Using Evolutionary Algorithms for Finding Optimal Initial Conditions in a Zero-Player Game

Manchikatla Navya Sri
Department of Electronics and Communication
Indian Institute of Technology Guwahati
Guwahati, Assam, India - 781039
m.navya@iitg.ac.in

Rohit Priyanka Nandwani
Department of Physics
Indian Institute of Technology Guwahati
Guwahati, Assam, India - 781039
nandwani@iitg.ac.in

***Abstract*—A game that evolves as determined by its initial state, requiring no further input from humans is considered a zero-player game. In this report, we use several evolutionary algorithms to solve a physics-based zero-player maze game. The goal is to reach a region of space, called the "win-zone" by choosing appropriate initial components of velocity, as quickly as possible.**

***Index Terms*—Evolutionary Algorithms, Zero-Player Game, Initial Conditions, Physics**

I. Introduction

In this report, we will be working with two versions of the same problem. In each problem, there is a "massy" point-particle (called "ball") which can move around in space, following Newton's laws of motion. The "environment" in which the *ball* exists is filled with multiple fixed elastic walls and a "bucket" with walls as its boundary. The *environment* is also laden with zones, called "end-zones". If the ball enters an *end-zone*, the simulation ends.

The objective of the game is to reach a special region of *end-zone*, called the "win-zone". *Win-zone* is the *lid* at the top of the bucket, passing through which guarantees entry inside the *bucket*.

- In the first problem, the objective is to reach the win-zone, *without any consideration of the time-taken* to do so.
- In the second problem, the objective is to *reach the win-zone in the least amount of time* possible.

To solve the problem using metaheuristic techniques, the objective function to be minimized is introduced in the next section, along with the exact problem statement.

II. Problem Statement

A. Setup and Data Points

As mentioned in the introduction, the first problem consists of walls with coefficient of restitution, $e = 1$. There are 12 walls (*line segments*) spread throughout the region, with the starting and ending coordinates of line segments given in TABLE I. Also, see Fig. 1 to see the orientation of x and y axes, and the position of origin.
The ball's starting position is fixed, and given in TABLE II.

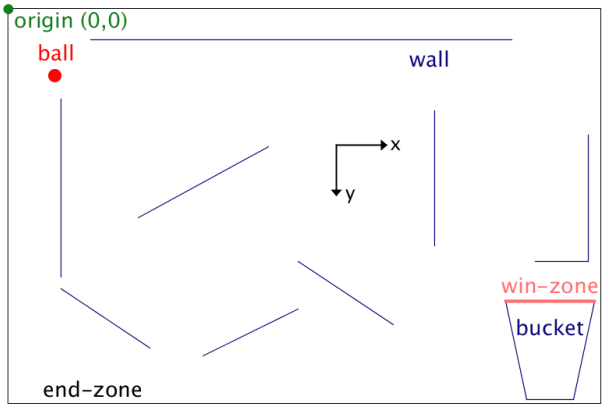

Fig. 1. Environment 1 : Elastic Walls

TABLE I
Coordinates of Walls in Environment-1

Wall	Point 1 (x,y) (in %)	Point 2 (x,y) (in %)
1	(9.$\bar{3}$, 23)	(9.$\bar{3}$, 68)
2	(9.$\bar{3}$, 71)	(24, 86)
3	(22, 53)	(44, 35)
4	(14.$\bar{6}$, 8)	(85.$\bar{3}$, 8)
5	(33.$\bar{3}$, 88)	(49.$\bar{3}$, 76)
6	(49.$\bar{3}$, 64)	(65.$\bar{3}$, 80)
7	(72, 26)	(72, 60)
8	(84, 74)	(87.5, 99)
9	(99, 74)	(95.5, 99)
10	(89.$\bar{3}$, 64)	(98.$\bar{6}$, 64)
11	(98.$\bar{6}$, 32)	(98.$\bar{6}$, 64)
12	(87.5, 99)	(95.5, 99)

TABLE II
Initial Coordinates of Ball in Environment 1

x-coordinate (in %)	y-coordinate (in %)
8	17

width of the outer end-zone = 760 units
height of the outer end-zone = 506.$\bar{6}$ units

B. Objective Function

In all of the problems, the **decision variables** are the components of initial velocity imparted,

$$\mathbf{u} = (u_x, u_y)$$

These values are given as the initial conditions for the *simulation*. After the simulation ends (when the ball reaches end-zone, which includes the win-zone), the last coordinate of the ball is recorded.

