A Comparative Study on Hybrid Genetic Algorithm-Boids as Evolutionary Algorithms

Mohamed Affan Dhankwala MDHANKW1@JH.EDU
Masters of Artificial Intelligence
Johns Hopkins University
Baltimore MA, USA

Abstract

Optimization of continuous real-value functions involves minimizing a function over a specific landscape. Evolutionary computation and swarm intelligence offer an efficient solution to handle these complex environments through established methods such as genetic algorithms (GAs) and particle swarm optimizations (PSOs). This paper introduces Reynolds's bird-oid (Boids) flock-simulating particles into the realm of continuous function optimization in the hybrid GA-Boids algorithm. Contrary to our hypothesis, this algorithm did not demonstrate statistically higher performance than our other algorithms in our three continuous evaluation functions, but demonstrated almost superior performance in the highest dimension. We believe this may be due to some clashing in the GA-Boids phases that is overlooked in higher dimensions, and would propose further research into performance in said higher dimensions.

1 Introduction

Optimization of continuous real-valued functions remains a cornerstone for evolutionary computation and swarm intelligence. These problems generally demonstrate multi-modality, non-linearity, noise, and complex shapes that render traditional deterministic gradient-based methods unsuitable. Back (1996) and Engelbrecht (2005) are among the many researchers who developed population-based metaheuristics that performed optimization based on probabilistic searches and biologically inspired mechanisms. This is the basis of evolutionary computation and swarm intelligence.

Evolutionary algorithms (EAs) follow the natural reproductive rules of selection, crossover, and mutation. They consist of a set of individual solutions that navigate an environment without prior knowledge of the landscape (Jong 1975; Holland 1975). Swarm intelligence (SI) methods are similar to EAs in that they likewise consist of particles. Rather than focusing on evolution over generations, SI methods rely on local communication and distributed exploration to navigate and optimize functions (Kennedy and Eberhart 1995; Engelbrecht 2005).

This paper selects a handful of classical and competitive EA and SI algorithms and demonstrates their capability on various continuous evaluation functions. The following page consists of brief introductions to each algorithm.

1.1 Introduction of Algorithms

Introduced by Holland (1975), genetic algorithms (GAs) consist of a set of individuals that evolve over generations. Evolution is performed through selection, crossover, and mutation rates. Over generations, these operators begin biasing the search towards more fit regions. Individuals are required to maintain adequately balanced exploration and exploitation rates in the landscape. This is especially true for larger or higher dimensional environments.

Particle swarm optimization (PSO) is a swarm-based algorithm in which particles have associated velocity vectors that dictate their motion through a state space (Kennedy and Eberhart 1995). Over time, cognitive and social components combine to update the velocity of each particle, and particles swarm around fitter areas of the state space.

Hybrid-PSOs are a branch of PSOs that combine additional heuristics or complementary methods, including local search or PSO-style updates, to improve metrics such as adaptability or convergence (Talbi 2002). More details on the selected hybrid PSO algorithm are provided in a later section.

Proposed by Mendes et al. (2004), fully informed particle swarm (FIPS) is a variant of the classic PSO algorithm which expands each particle's communication to not just be limited to it's best neighbor, but rather to all neighboring particles. This increased flow of information enables smoother movement and more distributed influence. Even in higher dimensions, FIPS demonstrates diverse particle trajectories and maintains swarm coherence.

Unlike the other algorithms, Reynolds's bird-oid (Boids) model is not an EA or SI algorithm. Rather, it is a means of simulating biological flocking and grouping behavior. It imposes the rules of alignment, cohesion, and separation upon particles in a groups, which results in them attempting to maintain a certain distance between one another while being informed of the group's heading. This dynamic allows individuals to access both local and global information. In the proposed GA-Boids framework, the Boids behavior is used to coordinate movement in the search space while GA operators guide the long-term adaption. More information on this hybrid is discussed in a later section.

1.2 Problem Statement

The success of EAs and SIs originates from their decentralized adaptation to the environment, as particles explore wide expanses of the state space. However, these algorithms are not perfect. As the dimensionality of the problem increases, the search space likewise grows and the search volume multiplies exponentially. More attention is placed on diversity and exploration efficiency to prevent stagnation or random drift (Jong 1975). This paper presents a handful of algorithms that offer evolutionary computation on a set of continuous evaluation functions and highlights each algorithm's strengths and weaknesses. We hypothesize that our devised GA-Boids hybrid model will converge at a lower value (better converged fitness) than the GA and PSO algorithms due to flock-like spacing, local communication with neighbors, and global communication with overall flock fitness. We also hypothesize that the GA-Boids model will perform competitively with Hybrid-PSO and FIPS due to it's population diversity and spatial dispersion qualities.

