Department of Electronics and Computer Engineering
Institute of Engineering, Pulchowk Campus, TU, Nepal
MSc in Computer System and Knowledge Engineering Program

Multi-Objective Optimization Using Membrane Inspired Evolutionary Algorithm

Student:
Bishwo Prasad Lamichhane (070MSCS655)
Department of Electronics and Computer Engineering,
IOE Pulchowk Campus, TU

Supervisor:
Sanjeeb Prasad Pandey, Ph.D.
Department of Electronics and Computer Engineering,
IOE Pulchowk Campus, TU

Date: April, 2023

Outline

- Introduction
- Statement of the Problem
- Objectives
- Literature Review
- Methodology
- Results
- Conclusions and future works
- Thesis Timeline

Introduction

- The goal of this research is to use the computational model proposed by Membrane Computing to
  - solve multi-objective optimization problems (MOOPs) and
  - compare quality of solutions and performance with standard multi-objective optimization algorithms.
- Standard test functions for multi-objective optimization is used
  - to develop the algorithm and
  - to compare quality of solutions and performance.

Statement of the Problem

- Current bio-inspired multi-objective optimization algorithms
  - start with random candidates/population across the entire decision space
  - perform some fitness evaluations, and
  - create next generation of population using features of candidates with better fitness values.
- Partitioning of decision space into hyper volumes can
  - enable parallel execution of evolutionary steps resulting faster convergence
  - computing model proposed by membrane computing can be utilized to achieve that goal

Statement of the Problem (contd.)

- The offspring generation process involves selecting
  - parent populations based on fitness value and other parameters
  - parents can come from different regions of the decision space
  - which may have differently shaped optimum spaces

- Partitioning the decision space and performing evolutionary steps within regions
  - results in better offspring
  - reduces the number of generations needed to achieve results closer to the true Pareto-Front.

Objectives

- To create evolutionary algorithm
  - capable of solving multi-objective optimization problem by,
  - utilizing computational model proposed by membrane computing.

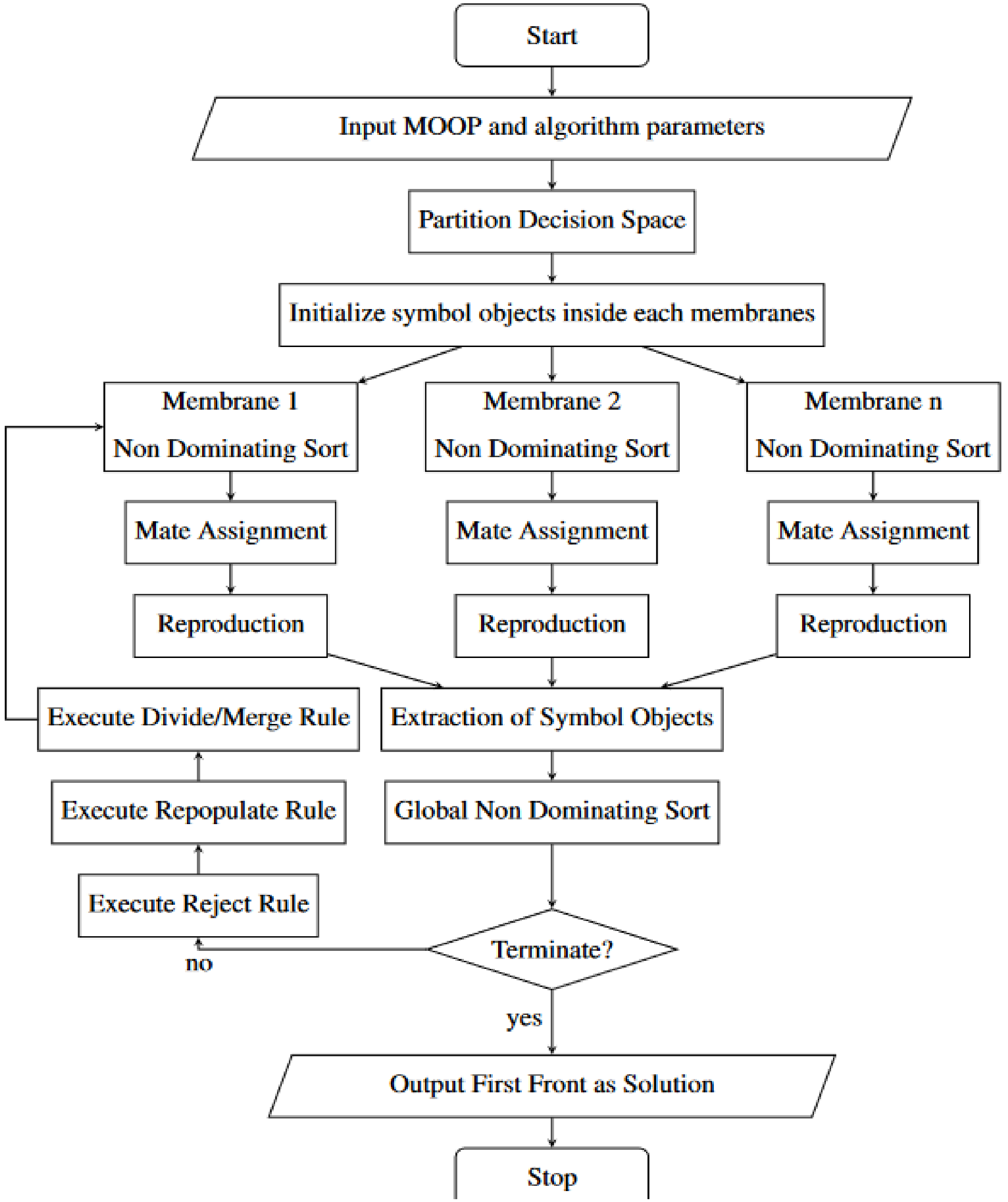
- Analyze the performance of developed algorithm with following benchmark MOOP functions
  - Zitzler–Deb–Thiele's Function 1 (ZDT1)
  - Zitzler–Deb–Thiele's Function 2 (ZDT2)
  - Zitzler–Deb–Thiele's Function 3 (ZDT3)

- Compare the performance of developed algorithm with following standard MOOP solving algorithms with following metrics
  - Generational Distance (GD)
  - Inverted Generational Distance (IGD)

