Democratic Algorithm (DA):

***Abstract*—Computational intelligence has produced nature-inspired algorithms as the new and popular problem-solving toolset in Multi-Objective Optimization Problems (MOOPs). In this paper, we present Democratic Algorithm (DA), a social-construct-based algorithm which is heavily inspired by hierarchical evolutionary algorithms and cultural algorithms derived from real-life based social establishments. Each candidate in the swarm takes part in the search process and contends for the position of the leader. Additional enhancements such as dynamism and memory retention capabilities boost the performance of DA. A comprehensive study is conducted with 18 benchmark functions to act as a proof-of-concept and lay the groundwork for future works. The results, so obtained, cement the competence of DA as it outperforms other algorithms in multiple unimodal and multimodal landscapes.**

***Index Terms*—Optimization, Swarm intelligence, Democratic Algorithm, Memoization, Dynamic Structure**

I. Introduction

A. Generalized Optimization

Optimization lies at the core of a multitude of problems ranging from financial analysis to the medical research. The parent field of optimization branches out into other sub-fields including, but not restricted to, non-linear programming, optimal control theory, dynamic programming, infinite-dimensional optimization and convex programming. In general, optimization problems are solved with an eye out for either minimizing or maximizing cost function(s), as presented . The history of solving optimization has been graced by scholars such as Fermat and Gauss, who introduced novel methods of solving generalized optimization problems. The well-deserved attention attracted by this field can be attributed to both, the theoretical and algorithmic section as well as its universal practical application-based front.

Optimization problems can be categorized in a multitude of ways, with the most general mode being, division based on the number of independent non-conflicting objective functions; Single Objective Optimzation Problems (SOOPs) and Multi-Objective Optimzation Problems (MOOPs). In this paper, optimization problems are considered without any regard to their classification, unless specified, as class conversion can be performed by either a preference-based approach or a divide-and-conquer technique. For example, MOOPs can be decomposed into multiple SOOPs under specified constraints and likewise SOOPs can be grouped into a collective MOOP by assigning priorities.

B. Approaches to Solving Optimization Problems

Broadly speaking, a solution for a given optimization problem can be obtained in a variety of ways. The two main tracks for solving an optimization problem are the use of traditional methods and application of evolutionary algorithms.

- Tradional methods: These techniques usually convert a MOOP into a set of SOOPs with the help of statistical tools. The optimization problem is then solved by following a set of deterministic transition rules iteratively, parameterized by multiple variables. The sheer simplicity of these algorithms make them appealing to the hoi-polloi. However, a lack of convergence guarentee, sub-optimal problem-algorithm mapping, sensitivity with respect to user parameters and an absense of inherent parallelism, contributed to the emergence of more efficient paths. Table I summarizes some of the infamous methods devised and utilized till date.
- Evolutionary Algorithms (EA) : These algorithms attempt to imitate and approximate the behaviour exhibited by natural entities. The use of multiple search agents reinforced by knowledge propogated through multiple generations, provide assurance of the algorithm's convergence. Moreover, stochastic operators are utilized, instead of deterministic operators, in order to train the algorithm on a general optimization problem. Information exchange and retrival through operations such as mutation and selection, simulate exploitation and exploration, resulting in an Pareto optimal solution set. Table II encapsulates the prominent algorithms used in this track.

C. A Dynammic Programming Extension

Dynamic Programming (DP) is a general programming technique which follows a divide-and-conquer stratergy and aims to find an optimal solution for each of the sub-problems. EAs require a proper representational space defined by the interactions between the possible soltuions with the search space and the effect genetic operators have on the candidate solutions. One possible representation might provide EAs a framework to tackle optimization problems through a dynamic programming outlook. The purpose of providing a DP perspective is to prove that problems which are solvable in polynomial or pseudo-polynomial time by a DP model, can be solved by an EA in the specified time complexity.

II. Notations

This section briefly describes the notations used for describing and discussing the different branches of DA. Relevant tables have been included to summarize the variables utilized in the particular sub-field.

