ETH zürich

DITET

Particle Swarm Optimization (PSO) – A Performance Study

Optimization Methods for Engineers – FS 2021

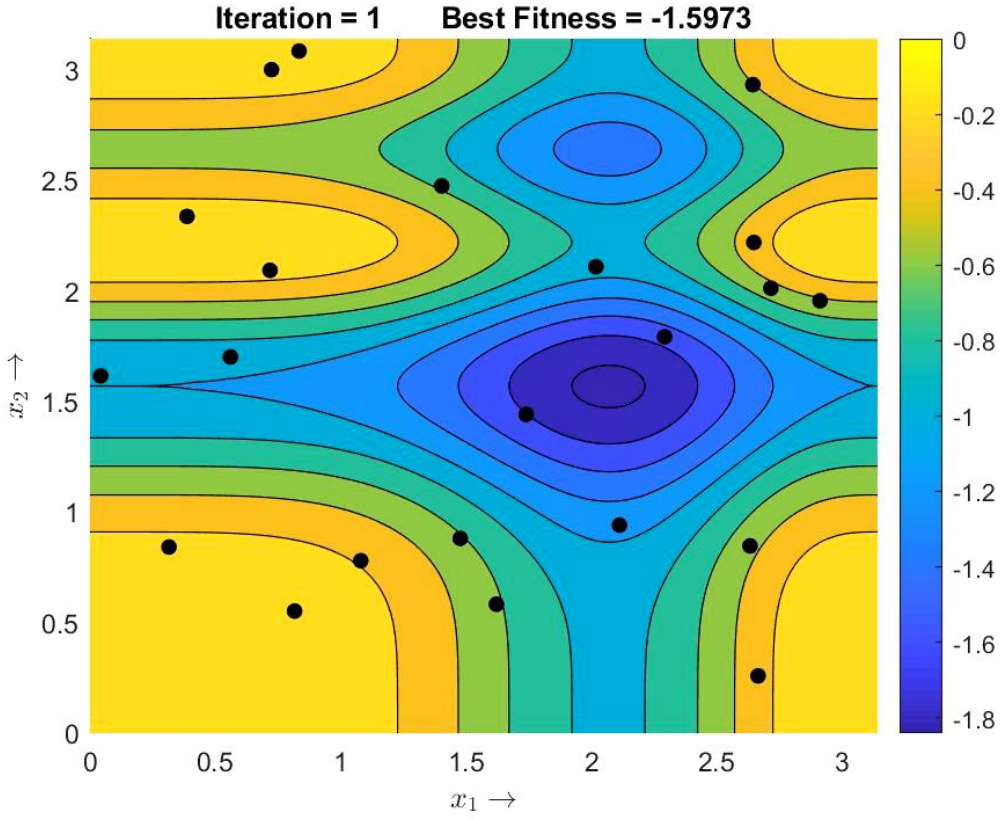
Content
- What we will talk about!

1. **Project Proposal**: Goal of the Project.

2. **Particle Swarm Optimization**:
   i. Basic Concept
   ii. Own Implementation

3. **Performance Study**:
   i. Benchmark Functions
   ii. Experimental Setup
   iii. Findings

4. **Summary & Outlook**



Project Proposal
- Goal of the Project

- **GOAL:**

  Implementation of own particle swarm optimizer with different swarm topologies/connectivity and hyperparameter selection.

  Can my algorithm outperform the commercial MATLAB optimizer?

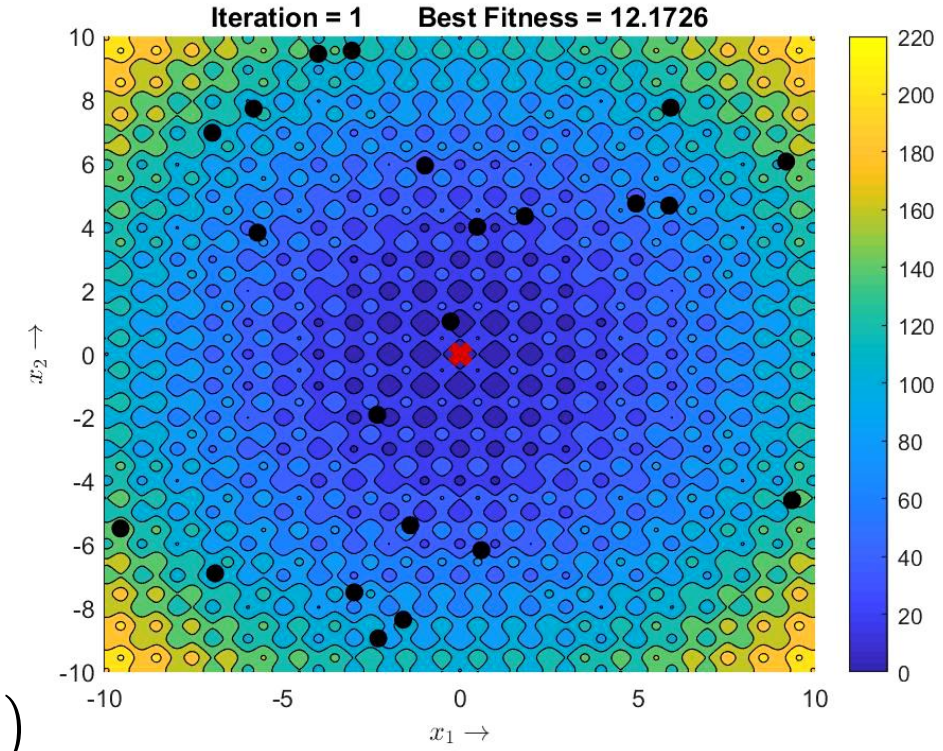
- **Project steps**:

  1. *Implementation* of basic PSO algorithm: *generic particle (object), global (full-mesh) PSO*
  2. *Topology* extensions: *'full-mesh', 'ring', 'wheel', 'von Neumann', 'random'*
  3. *Hyperparameter* extensions*:* *'static', 'dynamic'*
  4. Monarchy-based PSO: subdivision of swarm into subgroups.
  5. Testbench & *benchmark function* implementation
  6. Testing: *Comparison of own implementation with commercial MATLAB particle swarm optimizer.*

Particle Swarm Optimization (PSO)
- Basic Concept

- *population-based* search algorithm imitating ***swarm intelligence***.
- **Concept**: *Swarm of particles is randomly initialized.* ***Each particle represents a possible solution*** *of the problem and move through the search space based on its own current state, memory, and the information provided by its neighbours until the swarm eventually* ***converges to optimal solution****.*
- **Formula**:
  - $v_i(t+1) = w \cdot v_i(t) + a_1 \cdot R_1 \cdot \left(p_i^{best} - p_i(t)\right) + a_2 \cdot R_2 \cdot \left(p_{gbest} - p_i(t)\right)$
  - $p_i(t+1) = p_i(t) + v_i(t)$

with w … inertia, a_1 … cognitive term, a_2 … Social Component, and $R_1, R_2 \sim U_{[0,1]}$ … random number between [0,1]



full-mesh social network with static hyperparameters (Rastrigin benchmark; $w = 0.7$, $a_1 = 1$ and $a_2 = 2$)

- **Control Parameters** *(according to Engelbrecht, 2017)*:
  1. Swarm Size N
  2. Number of maximum iterations i_{max}
  3. Acceleration Coefficients w, a_1, a_2
  4. Neighbourhood size ⇔ Topology
  5. Velocity clamping

Particle Swarm Optimization (PSO)
- Exploration vs. Exploitation

- ***Exploration***: Ability to “explore different regions of the search space in order to locate a good optimum.” (Engelbrecht, 2007)
- ***Exploitation***: Ability to “concentrate the search around a promising area in order to refine the candidate solution.” (Engelbrecht, 2007)

- **Idea**: **Monarchy-based PSO**

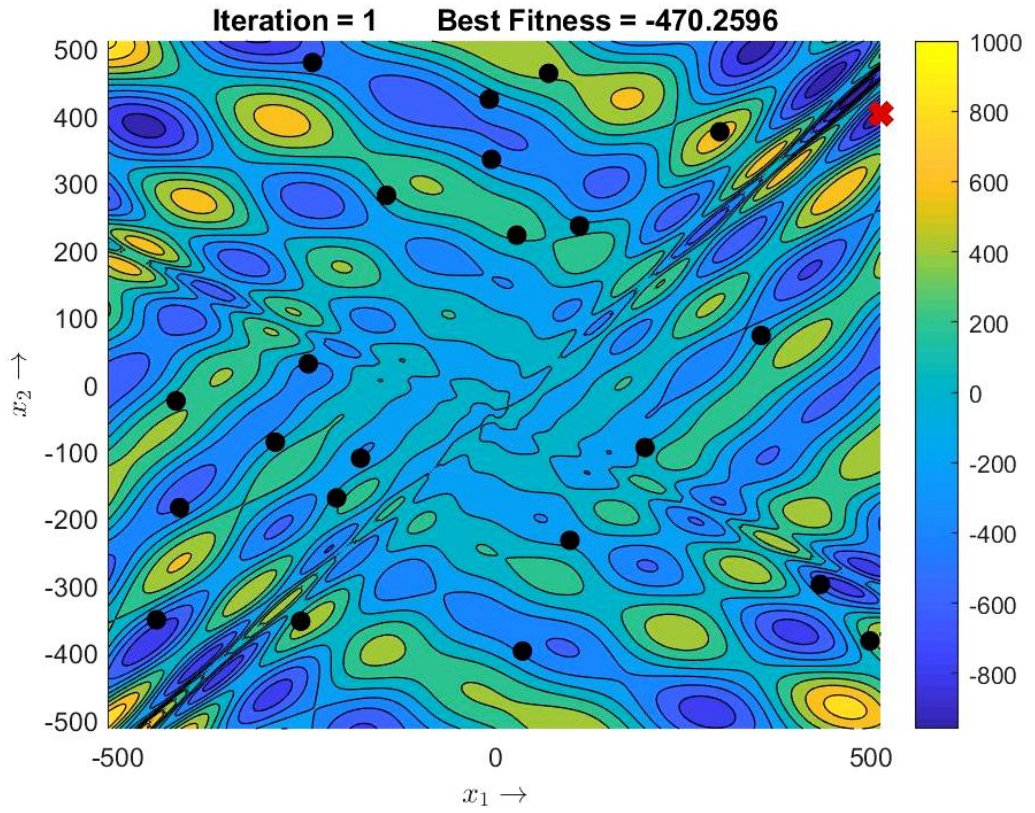
  ⇒ Subdivide swarm in 2 task groups

  1. ***Monarch/Ruler***: responsible for *exploration*

     selfish, self-confident ⇔ large w, a_1 and small a_2
  2. ***Worker***: responsible for *exploitation*

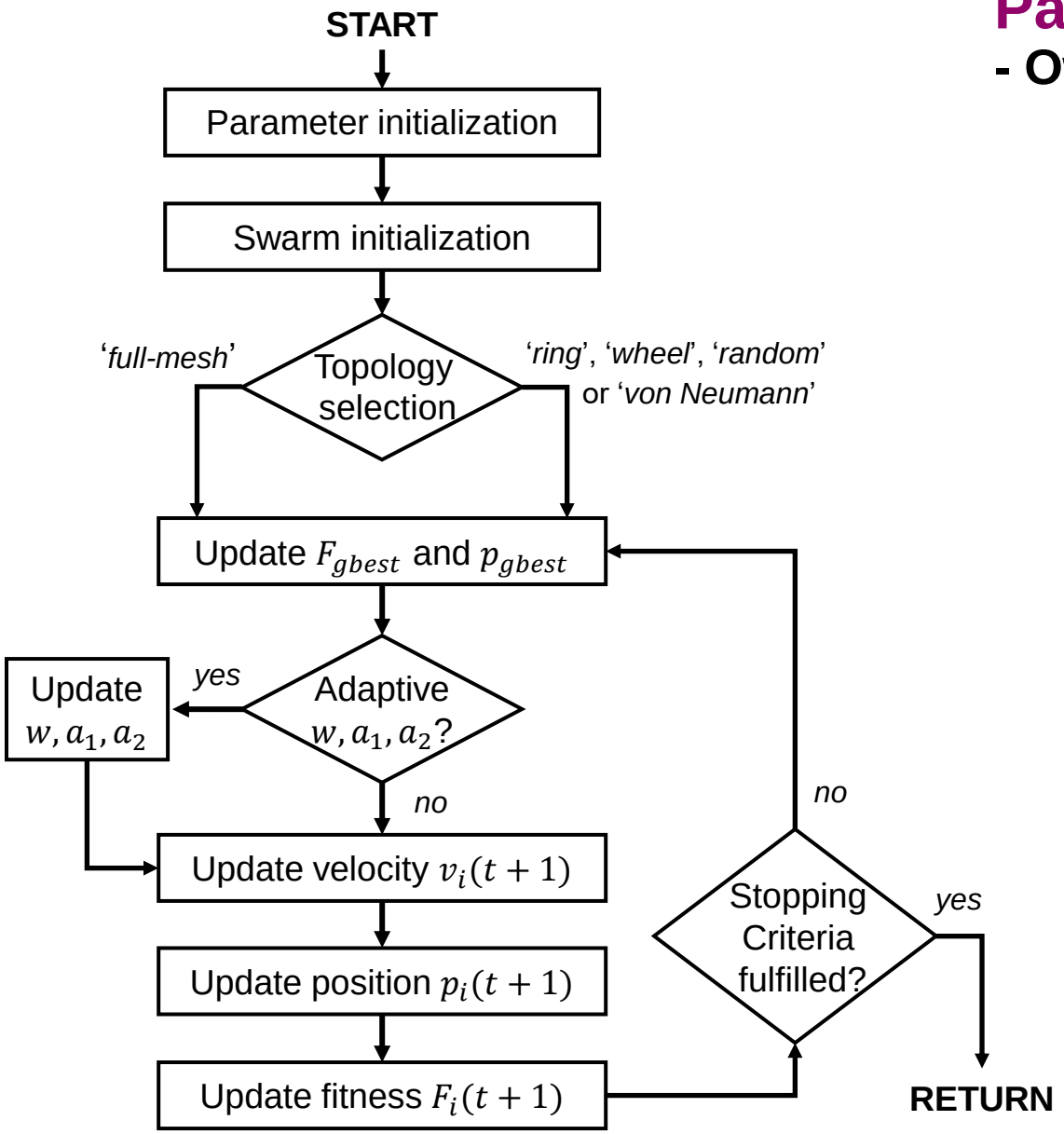
     trusty, social ⇔ large a_2 and small w, a_1

- **Implementation**:
  1. Define ‘Monarch-to-Worker’ ratio and assign duty randomly to each particle. (*E.g., 1/3 monarchs, and 2/3 workers.*)
  2. Assign static hyperparameters w, a_1, a_2 to each group.
  3. Select topology (recommended: *full-mesh* configuration).

