Acknowledgement

I would like to express my sincere gratitude to our Head of Department Dr. Jyoti Tandukar, Prof. Dr. Sashidhar Ram Joshi, Prof. Dr. Subarna Shakya, Dr. Aman Shakya, for their encouragement and precious guidance. I am thankful to our program coordinator Anand Kumar Sah for providing suitable platform to prepare this thesis work.

I express my deepest appreciation to Dr. Sanjeeb Prasad Panday, my supervisor, for his guidance, support, and encouragement.

I extend my gratitude to the faculty of Department of Electronics and Computer Engineering and my fellow classmates for sharing their perspectives and offering guidance on my thesis work.

ABBREVIATIONS

DM	Decision Maker
IDE	Integrated Development Environment
$JSON$	JavaScript Object Notation
LHS	Latin Hypercube Sampling
$MFEA$	Multi-Factorial Evolutionary Algorithm With Online Transfer Parameter Estimation
$MOEA$	Multi-objective Evolutionary Algorithm
$MOOP$	Multi-objective Optimization Problems
$NSGA$	Non-dominating Sorting Genetic Algorithm
ZDT	Zitzler–Deb–Thiele's function

List of Figures

LIST OF TABLES

Contents

Chapter 1

Introduction

1.1 Background

Multi-objective optimization is a field of mathematical optimization problems involving more than one objective functions. It is also called vector optimization, multi-objective programming, multi-criteria optimization or Pareto optimization. Multi-objective optimization has been used in various applications where optimal decision is required in the presence of multiple trade-offs. In practical cases where multiple objectives are to be satisfied no single solution exists that is optimal for all the objectives. These type of objectives are referred to as conflicting objectives where improvement of one may degrade the other(s). A solution is called non-dominating or Pareto-optimal if none of the objectives can be improved without degrading the other objectives. Unlike single objective optimization multi-objective optimization usually has more than one solutions that are Pareto-optimal. Purpose of various multi-objective optimization algorithms is to find these non-dominating solution(s).

Membrane computing model provides a framework to utilize the effectiveness of working of biological cells by abstraction of various activities performed by the cell and its interaction with other cells and the environment. Equivalently membrane computing is an area of computer science aiming to abstract computing ideas and models from the structure and the functioning of living cells, as well as from the way the cells are organized in tissues and higher order structures. This type of abstraction has vast potential for parallel as well as

distributed processing.

Similar to other bio-inspired optimization algorithms membrane computing can offer greater exploration of solution space and is suitable for Pareto based method of optimization. This method can produce multiple solutions that are close to Pareto-optimal solutions. With these solutions decision maker (DM) can make suitable decision. Bio-inspired methods are particularly suitable in cases where preferences for objectives (a-priori preference expression) cannot be assigned before optimal solutions are available.

1.2 Statement of the Problem

Multi-objective optimization problems require achieving optimal solution among conflicting requirements. More than one optimal solution is possible but decision maker may prefer on solution over another even if mathematically they arrive at same optimal value. So full exploration of Pareto-optimal solutions also known as Pareto Front is necessary.

1.3 Objectives

To create membrane computing inspired evolutionary algorithm to solve multi-objective optimization problem and compare its performance with current bio-inspired algorithms.

Chapter 2

Literature Review

Computing with membranes is a branch of Molecular Computing initiated by Gheorghe Paun by the paper Computing with Membranes[1]. This computing model is also called P systems[2] which have the same computing power with a Turing universal computing model. Membrane systems mainly focuses on the various computational features of the membranes such as transferring chemical substances between membranes or chemical reaction in the region of the membrane, instead of modeling the biological membranes. In strict sense number of principles are abstracted underlying the functioning of biological membranes, and this abstraction is used as the working mechanism of the computing model.

Based on the above-mentioned context, an integral membrane system includes the nested membrane structure, multisets, and reaction rules. Multiset, which consists of a collection of symbol-objects, is placed in the compartments defined by the membrane structure, and it is evolved by executing the reaction rules in a non-deterministic and maximally parallel manner[1]. The membranes except the skin membrane can be dissolved and divided by invoking the corresponding reaction rules. Because the structure of membrane systems provides an enhanced parallelism, membrane systems can solve intractable problems in a polynomial time [2]. More specifically, the membrane system with an enhanced parallelism is able to trade space for time, that is to say, it can solve intractable problems in a feasible time due to making use of an exponential space.

