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 2 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

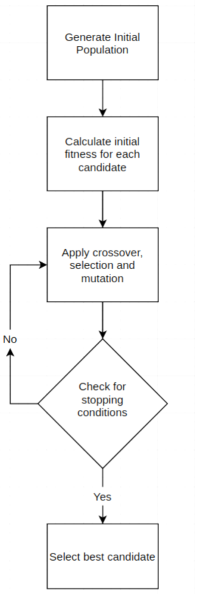

Fig. 1. Genetic Algorithm

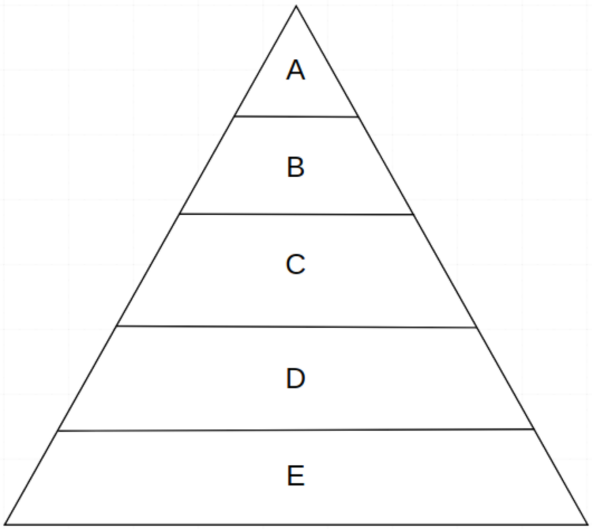

Fig. 2. Structure of DA

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 inflluence on the decision making process.

We follow an extended social hierarchical structure to incorporate some fundamentals of SI into this algorithm. The structure is displayed in figure

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.