Universiteit Leiden
The Netherlands

Opleiding Informatica

Demystifying the Raven Roost Metaheuristic

M.S.Halsema

Supervisors:
Dr. Niki van Stein & Diederik Vermetten, Msc.

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

22/8/2023

1 General Introduction

Within the field of metaheuristics, natural processes have been a popular source of inspiration for novel optimization methods. This popularity can be attributed to the successes of a few well-known nature-inspired metaheuristics, such as Genetic Algorithms (Bremermann [5]), Evolutionary Strategy (ES) (Rechenberg [23]), Particle Swarm Optimization (PSO) (Kennedy & Eberhart [16]) or Simulated Annealing (Kirkpatrick et al. [17]). These methods not only provided an effective way of finding a near-optimal solution, but also an intuitive understanding due to the source of inspiration.

Using metaphors to describe the inner workings of the methods contributed to the applicability of these methods, and thus adhere to the more general sense of metaheuristics: metaheuristics as a "framework that provides a set of guidelines or strategies to develop heuristic optimization problems" (Sörensen [27]). The word metaheuristics, according to Sörensen, has however also been used to denote a "specific implementation of a heuristic optimization algorithm", possibly due to some of the more popular metaheuristics explicitly presenting it as such (both Ant Colony Optimization (ACO) (Dorigo [9]) and Simulated Annealing (Kirkpatrick et al. [17])).

The successes of well-known metaheuristics has led to an influx of nature-inspired, metaphor-based methods. In his paper, Sörensen [27] argues that many of the metaphor-based methods do not present any new ideas apart from the metaphor; many of these "novel" methods use ideas already well-established, simply renamed to suit the metaphor. He argues that, frequently, no attention is given to *why* a method is effective. Justification for publishing is found in what he denotes as an *up-the-wall-game*, where a method (frequently: an algorithm) is used as a black box: if better performance results are produced than another well-known metaheuristic implementation, it is published, regardless of whether or not this might be due to parameter tuning, or other problem-specific tailoring.

This is highlighted by Campelo and Aranha [7], who in addition to drawing attention to the high frequency of papers published based on metaphors, illustrate this problem in their catalog *Evolutionary Computation Bestiary* (Campelo and Aranha [6]), in which a great many metaphor-based methods are listed. They argue that the use of metaphors obscures the underlying mechanics and similarities to other metaheuristics, fragmenting the literature into niches. Furthermore, they stress that especially application-oriented journals seem vulnerable to the publication of novel metaheuristics, as researchers might not be as familiar with the field and thus gravitate towards recent papers with a lot of citations. Sörensen, Campelo and Aranha all argue that the reason metaphor-based metaheuristics are popular is due to the ease of finding "novel" metaphors, and thus producing papers, alleviating the pressure to publish. Sörensen emphasizes that many of the authors involved in publishing these kinds of papers also have a tendency to cite one another.

The problem of metaphor-based heuristics ultimately accumulated in a statement (Aranha, Villalón, Campelo et al. [1]) calling journals to not publish any "novel" metaheuristics unless the authors of these papers are able to properly

present their method within the known literature. This statement was subsequently signed by 95 prominent researchers within the field worldwide, showing that the issue is considered sufficiently important.

There have been attempts to analyze some of the more popular "new" metaheuristics in order to elucidate their position within the field as a whole. For example, the Intelligent Water Drops (Shah-Hosseini [26]), Cuckoo Search (Yang & Deb [39]), Grey Wolf (Mirjalili & Lewis [19]), Firefly (Yang [37]) and Bat (Yang [38]) algorithms have been analyzed by Villalón, Stützle and Dorigo [31], [33], [32], showing these to be either special cases of ACO (Intelligent Water Drops), $(\mu + \lambda)$ ES (Rechenberg [23]) (Cuckoo Search) or PSO. Harmony Search (Geem et al. [12]) has been analyzed as a special case of $(\mu + 1)$ ES (Schwefel [25]) by Weyland [36]. Many of these attempts focused on algorithms with a high number of citations in order to bring attention to how this reflects on the field.

2 Raven Roost Optimization

In this thesis, we analysed and benchmarked one of the not-yet-examined metaheuristics in the Evolutionary Computation Bestiary[6]. We will discuss the novelty of the mechanics and compare them to previous metaheuristics. To this end, we have chosen the Raven Roost Optimization algorithm (henceforth denoted RRO), first described by Brabazon, Cui and O'Neil in their titular 2015 paper [4]. The amount of citations at the moment of writing is around thirty, making it a reasonably popular paper. It is however interesting as to how other authors have appropriated RRO by publishing either an "improved" or hybrid form of the algorithm. We will discuss this last point in section 3.3.

RRO is inspired by the foraging behaviour of ravens, specifically the role of roosting as a form of information sharing. The use of birds as a metaphor is not new: it has been in use since BOIDS (Reynolds [24]), and has been a source of inspiration for PSO (Kennedy & Eberhart [16]). For RRO, the authors place the emphasis on the *social aspect* of foraging behaviour rather than the flight patterns, used in BOIDS and PSO.

The authors of RRO try to make a compelling case for why the avenue of social foraging for the use of algorithms could be interesting. In their paper, they seem to be at least familiar with recent literature in the specific subfield of ecology on foraging behaviour. The interest is highlighted in questions as to how food sources are found, and how these are communicated to other members of the species. It is remarked that these questions could potentially be relevant for any optimization strategies. Considerable time is spent on explaining the benefits of sharing information about food sources. The foraging behaviour of information sharing at a single site is ultimately picked as a focus of the paper, and given shape in the form of ravens roosting. An explanation as to *why* ravens exhibit roosting behaviour is given in the *Information Center Hypothesis* (Ward & Zavahi [35]). This hypothesis would suggest that roosts act as a place where ravens share information about food sources, and "recruit" other ravens for the purpose of safety and potential mate-finding.

For the development of the algorithm, four elements considered critical to the simulation of foraging behaviour are isolated: *individual perception of environment*, *personal memory*, *social transmission of information* and a *stochastic search component*. The authors remark that all foraging-inspired methods use these elements, albeit in "varying degrees". The RRO algorithm proposed incorporates all of these elements explicitly. They compare RRO with ACO[9] as well as PSO[16], and suggest that neither of these incorporate the element of individual perception explicitly, nor does ACO have an element of personal memory. Furthermore, only RRO has a central information point, which they call *social roosting*.

2.1 Algorithm Description

The algorithm proposed by the authors to test the effectiveness of social roosting and individual perception proceeds thus. First, during initialization, a random location within the domain is chosen, which is fixed for the runtime of the algorithm. This random location is denoted as *Roost*. The main loop starts from this point. N solutions, denoted *Ravens*, are selected randomly within the domain, evaluated, and initially saved to memory as personal best. The global best solution is selected, and called the *Leader*. For every solution, there is a probability $Perc_{follow}$ it will move towards their personal best, or select a random destination around the found global optimum in a hypersphere with radius R_{leader}. In either case, the solution will move from the roost position towards their destination (simulating a flying raven) in a given number of steps N_{steps}, with each step moving the solution a random amount [1] closer to the destination. At each stop, the current location is evaluated. Furthermore, a number of random solutions or *perceptions* N_{pcpt} are selected within a hypersphere with radius R_{pcpt} around the current position. If a better solution is found than the current personal best, there is a probability $Prob_{stop}$ that the raven does not continue moving towards its destination. For the ravens which follow the *Leader* and arrive at their destination, the destination is evaluated and the personal best is updated. Once all the ravens have stopped flying, the loop starts again: N random solutions are generated and accepted if evaluated as better. The global best solution is updated, and the *Leader* is chosen. The algorithm runs the main loop until a terminating condition is met. The algorithm in pseudocode can be found in Algorithm 1, and our interpreted pseudocode can be found in Algorithm 2. A visual representation of how the algorithm behaves in 2 dimensions can be found in Figure 1

2.2 Parameter Settings

When initialized, RRO uses quite a few parameters. Apart from the parameters known to every other swarm-based algorithm (i.e. number of particles N and

[1] Limited to a fraction of the total distance. In our implementation, we used $step_i = \frac{n_i \in [1, N_{steps}/2]}{Sum}$

Algorithm 1 Original Pseudocode for Raven Roost Optimization

Randomly select a roosting site;
repeat
  The N foraging ravens are assigned to N random locations in the search space;
  Evaluate the fitness of each raven location;
  Update the personal best location of each raven;
  The location of the best solution is denoted as LEADER;
  Recruit $Perc_{follow}$ percentage of the N foragers from the roosting site which will search in the vicinity of the LEADER (within the range of radius R_{leader} and the rest of the ravens will seek to travel to their personal best locations;
  Set $step = 0$;
  while $step < N_{steps}$ **do**
    On the way to its destination (whether the destination is the LEADER's vicinity or their personal best location), each raven flies for a while and searches in the vicinity of its current position (within the range with radius R_{pcpt});
    if *a better solution is found than the bird's personal best* **then**
      There is a $Prob_{stop}$ chance the raven will stop;
      Update the personal best location;
    else
      It continues to fly;
    end if
    $step = step + 1$;
  end while
  For the ravens which finally arrive at their destinations (the LEADER's vicinity or their personal best), update their personal best locations if necessary (to the fitness of the location) and evaluate the fitness for each forager;
  Update location of the best solution found so far if necessary;
until *terminating condition*;

the maximum number of iterations), it has multiple parameters related to its stochastic searching: $Perc_{follow}$, R_{leader}, R_{pcpt}, N_{pcpt}, N_{steps} and $Prob_{stop}$. To combat the inevitable complexity introduced for anyone who tries to apply the algorithm, a series of comparisons[2] were made by the authors and a "canonical" RRO defined, denoted RRO0: $R_{pcpt} = R_{leader} = \frac{R}{3.6\sqrt[D]{N}}$, where R is the radius of the search space[3], and D is the dimensionality of the problem, with $\frac{R}{1.8\sqrt[D]{N}}$ as alternatives in either, $N_{pcpt} = N_{steps} = 10$, with 5 and 10 as alternatives, $Perc_{follow} = 0.2$, with 0.4, 0.6 and 0.8 as alternatives, and finally $Prob_{stop} = 0.1$, with either 0.2 or 0.4 as the alternatives.

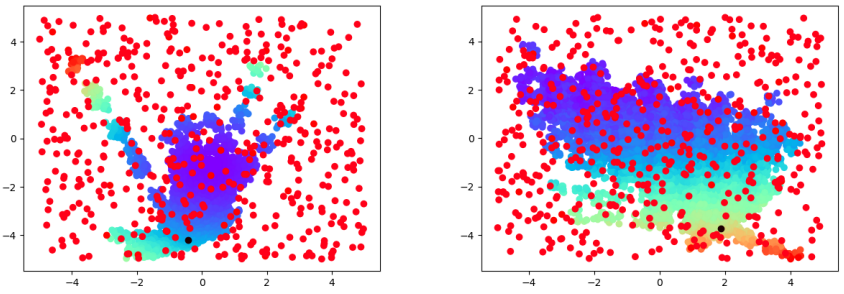

Figure 1: Graphical representation of RRO0 at $D = 2$ with $N = 30$ and 50000 evals. The black dot is the roost position, red dots are random solutions generated during the start of the loop, and the colouring of the other dots denoted the y-value of the solution found, with colder colours tending towards more optimal points.

2.2.1 Individual Perception

The authors emphasize the *individual perception* mechanism. To this end, they compared 13 versions of RRO (RRO0 to RRO12) with different selections of parameters, and a special version without the individual perception mechanism, but otherwise identical to canonical RRO (denoted RROv). The results show that RROv is outperformed by canonical RRO on almost every problem, and on several problems it performs worse than "canonical PSO". The outlier is Rastrigin, where RROv performs similarily and even outperforms canonical RRO in high dimensionality (i.e. 60). The authors conclude that individual perception is an important part of RRO. No attempt is made to see if a perception mechanism on PSO would improve the performance of PSO.

A similar analysis for sensitivity is performed for the size of the perception radius R_{pcpt}. RRO2 and RRO3 have a perception radius twice the size of canonical RRO. For most of the problems, RRO2 and RRO3 perform worse than other

[2]On DeJong, Griewank, Rastrigin and Rosenbrock with $d = [20, 40, 60]$

[3]In our implementation, we interpreted this as the average of all dimensions divided by 2

version of RRO, once again with the exception of Rastrigin, in which RRO3 performs even better than canonical RRO at high dimensionalities. Our suspicion is that Rastrigin is less suitable for the perception mechanism; this might be due to the global structure of Rastrigin, whereas a function like DeJong simply benefits from *any* area being searched thoroughly.

Finally, the authors analyze the sensitivity of the amount of perceptions per timestep. RRO4 and RRO5 perform 5 and 20 perceptions per timestep as opposed to the canonical 10. In all problems, similar performance to RRO0 is shown.

2.2.2 Stop Probability

RRO11 and RRO12 have the same settings as canonical RRO, but have $Prob_{stop}$ set to 0.2 and 0.4 respectively. In the orignal paperś results, it is shown that both versions perform significantly worse than canonical RRO. The authors conclude that "RRO0 > RRO11 > RRO12". Their explanation for this is when ravens stop more frequently, the search space is traversed less thoroughly. Interestingly, the authors do not attempt to compensate for ravens stopping by allowing for more evaluations elsewhere: by stopping, the algorithm simply loses evaluations[4]. Especially in such an implementation, it might have been valuable to compare the canonical RRO to a version which does not have stopping at all. The authors, however, do not consider this. Our suspicion is that removing the stop mechanism from RRO entirely, might have improved it in all cases. To verify this, we will test a version of RRO with the stopping mechanism removed by setting the parameters to 0 (see section 5).

2.3 Novelty of Raven Roost Optimization

The novelty of the Raven Roost Optimization heuristic is to be found in either the *social roosting* approach towards social transmission, the *individual perception* of each raven or the *step mechanism* approach towards stochastic search. The latter two will overlap somewhat, but are considered separately as the authors regard them as distinct. Personal memory simply uses personal best, and we will therefore not consider it.

2.3.1 Social Roosting

The simulation of *social roosting* is done by designating a fixed point within the search space from where the ravens start moving, and sharing the global best solution to exploit current found optima. A percentage of the ravens will follow the leader towards a destination near the found global best. The sharing of information about the global best to influence the search for solutions has been in use since PSO. In RRO, it is not the case that all the ravens are partly influenced by the global best solution, but instead only a part of the ravens is "fully" influenced. Which ravens are selected is randomly determined. Arguably, this

[4]Something that we did compensate for in our evaluation.

