# Black Wolf Architechture: A Hybridized Self-Adaptive and Automated Computational Framework

***Abstract*—Multiple nature-inspired algorithms have been successful in selective optimization domains. Computational intelligence branches into Swarm Intelligence (SI) and Evolutionary Algorithms (EA), each providing a unique approach to multi-objective optimization based problem solving. In this paper, we present Black Wolf Architecture (BWA), an intuitive synergy of the best attributes of SI and EA techniques. BWA has boosted accuracy levels to 95-97% for real-world oncological datasets, which is a significant improvement on the current accuracy levels of 75-87.5%. BWA incurs overhead in terms of time and computational resources taken. We have overcome this pitfall by accelerating BWA on HPC platforms with multiple GPUs, that delivered a markup performance improvement of up to 517x.**

***Index Terms*—BWA, Multi-Objective Optimization, Computational Intelligence, Swarm Optimization, Evolutionary Algorithms, GPU acceleration.**

## I. Introduction

### A. *Nature Inspired Multi-Objective Models: An Overview*

The traditional search and optimization techniques for either single-objective or multi-objective functions employ a single candidate solution which is tweaked to find the sweet spot of the global optimum. Multi-objective Functions have a general the form described in Equation 1. A single Multi-objective Function can be viewed as $M$ single-objective functions [1], having $K$ equality bounds and $J$ inequality bounds. The $n$ input vector components $x_i$ are also bounded within a lower ${x_i}^L$ and upper ${x_i}^U$ bound.

$$\begin{aligned} Maximize/Minimize: f_{\mathrm{m}}(x) \qquad & m = 1, 2 \ldots, M \\ constraints: g_{\mathrm{j}}(x) \geq 0 \qquad & j = 1, 2 \ldots, J \\ h_{\mathrm{k}}(x) = 0 \qquad & k = 1, 2 \ldots, K \\ {x_{\mathrm{i}}}^{\mathrm{L}} \leq x_{\mathrm{i}} \leq {x_{\mathrm{i}}}^{\mathrm{U}} \qquad & i = 1, 2 \ldots, n. \end{aligned} \tag{1}$$

Solving these objective functions using traditional, single-candidate, unguided methods such as dynamic programming or greedy search has been successful when problems have a strong mathematical representation. Recent research focuses on unorthodox approaches in nature-based computational intelligence, employing both heuristic and meta-heuristic techniques, that offer approximate solutions to truly multi-objective functions. Whilst heuristic algorithms provide an acceptable yet non-optimal solution, meta-heuristics take this to the next level by introducing a fluid design of generation and selection of a set of feasible candidate solutions.

Fig. 1 outlines two major nature-based optimization techniques, Evolutionary algorithms (EA) and Swarm Intelligence (SI) based algorithms [2]. EA adopts the process of evolution and theory of natural selection to account for the survival and

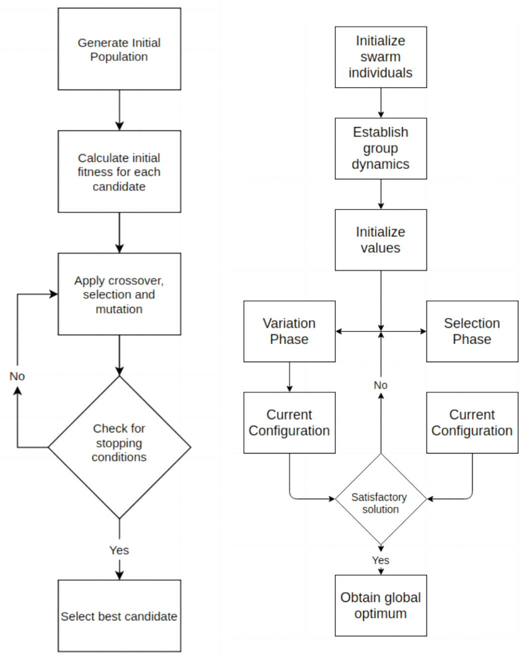


Fig. 1: (1a). Evolutionary Algorithms (EA) (1b). Swarm Intelligence (SI)

adaptability of candidate solutions. SI models the collective interaction between the candidate solutions to self organize, share information amongst each other, and reach a common global optimum.

## II. Existing Methodology

### A. *Base Learning Models*

Any existing machine learning framework involves a programmatic approach of tuning the internal hyperparameters for the desired outputs for a set of inputs. Architectures like neural networks have an extensive set of hyperparameters making them extremely sensitive to their configuration, as it affects the learning process. The most common technique used for selecting the best configuration is a grid search which involves trying out every possible configuration. An improvement in this is made in the form of random search which randomly chooses a configuration and rates it against a fitness function.