TABLE I: CLASSICAL SOLVING METHODS

Subclass	Optimization Equation	Advantages	Disvantages
Weighting Method [4]	$F_{\mathrm{m}}(x) = \sum_{m=1}^{M} w_{\mathrm{m}} f_{\mathrm{m}}(x)$	Simple, Intuitive	No constraints on weights, Not optimal for nonconvex problems, Interactive
ϵ-Constraint Method [5]	$F_{\mathrm{l}}(x)$ w.r.t $F_{\mathrm{m}}(x) \leq \epsilon_{\mathrm{m}}$	Ensures Pareto optimality , and uniqueness	Not scalable, Range requirement, Interactive
Neutral Compromise Solution	$\frac{F_{\mathrm{m}}(x) - ((z_{\mathrm{i}}^{*} + z_{\mathrm{i}}^{\mathrm{nad}})/2)}{z_{\mathrm{i}}^{\mathrm{nad}} - z_{\mathrm{i}}^{*}}$	Approximate Solution, Rapid Process	Weakly Pareto, Utopian and Nadir solution required
Weighted Metrics [6]	$(\sum_{m=1}^{M} w_{\mathrm{m}} \lvert f_{\mathrm{m}}(x) - z_{\mathrm{m}}^{*} \rvert^{\mathrm{p}})^{\frac{1}{p}}$	Smaller feasible region	Nondifferentiable in some cases, Convexity is required
Value Function Method [7]	$\mathbb{U}(F(x))$	Excellent results when mathematical representation is provided	Mandatory and Over simplified mapping

TABLE II: EVOLUTIONARY METHODS

Subclass	Advantages	Disadvantages
Preference relation [8]	Dominant Pareto guarentee	Required reference point
Light Beam Method [9]	Optimal distance crowding	Reference direction threshold
Imprecise value function [10]	Solution based ranking	Dominant solutions
Biased Crowding	Desired trade-off	Crowding of solutions

A. For Optimization

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

Equation 1 describes a general optimization problem consisting of *M* single-objective functions 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.

Solution analysis must be conducted with respect to the optimization problem to authenticate their validity as in some cases the ideal solution is non-existent.

In order to compare the performance of diffferent algorithms on test functions, the concept of dominance is introduced initially, following which Pareto optimality is discussed. Consider algorithms A and B, each producing solution x_{A} and x_{B} respectively. Solution x_{A} is said to weakly-dominate over solution x_{B} for *M* test functions, *iff* the following conditions are met:

- x_{A} is no worse than x_{B} in all f_{i}, $\forall i \in [1, 2, \ldots, M]$ test functions.
- x_{A} is strictly better than than x_{B} in at least one test function f_{i}, for $i \in [1, 2, \ldots, M]$.

On the other hand, x_{A} is said to strongly-dominate x_{B} for *M* test functions, *iff* x_{A} is strictly better than x_{A} for all *M* test functions.

Now, let us consider the following case; x_{A} dominates x_{B} on f_{i} and x_{B} dominates x_{A} on f_{j}. A set of such solutions are said to be non-dominant with respect to each other and any solution outside this set would be dominated by at least one solution in this set. This subset of non-dominated solutions from the entire set of possible solutions is called a Globally Pareto-optimal set or simply a Pareto-optimal set. The purpose of introducing the optimization-relevant notation in this text is to provide a relative order of dominance among multiple algorithms. Further discussions on the choice of the algorithm can be made by incorporating problem-specific information.

B. Dynamic Programming

Let *X(t)* represent the soltuion state at time *t* and *U(t,X(t))* be the control policy associated to state *X(t)*. Equation 2 represents the state definition at initial time t_0 and at arbritary time $t \in [t_0, t_1, \ldots, T]$, as a function of its preceding state value and control policy. The state at time *t*+1 is stated as a function of the state at time *t* and its corresponding policy *U(t, X(t))*.

$$\begin{aligned} X(t_0) &= x_0 \\ X(t+1) &= f(t, X(t), U(t, X(t))) \end{aligned} \tag{2}$$

Note that reduced notations have been used from this point onwards. Refer to Table III for the corresponding mapping. The control policy updation is performed as per the score of a cost function $J_{t,x}(u)$ and is described in 3.