Algorithm 2 Interpreted Pseudocode Raven Roost Optimization

```
Roost ← 𝒰(lb, ub);
repeat
    for i ∈ N do
        Evaluate s ← 𝒰(lb, ub);
        Update personal best;
        Update global best if necessary: LEADER ← i;
        x_i ← Roost;
    end for
    Select Perc_follow of N ravens as follower;
    for i ∈ N do
        if follower then
            destination ← random within radius R_leader hypersphere of p_leader;
        else
            destination ← p_i;
        end if
        Determine step sizes: d_i ← [1..N_steps + 1] ∈ (0, 1) summing to 1;
        d_i = d_i(destination − Roost)
    end for
    step ← 0;
    while step < N_steps do
        for i ∈ N do
            x_{i,t} = x_{i,t−1} + d_{i,t}
            Evaluate x_{i,t} and N_pcpt solutions within radius R_pcpt;
            if found solution better than personal best then
                Update personal best;
                Stop flying with probability Prob_stop;
            end if
        end for
        step = step + 1;
    end while
    for every follower do
        if not stopped then
            Evaluate destination;
            Update personal best;
        end if
    end for
until terminating condition;
```

is to facilitate exploration: "... the best-so-far location does not influence the search process of all birds in each iteration of the algorithm".

The use of a single point, fixed for the entirety of the runtime, can be considered new. However, many algorithms use a single point within the domain towards special effect, such as PSO or Spiral Dynamics (Tamura & Yasuda 2011). In all of these, the "central" point is updated during runtime to either global best, or another location to facilitate exploration. In contrast, the roost location is never updated. Combined with how the stochastic search is performed, this leads to a significant emphasis on solutions found around the roost location. We will consider this in detail in section 2.5.

2.3.2 Individual Perception

The *individual perception* of the algorithm is simulated by random sampling at stopped locations within a hypersphere. The authors argue that neither PSO nor ACO have an individual perception mechanism, suggesting that this is a new approach. This is true for sampling within the neighbourhood of stopped locations, although particles within PSO evaluate each position after each timestep. The authors state that only one solution is evaluated at a time in PSO/ACO, whereas multiple solutions are evaluated every iteration in RRO. Whilst this is a definite difference from PSO/ACO, the idea of random sampling within a hypersphere is not new. Because multiple samplings are done each time the raven moves, the space roughly between the roost and the global best solution is thoroughly searched. Considering that there is no further mechanism to individual perceptions beyond this (i.e. more sampling further from the roost, location of random samples shifted towards a known solution etc.), it is not clear how the authors want to match intuition of individual perceptions to the mechanical reality.

2.3.3 Stochastic Search

The stochastic search of RRO can be divided into four parts adding randomness to the search. One of these parts is individual perception, which we already considered. The other three parts are: random step size, the probability of stopping, and generation of random solutions at each iteration. The random selection of followers could also be considered part of the stochastic search, but this has likewise been discussed.

On their way from the roost to the destination, each raven will stop along the way N_{steps} times. The formula for updating the position of the raven is given as follows:

$$x_{i,t} = x_{i,t-1} + d_{i,t}$$

where $x_{i,t}$ and $x_{i,t-1}$ are current and previous positions, and $d_{i,t}$ is a random (predetermined) step size, multiplied by the vector from roost to destination. The step sizes are determined per individual by "dividing their flight into N_{steps}". How these steps are divided exactly is not made clear, apart from being random. The movement of the ravens is thus entirely a straight

line from roost to destination, and differs from (canonical) PSO by not having "velocity-based movement". During "flight", the raven can "stop" at any found or perceived solution better than its current with a probability $Prob_{stop}$. It will only stop at better solutions, and can thus be considered greedy. The reason for the raven's stopping, is not made clear; it does not accept a worse solution for potential exploration, and any found solution is roughly between the previously found global or personal best and the roost. Therefore, there is a high probability that the same path is threaded in the next iteration as during the current, only more intensively, as it is divided over the same amount of steps.

2.3.4 Similarity to other Metaphor-based algorithms

At least three other metaphor-based algorithms use a similar pattern when it comes to exploration. All of these work by taking a single point and searching in straight lines outwards. The Fireworks Algorithm (Tan & Zu, 2010) picks random points from which solutions, or "sparks", are produced by random sampling within a hypersphere. The Grenade Explosion Algorithm (Ahrari & Atai 2010) and the Mine Blast Algorithm (Sadollah et al. 2012) do something similar, instead choosing "shrapnel" to find the highest places of "loss" (fitness). None of these sample randomly along the way, however.

The similarity to other swarm-based metaphors might be more apt from an "intuition" perspective. However, we feel that the original criticism of metaphor-based heuristics is that the metaphors are ultimately irrelevant, compared to actual search mechanics. As we could find three metaphor-based algorithms using similar search mechanics, whilst being intuitively starkly different from RRO, illustrates this. Categorizing RRO based on the intuition would ascribe unwaranted importance to the metaphor, and therefore misses the point.

2.4 Comparison to PSO

For PSO, the pseudocode can be described as follows:

Algorithm 3 Particle Swarm Optimization

```
for i ∈ N do
    x_i ← U(lb, ub);
    p_i ← x_i;
    Update global best if necessary: g ← p_i;
    v_i ← U(lb, ub);
end for
while termination condition not met do
    for i ∈ N do
        v_{i,t} = αv_{i,t−1} + φ1 r1(p_i − x_{i,t−1}) + φ2 r2(g − x_{i,t−1});
        x_{i,t} = x_{i,t−1} + v_{i,t};
        Evaluate x_{i,t};
        Update personal best;
        Update global best if necessary: g ← p_i;
    end for
end while
```

Every particle is assigned a random vector within the search space as their original position and thus personal best. The global best is denoted and saved to memory. All particles are assigned a random velocity vector. For every particle, until a stop criterium is met, their velocity $\overrightarrow{v}$ is updated according to the formula:

$$\overrightarrow{v}_{i,t} = \alpha \overrightarrow{v}_{i,t-1} + \phi_1 r_1(\overrightarrow{p}_i - \overrightarrow{x}_{i,t-1}) + \phi_2 r_2(\overrightarrow{g} - \overrightarrow{x}_{i,t-1}) \tag{1}$$

Where $\overrightarrow{v}_{i,t}$ is the velocity vector of particle i at time step t, α is the inertia weight, $\phi_1 and \phi_2$ are respectively the cognitive and social coëfficient, r is a random factor, $\overrightarrow{p}_i$ is the personal best, and $\overrightarrow{g}$ is the current global best. The position of the particle is then updated according to the formula

$$\overrightarrow{x}_{i,t} = \overrightarrow{x}_{i,t-1} + \overrightarrow{v}_{i,t} \tag{2}$$

When looking at the pseudocode, it seems that RRO and PSO have a very similar general structure, although RRO has a lot more parameters to keep track of. Both algorithms start with a randomization step, an assigning of the initial positions, and give a special role to the best-found solution. Both then follow a main loop until a termination criterium is met, although in the case of RRO this is divided up into steps. The movement of the particles is rather different at first sight, mainly because PSO's movement has a few more factors; the updating of the position is highly similar. Hence we will try to reduce PSO's movement to RRO.

First, we can state $\alpha = 0$ to remove all inertia, and $r_1 = r_2 = r$, our formula could be reduced to:

$$\overrightarrow{v}_{i,t} = \phi_1 r(\overrightarrow{p}_i - \overrightarrow{x}_{i,t-1}) + \phi_2 r(\overrightarrow{g} - \overrightarrow{x}_{i,t-1}) \tag{3}$$

$$\overrightarrow{v}_{i,t} = r[\phi_1(\overrightarrow{p}_i - \overrightarrow{x}_{i,t-1}) + \phi_2(\overrightarrow{g} - \overrightarrow{x}_{i,t-1})] \tag{4}$$

Because we can define any position within the search space as a combination of vectors, we can add a third vector, $\overrightarrow{u}_i - \overrightarrow{g}$, assign it ϕ_2 as a coëfficient, and ignore the personal best by setting $\phi_1 = 0$, to simulate a raven moving towards a random point within a hypersphere of the global best g with a random step size:

$$\overrightarrow{v}_{i,t} = r\phi_2[(\overrightarrow{u}_i - \overrightarrow{g}) + (\overrightarrow{g} - \overrightarrow{x}_{i,t-1})] \tag{5}$$

Likewise, we could set $\phi_2 = 0$ to simulate the ravens moving toward their personal best with a random step size:

$$\overrightarrow{v}_{i,t} = r\phi_1(\overrightarrow{p}_i - \overrightarrow{x}_{i,t-1}) \tag{6}$$

Furthermore, as initially stated in Kennedy and Eberhart's original paper on PSO [16], dividing particles into groups with special purpose is an expected and considered mechanism of PSO: "Another version considered using two types of agents, conceived as "explorers" and "settlers." [...] The hypothesis was that explorers would extrapolate outside the "known" region of the problem domain, and the settlers would hill-climb or micro-explore regions that had been found to be good."

Instead of using random r for every timestep, we could use an array (or vector) $r_{i,1}, r_{i,2}, ..., r_{i,n}$ with $n = N_{steps}$ and the further constraints that $\sum_{j=1}^{n} r_{i,j} = 1$, and instead of from the current position, the vector is calculated from the roost. Finally, a random fixed location is chosen as the initial location for each particle, and reset during each iteration. This way, the behaviour of particles would be the same as the ravens from RRO.

Thus, save for the "inner loop", stopping mechanism and perception mechanism, RRO could be rewritten as a special case of PSO. Both PSO and RRO leave the termination criterium up for interpretation. In the canonical implementation of RRO the stopping mechanism simply removes evaluations. Therefore RRO could be considered a special case of PSO, plus random sampling of the environment "along the way".

2.5 Inverse Square Law

As all ravens will start their movement at every iteration at the roost, stop after a random distance and sample the environment a certain amount of times, the space searched most thoroughly will always be around the roost point. All ravens potentially sample the same amount of times on the way to their destination

(ignoring the destination sampling of followers), potentially even less amount of times at the end of their flight due to the stopping mechanism. Due to this methods of sampling, we suspect the inverse square law of distance roughly applies to the intensity of sampling around the roost, with a stronger effect in higher dimensions. No apparent effort by the authors was made to remedy this problem: the dividing of the flight path is done randomly.

In order to see if our prediction is true, we plotted the euclidian distance of every sampled position from either the roost point and the optimal point in a box plots below. For this experiment we used $N = 30$, 50000 evaluations per run and repeated the experiment for 30 runs for dimensions 2, 5 and 10. Otherwise, the parameter settings were identical to RRO0. The box plots below show the cumulative sample distances over all runs.

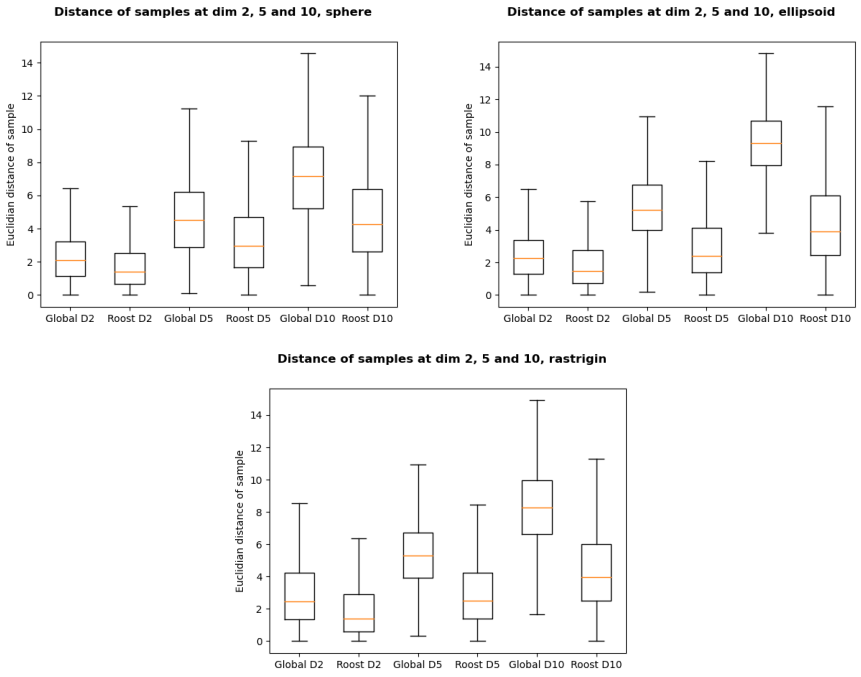

Figure 2: Plotted euclidian distance to either the global optimum or roost position, with N=30, 50000 evals and 30 repeats.

From the figures above we can clearly see that RRO has a strong tendency to sample around the roost point, and that the difference between the roost and the global optimum becomes more pronounced at higher dimensionalities. Although some sampling around a starting point could be considered logical, we feel that a lot of unnecessary sampling could be avoided by having a more structured approach towards either the perception mechanism or the step division.

3 Other work

RRO has been cited a fair amount of times as per the time of writing. Many of the citations are either a review of metaphor-based metaheuristics, or simply cite RRO without using it (i.e. Hussain et al [15] and Rajwar et al. [22]), possibly in order to count which metaphors are used. Other authors mention it when discussing applications, i.e. task scheduling (Nabi et al. [21] or Barghavi et al. [2], the latter finding RRO highly applicable), or when introducing their own novel metaheuristic (Moldovan [20]). Two citations of RRO stand out in particular. One by the orignal author of RRO, and another by other researchers for the introduction of an improved version. These will be discussed in 3.2 and 3.3 respectively.

3.1 Earlier analysis

Earlier attempts to explicitly place RRO within the literature have been made by Eugwu et al. [11], defining RRO as a swarm-based algorithm. Dragoi & Dafinescu [10], place RRO within the category of "food-based bird-inspired" methods, along with eight other methods, showing foraging behaviour to be a rather inspiring topic to many authors. Neither paper goes into much detail about the workings of RRO, attempting to serve as overviews of metaphor-based algorithms.

3.2 Response to criticism of metaphor-based algorithms

In a talk given in 2017 [3] the main author of RRO, Anthony Brabazon, acknowledges the criticism of Sörensen [27] towards the field of metaheuristics. The problem of identifying differences and similarities between "novel", specifically foraging-inspired, metaphor-based algorithms is recognized as warranting consideration. The author's response to this problem is by suggesting taxonomies of different foraging-inspired swarm metaheuristics, in order to make a better comparance. As we saw before, more attempts have been made by other authors [11] [10] to provide a taxonomy of metaphor-based algorithms, so this is a plausible approach. Some of the suggested taxonomies are:

- *Tree of Life* - taxonomy based on the relationship of the species in the animal kingdom.
- *Foraging Behaviour* - taxonomy based on whether the agents are solitary or neighbour-influenced.
- *Foraging Capabilities* - taxonomy based on different attributes or mechanisms of the species, including:
  - *Sensory* - what sensory modalities the species / algorithm has. i.e. vision, sound, chemicals etc.
  - *Memory* - how do the agents remember. i.e. personal, global, environment-based etc.

– *Communication* - how do the agents communicate? i.e. through pheromones, roosting site etc.
– *Stochastic* - what stochastic element is used in the traversal of the search space?