Let

$\mathbf{r} = (x, y)$ be the last coordinate of the ball.
$\mathbf{r_c} = (x_c, y_c)$ be the mid-point of the win-zone.

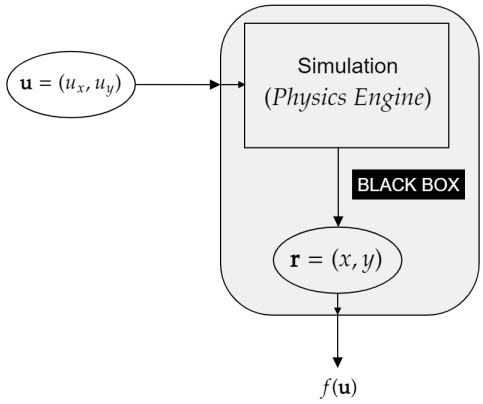

Fig. 2. Objective Function

TABLE III
COORDINATES OF MID-POINT OF WIN-ZONE

x-coordinate, x_c (in %)	y-coordinate, y_c (in %)
91.5	74

The objective function $f(\mathbf{u})$ is defined as follows:

$$f(\mathbf{u}) = \begin{cases} \mathrm{d}(\mathbf{r},\mathbf{r}_c) & , \mathbf{r} \in \text{win-zone} \\ \lambda \cdot \mathrm{d}(\mathbf{r},\mathbf{r}_c) & , \mathbf{r} \notin \text{win-zone} \end{cases}$$

,where the factor of $\lambda(= 10)$ is introduced to create a bias towards the win-zone, in contrast to the remaining end-zone.

III. RESULTS - PROBLEM 1

A. Brute Force Shallow Search

Despite the fact of Problem 1 (*elastic walls*) being the least complex problem amongst all three problems considered; the time required, number of computations performed, and plainly the number of objective function calls is significant.

Still a shallow search for "solutions" (velocity values that lead to particle ending in win-zone) was performed on the interval:

$$u_x \in [0, 5]$$
$$u_y \in [-5, 5]$$
$$\text{step-size} = 0.01$$

This will help us analyse the behaviour of metaheuristic techniques in this relatively-simpler search space.

*Note:

"Solutions" have objective function values less than or equal to half the width of win-zone.

$$\implies \mathbf{u} \text{ is a solution} \iff f(\mathbf{u}) \leq \frac{\text{width of win-zone}}{2} = 57$$

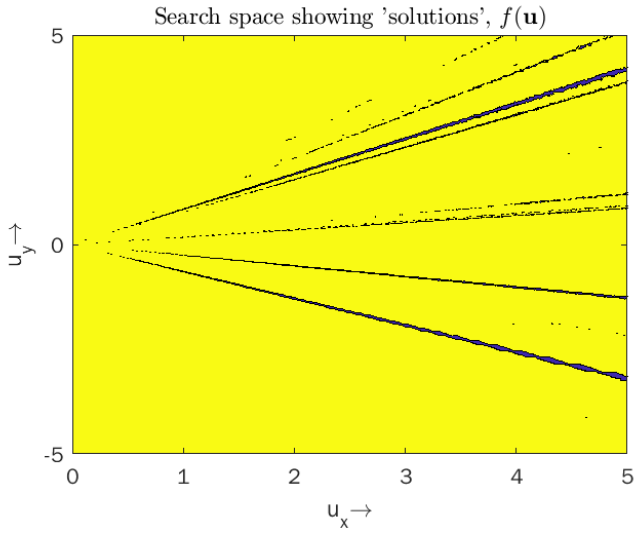

Fig. 3. Brute Force Shallow Search Results

This simple shallow search took $10\ to\ 12\ hours$ because of the complexity of the Objective function. Almost all solutions lie on linear *bands* passing through origin. The relatively sharp behaviour introduced due to the λ factor can be seen in Fig. 4 and Fig. 5.