The most common form for a good search candidate for an ideal output vector $\vec{f^0}$ is given in Equation 2 [3].

TABLE I: PREVIOUS WORKS ON GWO

| Previous Work | Result Description | Hardware platform |
|---|---|---|
| Zheng et al, [6] | Better results than basic GWO and other algorithms. Improved results on higher dimensional data | Xeon E3-1231 v3 GeForce GTX 750 Ti |
| Jayapriya et al, [7] | Appliocation driven implementation. Reduced computational time | Quadro 4000 |
| S Arora et al, [8] | Providing comprehensive superiority in solving the feature selection problems | Intel i5 3210 |
| E Emary et al, [9] | Optimal Feature combination, minimizing the number of selected features. | None |
| N Singh et al, [10] | Modified position calculation | Intel i5 430 M |

$$f(\vec{x}) = \sum_{i=1}^{k} \frac{\vec{f^0} - f(\vec{x})}{\vec{f^0}} \tag{2}$$

However, high-quality results cannot be guaranteed in a short time and inner tinkering cannot be performed as these networks function as black boxes. A general neural network architecture is shown in Algorithm 1 for *n* training examples in dataset *D*.

### B. Metaheuristic Optimizers

Optimizers belonging to population-based meta-heuristics like SI are extremely efficient due to the distribution of workload and a hive mind based intelligence. Usually, a social construct is also introduced for better memory utilization and preservation of information over the herd for the best results.

Grey Wolf Optimizer (GWO) [4] is one such nature-inspired meta-heuristic that has been put to active effect in various fields. This optimizer impersonates the hierarchical structure of grey wolves as observed in nature. Wolves can be categorized into 4 main groups: $\alpha$, $\beta$, $\delta$, $\omega$. The main steps involved in reaching the global optimum are (a) searching for prey/global optimum, (b) encircling the prey/global optimum, and (c) hunting and attacking the prey/global optimum. The pseudocode for GWO is given in Algorithm 2.

The GWO process aims to imitate the hunting cycle of the entire pack with the parameters being reset at the beginning of each hunt. GWO is a relatively simple algorithm as there are only two main parameters to be tweaked at the beginning of the process. Table *I* enlists several modified meta-heuristics based on GWO that have been developed over the past decade. Whilst the simplistic idea is quite popular and effective, it still suffers from several disadvantages in terms of slow convergence rate, low precision, selective hierarchical structure with skewed mean calculation resulting in a dip in performance.

In this work, we introduce the Black Wolf Architecture (BWA). The BWA is an intuitive synergy of the best attributes of SI and EA techniques, that provides boosted accuracy levels of 95-97% of real-world oncological datasets, which is a significant improvement on the existing 75-87.5% levels. We also take advantage of the parallel nature of the BWA by accelerating the process of multiple GPUs for performance improvement of up to 517x. The remaining sections of this paper describe the BWA, the application of BWA on real-world data sets, and the results from various accelerated computing environments.

**Algorithm 1** General Neural Network

**Input:** D = $\{(x_k, y_k)\}\ \forall\ k \in (1, n)$, hyperparameter configuration = default
**while** Stopping conditions are achieved **do**
    Evaluate $y_k$ based on input
    Calculate $\delta$ i.e difference between $\hat{y}_k$ and $y_k$
    Update weights according to hyperparameter configuration
    Backpropogate the errors
    Evaluate results against standard metrics
**end while**

**Algorithm 2** GWO

**Input:** Population size *n*, random vectors $\vec{r_1}$, $\vec{r_2}$ $\in$ [0,1], initial prey location $\vec{D}$, number of iterations *I*, fitness function *f*, coefficient vectors $\vec{A}$, $\vec{C}$ and $\vec{a}$
Set $t$ = 0
**for** i $\in$ [1,n] **do**
    Generate wolf pack population $X_i(t)$ at instance *t*
    Evaluate each individual against the fitness function
**end for**
Assign $\alpha, \beta, \delta$ titles to the top three solutions
Evaluate $\vec{D} = |\vec{C} \cdot \vec{X}_{\mathrm{p}}(t) - \vec{X}(t)|$
**for** *i* in *I* **do**
    **for** Each individual in *n* **do**
        $\vec{X_1} = \vec{X_\alpha} - \vec{A_1} \cdot \vec{D_\alpha}$
        $\vec{X_2} = \vec{X_\beta} - \vec{A_2} \cdot \vec{D_\beta}$
        $\vec{X_3} = \vec{X_\delta} - \vec{A_3} \cdot \vec{D_\delta}$
        Evaluate $\vec{X}(t+1) = \frac{\vec{X_1} + \vec{X_2} + \vec{X_3}}{3}$
    **end for**
    Update the coefficient vectors $\vec{A}$ and $\vec{C}$
    $\vec{A} = 2\vec{a} \cdot \vec{r_1} - \vec{a}$
    $\vec{C} = 2\vec{r_2}$
    Linearly decrease $\vec{a}$ from 2 to 0
    Update $\vec{X_\alpha}, \vec{X_\beta}, \vec{X_\delta}$
    Increment *t*
**end for**
$\vec{X_\alpha}$ corresponds to the global optimum.

