合肥大學
HEFEI UNIVERSITY

Solving the Space Optimization Competition 4 Challenge 2 of the European Space Agency

— Thomas Weise, Sina Abdipoor, Zhize Wu, and Mingxuan Li —

Thomas Weise (汤卫思)
tweise@hfuu.edu.cn

School of Artificial Intelligence and Big Data
Hefei University
Hefei, Anhui, China

人工智能与大数据学院
合肥大学
中国安徽省合肥市

Version: 2026-06-29

Introduction

- The Space Optimization Competition (SpOC) is organized by the European Space Agency (ESA) at the Genetic and Evolutionary Computation Conference (GECCO).

Thomas Weise (汤卫思), Sina Abdipoor (思纳), Mingxuan Li (李明轩), and Zhize Wu (吴志泽). "Solving the SpOC 4 Challenge 2 with a Multi-Step Permutation Encoding for the Keplerian Traveling Salesperson". In: *Genetic and Evolutionary Computation Conference Companion (GECCO'2026)*. July 13, 2026–July 17, 2016, San José, Costa Rica. Association for Computing Machinery (ACM), 2026. ISBN: 979-8-4007-2488-6. doi:10.1145/3795101.3815557

Introduction

- The Space Optimization Competition (SpOC) is organized by the European Space Agency (ESA) at the Genetic and Evolutionary Computation Conference (GECCO).
- It poses optimization problems and challenges from space exploration and travel.

Thomas Weise (汤卫思), Sina Abdipoor (思纳), Mingxuan Li (李明轩), and Zhize Wu (吴志泽). "Solving the SpOC 4 Challenge 2 with a Multi-Step Permutation Encoding for the Keplerian Traveling Salesperson". In: *Genetic and Evolutionary Computation Conference Companion (GECCO'2026)*. July 13, 2026–July 17, 2016, San José, Costa Rica. Association for Computing Machinery (ACM), 2026. ISBN: 979-8-4007-2488-6. doi:10.1145/3795101.3815557

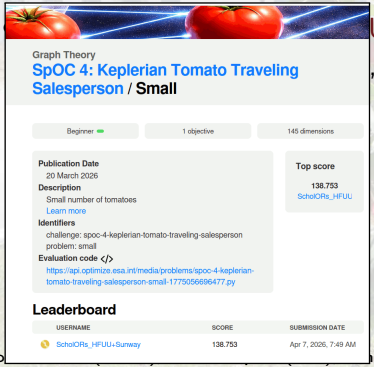
Introduction

- The Space Optimization Competition (SpOC) is organized by the European Space Agency (ESA) at the Genetic and Evolutionary Computation Conference (GECCO).
- It poses optimization problems and challenges from space exploration and travel.
- We are the ScholORs_HFUU+Sunway team, a cooperation between researchers from Hefei University (合肥大学), China and Sunway University, Malaysia.

Thomas Weise (汤卫思), Sina Abdipoor (思纳), Mingxuan Li (李明轩), and Zhize Wu (吴志泽). "Solving the SpOC 4 Challenge 2 with a Multi-Step Permutation Encoding for the Keplerian Traveling Salesperson". In: *Genetic and Evolutionary Computation Conference Companion (GECCO'2026)*. July 13, 2026–July 17, 2016, San José, Costa Rica. Association for Computing Machinery (ACM), 2026. ISBN: 979-8-4007-2488-6. doi:10.1145/3795101.3815557

Introduction

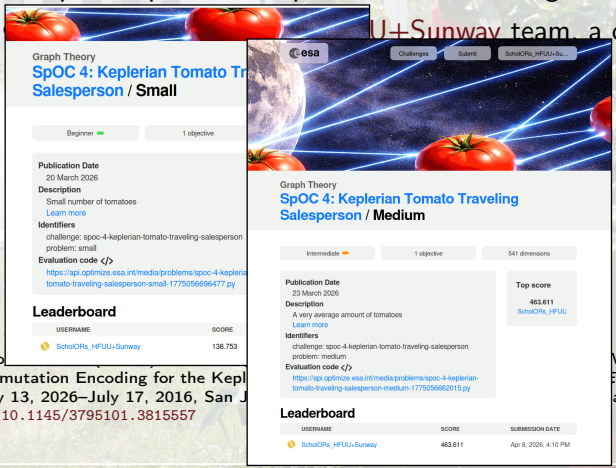
- The Space Optimization Competition (SpOC) is organized by the European Space Agency (ESA) at the Genetic and Evolutionary Computation Conference (GECCO).
- It poses optimization problems and challenges from space exploration and travel.
- U+Sunway team, a cooperation between researchers from , China and Sunway University, Malaysia.



Tho gxuan Li (李明轩), and Zhize Wu (吴志泽). "Solving the SpOC 4 Challenge 2 with a Multi-Step Permutation Encoding for the Keplerian Traveling Salesperson". In: *Genetic and Evolutionary Computation Conference Companion (GECCO'2026)*. July 13, 2026–July 17, 2016, San José, Costa Rica. Association for Computing Machinery (ACM), 2026. ISBN: 979-8-4007-2488-6. doi:10.1145/3795101.3815557

Introduction

- The Space Optimization Competition (SpOC) is organized by the European Space Agency (ESA) at the Genetic and Evolutionary Computation Conference (GECCO).
- It poses optimization problems and challenges from space exploration and travel.
- U+Sunway team, a cooperation between researchers from University, Malaysia.