$$J_{t,x}(u) = \sum_{t_0}^{T-1} F(t, x, u) + \psi(x + W(t)) \tag{3}$$

The cost function defined in Equation 3 forms a part of a stochastic optimal control problem, parameterized by reward function $\psi(x)$, in which randomness being introduced throughout the stages in the form of noise *W(t)*. The most generalized and effective technique for solving any optimization problem, as per DP, is to divide the problem into multiple sub-problems and finding optimal solutions to those sub-problems as shown in Equation 4.

$$J_{t,x}(u) = \begin{cases} j_{t,x,1}(u) \\ j_{t,x,2}(u) \\ . \\ . \\ j_{t,x,n}(u) \end{cases} \tag{4}$$

The control variable *u* also has an element of randomness as it is selected from a set of Markov control policies. Moreover, only an expected value of the functional result of a given policy is achieved. This expected value is computed as a probability measure prompted by the transitions underwent and their corresponding state densities.

$$J_{t,x}(u) = E(\sum_{t_0}^{T-1} F(t,x,u) + \psi(x + W(t))) \tag{5}$$

DP minimizes/maximizes $j_{t,x,i}(u)\ \forall\ \mathrm{i} \in [1, 2, \ldots, n]$, which optimizes the overall cost $J_{t,x}(u)$. The parameter set corresponding to the optimal cost are congruous with the target solution.

C. Evolutionary Algorithms

EAs utilize an analytic approach guided by operations like mutation and crossover. Endogenous parameters like mutation rate and recombination rate dictate the overall behaviour of any evolutionary algorithm. The general parameters are specified in Table IV.

III. Democratic Algorithm (DA)

In this section, we present a novel social-construct based algorithm heavily inspired by swarm intelligence and real life social establishments. To find the optimal solution, each candidate aims to transcend to the next population with the help of the previous generations.

A. Inspiration

The term "Democracy" was coined in Athens, and translates to "Strength of the People". The two 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 N population sets α_{i} for $i \in N$. The case for $N = 3$ is shown in figure 1.

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

- Exploitation Phase
- Exploration Phase

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

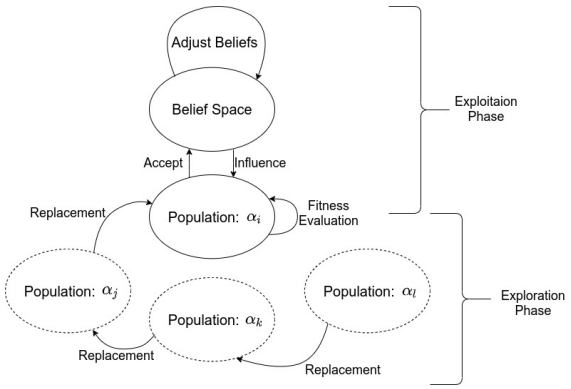

Fig. 1: Structure of DA

Algorithm 1 Exploitation Phase

1: **Input:** Population size of population α_{top}: n, Set of stopping criteria λ_{i}, fitness function f
2: Create n candidate solutions S_i, $i \in [1, n]$
3: **for** Each candidate S_i **do**
4: Apply f on s_i
5: **end for**
6: Establish group dynamics based on fitness hierarchy
7: **while** any of λ_{i} are not met **do**
8: **if** predicted fitness is acceptable **then**
9: Re-emphasize recombination through $\mathcal{R}_{\mathrm{r}}$ and $\mathcal{R}_{\mathrm{o}}$
10: **else**
11: Re-emphasize mutation through $\mathcal{M}_{\mathrm{r}}$ and $\mathcal{M}_{\mathrm{o}}$
12: **end if**
13: **end while**
14: Predicted solution corresponds to global optimum

B. Exploitation Phase

This phase has the sole purpose of finding the optimal solution. Candidates belonging to the popolution corresponding to the decision-making position have the paramount responsibility for finding the solution. The designated survivors act as advisers which help maintain the social structure and prevent the search process from falling into local optima. To improve the throughput of DA, a dynamic extension, in the form of memoization table, is introduced in the exploitation phase. The solutions obtained by the premier population are continuously stored in the table for any future reference. The updation of the current position is done based on a weighted sum of the current position of all the populations. Based on the difference between the solution provided and the true solution, the leading population's reward score is either incremented or decremented by 1. The algorithm is depicted in Algorithm 1.

