Towards Exploratory Landscape Analysis for Large-scale Optimization: A Dimensionality Reduction Framework

Ryoji Tanabe

Yokohama National University
Yokohama, Japan

Exploratory Landscape Analysis (ELA) [Mersmann 11]

ELA provides a set of numerical features $\mathcal{F}$ based on a sample $\mathcal{X}$

- The features are used to characterize a fitness landscape of a black-box optimization problem by machine learning
- Applications of the ELA approach include:
  - high-level property classification, algorithm selection, performance prediction, per-instance algorithm configuration, etc.
- For details of ELA
  - Session: GECH4+IMPACT (from 13:20)
  - "(A Brief Look at) The Evolution of Exploratory Landscape Analysis" by Pascal Kerschke and Mike Preuss
- The R-package `flacco` [Kerschke 19] is used to compute features
  - `https://github.com/kerschke/flacco`
  - `https://github.com/Reiyan/pflacco` (Python interface)

The R-package flacco [Kerschke 19]: Feature computation software

flacco provides 17 feature classes (343 features in total)

- This work excludes the 3 classes that require additional fevals
- This *presentation* excludes the 5 cell mapping classes

9 feature classes used in this presentation (107 features in total)

Feature class	Name	Num. features
ela_level [Mersmann 11]	levelset	20
ela_meta [Mersmann 11]	meta-model	11
ela_distr [Mersmann 11]	y-distribution	5
nbc [Kerschke 15]	nearest better clustering (NBC)	7
disp [Kerschke 15]	dispersion	18
ic [Munoz 15]	information content	7
basic [Kerschke 19]	basic	15
limo [Kerschke 19]	linear model	14
pca [Kerschke 19]	principal component analysis	10

- Each feature class consists of more than one feature
- ela_level and ela_meta are effective feature classes

Two contributions of this work

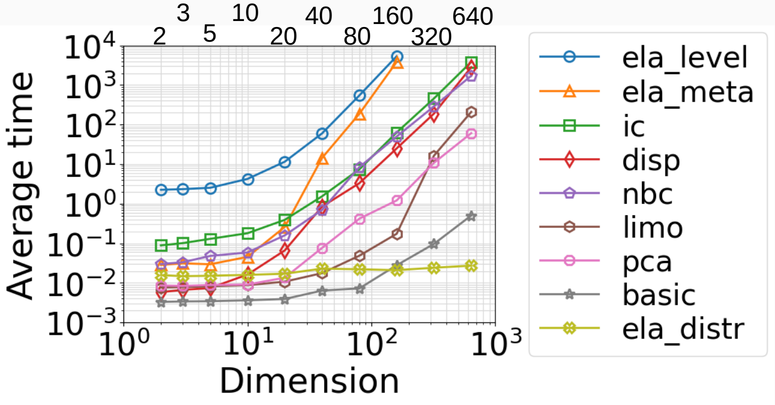
1. **Revealing computational cost issue in ELA**
   - Most previous studies focused on moderate-dim. problems ($n \leq 20$)
     - The scalability of the ELA approach has been overlooked
   - Our results reveal that `ela_level` and `ela_meta` cannot be applied to large-scale optimization due to their high computational cost
2. **Proposing a dimensionality reduction framework**
   - It can speed up the computation of the two feature classes

First experiment: Investigating the feature computation time

- Research questions:
  - How does the feature computation time scale w.r.t. dimensions?
  - Are the 9 feature classes available for large-scale optimization?
- The 24 BBOB functions in COCO [Hansen 21]
  - For $n \in \{2, 3, 5, 10\}$: The noiseless BBOB function set [Hansen 09]
  - For $n \in \{20, 40, 80, 160, 320, 640\}$: Its large-scale version [Varelas 20]
  - But, this first experiment used only the first instance of f_1 (Sphere)
- The (not improved) Latin hypercube sampling method
  - Sample size $|\mathcal{X}| = 50 \times$ dimension n as in most previous studies
  - Time to make $\mathcal{X}$ and calculate $f(\mathcal{X})$ are not considered
- Other settings
  - This work used the Python interface of `flacco` (`pflacco`)
  - A workstation with a 40-Core Xeon 2.1GHz and 384GB RAM
  - Ubuntu 18.04, Python 3.8, R 3.6.