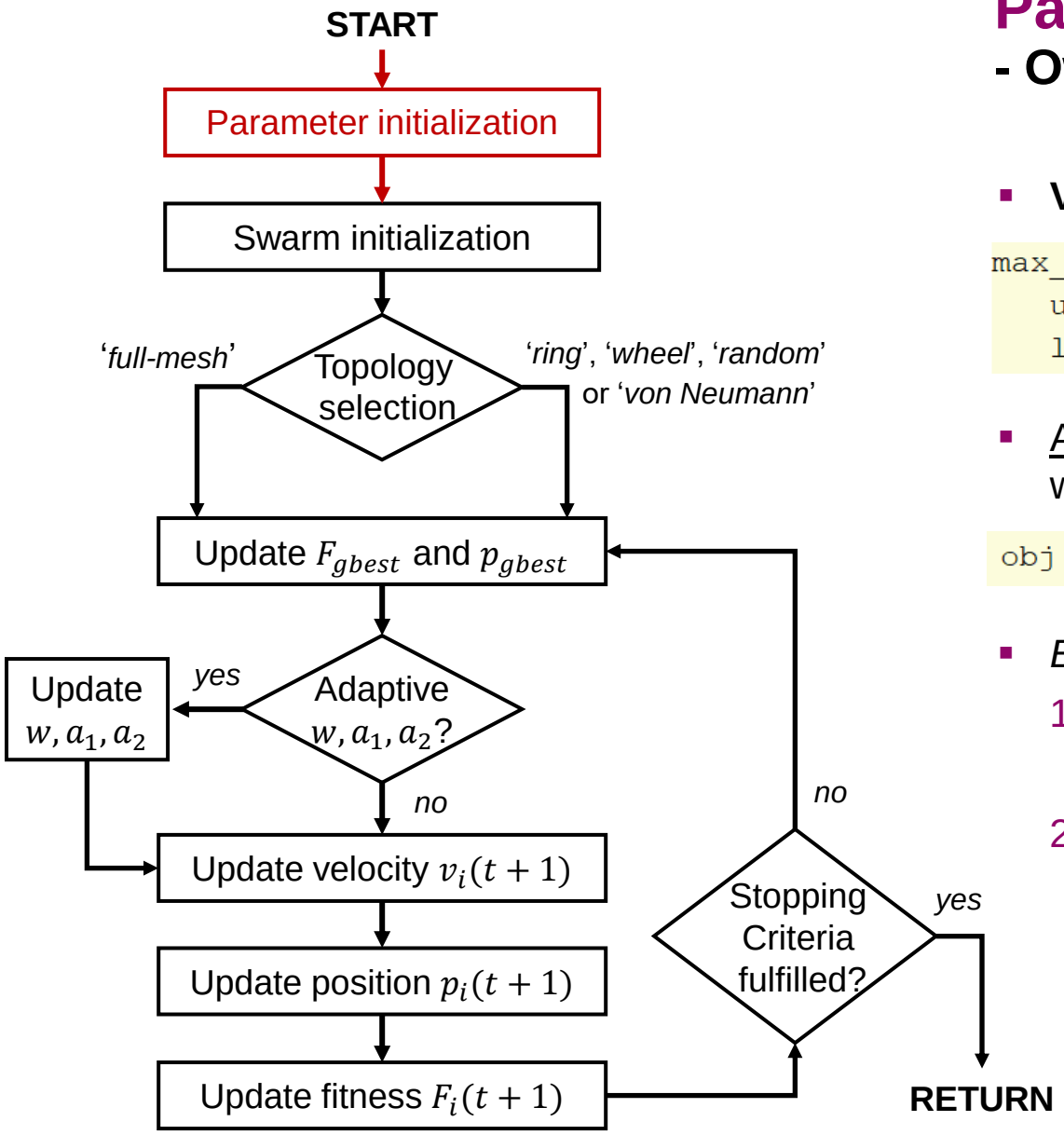


Monarchy-based PSO with *full-mesh* connectivity (Eggholder benchmark)

Particle Swarm Optimization (PSO)
- Own Implementation



Particle Swarm Optimization (PSO)
- Own Implementation



- **Velocity Clamping** (*static*): bound velocity of particles (v_{max})

```
max_step_size = (ub-lb).*vel_rel; %% vel_rel selected by user
   ub_velocity = max_step_size;
   lb_velocity = -ub_velocity;
```

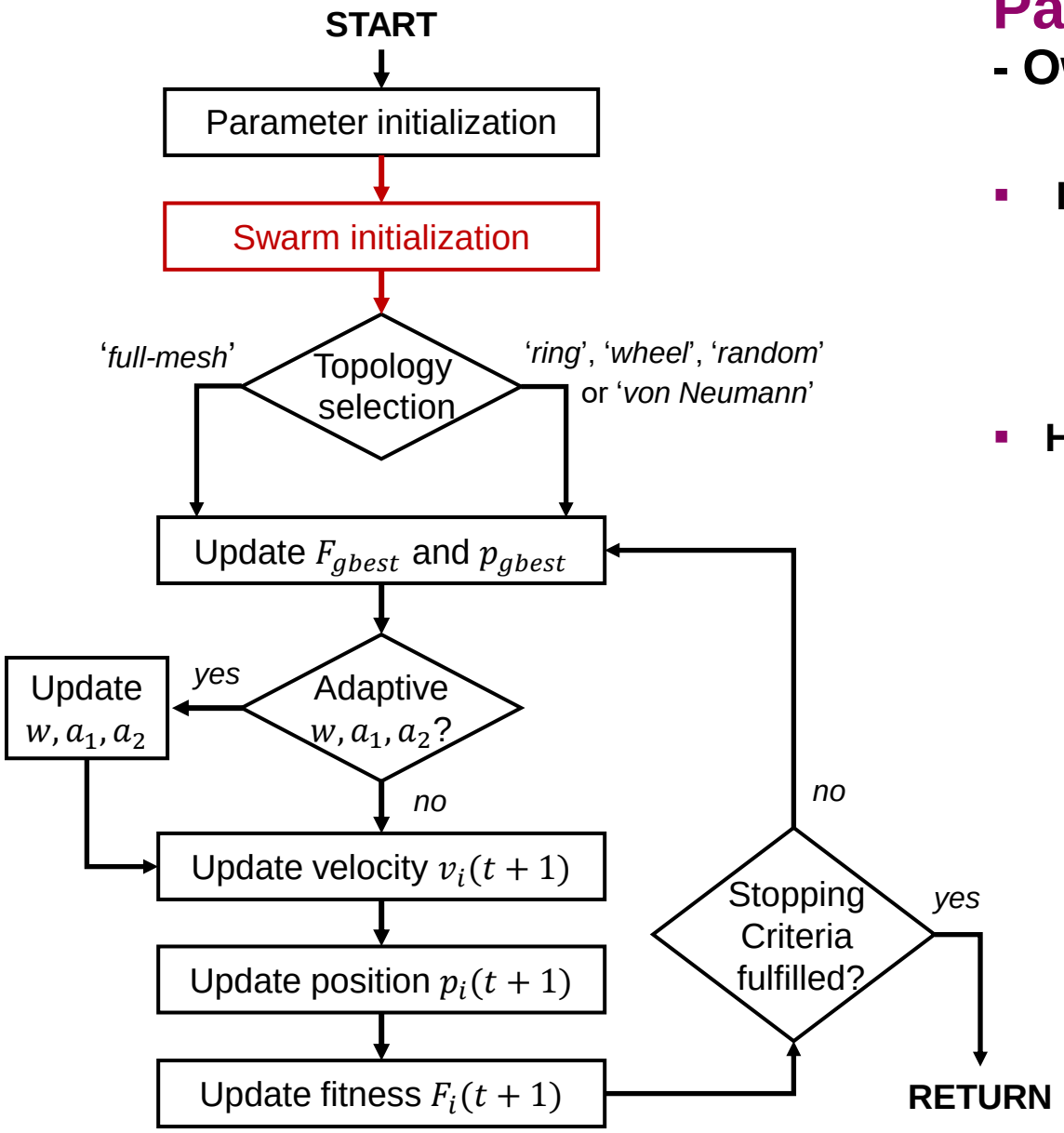
- Algorithm: each particles is responsible that its velocity lies within the pre-defined bounds.

```
obj.velocity = min(max(v_new,obj.lb_vel),obj.ub_vel);
```

- *But why?* *(Oldewage, 2017 + Engelbrecht, 2007)*
  1. Prevent *uncontrolled acceleration* of particles, and probably early "leaving" of the search space (*roaming behaviour*).
  2. Allows to control the ***granularity*** of the search
     - *i.* *Coarse-grained*: promote global exploration, but with the risk to miss good solutions
     - *ii.* *Fine-grained*: promotes local exploitation, but if selected too small then the swarm may not explore the search space sufficiently.

  ⇔ *exploration-exploitation* trade-off.

Particle Swarm Optimization (PSO)
- Own Implementation

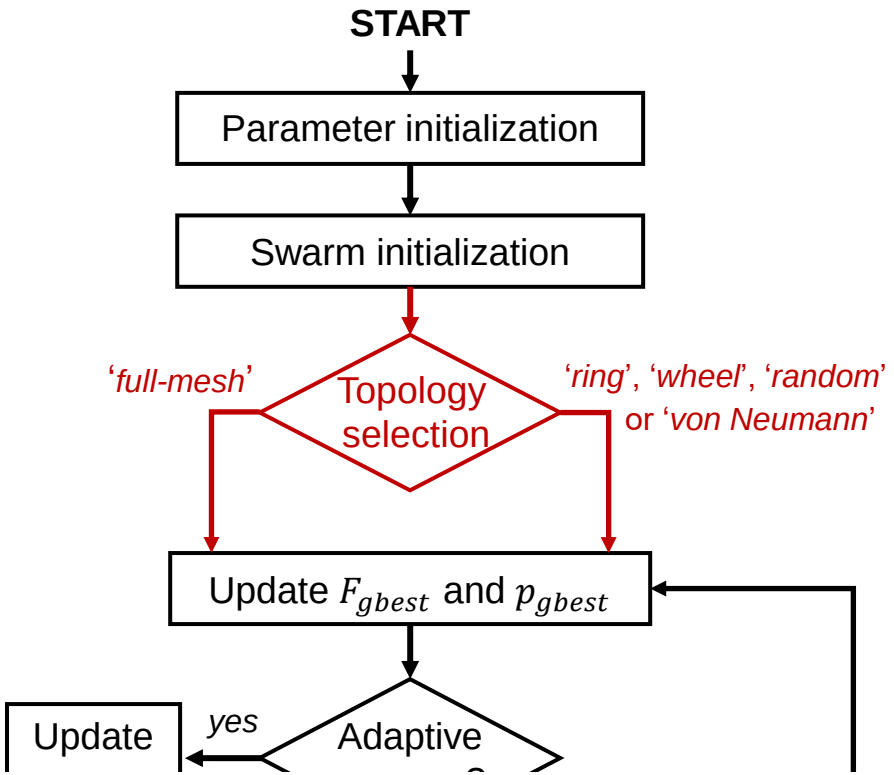


- **Initial values**:

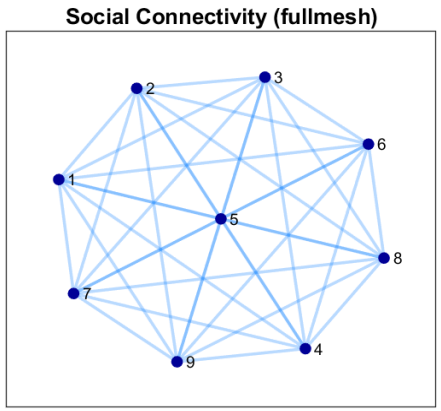
$$p_i(0) = x_{i,min} + R \cdot (x_{i,max} - x_{i,min}), \qquad R \sim U_{[0,1]}$$
$$v_i(0) = v_{i,min} + R \cdot (v_{i,max} - v_{i,min})$$

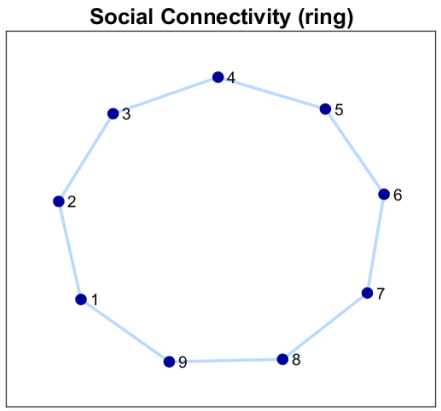
- **Hyperparameters**:
  1. *Static*: assign to each particle the same swarm/control parameters w, a_1, a_2 through the entire lifetime of the swarm (e.g. $w = 0.7, a_1 = 1,$ *and* $a_2 = 2$*).*
  2. *Dynamic/Adaptive*: Linear change of hyperparameter over the number of iterations (*Ratnaweera, 2004*).
     - $w = (w^{max} - w^{min}) \cdot \frac{i_{max} - i}{i_{max}} + w^{min}$
     - $a_1 = (a_1^{min} - a_1^{max}) \cdot \frac{i}{i_{max}} + a_{1,max}$
     - $a_2 = (a_2^{max} - a_2^{min}) \cdot \frac{i}{i_{max}} + a_{2,min}$
  3. *Monarchy-based*: group-specific, static hyperparameters
     e.g. $w_{monarch} = 1,\ a_{1,monarch} = 4,\ a_{2,monarch} = 0.2$
     $w_{worker} = 0.4,\ a_{1,worker} = 0.3,\ a_{2,worker} = 2$

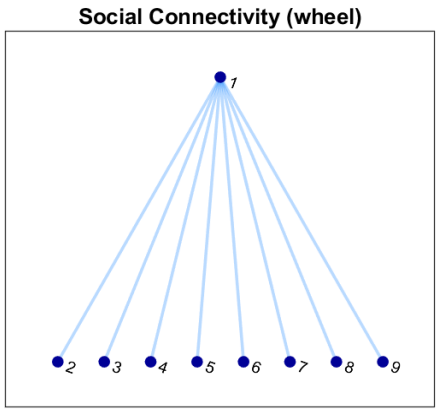
Particle Swarm Optimization (PSO)
- Own Implementation

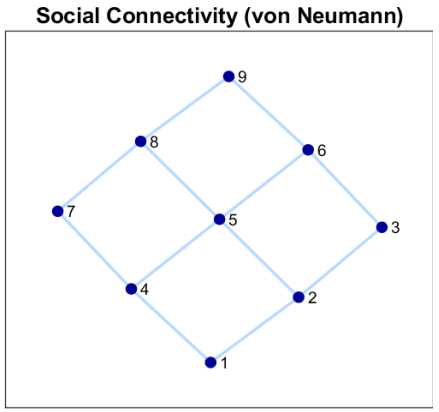


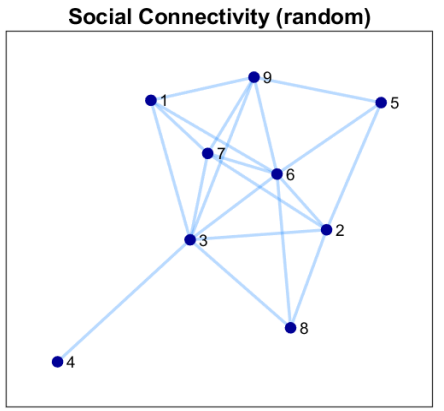
- Implementation of 5 *topologies*:
  1. **Full mesh**: all particles are interconnected (→ *global PSO*)
  2. **Ring**: each particle communicates with *dim** neighbours.
  3. **Wheel**: <u>one</u> particle acts as communication interface between all other *isolated* particles.
  4. **Von Neumann**: particles are connected in a grid-like structure.
  5. **Random**: each particle is connected with <u>at least</u> one other randomly selected particle.