C. Exploration Phase

After a set period T if the total reward score does not exceed a threshold λ then the leading population is replaced by the popluation inferior to it. If λ is exceeded, then the next problem set is loaded with the current configuration.

The entire process of the voting season is displayed in Algorithm 2.

TABLE III: REDUCED PARAMETER LISTINGS

Symbol	Variable	Symbol	Variable
t_0	Initial Time	x = X(t)	Solution State at time t
T	Final Time	u = U(t,X(t))	Control Policy parameterized at state x
t	Time instant $\in [t_0, \ldots, T]$	$\psi(x)$	Reward Function at time t
W(t)	Gaussian Noise	$J_{t,x}(u)$	Cost Function

TABLE IV: EVOLUTIONARY PARAMETER LISTINGS

Symbol	Variable	Symbol	Variable
μ	Evolutionary Stratergy	S	Initial Seeding
$\mathcal{R}_r$	Recombination Rate	$\mathcal{R}_o$	Recombination Operator
$\mathcal{M}_r$	Mutation Rate	$\mathcal{M}_o$	Mutation Operator
$\mathcal{C}_r$	Crossover Rate	$\mathcal{C}_o$	Crossover Operator

Algorithm 2 Exploration Phase

Input: Population size *n*, random vector set $\vec{r_1}_j$, $\vec{r_2}_j \in [0,1]$, weight vector $\vec{w}$, initial estimate of global optimum X_g, time period *T*, threshold λ, fitness function *f*, coefficient vector sets $\vec{\gamma}_j$, $\vec{\beta}_j$ and $\vec{a}_j$
Set $t = 0$
Initialize vector set $\vec{\gamma}_j = 2\vec{a} \cdot \vec{r_1}$ - $\vec{a}$
Initialize vector set $\vec{\beta}_j = 2\vec{r_2}$
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 population set
    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 $\alpha_i \; \forall \; i \in N$ populations as per the score against the fitness function.
    Evaluate $\vec{D}_j = |\vec{\beta} \cdot \vec{X}_g(t) - \vec{X}_j(t)|$
    while t $<$ T **do**
        for Each individual $X_{i,j}(t)$ in each population *j* **do**
            Update $X_{i,j}(t) := X_j(t)$ - $\vec{\alpha}(\vec{D}_j)$
            $X_{i,j}(t+1) = \frac{\vec{w} \cdot X_{i,j}(t)}{N}$
        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 highest population set
    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 $\vec{\gamma}$ and $\vec{\beta}$ accordingly
    X_{top} represent required set of optimal solutions
    Add corresponding function and solution to *H* for future reference
end if

D. Cultural Algorithm

As stated earlier, DA has its core ideas derived from real-life social establishments and hence falls under the umbrella of Cultural Algorithms (CA). The knowledge-intensive framework of DA extracts the idea of an auto-adjusting cultural system and produces solutions for an extended set of problems. As displayed in Figure 1 the social fabric of interactions between the population and a predefined belief space, is the leading force of a self-coorecting system. The population consists of potential solutions to the problem presented and the belief space acts as a repository of knowledge acquired by the system so far, along with a compendium vector pointing to the target solution. The memoization table H specified in Algorithm 2 is stored in the belief space. Note that CA must note only simulate the strengthening of the fabric, but also be capable of replacing the bond as and when the results are less than satisfactory. In the case of DA, sub-populations are set in place for the succession if and when the replacement criterion is satisfied.

