Making Your Research (More) Reproducible
in Optimization, Evolutionary Computation, Metaheuristics, . . .

Manuel López-Ibáñez

manuel.lopez-ibanez@manchester.ac.uk
http://lopez-ibanez.eu

The University of Manchester
Alliance Manchester Business School

SIGEVO Summer School 2022

Optimization as an Empirical Science

Scientific Method

1. Observe a phenomenon
   ☞ EAX crossover shows local optimisation in the TSP

   [Nagata & Kobayashi, 1997]

2. Construct a hypothesis
   ☞ EA + EAX produces better solutions for the TSP than EA + other known crossovers

3. Conduct an experiment

4. Draw conclusion about hypothesis:

   either *provisionally accepted* or *falsified*

   (with some statistical confidence)

Why reproducibility?

- Scientific method (empirical): Falsifiability and community consensus
- Building upon the work of others
  - ☞ Typical first step: reproduce previous results
- Quality control and error correction

What is Reproducibility?

What is Reproducibility?

- Repeat *your own* experiment
  and confirm your previous conclusion?

What is Reproducibility?

- Repeat *your own* experiment
  and confirm your previous conclusion?

- Repeat *someone else's* experiment using their software and data and confirm their conclusion?

What is Reproducibility?

- Repeat *your own* experiment
  and confirm your previous conclusion?
- Repeat *someone else's* experiment using their software and data and confirm their conclusion?
- Repeat someone else's experiment using *your own re-implementation* and confirm their conclusion?

What is Reproducibility?

✘ No consensus in terminology

[Claerbout & Karrenbach, 1992]

[Plesser, 2018]

What is Reproducibility?

✘ No consensus in terminology

[Claerbout & Karrenbach, 1992]

[Plesser, 2018]

- ACM distinguishes between:

Repeatability, *Reproducibility* and *Replicability*

What is Reproducibility?

✘ No consensus in terminology

[Claerbout & Karrenbach, 1992]
[Plesser, 2018]

- ACM distinguishes between:

  Repeatability, *Reproducibility* and *Replicability*

- López-Ibáñez, Branke, and Paquete [2021] define the terms more precisely and distinguish between:

  Repeatability, *Reproducibility*, *Replicability* and *Generalisability*

Terminology

Artifact [ACM, 2020]

"A digital object that was either created by the authors to be used as part of the study or generated by the experiment itself"

algorithm implementations, benchmark instances,
data pre/post-processing scripts, . . .

Terminology

Artifact [ACM, 2020]

"A digital object that was either created by the authors to be used as part of the study or generated by the experiment itself"

algorithm implementations, benchmark instances,
data pre/post-processing scripts, ...

Measurement [López-Ibáñez, Branke, and Paquete, 2021]

"data that results from an experiment"

- measures of quality, computational effort, etc.
- NOT summary statistics

ACM's Terminology

[ACM, 2020]

Repeatability (Same team, same experimental setup)

Reproducibility (Different team, same experimental setup)

Replicability (Different team, different experimental setup)

ACM's Terminology

[ACM, 2020]

Association for Computing Machinery

Repeatability (Same team, same experimental setup)

> *"The measurement can be obtained with stated precision by the same team using the same measurement procedure, the same measuring system, under the same operating conditions, in the same location on multiple trials."*

Reproducibility (Different team, same experimental setup)

Replicability (Different team, different experimental setup)

Association for Computing Machinery

Repeatability (Same team, same experimental setup)

Reproducibility (Different team, same experimental setup)

> *The measurement can be obtained with stated precision by a different team using the same measurement procedure, the same measuring system, under the same operating conditions, in the same or a different location on multiple trials.* [...]
>
> *[A]n independent group can obtain the same result using the author's own artifacts.*

Replicability (Different team, different experimental setup)

ACM's Terminology

Association for Computing Machinery

Repeatability (Same team, same experimental setup)

Reproducibility (Different team, same experimental setup)