2 Related Work

Earlier work established the foundations of population-based search with Holland (1975) and Back (1996) demonstrating the excellence of recombination and mutation mechanics for exploring complex landscapes. Eiben and Smith (2015) highlighted how subsequent developments of evolution strategies and genetic programming (GP) have matured into powerful tools to counteract the complexity of multi-modality, noise, and non-convexity in continuous domains.

GAs are one of the most well known classic optimizers. Many studies have been conducted, including Jong (2006) that examine and conduct studies on selection, crossover, and mutation designs to influence GA optimization. Elitism, adaptive mutation rates, and niching methods are among the many improvements that have consistently demonstrated greater stability and maintain solution diversity among optimization tasks. Due to their foundational importance, GAs have become the central baseline for comparative evaluation of new evolutionary and swarm algorithms.

Originally derived from social behavioral models, PSOs emphasize distributed learning and information sharing among particle populations (Kennedy and Eberhart 1995). Similarly to GAs, there have been many extensions to PSO algorithms to improve aspects including convergence stability, exploration-exploitation balancing, and enhanced high dimensionality handling. Notable PSO variants include Clerc and Kennedy's constricted PSO, adaptive PSO, and neighborhood-based formulations that restrict or enhance communications.

Although the previously mentioned PSO variants are not evaluated in this paper, FIPS is another powerful PSO variant, and we evaluate it's performance within this paper. This algorithm relies on information from all neighbors rather than a single best neighbor (Mendes et al. 2004). This reduces sensitivity to stochastic fluctuations and allows for a more distributed influence of information. There is also improvement in coordination and maintenance of swarm diversity even in high dimensional, complex, and nonlinear environments.

Another variant that this paper studies involves the broad field of hybrid algorithms. Hybrid-PSOs blend mechanisms from PSO and other algorithms to improve probabilistic search, gradient approximations, or memetic strategies for convergence efficiency (Talbi 2002; Ong et al. 2006). The combination of algorithms allows hybrid-PSO algorithms to capture the complementary strengths of each parent algorithm and employ them to maintain diversity while improving exploitation. The workings of the hybrid-PSO algorithm are discussed in a further section.

Reynolds (1987)'s Boids model has been increasingly referenced in optimization research albeit it's origin in computer graphics and artificial life simulation. The powerful rules of separation, alignment, and cohesion dictate movements of collective motion, all without the need for centralized control. Couzin et al. (2005) and Blackwell and Bentley (2002) have examined adoptions of flocking principles for optimization, routing, clustering, and multi-agent coordination and determined that they may improve the robustness in multi-agent search. This paper will introduce a hybrid Boids algorithm and it's details are covered in a later section.

3 Algorithms

As mentioned previously, this paper introduces a GA-Boids hybrid algorithm to employ on continuous evaluation functions and compares its performance with the established algorithms of GA, PSO, FIPS, and hybrid-PSO. Below we present the implementations of each of the algorithms as well as details on the three Boids rules.

Most of these algorithms begin by identifying a state space and then initializing particles with D dimensions to exist within that space. Each dimension also represents a gene of the individual. Fitness is calculated off of the continuous function's evaluation formula. For example, the formula for the continuous function of the sphere is $f(x) = \sum_{j=1}^{D} x_j^2$ where x_j refers to each gene of an individual x. The other function's formulae are shown in a later portion of the paper, but they are all minimizing functions.

3.1 GA

After initialization of the original set of individuals, GAs evolve these individuals in generations. In each generation, parents are selected using a k-way tournament selection. This involves randomly selecting k individuals from the entire population and preserving the best fitness (lowest fitness value) individual. This is done separately for both parents, and it is possible for both parents to be the same individual.

Once the selection process is conducted, GA creates two children based on the rules of crossover and mutation. Crossover refers to the recombination of parent genes to create better performing offspring. We utilize SBX crossover so that each gene, j, within the two offsprings, $c1$ and $c2$, is computed as $c_{1,j} = \frac{1}{2}[(1+\beta)x_{1,j} + (1-\beta)x_{2,j}]$ and $c_{2,j} = \frac{1}{2}[(1-\beta)x_{1,j} + (1+\beta)x_{2,j}]$ where the β coefficient serves as weight on how similar the offspring's genes are to their parents.

Mutation occurs according to a gaussian mutation in which each gene in an individual, x, has a probability chance to be altered by a normally distributed random variation, $\mathcal{N}(0, \sigma^2)$. The formula for mutation is $x'_j = x_j + \mathcal{N}(0, \sigma^2)$.

This algorithm also supports the concept of elitism. This refers to copying the top E individuals from each generation into the next generation. This ensures that the population does not lose fitness over generations. This algorithm also tracks the best fitting individual across each generation and saves all these values. The algorithm terminates when a set number of generations have elapsed.