## III. BLACK WOLF ARCHITECTURE: A HYBRIDIZED APPROACH

Black wolves [5] are a genetically modified version of grey wolves found almost exclusively in North America. Often cloaked as a genetic mystery, the introduction of black hair, or *K* variant, into the gene pool serves no visible use to the wolves. The name Black Wolf Architecture (BWA) is derived from the resultant mutation experienced by grey wolves. The BWA pipeline is depicted in Fig. 2.

### A. Remodifying the Basis Function

To improve on the existing methodology, we had to break down the existing architecture into an atomic format and rebuild it with some added functionality. While neural networks are extremely sensitive to hyperparameter tuning, reconfiguring the hyperparameter set for each problem is too tedious and not feasible [11]. The secondary option of exhausting the entire search space was also discarded due to a lack of scalability and the sheer volume of computational resources required. We introduced GA into the mix, resulting in a Genetic Neural Networks (GNN), as illustrated in Algorithm 3 and in Fig. 3. GNNs view hyperparameters as generational values instead of iterative values. The main difference between these two views lies in the updation pattern [12]. While a traditional neural network updates the values based on a set rule, GNNs apply selection, crossover, and mutation to select those values over generations.

In BWA, hyperparameters such as learning rate-dependent weights, momentum, dropout value, batch size, and initial esti-

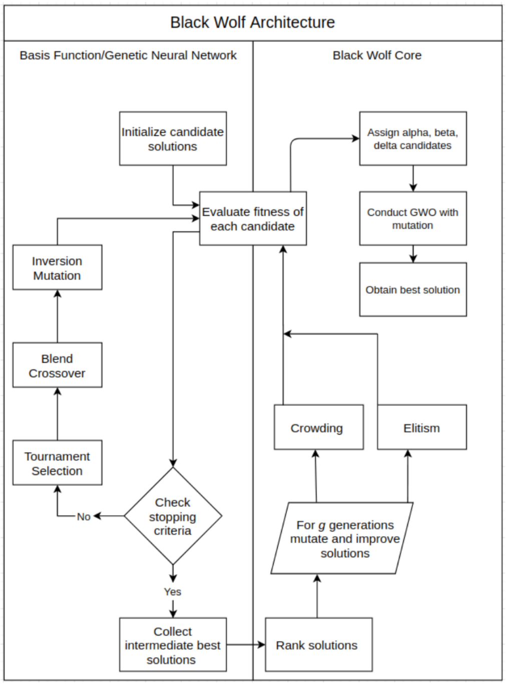


Fig. 2: Overview of BWA

**Algorithm 3** Genetic Neural Netowrk (GNN)

**Input:** Input vector $\vec{x}_i$ and target vector $\vec{t}_i$, population size $n$, fitness function $f$, number of generations $g$
**for** Each individual in $n$ **do**
  Set selection technique as Tournament Selection
  Set Blend crossover method
  Set Inversion mutation technique
**end for**
Create class responsible for genetic grid search encapsulating all techniques specified
Deploy class over $g$ and obtain hyperparameter configuration
Call Algorithm 1 with the hyperparameter configuration and $(\vec{x}_i, \vec{t}_i)$

mates are selected from a pool of candidates by evaluating them against benchmark functions. The proposed genetic framework is developed upon the idea of a conventional grid search. GNN creates a grid and ranks various hyperparameter against each other. Other key details of selection, crossover, and mutation techniques are hashed out manually for optimal results.