> “*The measurement can be obtained with stated precision by a different team using the same measurement procedure, the same measuring system, under the same operating conditions, in the same or a different location on multiple trials.* [...]
> *[A]n independent group can obtain the same result using the author's own artifacts.*”

Replicability (Different team, different experimental setup)

> “*The measurement can be obtained with stated precision by a different team, a different measuring system, in a different location on multiple trials.*[...]
> *[A]n independent group can obtain the same result using artifacts which they develop completely independently.*”

Dimensions of reproducibility

- *Artifacts:* Re-use of the original artifacts should allow to repeat the exact same experiment described in the original publication
- *Random factor:*
  - The experiment evaluates a random sample
  - The experimental claim applies to a range or probability distribution
  - Random seeds
- *Fixed factor:*
  - The experiment evaluates specific chosen values
  - The experimental claim is supported only for those specific values
  - Parameter settings, benchmark problems, computational budget

  . . . *unless randomized*

Proposed terminology

of reproducibility studies

[López-Ibáñez, Branke, and Paquete, 2021]

Label	**Artifacts**	**Random factors**	**Fixed factors**	**Purpose of the study**
Repeatability	Original	Original	Original	Exactly repeat the original experiment, generating precisely the same results.
Reproducibility	Original	New	Original	Test whether the original results were dependent on specific values of random factors and, hence, only a statistical anomaly.
Replicability	New	New	Original	Test whether it is possible to independently reach the same conclusion without relying on original artifacts.
Generalisability	Original or New	New	New	Test whether the conclusion extends beyond the experimental setup of the original paper. When new artifacts are used, generalisability should come after a replicability study.

Artifacts

- Permanently accessible — ACM badge *Artifacts Available*
- Complete — ACM badge *Artifacts Evaluated*
- Re-usable — ACM badge *Artifacts Reusable*

Rule-of-thumb heuristic

A person who only has access to the published paper and the artifacts provided should be able to reproduce the results shown in the paper without having to contact the original authors

Artifacts

- Permanently accessible: ACM badge *Artifacts Available*
  - ✘ Personal / research group webpages or repositories

Artifacts

- Permanently accessible:

ACM badge *Artifacts Available*

  - ✘ Personal / research group webpages or repositories
  - ✘ Development repository GitHub / GitLab / Bitbucket
  - ✔ Git tag or SHA commit

    https://github.com/NEO-Research-Group/irace-sumo/tree/62304739940199b3326cf8b34837c540cad6a68d

Artifacts

- Permanently accessible: ACM badge *Artifacts Available*

  ✘ Personal / research group webpages or repositories

  ✘ Development repository GitHub / GitLab / Bitbucket

  ✔ Git tag or SHA commit
  https://github.com/NEO-Research-Group/irace-sumo/tree/62304739940199b3326cf8b34837c540cad6a68d

  ✔✔ figshare credit for all your research zenodo OSF

  ✔✔ Digital object identifier (DOI)

doi:10.5281/zenodo.4500973

Artifacts: Complete

✔ All source code and input data — ACM badge *Artifacts Evaluated*

- Use different folders for different purposes:

  `Pre-processing/` `Algorithm/`
  `Analysis/` `Presentation/`

Artifacts: Complete

✔ All source code and input data

ACM badge *Artifacts Evaluated*

- Use different folders for different purposes:

  `Pre-processing/` `Algorithm/`
  `Analysis/` `Presentation/`

✔✔ Step-by-step documentation and flexible reproduction scripts

- Use scripts (e.g., bash, PowerShell, R, Python . . .) to encode steps
- Separate intermediate steps in different scripts

  `1-preprocess.sh` `2-run-experiment.sh`
  `3-analysis.R` `4-plots.py`

- Dynamic documentation and reproducible notebooks:

  Rmarkdown, Knitr, Jupyter notebooks

Artifacts: Complete

✔ All source code and input data

ACM badge *Artifacts Evaluated*

- Use different folders for different purposes:

  `Pre-processing/` `Algorithm/`
  `Analysis/` `Presentation/`

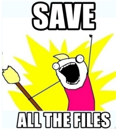
✔✔ Step-by-step documentation and flexible reproduction scripts

- Use scripts (e.g., bash, PowerShell, R, Python . . .) to encode steps
- Separate intermediate steps in different scripts

  `1-preprocess.sh` `2-run-experiment.sh`
  `3-analysis.R` `4-plots.py`

- Dynamic documentation and reproducible notebooks:

  Rmarkdown, Knitr, Jupyter notebooks



✔✔✔ Raw intermediate data, generated data (*decision vectors*), *random seeds*, tables and plots . . .

