Democractic Algorithm: A Novel Approach to Hierarchical Swarm Intelligence

***Abstract*—Computational intelligence has produced nature inspired algorithms as the new and popular problem solving tool in diverse fields such as soft computing, functional optimization, data science and machine learning. These algorithms can be further sub-divided, with swarm intelligence, physics-inspired, social, evolutionary algorithms a few noteworthy. In this paper we present Democratic Algorithm (DA) , a social-construct based algorithm heavily inspired from hierarchy swarm intelligence and real life social establishments. Our proposal extends the existing methodology for hierarchical structures by framing experience into a viable component for future applications. A comprehensive study is conducted with 16 benchmark functions and real-life data to cement the competence of the algorithm with DA providing a markup of 2-7.4X of convergence rate.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Optimization lies at the very core of majority of the problems ranging from machine learning to business planning. The level of importance awarded to optimization is due to the limited computational resources at hand. To compensate for the lack of resources, smarter algorithms are developed on an almost daily basis. These algorithms are capable of working efficiently in a constrained environment.

A. Multi-Objective Optimization Problem Definition

As the name suggests, Multi-Objective Optimization Problems (MOOPs) require simultaneous solutions to multiple single-objective optimization problems each with it's own set of constraints.

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

Equation 1 describes a general MOOP consisting of *M* single-objective functions having *K* equality bounds and *J* inequality bounds [1]. The *n* input vector components x_i are also bounded within a lower ${x_i}^L$ and upper ${x_i}^U$ bound.

Usually MOOPs are solved by decomposing them into multiple single-objective optimization problems. However, the striking difference in this method and directly solving a MOOP lies in the decision space. For a pure MOOP, the multi-dimensional decision space is accompanied with an objective space *Z*. For each solution *x*, there is a point in the objective space denoted by $z = (z_1, z_2, z_3, \ldots, z_M)$. Hence pure MOOPs optimize a vector whereas multiple single-objective optimization problems optimize singular variables.

B. Orthodox Methods of Solving MOOPs

The techniques developed in this section describe classical techniques which are based on mathematical and statistical reasoning. These algorithms are rather simplistic and quite intuitive. Some of the noteworthy techniques are discussed here:

- Weighting Sum Method:
  In this algorithm, the MOOP is decomposed into multiple singular problems and each problem is assigned a weight corresponding to it's respective importance to the global front. The weights must be quantified based on prior information and the singular objective functions must be normalized for proper setting of the weight vector. Equation 2 decomposes the MOOP $F_{\mathrm{m}}(x)$ into *m* single-objective optimization problems and weight vector w_{m}, subject to constraints mentioned in Equation 1.

$$Optimize : F_{\mathrm{m}}(x) = \sum_{m=1}^{M} w_{\mathrm{m}} f_{\mathrm{m}}(x) \tag{2}$$

  - Advantages:
    1) Simple and intuitive
  - Disadvantages:
    1) Required conversion of minmax problems into 1 type
    2) Difficulity in proper weight vector setting
- Weighted Metric Method:
  This technique is a general extension of weighting sum method with a varient of combination of multiple objectives. Equation 3 represents this method where l_p is the distance measure between candidate solution *x* and solution z^*, parameter $p \in [1, \infty]$. For p = 1, weighted metric decomposes into weighting sum method.

$$Optimize : l_{\mathrm{p}}(x) = \left(\sum_{m=1}^{M} w_{\mathrm{m}} |f_{\mathrm{m}}(x) - {z_{\mathrm{m}}}^{*}|^{\mathrm{p}}\right)^{\frac{1}{p}} \tag{3}$$

  - Advantages:
    1) Simple yet generic distance metric
  - Disadvantages:
    1) Prerequistive knowledge of individual minima and maxima of each function is required
- Value Function Method:

In this method, the user must provide a value mapping function $\mathbb{U} : \mathbb{R}^{\mathrm{M}} \rightarrow \mathbb{R}$ for all *M* functions. The optimization process is then reduced to Equation 4, subject to the same constraints as Equation 1

$$Optimize : \mathbb{U}(f(x)) \tag{4}$$

- Advantages:
  1) Ideal when prior information is available
- Disadvantages:
  1) Problem of over-simplified value mapping function

Solving MOOPs using classical methods requires either prior information about the subproblems or assumptions for decomposition. These algorihtms have been put to use in multiple practical applications involving huge sets of internal parameters, which incur an overhead of optimal configuration setting.

II. Modern Techniques for solving MOOPs

In this section, we provide a brief overview of modern techniques which provide a computationally cheaper substitute to classical computation for finding solutions in artificial landscapes.