E. Key Points

Based on the algorithmic design, the following observations can be made:

- The updation process occurs at an individual and at a set level to save the best solutions obtained so far
- The parameters $\vec{\gamma}$ and $\vec{\beta}$ assist the solutions to have higher dimensionality
- Due to the updation and introduction of a memory element in the form of a memoization table, the problems functions will saturate at one particular population
- The exploitation phase provides weighted rights to each candidate in each popluation. The weights are assigned based on the individual's fitness which in turn is dependent on their population's position in the hierarchy
- As the population rank increases, the number of individual candidate in the population decreases to promote elitism. This would ensure survival of the best solutions in future problems.
- As the population rank decreases, the number of individual candidate in the population increase to promote diversity. This would open up new avenues to explore and possibly increase chances of finding the gobal optimum.

IV. PROOF OF CONCEPT

A. Proof of Convergence

DA is one instance of Evolutionary Algorithms and follows a similar convergence proof with slight modifications as proposed by G. Rudolph and A. Agapie [11]. They related the choice of evolutionary operations to the positiveness of a variation kernel. If the transition probability from a set of

TABLE V: BENCHMARK FUNCTIONS

Function	ID	Formula	Modality	Range	Minimum Value	Popoulation Set
Matyas	F1	$0.26(x_2+y_2)-0.48xy$	Uni	[-10,10]	0 at x^* = (0,0)	α_{i}
Booth	F2	$(x+2y-7)^2+(2x+y-5)^2$	Uni	[-10,10]	0 at x^* = (1,3)	α_{i}
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)	α_{i}
Gramacy and Lee	F4	$\frac{sin(10\pi x)}{2x}+(x-1)^4$	Uni	[-0.5,2.5]	-0.86 at x^* = (0.54)	α_{i}
Leon	F5	$100(y-x^3)^2$ + $(1-x)^2$	Uni	[-10,10]	0 at x^* = (1,1)	α_{j}
Ackley	F6	$-20exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2})$+ e + 20 - $exp(\frac{1}{n}\sum_{i=1}^{n}cos(2\pi x_i))$	Multi	[-32,32]	0 at x^* = (0,0)	α_{i}
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)	α_{i}
Branin	F8	$a(x_2-bx_1{}^2+cx_1-r)^2$ + $s(1-t)cos(x_1)$ + s	Multi	[-5,10], [0,15]	0.39 at x^* = ($-\pi$, 12.275)	α_{j}
Egg Crate	F9	x^2 + y^2 + $25(sin^2(x)+cos^2(x))$	Multi	[-5,5]	0 at x^* = (0,0)	α_{j}
Qing	F10	$\sum_{i=1}^{n}(x^2-i)^2$	Multi	[-500,500]	0 at x^* = ($\pm\sqrt{i}$)	α_{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)	α_{k}
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)	α_{k}
Bird	F13	$sin(x)e^{(1-cos(y))^2}+cos(y)e^{(1-sin(x))^2}+(x-y)^2$	Multi	[-2π, 2π]	-106.76 at x^* = (4.70,3.15),(-1.58,-3.13)	α_{i}
Powell Sum	F14	$\sum_{i=1}^{n}\lvert x_i\rvert^{i+1}$	Multi	[-1,1]	0 at x^* = 0	α_{i}
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)	α_{j}
Shubert	F16	$\prod_{i=1}^{n}\left(\sum_{j=1}^{5}cos((j+1)x_i+j)\right)$	Multi	[-10,10]	-186.7309	α_{i}
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)	α_{i}
Alpine	F18	$\prod_{i=1}^{n}\sqrt{x_i}sin(x_i)$	Multi	[0,10]	2.80 at x^* = (7.91)	α_{j}

chromosomes to any solution is non-zero positive, then the transition process of the operations have a positive variation kernel. Extending this across with a homogeneous Markov chain having a positive transition matrix, it has been proven the all Pareto-optimal solutions will be the global solution in finite time with probability one. To prove the stochastic convergence, we first need to define a measure for the mapping.

