N. Langrené, P. Menéndez, Visual diagnostics for constrained optimisation with application to guided tours, The R Journal 13 (2021) 624–641. doi:10.32614/RJ-2021-105.
[10] J.-S. Chou, D.-N. Truong, A novel metaheuristic optimizer inspired by behavior of jellyfish in ocean, Applied Mathematics and Computation 389 (2021) 125535. doi:10.1016/j.amc.2020.125535.
[11] K. Rajwar, K. Deep, S. Das, An exhaustive review of the metaheuristic algorithms for search and optimization: taxonomy, applications, and open challenges, Artificial Intelligence Review (2023) 1–71. doi:10.1007/s10462-023-10470-y.
[12] J.-S. Chou, A. Molla, Recent advances in use of bio-inspired jellyfish search algorithm for solving optimization problems, Scientific Reports 12 (2022) 19157. doi:10.1038/s41598-022-23121-z.
[13] U. Laa, D. Cook, Using tours to visually investigate properties of new projection pursuit indexes with application to problems in physics, Computational Statistics 35 (2020) 1171–1205. doi:10.1007/s00180-020-00954-8.
[14] D. Bertsimas, J. Tsitsiklis, Simulated Annealing, Statistical Science 8 (1993) 10 – 15. URL: https://doi.org/10.1214/ss/1177011077. doi:10.1214/ss/1177011077.
[15] C. Posse, Projection pursuit exploratory data analysis, Computational Statistics and Data Analysis 20 (1995) 669–687. URL: https://www.sciencedirect.com/science/article/pii/0167947395000028. doi:https://doi.org/10.1016/0167-9473(95)00002-8.
[16] E. Lee, D. Cook, S. Klinke, T. Lumley, Projection pursuit for exploratory supervised classification, Journal of Computational and Graphical Statistics 14 (2005) 831–846. URL: https://doi.org/10.1198/106186005X77702.
[17] K. Grimm, Kennzahlenbasierte Grafikauswahl, doctoral thesis, Universität Augsburg, 2016.
[18] P. J. Huber, Projection pursuit, Ann. Statist. 13 (1985) 435–475. URL: https://doi.org/10.1214/aos/1176349519. doi:10.1214/aos/1176349519.
[19] A. B. Ursula Laa, Dianne Cook, G. Valencia, Hole or grain? a section pursuit index for finding hidden structure in multiple dimensions, Journal of Computational and Graphical Statistics 31 (2022) 739–752. URL: https://doi.org/10.1080/10618600.2022.2035230.
[20] L. Wilkinson, A. Anand, R. Grossman, Graph-theoretic scagnostics, in: IEEE Symposium on Information Visualization, 2005. INFOVIS 2005., 2005, pp. 157–164. doi:10.1109/INFVIS.2005.1532142.
[21] L. Wilkinson, G. Wills, Scagnostics distributions, Journal of Computational and Graphical Statistics 17 (2008) 473–491. URL: https://doi.org/10.1198/106186008X320465. doi:10.1198/106186008X320465. arXiv:https://doi.org/10.1198/106186008X320465.