Average computation time (sec) on the first instance of f_1 over 31 runs

- The computation time of all the 9 feature classes (except for `ela_distr`) increases exponentially w.r.t. dimension n
- The computation of `ela_meta` and `ela_level` are time-consuming
  - For $n = 160$, they took approximately 1.1 hours and 1.5 hours, res.
  - For $n \geq 320$, they did not finish within 3 days



Q. Why is the computation of `ela_level` and `ela_meta` time-consuming?

A. Because they use machine learning techniques

- $\mathcal{X}$ is a set of solutions, and $f(\mathcal{X})$ is a set of their objective values
- `ela_level` repeatedly applies three classifiers to $\mathcal{D} = \{\mathcal{X}, f(\mathcal{X})\}$
  - The `ela_level` features are the mean classification errors of classifiers over a 10-fold cross-validation
- `ela_meta` fits linear and quadratic regression models to $\mathcal{D}$
  - The `ela_meta` features are the model-fitting results

When both $|\mathcal{X}|$ and the dimension are large, `ela_level` and `ela_meta` are computationally expensive

- When *only either of* $|\mathcal{X}|$ and the dimension is large, `ela_level` and `ela_meta` are still computationally cheap
  - It is possible to reduce their computation time by setting $|\mathcal{X}|$ to 100
- But, $|\mathcal{X}|$ should be large for large dimensions to obtain effective fea.
  - Recall that $|\mathcal{X}|$ increases linearly w.r.t. dimension n ($|\mathcal{X}| = 50 \times n$)

Proposed: A dimensionality reduction framework

Goal: computing `ela_level` and `ela_meta` in a practical time

- `ela_level` and `ela_meta` are important feature classes in ELA
- It is worth making them available for large-scale optimization

It is inspired by dimensionality reduction strategies in Bayesian opt.

- E.g., GPEME [Liu 14], REMBO [Wang 16], PCA-BO [Raponi 20]
- reduce the original dimension n to a lower dim. m ($m < n$)
- fit the Gaussian process model in the reduced m-dimensional space
- can reduce the computation time for large-scale optimization

Is it possible to reduce the computation time of `ela_level` and `ela_meta` in the same way?

Proposed: A dimensionality reduction framework

Simple two-step procedure to compute features

1. Apply a dimensionality reduction method to $\mathcal{X}$ to obtain $\hat{\mathcal{X}}$
   - We used the weighting strategy-based PCA approach [Raponi 20]
   - It efficiently utilizes the information about the objective values $f(\mathcal{X})$
   - A solution $\boldsymbol{x} \in \mathbb{R}^n$ is mapped to a point $\hat{\boldsymbol{x}} \in \mathbb{R}^m$
   - Let $\hat{\mathcal{X}}$ be a set of m-dimensional points
2. Compute features based on $\hat{\mathcal{X}}$ and $f(\mathcal{X})$, instead of $\mathcal{X}$ and $f(\mathcal{X})$
   - When m is small enough, it is expected that the computation of `ela_level` and `ela_meta` can be fast
   - The proposed framework does not require additional fevals

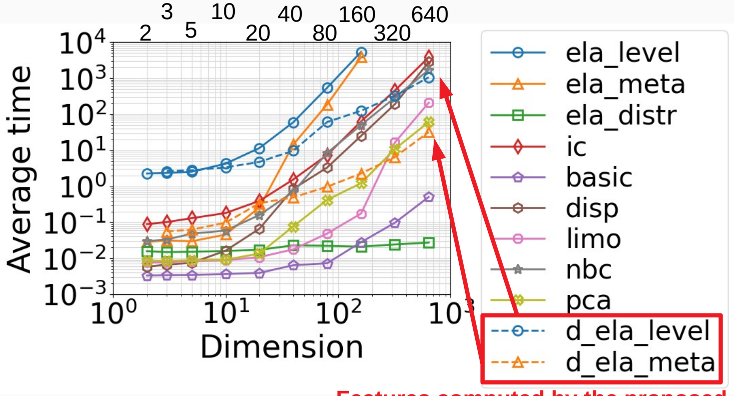
Second experiment: Investigating the computation time of the proposed fr.

