An Exploration of Metaheuristic Swarm Intelligence Algorithms

In recent decades, metaheuristic algorithms have emerged as powerful optimization techniques, sparking a surge of active interest, especially in the research community. Although perhaps not as popular as their deterministic and gradient-based counterparts, these algorithms offer several distinct advantages. While they do not guarantee optimality or convergence, and are quite sensitive to their random initializations and parameter settings, metaheuristic algorithms excel at generating near-optimal solutions within a reasonable timeframe, even in non-convex cases. What makes them especially captivating, though, is the fact that many are inspired by the collective behavior of natural systems. Metaheuristic swarm intelligence algorithms, in particular, draw inspiration from various biological and social systems, such as bird flocks, ant colonies, and fish schools, to solve complex optimization problems in an efficient manner. In this sense, there appears to be a potential link between such algorithms and the concept of self-organization. In fact, many swarm intelligence algorithms appear to exhibit self-organizing behavior by involving agents who independently follow simple rules, yet collectively produce sophisticated solutions to the optimization problem. A prime example is particle swarm optimization (PSO), a seemingly simple algorithm that produces surprising results through self-organizational mechanisms. The following work delves further into the PSO algorithm, describing its inner workings while juxtaposing its performance against other swarm intelligence algorithms on the task of global optimization. The goal of this exploration is not only to further explore the fascinating landscape of swarm intelligence algorithms but also better appreciate their relationship to the overarching concept of self-organization.

The original PSO algorithm is directly inspired by the concept of flocks of birds or schools of fish searching for food (Kennedy & Eberhart 1995). More generally, PSO simulates

the social behavior of individuals (i.e. particles) in a swarm searching for the optimal solution in a multi-dimensional search space. It achieves this by first randomly initializing a population of particles with the search space such that each particle $P_i^t = [x_{0,i}^t, x_{1,i}^t, \ldots, x_{n,i}^t]$ represents a potential solution to the optimization problem. Each particle is also randomly assigned a velocity $V_i^t = [v_{0,i}^t, v_{1,i}^t, \ldots, v_{n,i}^t]$, which facilitates its exploration of the search space. As the particles move throughout the search space, each of them maintains its personal best position *pbest* based on its best achieved fitness value thus far. This fitness value represents the quality of the particle as a solution, which is evaluated based on the objective function of the optimization problem. The algorithm also keeps track of the position with the best fitness value out of all the particles in the swarm as the global best position *gbest*. If at any time a particle finds a solution better than the current *gbest*, it is updated to reflect the new position. Furthermore, in terms of the actual movement of the particles, the velocities, much like the positions of the particles themselves, are dynamic in nature over every iteration of the algorithm. In fact, they follow the update rule $V_i^{t+1} = wV_i^t + c_1r_1(P_{pbest}^t - P_i^t) + c_2r_2(P_{gbest}^t - P_i^t)$ such that the new velocity for each particle is not only dependent on its previous velocity but its *pbest* and the overall *gbest*. The first term wV_i^t is referred to as the inertia where w is a hyperparameter that essentially governs the tradeoff between the exploration of the surrounding search space and the exploitation of the existing best solutions. Lower w values correspond to greater exploitation. The following term $c_1r_1(P_{pbest}^t - P_i^t)$ is the cognitive (i.e. personal or instinctual) component where r_1 is a small random weight for the cognitive acceleration coefficient $c1$, which determines the level of influence each particle's *pbest* has on its search strategy. On the other hand, the last term $c_2r_2(P_{gbest}^t - P_i^t)$ is the social (i.e. global or imitation) component where r_2 is a small random weight for the social acceleration coefficient c_2, which determines the level of influence *gbest* has on each particle. Just as with w,