A. Quantified problems in Classical Computation

The algorithms described in the preceeding section can be classified as either direct search or gradient based search techniques. Direct search algorithms user described constriants are utilized in their raw format to guide the solution. Gradient based search techniques use derivatives of the objective function and constraints to guide the solution. However, both these techniques encouter similar problems as described:

- Result is dependent on selection of initial state
- Tendency to get stuck in local optima
- Lack of genralization due to dependence on problem statement
- Requirement of mathemtical representation
- Inefficient when it comes to non-differentiable or discontinous problems
- Lack of parallelization in majority of the algorithms
- Time incurred is high due to single search guidance system

These problems resulted in the development of new fields in Computational Intelligence, namely Evolutionary Algorithms (EA) and Swarm Intelligence (SI). The main motivation for the development of these algorithms is to alleviate the hurdles faced by classical algorithms.

B. Swarm Intelligence to the Rescue

SI proposes emulating naturally occuring groups and their behaviour for optimal functioning. *Swarm* refers to a group of disorganized individuals working towards the common goal of finding the global optimum. When viewed seperatley, these individuals are particularly ineffective. However when properly instructed, these individuals transform into a centralized, collective and self-organized entity. When deployed accurately, a dynamic search pattern, better than random search, appears. This intelligence and self-organizing technique is called Swarm intelligence and is depicted in Algorithm 1.

Algorithm 1 Swarm Intelligence

```
Input: Swarm population size n, Set of stopping criteria λ_i, fitness function f
Create a Swarm of candidate solutions S of size n
for Each candidate s_i in S do
    Apply f on s_i
end for
Based on selected architecture, establish group dynamics
while Any of λ_i are not met do
    if Predicted Fitness is acceptable then
        Apply algorithms related to the selection phase
    else
        Apply algorithms related to the variation phase
    end if
end while
Predicted solution corresponds to global optimum
```

C. Genetic Modifiers

Under the umbrella of EA, Genetic Algorithms (GA) has emerged as one of the most popular search and optimization techniques. The core of GA lies in the feedback process, which updates the candidate solutions in accordance with 3 main subprocesses. Figure 1 summarizes the entire process of GA.

- Selection: This step involves selecting the candidates having the best fitness score
- Crossover: The characteristics of the parents selected at the selection stage are intermixed to generate better candidates in the next generation
- Mutation: As an add-on to the generational transition, random changes are added to add diversity and possibly improve the candidates

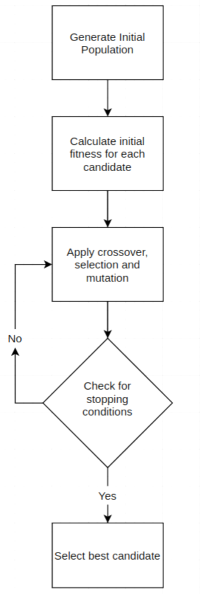

Fig. 1: Genetic Algorithm

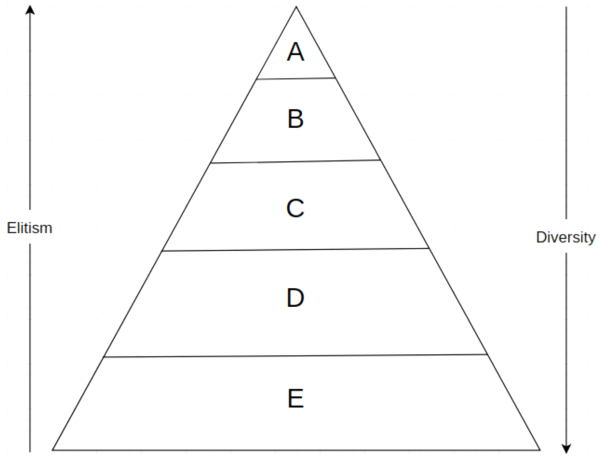

Fig. 2: Structure of DA

III. Democratic Algorithm (DA)

In this section we present DA, a novel social-construct based algorithm heavily inspired from swarm intelligence and real life social establishments.

A. Inspiration

The term "Democracy" was coined in Athens, Greece and literally translates to "Strength of the People". The 2 main characteristics borrowed from the social ideology for this algorithm are as follows.

- Equal opportunity for any candidate to ascend and claim the global decision making spot
- Proportional influence on the decision making process.

We follow an extended social hierarchical structure to incorporate some fundamentals of SI into this algorithm. The initial population set is divided into five departments: A, B, C, D, E and are arranged as shown in figure 2.