Artifacts: Re-usable

ACM badge *Artifacts Reusable*

- ✔ Detailed documentation of the code:
  - Use document generators: Sphinx, Doxygen, Roxygen2 . . .

Artifacts: Re-usable

ACM badge *Artifacts Reusable*

- ✔ Detailed documentation of the code:
  - Use document generators: Sphinx, Doxygen, Roxygen2 . . .
- ✔ Independent solution checker

Artifacts: Re-usable

ACM badge *Artifacts Reusable*

- ✔ Detailed documentation of the code:
  - Use document generators: Sphinx, Doxygen, Roxygen2 . . .
- ✔ Independent solution checker
- ✔ Open-source license:
  - ☞ Allow re-distribution and modification (GPL, MIT, Apache, etc)
  - ☞ Mentioned prominently (ideally in every source file)

Artifacts: Re-usable

ACM badge *Artifacts Reusable*

- ✔ Detailed documentation of the code:
  - Use document generators: Sphinx, Doxygen, Roxygen2 …
- ✔ Independent solution checker
- ✔ Open-source license:
  - ☞ Allow re-distribution and modification (GPL, MIT, Apache, etc)
  - ☞ Mentioned prominently (ideally in every source file)
- ✔ Open-data formats (✔ CSV ✔ MySQL ✘ Excel, ✘ Oracle, …)

Artifacts: Re-usable

ACM badge *Artifacts Reusable*

- ✔ Detailed documentation of the code:
  - Use document generators: Sphinx, Doxygen, Roxygen2 ...
- ✔ Independent solution checker
- ✔ Open-source license:
  - ☞ Allow re-distribution and modification (GPL, MIT, Apache, etc)
  - ☞ Mentioned prominently (ideally in every source file)
- ✔ Open-data formats (✔ CSV ✔ MySQL ✘ Excel, ✘ Oracle, ...)
- ✔ Testsuite:
  - ☞ Ensure that the results are correct if someone else repeats the experiment

Zenodo Example

https://doi.org/10.5281/zenodo.4500973

zenodo | Search | Upload | Communities | Log in | Sign up

February 4, 2021 — Conference paper — Open Access

Unbalanced Mallows Models for Optimizing Expensive Black-Box Permutation Problems

Irurozki, Ekhiñe; López-Ibáñez, Manuel

Reproducible Artifacts for the paper:

Ekhine Irurozki and Manuel López-Ibáñez. **Unbalanced Mallows Models for Optimizing Expensive Black-Box Permutation Problems.** In *Genetic and Evolutionary Computation Conference (GECCO '21)*, July 10–14, 2021, Lille, France. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3449639.3459366

Expensive black-box combinatorial optimization problems arise in practice when the objective function is evaluated by means of a simulator or a real-world experiment. Since each fitness evaluation is expensive in terms of time or resources, only a limited number of evaluations is possible, typically several orders of magnitude smaller than in non-expensive problems. In this scenario, classical optimization methods such as mixed-integer programming and local search are not useful. In the continuous case, Bayesian optimization, in particular using Gaussian processes, has proven very effective under these conditions. Much less research is available in the combinatorial case. In this paper, we propose and analyze UMM, an estimation-of-distribution (EDA) algorithm based on a Mallows probabilistic model and unbalanced rank aggregation (uBorda). Experimental results on black-box versions of LOP and PFSP show that UMM is able to match, and sometimes surpass, the solutions obtained by CEGO, a Bayesian optimization algorithm for combinatorial optimization. Moreover, the computational complexity of UMM increases linearly with both the number of function evaluations and the permutation size.