Theorem 1. *If A and B are subsets of a finite population set X the $d(A,B)$ = $\lvert A\cup B\rvert-\lvert A\cap B\rvert$ is a metric on the power set of X*

Proof. Let X = $\{X^1, X^2\ldots, X^{\mathrm{n}}\}$ be the candidates and a_{i} and b_{i} be the candidate and solution incidence vectors.

$$\begin{aligned} d(A,B) &= \sum_{i=1}^{n}(a_{\mathrm{i}}+b_{\mathrm{i}}-2a_{\mathrm{i}}b_{\mathrm{i}}) \\ &= \sum_{i=1}^{n}[(1-b_{\mathrm{i}})a_{\mathrm{i}}+(1-a_{\mathrm{i}})b_{\mathrm{i}}] \\ &= \sum_{i=1}^{n}\lvert a_{\mathrm{i}}-b_{\mathrm{i}}\rvert=\lvert\lvert a-b\rvert\rvert_1 \end{aligned}$$

Therefore the Hamming Distance between the incidence vectors can act as a metric for mapping. A secondary measure utilized in the mapping is $\delta_{\mathrm{B}}(A)$ = $\lvert A\rvert-\lvert A\cap B\rvert$ which counts the number of candidates in set A but not in set B. Stochastic convergence proof is discussed next.

Theorem 2. *Let n_t be the population of DA at time $t\geq 0$ and $F_t = f(n_t)$ be the optimal mapping. DA is said to converge at probability 1 to the entire set if*

$$d(F_t, M^*)\to 0, t\to\infty$$

where convergence to set of minimal elements M^* occurs at probability 1 if

$$\delta_{M^*}(F_t)\to 0$$

with probability 1 as $t\to\infty$

The implication of $d(F_{\mathrm{t}}, M^*)\to 0$ implies $\delta_{\mathrm{M}^*}(F_{\mathrm{t}})\to 0$ extends to the worst case scenario of $n = M^*$. However, if a Markov Chain is maintained dynamically, $\lvert M^*\rvert= 1$ and the limited population size will converge to a critical point. The rate of convergence is another important piece in this discussion.

Theorem 3. *Let $n_{i,t}$, F, GM and λ be the population of the i^{th} department of DA at time t, fitness function, rate of cross-over-mutation and threshold for the memory element respectively. The rate of population progression from department i to j for a given problem can be expressed as*

$$\frac{d(Population_{i\to j})}{dt}\propto f(n_{i,t},\frac{1}{F},GM,\lambda)$$

Proof. Consider $n_{\mathrm{i,t}}$ and F as a combined quantity $Q(i,j,t)$. The distribution of $Q(i,j,t)$ will be binomial as the peak is achived with maximum population and highest fitness score. The probability of candidate c at department i to continue into department j at time $t+1$ will be given by

$$P(j|i,t+1)=\binom{n_{i,t}}{n_{i,t}}GM^{n_{i,t}}$$

Based on this distribution, diffusion equations with initial state p, relative frequency of candidate selection till current department μ and selection strength s can be written as:

$$\frac{\partial\mu(p,t)}{\partial t}=\frac{p(1-p)}{2}\frac{\partial^2\mu(p,t)}{\partial^2 p}+sp(1-p)\frac{\partial mu(p,t)}{\partial mu}$$

Fixing this probability when $t\to\infty$ as

$$u(p)=\lim_{t\to\infty}\partial(p,t)$$

We get the rate of transition of a candidate as a function of the proficiency over time t, $u(p)^t$.

B. Proof of NP-Completeness for Population Formation

The populations $\alpha_{\mathrm{i}}\ \forall\ i\in N$ are divided based on their initial fitness value. Given a set of $\mathcal{T}$ tasks to be completed with $\mathcal{N}$ population sets, parameterized by penalty function p, the population formation problem posists an optimal mapping

between candidate solutions and every task with anadditional constraint that a solution can be assigned to only one task. In order to prove the NP-Completeness of the population problem we shall first place it in the NP space and then reduce a known NP-Complete problem to the population problem.