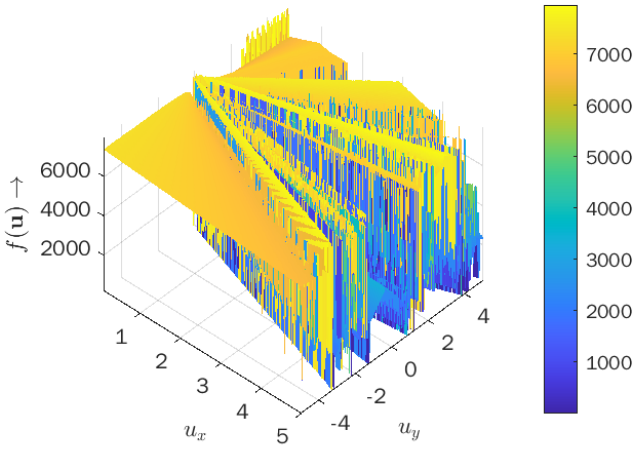

Fig. 4. Objective Function Plot

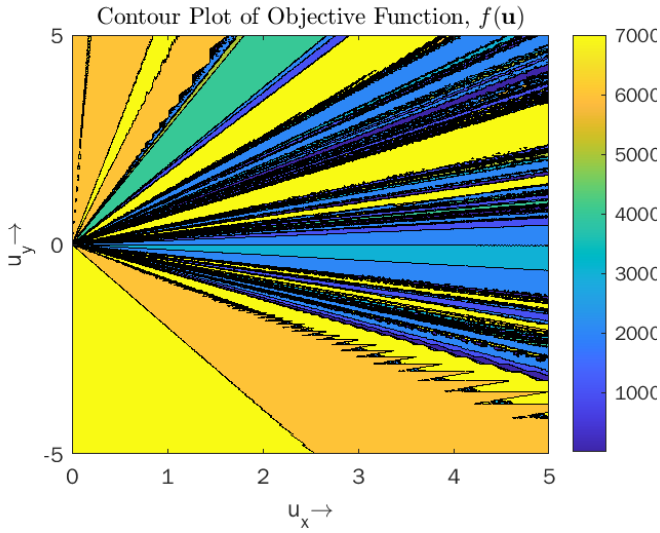

Fig. 5. Nature of the Objective Function

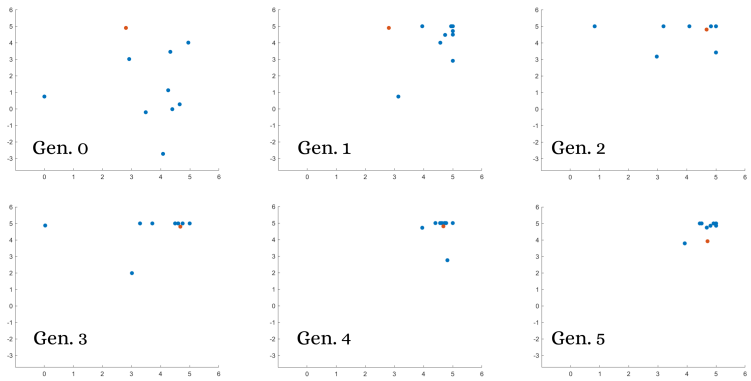

Fig. 6. Particle Swarm Optimization (first five generations)
Global Best and Swarm

B. Comparison of Algorithms

Considering the sharp behaviour of objective function, the algorithms (*and their respective parameters*), for which the performance was compared are given in TABLE IV.

Average behaviour of each of these algorithms starting from the same random population can be seen in Fig. 7, Fig. 8 and Fig. 9.

Mean behaviour of RCGA (for the chosen parameters atleast) is the worst. TLBO, PSO and ABC perform the best. One important thing to keep in mind is that these graphs only represent the average behaviour. This means all the outliers and random fluctuations will easily get damped leading to a more representative behaviour.

But, it would be unwise to not consider the total number of functional evaluations, as shown in TABLE IV. Clearly, keeping the complexity and time taken for calculating the objective function of a single value in mind, the *"best"* algorithm to choose amongst these 5 would be **Particle Swarm Optimization** (PSO).

Fig. 6 shows the distribution of swarm and global best solution for a single run (5 generations).