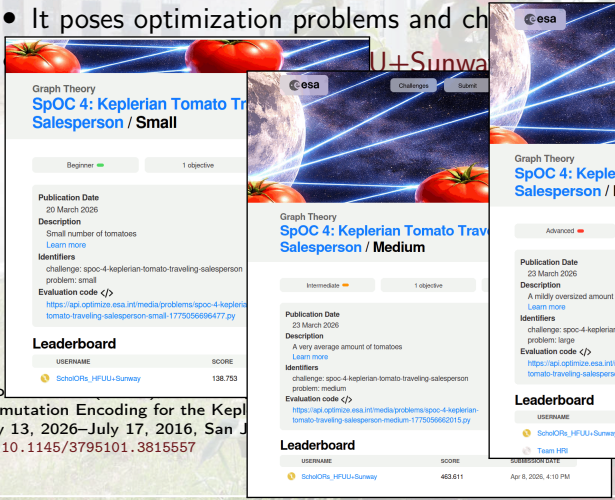


Tho
Permutation Encoding for the Kepl
July 13, 2026–July 17, 2016, San J
doi:10.1145/3795101.3815557

Vu (吴志泽). "Solving the SpOC 4 Challenge 2 with a Multi-Step
Evolutionary Computation Conference Companion (GECCO'2026).
achinery (ACM), 2026. ISBN: 979-8-4007-2488-6.

Introduction

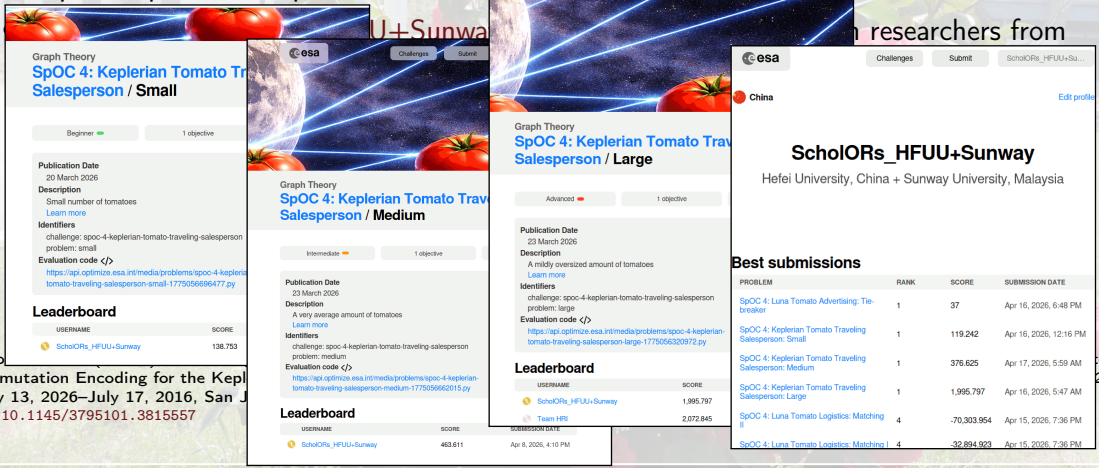
- The Space Optimization Competition (SpOC) is organized by the European Space Agency (ESA) at the Genetic and Evolutionary Computation Conference (GECCO).
- It poses optimization problems and ch n and travel.
- researchers from .



Tho C 4 Challenge 2 with a Multi-Step Permutation Encoding for the Kepl erence Companion (GECCO'2026). July 13, 2026–July 17, 2016, San J 979-8-4007-2488-6. doi:10.1145/3795101.3815557

Introduction

- The Space Optimization Competition (SpOC) is organized by the European Space Agency (ESA) at the Genetic and Evolutionary Computation Conference (GECCO).
- It poses optimization problems and ch n and travel.



researchers from

...

Tho ep
Permutation Encoding for the Kepl 26).
July 13, 2026–July 17, 2016, San J
doi:10.1145/3795101.3815557

The Challenge

- Given are n objects orbiting our moon with their orbital data.

The Challenge

- Given are n objects orbiting our moon with their orbital data.
- There are three instances **small**, **medium**, and **large**, with $n = 49$, 181, and 1051.

The Challenge

- Given are n objects orbiting our moon with their orbital data.
- There are three instances **small**, **medium**, and **large**, with $n = 49$, 181, and 1051.
- The goal is to visit each of the objects exactly once as fast as possible, starting at an object of choice.

The Challenge

- Given are n objects orbiting our moon with their orbital data.
- There are three instances **small**, **medium**, and **large**, with $n = 49$, 181, and 1051.
- The goal is to visit each of the objects exactly once as fast as possible, starting at an object of choice.
- For each transfer, our spacecraft may apply a specific change ΔV of velocity.

The Challenge

- Given are n objects orbiting our moon with their orbital data.
- There are three instances **small**, **medium**, and **large**, with $n = 49$, 181, and 1051.
- The goal is to visit each of the objects exactly once as fast as possible, starting at an object of choice.
- For each transfer, our spacecraft may apply a specific change ΔV of velocity.
- ΔV is limited by ΔV_{max}, but 5 times we are allowed to do a larger velocity change $\Delta V \leq \Delta V_{max}^{exception}$.

The Challenge

- Given are n objects orbiting our moon with their orbital data.
- There are three instances **small**, **medium**, and **large**, with $n = 49$, 181, and 1051.
- The goal is to visit each of the objects exactly once as fast as possible, starting at an object of choice.
- For each transfer, our spacecraft may apply a specific change ΔV of velocity.
- ΔV is limited by ΔV_{max}, but 5 times we are allowed to do a larger velocity change $\Delta V \leq \Delta V_{max}^{exception}$.
- Limits for the shortest transfer and maximum total time are also given.