balancing the two hyperparameters is key to avoid overly exploring or exploiting the search space. More importantly, selecting the appropriate values, or range of values, for these critical hyperparameters ensures that the algorithm operates nominally in terms of its convergence rate and solution quality. Although advanced methods exist to dynamically adjust the optimal hyperparameter values for PSO, for simplicity, the following experimentation leverages the best static values of $w = 0.76$ and $c_1 = c_2 = 2.05$, as determined by the literature (Clerc & Kennedy 2002). The number of particles is another hyperparameter that influences the performance of the algorithm, but it is not tuned for the scope of this work, as explained below. Ultimately, once the velocities have been updated, the particle positions are then finally updated via the rule $P_i^{t+1} = P_i^t + V_i^{t+1}$. This process continues until a predetermined number of iterations have been completed or a satisfactory solution has been found. See Figure 1 for an example visualization of this process. Taking a step back to examine PSO from the lens of self-organization, it is increasingly clear that the self-organizational relationship primarily stems from the collective behavior of the particles within the swarm. Each particle's motion is based on not only its own experiences but the experiences of the surrounding swarm as well, fostering local interactions that drive the self-organization of the swarm towards more promising regions of the search space. This collaborative exploration and exploitation, in turn, enables PSO to efficiently converge to near-optimal solutions without requiring any explicit external coordination, thus demonstrating some of the key principles of self-organization.

The other swarm intelligence algorithms selected alongside PSO are the genetic algorithm (GA), firefly algorithm (FA), artificial bee colony optimization (ABC), and cuckoo search optimization (CSO). For brevity, they are not explored in as much detail as PSO but brief details regarding their inner workings are provided to showcase their similarities to PSO and

their link to self-organization. For instance, GA works by maintaining a set of candidate solutions (i.e. individuals) from which fitter solutions (i.e. parents) are selected to crossover and exchange information to produce new solutions (i.e. offspring). Mutations are occasionally randomly introduced to the offspring, but as newer generations are created, the population tends to evolve so that fitter solutions gradually start to dominate (Holland 1992). More similarly to PSO, FA models the flashing behavior of fireflies to attract mates or prey with brighter lights indicating better solutions. Fireflies gravitate towards brighter lights but attraction strength decreases with distance between fireflies (Yang 2009). In a similar sense, ABC simulates the behavior of honeybee colonies with foraging bees communicating their findings to the rest of the hive such that stronger solutions receive more attention from awaiting bees. Upon excessive stagnation in any given solution, exploring bees search for alternate solutions (Karaboga & Basturk 2007). Lastly, CSO is inspired by the brood parasitism behavior of some cuckoo species which lay their eggs in other birds' nests. Cuckoos lay eggs probabilistically in nests and may abandon their existing nest in favor of better alternatives over time (Gandomi et al. 2013). FA, ABC, and CSO possess many of the same characteristics as PSO and, thus, closely adhere to its core self-organizational principle in terms of collective exploratory efforts and information sharing driving the population towards more promising solutions. The structure of GA obscures this fact slightly but it too follows the same paradigm with newly discovered fit solutions slowly propagating throughout the candidate set over time, yielding even fitter solutions in the next generation and, thus, resulting in a collective convergence towards more optimal values.

While there are countless variants to each of the above algorithms that typically enhance their usability and robustness, the following experimentation only focuses on the most basic versions of these algorithms in order to ensure a more level playing field for comparison. Similar

to PSO, the best known hyperparameter values, as determined by literature, are used for each algorithm to bolster performance. To further facilitate even comparisons and reduce the confounding influence of another variable, the population size is set as 50 across the board, regardless of whether this is the optimal value for each algorithm in its current configuration. The number 50 is chosen as it appears to be the best value for PSO in preliminary experiments with the given hyperparameters. The algorithms are each run for a maximum of 5,000 iterations to provide enough time for convergence while also being mindful of the computational complexity. Since the objective here is to assess the algorithms' performance on global optimization tasks, five common benchmark functions of varying dimensionality and complexity are chosen to test the algorithms' ability to locate the global minimum. The details of these benchmark functions are provided in Table 1 and Figure 2. Each algorithm is run 10 times on each of the benchmark functions and the results of the runs are averaged to provide a comprehensive overview of performance. Interestingly, all the algorithms appear to exhibit convergent behavior within the specified number of iterations, albeit not necessarily converging to or near the optimal solution. See Figure 3 for an example of PSO's convergence profile on a particular benchmark function. Noting this, a generalized definition of convergence is used herein where an algorithm is considered to have converged if its global best value remains stagnant for at least 1,000 iterations. If this criterion is not met, then it is assumed that the algorithm converges at iteration 5,000 given the general trajectory of the convergence profile. This assumption is made to facilitate simpler comparisons between the convergence speeds of different algorithms. Beyond measuring the minimum, maximum, and average convergence times of each algorithm, key metrics include the average actual runtime of the algorithm and the minimum, maximum, and average absolute error upon convergence (i.e. distance to the true

global minimum). The full results of the experimentation are not included here but can be found in the attached CSV file. Due to its considerably longer runtime, the FA results are located in a separate CSV file.

