Revisiting Population Models in Differential Evolution on a Limited Budget of Evaluations

Ryoji Tanabe

Yokohama National University, Yokohama, Japan

1. Introduction

- DE has never shown state-of-the-art performance for expensive optimization
  - Even a surrogate-assisted DE has never outperformed a non-surrogate-assisted ES
- This work revisits population models in DE for expensive optimization
  - A population model determines how to update the population for each iteration
  - DE uses the synchronous model, which was designed for *inexpensive* optimization

Q. Can the performance of DE be improved by using a suitable population model?

2.1. Synchronous model (Syn) [Storn 97]

Initialize $\boldsymbol{P} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_\mu\}$ randomly;
while not happy **do**
  for $i \in \{1, ..., \mu\}$ **do**
    $\boldsymbol{u}_i \leftarrow$ Generate a child;
  for $i \in \{1, ..., \mu\}$ **do**
    if $f(\boldsymbol{u}_i) \leq f(\boldsymbol{x}_i)$ **then** $\boldsymbol{x}_i \leftarrow \boldsymbol{u}_i$;

- Population size μ, population $\boldsymbol{P}$, parent individual $\boldsymbol{x}$, child $\boldsymbol{u}$
- Syn updates all individuals in $\boldsymbol{P}$ simultaneously
- The index-based niching mechanism in Syn promotes diversity

2.2. Asynchronous model (Asy) [Wormington 99]

Initialize $\boldsymbol{P} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_\mu\}$ randomly;
while not happy **do**
  for $i \in \{1, ..., \mu\}$ **do**
    $\boldsymbol{u} \leftarrow$ Generate a child;
    if $f(\boldsymbol{u}) \leq f(\boldsymbol{x}_i)$ **then** $\boldsymbol{x}_i \leftarrow \boldsymbol{u}$;

- Immediately after generating $\boldsymbol{u}$, the parent $\boldsymbol{x}_i$ is compared to $\boldsymbol{u}$
- Asy is generally faster than Syn
- Asy can exploit a new superior individual for the search immediately

2.3. $(\mu + \lambda)$ model (Plus)

Initialize $\boldsymbol{P} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_\mu\}$ randomly;
while not happy **do**
  $\boldsymbol{Q} \leftarrow \emptyset$;
  for $i \in \{1, ..., \lambda\}$ **do**
    $\boldsymbol{u} \leftarrow$ Generate a child;
    $\boldsymbol{Q} \leftarrow \boldsymbol{Q} \cup \{\boldsymbol{u}\}$;
  $\boldsymbol{P} \leftarrow \mu$ best individuals in $\boldsymbol{P} \cup \boldsymbol{Q}$;

- Only a few DEs use Plus
- The so-called target vector is randomly selected from the population $\boldsymbol{P}$
- Syn may discard a child that is worse than its parent but better than others
- In contrast, Plus does not do that

2.4. Worst improvement model (WI) [Ali 11]

Initialize $\boldsymbol{P} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_\mu\}$ randomly;
while not happy **do**
  $\boldsymbol{K} \leftarrow$ IDs of λ worst individuals in $\boldsymbol{P}$;
  for $i \in \boldsymbol{K}$ **do**
    $\boldsymbol{u}_i \leftarrow$ Generate a child;
  for $i \in \boldsymbol{K}$ **do**
    if $f(\boldsymbol{u}_i) \leq f(\boldsymbol{x}_i)$ **then** $\boldsymbol{x}_i \leftarrow \boldsymbol{u}_i$;

- WI is similar to Syn
- Only λ worst parents generate children
- A better $\boldsymbol{x}$ is rarely replaced with its $\boldsymbol{u}$
  - Generating $\boldsymbol{u}$ is wasteful
- FEs can be reduced by allowing only λ worst parents to generate their children

2.5. Subset-to-subset model (STS) [Guo 19]

- Individuals in $\boldsymbol{P} \cup \boldsymbol{Q}$ are divided into s groups based on the index-based ring topo.
- Individuals in each group is compared with each other

3. Experimental setup

- Setting for test functions
  - BBOB noiseless function set [Hansen 09] in COCO [Hansen 16]
  - All ECDF figures were generated by COCO with the option `--expensive`
  - Dimensionality $n \in \{2, 3, 5, 10, 20, 40\}$
  - Maximum number of evaluations $= 100 \times n$, number of runs $= 15$
- Two parameter settings for DE
  1. Hand-tuned parameters
     - Configurator: Ryoji Tanabe. Training problem set: the Sphere function
  2. Automatically-tuned parameters
     - Configurator: SMAC [Hutter 11]. Training problem set: CEC2013 [Liang 13]
- Source code and performance data are available:
  - `https://github.com/ryojitanabe/de_expensiveopt`