The author argues that more taxonomies are possible, and that the list is by no means exhaustive. Interestingly enough, the mechanisms mentioned as part of foraging capabilities are very similar to the four mentioned in the RRO paper to compare the algorithm with PSO and ACO. Three of these four capabilites we used earlier to analyse the novelty of RRO, in order to easily compare to what the authors claimed. The capabilities mentioned however, were ultimately still inspired by *what biologists consider foraging behaviour*. Due to the high overlap of elements (i.e. the RRO perception mechanism is still just a stochastic element) and the emphasis on the metaphor as basis for classification, the taxonomies suggested seem of limited use. This last point is mentioned by the author himself, cutting to the heart of the matter: "... the ways in which sensory mechanisms are implemented in algorithms can sometimes be quite similar, despite claims that they arise from different sensory modalities".

The author goes on to argue that by using the four mentioned capabilities, it is possible to construct a metaframework for foraging-based algorithms: "In spite of the multiplicity of foraging inspired algorithms, a relatively compact metaframework can be outlined which encapsulates most existing algorithms". Generalizing the four capabilities would allow an algorithm designer to compare their novel creation to already-existing methods, and thus avoid the pitfall of reinventing the wheel, mentioned earlier. This allows the designer quite a bit of creative freedom: "[a]n unlimited number of specific algorithms, with differing search characteristics, can be created within this general metaframework". The author ultimately still emphasizes the metaphor, and points of comparison inspired by the metaphor, to solve this problem.

3.3 Improved Raven Roost Optimization

The most prolific referencers of RRO are Torabi and Safi-Esfahani, as being the authors of the *Improved Raven Roosting Algorithm* [28], henceforth denoted IRRO. In their paper, they recognize some problems with RRO, namely the lack of exploration and premature convergence. The solution they propose is twofold: they replace the $Prob_{stop}$ with a $Food_{st}$, which is time-based, and select the ravens following the leader based on personal best, denoting them *weak* and *greedy* ravens. In the same way RRO is benchmarked, they test their parameters in different versions (denoted IRRO0 to IRRO8) and compare them to all the versions of RRO and some well-known algorithms (PSO and Chicken Swarm Optimization [18]).

The $Food_{st}$ parameter starts at $Food\_Max$ (in their implementation set to 1), and gradually diminishes over time according to the formula:

$$Food_{st_{t+1},j} = Food\_Max \frac{MaxIt - i}{MaxIt}$$

Where $MaxIt$ is the maximum number of iterations (meaning the amount of times it passes **repeat** in the pseudocode), and i is the current iteration. This parameter is updated at the end of every loop, before the roosting stage. Considering the ravens will probably stop more easily before reaching their destination, it seems that this would not encourage exploration and in fact, worsens the tendency to search around the roost. As the $MaxIt$ parameter has to be known beforehand, the authors' implementation cannot accomodate for any "lost" iterations either.

The division between *weak* and *greedy* ravens is done based on personal best. The percentage "Weak_Population_Percent" with the lowest evaluated personal best is denoted *weak* and will follow the leader. Why this would help either premature convergence or exploration is not made clear by the authors. This mechanism tends more towards greed, and would therefore be more prone to getting stuck in local optima, a point acknowledged by the authors.

The values tested in IRRO for "Weak_Population_Percent" are $0.2, 0.4, 0.6$ and 0.8. The percentage of 0.4, claimed by the authors as the best setting, is tested in version IRRO6. This version however, does not consistently outperform IRRO0 (same as canonical RRO, with 0.2 as percentage), nor does IRRO6 perform the best of all the other algorithms in any problem. The authors claim that it performs better in higher dimensionalities, yet the published data (as appendix to the paper) does not support this. This claim is made once more in their later papers [30], but again is not supported.

According to the results published, RRO is rather consistently outperformed by IRRO. IRRO5 seems to steadily outperform the other versions. IRRO5 is "canonical" IRRO (same parameters as canonical RRO) with more steps (20 as opposed to 10). We assume the number of iterations is changed to accomodate the amount of steps, although this is nowhere mentioned by the authors.

Considering we suspect that stopping is not a useful mechanism at all, it is rather disappointing that the authors of IRRO do not test whether or not this is an actual improvement on the stop mechanism. Both changes are not tested individually, but at the same time. Therefore it is not in fact clear whether or not food mechanism is an improvement. For this reason, we will test this ourselves in different versions of RRO (see section 5).

The authors of IRRO use their novel algorithm in two more papers. In both they combine it with Chicken Swarm Optimization (CSO) [18], another rather popular metaphor-based metaheuristic, showing some structural similarities to PSO, and apply it to the problem of task scheduling. The reasoning is that CSO suffers from premature convergence, an issue apparently solved by IRRO when compared to RRO (although never explained how). The resulting algorithm, IRRO-CSO [30] [29] is simply a concatenation of IRRO and CSO: first IRRO is run, after which the population of ravens (presumably meaning the personal bests) is used to initialize the positions for the chickens in CSO. It is not made clear how this solves premature convergence, and nowhere is the algorithm explained further beyond a flowchart placing the entirety of IRRO before CSO.

Considering CSO already performs well on the problem of task scheduling,

it is not entirely surprising that using IRRO for pre-selection instead of random initialization generates better results. The study is considered a success, and suggestions are made for further research into using IRRO in combination with other well-known algorithms.

4 Method

We will compare the performance of several version of RRO to a few well-known standard metaheuristics (PSO, random search and CMA-ES-CSA [14]) on the Black-Box Optimization Benchmarking (BBOB) [13] suite of 24 continuous problems. BBOB contains many different functions for testing optimization algorithms, and has been used extensively in the past decade. CMA-ES-CSA is a version of evolution strategies considered to be state-of-the-art since the introduction in 2004, and is thus used to place RRO and PSO within the field as a whole. For each algorithm we will allow a budget of 50,000 evaluations, and repeat the experiment 100 times on 2, 5, 10 and 20 dimensional problems with a population of $N = 30$.

Furthermore, we will compare the performance of canonical RRO to a version without stopping. To compensate for the amount of evaluations, our interpretation allows for more evaluations when a flight is prematurely abandoned. The extra evaluations will "spill over" into the next round of random sampling and flying.

4.1 Implementation

The experiments were done on an AMD Ryzen 7 5700G at 3.8 GHz and 16GB RAM. The implementation of the RRO was done in Python 3.11. The benchmarking software used was IOHprofiler (Doerr, Wang, Ye, van Rijn, Bäck [8]), using the BBOB problem suite, and comparison was done using the IOHanalyzer website (Wang, Vermetten, Ye, Doerr, Bäck [34]). The comparison results of PSO, random search and CMA-ES are part of the IOHanalyzer.

4.2 Interpretation of Metaphors

When designing the implementation, we tried to stay as true to the proposed algorithm as possible. Due to the wording of RRO, a few liberties in interpretation had to be taken. The original algorithm can be found in Algorithm 1, and our interpretation in Algorithm 2.

4.2.1 Stopping and flying

In the original pseudocode of the algorithm, a raven only explicitly continues to fly if the current solution found is not better than the personal best. If a better solution is found, there is a probability it will stop, but not explicitly mentioned it will continue flying if not. In the verbatim description of the algorithm it

is strongly implied that the raven will continue flying if it does not stop, and furthermore only stops for the rest of the iteration if it does.

4.2.2 Updating personal best

Likewise, in the original pseudocode, the personal best of any one raven is only updated during either initialization, roosting, arriving at their destination as a follower or when it stops during flight. It is not made clear if personal best is always updated even when the raven does not stop: updating the personal best is done after stopping. The pseudocode states the updating of the personal best quite explicitly, suggesting this is not simply done after every sampling. In the textual description of the algorithm, it is simply stated "If a better location is perceived [...] there is $Prob_{stop}$ chance that the raven stops its flight at that point and forages at the newly-found location; otherwise, it takes another step and continues to fly to its destination". In our implementation the algorithm seemed much more prone to exploration when the personal best was always updated: there are simply less moments when it can stop as it already accepted a better solution. This interpretation, however, makes it hard to justify the purpose of the stopping from a mechanical perspective. Less of the search domain is explored, and upon subsequent iterations the raven will, with high probability, fly to this point again, intensifying sampling along a path already searched.

5 Experiments

In order to thoroughly analyze RRO, a few distinct versions were benchmarked. First, canonical RRO (`RavenRoost`) with all the parameters as the original authors recommended. Second, to test whether or not stopping is a useful mechanic, another version with $Prob_{stop} = 0$ is compared (`RavenRoostNoStop`), effectively removing the stop mechanism. The other parameters are identical to RRO0.

To test which of the mechanisms of IRRO improve the algorithm, they are tested in two different versions. The first, `RavenRoostFood`, uses the food mechanism instead of a stop percentage. Considering our implementation compensates for evaluations lost, the formula for the food percentage uses evaluations instead of iterations. The $Food_Max$ factor is still 1 as in original IRRO. Although not entirely the same to the mechanism intended, the overall working can be expected to be the same. The second, `RavenRoostWeak`, uses the elitist selection mechanic introduced by IRRO. This mechanism is exactly the same as the authors of IRRO intended. Finally, a combined version of these two is tested, with both the food and the elitist mechanic (`ImprovedRavenRoost`). Apart from the mechanisms, all the parameters are the same as canonical RRO (including the percentage used to recruit "weak" birds, differing from the IRRO paper).

6 Results

The results we are most interested in are the distribution of the expected best value, per algorithm, per function, and the expected value over runtime. The former gives a good overview of the performance of all algorithms for a given function, and gives a reliable image whilst being resistant to outliers in the data. For this we used a probability density function, colloquially known as a violin plot. The latter shows a good estimate of what can be expected after a certain amount of iterations, and furthermore gives a good estimation of the rate of convergence, prematurely or otherwise.

For most of the functions a proper comparison between CMA-ES-CSA, PSO, Random Search and the different versions of RRO could be made. The exception is the *LinearSlope* function, for which not enough data was available for PSO and CMA-ES-CSA. For this reason, RRO is only compared to Random Search in this function. For all the other versions, the data insofar available was used to compare.

Furthermore, functions *StepEllipsoid*, *Gallagher101* and *Gallagher21* with $D = 2$ did not display correctly in the IOHAnalyzer software. To remedy this, the experiments were rerun on $D = 20$.

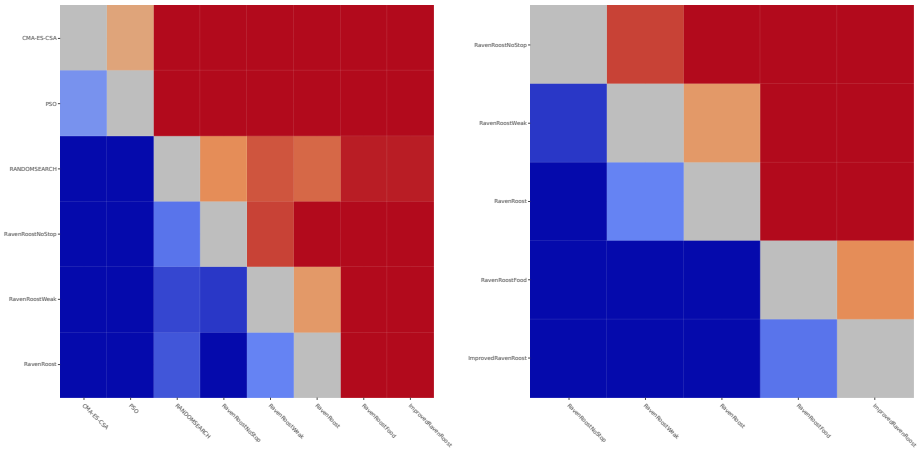

Figure 3: The IOHAnalyzer ranking heatmaps for all tested dimensions. X compares favourably to Y if it is red, unfavourably if blue.

The expected value over runtime plots generated for all algorithms are to be found in Appendix A. The results per function, condensed in probability distribution plots of best-found result after 50,000 evaluations, are to be found in Appendix B.

7 Discussion

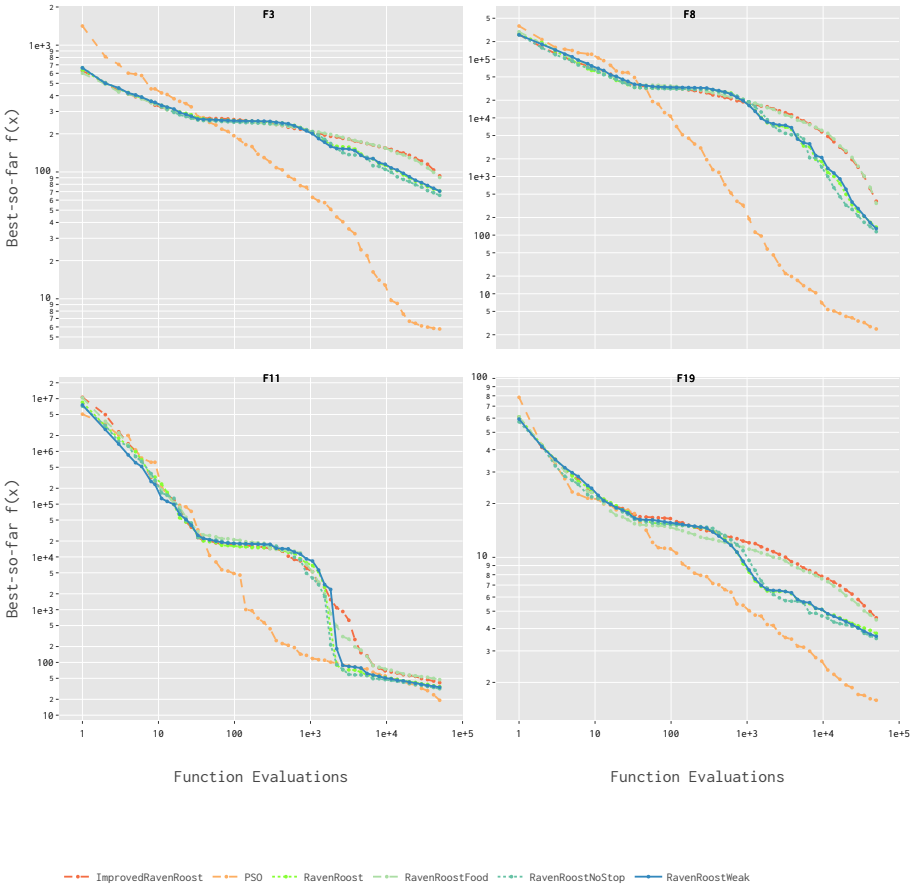

Figure 4: The expected value over time on Rastrigin (f3), Rosenbrock (f8), Discus (f11) and GriewankRosenbrock (f19) for D=10

7.1 Performance compared to other algorithms

From the results in Appendix A and B, it is first and foremost clear that CMA-ES-CSA outperforms the other algorithms by a fairly large margin. Surprisingly, and in contrast to the results from the authors of both RRO and IRRO, PSO steadily outperforms them both. At worst, it seems to perform similarily in Katsuura at dimensionality 10 (Figure 22). Somewhat strikingly, RRO performs marginally better than Random Search for the same amount of evaluations, sometimes being beat by it when it comes to the average best result (i.e. *BuecheRastrigin* D=2 (Figure 12), *LinearSlope* D=2 (Figure 13). In the case of

LinearSlope). This could be attributed to the algorithm not going beyond its own best-so-far, see Figure 1, and thus not reaching the edges where the optimal solutions are.