An evolutionary algorithm is proposed in [3] which is based on membrane systems to solve the global numerical optimization problems for single objective functions. The algorithm

employs fundamental ingredients of membrane systems, including multisets, reaction rules and membrane structure. In addition, the algorithm incorporates information of the adjacent symbol-objects, to guide the evolution toward the global optimum, efficiently.

In multi-objective optimization more than one objectives are needed to be reached. When decision maker cannot express the preference information about these objectives such multi-objective optimization methods can be classified as no-preference methods [4].

Legacy works in multi-objective optimization include Weighing Method [4] where weighted sum of all objective functions are merged into one objective and solved as single optimization problem. This will produce one solution among the many Pareto-optimal solutions. Changing of the individual weights can produce other solutions. This method has shortcomings where problem is non-convex. In ε-Constraint Method one of the objective functions is selected to be optimized, the others are converted into constraints [4]. Solving non-convex single-objective optimization problems to global optimality or at least with some guaranteed proximity to the optimal value is known to be a challenge in mathematical optimization [5].

Several evolutionary algorithms are proposed over the years with capabilities to resolve challenges with classical optimization methods [6], [7], [8], [9].

Earlier evolutionary algorithm such as Non-dominating Sorting Genetic Algorithm (NSGA) [6] uses concepts from genetic algorithm such mutation, selection and ranks the results in terms their non-dominance.

NSGA was further improved in NSGA-II [7] and NSGA-III [8] reducing computational complexity and improvement of diversity and optimal values of solutions.

Multi-objective Evolutionary Algorithm Based on Decomposition (MOEA/D) [9] combines classical method of optimization known as decomposition with concepts form evolutionary algorithm and is comparable to performance with NSGA-II.

Estimation of Pareto Front (PF) and selection of offspring based on the proximity of offspring to the estimated PF is discussed in [10]. Unlike other evolutionary algorithms this method takes the shape of PF into account while exploring solution space.

Chapter 3

Research Methodology

This section describes proposed optimization algorithm and benchmark function to evaluate the algorithm which will be developed during the course of this research.

3.1 Components

3.1.1 Multi-objective optimization problem

Multi-objective Optimization Problems (MOOP) involves a set of function that are to be maximized or minimized. Mathematically,

$$\text{min } f_m(x), \text{ where } m = 1, 2, 3, ..., M$$

subject to

$$g_i(x) \geq 0, \text{ where } i = 1, 2, 3, ..., I$$

$$h_j(x) = 0, \text{ where } i = 1, 2, 3, ..., J$$

$$\text{and } x_L \leq x \leq x_U, \text{ where}$$

- x_L is lower bound and x_U is upper bound

- $f_m(x)$ are objectives to be minimized
- $g_i(x)$ are constraints with inequality
- $h_j(x)$ are constraints

If an objective is to maximize, negation or inverse of the objective becomes minimizing problem. Similar operations can be done to constraints to standardize optimization problem.

Multi-objective optimization problems in principle have a set of optimal solutions also known as Pareto-optimal solutions. Goal of any multi-objective optimization algorithm is to find as many and diverse set of Pareto-optimal solutions.

3.1.2 Membrane System

The membrane structure is a hierarchical arrangement of membranes embedded in a skin membrane which separates the system from its environment.