Computing Transfers

- We begin by developing a tool $opt\Delta V$ that finds the earliest transfer from one object to another one within a given time window $[t_s, t_d]$, given a ΔV-limit ΔV_{max}^{lim}.

Computing Transfers

- We begin by developing a tool $opt\Delta V$ that finds the earliest transfer from one object to another one within a given time window $[t_s, t_d]$, given a ΔV-limit ΔV_{max}^{lim}.
- If no such transfer exists, the function returns and error.

Computing Transfers

- We begin by developing a tool $opt\Delta V$ that finds the earliest transfer from one object to another one within a given time window $[t_s, t_d]$, given a ΔV-limit ΔV_{max}^{lim}.
- If no such transfer exists, the function returns and error.
- We approach this as numerical minimization problem with objective function ψ:

Computing Transfers

- We begin by developing a tool $opt\Delta V$ that finds the earliest transfer from one object to another one within a given time window $[t_s, t_d]$, given a ΔV-limit ΔV_{max}^{lim}.
- If no such transfer exists, the function returns and error.
- We approach this as numerical minimization problem with objective function ψ:

$$\psi(t_a, t_b) = \begin{cases} \Delta V & \text{if } \Delta V > \Delta V_{max}^{lim} \\ -1/t_b & \text{otherwise} \end{cases} \tag{1}$$

- If the transfer at times $\{t_a, t_b\}$ is infeasible, the positive ΔV value is returned and can further be minimized until $\Delta V \leq \Delta V_{max}^{lim}$ is reached.

Computing Transfers

- We begin by developing a tool $opt\Delta V$ that finds the earliest transfer from one object to another one within a given time window $[t_s, t_d]$, given a ΔV-limit ΔV_{max}^{lim}.
- If no such transfer exists, the function returns and error.
- We approach this as numerical minimization problem with objective function ψ:

$$\psi(t_a, t_b) = \begin{cases} \Delta V & \text{if } \Delta V > \Delta V_{max}^{lim} \\ -1/t_b & \text{otherwise} \end{cases} \tag{1}$$

- If the transfer at times $\{t_a, t_b\}$ is infeasible, the positive ΔV value is returned and can further be minimized until $\Delta V \leq \Delta V_{max}^{lim}$ is reached.
- Once we found such transfer, $-1/t_b$ minimizes the arrival time t_b.

Computing Transfers

- We begin by developing a tool $opt\Delta V$ that finds the earliest transfer from one object to another one within a given time window $[t_s, t_d]$, given a ΔV-limit ΔV_{max}^{lim}.
- If no such transfer exists, the function returns and error.
- We approach this as numerical minimization problem with objective function ψ:

$$\psi(t_a, t_b) = \begin{cases} \Delta V & \text{if } \Delta V > \Delta V_{max}^{lim} \\ -1/t_b & \text{otherwise} \end{cases} \tag{1}$$

- If the transfer at times $\{t_a, t_b\}$ is infeasible, the positive ΔV value is returned and can further be minimized until $\Delta V \leq \Delta V_{max}^{lim}$ is reached.
- Once we found such transfer, $-1/t_b$ minimizes the arrival time t_b.
- We use a variety of algorithms, including Nelder&Mead, BFGS, and CMA-ES.

Computing Transfers

- We begin by developing a tool $opt\Delta V$ that finds the earliest transfer from one object to another one within a given time window $[t_s, t_d]$, given a ΔV-limit ΔV_{max}^{lim}.
- If no such transfer exists, the function returns and error.
- We approach this as numerical minimization problem with objective function ψ:

$$\psi(t_a, t_b) = \begin{cases} \Delta V & \text{if } \Delta V > \Delta V_{max}^{lim} \\ -1/t_b & \text{otherwise} \end{cases} \tag{1}$$

- If the transfer at times $\{t_a, t_b\}$ is infeasible, the positive ΔV value is returned and can further be minimized until $\Delta V \leq \Delta V_{max}^{lim}$ is reached.
- Once we found such transfer, $-1/t_b$ minimizes the arrival time t_b.
- We use a variety of algorithms, including Nelder&Mead, BFGS, and CMA-ES.
- If a feasible transfer with end time t' is found, we search again in $[t_s, t')$ until the search fails, retaining and returning the best result.

Infeasible Transfers and Preprocessing

- We first assemble a transfer matrix $\mathcal{T}$ for each instance, telling us whether we can travel from object to another either *not at all*, with excess velocity change, or with a normal velocity change.

Infeasible Transfers and Preprocessing

- We first assemble a transfer matrix $\mathcal{T}$ for each instance, telling us whether we can travel from object to another either *not at all*, with excess velocity change, or with a normal velocity change.
- The matrices $\mathcal{T}$ are sparse, most transfers are infeasible!