Proof. Let $X_{n,t}$ be a variable indicating the mapping from task $t \in \mathcal{T}$ to candidate solution $n \in \mathcal{N}$.

$$X_{n,t} = \begin{cases} 1, & \text{if } n \text{ -> } t \\ 0, & \text{otherwise} \end{cases}$$

The problem can then be reduced to an optimization problem, as shown in Equation 6

$$min \sum_{t \in \mathcal{T}} \sum_{n \in \mathcal{N}} p(n,t) X_{n,t} \tag{6}$$

$$\sum_{s \in n} X_{n,t} \geq 1, \forall s \in t, \forall t \in \mathcal{T}$$

$$\sum_{t \in T} X_{n,t} \leq 1, \forall n \in \mathcal{N}$$

Consider the trivial case of cardinality $|\mathcal{T}| = 1$ and $p(n,t) = 1\ \forall n \in \mathcal{N}$. This reduces Equation 6 to 7.

$$min \sum_{n \in \mathcal{N}} X_{n,t} \tag{7}$$

$$\sum_{s \in n} X_{n,t} \geq 1, \forall s \in t \tag{8}$$

Equation 7 is the formulation of minimum set cover problem, which is proven to be NP-Complete. However, this does not prove the population formulation problem to be NP-Complete. We need to find a subset of k solutions which will complete task t. Let $M_{n,t}$ be the mapping from solution $n \in k$ to task t. We have to find a corresponding set $C \subseteq \mathcal{N}$, such that $|C| = k$ and $\cup_{n \in C} M_n = M_t$. The length of the set C and the union check of set C can be verifed in polynomial time. Hence population problem is now in the NP set.

In the minimum set cover problem, we have a universal set U and a set of subsets $S_i \in S$ such that $S_i \subseteq U$. We have to find at most m subsets from S called C, $|C| = m$ such that $\cup_{S' \in C} S' = U$.

By the mapping of S_t to U, $S_i \in S$ to $n \in \mathcal{N}$ and k to m, the problems become equivalent. Hence the population formation problem is NP-Complete. As the next best alternative, the capabilities of EA have been utilized to form the population sets sub-optimally.

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 premier populations 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 popultion does not function as per requirement, the population next-in-line succeeds it. A memory component is introduced in the form of a memoization table that maps the previously encountered inputs to their corresponding outputs.

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] H. Liu, F. Gu and Q. Zhang, "Decomposition of a Multiobjective Optimization Problem Into a Number of Simple Multiobjective Subproblems," in IEEE Transactions on Evolutionary Computation, vol. 18, no. 3, pp. 450-455, June 2014, doi: 10.1109/TEVC.2013.2281533.

[3] K. Deb, K. Miettinen and S. Chaudhuri, "Toward an Estimation of Nadir Objective Vector Using a Hybrid of Evolutionary and Local Search Approaches," in IEEE Transactions on Evolutionary Computation, vol. 14, no. 6, pp. 821-841, Dec. 2010, doi: 10.1109/TEVC.2010.2041667.

[4] Marler, R.T., Arora, J.S. The weighted sum method for multi-objective optimization: new insights. Struct Multidisc Optim 41, 853–862 (2010). https://doi.org/10.1007/s00158-009-0460-7

[5] K. Cooper, S. R. Hunter and K. Nagaraj, "An epsilon-constraint method for integer-ordered bi-objective simulation optimization," 2017 Winter Simulation Conference (WSC), Las Vegas, NV, 2017, pp. 2303-2314, doi: 10.1109/WSC.2017.8247961.

[6] N. Ryu, W. S. Song, Y. Jung and S. Min, "Multi-Objective Topology Optimization of a Magnetic Actuator Using an Adaptive Weight and Tunneling Method," in IEEE Transactions on Magnetics, vol. 55, no. 6, pp. 1-4, June 2019, Art no. 7202504, doi: 10.1109/TMAG.2019.2899893.

[7] A. Sinha, K. Deb, P. Korhonen and J. Wallenius, "Progressively interactive evolutionary multi-objective optimization method using generalized polynomial value functions," IEEE Congress on Evolutionary Computation, Barcelona, 2010, pp. 1-8, doi: 10.1109/CEC.2010.5586278.

[8] J. Kim, J. Han, Y. Kim, S. Choi and E. Kim, "Preference-Based Solution Selection Algorithm for Evolutionary Multiobjective Optimization," in IEEE Transactions on Evolutionary Computation, vol. 16, no. 1, pp. 20-34, Feb. 2012, doi: 10.1109/TEVC.2010.2098412.