- Research question
  - Can the proposed framework reduce the computation time of `ela_level` and `ela_meta` for large dimensions?
- The same experimental setting as in the first experiment (recall p.5)
- The reduced dimension m in the proposed framework
  - m was set to 2
  - Its performance is not sensitive to m unless $m \in \{2, ..., 5\}$

Average computation time (sec) on the first instance of f_1 over 31 runs

Computation time of d_ela_meta and d_ela_level is acceptable

- $n = 320$: the computation of ela_level didn't finish within 3 days
  - That of d_ela_level requires only approx. 5 minutes



Features computed by the proposed Dimensionality reduction framework

Third experiment: Evaluating the effectiveness of features for predicting the high-level properties of the 24 large-scale BBOB functions [Varelas 20]

- Research question:
  - Are the two feature classes computed by the proposed framework (`d_ela_meta` and `d_ela_level`) effective for property classification?
- 7 high-level property classification of the 24 BBOB fun. [Mersmann 11]
  - (1) multimodality, (2) global structure, (3) separability, (4) variable scaling, (5) search space homogeneity, (6) basin sizes, (7) glo. to loc. opt. contrast
  - For each function, the degree of a property was labeled by experts
    - E.g., (1) multimodality: "none", "low", "medium", and "high"
    - While that of f_1 (Sphere) is "none", that of f_3 (Rastrigin) is "high"
  - Task: Predicting the degree of a property of an unseen problem
    - E.g., What is the degree of multimodality of f_2 (Ellipsoidal func.)?
  - Leave-one-problem-out cross-validation (a 24-fold cross-validation)
    - In the i-th fold, f_i is used for test, $\{f_1, ..., f_{24}\} \setminus f_i$ are used for train.
    - 15 instances were used for each f
    - The performance of a classifier is evaluated based on the mean accur.
  - Classifier: Random forest [Breiman 01]
    - The scikit-learn implementation

Three feature sets: C7, C7-E2, and C7-D2

Name	Feature classes	N. features
C7	{ela_distr, basic, ic, disp, nbc, pca, limo}	76
C7-E2	C7 ∪ {ela_level, ela_meta}	107
C7-D2	C7 ∪ {d_ela_level, d_ela_meta}	107

- C7: a set of the 7 computationally Cheap feature classes
- C7-E2: C7 with the 2 computationally Expensive feature classes
  - C7-E2 is available only for $n < 320$
- C7-D2: C7 with the Dimensionality reduction versions of the 2 computationally expensive feature classes
- Feature selection was not performed

Average accuracy of classifiers using C7, C7-E2, and C7-D2 (Multimodality)

Features computed by the propped framework are effective

- C7-D2 is more effective than C7 for $n \geq 320$
  - Similar results were observed for 5 properties (exc. *separability*)
- C7-E2 is more effective than C7-D2 for $n < 320$
  - It would be better to use C7-E2 when it is available (i.e., $n < 320$)

	C7	C7-E2	C7-D2
2	0.642	0.703	Na
3	0.569	0.597	0.578
5	0.536	0.625	0.622
10	0.522	0.631	0.647
20	0.531	0.725	0.639
40	0.556	0.689	0.617
80	0.600	0.783	0.622
160	0.522	0.650	0.561
320	0.544	Na	0.578
640	0.514	Na	0.583

Conclusion

Contribution 1. Revealing computational cost issue in ELA

- Two important feature classes (`ela_level` and `ela_meta`) are not available for large-scale opt. due to their time-consuming process

Contribution 2. Proposing a dimensionality reduction framework

- Our results on the BBOB functions with up to 640 dim. show:
  - the framework can speed up the computation of features classes
  - the effectiveness of features computed by the proposed framework

Sub-contributions not shown in this presentation

- The framework can extend the cell mapping features to large dim.
- An analysis of the similarity between features and their d. r. versions

Future work

- Feature-based algorithm selection for large-scale optimization
- Design of a new computationally cheap feature class