7. Summary

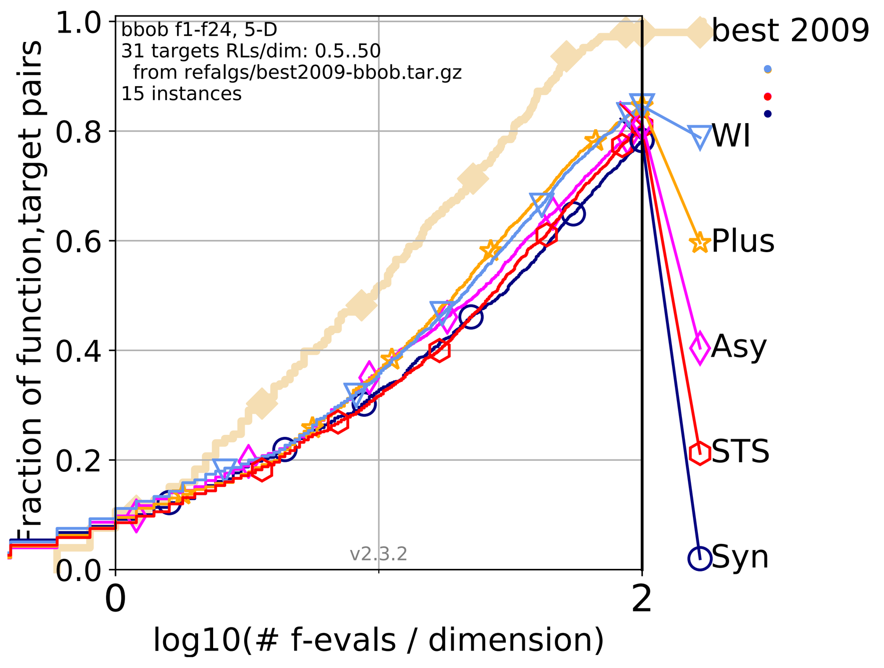
A. Yes, the performance of DE can be improved by using a suitable population model

- The $(\mu + \lambda)$ and worst improvement models are suitable for expensive optimization
- DE with the two models perform better than or similar to CMA-ES depending on FEs and dim n, especially for small FEs (e.g., $10 \times n$) and/or low n (e.g., $n \leq 10$)
- CMA-ES with the auto-tuned parameters significantly outperforms DE for $n \geq 20$
- Future work
  - incorporate a parameter adaptation method for F and C into DE
  - design a surrogate-assisted DE with the $(\mu + \lambda)$ and worst improvement models

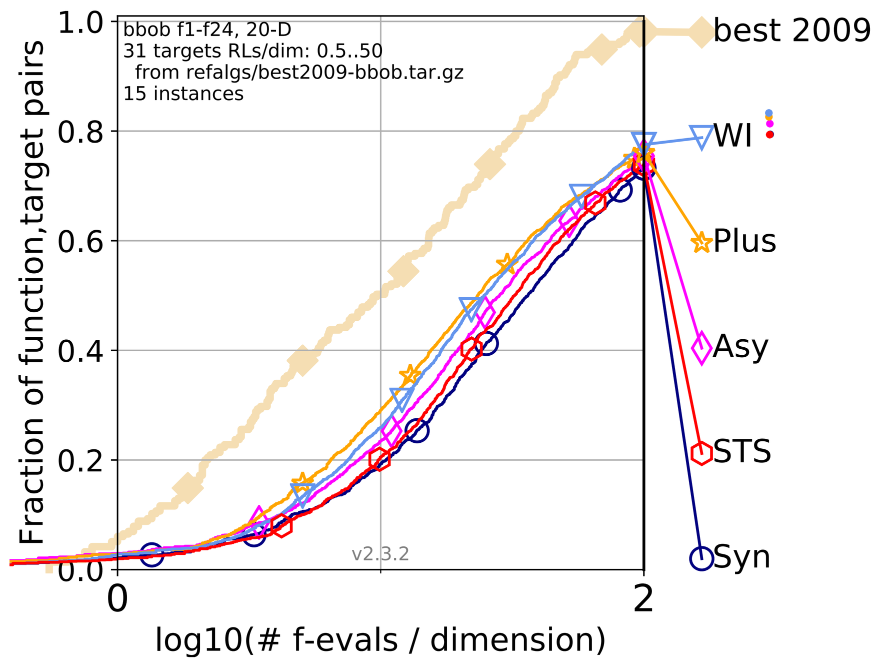
4. Results of DE with the hand-tuned parameters

- The worst improvement and $(\mu + \lambda)$ models show the best performance
- The $(\mu + \lambda)$ performs better than the worst improvement model at the early stage
- The traditional synchronous model performs the worst among the 5 models

5 dim.