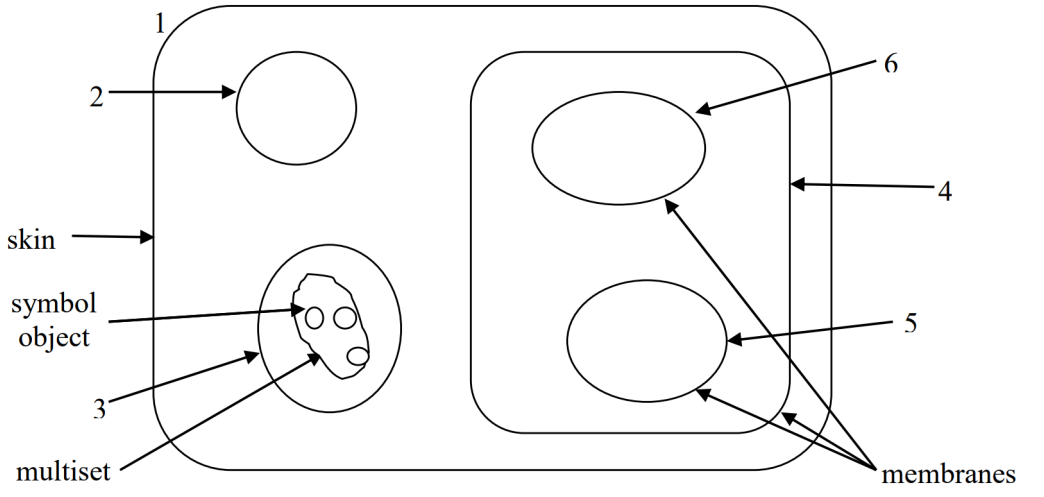

Figure 3.1: Membrane Structure

To further understand a cell-like membrane system, its basic structure with degree n is simply described in eq. (3.1).

$$\Pi = (V, T, \mu, w_1, w_2, w_3, ..., w_n, R) \tag{3.1}$$

where,

- V is the alphabet. Its element is named as an object. An object is the abstract representation of atomic, molecular or the other chemical substances. The object may be represented by symbol or string, also named as symbol-object
- $T \subseteq V$, where T is the output alphabet
- μ is a membrane structure with degree n
- $w_i \in V^*$, $1 \leq i \leq n$, w_i represents the multiset in the i^{th} region of the membrane structure μ
- R represents the reaction rules in the region of membranes

To solve optimization problems using membrane system symbol objects are used as potential solutions. Then using reaction rule and functional rule as described in [1] membrane system can be adapted to solve multi-objective optimization problem.

3.1.3 Latin Hypercube Sampling

Latin Hypercube Sampling (LHS) is a statistical technique that generates a group of evenly distributed points in a multi-dimensional space. It is a popular method for creating an initial population in optimization problems. This approach involves dividing the dimensions of the decision space into intervals and randomly selecting one point from each interval to ensure a representative sample of all intervals. This results in a well-distributed set of points that offers a diverse initial population for the optimization process. LHS is particularly beneficial for solving computationally intensive engineering and scientific problems that require a well-distributed initial population. Compared to random sampling, LHS generates a more uniform distribution of points, which leads to faster convergence of the optimization algorithm and better-quality solutions.

3.2 Evolutionary Optimization Algorithm based on Membrane Computing

This algorithm works by partitioning decision space into multi-dimensional volumes (membranes). Dimension of the membranes depends on the dimension of decision space. Symbol objects (solution, objective pairs) are initialized in each partition. Reproduction and mutation rules are applied inside each membrane. This will double the symbol object count. Then symbol objects are extracted from each membrane and sorted by non-domination. Half of the symbol objects are then rejected and remaining symbol objects are sent into corresponding membranes. At this point some of the membranes may contain more or less symbol objects than at the start. If a membrane contains more symbol objects; division rule is applied to it. If a membrane contains less symbol objects it is merged with its neighbor.

3.2.1 Partitioning of decision space

For an MOOP with d decision variables and n partitions per dimension. Number of membranes will be n^d. Lower bound l and upper bound u for each membrane is also be assigned.

3.2.2 Initialization

If s is the number of symbol objects per membrane then there will be $(n^d) * s$ symbol objects. 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

3.2.4 Mate Assignment

Sorted symbol objects are divided into two parts Elites solutions and non-elite solutions. A parent pair is generated by selecting one from the elite solutions and one from all of the solutions.

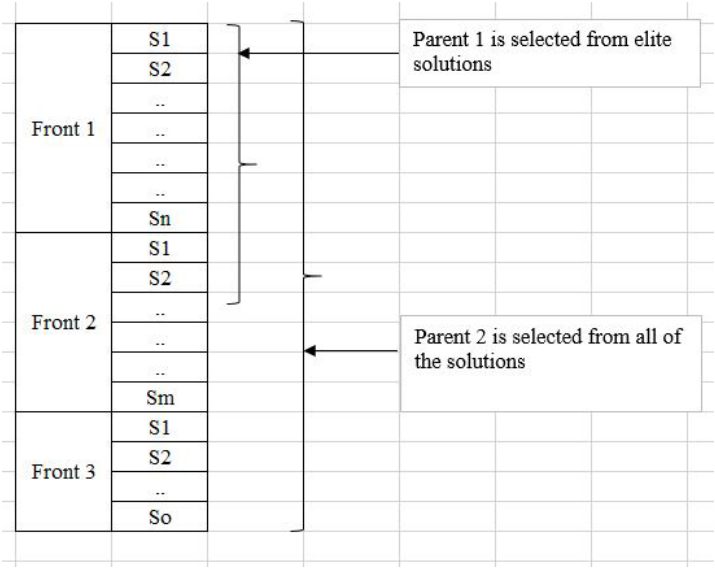

Figure 3.4: Parent Selection

3.2.5 Reproduction

An offspring is created by combining the features of parent pair. If one of the parent is dominating other offspring gets more characteristics from dominating parent. Otherwise offspring gets equal characteristics from both parents. Small mutation is introduced into offspring.