The key steps used in Algorithm 3 are:

- **Tournament selection** :
  Potential candidates compete against each other in a round-robin manner till the best candidates remain as depicted in Algorithm 4.
- **Blend crossover(BLX)**:
  Crossover methods take the selected candidates from the selection stages and perform crossover to generate the children of the parent solutions. These offsprings inherit the best characteristics of both its parents and hence is a better candidate solution. In BLX crossover, each offspring is randomly selected from an interval generated by its parents. Equation 3 represents selection of offspring $O_i$ from the range generated by parents $P_{1,i}$ and $P_{2,i}$ where $i$ represents the $i^{th}$ generation and $\alpha$ lies between 0 and 1.

$$O_i = [P_{1,i} - \alpha(P_{2,i} - P_{1,i}), P_{2,i} - \alpha(P_{1,i} - P_{2,i})] \tag{3}$$

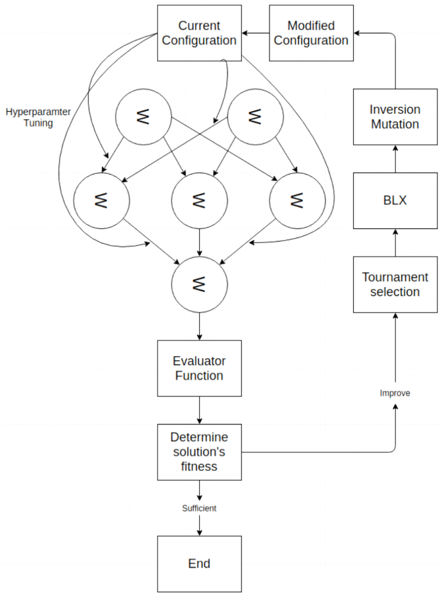


Fig. 3: Genetic Neural Network

**Algorithm 4** Tournament Selection

**Input:** Candidate solution set $S$, competition size $k$, tournament size $m$ and fitness function $f$
**for** Each round in $m$ **do**
  Select $k$ solutions from $S$ without repetition and compare their fitness scores
  Advance the qualified candidate
**end for**

- **Inversion mutation**:
  Mutation is a probability based process which introduces a random change in a candidate solution. In inversion mutation, random hyperparameters $h$ are selected from multiple candidates and swapped. This process is intended to introduce and promote diversity in the solutions.

The process described in Algorithm 3 is repeated for limited runs and a set of best possible network configurations are generated. Iterating over multiple generations and re-evaluating the network certainly has its drawbacks in the form of computational resource requirements and time taken.

### B. BWA Core

Black wolves follow a hierarchical pack order similar to grey wolves, which offers rank based solutions. GWO is formatted minimally, leaving a lot of room for improvement. The BWA Core improves on the existing optimizer by introducing a parallel niching along with mutation to engulf every local run into a global competition.

*1) Niching Techniques:* At this stage, we extend the architecture to introduce niching which increases competition for the best resources. Difficult classification problems are computationally expensive and as a result parallel niching techniques are introduced. One of the most famous parallel niching techniques is crowding, depicted in Algorithm 5 which introduces deterministic variation.

*2) Elitism over generations:* Elitism counteracts the effects introduced by mutation and promotes survival of the Hall of

**Algorithm 5** Crowding

**Input:** Number of candidate solutions after genetic optimization $n$, number of generations $g$, fitness function $f$
**for** Every generation in $g$ **do**
  **while** $n$ is exhausted **do**
    Randomly select 2 candidate solutions $n_1$ and $n_2$
    Apply crossover and mutation to obtain solutions $m_1$ and $m_2$
    **if** $\delta(\mathrm{n}_1,\mathrm{m}_1) + \delta(\mathrm{n}_2,\mathrm{m}_2) \leq \delta(\mathrm{n}_1,\mathrm{m}_2) + \delta(\mathrm{n}_2,\mathrm{m}_1)$ **then**
      **if** $f(m_1) > f(n_1)$ **then**
        Replace $n_1$ with $m_1$
      **end if**
      **if** $f(m_2) > f(n_2)$ **then**
        Replace $n_2$ with $m_2$
      **end if**
    **else**
      **if** $f(m_2) > f(n_1)$ **then**
        Replace $n_1$ with $m_2$
      **end if**
      **if** $f(m_1) > f(n_2)$ **then**
        Replace $n_2$ with $m_1$
      **end if**
    **end if**
  **end while**
**end for**

**Algorithm 6** Elistism

**Input:** Candidate set $S$, threshold value $\lambda$, and fitness function $f$
**for** Each solution in $S$ **do**
  Obtain fitness value for solution
  **if** Score exceeds $\lambda$ **then**
    Consider solution as elite
  **end if**
**end for**