RRO performs somewhat more competatively to PSO at lower dimensions. The issue with RRO seems to do with a lack of exploration (having no explicit mechanism to go "beyond" its own borders like PSO does), and at lower dimensions it might simply be more favourable to thoroughly exploit a certain area, something that RRO excels at.

7.2 Removal of stop mechanism

The heatmaps (Figure 3), and to a lesser extent the expected best-so-far found graphs (Figure 4) and cumulative distribution graphs (Appendix B), show that RRO without a stopping mechanism performs as well if not better than all other used versions. This shows that the stop mechanism is at best insignificant and at worst detrimental to the performance of RRO. Considering the original RRO lost evaluations when stopping, it is to be expected that in that case the difference would be even more pronounced.

7.3 IRRO and mechanisms

The results in Figure 3 demonstrate that in most cases $RROweak > RROfood >= IRRO$. Whilst RROweak shows a slight advantage over original RRO, the performance is very similar. As it is an elitist strategy, compared to random selection of followers, it is to be expected. IRRO on the whole fares worse than both of its mechanisms separately. This could be explained by the elitist strategy being offset by a much more harsh stopping mechanic, only "loosening up" well after being stuck in local optima.

As the results show that `RavenRoostNoStop` performed on average the best of all the RRO versions, the performance of IRRO might very well be ascribed to making stopping a more central part of the algorithm.

8 Conclusion

The use of the metaphor in the case of RRO is questionable; neither the roosting nor the individual perceptions constitute more than either a single fixed point or random sampling in the area. The intuition of "flight" and stops towards a destination makes sense in the context of the metaphor, but could just as well have been framed in the language of PSO. Furthermore, due to overreliance on metaphor, it is sometimes not clear how the algorithm proceeds exactly, nor what the purpose of certain mechanisms is.

Due to how the roosting mechanism and the movement is structured, it leads us to believe RRO was designed without explicit solutions to the problem of oversampling around the roost location. This in order to test roosting and individual perception along a flight path as useful concepts, and leave room

for potential further experimentation. There is some parameter tuning to test the effectiveness of different components, and the original paper explicitly tests the performance of RRO with and without perceptions. There is however, little analysis as to "why" certain components work. For example, the reason for stopping during flight is never made clear, apart from simply suiting the metaphor. In our experiments we were able to show that this mechanism is irrelevant at best, and detrimental at worst.

For almost all of the functions it can be shown that PSO outperforms all versions of RRO. This is in direct contrast to the claims made by the authors of RRO/IRRO. As we had to take some liberties in the interpretation of RRO, this might still be up for debate. IRRO performed considerably worse, diverging from the results by the authors. It is possible that this is due to how we implemented the food mechanism, but this seems highly unlikely.

Because of the highly similar structure of RRO to PSO (arguably simply being a special case of), benchmarking the application of the individual perception mechanism in PSO might be a worthwhile avenue. Considering the limited success of RRO in the sense of citations, testing whether or not PSO is improved by the use of random sampling might have been a more fruitful endeavour for the authors, firmly placing it within the literature.

References

[1] Aranha, C., Villalón, C., Campelo, F., Dorigo, M., Ruiz, R., Sevaux, M., Sörensen, K., and Stützle, T. Metaphor-based metaheuristics, a call for action: the elephant in the room. *Swarm Intelligence 16* (2021).

[2] Bhargavi, K., Babu, B., and Pitt, J. Performance modeling of load balancing techniques in cloud: Some of the recent competitive swarm artificial intelligence-based. *Journal of Intelligent Systems 30* (2020), 40–58.

[3] Brabazon, A. Foraging inspired algorithms: A design perspective. *2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery* (2017).

[4] Brabazon, A., Cui, W., and O'Neill, M. The raven roosting optimisation algorithm. *Soft Computing 20* (2015).

[5] Bremermann, H. e. a. Optimization through evolution and recombination. *Self-Organizing Systems* (1962), 93–106.

[6] Campelo, F., and Aranha, C. Evolutionary computation bestiary. online: https://fcampelo.github.io/ec-bestiary, 2021.

[7] Campelo, F., and Aranha, C. Sharks, zombies and volleyball: Lessons from the evolutionary computation bestiary. *LIFELIKE Computing Systems Workshop 2021* (2021).

[8] Doerr, C., Wang, H., Ye, F., van Rijn, S., and Bäck, T. Iohprofiler: A benchmarking and profiling tool for iterative optimization heuristics. *arXiv e-prints:1810.05281* (2018).

[9] Dorigo, M. *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, Italy, 1992.

[10] Dragoi, E., and Dafinescu, V. Review of metaheuristics inspired from the animal kingdom. *Mathematics 9* (2021), 2335.

[11] Ezugwu, A., Shukla, A., Nath, R., Akinyelu, A., Agushaka, O., Chiroma, H., and Muhuri, P. Metaheuristics: a comprehensive overview and classification along with bibliometric analysis. *Artificial Intelligence Review 54* (2021), 1–79.

[12] Geem, Z., and Kim, J. e. a. A new heuristic optimization algorithm: harmony search. *Simulation 76* (2001).

[13] Hansen, N., Finck, S., Ros, R., and Auger, A. Real-parameter black-box optimization benchmarking 2009: Noisy functions definitions. *[Research Report] RR-6829, INRIA* (2009).

[14] Hansen, N., and Ostermeier, A. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation 9* (2001), 159–195.

[15] Hussan, K., Salleh, M., Cheng, S., and Shi, Y. Metaheuristic research: a comprehensive survey. *Artificial Intelligence Review 52* (2019), 2191–2233.

[16] Kennedy, J., and Eberhart, R. Particle swarm optimization. *Proceedings of ICNN'95-international conference on neural networks 4* (1995), 1942–1948.

[17] Kirkpatrick, S., Gelatt, C., and Vecchi, M. Optimization by simulated annealing. *Science 220*, 4598 (1983), 671–680.

[18] Meng, X., Liu, Y., Gao, X., and Zhang, H. A new bio-inspired algorithm: Chicken swarm optimization. *Advances in Swarm Intelligence* (2014), 86–94.

[19] Mirjalili, S., Mirjalili, S., and Lewis, A. Grey wolf optimizer. *Advances in Engineering Software 69* (2014), 46–61.

[20] Moldovan, D. Horse optimization algorithm: A novel bio-inspired algorithm for solving global optimization problems. *Computer Science On-line Conference* (2020).

[21] Nabi, S., Ahmad, M., Ibrahim, M., and Hamam, H. Adpso: Adaptive pso-based task scheduling approach for cloud computing. *Sensors 22* (2022), 920.

[22] Rajwar, K., Deep, K., and Das, S. An exhaustive review of the metaheuristic algorithms for search and optimization: taxonomy, applications, and open challenges. *Artificial Intelligence Review* (2023).

[23] Rechenberg, I. Cybernetic solution path of an experimental problem. *Royal Aircraft Establishment 1122* (1965).

[24] Reynolds, C. Flocks, herds, and schools: A distributed behavioral model. *ACM SIGGRAPH Computer Graphics 21* (1987), 25–34.

[25] Schwefel, H. Numerical optimization of computer models. *Journal of The Operational Research Society - J OPER RES SOC 33* (1981).

[26] Shah-Hosseini, H. Problem solving by intelligent water drops. *2007 IEEE Congress on Evolutionary Computation* (2007), 3226–3231.

[27] Sörensen, K. Metaheuristics – the metaphor exposed. *International Transactions of Operations Research In Press* (2013).

[28] Torabi, S., and Safi-Esfahani, F. Improved raven roosting optimization algorithm. *Swarm and Evolutionary Computation* (2017).

[29] Torabi, S., and Safi-Esfahani, F. A dynamic task scheduling framework based on chicken swarm and improved raven roosting optimization methods in cloud computing. *The Journal of Supercomputing 74* (2018).

[30] Torabi, S., and Safi-Esfahani, F. A hybrid algorithm based on chicken swarm and improved raven roosting optimization. *Soft Computing 23* (2019).

[31] Villalón, C., Dorigo, M., and Stützle, T. The intelligent water drops algorithm: why it cannot be considered a novel algorithm: A brief discussion on the use of metaphors in optimization. *Swarm Intelligence 13* (2019).

[32] Villalón, C., Dorigo, M., and Stützle, T. An analysis of why cuckoo search does not bring any novel ideas to optimization. *Computers Operations Research 142* (2022), 105747.

[33] Villalón, C., Stützle, T., and Dorigo, M. Grey wolf, firefly and bat algorithms: Three widespread algorithms that do not contain any novelty. *Swarm Intelligence* (2020), 121–133.

[34] Wang, H., Vermetten, D., Ye, F., Doerr, C., and Bäck, T. Iohanalyzer: Detailed performance analyses for iterative optimization heuristics. *ACM Transactions on Evolutionary Learning and Optimization 2*, 1 (2022).

[35] Ward, P., and Zahavi, A. The importance of certain assemblages of birds as "information-centres" for food-finding. *Ibis 115* (1973), 517 – 534.

[36] Weyland, D. A rigorous analysis of the harmony search algorithm - how the research community can be misled by a "novel" methodology. *International Journal of Applied Metaheuristic Computing 1-2* (2010), 50–60.

[37] Yang, X. Firefly algorithms for multimodal optimization. *Proceedings of the 5th international conference on Stochastic algorithms: foundations and applications 5792* (2010).

[38] Yang, X. *A New Metaheuristic Bat-Inspired Algorithm.* Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 65–74.

[39] Yang, X., and Deb, S. Cuckoo search via levey flights. *2009 World Congress on Nature and Biologically Inspired Computing, NABIC 2009 - Proceedings* (2010), 210 – 214.