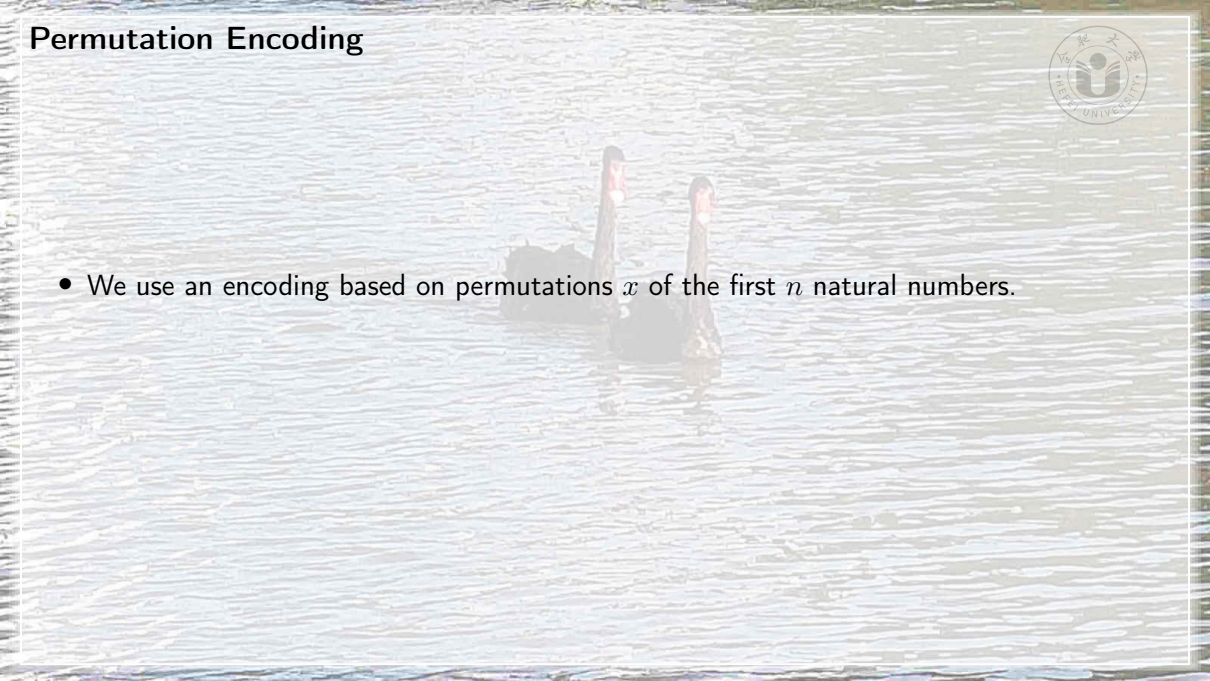
Infeasible Transfers and Preprocessing

- We first assemble a transfer matrix $\mathcal{T}$ for each instance, telling us whether we can travel from object to another either *not at all*, with excess velocity change, or with a normal velocity change.
- The matrices $\mathcal{T}$ are sparse, most transfers are infeasible!
- On instance **small**, only 2.8 other objects can be reached with $\Delta V \leq \Delta V_{max}$ in average!

Infeasible Transfers and Preprocessing

- We first assemble a transfer matrix $\mathcal{T}$ for each instance, telling us whether we can travel from object to another either *not at all*, with excess velocity change, or with a normal velocity change.
- The matrices $\mathcal{T}$ are sparse, most transfers are infeasible!
- On instance **small**, only 2.8 other objects can be reached with $\Delta V \leq \Delta V_{max}$ in average!
- The vast majority of visit sequences will not be feasible, so always using $opt\Delta V$ would be a huge waste of runtime...

Permutation Encoding



- We use an encoding based on permutations x of the first n natural numbers.

Permutation Encoding

- We use an encoding based on permutations x of the first n natural numbers.
- A permutation x is decoded to a solution vector y with the actual travel times using a three-stage decoding function γ.

Permutation Encoding: Stage 1

- First, $\gamma(x)$ processes x from beginning to end and divides it into *fragments* π using the transfer matrix $\mathcal{T}$.

Permutation Encoding: Stage 1

- First, $\gamma(x)$ processes x from beginning to end and divides it into *fragments* π using the transfer matrix $\mathcal{T}$.
- A fragment π is a sub-sequence of x such that normal ΔV can be used to travel from one object to the next.

Permutation Encoding: Stage 1

- First, $\gamma(x)$ processes x from beginning to end and divides it into *fragments* π using the transfer matrix $\mathcal{T}$.
- A fragment π is a sub-sequence of x such that normal ΔV can be used to travel from one object to the next.
- Fragments that can be stitched together by pre-pending and appending without violating this condition are merged.

Permutation Encoding: Stage 1

- First, $\gamma(x)$ processes x from beginning to end and divides it into *fragments* π using the transfer matrix $\mathcal{T}$.
- A fragment π is a sub-sequence of x such that normal ΔV can be used to travel from one object to the next.
- Fragments that can be stitched together by pre-pending and appending without violating this condition are merged.
- Fragments that can be inserted into each other without violating the condition are merged.

Permutation Encoding: Stage 1

- First, $\gamma(x)$ processes x from beginning to end and divides it into *fragments* π using the transfer matrix $\mathcal{T}$.
- A fragment π is a sub-sequence of x such that normal ΔV can be used to travel from one object to the next.
- Fragments that can be stitched together by pre-pending and appending without violating this condition are merged.
- Fragments that can be inserted into each other without violating the condition are merged.
- This is repeated with the relaxed requirement to allow excess velocity changes.

Permutation Encoding: Stage 1

- First, $\gamma(x)$ processes x from beginning to end and divides it into *fragments* π using the transfer matrix $\mathcal{T}$.
- A fragment π is a sub-sequence of x such that normal ΔV can be used to travel from one object to the next.
- Fragments that can be stitched together by pre-pending and appending without violating this condition are merged.
- Fragments that can be inserted into each other without violating the condition are merged.
- This is repeated with the relaxed requirement to allow excess velocity changes.
- Finally, all remaining fragments are stitched together regardless, yielding a new permutation x'.