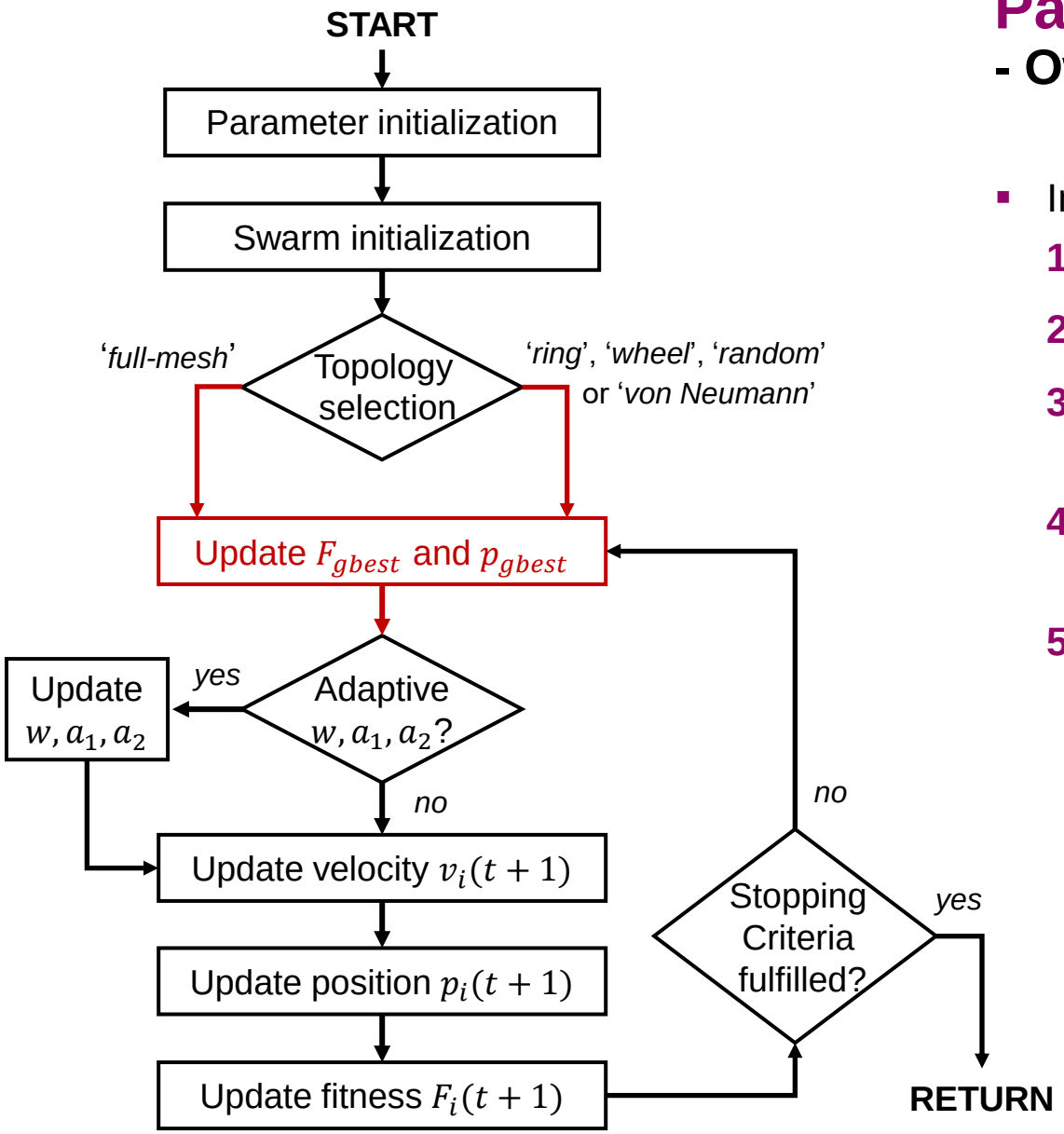






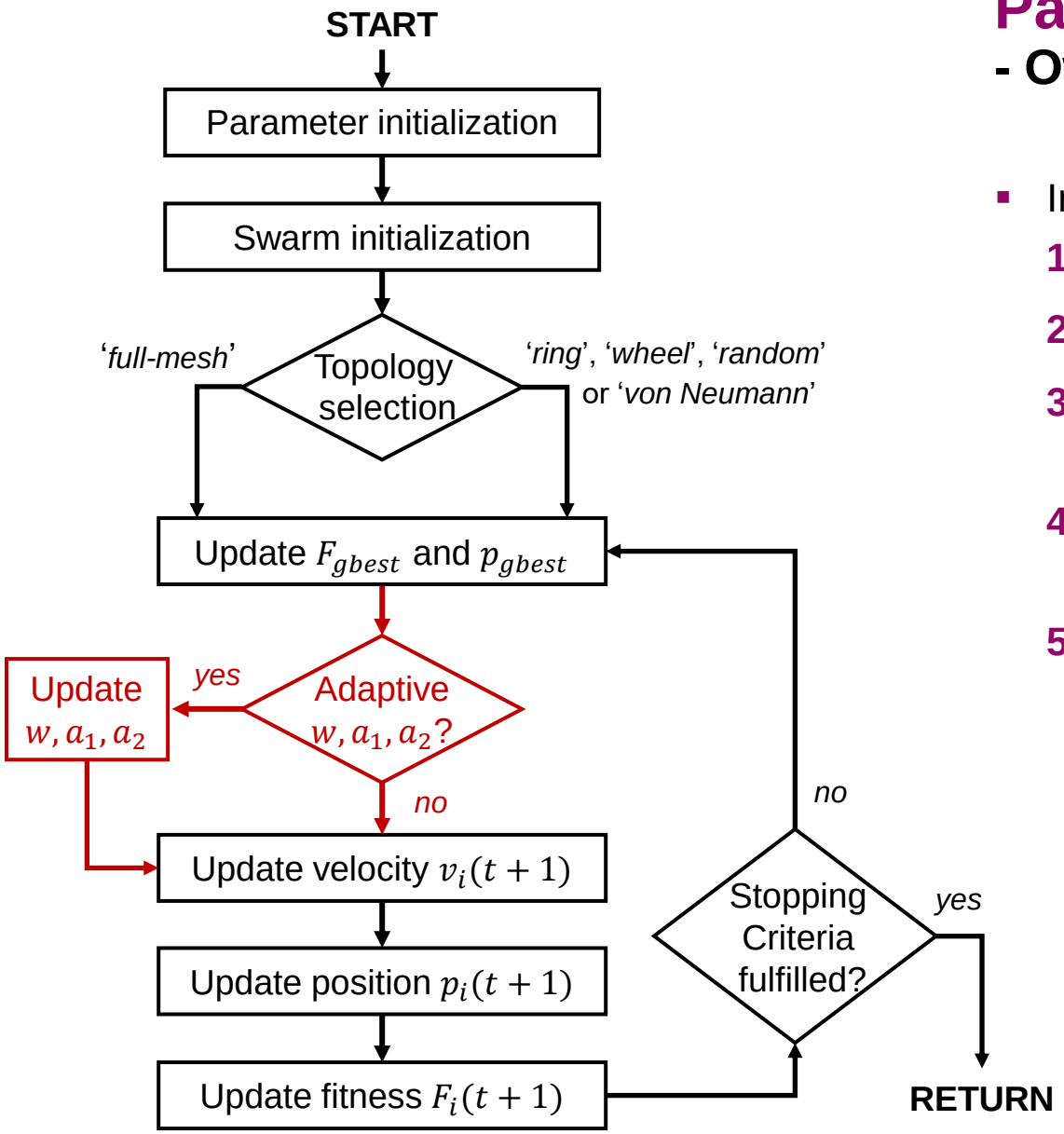
* ***dim*** describes the dimension of the problem (e.g. 2-dimensional ⇒ *dim = 2*)

Particle Swarm Optimization (PSO)
- Own Implementation



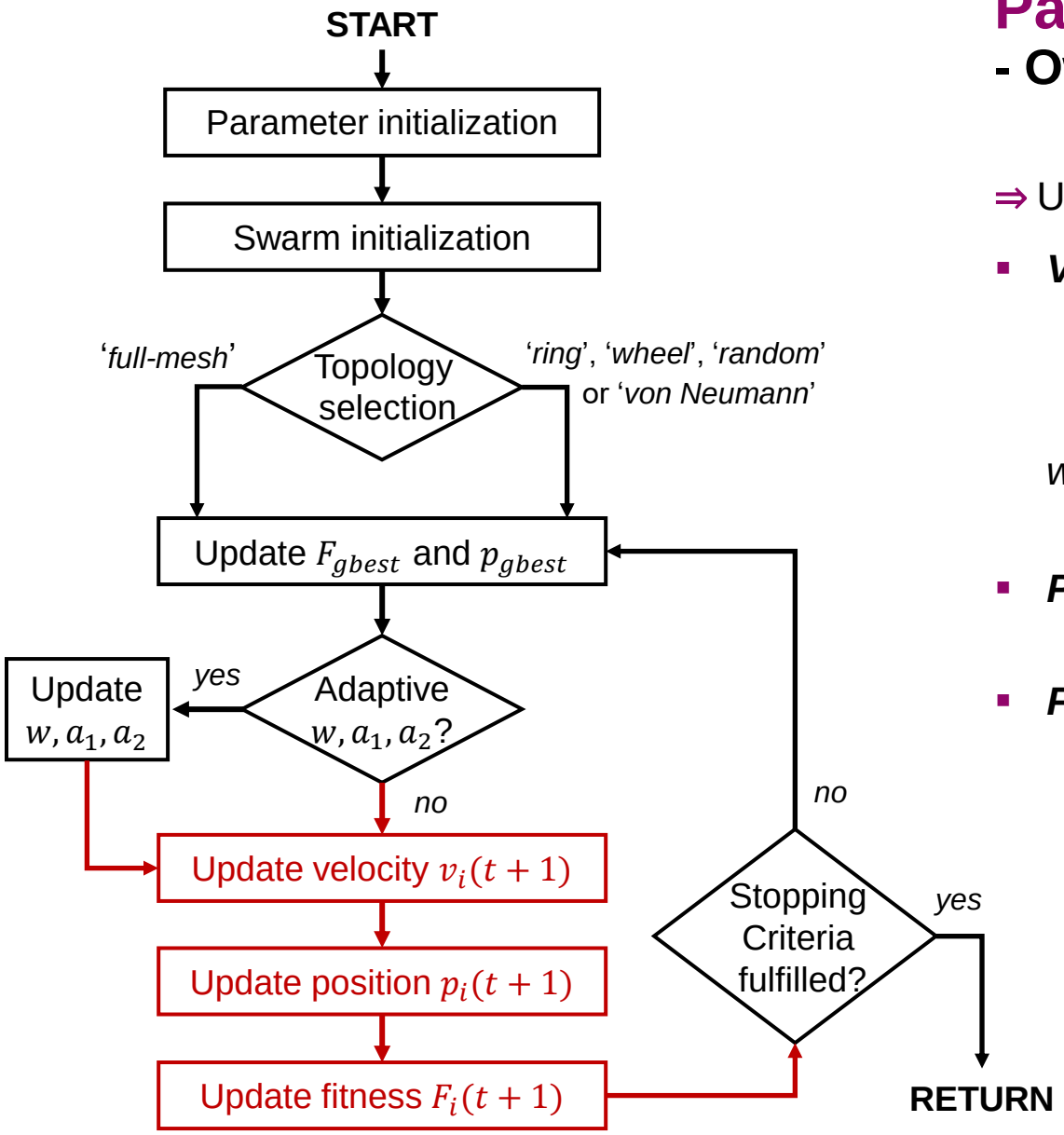
- Implementation of 5 *topologies*:
  1. **Full mesh**: all particles are interconnected (→ *global PSO*)
  2. **Ring**: each particle communicates with *dim** neighbours.
  3. **Wheel**: one particle acts as communication interface between all other *isolated* particles.
  4. **Von Neumann**: particles are connected in a grid-like structure.
  5. **Random**: each particle is connected with at least one other randomly selected particle.

Particle Swarm Optimization (PSO)
- Own Implementation



- Implementation of 5 *topologies*:
  1. **Full mesh**: all particles are interconnected (→ *global PSO*)
  2. **Ring**: each particle communicates with *dim** neighbours.
  3. **Wheel**: one particle acts as communication interface between all other *isolated* particles.
  4. **Von Neumann**: particles are connected in a grid-like structure.
  5. **Random**: each particle is connected with at least one other randomly selected particle.

Particle Swarm Optimization (PSO)
- Own Implementation



⇒ Update according to *PSO rules*:

- ***Velocity*** v_i :

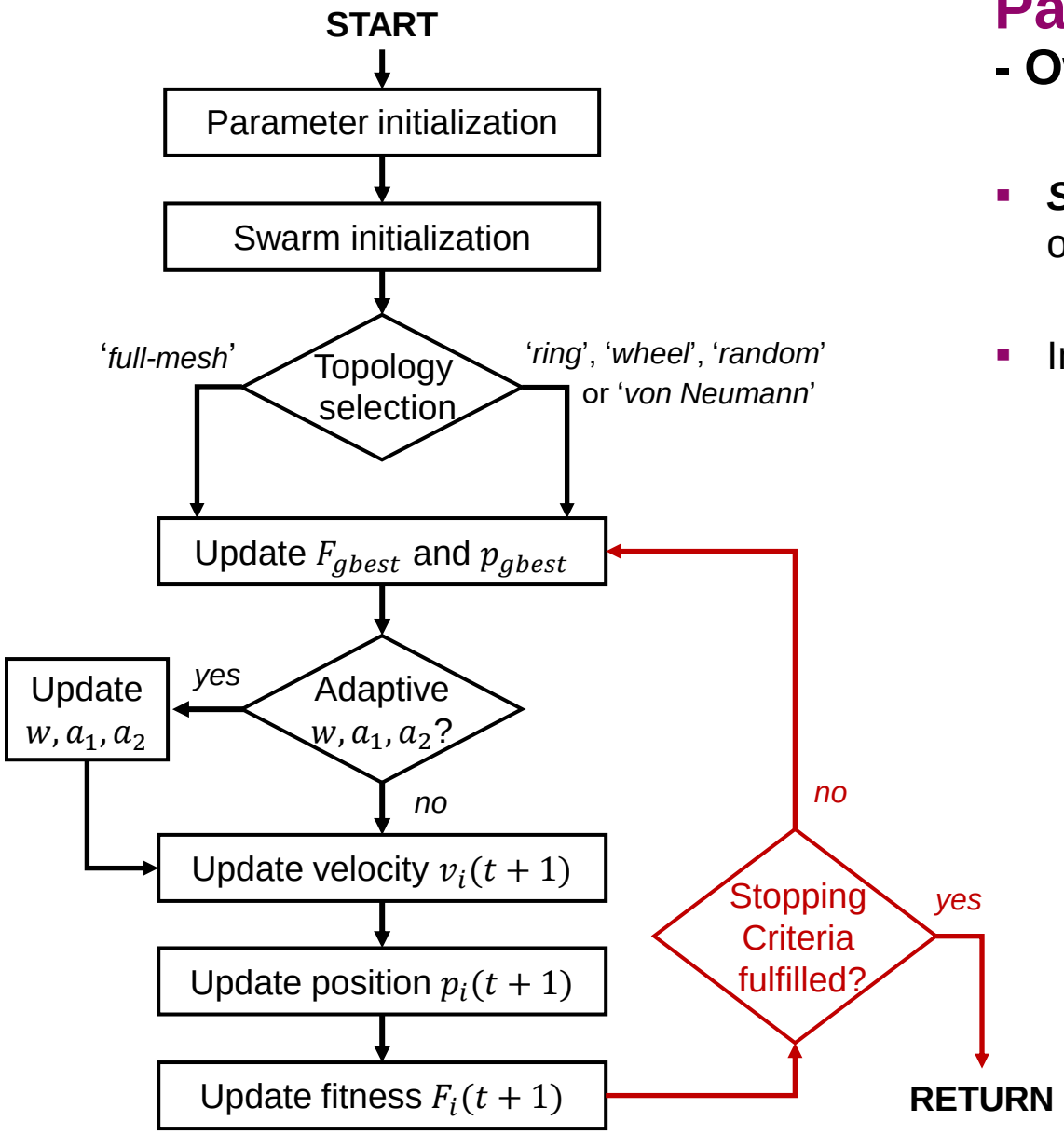
$$v_i(t+1) = w \cdot v_i(t) + a_1 \cdot R_1 \cdot \left(p_i^{best} - p_i(t)\right) + a_2 \cdot R_2 \cdot \left(p_{gbest} - p_i(t)\right)$$

  with $R_1, R_2 \sim U_{[0,1]}$

- ***Position*** p_i : $p_i(t+1) = p_i(t) + v_i(t)$

- ***Fitness*** F_i : $F_i(t+1) = F(p_i(t+1))$

Particle Swarm Optimization (PSO)
- Own Implementation



- ***Stopping condition*** is responsible for a controlled termination of the iterative search process.