Preview

Publication date:
February 4, 2021

DOI:
DOI 10.5281/zenodo.4500974

Keyword(s):
Combinatorial optimization | Bayesian optimization | Expensive black-box optimization | Estimation of distribution algorithms

License (for files):

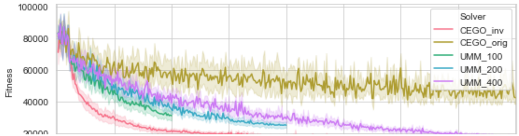
localhost:8888/notebooks/4-analysis.ipynb

jupyter 4-analysis Last Checkpoint: 11/02/2021 (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

```
UMM_100     0           100        1000
UMM_200     0           200        2000
UMM_400     0           400        4000
dtype: int64
optimum:  9000.0
Saving to img/fitness_lop_IO_N-t59d11xx.pdf
```



Summary table of the results

Table with the summary of the results

In [15]:
```
# The maximum time per seed and evaluation is the final time.
dftime = df.groupby(['Solver','Problem','instance','seed']).run_time.max().reset_index()
dftime
# Then we calculate the mean per instance.
dftime['run_time'] =  (dftime['run_time'] / 60.0).round(1)
dftime = dftime.pivot_table(index=['Problem','instance'], columns='Solver', values='run_time').reset_index()
```

Detailed experimental conditions

✘ Results are often sensitive to computational platform:

Detailed experimental conditions

✘ Results are often sensitive to computational platform:

✔ Report relevant hardware details:

- CPU model and clock frequency — *actual speed depends on both!*
- Cache type and memory — *often much more important than RAM!*
- Version of OS, software packages and libraries — *Linux's `pow()` bug 13932*
- Compilation options — *affect runtime and floating-point calculations*

Detailed experimental conditions

✘ Results are often sensitive to computational platform:

✔ Report relevant hardware details:

- CPU model and clock frequency *actual speed depends on both!*
- Cache type and memory *often much more important than RAM!*
- Version of OS, software packages and libraries *Linux's `pow()` bug 13932*
- Compilation options *affect runtime and floating-point calculations*

✔✔ Provide *calibration benchmark* running times

☞ Calibration benchmark: independent, simple, publicly available and deterministic code for the particular problem domain.

Detailed experimental conditions

✘ Results are often sensitive to computational platform:

✔ Report relevant hardware details:

- CPU model and clock frequency — *actual speed depends on both!*
- Cache type and memory — *often much more important than RAM!*
- Version of OS, software packages and libraries — *Linux's `pow()` bug 13932*
- Compilation options — *affect runtime and floating-point calculations*

✔✔ Provide *calibration benchmark* running times

☞ Calibration benchmark: independent, simple, publicly available and deterministic code for the particular problem domain.

✔✔✔ Virtual machines, containers docker
experimental platforms CODE OCEAN OSF

- ✔ *Repeatable* computational environment
- ✘ May be difficult to *replicate*
- ✘ Are these artifacts?

Detailed experimental conditions

✘ Hidden / unfair / biased parameter tuning:

"we use the default parameter settings given by . . . "
"we found these parameter settings after preliminary experiments"

Detailed experimental conditions

- ✘ Hidden / unfair / biased parameter tuning:

> "we use the default parameter settings given by ..."
> "we found these parameter settings after preliminary experiments"

- ✘ Avoid over-tuning

Detailed experimental conditions

- ✘ Hidden / unfair / biased parameter tuning:

> "we use the default parameter settings given by ..."
> "we found these parameter settings after preliminary experiments"

- ✘ Avoid over-tuning
- ✔ Report: effort, domains, tuning vs. testing problem instances

Detailed experimental conditions

✘ Hidden / unfair / biased parameter tuning:

> "we use the default parameter settings given by . . . "
> "we found these parameter settings after preliminary experiments"

✘ Avoid over-tuning

✔ Report: effort, domains, tuning vs. testing problem instances

✔✔ Parameter tuning procedure should be reproducible:

- Design of Experiments [Montgomery, 2012]
- Automatic configuration tools:
  - *irace* [López-Ibáñez et al., 2016], SPOT [Bartz-Beielstein et al., 2010b], Optuna [Akiba et al., 2019] . . .
  - Tutorial: *Automated Algorithm Configuration and Design* https://doi.org/10.1145/3449726.3461404

Measure and report with reproducibility in mind

- What is the claim that you are trying to support ?
  - Low level: Implementation A finds better solutions on average when running T seconds than implementation B on problem instance X using machine Z ?
  - High level: Or algorithm A finds the optimal in fewer evaluations on average than algorithm B on problems of type X ?

☞ Measure and report according to the level of abstraction!

[McGeoch, 2012]

Measure and report with reproducibility in mind

- What is the claim that you are trying to support ?
  - Low level: Implementation A finds better solutions on average when running T seconds than implementation B on problem instance X using machine Z ?
  - High level: Or algorithm A finds the optimal in fewer evaluations on average than algorithm B on problems of type X ?

☞ Measure and report according to the level of abstraction!

[McGeoch, 2012]

✘ Excessive precision or confidence

Example: reporting sub-second runtimes when the noise / variance is larger than a second

Measure and report with reproducibility in mind

- What is the claim that you are trying to support ?
  - Low level: Implementation A finds better solutions on average when running T seconds than implementation B on problem instance X using machine Z ?
  - High level: Or algorithm A finds the optimal in fewer evaluations on average than algorithm B on problems of type X ?

☞ Measure and report according to the level of abstraction!

[McGeoch, 2012]

✘ Excessive precision or confidence

Example: reporting sub-second runtimes when the noise / variance is larger than a second

- In EC, most claims are statistical inferences

✔ Mean and *variances*

✔✔ Confidence intervals, p-values and size effects estimates
Cohen [1995], Lilja [2000], Bartz-Beielstein [2006], Shilane et al. [2008], Bartz-Beielstein et al. [2010a], McGeoch [2012], Derrac et al. [2011], Buzdalov [2019], Bartz-Beielstein et al. [2020] ...

A checklist for reproducibility (1): Artifacts

- ☐ Long-term (permanently) accessible repository ~~Personal website~~
- ☐ Permanent link / DOI to specific version ~~GitHub repo~~
- ☐ Step-by-step documentation to reproduce the experiment AND analysis
- ☐ All source code
- ☐ All input data: problem instances, random seeds, . . .
- ☐ Analysis and presentation scripts
- ☐ Raw generated data (objective and decision vectors)
- ☐ Solution checker
- ☐ Calibration code and its runtime
- ☐ Testsuite
- ☐ Open-source license (reading, distributing, running and reusing)
- ☐ Open-data formats (✔ CSV ✔ MySQL ✘ Excel, ✘ Oracle, etc.)

A checklist for reproducibility (2) Report / Document

- ☐ Relevant hardware details (CPU details, memory / cache sizes)
  - ☐ Provide a container (e.g., Docker)
  - ☐ Provide link to virtual platform (e.g., Code Ocean)
  - ☐ Provide reviewer access to special hardware (e.g., GPUs)
- ☐ Precise versions of any additional software, packages, simulators, compilers / interpreters, and OS
- ☐ (Hyper-)parameters, including types and domains
- ☐ Parameter tuning process (must be reproducible)
- ☐ Separate problem instances for development/tuning and for benchmarking / hypothesis testing (avoid over-tuning)
- ☐ Statistical inference procedure details
  variances, test type, confidence, effect sizes . . .
- ☐ Confidence intervals (or p-values)

Learning more

Reproducibility in Evolutionary Computation.
Manuel López-Ibáñez, Juergen Branke and Luís Paquete.
ACM Transactions on Evolutionary Learning and Optimization, 1(4):1–21, 2021.
http://doi.org/10.1145/3466624
https://arxiv.org/abs/2102.03380

Questions

References I

ACM. Artifact review and badging version 1.1. https://www.acm.org/publications/policies/artifact-review-and-badging-current, Aug. 2020.