3.2 PSO

Once the initial particles are created within the state space, each individual is given a uniformly random constrained velocity value. This corresponds to the traversal direction of the particle within the state space. Each particle keeps track of it's best position, p_i and the respective best fitness value at that position, $f(p_i)$. The swarm also tracks the best position explored by any individual, g_{best}.

At each generation, t, the PSO algorithm performs velocity updates to ensure that each particle balances its exploration and exploitation actions. This is done using the standard PSO equation: $v_i^{t+1} = wv_i^t + c_1 r_{1,i}^t (p_i - x_i^t) + c_2 r_{2,i}^t (g - x_i^t)$ where w is a tuned inertia weight, c_1 is the cognitive acceleration coefficient, c_2 is the social acceleration coefficient and $r_{1,i}$

and $r_{2,i}$ are random uniform coefficients. The cognitive acceleration coefficient influences how strongly a particle is pulled by it's personal best position while the social acceleration coefficient influences how strongly it is pulled towards the global best position. The two random uniform coefficients introduce stochasticity to prevent deterministic movement and mitigate the risk that all particles take identical trajectories. After updating the velocity, the position of each particle is adjusted via a simple increment $x_i^{t+1} = x_i^t + v_i^{t+1}$. Both the velocity and the position vectors of these particles are constantly constrained by predefined bounds.

3.3 FIPS

As a continuation of the PSO algorithm, FIPS also stores the current individuals best position. However, each individual is also simultaneously influenced by its k neighbors. Therefore, the influence factor, I, is defined as: $I_i = \sum_{j \in N_i} r_{ij}(pbest_j - x_i)$ where N_i is a set of randomly selected k neighbors, r_{ij} is a uniformly distributed random value per dimension stochastic scaling, $pbest_j$ is the best neighbor's (j), personal best, and x_i is the current particle's position.

The velocity update for each particle is $v_i^{t+1} = wV_i^t + \frac{c}{k} \times I_i$ where w is the inertia weight and c is a unified acceleration coefficient. The position update is of similar format to traditional PSO and both velocity and position are bounded by predefined values. The unified acceleration coefficient is a tuned constant scalar that controls the overall attraction strength to the set of neighbors. Similar to the cognitive and social acceleration coefficients in the PSO algorithm. This represents the fundamental difference between PSO and FIPS in that an individual in the former algorithm is aware of itself and the global best position, while an individual in the latter algorithm is aware of itself and its neighborhood.

3.4 Hybrid-PSO

This hybrid algorithm functions in the same way that a traditional PSO algorithm operates. This includes preserving each particle's individual best position and best fitness as well as the global best position and fitness. The velocity and position updates are identical to traditional PSO algorithms. The hybrid component of this algorithm is the utilization of a mutation phase pulled from GA algorithms. When the swarm has not improved for a predefined number of generations, each particle has the potential to be subjected to a uniformly distributed random mutation $x'_j = x_j + \mathcal{N}(0, \sigma^2)$. This is done to avoid stagnation and help the swarm escape local minima. This is the only change that has been implemented in this algorithm to differentiate it from classical PSO algorithms.

3.5 Boids

Although not an optimization function, boids display flock-like separation of individuals that can be crucial for well balanced exploration-exploitation phases. There are three rules that each particle of this swarm must follow. These are the laws of separation, alignment, and cohesion.

Separation is the avoidance of crowding neighbors. In other words, the boids should not overlap or intersect other nearby boids. It is defined as $v_i^{sep} = \sum_{j \in N_i} \frac{r_i - r_j}{|r_i - r_j|}^2$ where r_i and r_j are positions of nearby boids i and j, and N_i is the set of all neighbors in a predefined radius of the boid i.

Alignment is the action of steering towards the average heading of neighbors. Instead of moving towards the neighbors, alignment steers boids to move where their neighbors are moving. It is defined as $v_i^{align} = \frac{1}{|N_i|} \sum_{j \in N_i} v_j$ where v_j is the velocity of neighboring boid j and N_i is the same as defined previously.

Cohesion is steering towards the average position of neighbors. This counteracts separation and keeps the boids from spreading too far from each other. It is defined as $v_i^{cohesion} = \frac{1}{|N_i|} \sum_{j \in N_i} (r_j - r_i)$.

During each generation, each particle's velocity vector is updated as such: $v_i^{total} = v_i^{sep} + v_i^{align} + v_i^{cohesion}$ This is a simple sum of the previously mentioned laws.

3.6 GA Boids

This algorithm works in two phases. The first phase is the boids phase, which performs velocity updates based on an updated version of the previously defined boids formulae. The second phase involves the GA phase that selects the best individuals for future generations.