Examining the results, several interesting patterns emerge. First, it appears as if PSO excels in finding near-optimal solutions in lower dimensionality cases. The other algorithms also flourish in these simpler cases, but PSO sets itself apart by not only finding more accurate solutions but also converging to them more quickly on average. The runtime for PSO is also relatively low across the board, underscoring the efficiency of the algorithm. Despite its quick convergence and runtimes, PSO's performance deteriorates significantly when presented with higher dimensional functions. This holds true for the other optimizers as well, except for ABC. Interestingly, ABC performs quite well on higher dimensional optimization problems, providing near-optimal solutions more often than not in a reasonably efficient manner. Although its runtime tends to be a little longer than the other algorithms, it certainly compensates for this with superior performance. It can be speculated that this increase in runtime and performance stems from ABC's scouting mechanism that randomly explores new solutions upon excessive stagnation in existing solutions. Thus, though more thorough experimentation is certainly required to validate these results and create a more robust experimental framework, preliminary results demonstrate the relative promise of all five algorithms for the task of global optimization with ABC emerging as the most consistent performer overall.

In conclusion, the implementation and exploration of swarm intelligence algorithms offer valuable insights not only into their distinct search characteristics but also into their collective utilization of self-organizational principles to seek out optimal solutions. While the presented work only focuses on a select subset of algorithms among the diverse array available, the

examination of these specific algorithms is enough to shed light on the inherent reliance of swarm intelligence on self-organization. This reliance emphasizes the necessity for swarm members to cooperate and coordinate in achieving desired outcomes, effectively demonstrating how complex, widespread patterns can emerge from seemingly simple, random interactions over time. Understanding this overarching concept is crucial for appreciating the potency of such natural phenomena and, thus, their potential applications in computational contexts moving forward.

Works Cited

Holland, J. H. (1992). Genetic Algorithms. *Scientific American, 267(1),* 66–73.

Karaboga, D., & Basturk, B. (2007). Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems. *Foundations of Fuzzy Logic and Soft Computing, 4529,* 789-798.

Clerc, M., & Kennedy, J. (2002). The Particle Swarm — Explosion, Stability, and Convergence in a Multidimensional Complex Space. *IEEE Transactions on Evolutionary Computation, 6(1),* 8–73.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *In Proceedings of ICNN'95 - International Conference on Neural Networks, 4,* 1942-1948.

Gandomi, A., & Yang, XS., & Alavi, A. (2013). Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. *Engineering With Computers, 29,* 245-245.

Yang, XS. (2009). Firefly Algorithms for Multimodal Optimization. *Stochastic Algorithms: Foundations and Applications, 5792.*