TABLE II: BENCHMARK FUNCTIONS

| Function | ID | Formula | Modality | Range | Minimum Value |
|---|---|---|---|---|---|
| Sphere | F1 | $\sum_{i=1}^{n}(x_i^2)$ | Multi | [-100,100] | 0 |
| Styblinski-Tang | F2 | $\frac{\sum_{i=1}^{n} x_i^4 - 16x_i^2 + 5x_i}{2}$ | Multi | [-5,5] | -78.332 |
| Three-hump Camel | F3 | $2x^2 - 1.05x^4 + \frac{x^6}{6} + xy + y^3$ | Multi | [-5,5] | 0 |
| Matyas | F4 | $0.26(x_2 + y_2) - 0.48xy$ | Uni | [-10,10] | 0 |
| Brent | F5 | $(x+10)^2 + (y+10)^2 + e^{-(x^2+y^2)}$ | Uni | [-10,10] | 0 |
| Booth | F6 | $(x+2y-7)^2 + (2x+y-5)^2$ | Uni | [-10,10] | 0 |

Fame (HOF) members. HOF members represent the globally best solution. Elitism improves the performance by conserving the best fits encountered so far, decreasing the overall time taken by avoiding pitfalls in the genetic flow. This process is described in Algorithm 6.

*3) Elemental Functionality of BWA:* At this stage, the best candidate solutions have been narrowed down through the pipeline. The candidates are now put through a GWO with a low mutation range for finally selecting the global optimum. This step is displayed in Fig. 4. The overall process can be understood with the following analogy. Consider a territorial distribution of wolf packs across a landmass with a single reward location. The genetically modified algorithm, along with niching and elitism, selects the best wolf from each pack. The final GWO stage makes the selected wolves compete against each other for the final treasure. This ensures global competition leading to increased assurance of guaranteed solution, increased parallelism, prevention of premature convergence, and automated hyperparameter tuning.

## IV. EXPERIMENTAL SETUP AND RESULT

The entire BWA process is computationally expensive and time-consuming due to the repeated testing of the candidates.

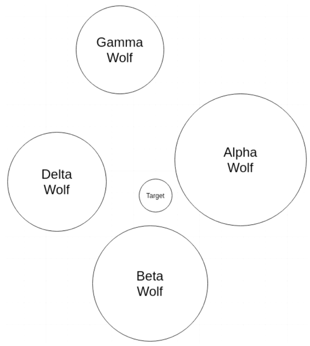


Fig. 4: Final GWO step of BWA

TABLE III: REAL-WORLD DATSETS

| Dataset | ID | Source | Number of features | Number of Attributes |
|---|---|---|---|---|
| Oncological Cohort | D1 | HealthCare Institute | 97 | 16 |
| Breast Cancer | D2 | UCI | 570 | 32 |
| Parkinsons | D3 | UCI | 197 | 23 |
| Liver Cancer | D4 | Kaggle | 345 | 7 |

We could exploit the algorithmic level parallelism of the individual components of BWA and have accelerated them on HPC systems enabled with multiple GPUs. The results obtained from BWA have been compared with 5 other algorithms including GWO [4], Particle Swarm Optimization (PSO) [13], Multiverse Optimization (MVO) [14], Cuckoo Search (CS) [15] and Bat Algorithm (BAT) [16].

### A. Data Description

We have used both classical functions and real-world datasets to demonstrate BWA. This forked approach helps prove BWA's competence and it's use with real-world data. We have considered six benchmark functions to compare the performance of BWA with the other algorithms. We have further divided these functions into unimodal and multimodal functions to test the algorithms, as described in Table $II$. The artificial landscapes provided by these functions evaluate the general performance of an algorithm when presented with different scenarios.

Benchmark functions provide a theoretical proof of an algorithm's competence. Application to real-world datasets provides conclusive proof of BWA's capabilities. We have used the data from a cohort study conducted in collaboration with a healthcare institute, with a cohort size of 100 oncology patients. Based on the imaging, we could perform the Radiomics based feature extraction and preprocessing, which resulted in an initial set of radiomics features, on which we have applied the BWA and other methods. Also, we have collected data from trusted sources like the UCI Machine Learning repository and Kaggle for an extensive study with elaborated data sets, as listed in Table $III$.

### B. BWA Performance Model

The BWA is modeled over a pipelined datapath consisting of $N_{Kernels}$ number of CUDA kernels, with adequate buffering. This section provides a performance model for the BWA on GPUs. The constants and definitions are listed in Table IV.