During the boids phase, v_i^{sep}, v_i^{align}, and $v_i cohesion$ are calculated based on a predefined radius size around each particle. We also make sure to determine the fitness at the best position g_best similar to a classic PSO algorithm. Thus, our velocity update function is $v_i^{t+1} = wv_i^t + c_{sep} v_i^{sep} + c_{align} v_i^{align} + c_{cohesion} v_i^{cohesion} + c_{gbest}(g_{best} - x^i)$ where each coefficient c represents a weight for each respective operand. The velocity and position are bounded by predefined values.

The GA phase implements the previously mentioned tournament selection, SBX crossover, gaussian mutation, and preservation of elites during each generation. This occurs in each generation right after the boids velocity and position updates.

4 Continuous Evaluation Functions

In this section, we introduce the three continuous evaluation functions that we chose. Each function description will be complimented with it's reasoning and formula. Visuals are provided below the three descriptions.

4.1 Sphere

The sphere function is a classic uni-modal function that simply tests an algorithm's ability to converge on a global optima. It serves as a baseline function to determine whether the EA and SI algorithms are able to converge at all. It is a smooth convex function $f(x) = \sum_{i=1}^{n} x_i^2$ proposed by Jong (1975).

4.2 Rosenbrock

Rosenbrock is a more complex function that offers a narrow valley quality where the global optimal is difficult to find within this banana-shaped landscape. It rigorously tests a model's ability to converge efficiency and robustness to large landscapes. It is defined as $f(x) = \sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$ and was proposed by Rosenbrock (1960).

4.3 Rastrigin

Proposed by Rastrigin (1974), the rastrigin landscape is highly multimodal and tests a model's ability to escape local minima and continue with global optimization. The function is defined as $f(x) = 10n + \sum_{i=1}^{n}[x_i^2 - 10cos(2\pi x_i)]$.

4.4 Visuals

This section contains each of the continuous evaluation functions visualized:

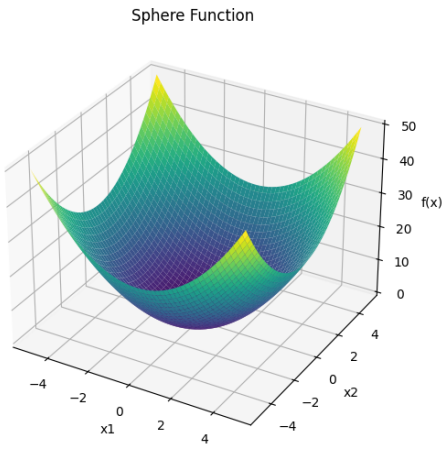

Fig. 1: Sphere Landscape

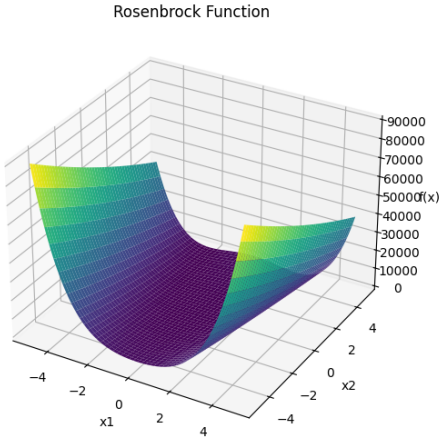

Fig. 2: Rosenbrock Landscape

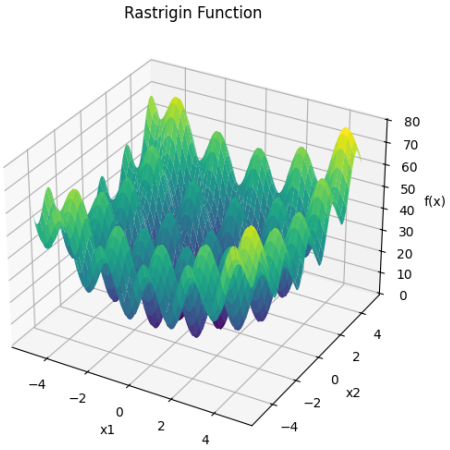

Fig. 3: Rastrigin Landscape

5 Methodology

To perform a fair comparative evaluation of each of the previously mentioned algorithm's performance on the three continuous evaluation functions, we tuned all of the algorithms on each of the continuous evaluation functions separately. Once an algorithm's set of hyperparameters was tuned, we ran the algorithm fifty times on each of the continuous evaluation functions. After each run, the best fitness for each generation was saved in a list and plotted along a graph to demonstrate the convergence rate. The best fitness among all generations was saved as the final converged optimized value. After running all algorithms fifty times through each of the continuous evaluation functions, we changed the dimensionality of the particles and repeated the method. This was done for the low, medium, and high dimension values of five, ten, and one hundred, respectively.