Methodology - Flowchart

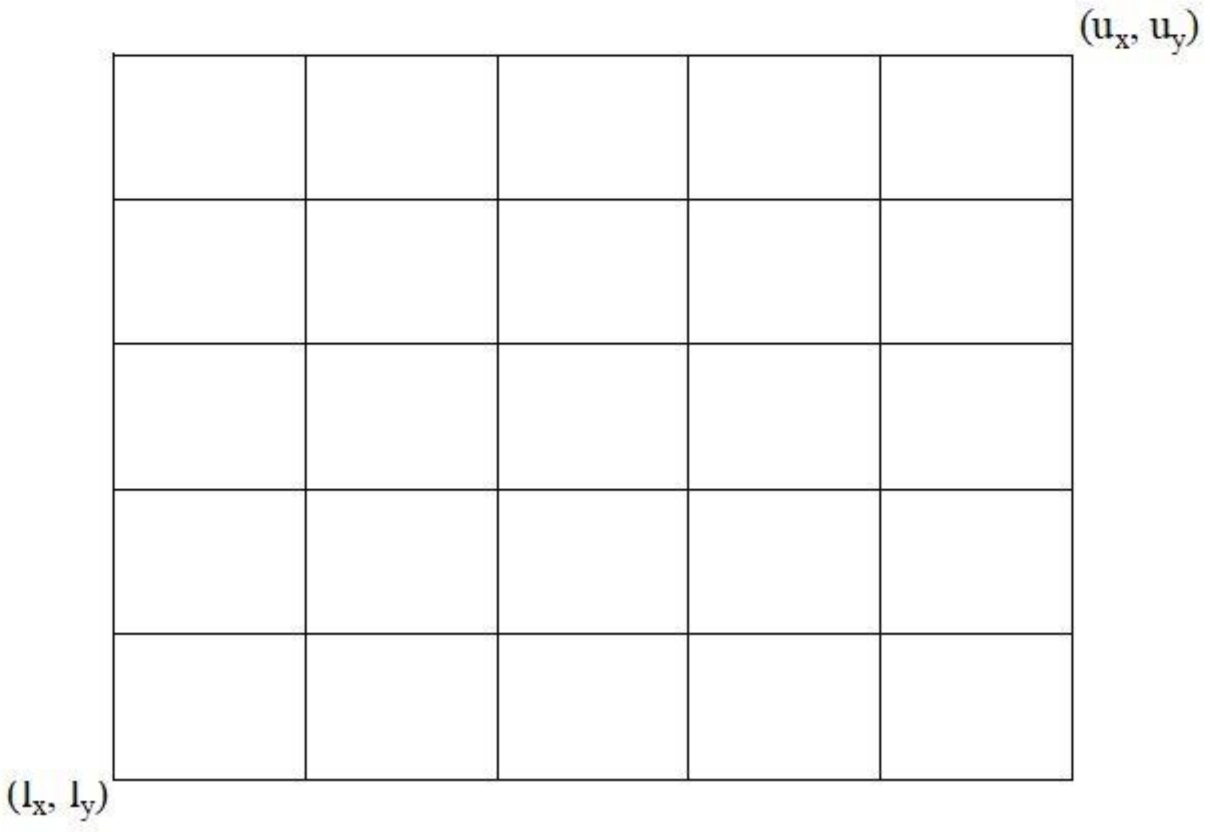
Following are the steps of algorithm :

1. Partitioning of decision space
2. Initialization
3. Non-dominating sort inside membrane
4. Mate Assignment
5. Reproduction
6. Global non-dominating sort
7. Reject Rule
8. Repopulate Rule
9. Divide and Merge Rule



Partitioning of decision space

- For an optimization problem with
  - $\boldsymbol{d}$ number of decision variables and
  - $\boldsymbol{n}$ number of partitions per dimension;
  - total number of membranes will be $\boldsymbol{n^d}$.
- The figure represents a two dimensional decision space with 5 partitions per dimension so that number of partitions will be $5^2 = 25$.
- $\boldsymbol{l_x}$, $\boldsymbol{l_y}$ are the lower bound and
- $\boldsymbol{u_x}$, $\boldsymbol{u_y}$ are upper bound for the decision variables $\boldsymbol{x}$ and $\boldsymbol{y}$



Initialization

- In this step symbol objects are initialized inside each membrane.
- Symbol objects are of the format

{
    'id' : symbolObjectId,
    'parentMembrane' : parentMembraneId,
    'coordinate' : $[x_1, x_2, x_3, \ldots, x_n]$,
    'objectives' : $[f_1, f_2, f_3, \ldots, f_m]$,
}

