An Exploratory Goal-Based Modeling of Optimized Collision Avoidance Action Selection in Autonomous Vehicles

Rabia Rauf
Mirpur University of Science and Technology
College Rd, New Mirpur City, Azad Jammu and Kashmir 10250
Phone: +92 58279 61037
rabia.csit@must.edu.pk

Faisal Riaz
Mirpur University of Science and Technology
College Rd, New Mirpur City, Azad Jammu and Kashmir 10250
Phone: +92 58279 61037
pakistanfaisal.riaz@must.edu.pk

Saeed Ahmeed
Mirpur University of Science and Technology
College Rd, New Mirpur City, Azad Jammu and Kashmir 10250
Phone: +92 58279 61037
saeed.ntc@must.edu.pk

Adnan Sohail
IQRA University
Karachi, Sindh, Pakistan; Residential
Phone: +92-21-35310816, 35310826
adnan.sohail@iqraisb.edu.pk

Samia Abid
Mirpur University of Science and Technology
College Rd, New Mirpur City, Azad Jammu and Kashmir 10250
Phone: +92 58279 61037
samia.shah355@gmail.com

Saeeda Kouser
Mirpur University of Science and Technology
College Rd, New Mirpur City, Azad Jammu and Kashmir 10250
Phone: +92 58279 61037
saeeeda.csit@must.edu.pk

Asma Jabeen
Allama Iqbal Open University
Ashfaq Ahmed Rd, H-8/2 H 8/2 H-8, Islamabad, Islamabad Capital Territory 44000, Pakistan
Phone: +92 51 111 112 468
asma_jabeenajk@yahoo.com

Somyyia Akram
International Islamic University
H-10, Islamabad, Islamabad Capital Territory 44000, Pakistan
Phone: +92 51 9257988
somyyia@hotmail.com

Abstract

The collision-free path planning is crucial for an autonomous vehicle. It saves life and helps to complete the task in time. The computational intelligence mimics human intelligence and solves these types of problems in which conventional techniques fail to provide optimal solutions. In literature, computational intelligence techniques except evolutionary techniques have not been utilized for land vehicles. In this regard, we employ clonal selection algorithms for collision avoidance in an autonomous vehicle. To check the effectiveness of our proposed scheme, we

compare the performance of the clonal selection algorithm with the genetic algorithm. The results show that the clonal selection algorithm performs better than the genetic algorithm in terms of computation time and also avoids accident within the required time to avoidance.

Keywords: Collision Avoidance; Clonal Selection Algorithm; Agent Design.

1. Introduction

Casualties, injuries, and economic losses are the bi-product of road collisions. According to (Zhang et al., 2016), the rate of cranial casualties is very high in road accidents. Mostly young people become a victim of roadside injuries in urban areas (Koopmans, et al. 2015). In Australia, the annual cost of road traffic accident is $17b (Connelly and Supangan 2006). These accidents are mainly caused by humans. Hence, these collisions can be avoided by replacing humen with the autonomous vehicles (AVs). An AV can affect road safety in dense traffic (Fagnant and Kockelman 2015). In (Baskar, et al. 2011), it has been stated that the AVs can sense their surrounding environment, communicate with other vehicles and roadside infrastructure to get precise information and hence can avoid a collision. Due to their sensing ability, AVs can detect and track the obstacles to avoid collision The number of casualties can be further decreased by proper collision-free path planning (Liang, et al. 2015).