5.1 Hyperparameter tuning

Hyperparameter tuning was conducted to ensure that each of the algorithms was given a fair chance to demonstrate it's optimal performance. For each algorithm, we identified the relevant set of hyperparameters and augmented a list of potential values that each of these parameters could take. We then ran the algorithm on all three continuous evaluation functions five times each. The averaged optimized value for all of these functions would be saved. After testing all possible hyperparameter combinations, we saved the tuned values based on the best performance in each of the three functions. Note that this meant that every algorithm had three sets of tuned hyperparameter values that correlated with the three continuous evaluation functions.

6 Experimental results

The convergence rates of all algorithms are defined below. Each graph has gray and black lines. The gray lines represent each of the fifty convergence trials, while the black line refers to the average performance at each generation. We show all nine graphs for the GA algorithm, but the lower dimensions of five and thirty generally show similar trends of early convergence across algorithms. For the sake of brevity, we only display the 100 dimension evaluations for the rest of the algorithms

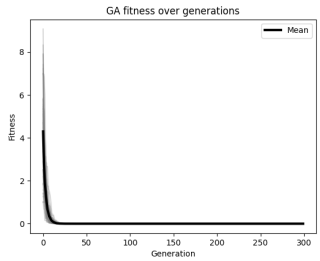

Fig. 4: Sphere 5D

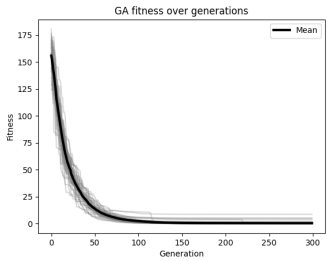

Fig. 5: Sphere 30D

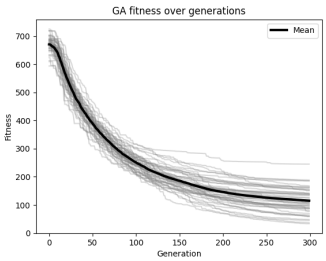

Fig. 6: Sphere 100D

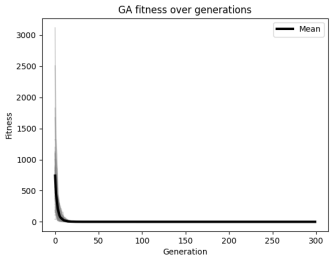

Fig. 7: Rosenbrock 5D