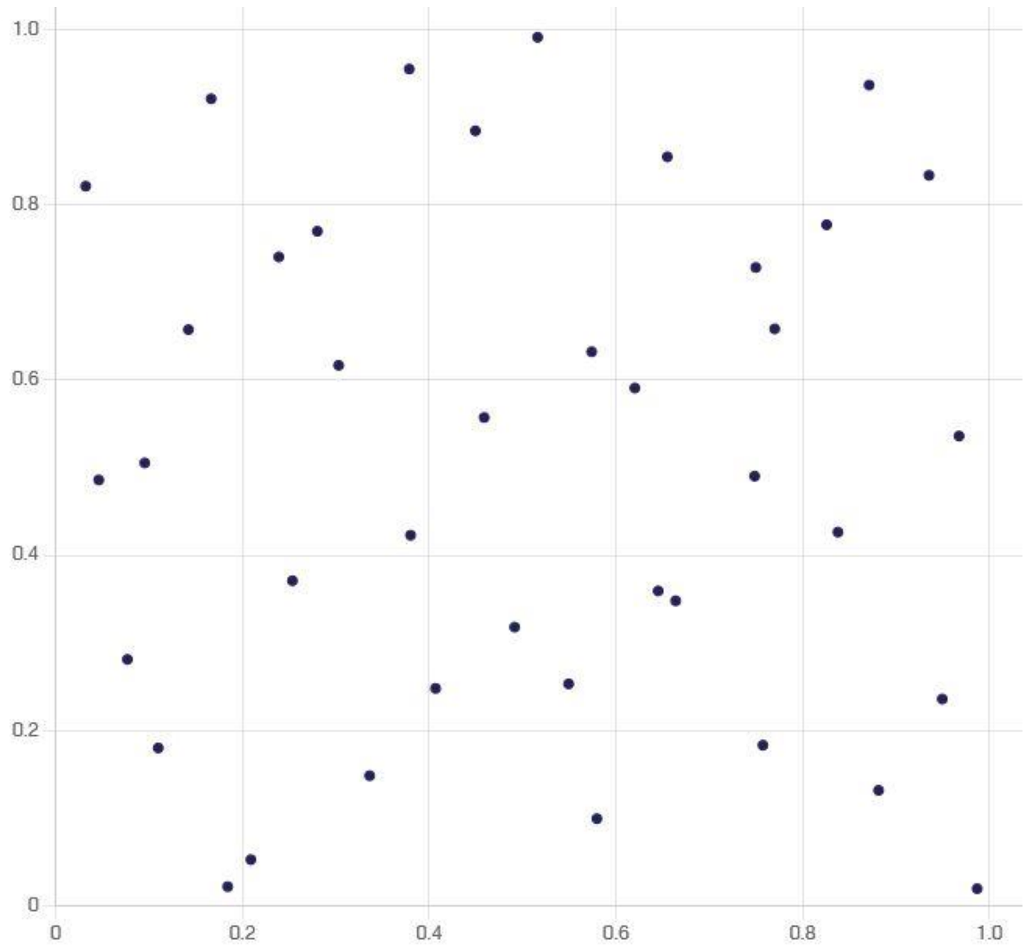
where,

n is number of decision variables

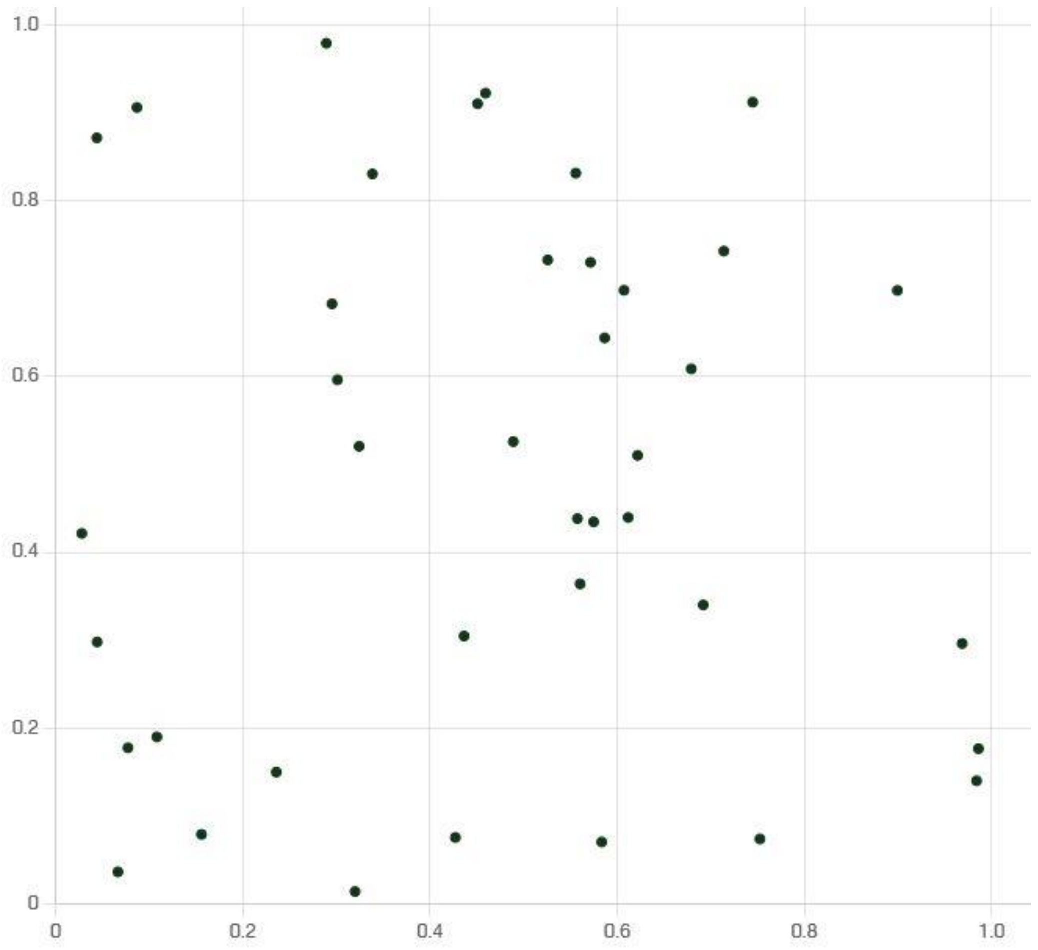
m is number of objective functions

- Latin Hypercube Sampling(LHS) is used to generate well distributed initial solutions.

Initialization (contd.)