The division of departments is based on the relative scores obtained against the set fitness function. Candidates belonging to the higher departments dictate the overall global optima searching process whilst guiding and taking advice from the lower departments. Following initialization, there are two main seasons involved in this algorithm.

- Guiding Season
- Voting Season

In the following subsections, models of the social hierarchy and the seasons are explained, followed by outlining of the entire algorithm.

B. Guiding Season

This phase has the sole purpose of finding the optimal solution. Candidates belonging to the highest department are responsible for finding the solution. The other candidates act as advisers which help maintain the social structure and prevent the search process from falling into local optima. In order to introduce a dynamic factor the search process, the solutions obtained by the top departments are continously stored in a table for any future reference. The updation of the current position is done based a a weighted sum of the current position of all the departments. Based on the difference between the solution provided and the true solution, the top department's reqard score is either incremented or decremented by 1.

Algorithm 2 DA pseudocode

Input: Population size n, random vector set $\vec{r_{1,\mathrm{j}}}$, $\vec{r_{2,\mathrm{j}}} \in [0,1]$, weight vector $\vec{w}$, initial estimate of global optimum X_g, time period T, threshold λ, fitness function f, coefficient vector sets $\vec{A}_\mathrm{j}$, $\vec{C}_\mathrm{j}$ and $\vec{a_\mathrm{j}}$
Set $t = 0$
Initialize vector set $\vec{A}_\mathrm{j} = 2\vec{a} \cdot \vec{r_1} - \vec{a}$
Initialize vector set $\vec{C}_\mathrm{j} = 2\vec{r_2}$
if Table H exists with current function problem **then**
    Refer to table H for solution
else
    Create table H mapping input to corresponding output and apply memoization
    Create table R to store the performance of each department
    for i $\in$ [1,n] **do**
        Generate population results $X_i(t)$ at instance t
        Evaluate each individual against the fitness function
    end for
    Assign A, B, C, D, E departments as per the score against the fitness function.
    Evaluate $\vec{D}_\mathrm{j} = |\vec{C} \cdot \vec{X}_\mathrm{g}(t) - \vec{X}_\mathrm{j}(t)|$
    while t $<$ T **do**
        for Each individual $X_{i,j}(t)$ in each department j **do**
            Update $X_{i,j}(t) := X_j(t) - \vec{A}(\vec{D}_\mathrm{j})$
            $X_{i,j}(t+1) = \frac{\vec{w} \cdot X_{i,j}(t)}{5}$
        end for
        Check fitness of X_{top} against f
        if Fitness score is acceptable **then**
            Increase score of X_{top} in R_{top} by 1
        else
            Decrease score of X_{top} in R_{top} by 1
        end if
    end while
    At $t = T$, check score of top department
    if $score_{top} \geq \lambda$ **then**
        Continue
    else
        Replace X_{top} with X_{top-1}
    end if
    Update $\vec{a} := 2(\frac{T-t}{T})$
    Update $\vec{A}$ and $\vec{C}$ accordingly
    X_{top} represent required set of optimal solutions
    Add corresponding function and solution to H for future reference
end if

C. Voting Season

After a set time period T if the total reward score does not exceed a threshold λ then the top department is replaced by the one just below it. If λ is exceeded, then the next problem is loaded with the current configuration.

The entire process of guiding and voting is displayed in Algorithm 2.

D. Key points

Based on the mathematical representation, the following observations can be made:

- The updation process occurs at an individual and departmental level to save the best solutions obtained so far
- The parameters $\vec{A}$ and $\vec{C}$ assist the solutions to have higher dimensionality

- Due to the dual updation and introduction of a memory element, the problems functions will saturate at a middle level department
- The voting season provides weighted rights to each candidate in each department. The weights are assigned based on the individual's fitness which in turn is dependent on their department's position
- As the department rank increases, the number of individual candidate in the department decreases to promote elitism. This would ensure survival of the best solutions in future problems.
- As the department rank decreases, the number of individual candidate in the department increase to promote diversity. This would open up new avenues to explore and possibly increase chances of finding the gobal optimum.
- The increased number of parameters requires fine tunning which can be automated

To sum up, DA starts with generating and evaluating a set of candidate solutions. The solutions are grouped according to their relative goodness of fit. The top candidates are responsible for finding the optimum solution while the other candidates are required to maintain the hierarchy, provide valuable input to the leaders and contest when the results begin deteriorating. When the top department does not function as per requirement, the lower departments replace it. A memory component is introduced in the form of a table which maps the previously encountered inputs to their corresponding outputs.