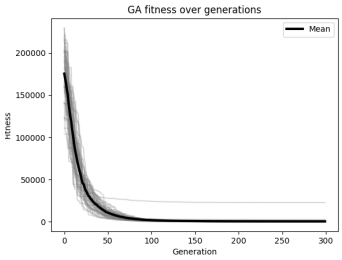

Fig. 8: Rosenbrock 30D

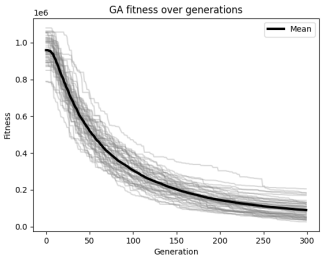

Fig. 9: Rosenbrock 100D

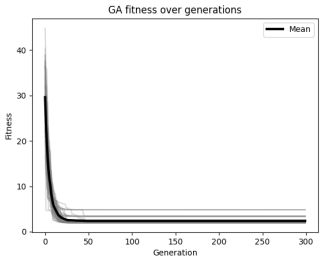

Fig. 10: Rastrigin 5D

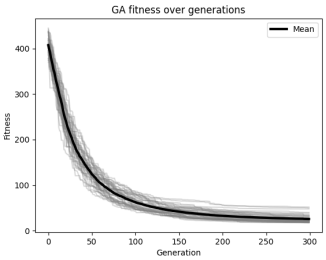

Fig. 11: Rastrigin 30D

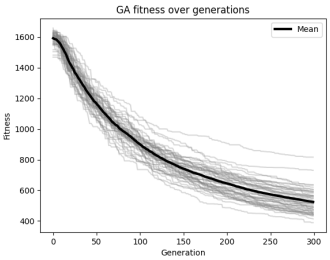

Fig. 12: Rastrigin 100D

Fig 4-12: GA on sphere, rosenbrock, and rastrigin functions

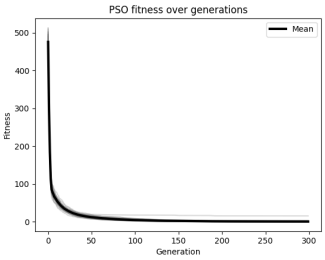

Fig. 13: Sphere 100D

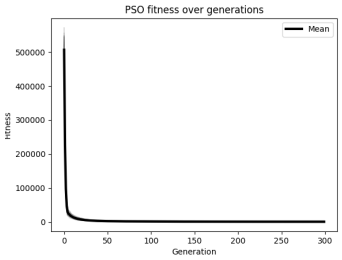

Fig. 14: Rosenbrock 100D

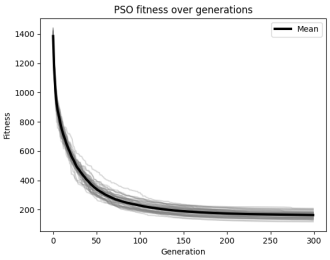

Fig. 15: Rastrigin 100D

Fig 13-15: PSO 100D on sphere, rosenbrock, and rastrigin functions

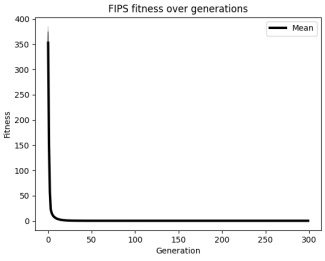

Fig. 16: Sphere 100D

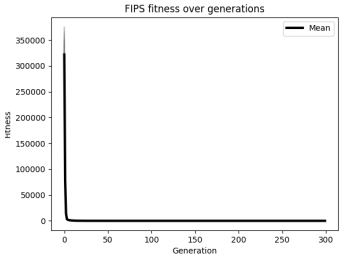

Fig. 17: Rosenbrock 100D

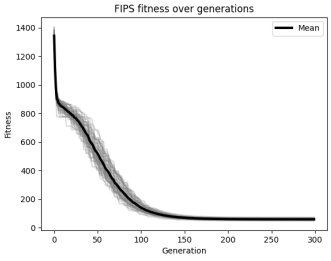

Fig. 18: Rastrigin 100D

Fig 16-18: FIPS 100D on sphere, rosenbrock, and rastrigin functions

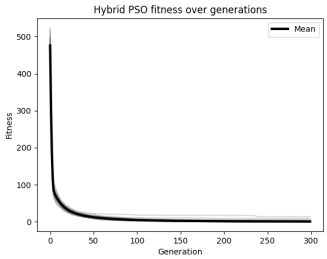

Fig. 19: Sphere 100D

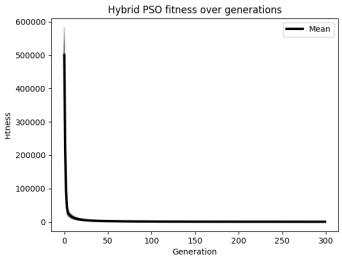

Fig. 20: Rosenbrock 100D

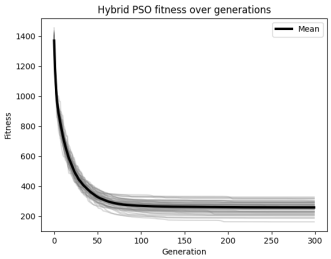

Fig. 21: Rastrigin 100D

Fig 19-21: Hybrid-PSO 100D on sphere, rosenbrock, and rastrigin functions

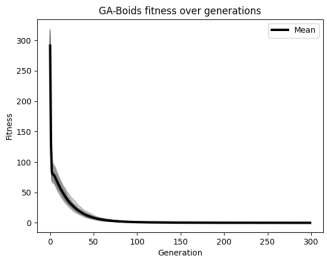

Fig. 22: Sphere 100D

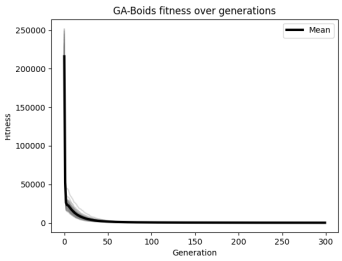

Fig. 23: Rosenbrock 100D

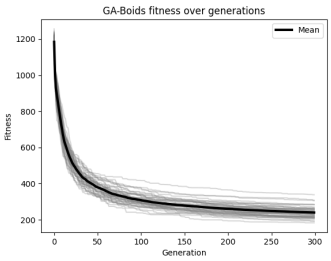

Fig. 24: Rastrigin 100D

Fig 22-24: GA-Boids 100D on sphere, rosenbrock, and rastrigin functions

The finalized performance tables are presented below.