Latin Hypercube Sampling



Random Sampling

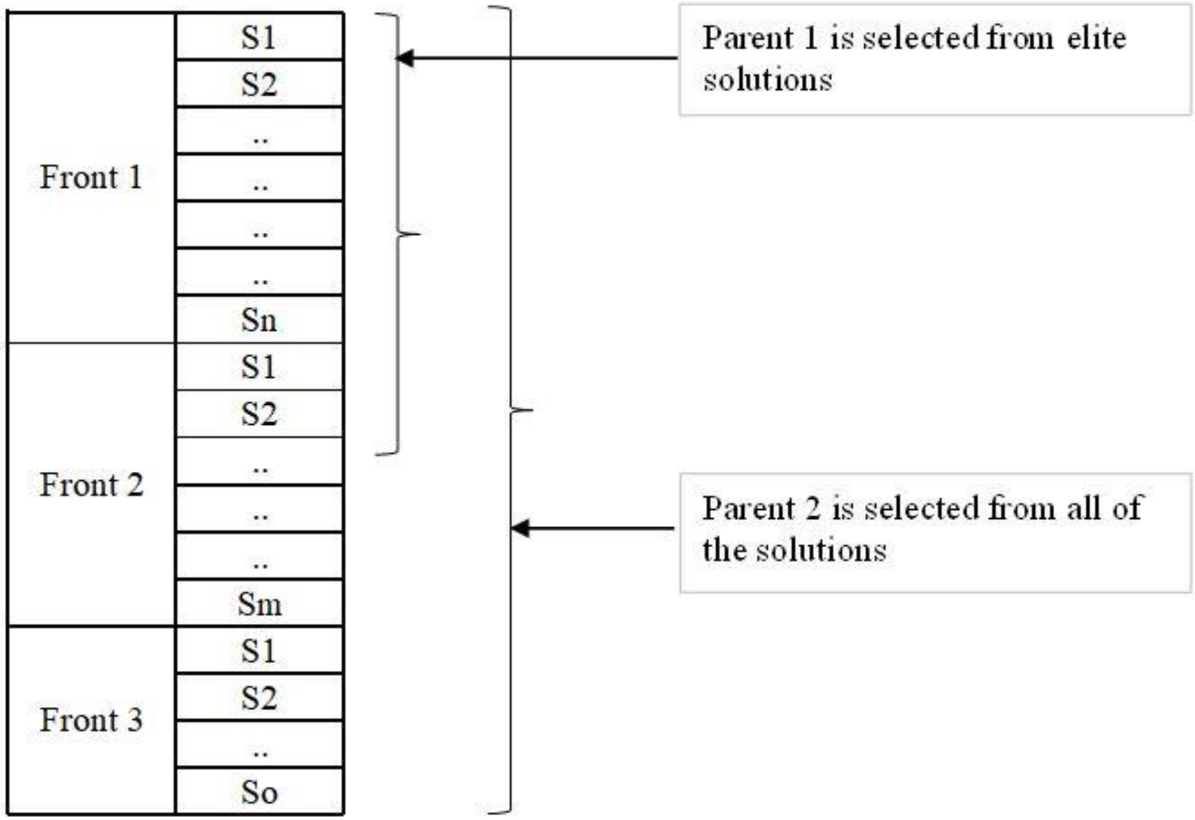
Non-dominating sort inside membrane

- Non-dominating sort is performed inside each membrane.
- Each membrane will have its own Pareto fronts.
- The figure shows an example of Pareto fronts inside each membrane.
- This step is independent of the operations within other partitions so has been done in parallel

Front 1	S1
	S2
	..
	..
	..
	..
	Sn
Front 2	S1
	S2
	..
	..
	..
	Sm
Front 3	S1
	S2
	..
	So

Mate Assignment

- Sorted symbol objects are divided into two parts
  - elite solutions and
  - non-elite solutions.
- A parent pair is generated by selecting
  - one from the elite solutions and
  - one from all of the solutions (except the first parent).
- Mate assignment process is executed in all membranes.
- Mate selection process in repeated until number of pairs are equal to number of symbol objects.



Reproduction

- An offspring is created by combining the features of parent pair
  - If one of the parent is dominating other the offspring gets more characteristics from dominating parent.
  - Otherwise offspring gets equal characteristics from both parents.
- Small mutation is also introduced into offspring.
- To generate next generation of symbol objects :

Given,

Mutation Chance = m

Elite Parent weight = w_e, this is one of the algorithm parameter and must be ≥ 0.5

Parent 1 Weight : w_1

Parent 2 Weight : w_2

Reproduction

- Then

$$w_1 = \begin{cases} 0.5 + \text{Rand}(-m, +m), & \text{if Parent 1 and Parent 2 are in same front} \\ w_e + \text{Rand}(-m, +m), & \text{if Parent 1 dominates Parent 2} \end{cases}$$

$$w_2 = 1 - w_1$$

$$x_{n+1} = w_1 * x_{1n} + w_2 * x_{2n}$$

where,

x_{1n} is decision variable value of parent 1

x_{2n} is decision variable value of parent 2

Global non-dominating sort

- A non-dominating sort of all of the symbol objects is performed.
- First pareto front of this step contains the best solutions up to this iteration.
- Optimization can be stopped at this stage if termination condition is met.

Reject and Repopulate Rule

- In this step half of the low ranking solutions (solution in higher fronts) are rejected.
- If a front is being partially rejected, then symbol objects in this front are rejected in such a manner such that
  - Every membrane gets similar number of symbol objects as much as possible.
  - This will ensure diversity of solutions.
- Accepted symbol objects are sent into their corresponding parent membranes.

Divide and Merge Rule

- Membranes may contain more or less symbol objects than at the start.
  - If a membrane contains more symbol objects, a division rule is applied to it.
  - If a membrane contains less symbol objects, it is merged with its neighbor.

- The division threshold is set at 150%, while the merge threshold is set at 50%.
- At the end of this step,
  - the number of membranes and symbol objects are the same as at initialization, but
  - the number of symbol objects inside each membrane is between the 50% and 150%

- A membrane is divided into two partitions,
  - by assigning symbol objects to the new membranes and
  - dividing them equally between elite and non-elite solutions

Divide and Merge Rule (contd.)

- At the end of this step number of membranes and number of symbol objects are same as they are at the initialization step.
- But number of symbol objects inside are between 50% to 150%.
- Repeat from "Non-dominating sort inside membrane" step until termination condition is reached.

Performance Indicators

- At the end of this step number of membranes and number of symbol objects are same as they are at the initialization step.
- But number of symbol objects inside are between 50% to 150%.
- Repeat from "Non-dominating sort inside membrane" step until termination condition is reached.

Output

- Implementation of the algorithm is done in PHP 8.2 programming language.
- A testing application is also created to check the performance of algorithm with different parameters.
- Program for
  - visualization of solutions in decision space,
  - visualization of objective values in objective space and
  - visualization of pareto fronts from stored details of algorithm execution

  is also created.
- Performance metrics are calculated with open source “pymoo” package written in Python language.

Output – Algorithm Test Application

Use the form below to:

- Change the parameters of algorithm.
- Select standard MOOP objective to optimize.