The BWA Functional Model can be represented as:

TABLE IV: CONSTANTS AND DEFINITIONS FOR BWA PERFORMANCE MODEL

| Symbols | Description |
|---|---|
| $N_{Kernels}$ | No. of kernels in BWA |
| $IP_{BWA}$ | Total Application Data for BWA |
| $B_{Size}$ | Batch Size |
| $C_{Size}$ | Data chunk size transferred from CPU to GPU for one iteration of execution. |
| $N_{Batch}$ | Total number of batches over which GPU process $IP_{BWA}$ amount of data |
| $N\_C_k$ | No. of cycles for kernel k |
| $N\_B_k$ | No. of thread blocks per SM for kernel k |
| $N\_W_k$ | No. of warps per thread block for kernel k |
| $N\_T_{wgk}$ | No. of threads per thread block for kernel k |
| $N\_T_{wfk}$ | No. of threads per warp for kernel k |
| $N\_SM_k$ | No. of SMs required for kernel k |
| $N_{SM}$ | No. of SMs in the GPU |
| $N_{CSM}$ | No. of cores per SM in the GPU |
| $N\_Cores_k$ | No. of cores required for kernel k, and $N\_Cores_k = N\_SM_k \times N_{CSM}$ |
| $P\_depth$ | Pipeline depth per core |
| $F_{GPU}$ | Clock rate of GPU |
| $N_{CORES}$ | Total no. of cores in GPU, and $N_{CORES} = N_{SM} \times N_{CSM}$ |
| $N\_T_{Total}$ | Total no. of threads for kernel k, and $N\_T_{Total} \gg N_{CORES}$ |
| $T_{Preprocessing}$ | CPU times for preprocessing batch input data |
| $T_{Postprocessing}$ | CPU times for postprocessing batch output data |
| $T_{CPU\_GPU}$ | Time spent to transfer buffer data from CPU to GPU |
| $T_{GPU\_CPU}$ | Time spent to transfer buffer data from GPU to CPU |
| $T_{kernel}$ | Time required for all the kernels to process the data in the current iteration |
| $N_{Batch}$ | Number of batches to cover the total application input data of $IP_{BWA}$ |
| $N_{iter}$ | Total number of iterations required for all $N_{Batch}$ number of input batches |
| $N\_Ops_{BWA}$ | Total number of operations required across all batches of BWA runs |

$$f(BWA) = f(\{k_1, k_2, \dots, k_{N_{Kernels}}\}) \tag{4}$$

For BWA application data, $IP_{BWA}$, a single batch of input is scheduled for execution for a kernel $k$ over $N_{Tx\_Rx}$ iterations.

$$N_{Tx\_Rx} = \frac{B_{Size}}{C_{Size}} \tag{5}$$

$$N_{Batch} = \frac{IP_{BWA}}{C_{Size} \times N_{Tx\_Rx}} \tag{6}$$

Following the parallel computing model, the total time required for a single iteration of inputs to exit the GPU can be given as:

$$T_{Gsi} = T_{Preprocessing} + \sum_{k=1}^{N_{Kernels}} (T_{CPU\_GPU} + T_{kernel} + T_{GPU\_CPU}) + T_{Postprocessing} \tag{7}$$

$$T_{BWA\_Kernel\_time} = \sum_{k=1}^{N_{Kernels}} (T_{kernel_k}) \tag{8}$$

The time taken for executing kernel $k$ is represented by:

$$T\_Exec_k = \frac{N\_C_k}{F_{GPU}} \tag{9}$$

*1) BWA Multi-GPU Model:* The time taken by a single GPU to run $k$ kernels over $N_{Batch}$, is denoted by:

$$T_{BWA\_Single} = \sum_{i=1}^{N_{iter}} (T_{Gsi}) \tag{10}$$

The average time taken for a single computation in BWA is given by:

$$t_{Single\_Op} = \frac{T_{BWA\_Single}}{N\_Ops_{BWA}} \tag{11}$$

The execution time of BWA on $N_{GPU}$ GPUs can now be modeled as:

$$T_{BWA\_Multi} \propto \left\{ t_{Single\_Op} \times \left\lceil \frac{N\_Ops_{BWA}}{N_{GPU}} \right\rceil \right\} \tag{12}$$

Thus, the performance and throughput of BWA has a scaling factor on HPC platforms with increase in the number of GPUs.

## C. Results

All of the six algorithms were initially tested on the benchmark functions with 50 runs and the results are tabulated in Table $V$. The mean and the standard deviation values highlight the algorithm's ability to edge towards the global optima.