Permutation Encoding: Stage 2

- We use the transfer matrix $\mathcal{T}$ to count the expected constraint violations in x'.

Permutation Encoding: Stage 2

- We use the transfer matrix $\mathcal{T}$ to count the expected constraint violations in x'.
- If constraint violations are expected, the decoding stops — this is very fast, no physics computation needed at all.

Permutation Encoding: Stage 3

- Otherwise, we process the permutation x' from beginning to end computing transfers using the actual physics computation $opt\Delta V$.

Permutation Encoding: Stage 3

- Otherwise, we process the permutation x' from beginning to end computing transfers using the actual physics computation $opt\Delta V$.
- We make sure that each solution uses all 5 permitted excess velocity changes (even if it does not need them), because then we can make travel faster.

Optimization

- We apply a randomized local search (RLS) to minimize the objective function f:

$$f(y) = \begin{cases} p(y) & \text{if } p(y) > 0 \\ -1/t_{tot}(y) & \text{otherwise} \end{cases} \tag{2}$$

Optimization

- We apply a randomized local search (RLS) to minimize the objective function f:

$$f(y) = \begin{cases} p(y) & \text{if } p(y) > 0 \\ -1/t_{tot}(y) & \text{otherwise} \end{cases} \tag{2}$$

- A solution y is infeasible if $p(y) > 0$, in which case the penalty p is returned.

Optimization

- We apply a randomized local search (RLS) to minimize the objective function f:

$$f(y) = \begin{cases} p(y) & \text{if } p(y) > 0 \\ -1/t_{tot}(y) & \text{otherwise} \end{cases} \tag{2}$$

- A solution y is infeasible if $p(y) > 0$, in which case the penalty p is returned.
- For feasible solutions, the total time t_{tot} is minimized.

Summary

- We tested several different algorithm setups implemented in moptipy[17].

Summary

- We tested several different algorithm setups implemented in moptipy[17].
- Our results are obtained with low computational cost on off-the-shelf laptops.

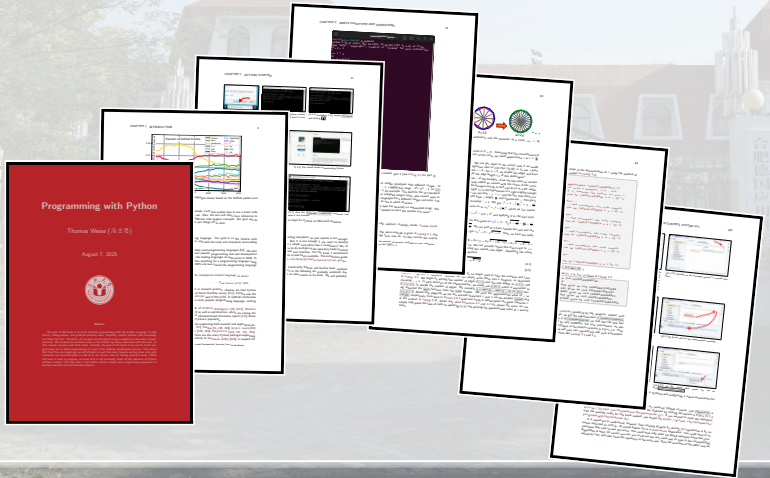
Summary

- We tested several different algorithm setups implemented in moptipy[17].
- Our results are obtained with low computational cost on off-the-shelf laptops.
- An early setup found a feasible solution with $t_{tot} = 2689.4$ for the **large** instance within 187 seconds.

Advertisement

Programming with Python

We have a freely available course book on *Programming with Python* at `https://thomasweise.github.io/programmingWithPython`, with focus on practical software development using the Python ecosystem of tools[15].

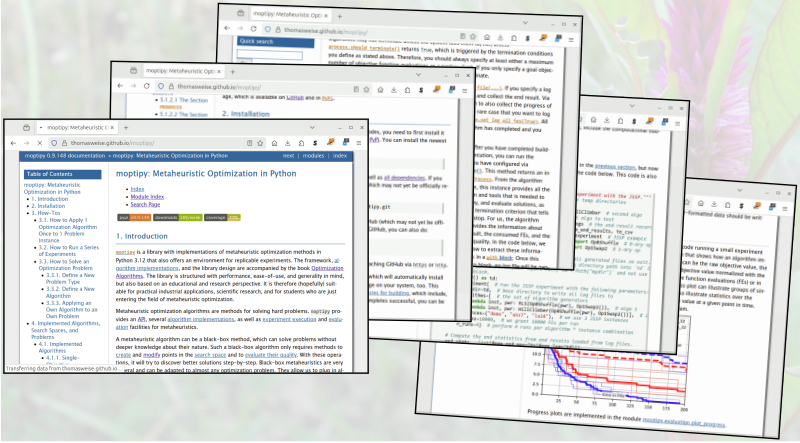


Databases

We have a freely available course book on *Databases* at `https://thomasweise.github.io/databases`, with actual practical examples using a real database management system (DBMS)[13].

Metaheuristic Optimization in Python: moptipy

We offer moptipy[17] a mature open source Python package for metaheuristic optimization, which implements several algorithms, can run self-documenting experiments in parallel and in a distributed fashion, and offers statistical evaluation tools.



谢谢您们！

Thank you!

Vielen Dank!