Algorithm Parameters

Initialization Method :	Latin Hypercube Initialization
Elite Mate Percentage :	--Default (50)--
Mutation Rate :	--Default (0.05)--
Merge/Divide Threshold :	--Default (50/150)--
Hyper Cube Dimension :	--Same as decision dimension (MAX : 5)--
Partition per dimension :	4
Symbol objects per membrane :	20

Objective Function

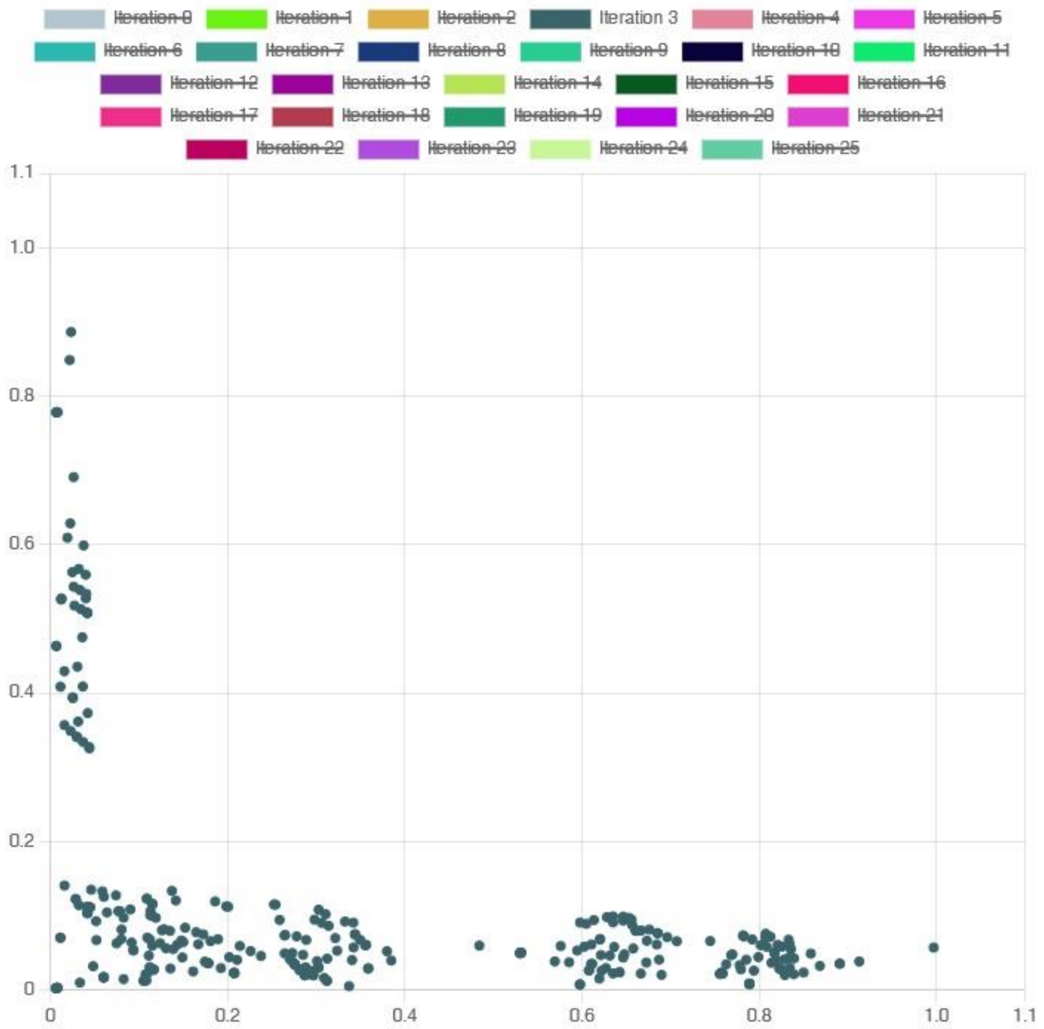
Objective to optimize :	zdt1
No of Decision Variables (Decision space Dimension) :	2
Lower Bound :	0
Upper Bound :	1
Iterations :	25