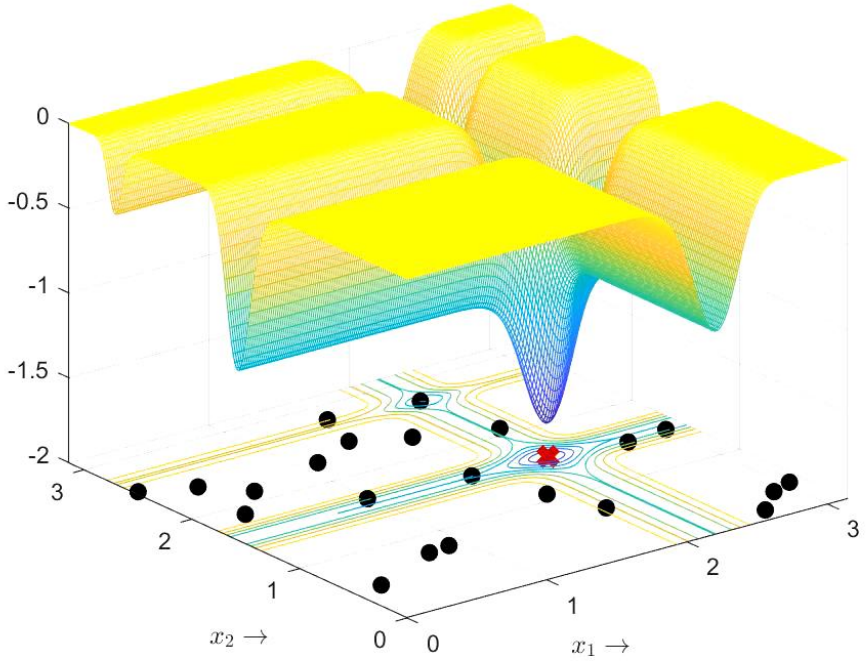
- Implemented *termination mechanisms* (→ stopping criteria):
  1. Limitation by ***maximum number of iterations***:
     ⇒ algorithm terminates the latest when the number of iterations exceeds a defined threshold (i_{max}).
     ⇒ *if* $i > i_{max}$ *then stop!*
  2. No ***improvement*** of global best fitness:
     ⇒ algorithm terminates if there is no improvement observed over a certain number of iterations.

```
f_count = f_count + 1;  % Increment f_count as default

if F_max < F_past       % If improvement, then...
    f_count = 1;        %   1. Reset counter
    F_past = F_max;     %   2. Update best fitness
else
    if f_count > e_threshold    % Terminate
        bool = 0;
    end
end
```

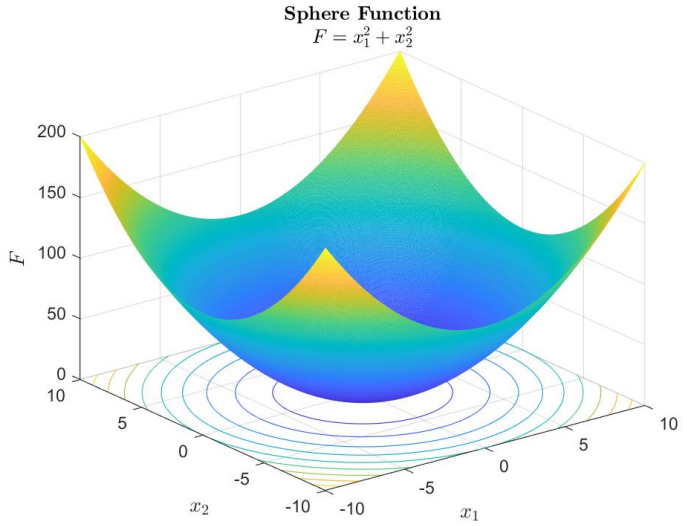
Experimental Setup and Benchmark Testing

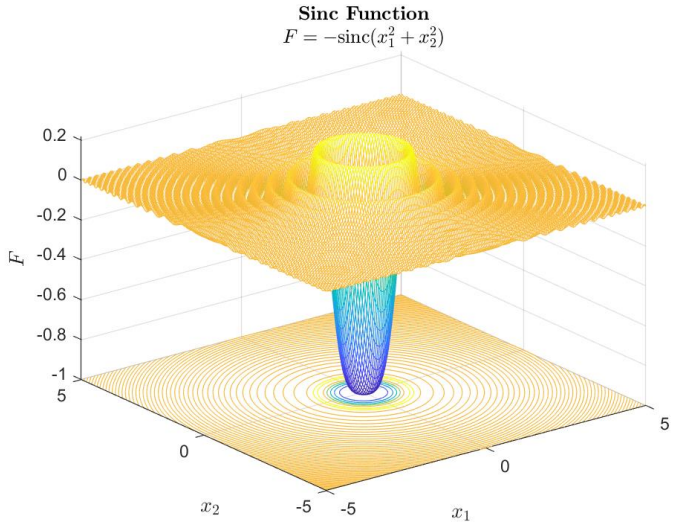
- Performance Evaluation -

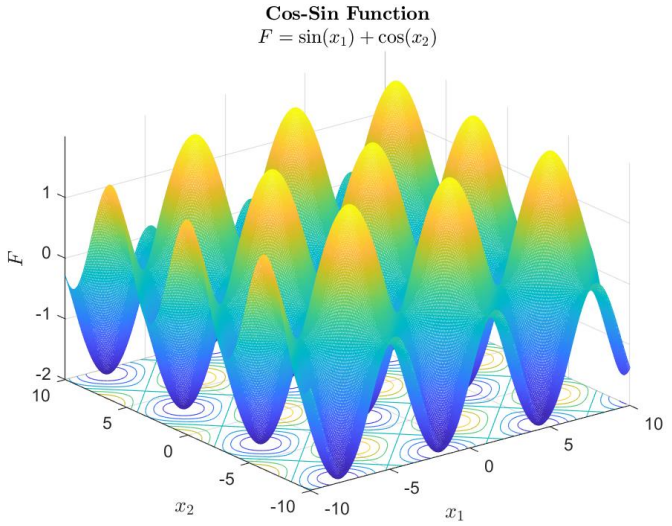


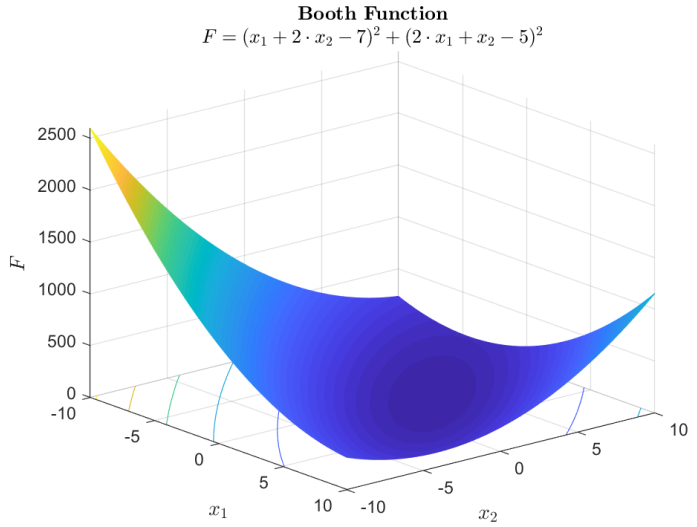
Performance Study

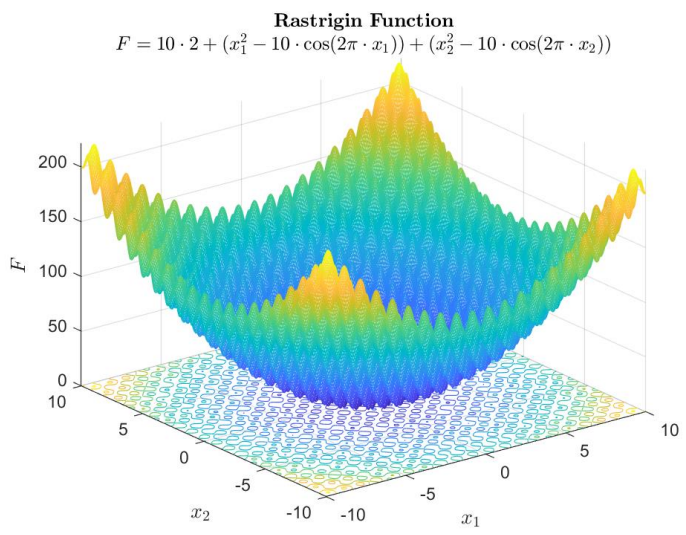
- Benchmark Functions (difficulty: *low to middle*)

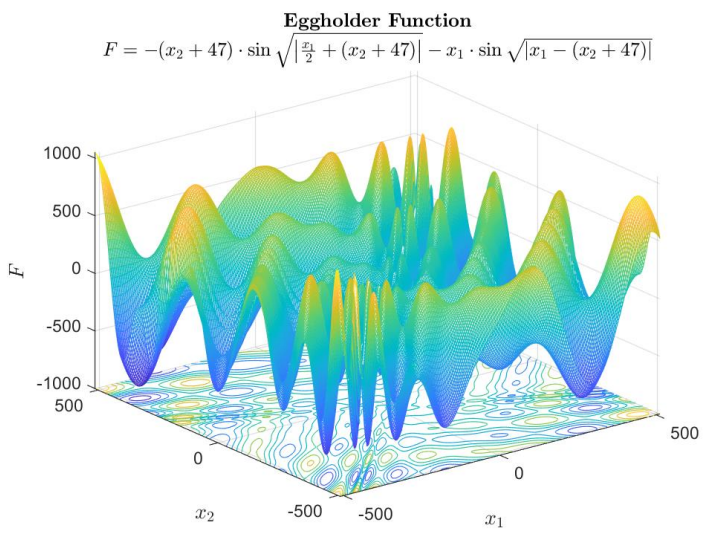








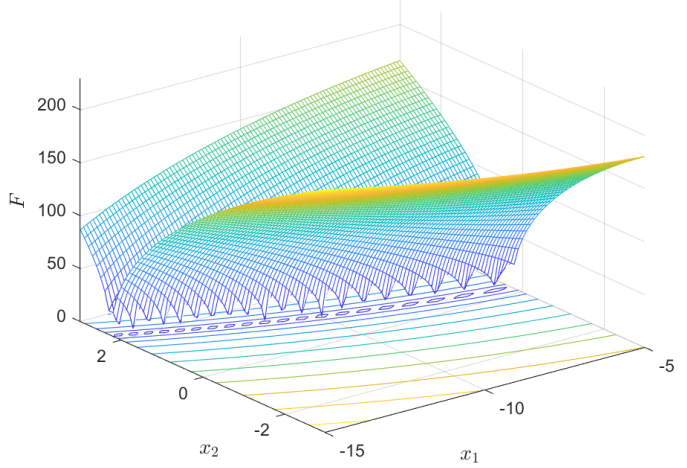




Performance Study
- Benchmark Functions (difficulty: *hard*)

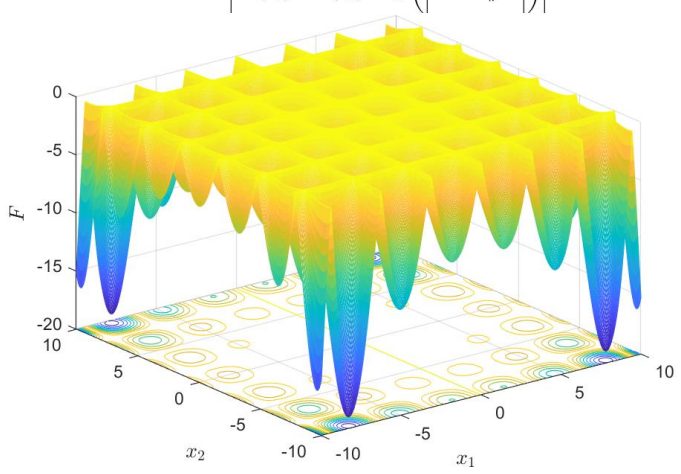
Bukin Function Nr. 6

$$F = 100 \cdot \sqrt{|x_2 - 0.01 \cdot (x_1^2)|} + 0.01 \cdot |x1 + 10|$$



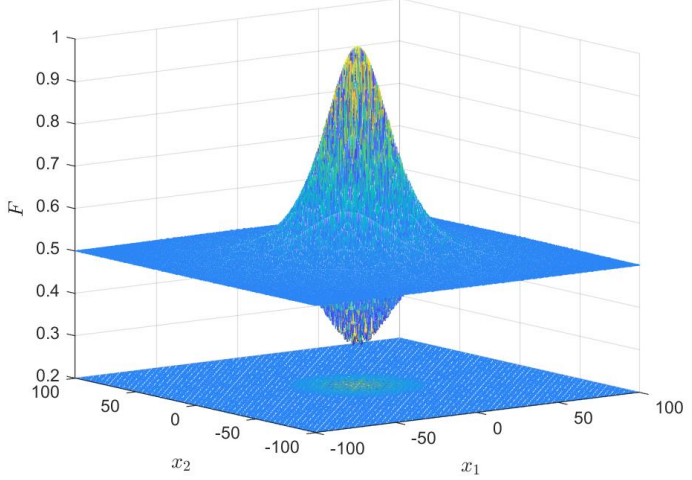
Hoelder Table Function

$$F = -\left|\sin(x_1) \cdot \cos(x_2) \cdot \exp\left(\left|1 - \frac{\sqrt{x_1^2+x_2^2}}{\pi}\right|\right)\right|$$



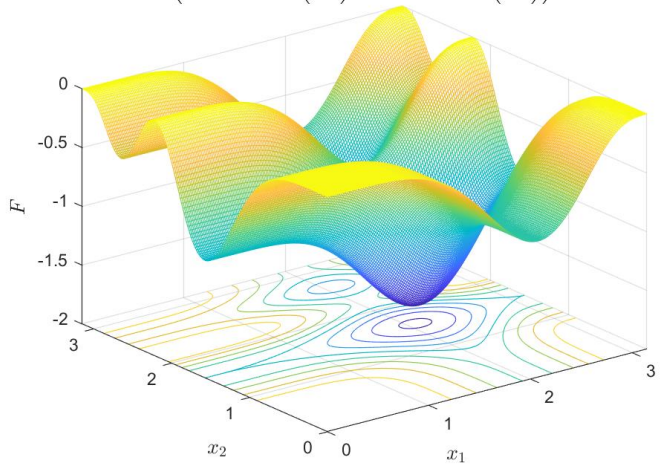
Schaffer Function Nr. 2

$$F = 0.5 + \frac{\cos(\sin|x_1^2 - x_2^2|)^2 - 0.5}{(1+0.001\cdot(x_1^2+x_2^2))^2}$$



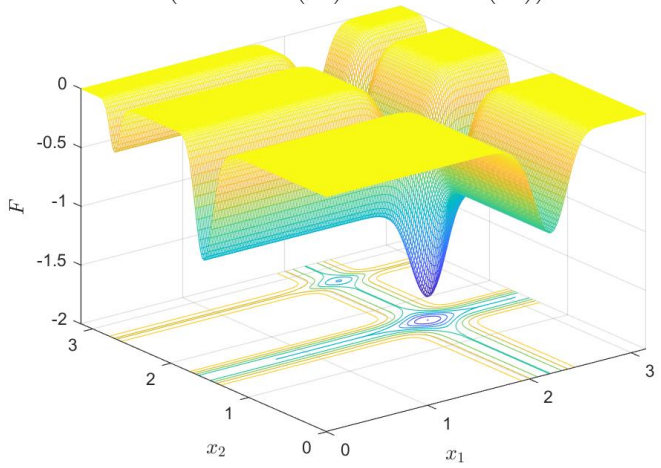
Michalewicz Function (smooth)

$$F = -\left(\sin(x_1) \cdot \sin^{2m}\left(\frac{1 \cdot x_1^2}{\pi}\right) + \sin(x_2) \cdot \sin^{2m}\left(\frac{2 \cdot x_2^2}{\pi}\right)\right), m = 1$$



Michalewicz Function (steep)

$$F = -\left(\sin(x_1) \cdot \sin^{2m}\left(\frac{1 \cdot x_1^2}{\pi}\right) + \sin(x_2) \cdot \sin^{2m}\left(\frac{2 \cdot x_2^2}{\pi}\right)\right), m = 10$$