T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. In Teredesai et al., editors, *25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2623—-2631. ACM Press, New York, NY, July 2019. doi: 10.1145/3292500.3330701.

T. Bartz-Beielstein. *Experimental Research in Evolutionary Computation: The New Experimentalism*. Springer, Berlin, Germany, 2006.

T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss, editors. *Experimental Methods for the Analysis of Optimization Algorithms*. Springer, Berlin, Germany, 2010a.

T. Bartz-Beielstein, O. Flasch, P. Koch, and W. Konen. SPOT: A toolbox for interactive and automatic tuning in the R environment. In *Proceedings 20. Workshop Computational Intelligence*, pages 264–273, Karlsruhe, 2010b. KIT Scientific Publishing.

T. Bartz-Beielstein, C. Doerr, D. van den Berg, J. Bossek, S. Chandrasekaran, T. Eftimov, A. Fischbach, P. Kerschke, W. La Cava, M. López-Ibáñez, K. M. Malan, J. H. Moore, B. Naujoks, P. Orzechowski, V. Volz, M. Wagner, and T. Weise. Benchmarking in optimization: Best practice and open issues. *Arxiv preprint arXiv:2007.03488 [cs.NE]*, 2020. URL https://arxiv.org/abs/2007.03488.

M. Buzdalov. Towards better estimation of statistical significance when comparing evolutionary algorithms. In M. López-Ibáñez, A. Auger, and T. Stützle, editors, *Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO Companion 2019*, pages 1782–1788. ACM Press, New York, NY, 2019. ISBN 978-1-4503-6748-6. doi: 10.1145/3319619.3326899.

J. Claerbout and M. Karrenbach. Electronic documents give reproducible research a new meaning. In *SEG Technical Program Expanded Abstracts 1992*, pages 601–604. Society of Exploration Geophysicists, 1992. doi: 10.1190/1.1822162.

P. R. Cohen. *Empirical Methods for Artificial Intelligence*. MIT Press, Cambridge, MA, 1995.

J. Derrac, S. García, D. Molina, and F. Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18, 2011.

D. J. Lilja. *Measuring Computer Performance: A Practitioner's Guide*. Cambridge University Press, 2000. doi: 10.1017/CBO9780511612398.

M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, T. Stützle, and M. Birattari. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016. doi: 10.1016/j.orp.2016.09.002.

References II

M. López-Ibáñez, J. Branke, and L. Paquete. Reproducibility in evolutionary computation. *ACM Transactions on Evolutionary Learning and Optimization*, 1(4):1–21, 2021. doi: 10.1145/3466624.

C. C. McGeoch. *A Guide to Experimental Algorithmics*. Cambridge University Press, 2012.

D. C. Montgomery. *Design and Analysis of Experiments*. John Wiley & Sons, New York, NY, 8th edition, 2012.

Y. Nagata and S. Kobayashi. Edge assembly crossover: A high-power genetic algorithm for the traveling salesman problem. In T. Bäck, editor, *ICGA*, pages 450–457. Morgan Kaufmann Publishers, San Francisco, CA, 1997.

H. E. Plesser. Reproducibility vs. replicability: A brief history of a confused terminology. *Frontiers in Neuroinformatics*, 11, Jan. 2018. doi: 10.3389/fninf.2017.00076.

D. Shilane, J. Martikainen, S. Dudoit, and S. J. Ovaska. A general framework for statistical performance comparison of evolutionary computation algorithms. *Information Sciences*, 178(14):2870–2879, 2008. doi: 10.1016/j.ins.2008.03.007.

Making Your Research (More) Reproducible

in Optimization, Evolutionary Computation, Metaheuristics, ...

Manuel López-Ibáñez

manuel.lopez-ibanez@manchester.ac.uk
http://lopez-ibanez.eu

MANCHESTER 1824
The University of Manchester
Alliance Manchester Business School

SIGEVO Summer School 2022

GECCO