IV. Results and Discussion

In this section the DA algorithm is benchmarked against 18 standard minimization functions. These functions are described in Table *I*. The DA was iterated 20 times on each of these functions and the results were compared against other popular SI algorithms such as Particle Swarm Optimization (PSO), Grey Wolf Optimization (GWO), Multiverse Optimization (MVO) and Cuckoo Search (CS). The metric based results such as mean and standard deviation is listed in Table

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.

TABLE I: BENCHMARK FUNCTIONS

Function	ID	Formula	Modality	Range	Minimum Value
Matyas	F1	$0.26(x_2 + y_2) - 0.48xy$	Uni	[-10,10]	0 at x^* = (0,0)
Booth	F2	$(x + 2y - 7)^2 + (2x + y - 5)^2$	Uni	[-10,10]	0 at x^* = (1,3)
Bohachevsky	F3	$x^2 + 2y^2 - 0.3cos(3\pi x) - 0.4cos(4\pi y) + 0.7$	Uni	[-100,100]	0 at x^* = (0,0)
Gramacy and Lee	F4	$\frac{sin(10\pi x)}{2x} + (x - 1)^4$	Uni	[-0.5,2.5]	-0.869011134989500 at x^* = (0.548563444114526)
Leon	F5	$100(y - x^3)^2 + (1 - x)^2$	Uni	[-10,10]	0 at x^* = (1,1)
Ackley	F6	$-20exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - exp(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)) + 20 + e$	Multi	[-32,32]	0 at x^* = (0,0)
Bartels	F7	$\lvert x^2 + y^2 + xy\rvert + \lvert sin(x)\rvert + \lvert cos(y)\rvert$	Multi	[-500,500], [-500,500]	1 at x^* = (0,0)
Branin	F8	$a(x_2 - b{x_1}^2 + cx_1 - r)^2 + s(1 - t)cos(x_1) + s$	Multi	[-5,10], [0,15]	0.397887 at x^* = (-π, 12.2.275)
Egg Crate	F9	$x^2 + y^2 + 25(sin^2(x) + cos^2(x))$	Multi	[-5,5]	0 at x^* = (0,0)
Qing	F10	$\sum_{i=1}^{n}(x^2 - i)^2$	Multi	[-500,500]	0 at x^* = ($\pm\sqrt{i}$)
Rastrigin	F11	$10n + \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i))$	Multi	[-5.12,5.12], [-500,500]	0 at x^* = (0,0)
Rosenbrock	F12	$\sum_{i=1}^{n}[b(x_{i+1} - x_i^2)^2 + (a - x_i)^2]$	Multi	[-5.12,5.12], [-5,10]	0 at x^* = (1)
Bird	F13	$sin(x)e^{(1-cos(y))^2} + cos(y)e^{(1-sin(x))^2} + (x - y)_2$	Multi	[-2π, 2π]	-106.764537 at x^* = (4.70104,3.15294), (-1.58214,-3.13024)
Powell Sum	F14	$\sum_{i=1}^{n}\lvert x_i\rvert^{i+1}$	Multi	[-1,1]	0 at x^* = 0
Schaffer	F15	$0.5 + \frac{sin^2(x^2 - y^2) - 0.5}{(1 + 0.001(x^2 + x^2))^2}$	Multi	[-100,100]	0 at x^* = (0,0)
Shubert	F16	$\prod_{i=1}^{n}\left(\sum_{j=1}^{5} cos((j + 1)x_i + j)\right)$	Multi	[-10,10]	-186.7309
Happy Cat	F17	$\left[\left(\lvert\lvert\mathbf{x}\rvert\rvert^2 - n\right)^2\right]^{\alpha} + \frac{1}{n}\left(\frac{1}{2}\lvert\lvert\mathbf{x}\rvert\rvert^2 + \sum_{i=1}^{n} x_i\right) + \frac{1}{2}$	Multi	[-2,2]	0 at x^* = (-1)
Alpine	F18	$\prod_{i=1}^{n}\sqrt{x_i}sin(x_i)$	Multi	[0,10]	2.808 at x^* = (7.917)

TABLE II: BENCHMARK RESULTS

Function	**PSO**		**GWO**		**MVO**		**CS**		**DA**	
	Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
F1	1.39E-08	6.24E-04	2.41E-06	4.98E-05	5.21E-06	8.23E-08	9.11E-06	4.39E-07	8.71E-06	5.11E-05
F2	4.89E-10	8.79E-04	6.76E-09	5.11E-06	6.18E-07	5.84E-08	9.51E-09	3.84E-07	7.62E-11	2.62E-06
F3										
F4										
F5										
F6										
F7										
F8										
F9										
F10										
F11										
F12										
F13										
F14										
F15										
F16										
F17										
F18										