To generate $n + 1^{th}$ decision variable of offspring :

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

Then,

$$w_1 = \begin{cases} 0.5 + \text{Rand}(-m, +m), & \text{if Parent 1 and Parent 2 are in same front} \\ w_e + \text{Rand}(-m, +m), & \text{if Parent 1 dominates Parent 2} \end{cases} \tag{3.2}$$

$$w_2 = 1 - w_1 \tag{3.3}$$

$$x_{n+1} = w_1 * x_{1n} + w_2 * x_{2n} \tag{3.4}$$

where,

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

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

3.2.6 Global non-dominating sort

The symbol objects are extracted from all of the membranes. Extracted symbol objects are then sorted according to no-domination. This operation generates a number of pareto-fronts.
Front 0 – No other solution dominate the solution in this front
Front 1 – Solutions in this front are dominated in exactly one objective by solution in front 0 and so on.
At this stage termination condition is checked. Termination condition includes number of iteration and first front solution count. If termination condition is met then front-0 is the result of optimization. If not algorithm proceeds to steps below.

3.2.7 Reject 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.

3.2.8 Repopulate Rule

Accepted symbol objects are sent into their corresponding parent membranes.

3.2.9 Divide and Merge Rule

At this point some of the membranes may contain more or less symbol objects than at the start. If a membrane contains more symbol objects; division rule is applied to it. If a membrane contains less symbol objects it is merged with its neighbor.

Threshold for merge and divide also constitutes one of the parameters of this algorithm.

Threshold for division = $thresholdDivide$ Represented in percentage
Threshold for merge = $thresholdMerge$ Represented in percentage
These parameters must satisfy
$thresholdDivide$ = $100 + thresholdMerge$

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 $thresholdMerge$ to $thresholdDivide$

Repeat from "Non-dominating sort inside membrane" step until termination condition is reached.

3.3 Performance analysis with benchmark functions

Following functions will be used to test the performance of the algorithm:

3.3.1 Schaffer Function 1

Minimize:

$f_1(x) = x^2$,
$f_2(x) = (x-2)^2$

Search Domain :

$-A \leq x \leq A$ where values of A from 10 to 10^5

3.3.2 Schaffer Function 2

Minimize:

$$f_1(x) = \begin{cases} -x, & \text{if } x \leq 1 \\ -x-2, & \text{if } 1 < x < 3 \\ 4-x, & \text{if } 3 < x \leq 4 \\ x-4, & \text{if } x > 4 \end{cases}$$

$f_2(x) = (x-2)^2$

Search Domain : $-5 \leq x \leq 10$

3.3.3 Zitzler–Deb–Thiele's Function 1

Also known as ZDT1 which defines the minimizing of:

$f_1(x) = x_1$
$f_2(x) = g(x)h(f_1(x), g(x))$

$g(x) = 1 + \frac{9}{29}\sum_{i=2}^{30} x_i$
$h(f_1(x), g(x)) = 1 - \sqrt{\frac{f_1(x)}{g(x)}}$

Search Domain :

$0 \leq x_i \leq 1$
$1 \leq i \leq 30$

3.3.4 Zitzler–Deb–Thiele's Function 2

Also known as ZDT2 which defines the minimizing of:

$f_1(x) = x_1$
$f_2(x) = g(x)h(f_1(x), g(x))$
$g(x) = 1 + \frac{9}{29}\sum_{i=2}^{30} x_i$
$h(f_1(x), g(x)) = 1 - (\frac{f_1(x)}{g(x)})^2$

Search Domain :

$0 \leq x_i \leq 1$
$1 \leq i \leq 30$

3.3.5 Zitzler–Deb–Thiele's Function 3

Also known as ZDT3 which defines the minimizing of:

$f_1(x) = x_1$
$f_2(x) = g(x)h(f_1(x), g(x))$
$g(x) = 1 + \frac{9}{29}\sum_{i=2}^{30} x_i$
$h(f_1(x), g(x)) = 1 - \sqrt{\frac{f_1(x)}{g(x)}} - \frac{f_1(x)}{g(x)} sin(10\pi f_1(x))$

Search Domain :

$0 \leq x_i \leq 1$
$1 \leq i \leq 30$

3.4 Performance comparison with standard algorithms

Performance of the proposed algorithm will be compared with following standard multi-optimization algorithms:

1. NSGA II
2. Multi-Factorial Evolutionary Algorithm With Online Transfer Parameter Estimation (MFEA-II)

3.5 Tools

1. Python 3
2. Netbeans IDE
3. PHP 8
4. Matlab 2022a
5. Visual Studio Code

Chapter 4

Results and Discussion

4.1 Works Accomplished

4.1.1 Implementation of algorithm

Implementation of algorithm described in 3.2 is completed in PHP 8.2 programming language. This implementation is capable of

1. Solving multi-objective optimization problems
2. Storing the progress details of each iteration of algorithm execution

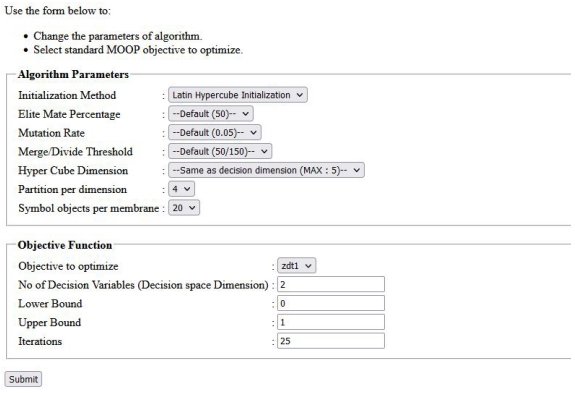

Figure 4.1: Algorithm Test Interface

4.3 Remaining Works

4.3.1 Fine tuning of algorithm parameters

Output of the algorithm can be improved by properly choosing algorithm parameters such as

1. Number of partitions (membranes)
2. Number of symbol objects
3. Mutation Rate
4. Elite parent preference

Continuous visualization and tuning can further improve the output of algorithm.

4.3.2 Performance analysis with benchmark functions

Comparison of output of the algorithm with true pareto front of the standard optimization function will be done. Mean deviation, average deviation and standard deviation from true pareto front will be presented.

4.3.3 Performance comparison with standard MOOP algorithms

Performance of the algorithm will be compared with standard multi-objcetive optimization algorithms :

1. NSGA-II
2. MOEA/D

Chapter 5

Work Schedule

The working schedule for this thesis is proposed as follows:

Table 5.1: Time Schedule for the Thesis Work

Task	**Weeks**											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Literature Review ✓												
Prepare suitable reaction rules ✓												
Prepare suitable evolution rules ✓												
Implementation ✓												
Testing and Debugging ✓												
Comparison and Analysis												
Documentation												

Reference list

[1] Gheorghe Paun. Computing with membrane. *J. Comput. System Sci.*, 61:108–143, 2000.

[2] Gheorghe Paun and Grzegorz Rozenberg. A guide to membrane computing. *Theory Computer Sci.*, 287:73–100, 2002.

[3] Min Han, Chuang Lui, and Jun Xing. An evolutionary membrane algorithm for global numerical optimization problems. *Elsevier Information Sciences*, 276, 2014.

[4] Jürgen Branke, Kalyanmoy Deb, Kaisa Miettinen, and Roman Słowi´nski (Eds.). Multiobjective optimization: Interactive and evolutionary approaches. *Applied Soft Computing*, 2008.

[5] Gabriele Eichfelder. Twenty years of continuous multiobjective optimization in the twenty-first century. *2019 2nd International Conference on new Trends in Computing Sciences (ICTCS)*, pages 1–6, 2019.

[6] N. Srinivas and K. Deb. Multiobjective function optimization using nondominated sorting genetic algorithms. *Evolutionary Computing*, 2(3):221—248, 1995.

[7] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):187, 2002.

[8] Kalyanmoy Deb, Himanshu Jain, and Roman Słowi nski (Eds.). An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4), 2014.

[9] Qingfu Zhang and Hui Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6), 2007.

[10] Annibale Panichella. An adaptive evolutionary algorithm based on non-euclidean geometry for many-objective optimization. *In Genetic and Evolutionary Computation Conference*, 2019.