Submit

Output – Symbol Object Visualization

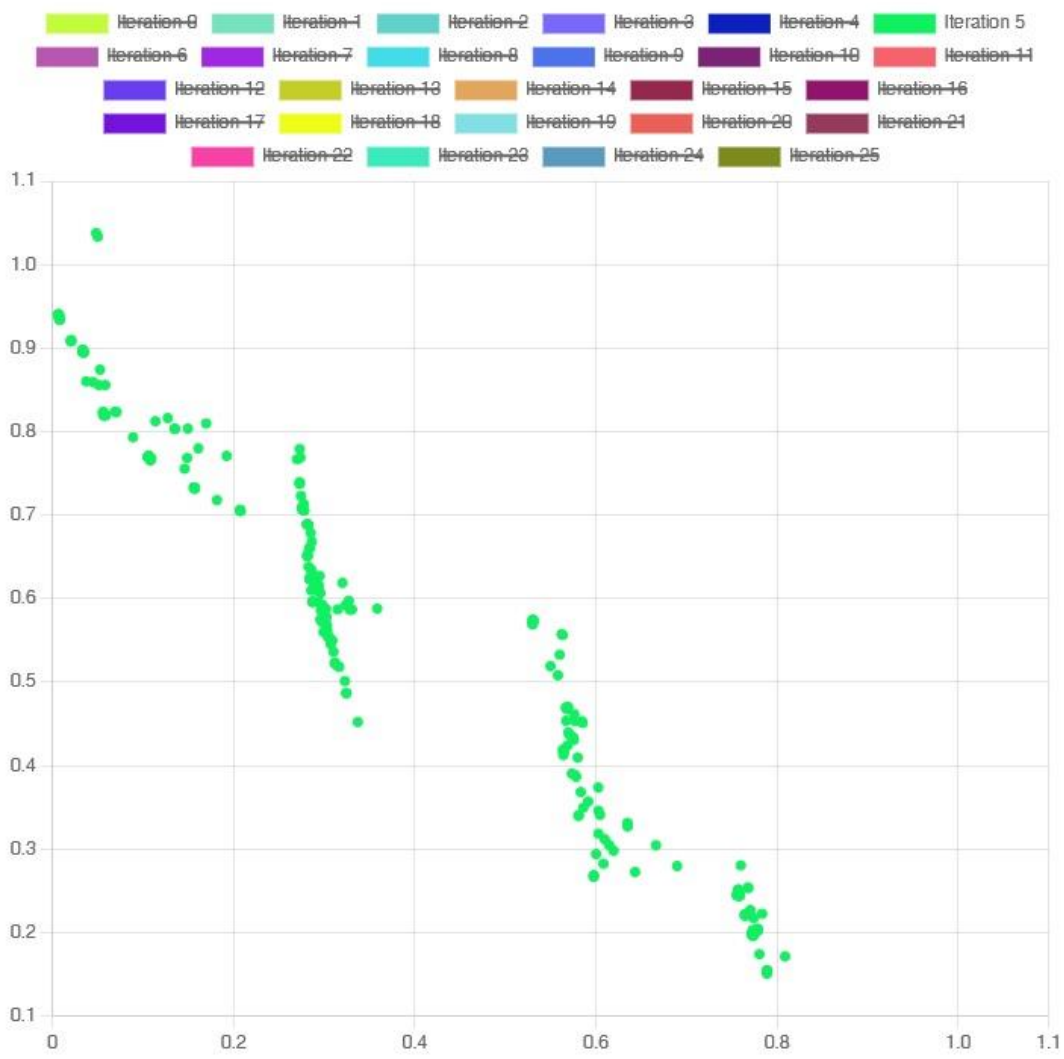
Symbol Objects in decision space

This graph shows where the symbol objects are in decision space in each iteration. Click on color box to display corresponding data.



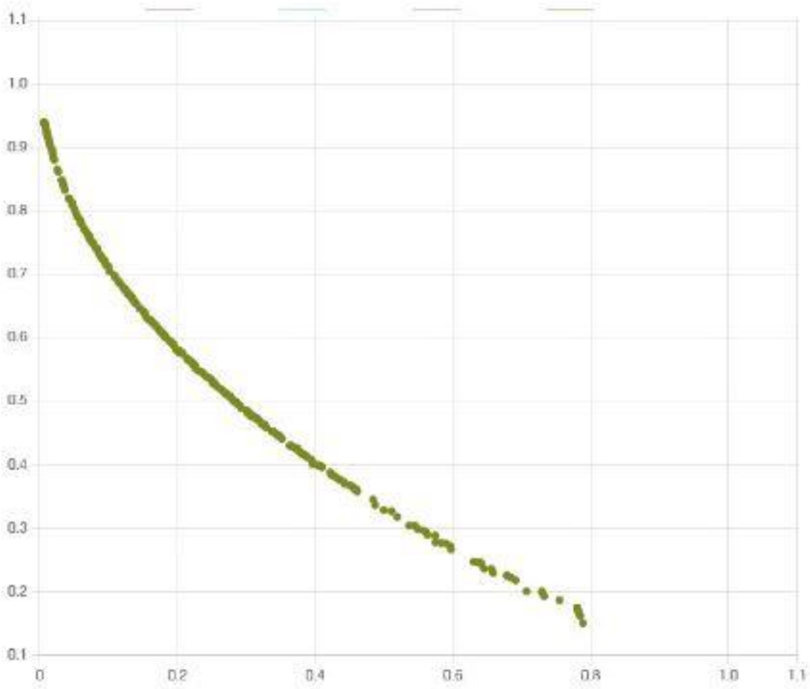
Output – Objective Visualization

This graph shows where the first pareto-front lie in each iteration. Click on color box to display corresponding data.

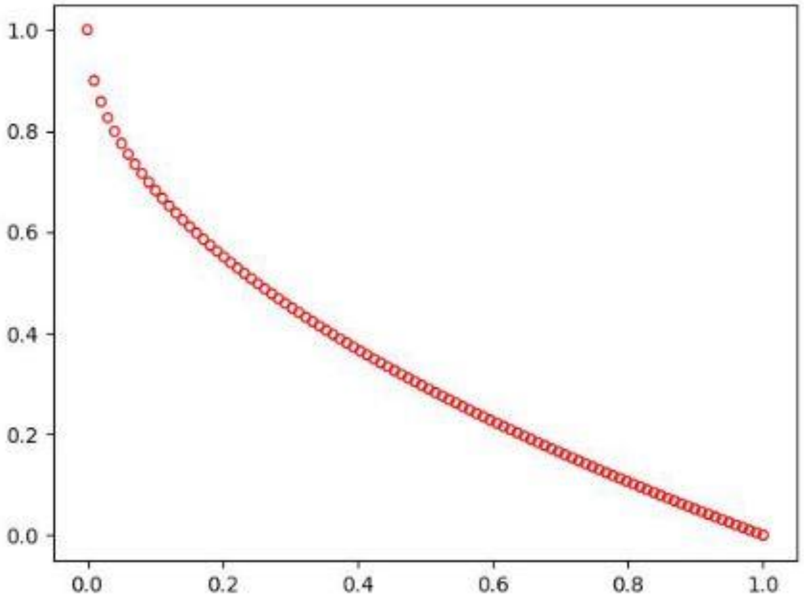


Output (contd.)

- The algorithm is producing satisfactory results with ZDT1, ZDT2 and ZDT3 objectives.