References I

[1] Thomas Bäck, David B. Fogel, and Zbigniew "Zbyszek" Michalewicz, eds. *Handbook of Evolutionary Computation*. Bristol, England, UK: IOP Publishing Ltd and Oxford, Oxfordshire, England, UK: Oxford University Press, 1997. ISBN: 978-0-7503-0392-7 (cit. on p. 52).

[2] Eduardo Carvalho Pinto and Carola Doerr. *Towards a More Practice-Aware Runtime Analysis of Evolutionary Algorithms*. arXiv.org: Computing Research Repository (CoRR) abs/1812.00493. Ithaca, NY, USA: Cornell Universiy Library, Dec. 3, 2018. doi:10.48550/arXiv.1812.00493. URL: https://arxiv.org/abs/1812.00493 (visited on 2025-08-08). arXiv:1812.00493v1 [cs.NE] 3 Dec 2018 (cit. on p. 52).

[3] Jiayang Chen (陈嘉阳), Zhize Wu (吴志泽), Sarah Louise Thomson, and Thomas Weise (汤卫思). "Frequency Fitness Assignment: Optimization Without Bias for Good Solution Outperforms Randomized Local Search on the Quadratic Assignment Problem". In: *16th International Joint Conference on Computational Intelligence (IJCCI'24)*. Nov. 20–22, 2024, Porto, Portugal. Ed. by Francesco Marcelloni, Kurosh Madani, Niki van Stein, and Joaquim Filipe. Porto, Portugal: SciTePress: Science and Technology Publications, Lda, 2024, pp. 27–37. ISSN: 2184-3236. ISBN: 978-989-758-721-4. doi:10.5220/0012888600003837 (cit. on p. 53).

[4] Stefan Droste, Thomas Jansen, and Ingo Wegener. "On the Analysis of the $(1 + 1)$ Evolutionary Algorithm". *Theoretical Computer Science* 276(1-2):51–81, Apr. 2002. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 0304-3975. doi:10.1016/S0304-3975(01)00182-7 (cit. on p. 52).

[5] John Hunt. *A Beginners Guide to Python 3 Programming*. 2nd ed. Undergraduate Topics in Computer Science (UTICS). Cham, Switzerland: Springer, 2023. ISBN: 978-3-031-35121-1. doi:10.1007/978-3-031-35122-8 (cit. on p. 53).

[6] Kent D. Lee and Steve Hubbard. *Data Structures and Algorithms with Python*. Undergraduate Topics in Computer Science (UTICS). Cham, Switzerland: Springer, 2015. ISBN: 978-3-319-13071-2. doi:10.1007/978-3-319-13072-9 (cit. on p. 53).

[7] Tianyu Liang (梁天宇), Zhize Wu (吴志泽), Jörg Lässig, Daan van den Berg, Sarah Louise Thomson, and Thomas Weise (汤卫思). "Addressing the Traveling Salesperson Problem with Frequency Fitness Assignment and Hybrid Algorithms". *Soft Computing* 28(17-18):9495–9508, July 2024. London, England, UK: Springer Nature Limited. ISSN: 1432-7643. doi:10.1007/S00500-024-09718-8 (cit. on p. 53).

References II

[8] Tianyu Liang (梁天宇), Zhize Wu (吴志泽), Jörg Lässig, Daan van den Berg, and Thomas Weise (汤卫思). "Solving the Traveling Salesperson Problem using Frequency Fitness Assignment". In: *IEEE Symposium Series on Computational Intelligence (SSCI'2022)*. Dec. 4–7, 2022, Singapore. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE), 2022. ISBN: 978-1-6654-8769-6. doi:10.1109/SSCI51031.2022.10022296 (cit. on p. 53).

[9] Tianyu Liang (梁天宇), Zhize Wu (吴志泽), Matthias Thürer, Markus Wagner, and Thomas Weise (汤卫思). "Generating Small Instances with Interesting Features for the Traveling Salesperson Problem". In: *16th International Joint Conference on Computational Intelligence (IJCCI'24)*. Nov. 20–22, 2024, Porto, Portugal. Ed. by Francesco Marcelloni, Kurosh Madani, Niki van Stein, and Joaquim Filipe. Porto, Portugal: SciTePress: Science and Technology Publications, Lda, 2024, pp. 173–180. ISSN: 2184-3236. ISBN: 978-989-758-721-4. doi:10.5220/0012888800003837 (cit. on p. 53).

[10] Mark Lutz. *Learning Python*. 6th ed. Sebastopol, CA, USA: O'Reilly Media, Inc., Mar. 2025. ISBN: 978-1-0981-7130-8 (cit. on p. 53).

[11] Francesco Marcelloni, Kurosh Madani, Niki van Stein, and Joaquim Filipe, eds. *16th International Joint Conference on Computational Intelligence (IJCCI'24)*. Nov. 20–22, 2024, Porto, Portugal. Porto, Portugal: SciTePress: Science and Technology Publications, Lda, 2024. ISSN: 2184-3236. ISBN: 978-989-758-721-4. doi:10.5220/0000195000003837.

[12] Sarah Louise Thomson, Gabriela Ochoa, Daan van den Berg, Tianyu Liang (梁天宇), and Thomas Weise (汤卫思). "Entropy, Search Trajectories, and Explainability for Frequency Fitness Assignment". In: *Parallel Problem Solving from Nature (PPSN XVIII)*. Vol. 1. Sept. 14–18, 2024, Hagenberg, Mühlkreis, Austria. Ed. by Michael Affenzeller, Stephan M. Winkler, Anna V. Kononova, Heike Trautmann, Tea Tušar, Penousal Machado, and Thomas Bäck. Vol. 15148 of Lecture Notes in Computer Science (LNCS). Cham, Switzerland: Springer. ISSN: 0302-9743. ISBN: 978-3-031-70054-5. doi:10.1007/978-3-031-70055-2_23 (cit. on p. 53).