As we can observe, BWA outperforms the other algorithms in majority of the runs. It must also be noted that algorithms like BAT and GWO do perform comparatively better than BWA for benchmark functions F3 and F4. Same is observed for the real-world data-sets D2 and D3. BWA, whilst an extremely competitive algorithm, is still not immune to the No Free Lunch theorem. Moreover, the performance of any algorithm is restricted by the data fed into it. The reason as to why BAT and GWO outperform BWA in certain fields can be attributed to the inherent characteristics of the dataset. However, it must be noted that BWA provides great results at each run for each function in a general front.

The existing framework for real-world oncological datasets consists of boosted decision trees and random forests, with average test accuracy levels ranginge from 75-87.5%. The six algorithms are applied to these datasets and the results are formulated in Table $VI$. As we can observe from the consolidated data, BWA outperforms its five competitors. This result was expected due to the strong base provided by GWO and the additional functionalities of genetic algorithms. However, it must also be noted that the time taken by BWA surpasses other algorithms by a factor of 20x-80x.

Exploiting the parallel nature of the algorithm, we ran the GNN component of BWA on accelerator platforms using GPUs. A list of the GPUs along with the accelerated BWA performance is shown in Table $VII$. These performance results are built on an Intel i5-7200U CPU host. We can see that with a single NVIDIA GeForce RTX 2080Ti GPU, the performance scaled by a factor of 85x in comparison with the base CPU run. The more-than-linear BWA performance scalability is evident with an acceleration of 517x using four NVIDIA GeForce RTX 2080Ti GPUs, following the theoretical predictions. It can be observed from table $VII$ that there is still scope for improved memory utilization and scalability with more GPUs.

TABLE V: BENCHMARK RESULTS

| Function | GWO | | PSO | | MVO | | CS | | BAT | | BWA | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Mean | Std | Mean | Std | Mean | Std | Mean | Std | Mean | Std | Mean | Std |
| F1 | 6.59E-07 | 6.34E-05 | 2.29E-06 | 6.94E-05 | 9.7E-06 | 3.87E-05 | 2.48E-08 | 3.67E-06 | 5.78E-06 | 4.90E-05 | 9.55E-07 | 5.67E06 |
| F2 | -79.6845 | 8.5612 | -64.8977 | 19.1565 | -70.3532 | 11.2847 | -91.805 | 60.1456 | -73.1563 | 5.8415 | -75.4213 | 8.2577 |
| F3 | 1.77E-18 | 8.78E-12 | 6.81E-15 | 1.79E-13 | 1.22E-15 | 3.5E-13 | 5.06E-11 | 7.67E-16 | 3.08E-16 | 4.04E-09 | 1.52E-15 | 1.72E-12 |
| F4 | 6.89E-14 | 0.0051 | 4.62E-21 | 6.01E-5 | 8.84E-15 | 9.66E-4 | 8.42E-19 | 0.55 | 5.64E-15 | 0.0042 | 4.53E-25 | 6.32E-05 |
| F5 | 2.58E-32 | 5.23E-07 | 2.43E-31 | 1.15E-04 | 7.23E-28 | 5.12E-03 | 1.52E-20 | 6.44E-05 | 3.81E-35 | 3.73E-03 | 3.59E-31 | 2.15E-05 |
| F6 | 8.43E-21 | 0.0031 | 6.41E-11 | 6.84E-05 | 5.58E-10 | 0.0074 | 3.12E-16 | 0.041 | 4.62E-15 | 5.34E-06 | 7.12E-24 | 0.0048 |

TABLE VI: DATASET METRICS

| Dataset | Accuracy | | | | | | Time Taken | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GWO | PSO | MVO | CS | BAT | BWA | GWO | PSO | MVO | CS | BAT | BWA |
| D1 | 77.54 | 71.24 | 74.67 | 68.15 | 76.48 | 95.84 | 33.12 | 27.15 | 14.85 | 18.74 | 35.18 | 1208.51 |
| D2 | 96.85 | 85.49 | 91.74 | 84.51 | 84.15 | 93.87 | 107.15 | 114.23 | 102.87 | 212.78 | 113.76 | 4152.21 |
| D3 | 91.45 | 88.74 | 94.51 | 95.15 | 92.15 | 89.17 | 76.12 | 65.56 | 43.85 | 68.35 | 49.23 | 2179.94 |
| D4 | 84.21 | 91.51 | 94.25 | 91.64 | 87.34 | 92.57 | 51.24 | 51.74 | 49.67 | 99.84 | 54.18 | 1201.72 |