Performance Study
- Experimental Setup

Relative velocity: 30% of search space

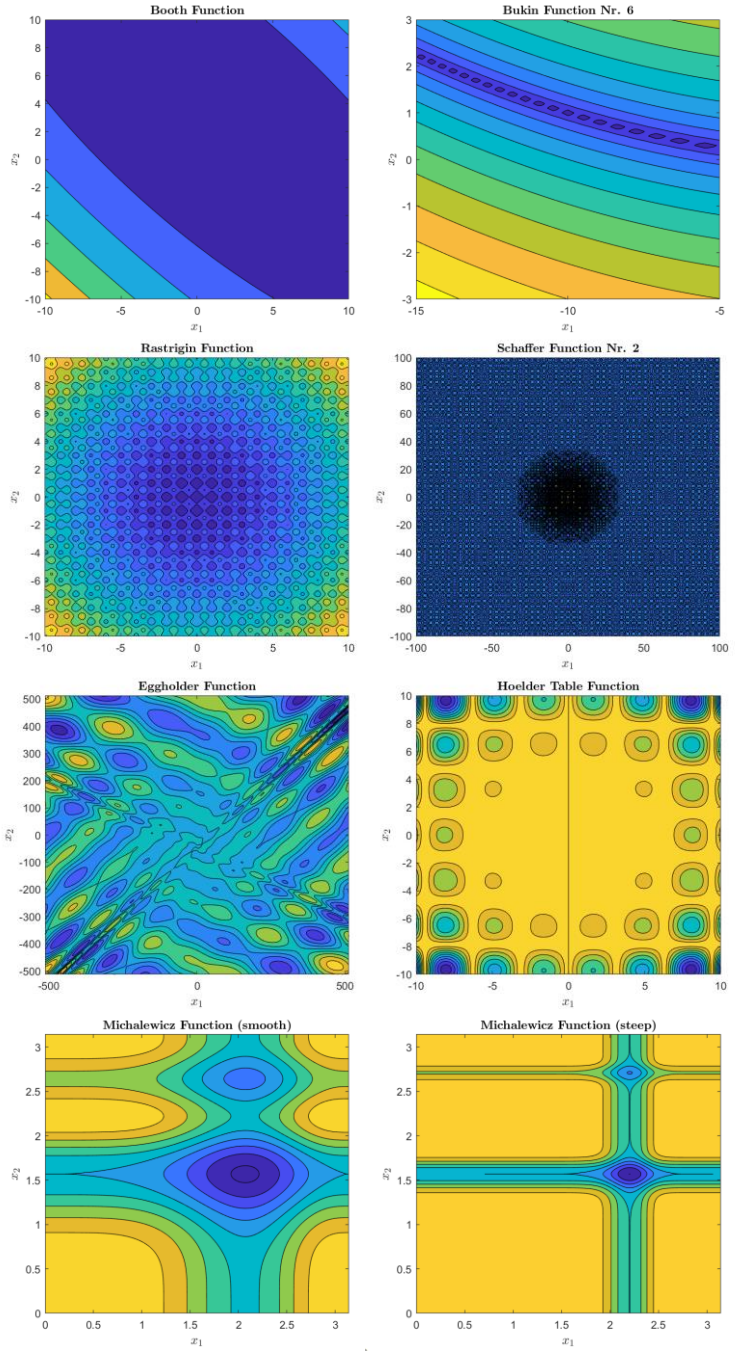
Selected topologies:

- *full mesh ('f'), ring ('r'), wheel ('w'), von Neumann ('n'), and random ('x')*

Selected hyperparameter:

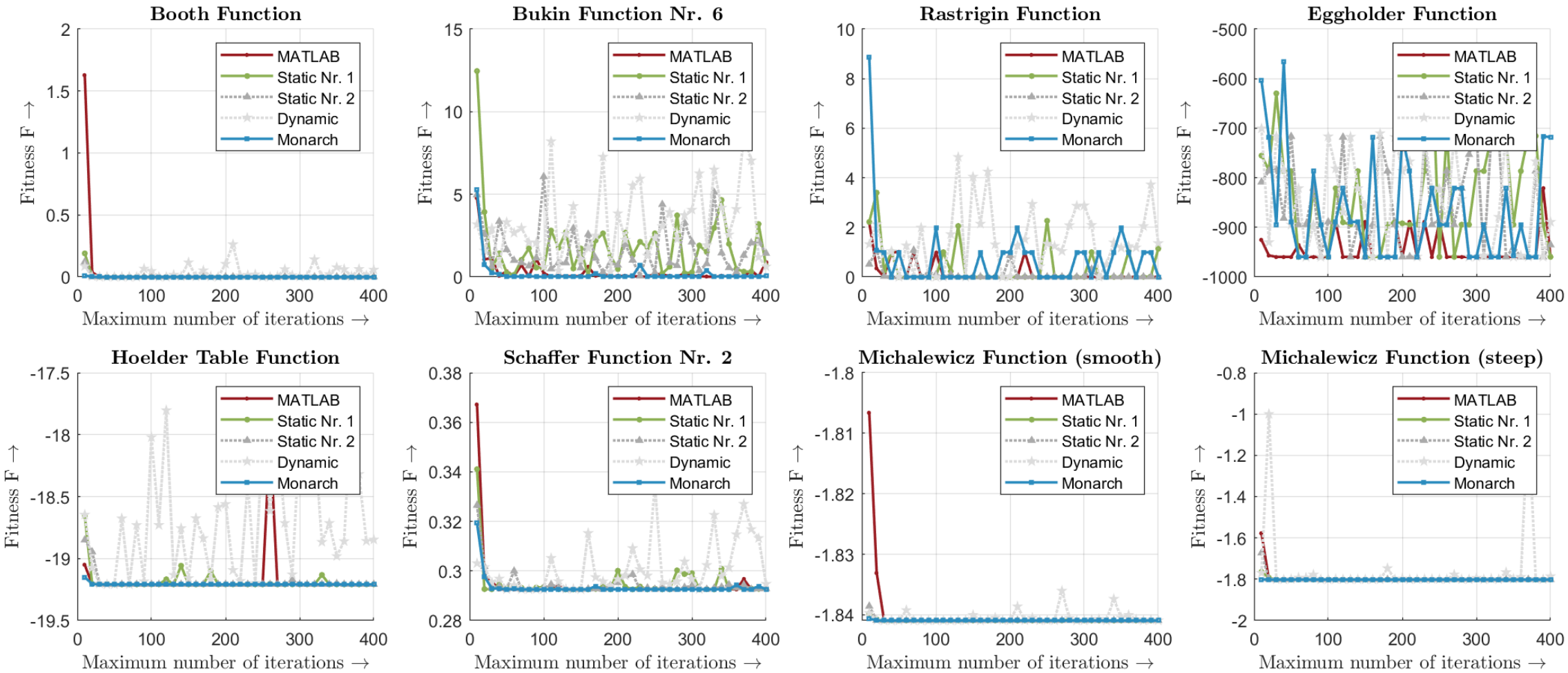
- MATLAB: $a_1 = a_2 = 1.49$, and $w = dyn. \in [0.1, 1.1]$
- Static #1: $a_1 = 1$, $a_2 = 2$, and $w = 0.7$
- Static #2: $a_1 = a_2 = 1.49$, and $w = 0.7$
- Dynamic: $a_1, a_2 \in [0.5, 2.5]$, and $w \in [0.4, 0.9]$
- Monarchy: *Monarchs*: $a_1 = 3$, $a_2 = 0.3$, and $w = 0.9$
  Workers: $a_1 = 0.5$, $a_2 = 2$, and $w = 0.5$

⇒ **Performance Measures**: *found optimum, CPU time and number of fitness evaluations*.



Performance Study
- Findings: Variation of maximum iterations

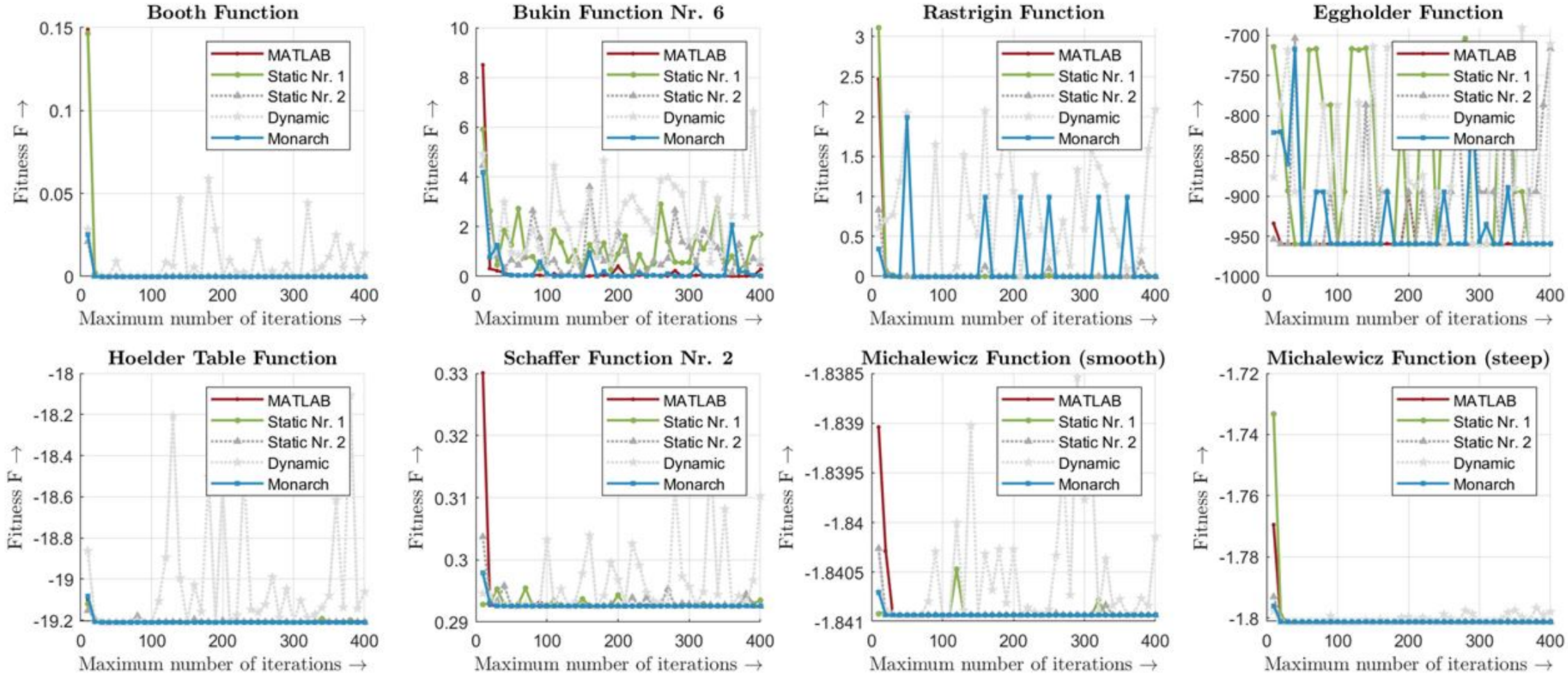
- Effect of the *maximum number of iterations* (i_{max}) on the **accuracy** of commercial and implemented PSO* (*N = 20*):



** Comparison of global PSO*

Performance Study
- Findings: Variation of maximum iterations

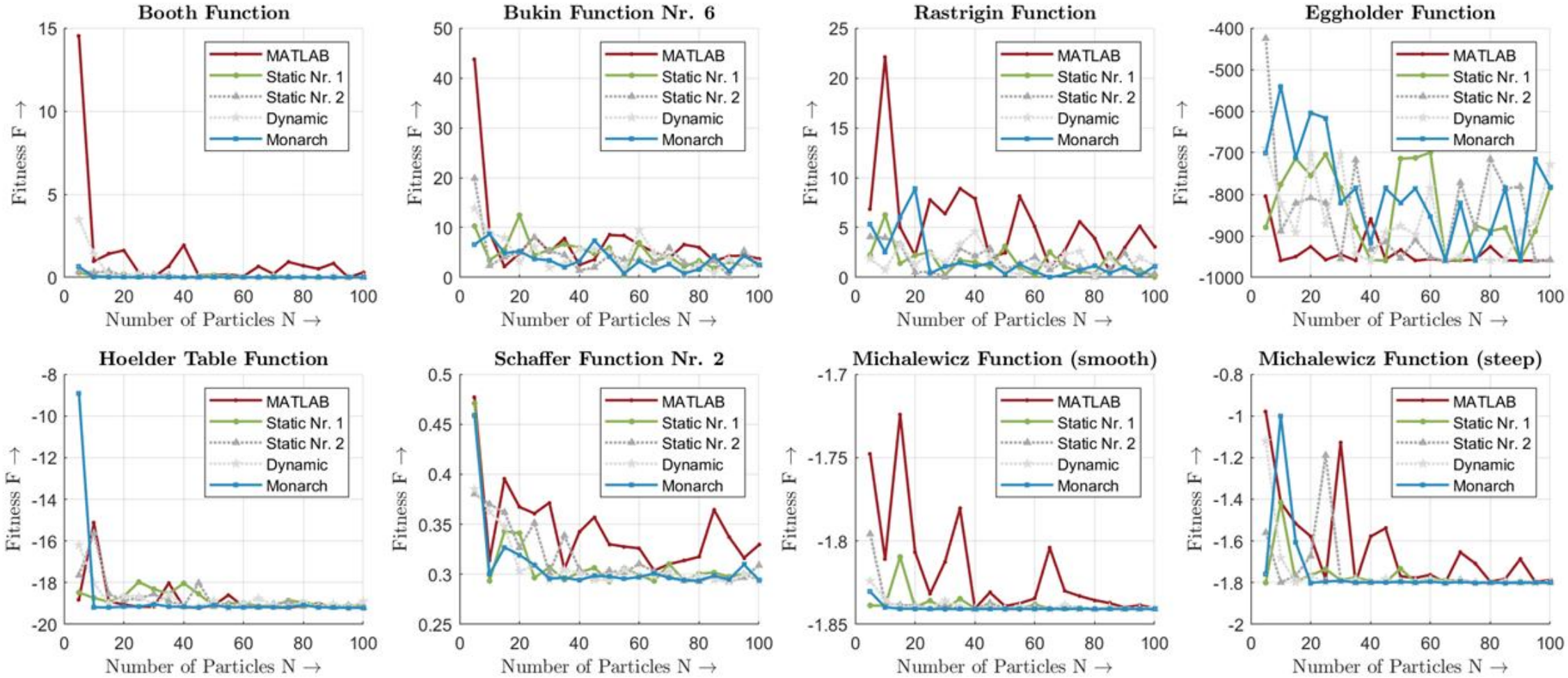
- Effect of the *maximum number of iterations* (i_{max}) on the **accuracy** of commercial and implemented PSO* (*N = 50*):



** Comparison of global PSO*

Performance Study
- Findings: Variation of swarm size

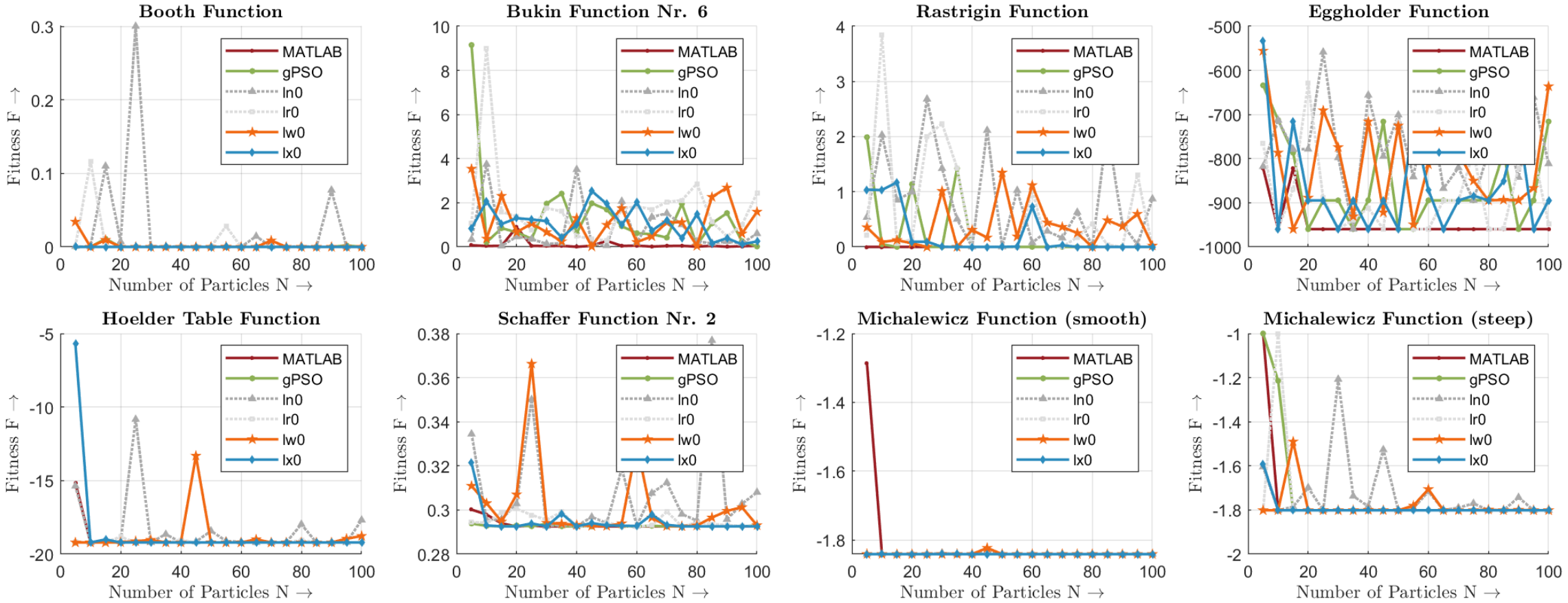
- Effect of *swarm size* (N) on the **accuracy** of commercial and private PSO* (i_{max} = *10*):