[9] S. An, S. Yang and Z. Ren, "Incorporating Light Beam Search in a Vector Normal Boundary Intersection Method for Linear Antenna Array Optimization," in IEEE Transactions on Magnetics, vol. 53, no. 6, pp. 1-4, June 2017, Art no. 7001304, doi: 10.1109/TMAG.2017.2664077.

[10] P. Limbourg and D. E. S. Aponte, "An optimization algorithm for imprecise multi-objective problem functions," 2005 IEEE Congress on Evolutionary Computation, Edinburgh, Scotland, 2005, pp. 459-466 Vol.1, doi: 10.1109/CEC.2005.1554719.

[11] G. Rudolph and A. Agapie, "Convergence properties of some multi-objective evolutionary algorithms," Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512), La Jolla, CA, USA, 2000, pp. 1010-1016 vol.2, doi: 10.1109/CEC.2000.870756.

[12] D. V. Arnold and R. Salomon, "Evolutionary Gradient Search Revisited," in IEEE Transactions on Evolutionary Computation, vol. 11, no. 4, pp. 480-495, Aug. 2007, doi: 10.1109/TEVC.2006.882427.

[13] P. A. Vikhar, "Evolutionary algorithms: A critical review and its future prospects," 2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC), Jalgaon, 2016, pp. 261-265, doi: 10.1109/ICGTSPICC.2016.7955308.

[14] Yan-fei Zhu and Xiong-min Tang, "Overview of swarm intelligence," 2010 International Conference on Computer Application and System Modeling (ICCASM 2010), Taiyuan, 2010, pp. V9-400-V9-403, doi: 10.1109/ICCASM.2010.5623005.

[15] K. F. Man, K. S. Tang and S. Kwong, "Genetic algorithms: concepts and applications [in engineering design]," in IEEE Transactions on Industrial Electronics, vol. 43, no. 5, pp. 519-534, Oct. 1996, doi: 10.1109/41.538609.

[16] W. Abdulal, O. A. Jadaan, A. Jabas, S. Ramachandram, M. Kaiiali and C. R. Rao, "Rank-Based Genetic Algorithm with Limited Iteration for Grid Scheduling," 2009 First ICCICSN, Indore, 2009, pp. 29-34, doi: 10.1109/CICSYN.2009.23.

[17] M. Takahashi and H. Kita, "A crossover operator using independent component analysis for real-coded genetic algorithms," Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546), Seoul, South Korea, 2001, pp. 643-649 vol. 1, doi: 10.1109/CEC.2001.934452.

[18] X. Deng, "Application of Adaptive Genetic Algorithm in Inversion Analysis of Permeability Coefficients," 2008 Second International Conference on Genetic and Evolutionary Computing, Hubei, 2008, pp. 61-65, doi: 10.1109/WGEC.2008.63.

[19] 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.

[20] D. Jitkongchuen, P. Phaidang and P. Pongtawevirat, "Grey wolf optimization algorithm with invasion-based migration operation," 2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS), Okayama, 2016, pp. 1-5, doi: 10.1109/ICIS.2016.7550769.

[21] H. Jia, X. Peng, W. Song, C. Lang, Z. Xing and K. Sun, "Multiverse Optimization Algorithm Based on Lévy Flight Improvement for Multithreshold Color Image Segmentation," in IEEE Access, vol. 7, pp. 32805-32844, 2019, doi: 10.1109/ACCESS.2019.2903345.

[22] M. Naik, M. R. Nath, A. Wunnava, S. Sahany and R. Panda, "A new adaptive Cuckoo search algorithm," 2015 IEEE 2nd International Conference on Recent Trends in Information Systems (ReTIS), Kolkata, 2015, pp. 1-5, doi: 10.1109/ReTIS.2015.7232842.

[23] R. W. Garden and A. P. Engelbrecht, "Analysis and classification of optimisation benchmark functions and benchmark suites," 2014 IEEE Congress on Evolutionary Computation (CEC), Beijing, 2014, pp. 1641-1649, doi: 10.1109/CEC.2014.6900240.