A Expected Target Value over Runtime

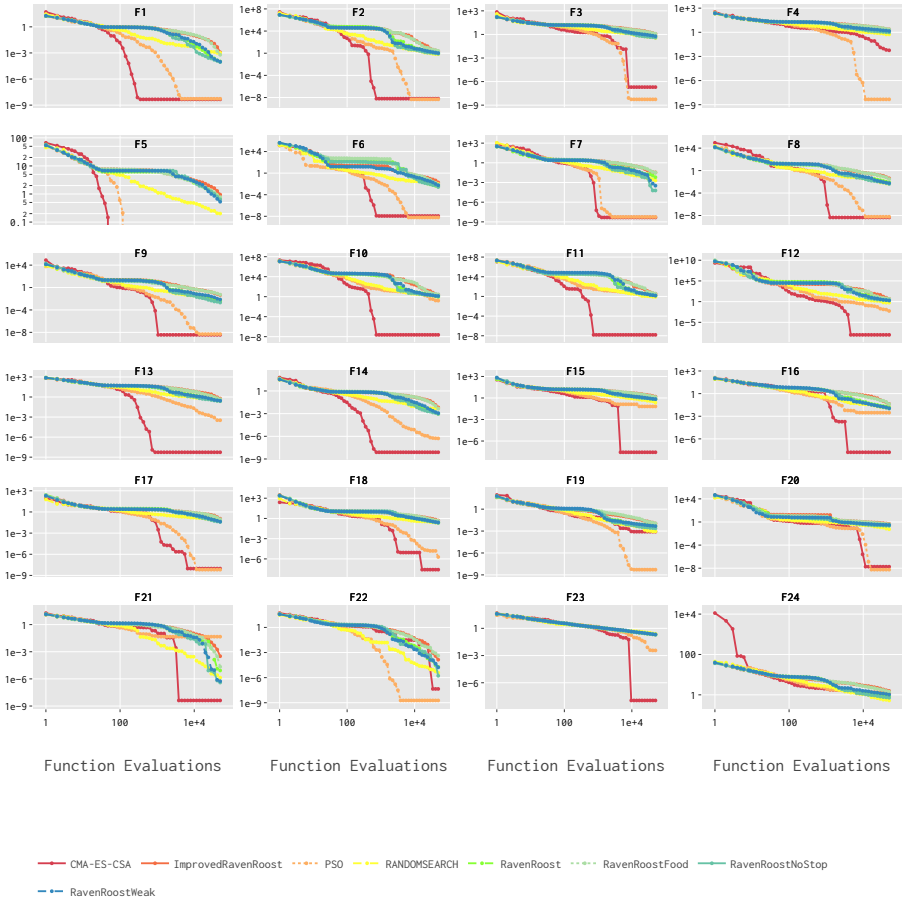

Figure 5: All algorithms, D=2

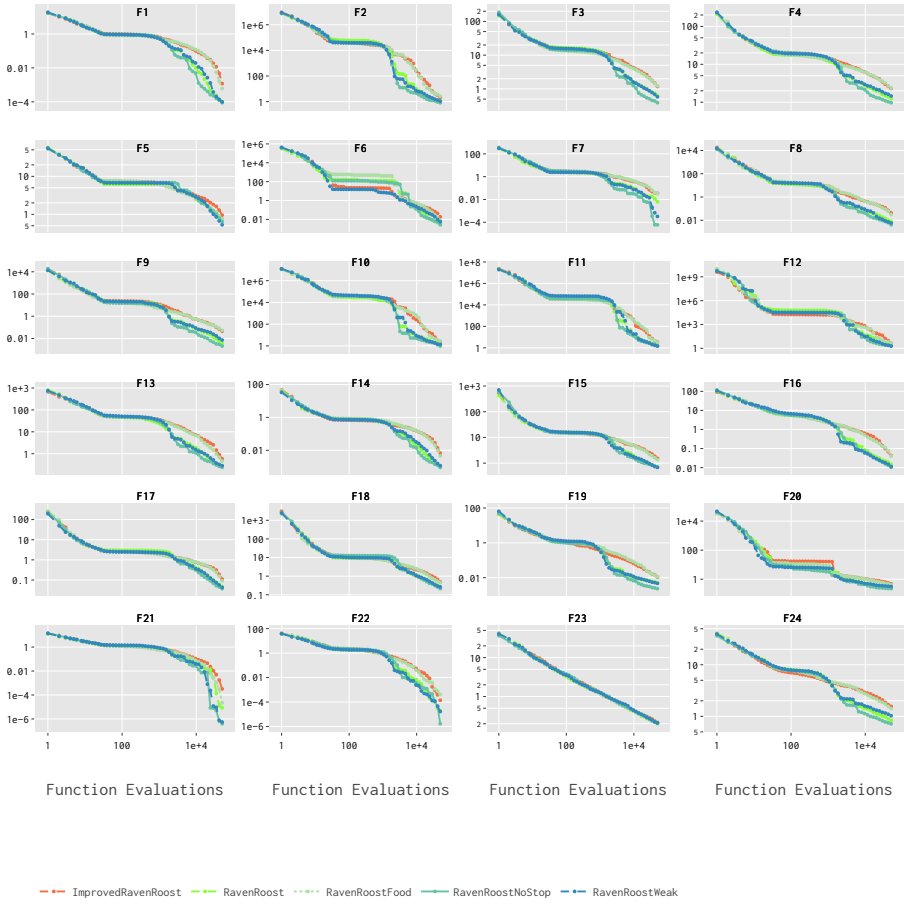

Figure 6: RRO algorithms, D=2

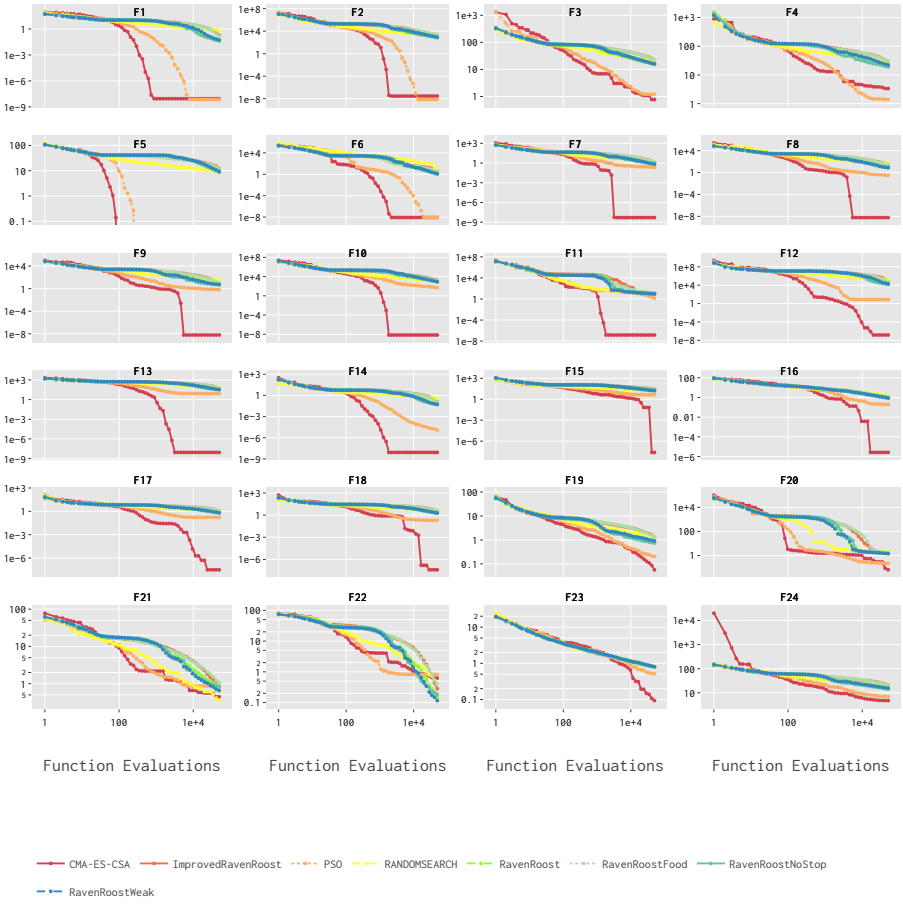

Figure 7: All algorithms, D=5

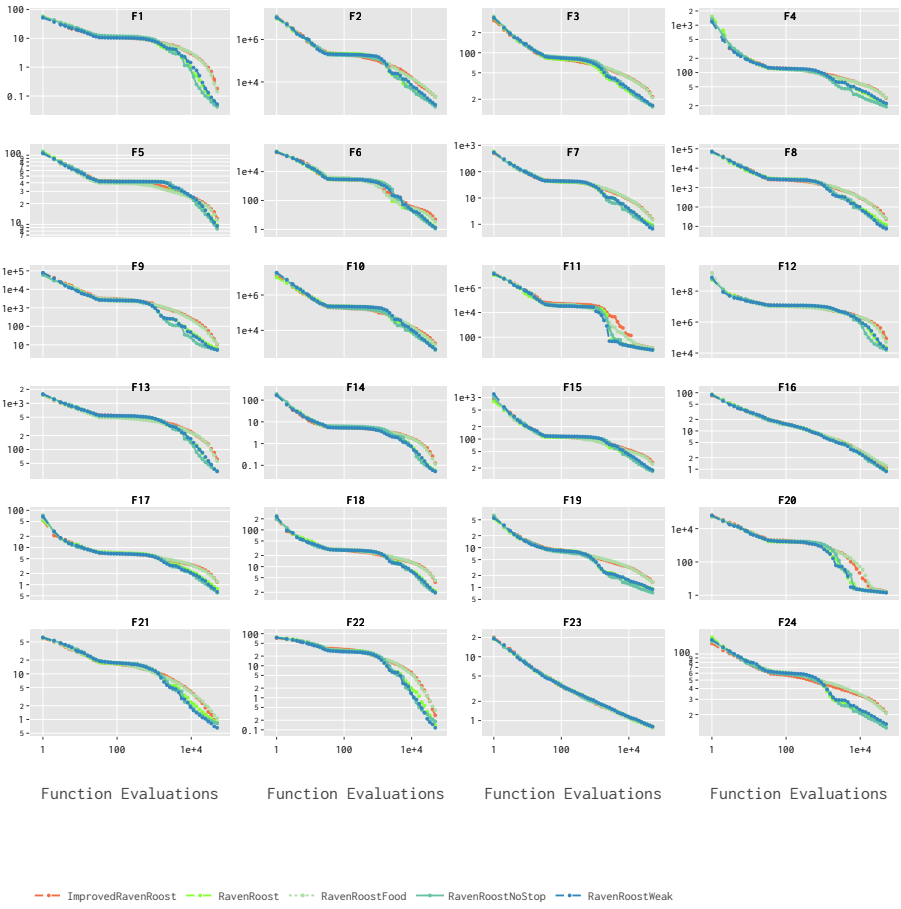

Figure 8: RRO algorithms, D=5

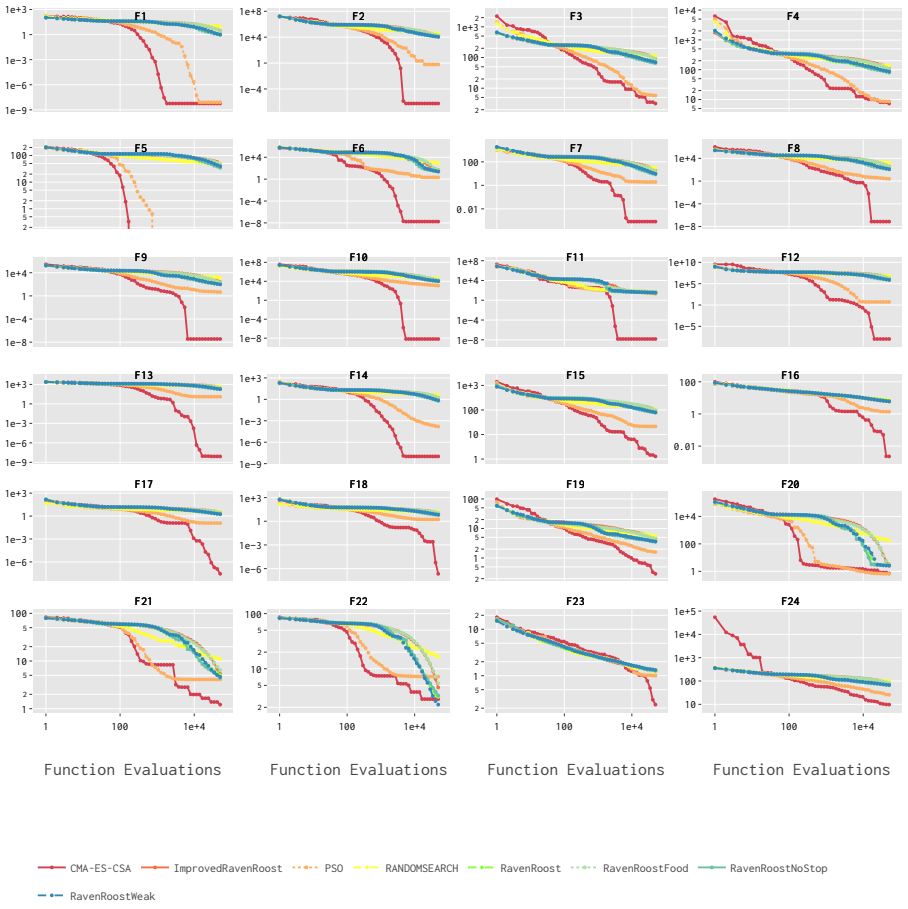

Figure 9: All algorithms, D=10

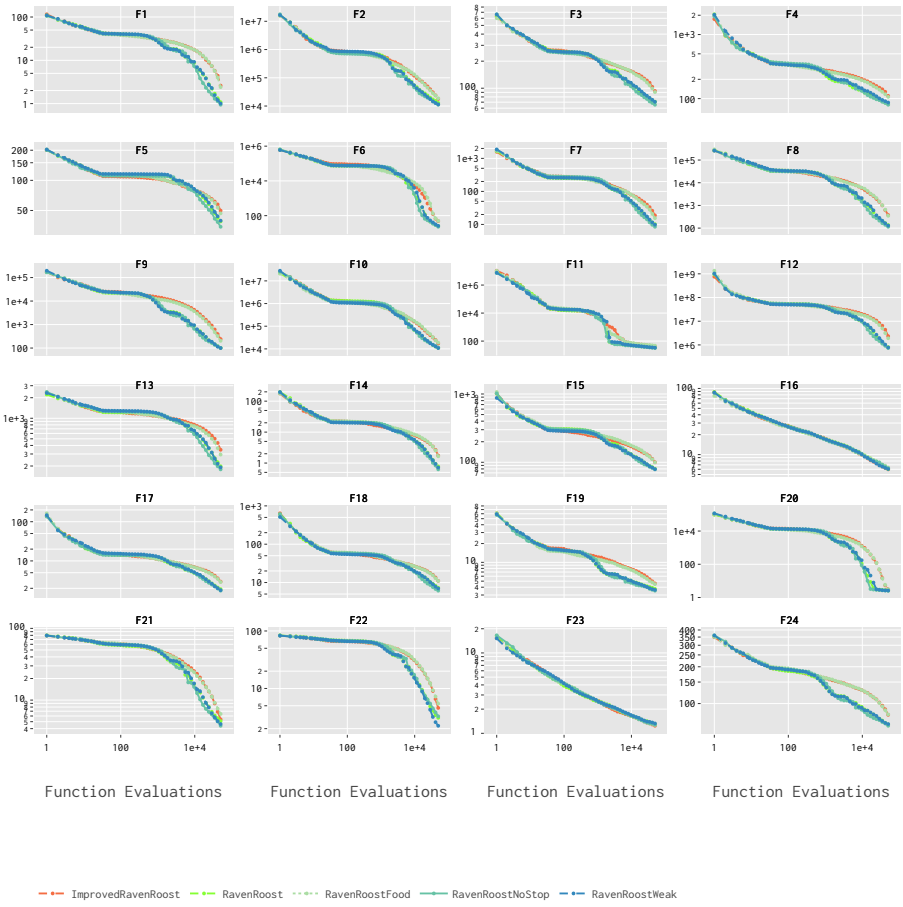

Figure 10: RRO algorithms, D=10

B Probability Density Functions

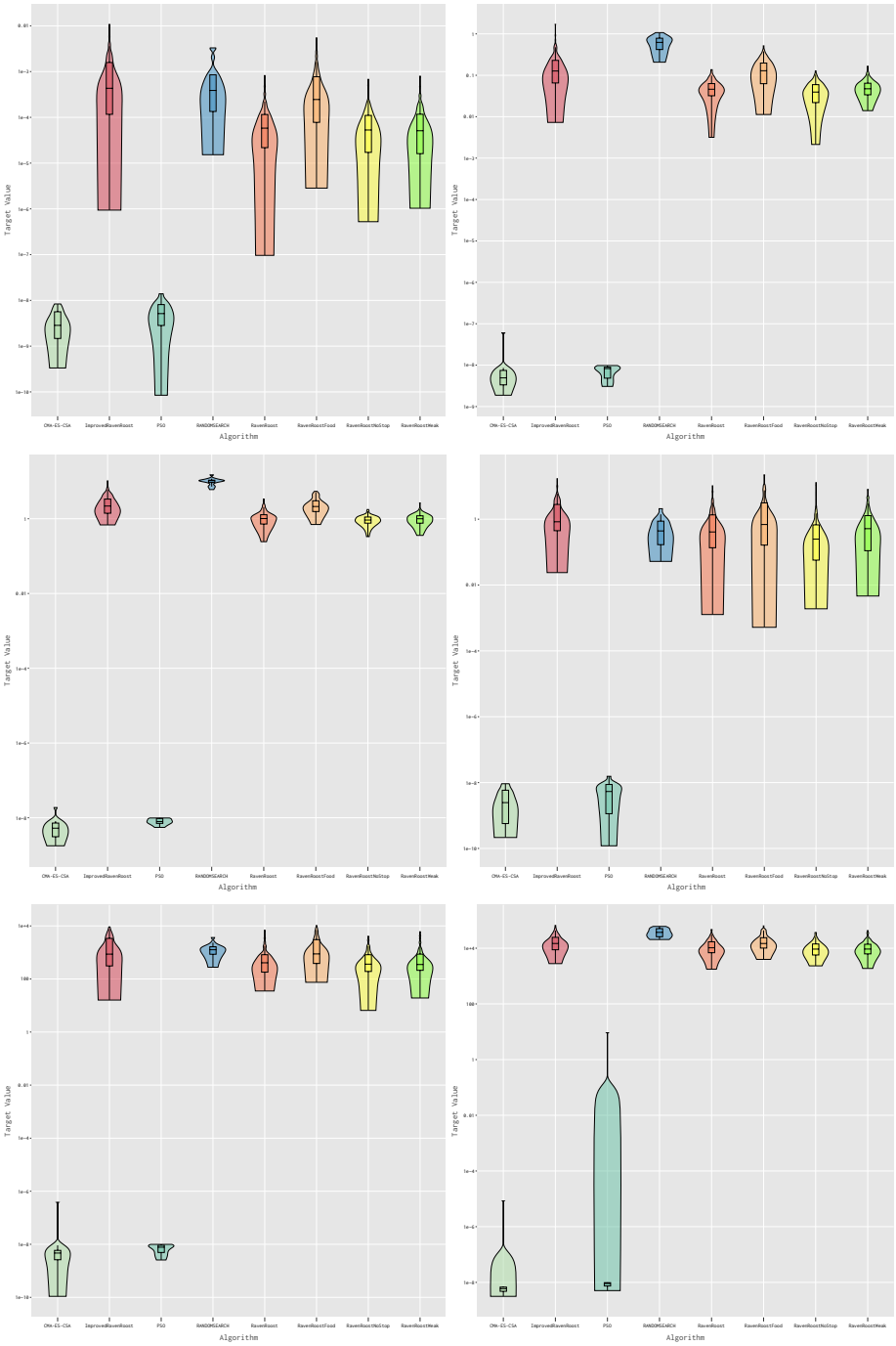

Figure 11: Left-to-right, top-to-bottom: Sphere D=[2,5,10], Ellipsoid D=[2,5,10]