** Comparison of global PSO*

Performance Study
- Findings: Variation of Topology (*case: static nr. 1*)

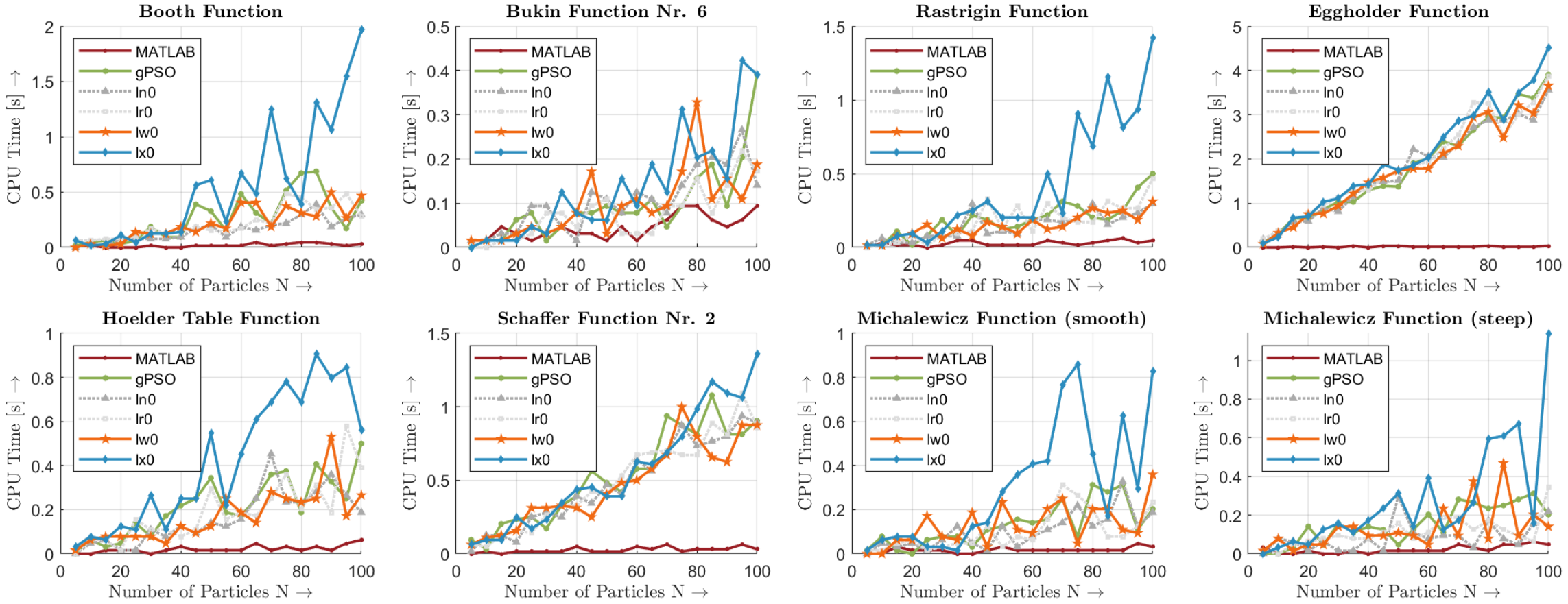
- Effect of *different topologies* on the **accuracy** of the algorithm ($i_{max} = 400$):



- ⇒ Comparable performance of random initialization and full mesh (gPSO) configuration.

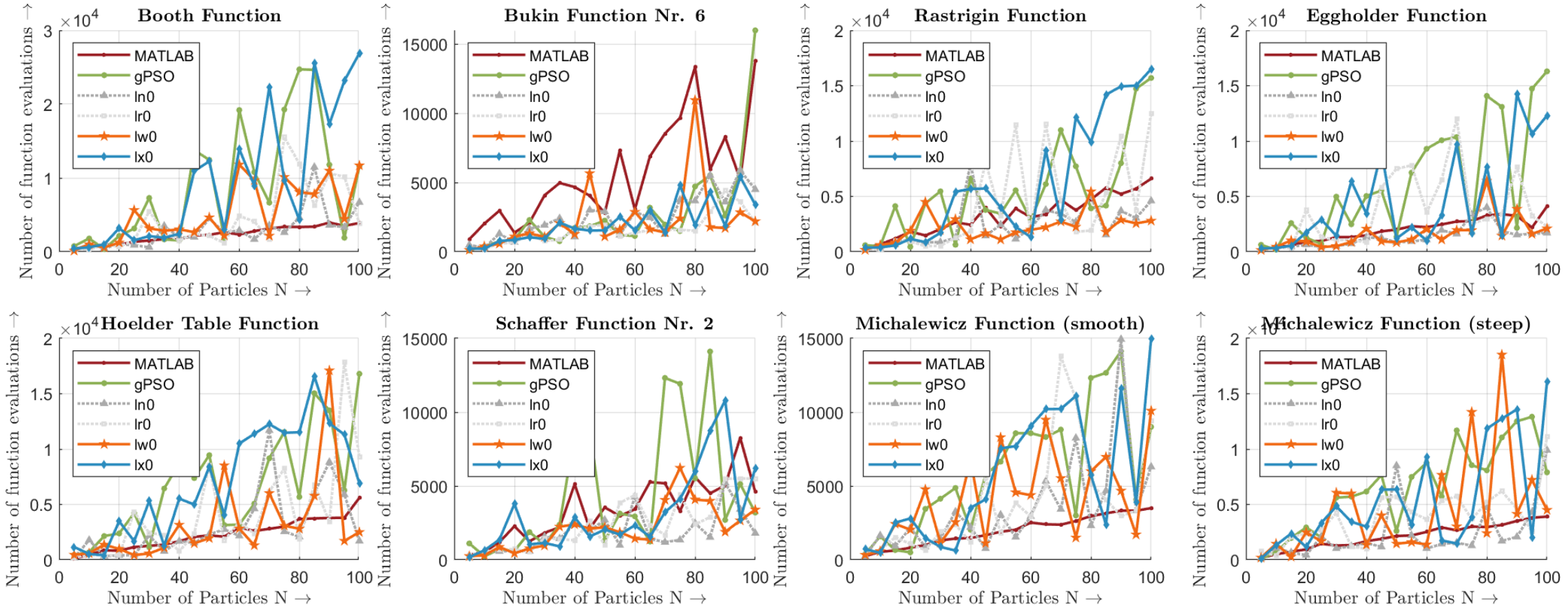
Performance Study
- Findings: Variation of Topology (*case: static nr. 1*)

- Effect of *different topologies* on the **execution time** of the algorithm ($i_{max} = 400$):



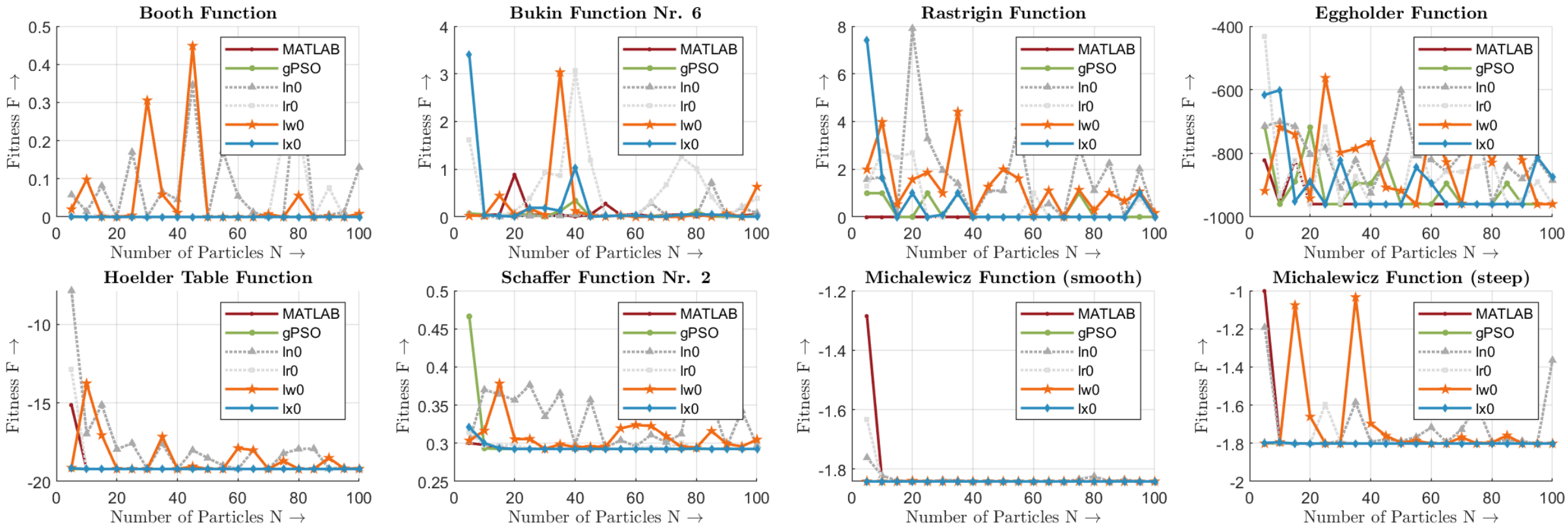
Performance Study
- Findings: Variation of Topology (*case: static nr. 1*)

- Effect of *different topologies* on the required number of **function evaluations** ($i_{max} = 400$):



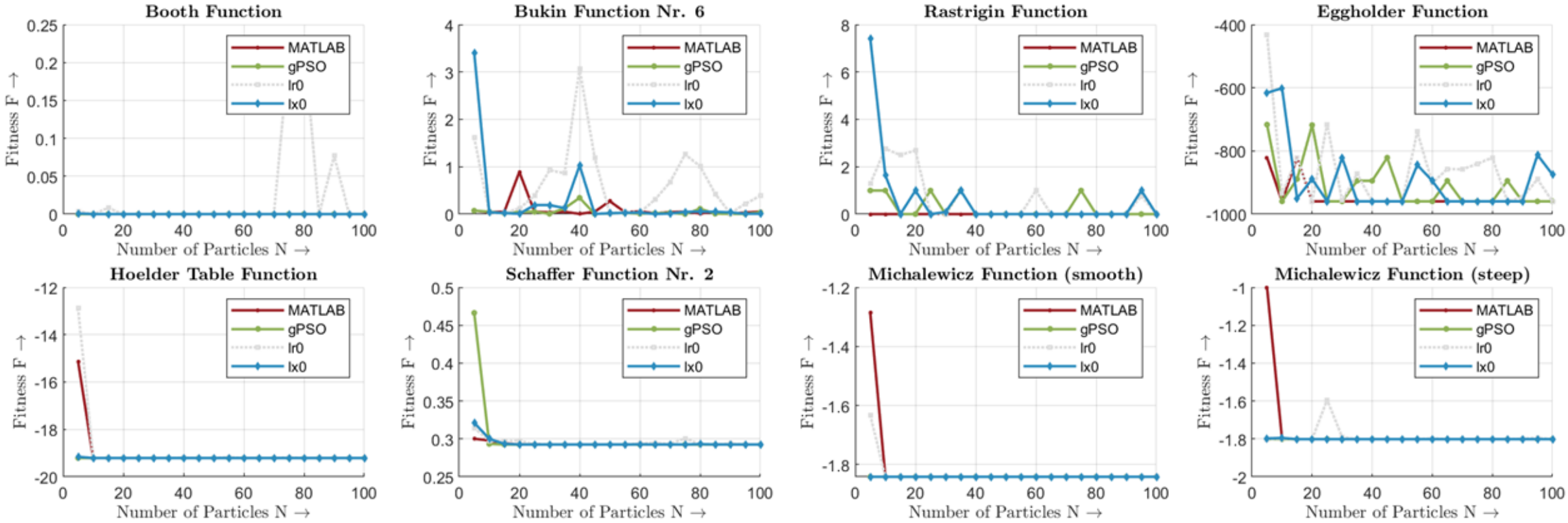
Performance Study
- Findings: Variation of Topology (*case: monarchy-based*)

- Effect of *different topologies* on the **accuracy** of the algorithm ($i_{max} = 400$):



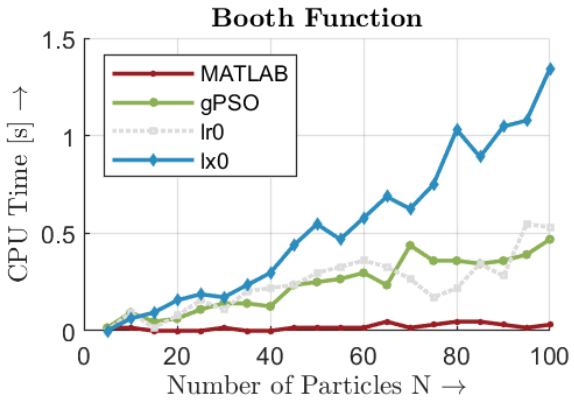
Performance Study
- Findings: Variation of Topology (*case: monarchy-based*)

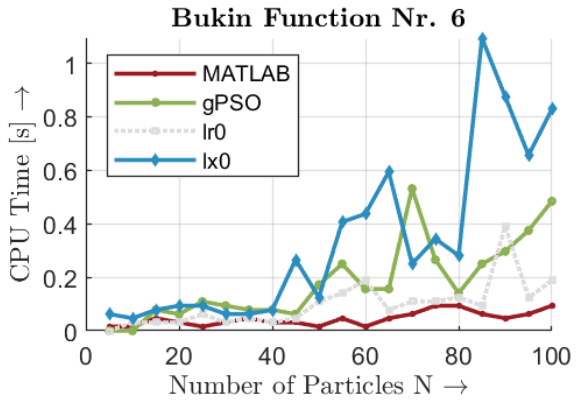
- Effect of *different topologies* on the **accuracy** of the algorithm ($i_{max} = 400$):

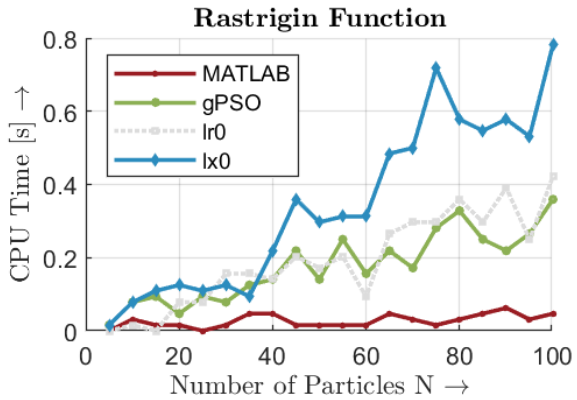


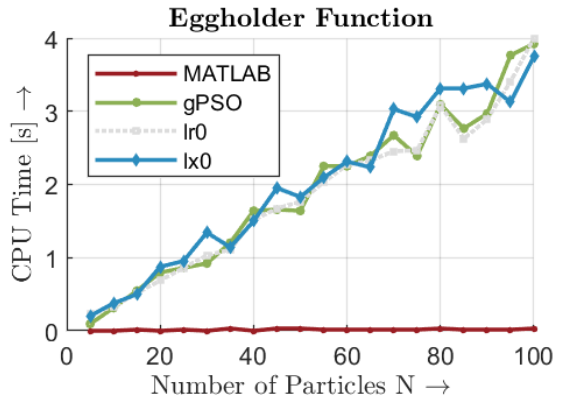
Performance Study
- Findings: Variation of Topology (*case: monarchy-based*)