Table 1: Means of algorithms for D=5

D=5			
Algorithm	Sphere	Rosenbrock	Rastrigin
GA	3.04E-25	0.4884	2.3851
PSO	4.72E-61	0.7257	2.8979
FIPS	5.46E-39	2.5564	2.3546
Hybrid-PSO	4.16E-61	0.8204	3.8952
GA-Boids	9.98E-04	1.3974	4.3660

Table 2: Means of algorithms for D=30

D=30			
Algorithm	Sphere	Rosenbrock	Rastrigin
GA	0.5540	648.9626	25.8208
PSO	2.85E-09	33.0095	40.8334
FIPS	2.10E-06	28.3950	16.1339
Hybrid-PSO	1.94E-07	32.3909	39.7533
GA Boids	2.75E-03	38.2818	40.4048

Table 3: Means of algorithms for D=100

D=100			
Algorithm	Sphere	Rosenbrock	Rastrigin
GA	115.1532	9.02E04	524.1952
PSO	0.8122	464.2251	163.1846
FIPS	0.2832	131.7568	59.3207
Hybrid-PSO	1.0292	373.5987	258.5050
GA Boids	0.1207	256.7517	239.9654

7 Discussion of Results

From the graphs, it can be deduced that all the algorithms manage to handle the sphere dataset fairly well, even when individuals are composed of 100 dimensions. The classic GA algorithm, however, demonstrated much slower convergence as the dimensionality grew and did not fully converge on the 100D functions after the predefined generations. The rastrigin evaluation function caused many of the algorithms to have slower convergence rates compared to their rates on other functions. FIPS even demonstrates some odd quick convergence followed by a new second convergence pattern.

Focusing on the performance of our GA-Boids algorithm, we see that it's convergence rate on the sphere function is slightly slower than FIPS and visually comparable to hybrid PSO and PSO. There also seems to be some downward trend near the end of the 300 generations in the rastrigin 100D function. This suggests that the algorithm had not fully converged. From Table 1, we see that GA-Boids had the worst convergence value for the sphere dataset by a magnitude of 10^{21} while PSO and hybrid PSO sported values 10^{57} times smaller. FIPS, although exceeding GA, failed to perform nearly as well as the other PSO methods, and this may hint at some detriment caused by the additional information from neighboring solutions. GA-Boids has elements of both GA and the same neighbor information that FIPS has so it's poor performance could be related to some negative transfer from these neighbors and evolution based on this data. GA-Boids had slightly worse but still comparable results for the rosenbrock and rastrigin functions.

As the dimensionality of the functions increased, GA's performance deteriorated exponentially while FIPS's performance became relatively high. GA-Boids sported slightly less performance to the PSO and hybrid PSO algorithms in the thirty dimensional functions but surpassed most of them in the one hundred dimensional functions. This could highlight the robustness of the flock-like behavior that the particles in GA-Boids exhibit in adapting to high dimensional environments.

Our original hypothesis was that GA-Boids would demonstrate a lower fitness value (higher performance) than GA and PSO and would remain competitive with the hybrid PSO and FIPS algorithm in continuous evaluation functions. To statistically determine this value, we utilize Welch's t-test $t = \frac{\bar{x_1} - \bar{x_2}}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$ where $\bar{x_1}$ and $\bar{x_2}$ are the mean fitness values of two algorithms, s_1^2 and s_2^2 are the sample variances, and n_1 and n_2 are the sample sizes. We also calculate the degrees of freedom (df) based on Welch-Satterthwaite degrees

of freedom and use the values of t and df to calculate a one-sided p value to determine whether our GA-Boids algorithm performs better than the other algorithms.

Comp. Perf	Algorithm	D=5	D=30	D=100
Sphere	GA	WORSE	BETTER	BETTER
	PSO	WORSE	WORSE	BETTER
	FIPS	WORSE	WORSE	BETTER
	Hybrid PSO	WORSE	WORSE	BETTER
Rosenbrock	GA	WORSE	WORSE	BETTER
	PSO	WORSE	WORSE	BETTER
	FIPS	BETTER	WORSE	WORSE
	Hybrid PSO	WORSE	WORSE	BETTER
Rastrigin	GA	WORSE	WORSE	BETTER
	PSO	WORSE	WORSE	WORSE
	FIPS	WORSE	WORSE	WORSE
	Hybrid PSO	WORSE	WORSE	BETTER

Fig. 25: Comparative Performance between GA-Boids and other algorithms

From figure 25, we can see that the GA-Boids algorithm performs statistically worse than the other algorithms in the small to medium dimensions. Only in the high dimensional functions do we see competitive results from the GA-Boids algorithm. Reflecting upon our original hypothesis, we can conclude that we were incorrect to expect GA-Boids to demonstrate better performance than GA or PSO and we can likewise see that there is no clear evidence of comparable performance with FIPS and the hybrid PSO algorithm. However, we have statistical evidence demonstrating that GA-Boids performs comparatively better than the other algorithms in 100D and only falls short of the FIPS algorithm.

7.1 Potential Areas of improvement

Overall, our GA-Boids algorithm performed worse than we had originally expected. There is statistical evidence that it performed better than some of the algorithms in the 100D function environments, but failed to do so in the lower dimensions. Upon reflecting on it's design, we have some ideas to explain this phenomenon. Since our GA-Boids algorithm is split into two phases, the Boids phase and the GA phase, there can be extra local motion in lower and medium dimensions as the two phases work against each other. Boids dynamics may encourage particles to push away from each other and promising locations, while GA evolves them to prioritize exploitation. In higher dimensions, the environment is much more vast, so, while relevant, this issue is not as impactful, thus local structure is preserved allowing individuals to balance exploration and exploitation.