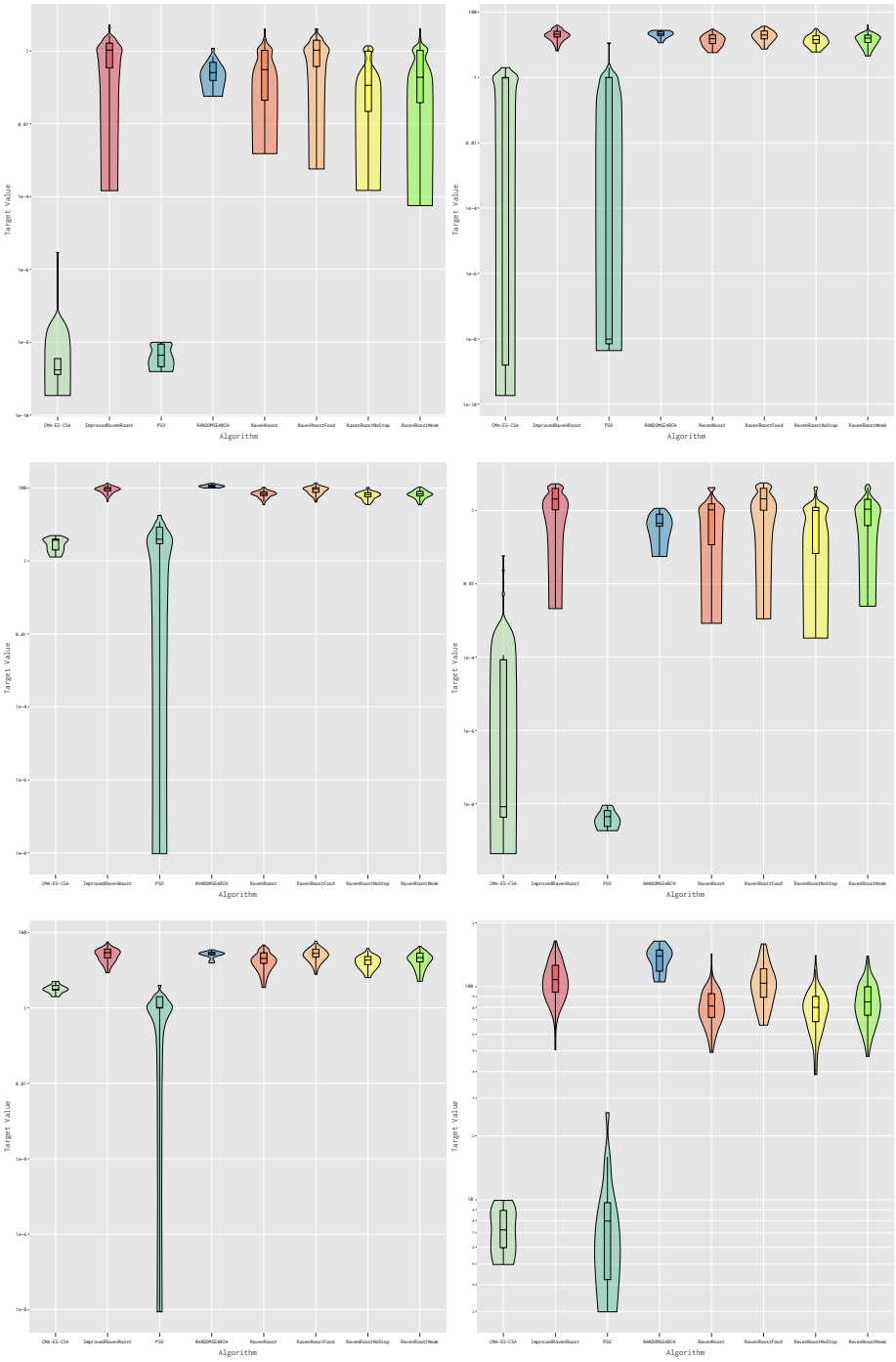

Figure 12: Left-to-right, top-to-bottom: Rastrigin D=[2,5,10], BuecheRastrigin D=[2,5,10]

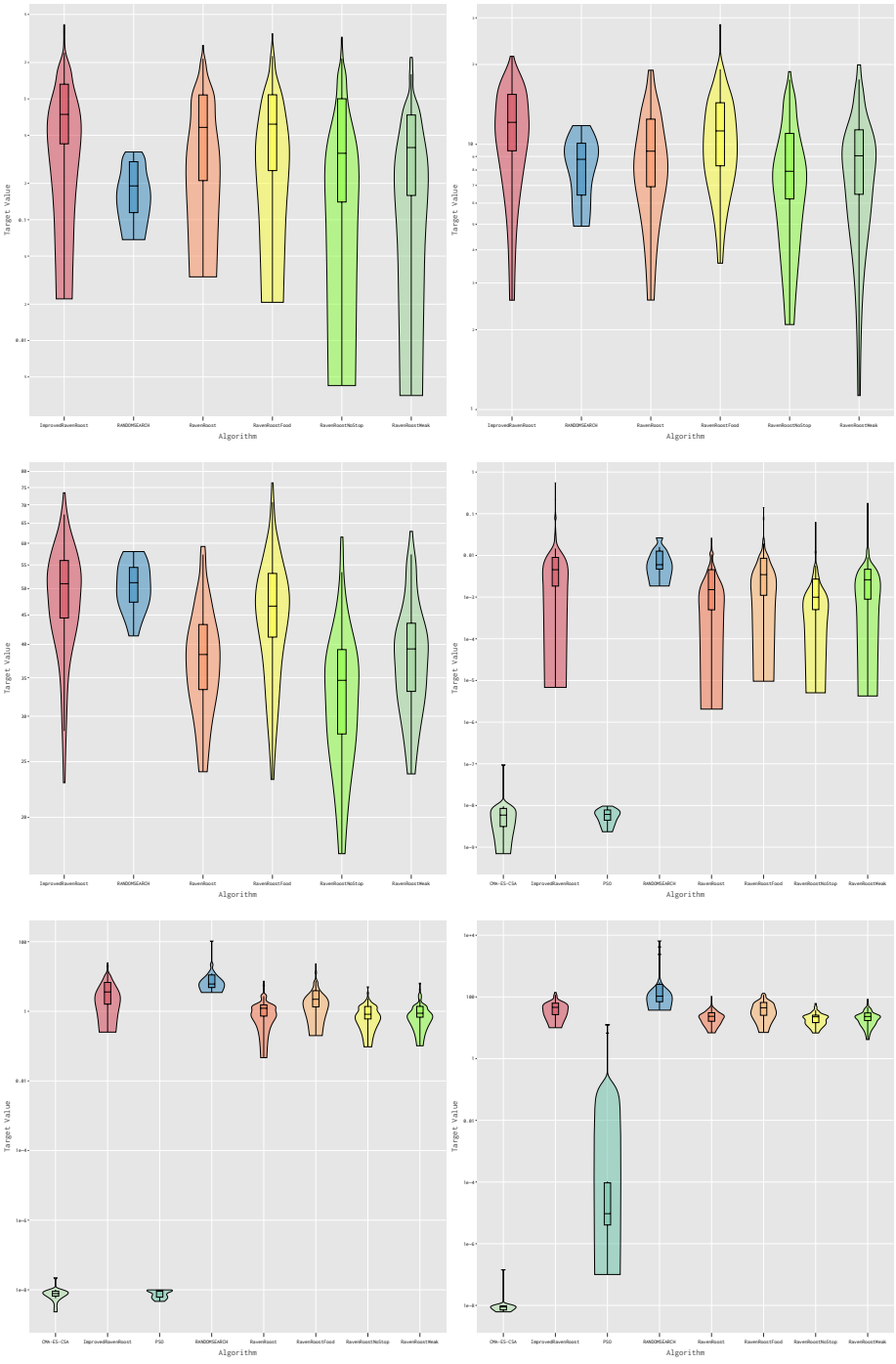

Figure 13: Left-to-right, top-to-bottom: LinearSlope D=[2,5,10], AttractiveSector D=[2,5,10]

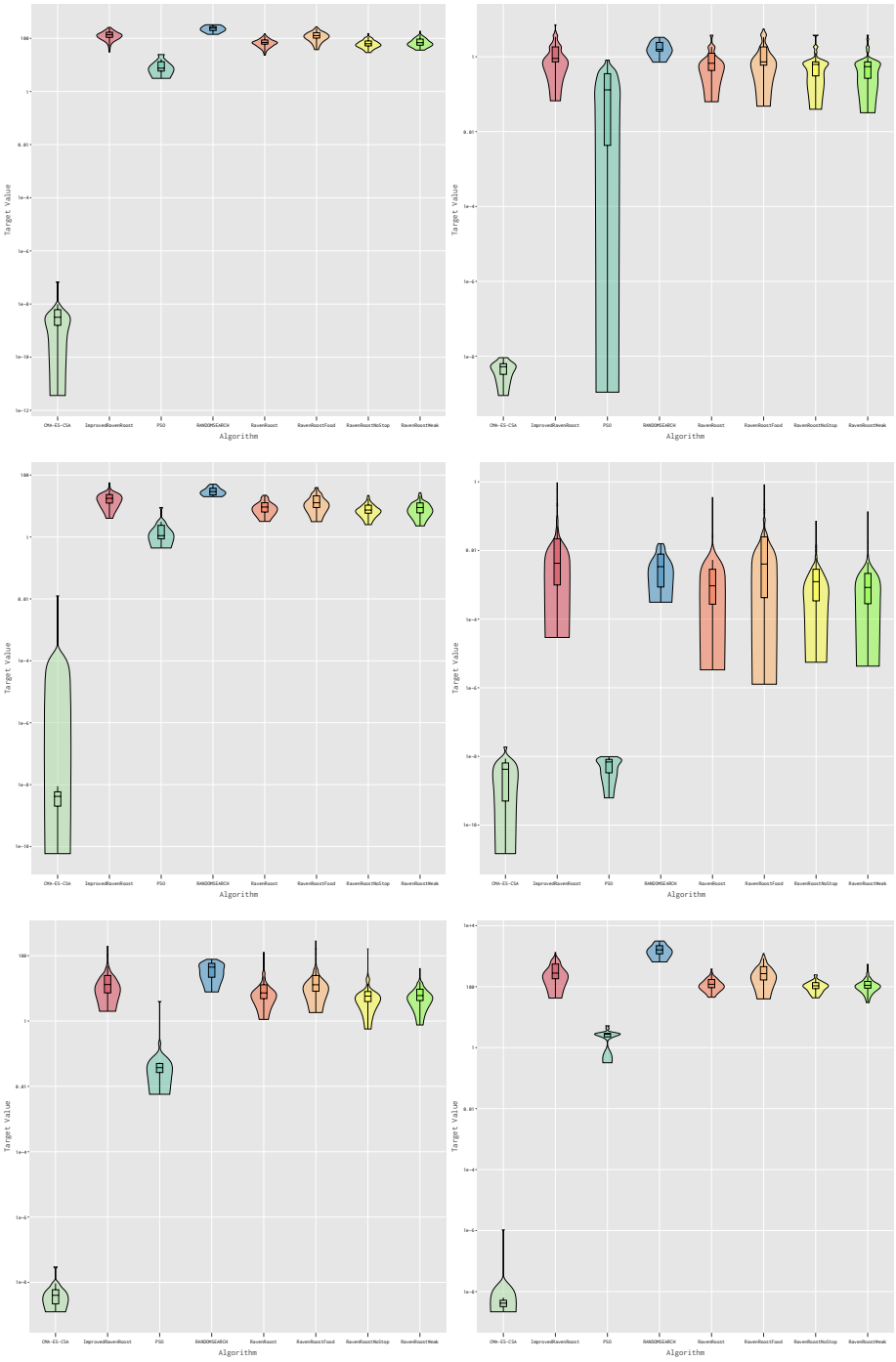

Figure 14: Left-to-right, top-to-bottom: StepEllipsoid D=[20,5,10], Rosenbrock D=[2,5,10]

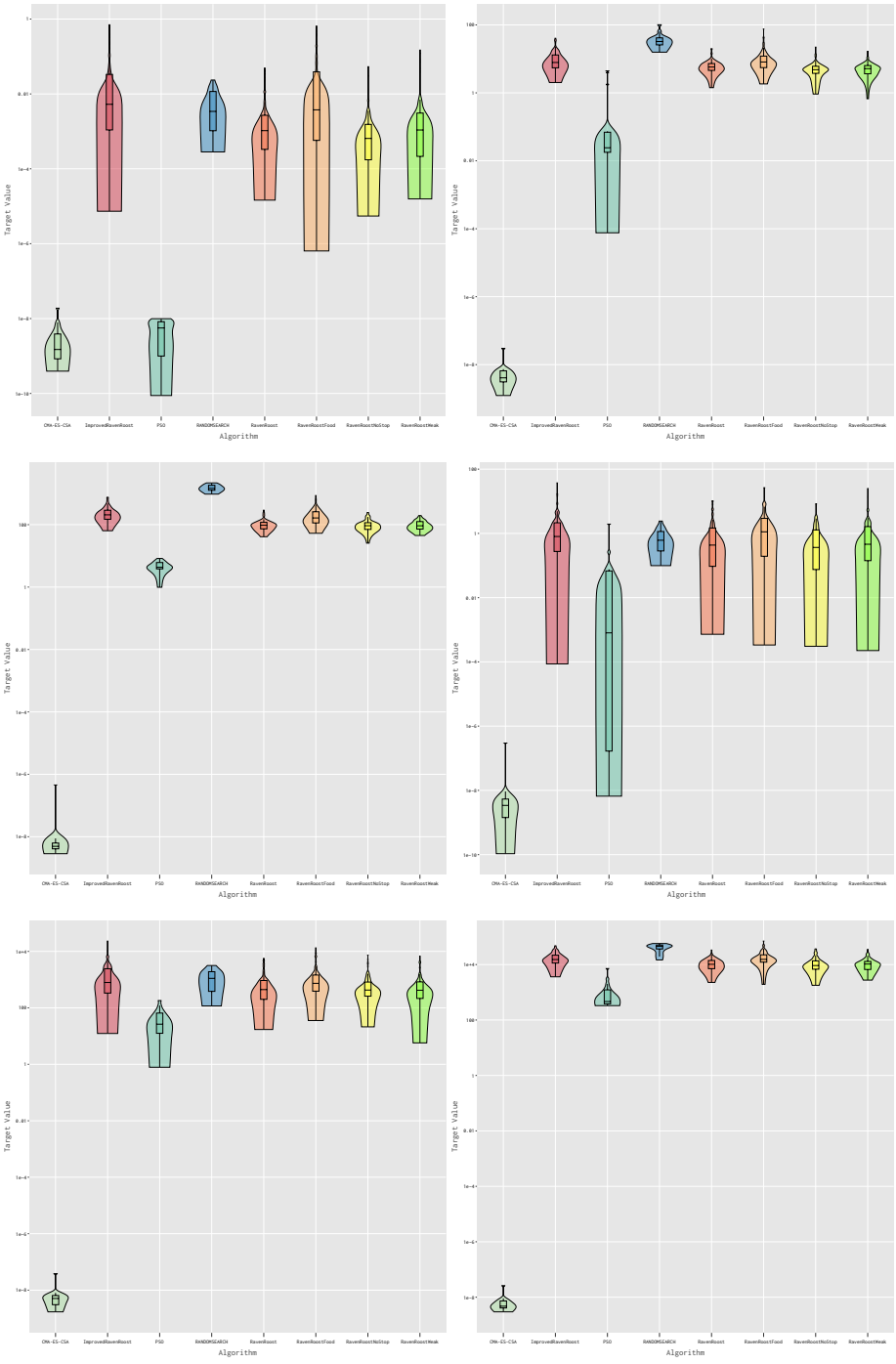

Figure 15: Left-to-right, top-to-bottom: RosenbrockRotated D=[2,5,10], EllipsoidRotated D=[2,5,10]