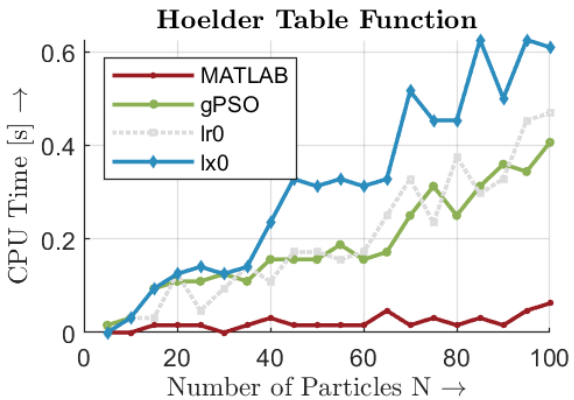
- Effect of *different topologies* on the **execution time** of the algorithm ($i_{max} = 400$):

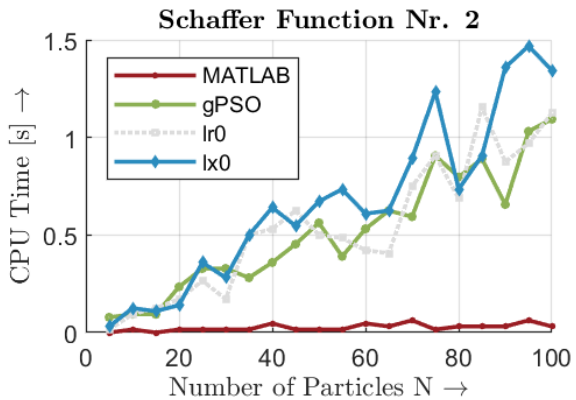


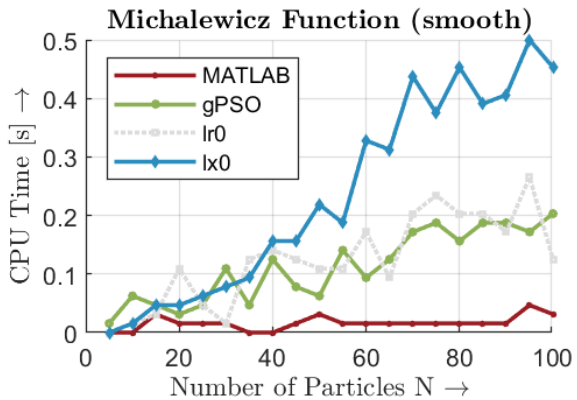


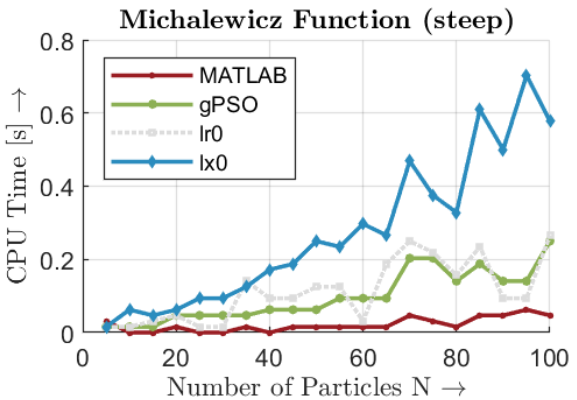






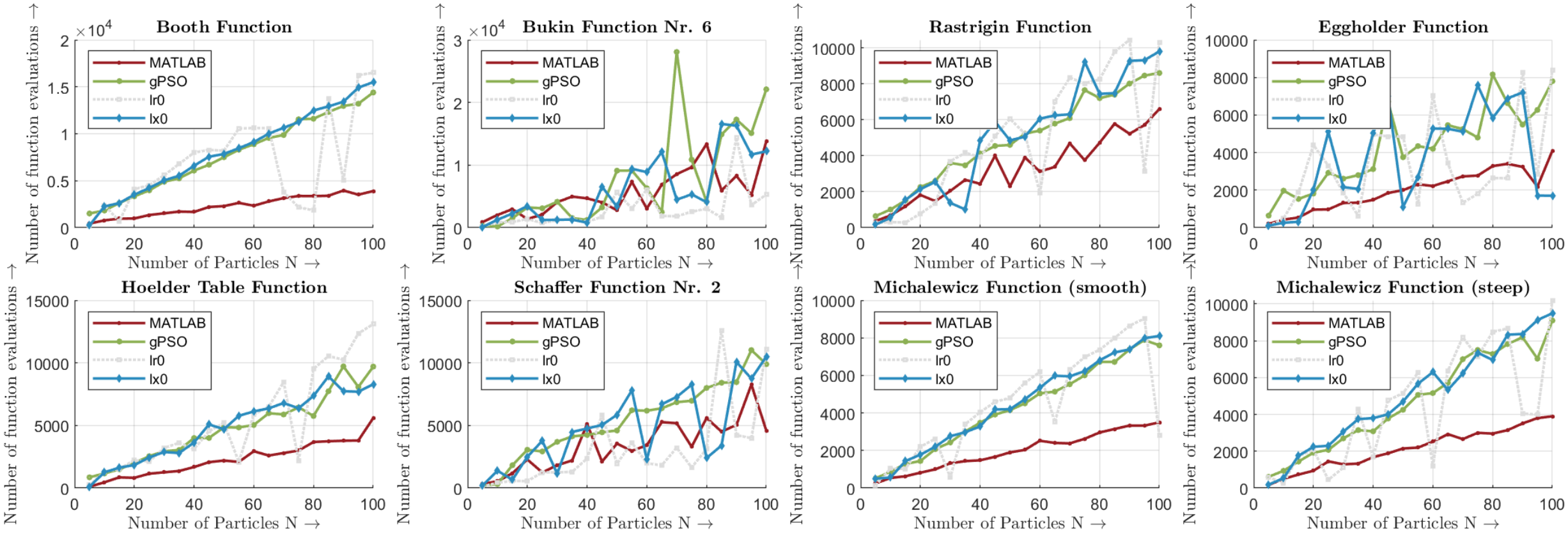






Performance Study
- Findings: Variation of Topology (*case: monarchy-based*)

- Effect of *different topologies* on the required number of **function evaluations** ($i_{max} = 400$):



Summary
- Conclusion & Outlook

Advantages:

- Implemented algorithm outperforms commercial MATLAB PSO-optimizer at low maximum iteration count.
- Stopping criteria allow high accuracy (e.g. in the range of $< 10^{-94}$) with the cost of higher iteration count.
  ⟶ *Question: Do we really need such a high accuracy? Probably not…*
- Provides full flexibility/adaptability to the user (*selection of topology, hyperparameters etc.*)

Disadvantages:

- Performance penalties due to extensive memory usage (⟶ potential for memory access optimization)
- Performance penalties due to “expensive” topology initialization and evaluation (*!! for-loop used*)
- Own algorithm required much more iterations & functions evaluations as commercial software by MATLAB.

Outlook:

- Adjustment of code to exploit thread- and data-level parallelism to allow HPC*.
  ⇒ MATLAB to CUDA/OpenMP code: enable GPU-accelerated computing.
- Improve memory usage ⇔ minimize memory accesses to improve computation efficiency.

* HPC = High Performance Computing

References
- Particle Swarm Optimization (PSO): A Performance Study

J. Kennedy and R. Eberhart, "Particle swarm optimization," *Proceedings of ICNN'95 - International Conference on Neural Networks*, 1995, pp. 1942-1948 vol. 4, doi: 10.1109/ICNN.1995.488968.

A.P. Engelbrecht, “Particle Swarm Optimization”. In *Computational Intelligence*, A.P. Engelbrecht (Ed.), pp. 289-358, October 2007, doi: 10.1002/9780470512517.ch16.

A. Ratnaweera, S. K. Halgamuge and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," in *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 240-255, June 2004, doi: 10.1109/TEVC.2004.826071.

Freitas D, Lopes LG, Morgado-Dias F. “Particle Swarm Optimisation: A Historical Review Up to the Current Developments”. *Entropy*. 2020; 22(3):362. doi: 10.3390/e22030362.

Z. Zhan, J. Zhang, Y. Li and H. S. Chung, "Adaptive Particle Swarm Optimization," in *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics),* vol. 39, no. 6, pp. 1362-1381, Dec. 2009, doi: 10.1109/TSMCB.2009.2015956.

E. T. Oldewage, A. P. Engelbrecht and C. W. Cleghorn, "The merits of velocity clamping particle swarm optimisation in high dimensional spaces," *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2017, pp. 1-8, doi: 10.1109/SSCI.2017.8280887.

ETH zürich

Thank you!

Any Questions?

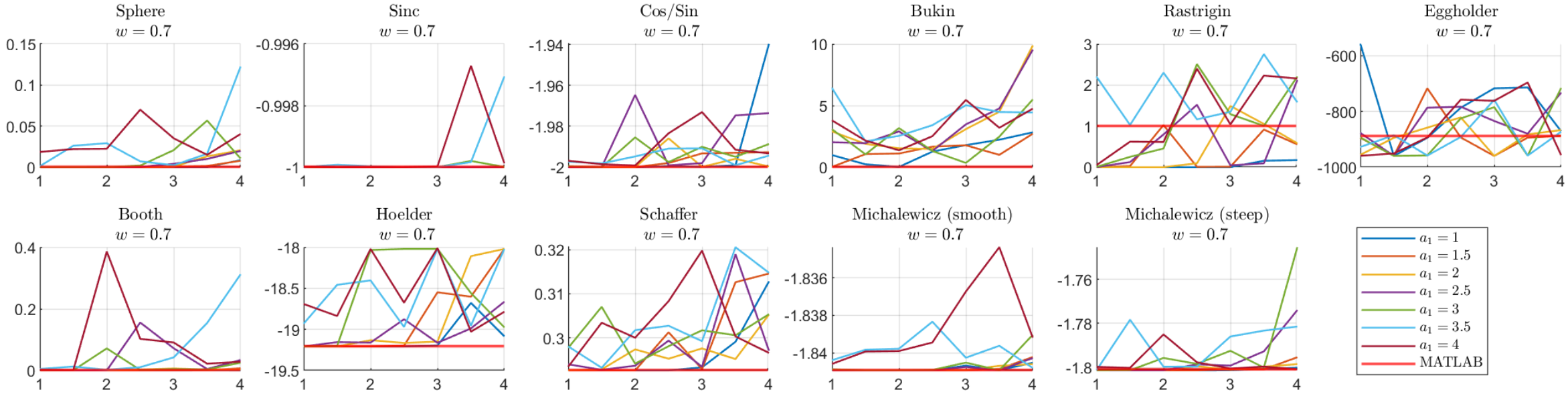
ETH zürich

Appendix

- **Appendix A**: Hyperparameter - How it was selected!
- **Appendix B**: Performance Study - Effect of Swarm Size (CPU time, function evaluations)
- **Appendix C**: Performance Study - Findings: Variation of Topology (case: dynamic)

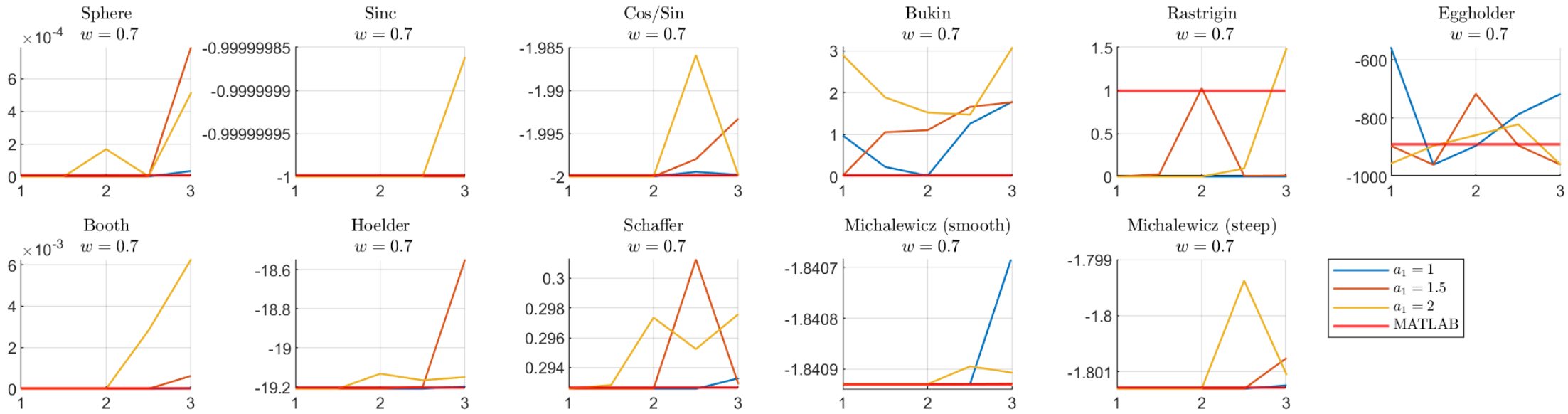
Appendix A: Hyperparameter
- How it was selected!

- **Step #1**: Evaluate PSO for variations of w, a_1, a_2 (44561 entries for distribution = ‘random’)
- **Step #2**: Find visually best $w \Rightarrow$ selected $w = 0.7$ (reduction to 4455 entries)
- **Step #3**: Pre-selection of $a_1, a_2 = [1, \dots, 4]$



Appendix A: Hyperparameter
- How it was selected!

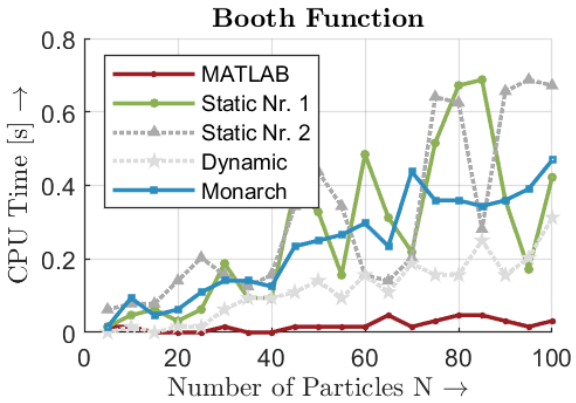
- **Step #1**: Evaluate PSO for variations of w, a_1, a_2 (44561 entries for distribution = 'random')
- **Step #2**: Find visually best $w \Rightarrow$ selected $w = 0.7$ (reduction to 4455 entries)
- **Step #3**: Pre-selection of $a_1, a_2 = [1, \dots, 4]$
- **Step #4**: Visual extraction of best a_1

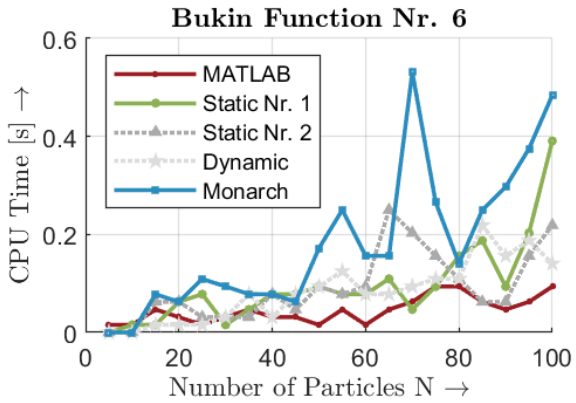


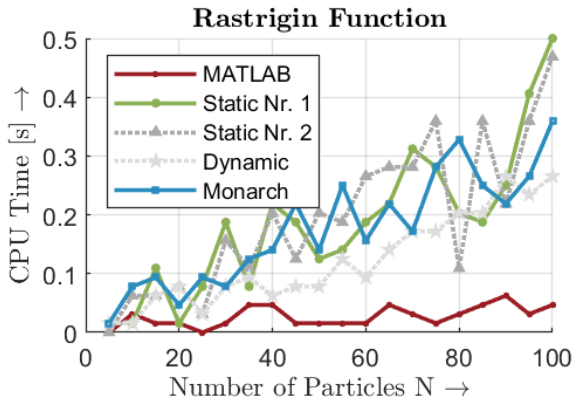
- **Step #5** : Final selection $\Rightarrow$ Winner: $w = 0.7, a_1 = 1, a_2 = 2$ (blue line; at least as good as MATLAB)

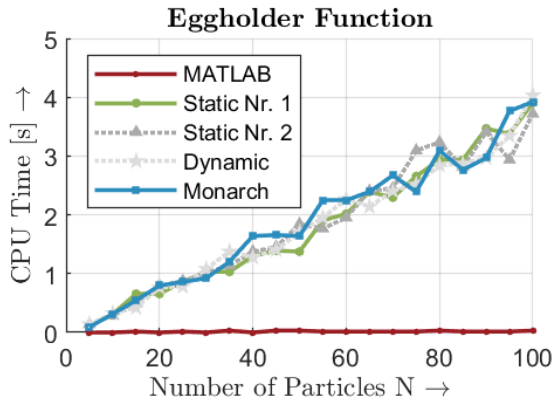
Appendix B: Performance Study
- Findings: Variation of swarm size