Appendix

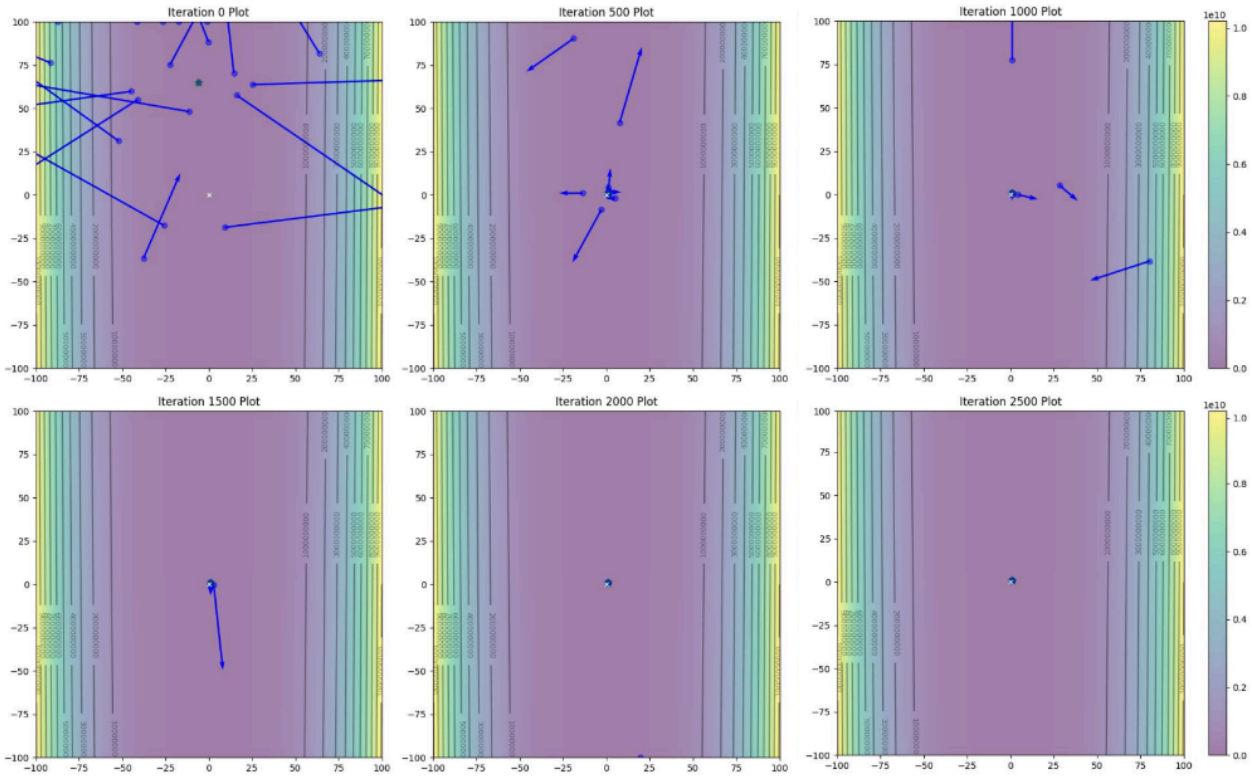

Figure 1: PSO update visualization on contour plot of 2D Rosenbrock function. Particles are blue dots, *gbest* is green star, and global minimum is white X.

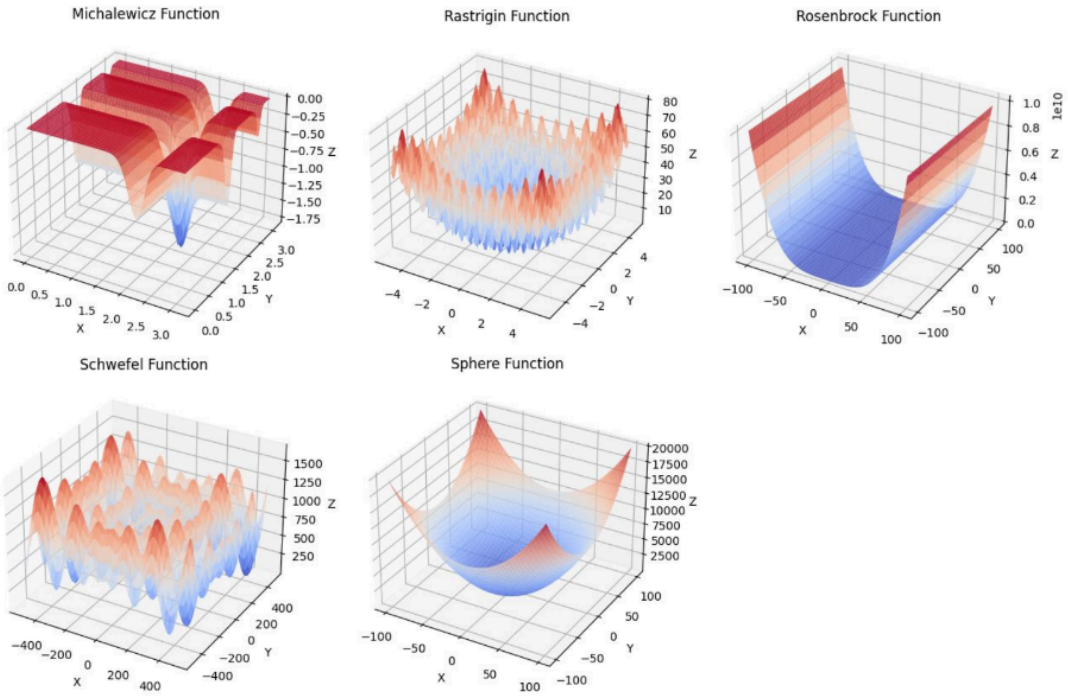

Figure 2: Benchmark function plots (all reduced to 2D for visualization purposes).