The performance of the metaheuristic technique can be seen as an animation at the following url :

https://cl643project.github.io/zeroplayer/animation1.mp4

IV. Results - Problem 2

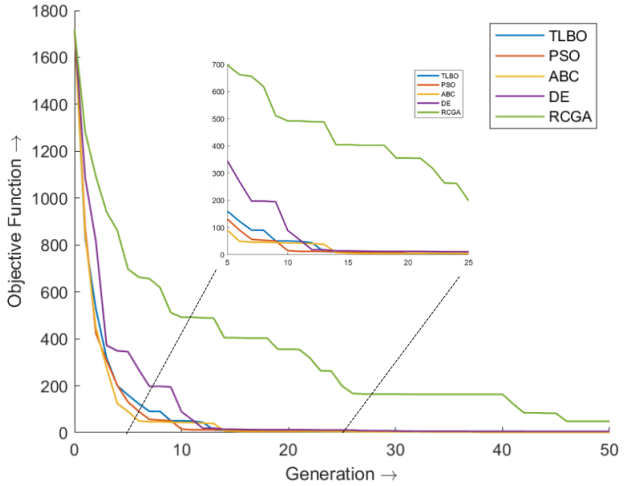

Fig. 7. Mean of Best Objective Function Values across 20 Runs

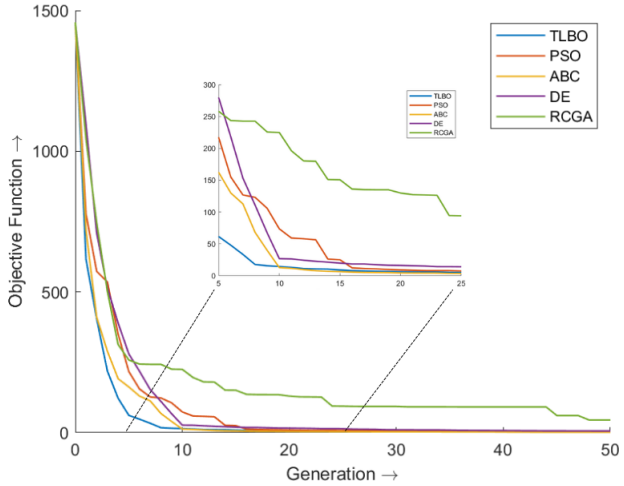

Fig. 8. Mean of Best Objective Function Values across 50 Runs

TABLE IV
LIST OF METAHEURISTIC TECHNIQUES USED

Algorithm	Parameters
TLBO	$N_p = 10$ $T = 50$ $\#FE = 1010$
PSO	$N_p = 10$ $T = 50$ $w = 0.7298$ $c_1 = 1.496$ $c_2 = 1.496$ $\#FE = 510$
DE/rand/1/binomial	$N_p = 10$ $T = 50$ $p_c = 0.8$ $F = 0.8$ $\#FE = 510$
GA (real-coded)	$N_p = 10$ $T = 50$ $\eta_c = 20$ $p_c = 0.8$ $\eta_m = 20$ $p_m = 0.2$ $\max \#FE = 510$
ABC	$N_p = 10$ $T = 50$ $limit = 500$ $(= N_p \cdot T)$ $\max \#FE = 1060$

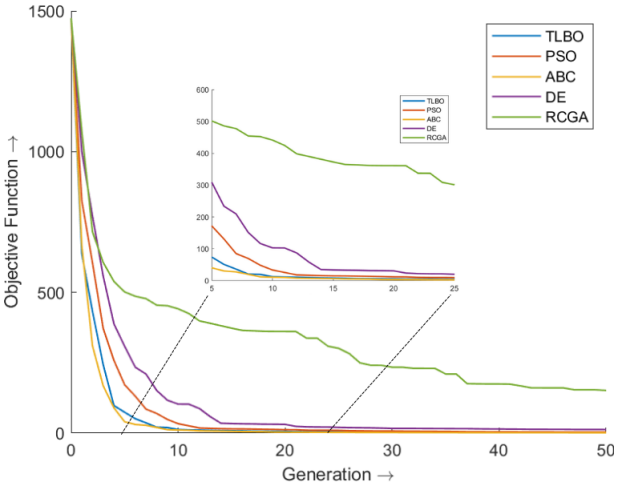

Fig. 9. Mean of Best Objective Function Values across 100 Runs