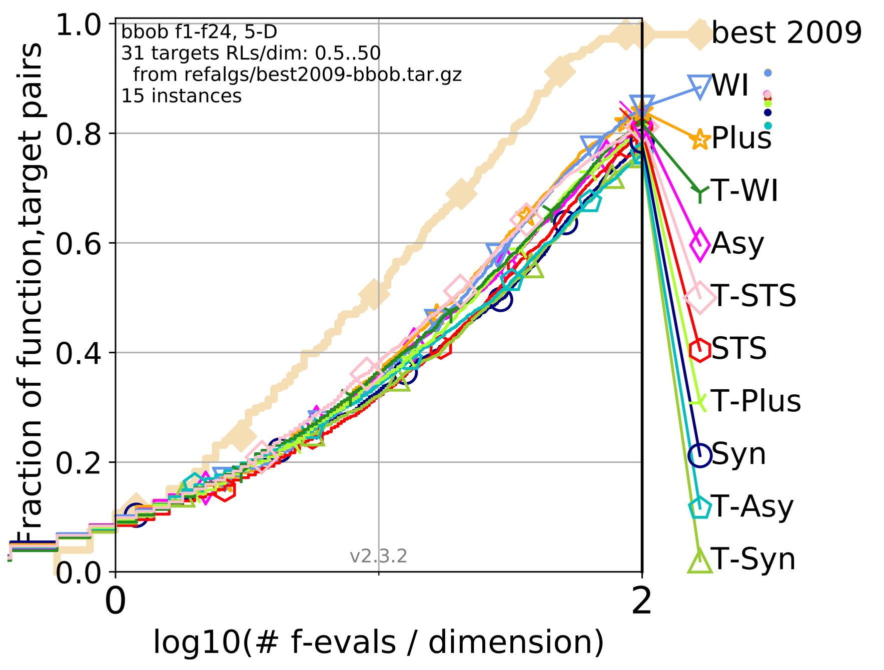
20 dim.



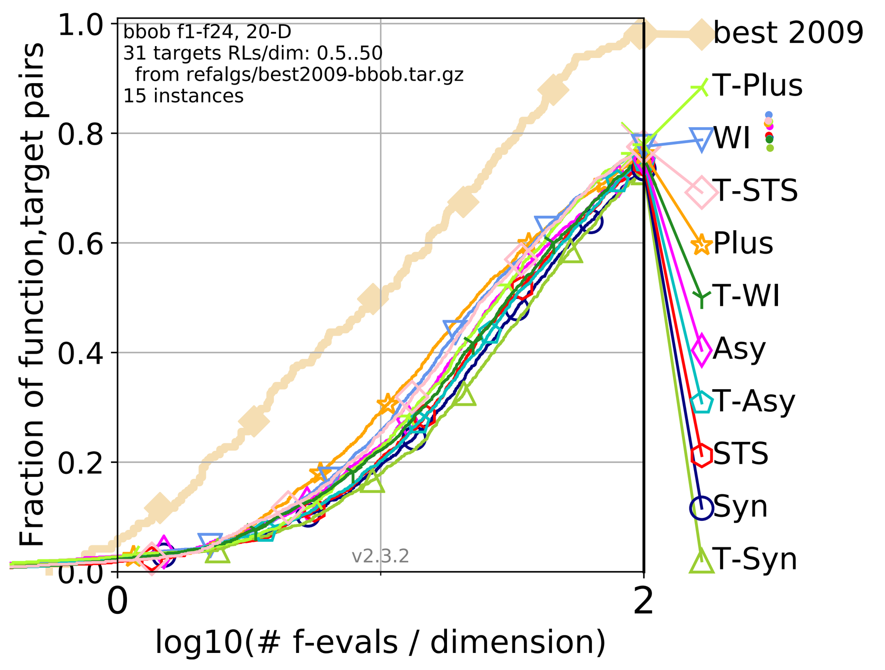
5. Results of DE with the auto-tuned parameters

- "T-" means that the corresponding optimizer uses the auto-tuned parameters
- For $n \in \{20, 40\}$, the auto-tuned parameters are more suitable in most cases
- The results here are almost consistent with the results with the hand-tuned param.

5 dim.