[13] Thomas Weise (汤卫思). *Databases*. Hefei, Anhui, China (中国安徽省合肥市): Hefei University (合肥大学), School of Artificial Intelligence and Big Data (人工智能与大数据学院), 2025. URL: https://thomasweise.github.io/databases (visited on 2025-01-05) (cit. on pp. 46, 52).

[14] Thomas Weise (汤卫思). *Global Optimization Algorithms – Theory and Application*. self-published, 2009. URL: https://www.researchgate.net/publication/200622167 (visited on 2025-07-25) (cit. on p. 52).

[15] Thomas Weise (汤卫思). *Programming with Python*. Hefei, Anhui, China (中国安徽省合肥市): Hefei University (合肥大学), School of Artificial Intelligence and Big Data (人工智能与大数据学院), 2024–2026. URL: https://thomasweise.github.io/programmingWithPython (visited on 2025-01-05) (cit. on pp. 45, 53).

References III

[16] Thomas Weise (汤卫思), Sina Abdipoor (思纳), Mingxuan Li (李明轩), and Zhize Wu (吴志泽). "Solving the SpOC 4 Challenge 2 with a Multi-Step Permutation Encoding for the Keplerian Traveling Salesperson". In: *Genetic and Evolutionary Computation Conference Companion (GECCO'2026)*. July 13, 2026–July 17, 2016, San José, Costa Rica. New York, NY, USA: Association for Computing Machinery (ACM), 2026. ISBN: 979-8-4007-2488-6. doi:10.1145/3795101.3815557 (cit. on pp. 2–8).

[17] Thomas Weise (汤卫思) and Zhize Wu (吴志泽). "Replicable Self-Documenting Experiments with Arbitrary Search Spaces and Algorithms". In: *Conference on Genetic and Evolutionary Computation (GECCO'2023), Companion Volume*. July 15–19, 2023, Lisbon, Portugal. Ed. by Sara Silva and Luís Paquete. New York, NY, USA: Association for Computing Machinery (ACM), 2023, pp. 1891–1899. ISBN: 979-8-4007-0120-7. doi:10.1145/3583133.3596306 (cit. on pp. 41–43, 47, 53).

[18] CAO Xiang (曹翔), Zhize Wu (吴志泽), Daan van den Berg, and Thomas Weise (汤卫思). "Randomized Local Search vs. NSGA-II vs. Frequency Fitness Assignment on The Traveling Tournament Problem". In: *16th International Joint Conference on Computational Intelligence (IJCCI'24)*. Nov. 20–22, 2024, Porto, Portugal. Ed. by Francesco Marcelloni, Kurosh Madani, Niki van Stein, and Joaquim Filipe. Porto, Portugal: SciTePress: Science and Technology Publications, Lda, 2024, pp. 38–49. ISSN: 2184-3236. ISBN: 978-989-758-721-4. doi:10.5220/0012891500003837 (cit. on p. 53).

[19] Kinza Yasar and Craig S. Mullins. *Definition: Database Management System (DBMS)*. Newton, MA, USA: TechTarget, Inc., June 2024. URL: https://www.techtarget.com/searchdatamanagement/definition/database-management-system (visited on 2025-01-11) (cit. on p. 52).

[20] Rui Zhao (赵睿), Tianyu Liang (梁天宇), Zhize Wu (吴志泽), Daan van den Berg, Matthias Thürer, and Thomas Weise (汤卫思). "Randomized Local Search on the 2D Rectangular Bin Packing Problem with Item Rotation". In: *Genetic and Evolutionary Computation Conference (GECCO'2024)*. July 14–18, 2024, Melbourne, VIC, Australia. Ed. by Xiaodong Li and Julia Handl. New York, NY, USA: Association for Computing Machinery (ACM), 2024, pp. 235–238. ISBN: 979-8-4007-0494-9. doi:10.1145/3638530.3654139 (cit. on p. 53).

[21] Rui Zhao (赵睿), Zhize Wu (吴志泽), Daan van den Berg, Matthias Thürer, Tianyu Liang (梁天宇), Ming Tan (檀明), and Thomas Weise (汤卫思). "Randomized Local Search for Two-Dimensional Bin Packing and a Negative Result for Frequency Fitness Assignment". In: *16th International Joint Conference on Computational Intelligence (IJCCI'24)*. Nov. 20–22, 2024, Porto, Portugal. Ed. by Francesco Marcelloni, Kurosh Madani, Niki van Stein, and Joaquim Filipe. Porto, Portugal: SciTePress: Science and Technology Publications, Lda, 2024, pp. 15–26. ISSN: 2184-3236. ISBN: 978-989-758-721-4. doi:10.5220/0012888500003837 (cit. on p. 53).

Glossary I

$(1+1)$ EA The $(1+1)$ EA is a local search algorithm that retains the best solution x_c discovered so far during the search[2,4]. In each step, it applies a unary search operator to this best-so-far solution x_c and derives a new solution x_n. If the new solution x_n is *better or equally good* when compared with x_c, i.e., not worse, then it replaces it, i.e., is stored as the new x_c. If the search space are bit strings of length n, then the $(1+1)$ EA uses a unary search operator that flips each bit independently with probability m/n, where usually $m = 1$. This operator is the main difference to randomized local search (RLS). The $(1+1)$ EA is a special case of the $(\mu+\lambda)$ evolutionary algorithm ($(\mu+\lambda)$ EA) where $\mu = \lambda = 1$.