Function	Dimensionality	Search Space Lower Bound	Search Space Upper Bound	Target (Global Min)
Michalewicz	5	0	π	-4.687658
Rastrigin	30	-5.12	5.12	0
Rosenbrock	2	-100	100	0
Schwefel	30	-500	500	0
Sphere	30	-100	100	0

Table 1: Benchmark function statistics.

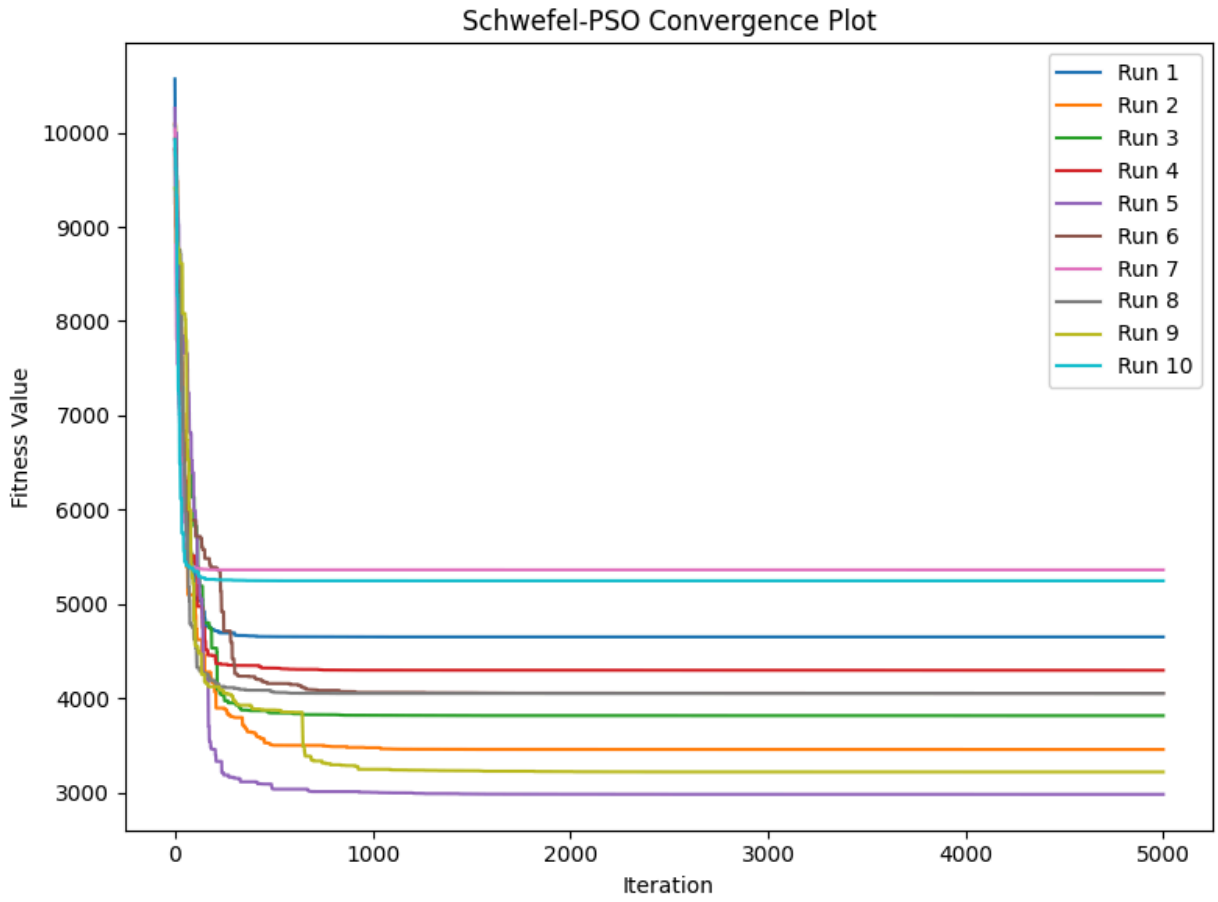

Figure 3: PSO convergence plots on Schwefel function over 10 runs.