Title: Velocity-position updation algorithm based on Multi-Particle Swarm Optimization (MOPSO)

# **Abstract:**

In this paper, we propose a new approach to raise the performance of multi-objective particle swarm optimization. The personal guide and global guide are updated using three kinds of knowledge extracted from the population based on cultural algorithms. An epsilon domination criterion has been employed to enhance the convergence and diversity of the approximate Pareto front. Moreover, a simple polynomial mutation operator has been applied to both the population and the non-dominated archive.

Our initial parameter values are randomly generated and we continue with multiple generations to validate our previous tests against newer cultures. Here, we deal with velocity and position updation techniques based on Pareto non-dominated schemes.

Multi-objective optimization is a class of problems with solutions that can be evaluated along two or more incomparable or conflicting objectives. These types of problems differ from standard optimization problems in that the end result is not a single “best solution” but rather a set of alternatives, where foreach member of the set, no other solution is completely better (the Pareto set). Multi-objective optimization problems occur in many different real-world domains, such as architecture (stability vs. cost), and automobile design (performance vs. fuel efficiency), and as such are a very important problem domain.

Our approach is based on guiding a flock of birds or swarm of fish to correct destination based on global and test parameters.

This paper introduces a proposal to extend the heuristic called “particle swarm optimization” (PSO) to deal with multi-objective optimization problems. Our approach uses the concept of Pareto dominance to determine the flight direction of a particle and it maintains previously found nondominated vectors in a global repository(here we have opted for randomly seeded vectors in a NumPy array) that is later used by other particles to guide their own flight. The approach is validated using several standard test functions from the specialized literature. Our results indicate that our approach is highly competitive with current evolutionary multi-objective optimization techniques.

The Combinatorial problems are real world decision making problem with discrete and disjunctive choices. When these decision making problems involve more than one conflicting objective and constraint, it turns the polynomial time problem into NP-hard. Thus, the straight forward approaches to solve multi-objective problems would not give an optimal solution. In such case evolutionary based meta-heuristic approaches are found suitable. In this paper, a novel particle swarm optimization based meta-heuristic algorithm is presented to solve multi-objective combinatorial optimization problems. Here a mapping method is considered to convert the binary and discrete values (solution encoded as particles) to a continuous domain and update it using the velocity and position update equation of particle swarm optimization to find new set of solutions in continuous domain and de-map it to discrete values. The performance of the algorithm is compared with other evolutionary strategy like SPEA and NSGA-II on pseudo-Boolean discrete problems and multi-objective 0/1 knapsack problem. The experimental results confirmed the better performance of combinatorial particle swarm optimization algorithm.

**Mathematical model & Methods:**

A multi-objective optimization problem can be stated in the general form :

Minimize/maximize fm(x),m =1,2,… M

subject to

gj (x)>=0 j=1,2,..J;

hk (x)=0 k=1,2,…K;

xi (l) < = xi <= xi(U)  i=1,2,…n

A solution *x* is a vector of n decision variables: .

A MOP global minimum problem with n objectives is defined as:

min f(*x*) = [f1(*x*), f2(*x*), . . . , fn(*x*)]

s.t.g(*x*) ≤ 0, h(*x*) = 0

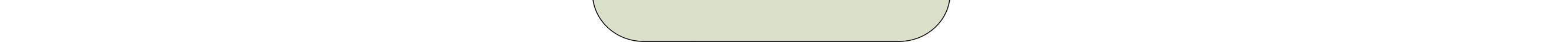
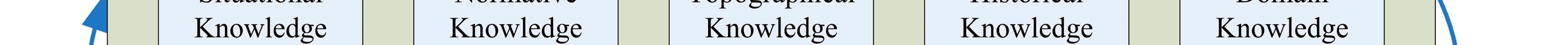
where g(*x*) ≤ 0 and h(*x*) = 0 represent the constraints.

**Definition 1.** *(Pareto dominance)* [26]*: x is said to dominate y (denoted as x < y) if* ∀*i, fi(x)* ≤ *fi(y),* ∃*i, fi(x) < fi(y).*

**Definition 2.** *(Pareto optimal set) [26]: P\* = {x*|¬∃*y* ∈ Ω*, f(y)* ≺ *f(x)}.*

**Definition 3.** *(Pareto front) [26]: PF\* = {f(x) = [f1(x), f2(x),* . . . *, fn(x)]|x*∈*P\*}.*

**Definition 4.** *(ε-Pareto dominance) [17]: x is said to ε-dominate y, denoted as x* ≺ *εy, if f(x)* · *(1 + ε)* ≺ *f(y).*



**Figure 1.** The framework of a cultural algorithm.

When tackling multi-objective problems however, a few modifications must be made. First, the objective is to find not one “global best” solution, but a set of solutions comprising the Pareto Front. To do this, an *archive* of non-dominated solutions is kept, where all non-dominated solutions found at each iteration are stored.