TABLE VII: GPU BASED METRICS

| GPU | Memory Utilization/GPU | Time Taken by BWA | Improvement Factor |
|---|---|---|---|
| 1X 940MX | 95.501% | 236.574 | 34X |
| 1X GeForce RTX 2080Ti | 91.504% | 103.465 | 85X |
| 2X GeForce RTX 2080Ti | 70.633% | 64.727 | 175X |
| 4X GeForce RTX 2080Ti | 74.659% | 17.211 | 517X |

### D. Conclusion and Future Work

BWA provides a new Pareto front when it comes to the genre of computationally intelligent algorithms. It has the capability of "sniffing out" the global optimum in a constrained multi-objective landscape. By merging GA and SI, BWA obtains the best of both the worlds and uses them as a foundation for a novel architecture.

For any future work, improvements can be made in the GPU memory utilization front as well as explore scalable acceleration. Launching multiple kernels and having complete control over thread deployment and utilization can augment performance. Moreover, the mutated GWO section of BWA can also be accelerated GPUs due to the parallel nature of the algorithm. The social hierarchy of the wolves can be extended to a general population and updated rules can be applied.

## REFERENCES


[1] Chankong V., Haimes Y, Thadathil J., Zionts S. (1985), Multiple Criteria Optimization: A State of the Art Review, Decision Making with Multiple Objectives, Lecture Notes in Economics and Mathematical Systems 242, Edited by Y.Y. Haimes, V. Chankong, SpringerVerlag, 36.

[2] Friedrich T, Hebbinghaus N, Neumann F. Rigorous analyses of simple diversity mechanisms.In: Proceedings of genetic and evolutionary computation conference (GECCO), London, UK,July 2007. p. 1219–1225.

[3] Osyczka, A: Multicriteria Optimization in Engineering, John Wiley, New York 1984.

[4] Seyedali Mirjalili, Seyed Mohammad Mirjalili, Andrew Lewis. Grey Wolf Optimizer, Advances in Engineering Software 69 (2014) 46–61

[5] Apollonio, M., Mattioli, L, Scandura, M. Occurrence of black wolves in the Northern Apennines, Italy. Acta Theriol 49, 281-285 (2004).

[6] Wang J, Hu S (2019) An improved grey wolf optimizer based on differential evolution and elimination mechanism. Sci Rep 9:7181

[7] Jayapriya J, Arock M (2015) A parallel gwo technique for aligning multiple molecular sequences. In: 2015 International conference on advances in computing, communications and informatics (ICACCI). IEEE, pp 210–215

[8] Arora, S., Singh, H., Sharma, M., and Sharma, S., Anand P (2019) A New Hybrid Algorithm Based on Grey Wolf Optimization and Crow Search Algorithm for Unconstrained Function Optimization and Feature Selection. IEEE Access 7:26343–26361, 2019.

[9] Emary, E., Zawbaa, H. M, Hassanien, A. E. (2016). Binary grey wolf optimization approaches for feature selection. Neurocomputing, 172, 371–381.

[10] Singh N, Singh SB (2017) A modified mean grey wolf optimization approach for benchmark and biomedical problems. Evol Bioinform 13:1–28

[11] Matthias Feurer and Frank Hutter. Hyperparameter optimization. In Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, editors, AutoML: Methods, Sytems, Challenges, chapter 1, pages 3-37. Springer, December 2018.

[12] Geoffrey MillerPeter M. ToddShailesh U. Hegde, Designing Neural Networks using Genetic Algorithms. Conference: Proceedings of the 3rd International Conference on Genetic Algorithms, George Mason University, Fairfax, Virginia, USA, June 1989

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

[14] H. Jia, X. Peng, W. Song, C. Lang, Z. Xing and K. Sun, "Hybrid Multiverse Optimization Algorithm With Gravitational Search Algorithm for Multithreshold Color Image Segmentation," in IEEE Access, vol. 7, pp. 44903-44927, 2019, doi: 10.1109/ACCESS.2019.2908653.

[15] C. Zefan and Y. Xiaodong, "Cuckoo search algorithm with deep search," 2017 3rd IEEE International Conference on Computer and Communications (ICCC), Chengdu, 2017, pp. 2241-2246, doi: 10.1109/CompComm.2017.8322934.

[16] D. Singh, R. Salgotra and U. Singh, "A novel modified bat algorithm for global optimization," 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), Coimbatore, 2017, pp. 1-5, doi: 10.1109/ICIIECS.2017.8275904.