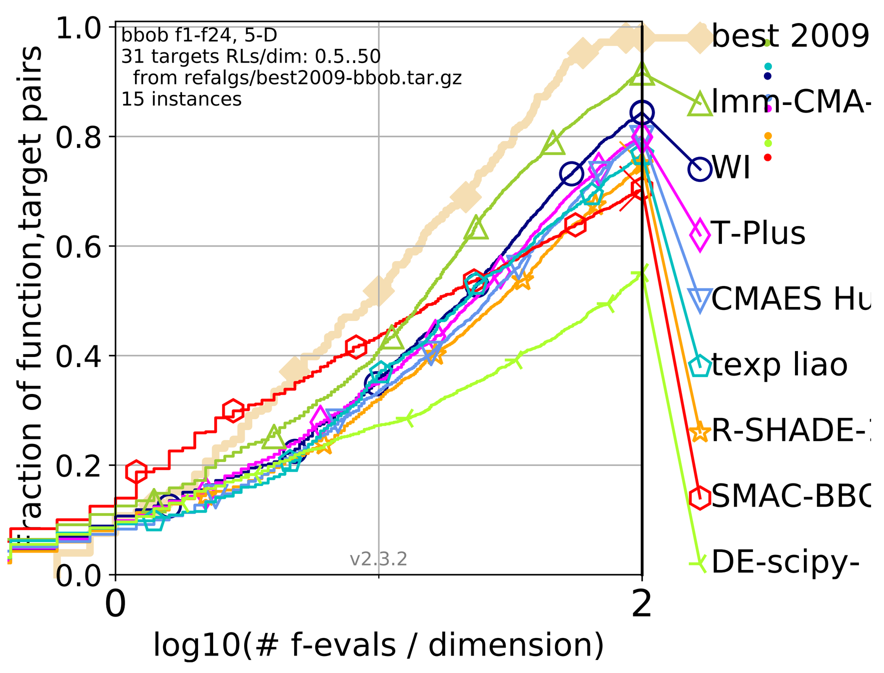
20 dim.



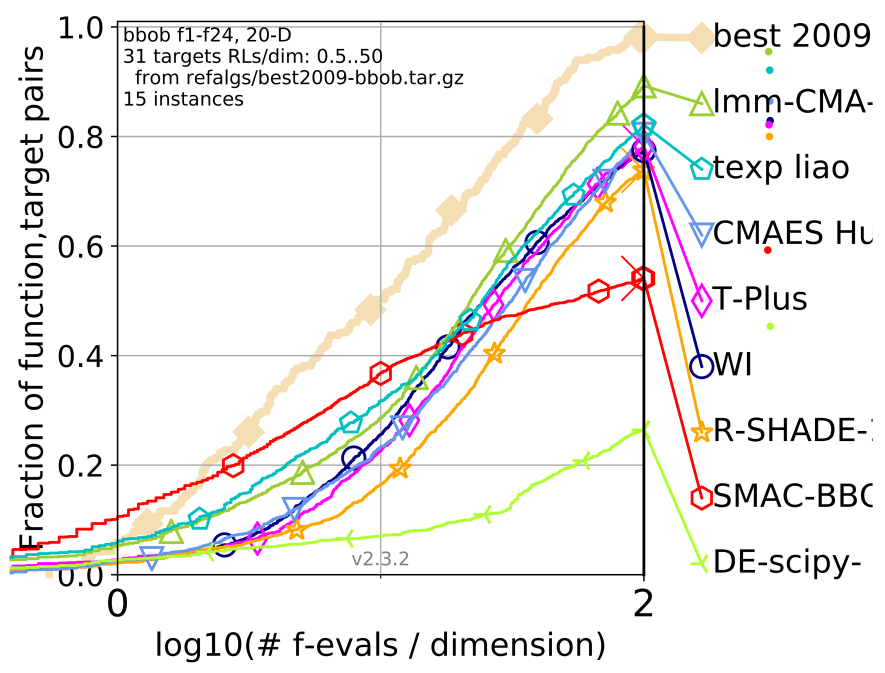
6. Comparison to state-of-the-art optimizers

- Two surrogate model-based optimizers (`SMAC-BBOB` and `lmm-CMA`) perform the best
- For any n, WI and $(\mu + \lambda)$ perform better than a SOTA DE (`R-SHADE-10e2`)
- For $n \leq 10$, WI and $(\mu + \lambda)$ perform better than or similar to `CMAES Hutter`
- For $n \geq 20$, WI and $(\mu + \lambda)$ perform better than `CMAES Hutter` at the early stage
- For $n \geq 20$, WI and $(\mu + \lambda)$ perform significantly worse than `texp liao` at anytime

5 dim.



20 dim.



- `SMAC-BBOB` [Hutter 13] is a Bayesian optimizer (almost EGO). `lmm-CMA` [Auger 13] is a surrogate-assisted CMA-ES
- `CMAES Hutter` [Hutter 13] is a CMA-ES with the default parameters
- `texp liao` [Liao 13] is a CMA-ES with auto-tuned parameters for expensive optimization
- `R-SHADE-10e2` [Tanabe 15] is a SHADE with auto-tuned parameters for expensive optimization
- `DE-scipy` [Varelas 19] is DE from the Python SciPy library