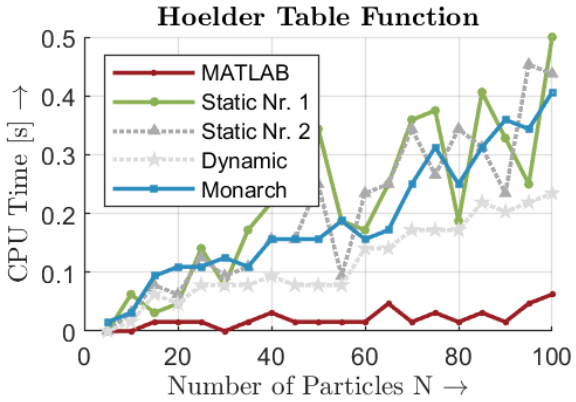
- Effect of *swarm size* (N) on the **execution time** (i_{max} = *400*):

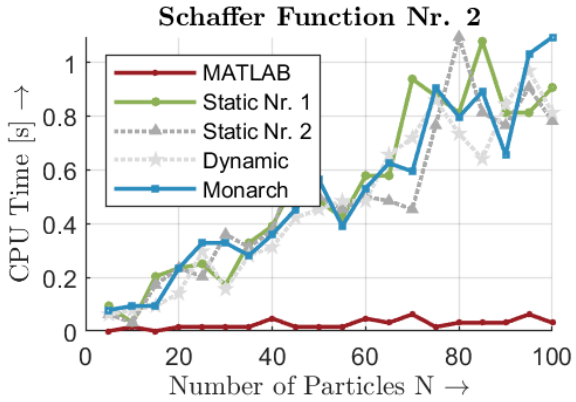


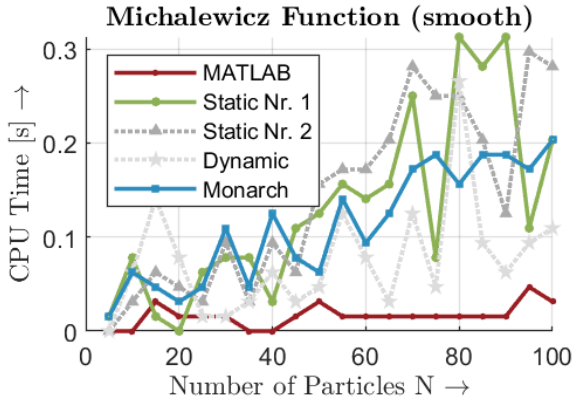


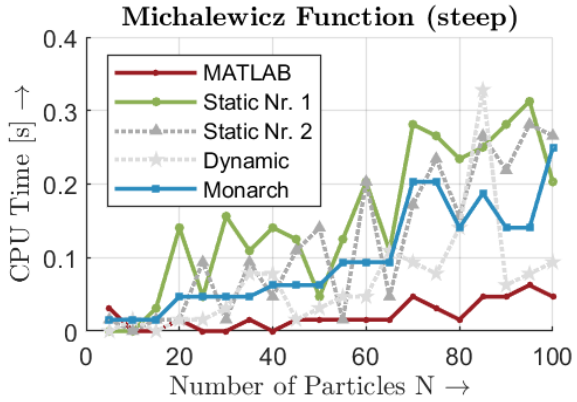






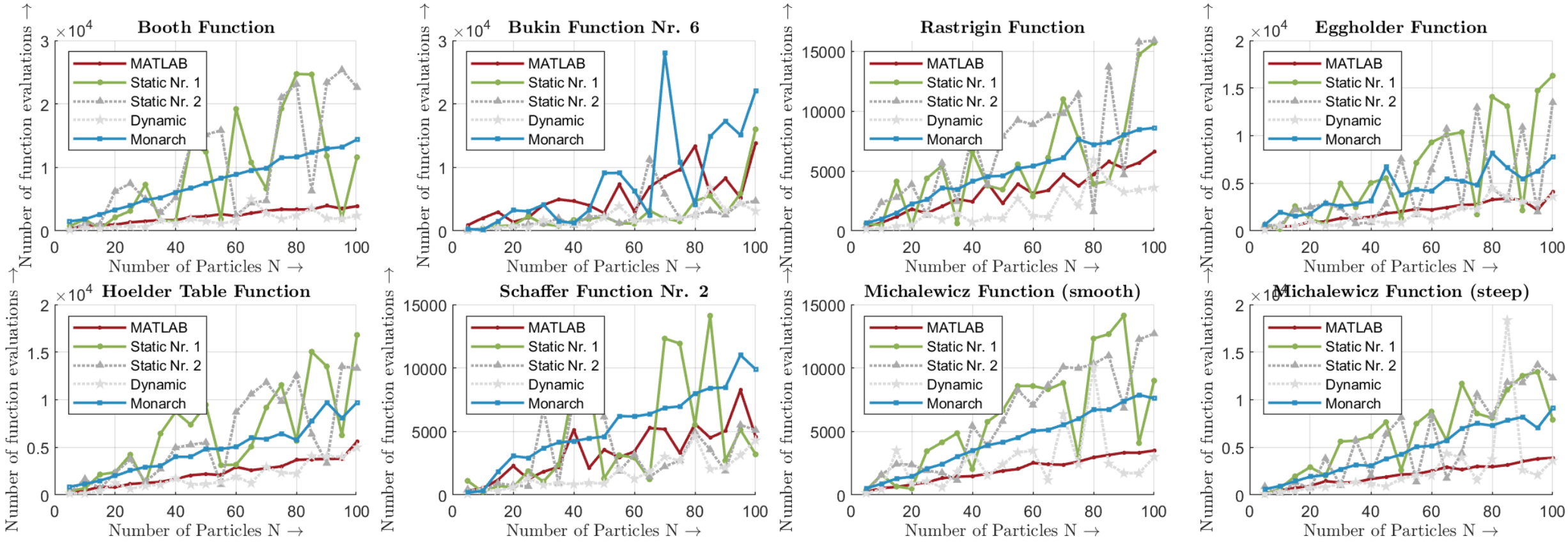






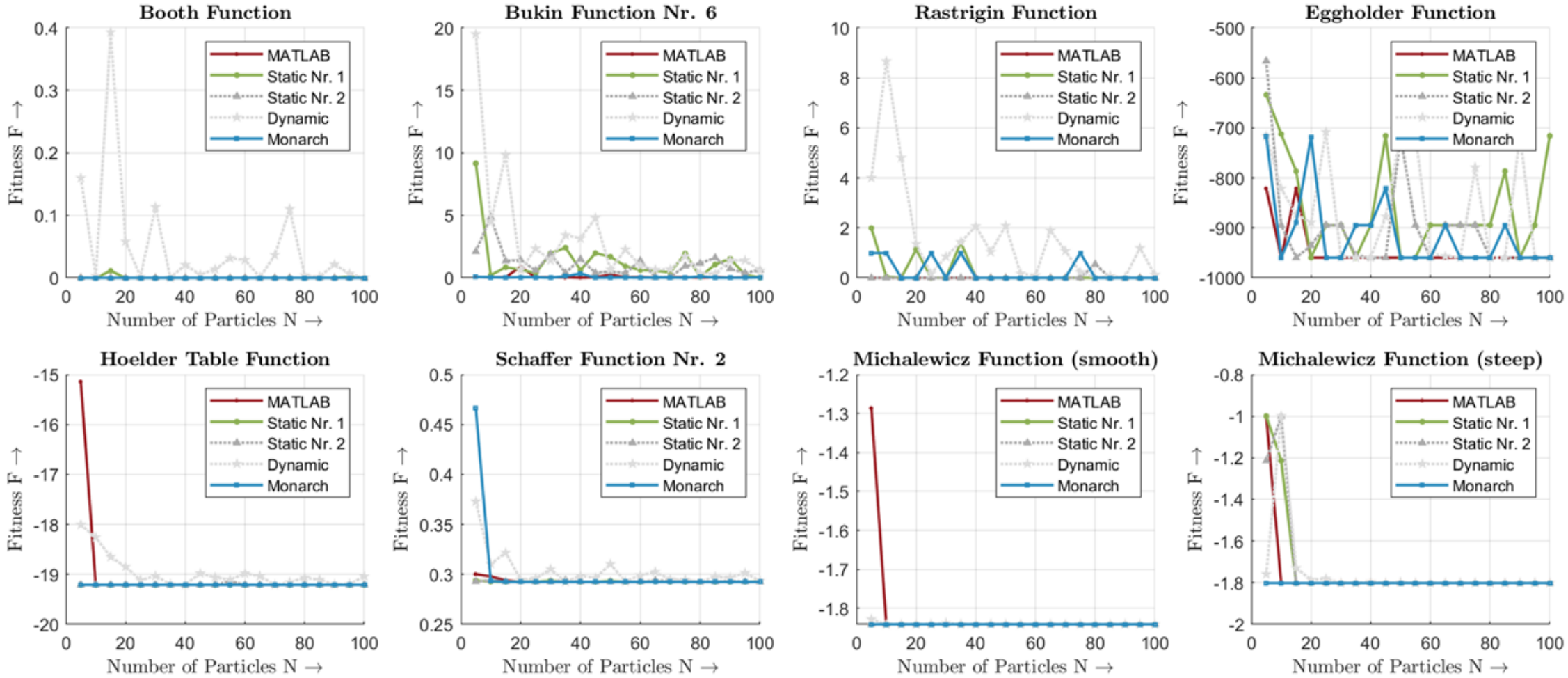
Appendix B: Performance Study
- Findings: Variation of swarm size

- Effect of *swarm size* (N) on the **function evaluations** (i_{max} = *400*):



Appendix B: Performance Study
- Findings: Variation of swarm size

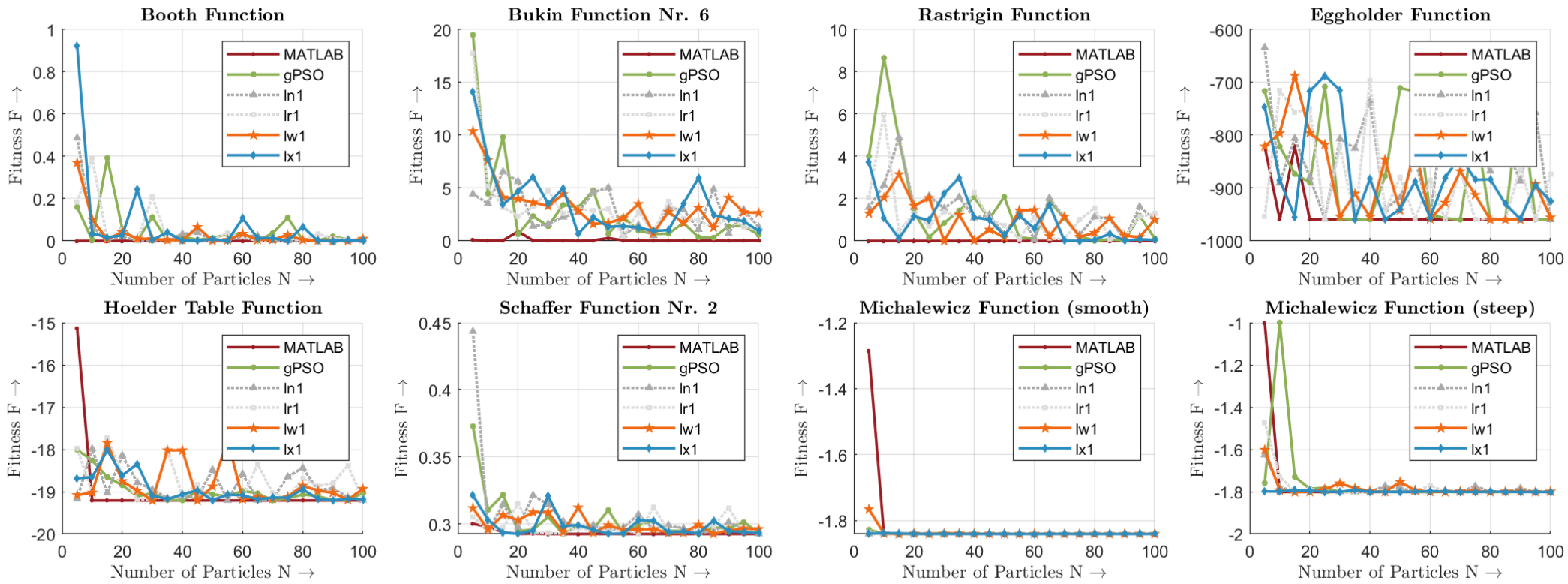
- Effect of *swarm size* (N) on the **accuracy** of commercial and private PSO* (i_{max} = *400*):



** Comparison of global PSO*

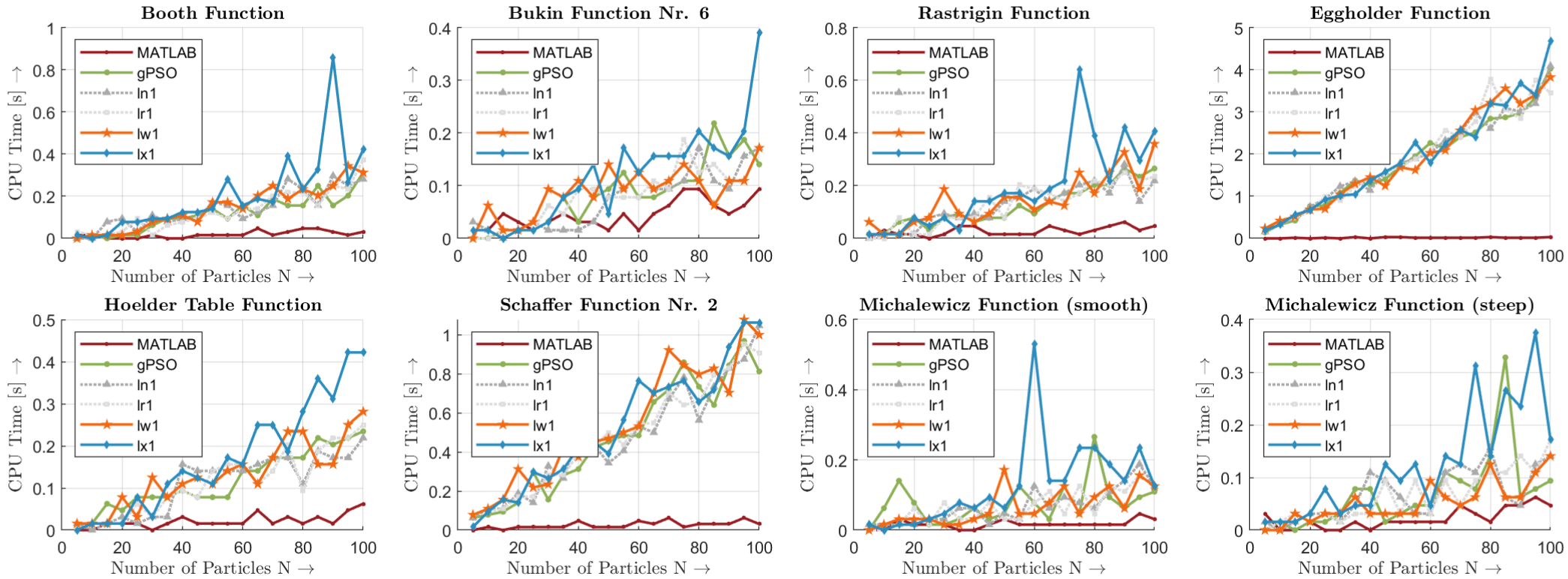
Appendix C: Performance Study
- Findings: Variation of Topology (*case: dynamic*)

- Effect of *different topologies* on the **accuracy** of the algorithm ($i_{max} = 400$):



Appendix C: Performance Study
- Findings: Variation of Topology (*case: dynamic*)

- Effect of *different topologies* on the **execution time** of the algorithm ($i_{max} = 400$):



Appendix C: Performance Study
- Findings: Variation of Topology (*case: dynamic*)

- Effect of *different topologies* on the required number of **function evaluations** ($i_{max} = 400$):