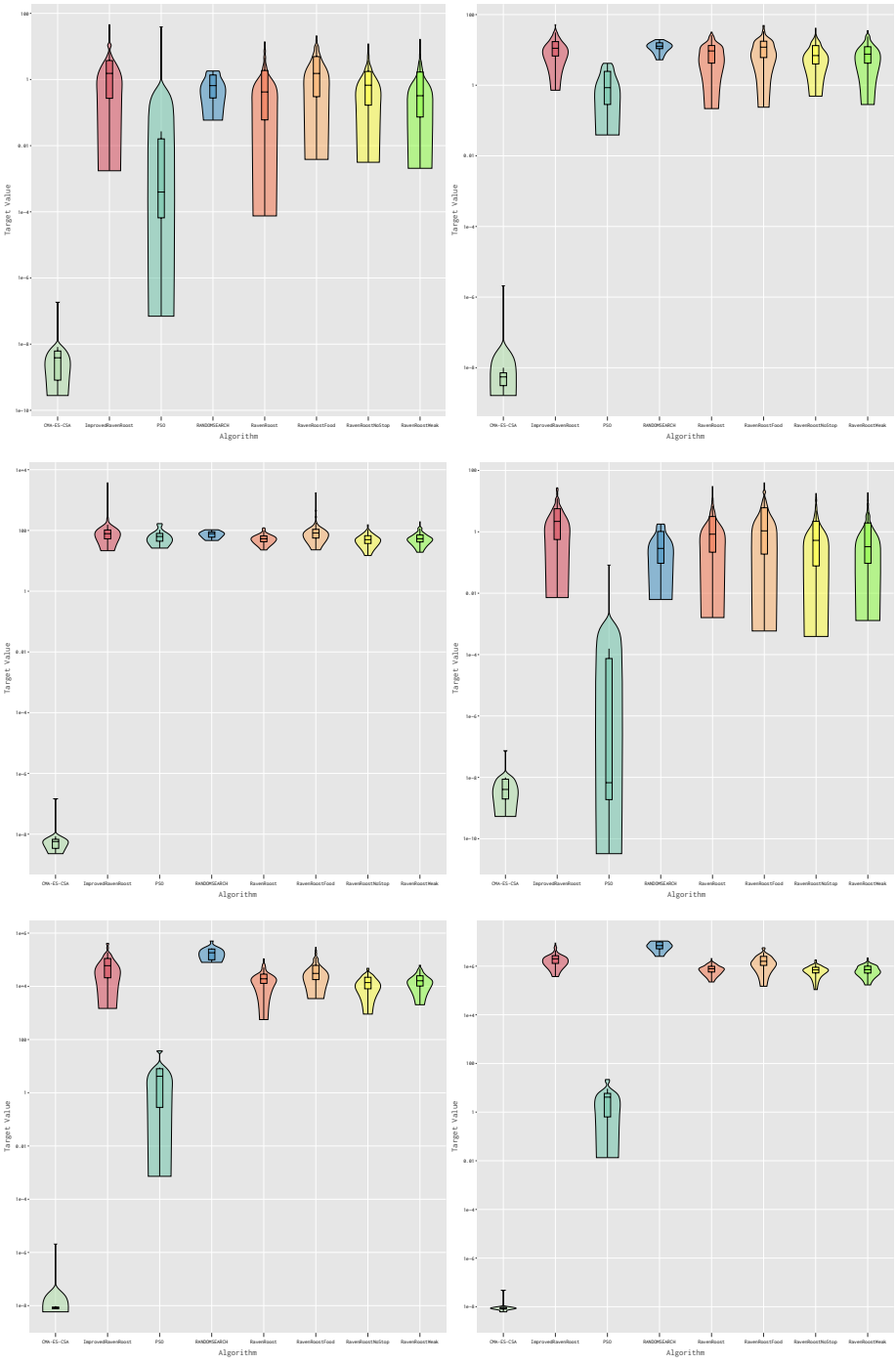

Figure 16: Left-to-right, top-to-bottom: Discus D=[2,5,10], BentCigar D=[2,5,10]

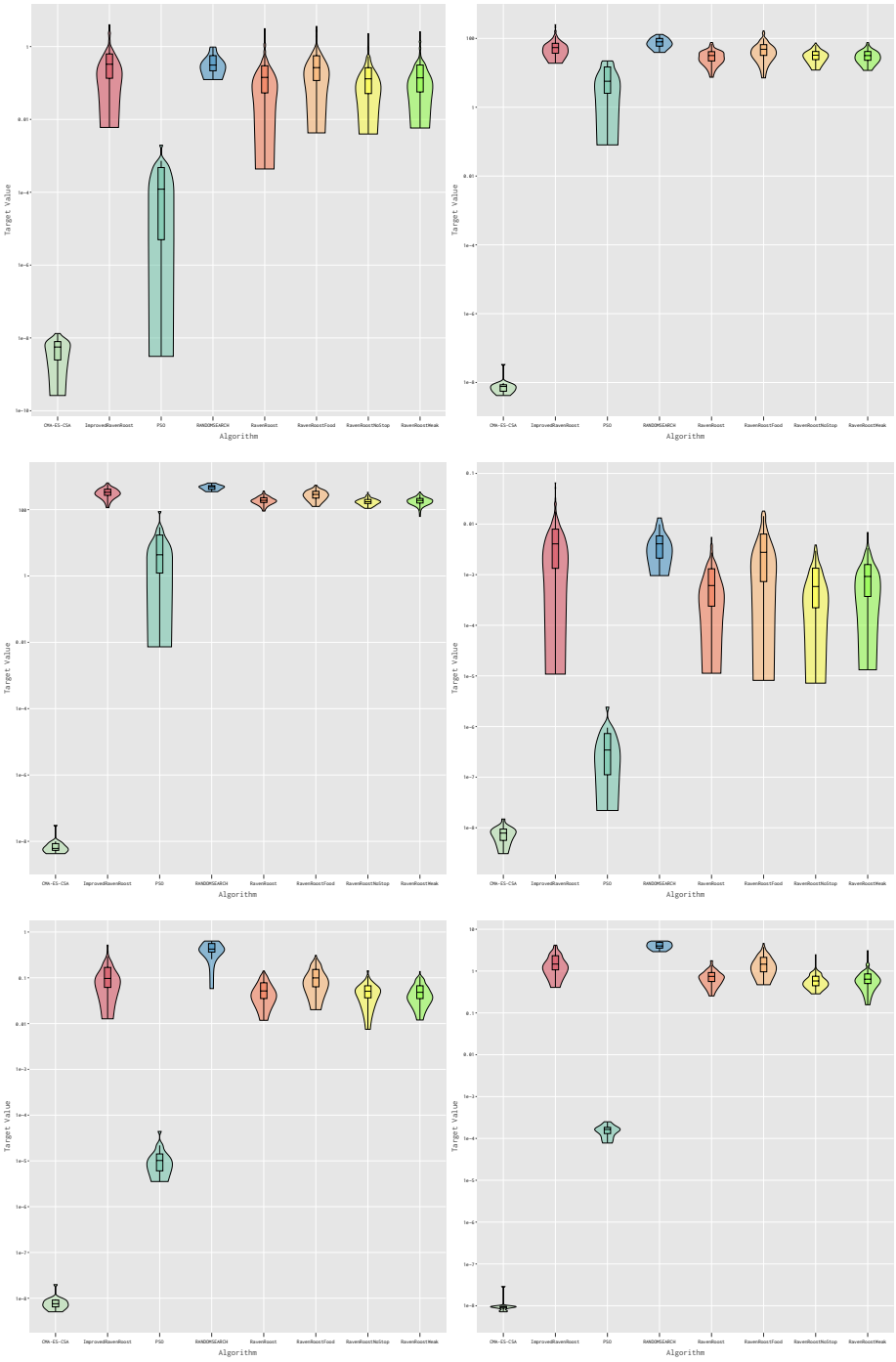

Figure 17: Left-to-right, top-to-bottom: SharpRidge D=[2,5,10], DifferentPowers D=[2,5,10]

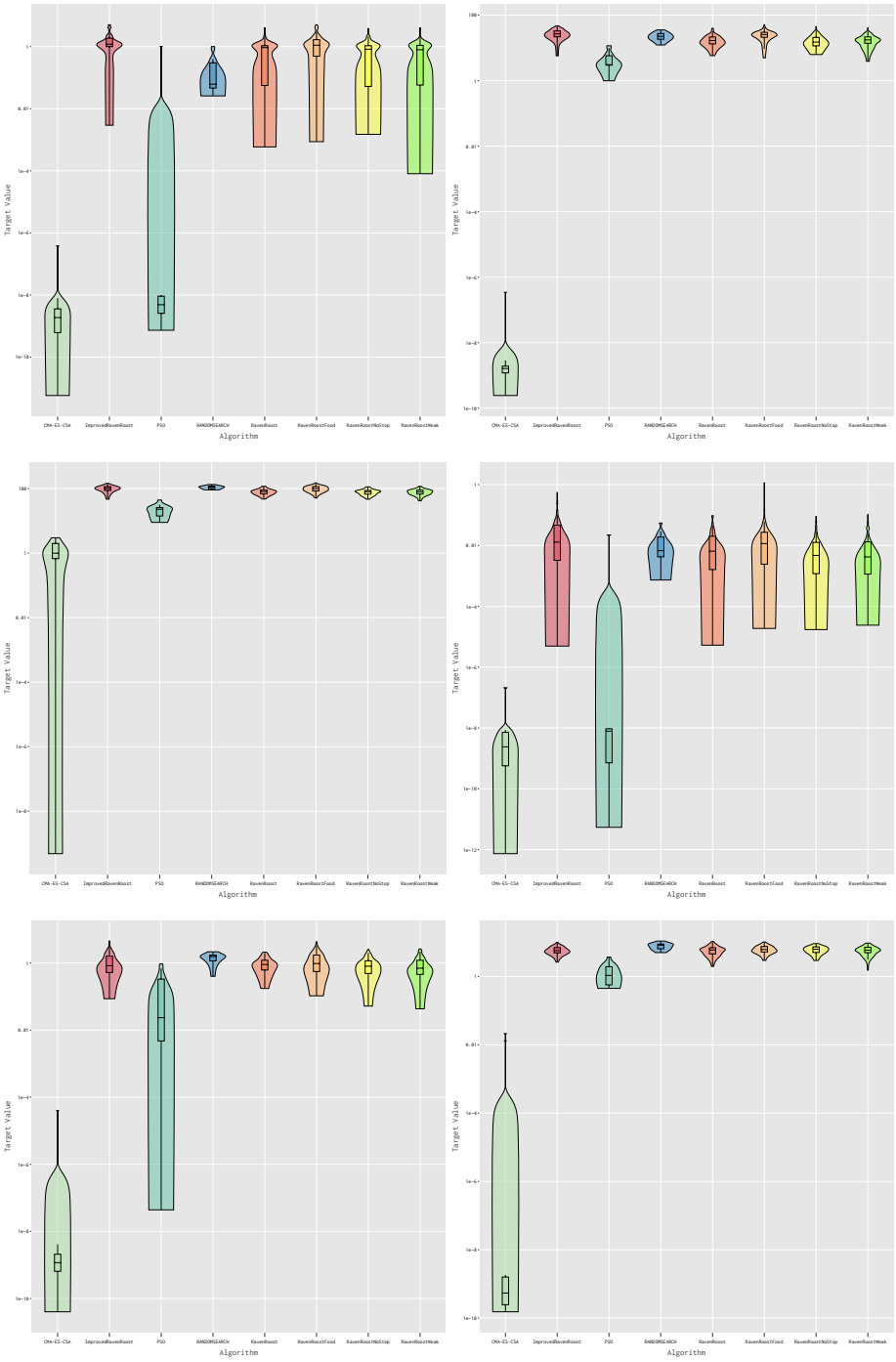

Figure 18: Left-to-right, top-to-bottom: RastriginRotated D=[2,5,10], Weierstrass D=[2,5,10]