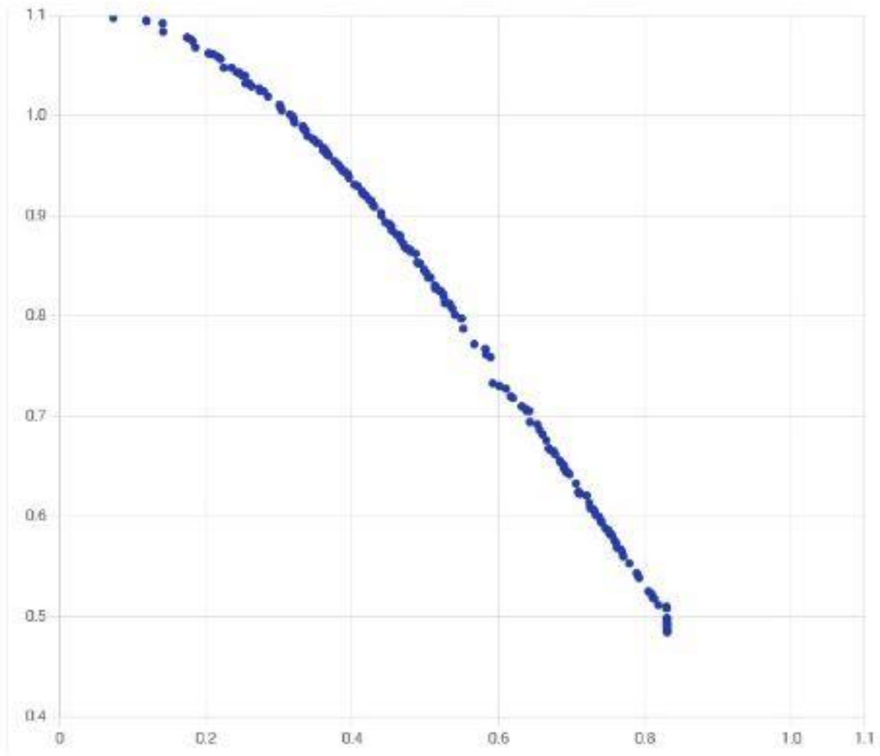


(a) Output of the Algorithm

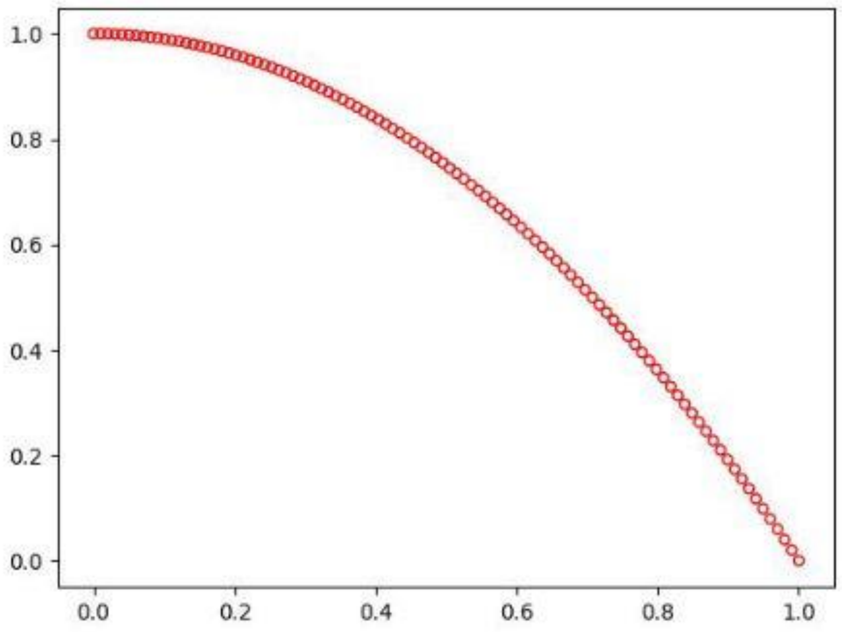


(b) True Pareto front of ZDT1

Output (contd.)