Improvements we can make upon our GA-Boids model to compensate for this clashing of phases is never performing both Boids and GA updates within the same generation

and either alternate between the two strategies or weighted random selection. If the issue continues to remain, it might hint that flocking dynamics, while powerful in maintaining population diversity and spacing, may not be compatible with GA.

Aside from this potential clashing of phases, another way of improving performance is incorporating more dynamic parameters including neighbor radius, weights, and elite counts. These variables would change according to the population diversity metrics, the best fitness, and the position of the swarm within the state space. Of course, if our original suspicion that Boids and GA may be incompatible is correct, these changes will still not resolve the underlying incompatibility issue.

8 Conclusion

From this experiment, we combined the Boids flocking mechanisms with GA evolution strategy results to create a GA-Boids hybrid algorithm. Originally, we hypothesized that this algorithm would optimize continuous evaluation functions with higher performance (lower final converged value) than classical GA and PSO algorithms and would compare with the PSO variant, FIPS, and hybrid-PSO. From tables 1, 2, and 3 and our comparative performance figure 25, we can see that our GA-Boids algorithm often performed worse than all the other algorithms in low and medium dimensions. In the 100 dimensional functions, however, our GA-Boids algorithm performed exceptionally well and surpassed most of the other algorithms. These results can be explained by potential clashing between GA and Boids frameworks as they attempt to counteract each other's exploration and exploitation efforts. This clashing could cause significant convergence issue in smaller to medium dimensions, but may have been overlooked in higher dimensions simply because of how vast the landscape is. Improvements for this design may be to reconsider the compatibility of GA and Boids or to consider making some of the hyperparameters dynamic to self-adjust over the course of optimization.

8.1 Further Research

As mentioned earlier, further research into this GA-Boids algorithm would be to test this algorithm on higher dimensions to determine if our performance on 100D was merely a fluke or whether there is statistical evidence pointing at better performance in higher dimensions. Since Boids is relatively new in the field of optimization, exploration of other hybrid options may give way to some new information and place Boids more firmly in the field of optimization. Further research could also be conducted to determine how well GA-Boids performs in other tasks that evolutionary computations and swarm intelligence should handle such as task scheduling and feature reduction.

References

T. Back. Evolutionary algorithms in theory and practice. *Oxford University Press*, 1996.

T. Blackwell and P. Bentley. Dynamic search with charged swarms. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 19–26, 2002.

M. Clerc and J. Kennedy. The particle swarm–explosion, stability and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6 (1):58–73, 2002.

I.D. Couzin, J. Krause, N. R. Franks, and S. A. Levin. Effective leadership and decision-making in animal groups. *Nature*, 433:513–516, 2005.

A. E. Eiben and J. E. Smith. Introduction to evolutionary computing. *Springer*, 2015.

A. P. Engelbrecht. Fundamentals of computational swarm intelligence. *Wiley*, 2005.

J. H. Holland. Adaptation in natural and artificial systems. *University of Michigan Press*, 1975.

K. A. De Jong. *An Anlysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, 1975.

K. A. De Jong. Evolutionary computation: A unified approach. *MIT Press*, 2006.

J. Kennedy and R. Eberhart. Particle swarm optimization. *Proceedings of the IEEE Interational Conferences on Neural Networks*, pages 1942–1948, 1995.

R. Mendes, J. Kennedy, and J. Neves. The fully informed particle swarm. *IEEE Transactions on Evolutionary Computation*, 8(3):204–210, 2004.

Y. S. Ong, M. H. Lim, N. Zhu, and K. Wong. Classification of adaptive memetic algorithms: A comparative study. *IEEE Transactions on Systems, Man, and Cybernetics*, 36(1):141–152, 2006.

L. A. Rastrigin. Systems of external control. *Mir, Moscow*, 1974.

C. W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *ACM SIGGRAPH*, 21(4):25–34, 1987.

H. Rosenbrock. An automatic method of finding the greatest or least value of a function. *The Computer Journal*, 3(3):175–184, 1960.

F. E. Satterthwaite. An approximate distribution of estimates of variance components. *Biometrics Bulletin*, 2(6):110–114, 1946.

E.-G Talbi. A taxonomy of hybrid metaheuristics. *Journal of Heuristics*, 8(5):541–564, 2002.

B. L. Welch. The generalization of student's problem when several different population variances are involved. *Biometrika*, 34(1–2):28–35, 1947.

B. L. Welch. On the comparison of several mean values: An alternative approach. *Biometrika*, 38(3–4):330–336, 1951.