EA An *evolutionary algorithm* is a metaheuristic optimization method that maintains a population of candidate solutions, which undergo selection (where better solutions are chosen with higher probability) and reproduction (where mutation and recombination create a new candidate solution from one or two existing ones, repectively)[1,14].

$(\mu+\lambda)$ EA The $(\mu+\lambda)$ EA is an evolutionary algorithm (EA) where, in each generation, λ offspring solutions are generated from the current population of μ parent solutions. The offspring and parent populations are merged, yielding $\mu+\lambda$ solutions, from which then the best μ solutions are ratained to form the parent population of the next generation. If the search space is the bit strings of length n, then this algorithm usually applies a mutation operator flipping each bit independently with probability $1/n$.

DB A *database* is an organized collection of structured information or data, typically stored electronically in a computer system. Databases are discussed in our book *Databases*[13].

DBMS A *database management system* is the software layer located between the user or application and the database (DB). The DBMS allows the user/application to create, read, write, update, delete, and otherwise manipulate the data in the DB[19].

ESA The European Space Agency (ESA) is an international organization dedicated to space exploration, research, and technology development. Learn more at https://www.esa.int.

GECCO The Genetic and Evolutionary Computation Conference (GECCO) is one of the largest conferences in the field of EAs. It has been established in 1999 and its proceedings are published by ACM. Learn more at https://sig.sigevo.org/GECCOs.

Glossary II

moptipy is the *Metaheuristic Optimization in Python* library[17]. It has been used in several different research works, including[3,7–9,12,18,20,21]. Learn more at https://thomasweise.github.io/moptipy and https://thomasweise.github.io/moptipyapps.

Python The Python programming language[5,6,10,15], i.e., what you will learn about in our book[15]. Learn more at https://python.org.

RLS Randomized local search retains the best solution x_c discovered so far during the search and, in each step, it applies a unary search operator to this best-so-far solution x_c and derives a new solution x_n. If the new solution x_n is *better or equally good* when compared with x_c, i.e., not worse, then it replaces it, i.e., is stored as the new x_c. If the search space are bit strings of length n, then RLS uses a unary search operator that flips exactly one bit. This operator is the main difference to $(1+1)$ evolutionary algorithm ($(1+1)$ EA).

SpOC The Space Optimization Competition (SpOC) is a competition organized by the Advanced Concepts Team of ESA at GECCO since 2022, where various optimization problems related to space exploration and travel are posed and researchers worldwide can compete. Learn more at https://optimise.esa.int.

$f(y)$ The *objective function* $f : \mathbb{Y} \mapsto \mathbb{R}$ can compute a cost or rating for each candidate solution y in the solution space $\mathbb{Y}$. The smaller this value, the better the solution, i.e., in the context of this work, objective functions are subject to minimization. Objective functions are not necessarily continuous and it is also quite common that they have integer or natural numbers as results, e.g., we often have $f : \mathbb{Y} \mapsto \mathbb{N}_0$ instead of $f : \mathbb{Y} \mapsto \mathbb{R}$..

$\gamma(x)$ The decoding function $\gamma : \mathbb{X} \mapsto \mathbb{Y}$ translates point from the search space $\mathbb{X}$ to one point y in the solution space $\mathbb{Y}$. It is a very common case that $\mathbb{X} = \mathbb{Y}$ and that the decoding function γ is the identity mapping..

$\mathbb{N}_0$ the set of the natural numbers *including* 0, i.e., 0, 1, 2, 3, and so on. It holds that $\mathbb{N}_0 \subset \mathbb{Z}$.

Glossary III

$\mathbb{R}$ the set of the real numbers.

$\mathbb{X}$ The search space $\mathbb{X}$ is the space through which an optimization algorithm navigates in order to find good solutions. The search operators of metaheuristics are used to sample points $x \in \mathbb{X}$, for example. While it is often the case, these are not necessarily the same data structures that the user is interested in. It is also common that the search space $\mathbb{X}$ is a simple and plain data structure that can be tackled with standard operations or mathematical tools, such as real vectors $\mathbb{R}^n$, permutations, or bit strings, whereas the solution space $\mathbb{Y}$ can be something more complicated, like a Gantt chart. Then, the user only cares about the solutions $y \in \mathbb{Y}$ and not about the search space $\mathbb{X}$. A decoding $\gamma : \mathbb{X} \mapsto \mathbb{Y}$ is employed to translate the $x \in \mathbb{X}$ to $y = \gamma(x)$ with $y \in \mathbb{Y}$..

x a point in the search Space $\mathbb{X}$..

$\mathbb{Y}$ the set of candidate solutions of an optimization problem is called the *solution space* $\mathbb{Y}$. The solution space corresponds to the data structure holding the candidate solutions y. These are the elements that are passed to the user after the optimization process is completed. They are also the parameters of the objective function(s) $f : \mathbb{Y} \mapsto \mathbb{R}$..

y a candidate solution $y \in \mathbb{Y}$ is one possible solution to an optimization problem. After the optimization process is completed, one or multiple such solutions are passed to the user..

$\mathbb{Z}$ the set of the integers numbers including positive and negative numbers and 0, i.e., . . . , -3, -2, -1, 0, 1, 2, 3, . . . , and so on. It holds that $\mathbb{Z} \subset \mathbb{R}$.