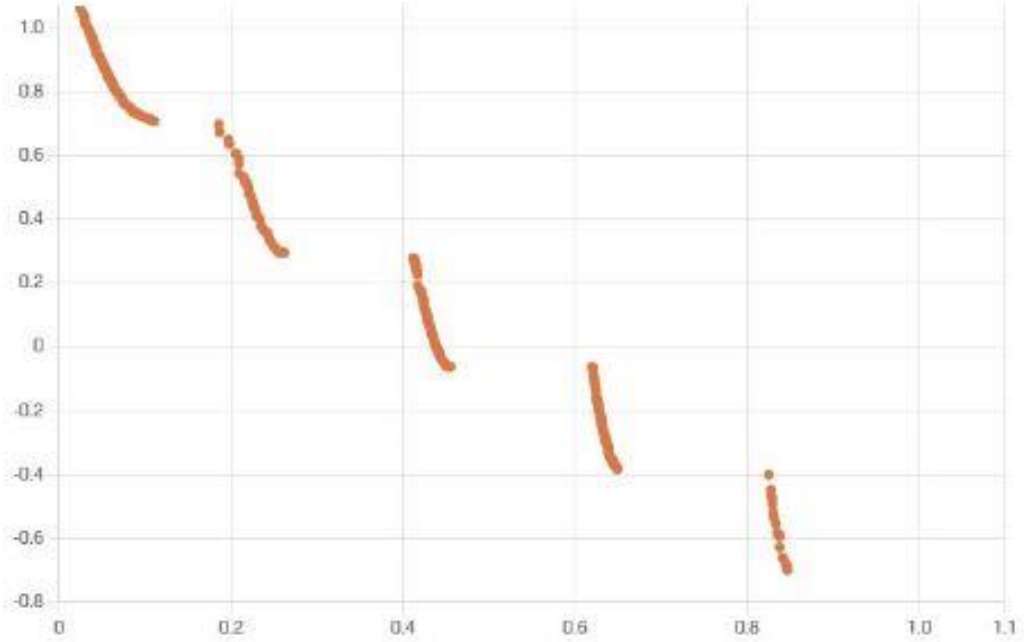


(a) Output of the Algorithm

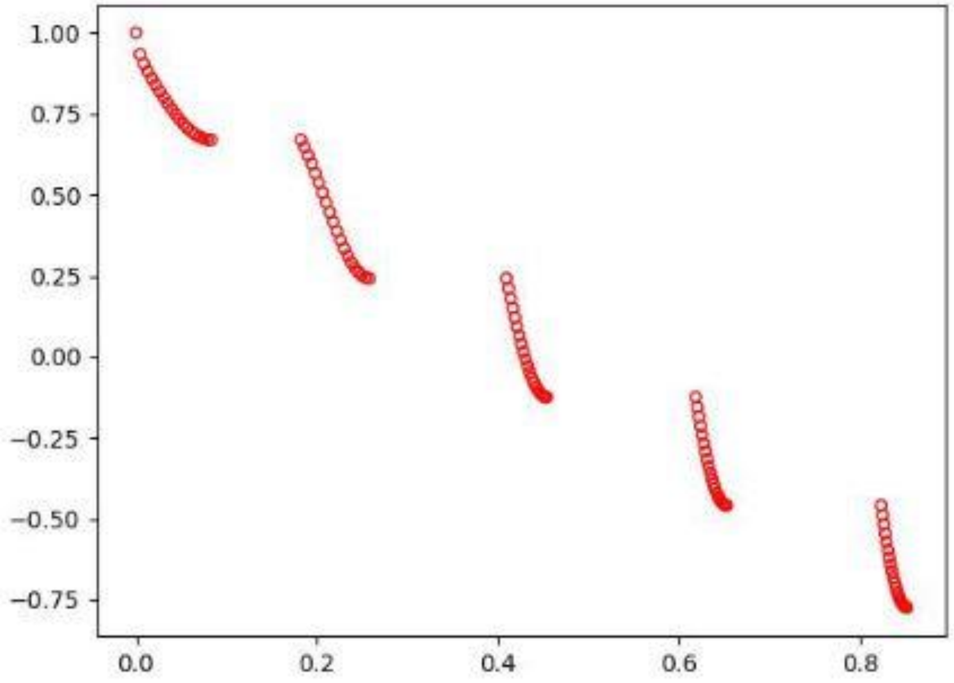


(b) True Pareto front of ZDT2

Output (contd.)



(a) Output of the Algorithm



(b) True Pareto front of ZDT3

Results – Experimental Setup

All of the

- programming,
- testing,
- debugging,
- execution and
- benchmarking of MOOP solving algorithms

are done in computer setup with:

- Intel Core I5 5200U processor (2 physical cores)
- 8 GB of DDR3 RAM
- Windows 10 pro operating system

Results – Changing Problem Complexity

- **Setup :**
  - Problem : ZDT1
  - Number of partitions : 4
  - Symbol objects per partition : 32
  - Number of iterations : 50
  - Number of executions : 20

SN	Dimension	Performance Indicators							
		GD				IGD			
		Best	Worst	Mean	Std. Dev.	Best	Worst	Mean	Std. Dev.
1	2	**3.54E-03**	**1.51E-02**	**7.18E-03**	**2.67E-03**	**3.48E-03**	**1.15E-02**	**6.36E-03**	**1.91E-03**
2	3	1.60E-02	5.82E-02	3.51E-02	1.10E-02	1.49E-02	5.77E-02	3.45E-02	1.11E-02
3	4	4.37E-02	8.48E-02	6.89E-02	1.22E-02	4.18E-02	8.46E-02	6.83E-02	1.24E-02

Results – Comparison with standard MOOP algorithms

- To compare the performance of proposed algorithm with NSGA-II and MOEA/D following parameters are used

- Parameters for membrane inspired algorithm
  - No. of partitions per dimension = 4
  - No. of symbol objects per partition = 32
  - Mutation chance = 0.05

- Parameters for NSGA-II
  - Population size = 512

- Parameters for MOEA/D
  - Sub-problems: N = 512,
  - Number of neighbors: T = 10
  - The probability of selecting parents from the neighbors = 0.9
  - Mutation rate: C R = F = 0.5

Results – Comparison with standard MOOP algorithms

GD	Mem. Algorithm	NSGA II	MOEA/D
ZDT1			
Best	**3.05E-03**	4.69E-03	4.44E-03
Worst	1.02E-02	7.54E-03	**7.13E-03**
Mean	6.32E-03	6.31E-03	**5.97E-03**
Std. Dev.	1.86E-03	8.84E-04	**8.38E-04**
ZDT2			
Best	6.18E-03	5.19E-03	**4.38E-03**
Worst	2.39E-02	**8.10E-03**	8.16E-03
Mean	1.50E-02	6.85E-03	**6.32E-03**
Std. Dev.	4.89E-03	**7.43E-04**	8.37E-04
ZDT3			
Best	**2.08E-03**	5.21E-03	4.72E-03
Worst	1.76E-02	**7.57E-03**	7.46E-03
Mean	9.79E-03	6.36E-03	**5.95E-03**
Std. Dev.	4.45E-03	7.54E-04	**8.68E-04**

IGD	Mem. Algorithm	NSGA II	MOEA/D
ZDT1			
Best	2.65E-03	1.27E-03	**2.65E-04**
Worst	8.76E-03	**3.79E-03**	4.03E-03
Mean	5.94E-03	2.71E-03	**2.19E-03**
Std. Dev.	1.66E-03	7.94E-04	**8.99E-04**
ZDT2			
Best	6.14E-03	3.71E-03	**3.02E-03**
Worst	2.37E-02	6.66E-03	**6.31E-03**
Mean	1.46E-02	5.09E-03	**4.64E-03**
Std. Dev.	4.68E-03	**9.32E-04**	9.77E-04
ZDT3			
Best	1.39E-03	6.15E-04	**2.65E-04**
Worst	1.74E-02	3.43E-03	**3.07E-03**
Mean	9.60E-03	1.97E-03	**1.55E-03**
Std. Dev.	4.54E-03	8.47E-04	**8.35E-04**

Conclusion

- The algorithm presented in this thesis was able to generate results that are closer to standard MOOP algorithms.
- In some cases, the proposed algorithm performed better than the standard MOOP solvers.
- However,
  - the mean and
  - standard deviation values of IGD (inverted generational distance)

  generated by the proposed algorithm are still worse than other algorithms, indicating a larger variance in the output generated.

Future Works

- Incorporating several concepts from membrane computing model can improve the quality of solutions.
- To adapt the algorithm for higher dimensions of the decision space, the additive nature of MOOP can be utilized, as presented in (H. Wang et al. 2022) [17].
- Candidates who fail in the mating selection stage can be
  - agitated using the Brownian motion method,
  - as suggested in K. Sowjanya (et al. 2021)[18], to better utilize the population.
- Finally, this algorithm can be adapted to solve multi-objective integer and mixed-integer programming problems using the hybrid approach proposed in (R. S. Burachi et al. 2021) [19].

Tools

SN	Programming Language/IDE	Version	Usage
1	Python	3.11.1	To use benchmark function and indicators
2	pymoo	0.6.0.1	MOOP open source implementations
3	Numpy	1.24.2	For mathematical calculation
4	PHP	8.2.0	For the implementation of algorithm presented in this thesis
5	Netbeans IDE	17	As a coding platform for PHP language
6	PyCharm IDE	2022.3.3	As a coding platform for python language

Queries/Comments/Feedback?

Thank You !