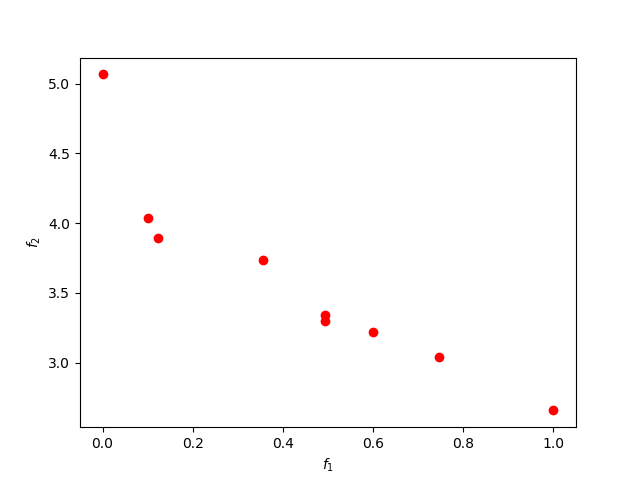
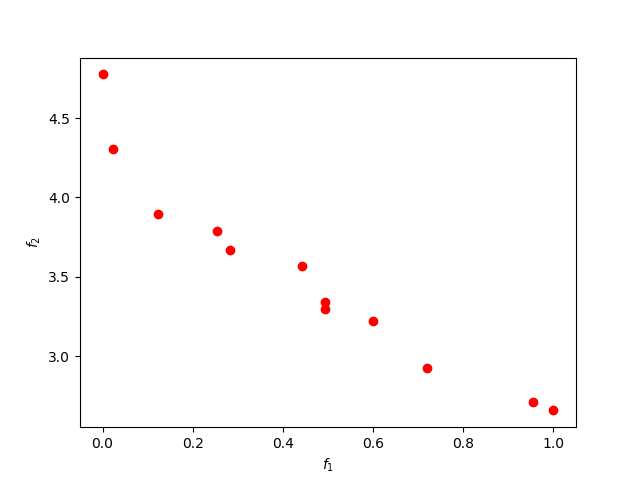
The MOPSO algorithm steps are:

1. Initialize the swarm & archive
2. No. of generation (g):=read the value from user or provide default
3. No. of simulations (s):=no. of generations \* (given value)
4. For each generation initialize the swarm of particles in form of a NumPy vector array
5. Assign upper bound to 1.0, and lower bound to 0.0
6. Assign incrementing value to 0.1 units
7. For each particle in the swarm:
   1. Select leader from the archive
   2. Update velocity
   3. Update position
   4. Plot the graph in a MatPlotLib.Pyplot in Python to simulate the process
8. Update the archive of non-dominated solutions
9. Repeat within each generation
10. After end of each generation, re-initialize the particles or anneal existing
11. Repeat for each generation

Now we use these above steps for n no. of generations just increasing the limit on culture samples in each generation (iteration of sample space).

**Example ILLUSTRATION:**

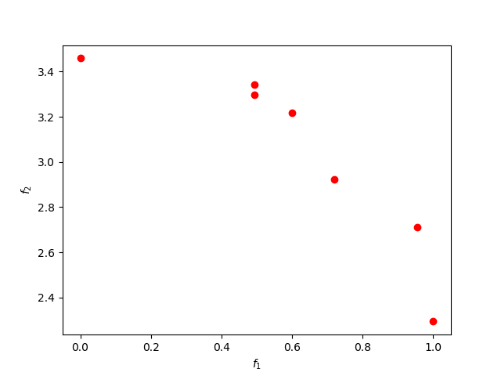
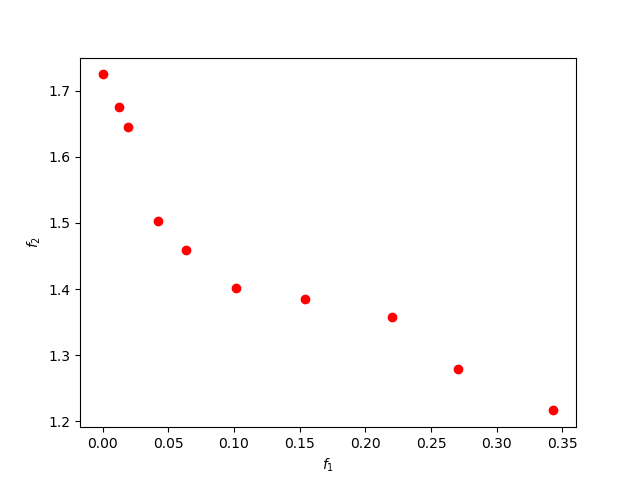
**For generation 1:**

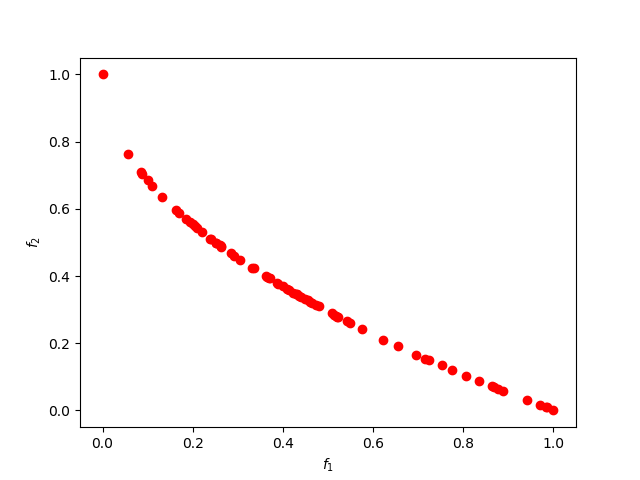
 

**Initial Seed** **Final result of generation 1**

Now as we iterate through the generations we will see the impact of dominated particles over non-dominated ones and, thus the model becomes more accurate.

**For generation 2:**

Our results at the end of spawning 350 samples and 12 generations look like: 

In demonstrations, each point represents a particle iterated as a sample in the entire sample space.

**References:**

Zhang, Q.; Li, H. MOEA/D: A multi-Objective evolutionary algorithm based on decomposition. *IEEE Trans. Evol. Comput.* **2007**, *11*, 712–731

Daneshyari, M.; Yen, G.G. Constrained Multiple-Swarm Particle Swarm Optimization within a Cultural Framework. *IEEE Trans. Syst. Man Cybern.* **2012**, *42*, 475–490

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197, 2002.