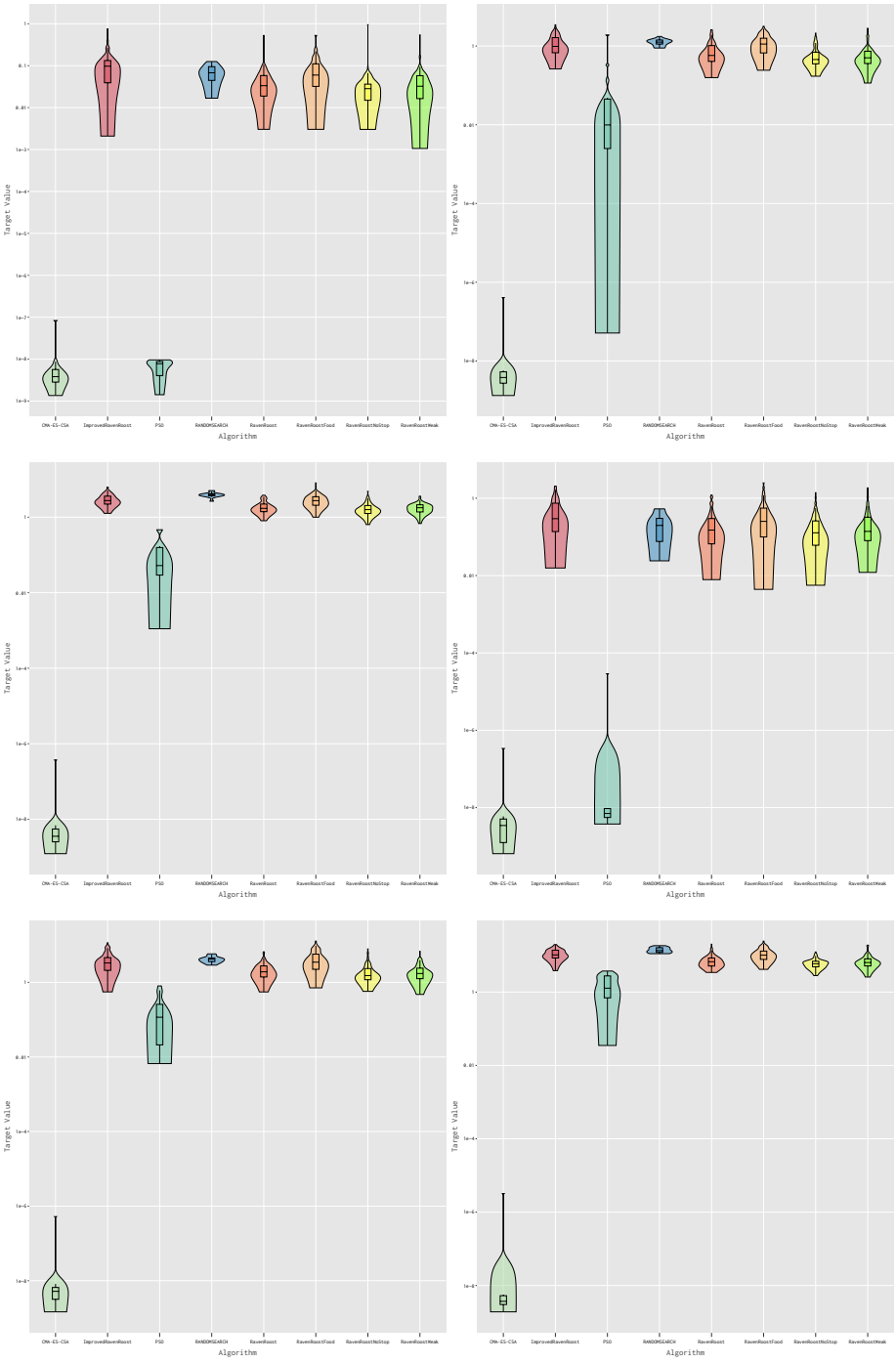

Figure 19: Left-to-right, top-to-bottom: Schaffers10 D=[2,5,10], Schaffers1000 D=[2,5,10]

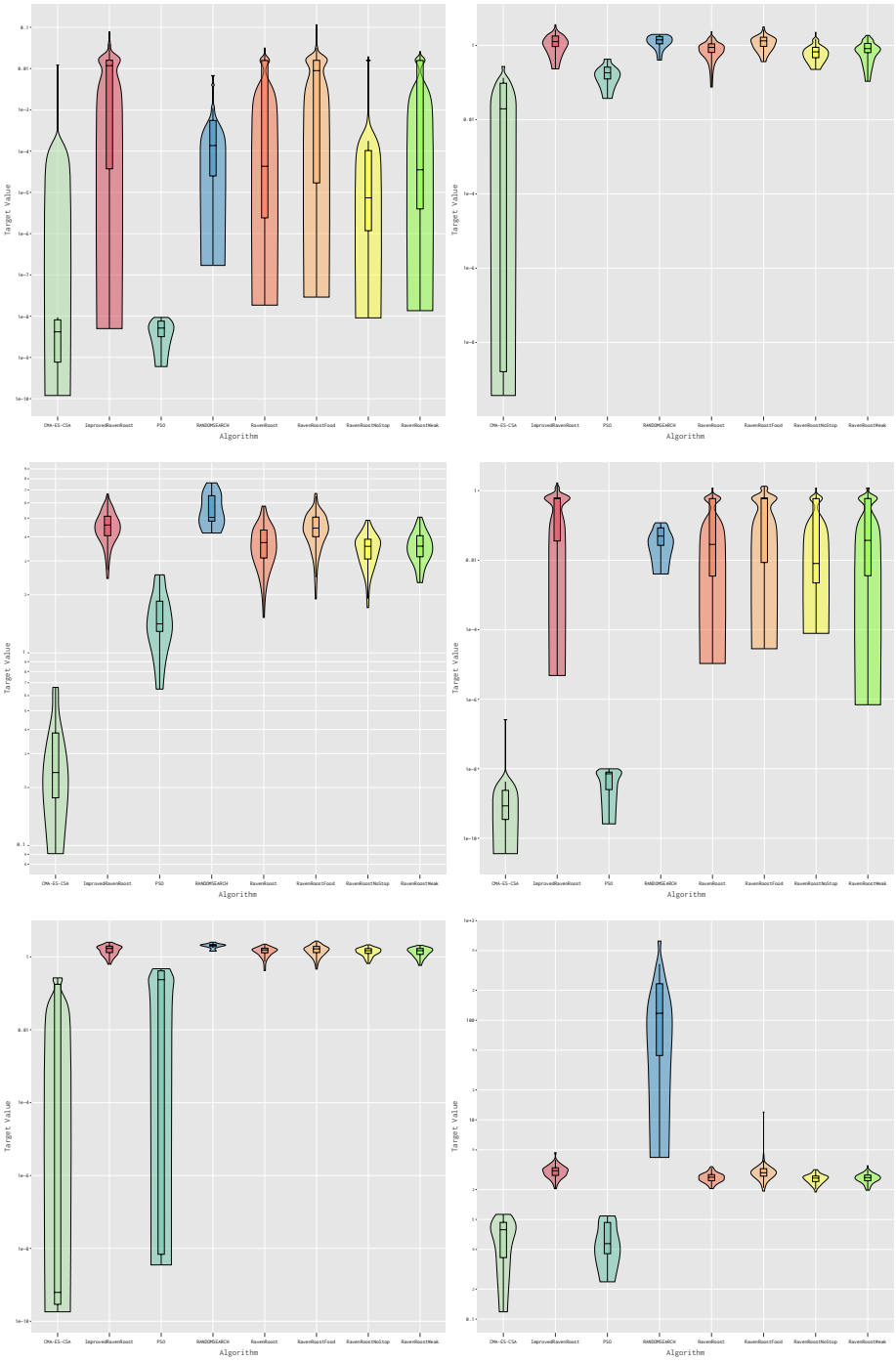

Figure 20: Left-to-right, top-to-bottom: GriewankRosenbrock D=[2,5,10], Schwefel D=[2,5,10]

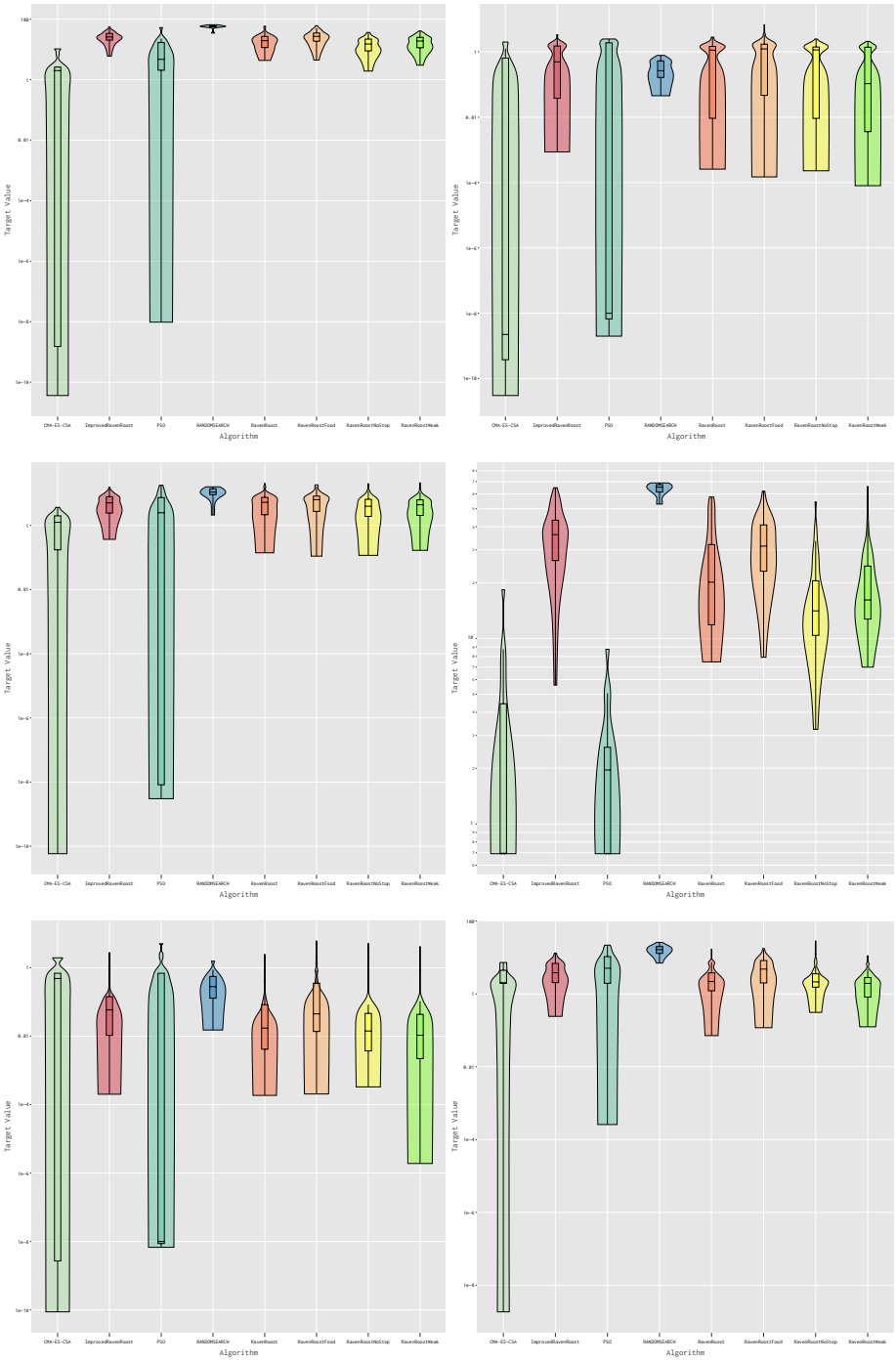

Figure 21: Left-to-right, top-to-bottom: Gallagher101 D=[20,5,10], Gallagher21 D=[20,5,10]

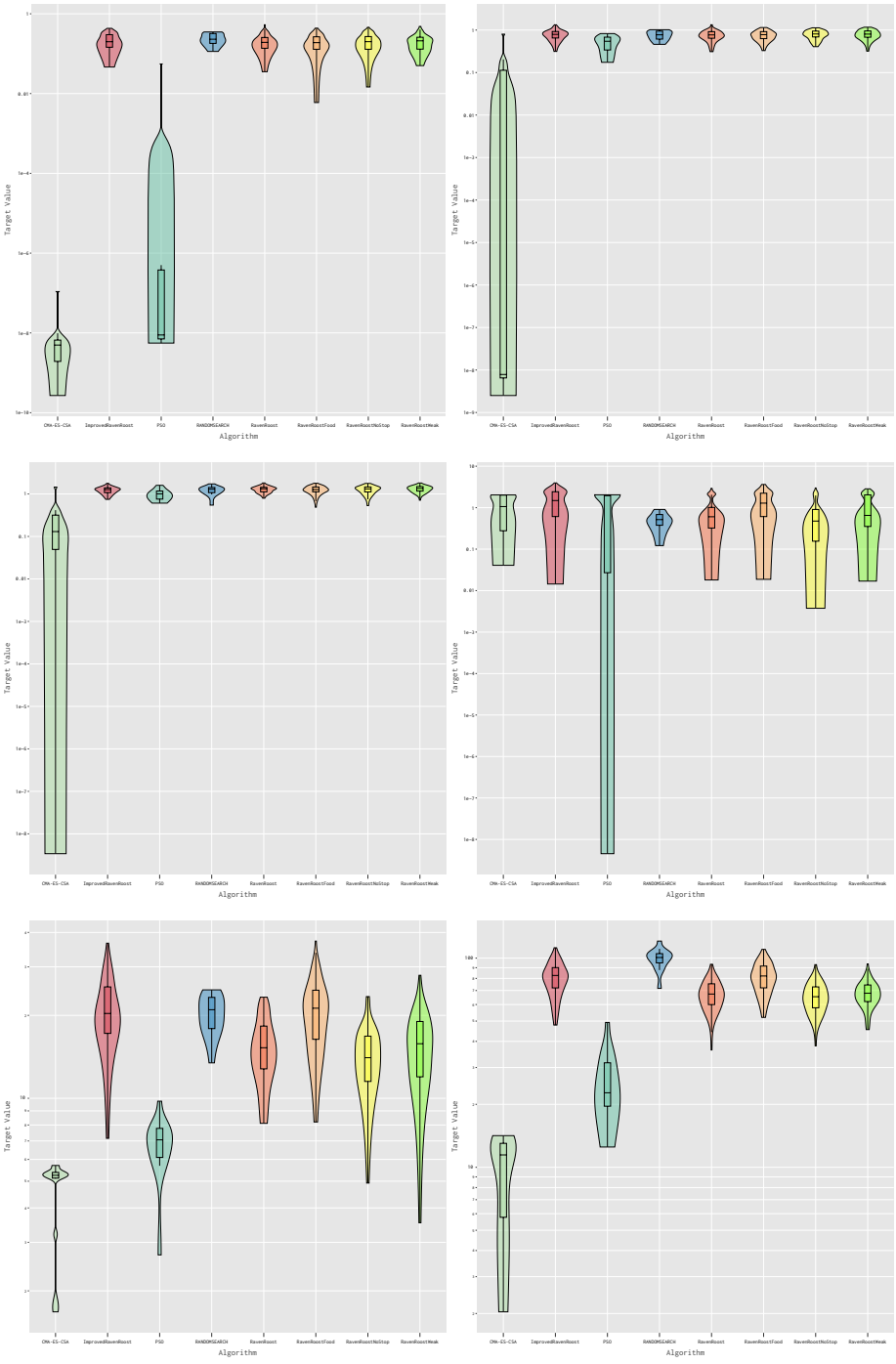

Figure 22: Left-to-right, top-to-bottom: Katsuura D=[2,5,10], LunacekBiRastrigin D=[2,5,10]