In literature, computational intelligence (CI) has been explored for better collison free path planning. Raiz et al. (Riaz, et al. 2015) have used the genetic algorithm (GA) for collision avoidance in AVs. In (Alejo, et al. 2013), particle swarm optimization technique has been used to plan a collision free path for autonomous unmanned aerial vehicles. In a research work, a fuzzy logic based collision avoidance steering controller has been presented for an autonomous surface vehicle (Hodge and Trabia 1999). CI-based algorithms have been used for underwater and unmanned arial vehicles. A very few researchers have used CI-based algorithms for path planning in land AVs. The purpose of our research work is to model CABC (Cognitive agent based computing) goal-based agent for accident avoidance. Agent-based modeling (ABM) and complex networks (CN) are two popular modeling tools for the understanding complex adaptive system (CAS). A unified framework known as CABC combining these two modeling paradigms was for the better understanding of CAS (Niazi and Hussain 2012). Hence, we justify the need for efficient optimization algorithm to perform efficient collision avoidance. The collision avoidance depends on a number of events as shown in figure 1. To avoid collision, a driver should be timely aware of risk, get alert on the life-threatening situation, timely decide and perform actions to avoid a collision.

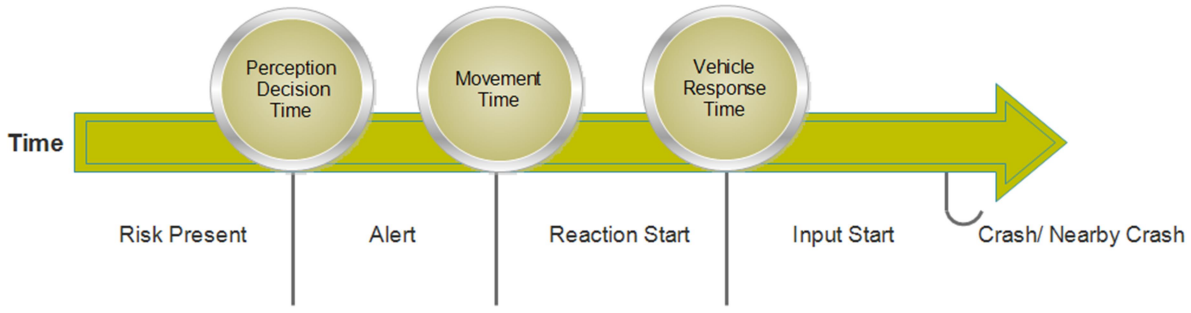

Figure 1. A time line for collision avoidance

To address the above mentioned problems, we have proposed goal based AV which follows the timeline for collision avoidance as in figure 2. The main goal of AV is safety. Our goal-based agent observes world state. On life-threatening situation run a clonal selection algorithm to achieve safety goal. The clonal selection algorithm timely provides optimal actions to avoid a collision. Our agent communicates these actions to the motor model to avoid a collision.

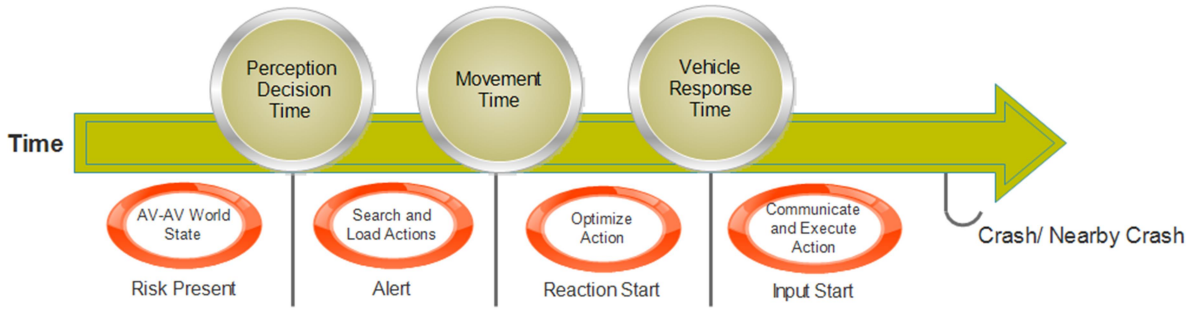

Figure 2. Time series line of collision avoidance in GBAV

Contribution: The contributions of this paper are as follows:

- First of all, we have implemented genetic algorithm and clonal selection algorithm for three types of congestions named as low, average and high congestions which has not been taken into account in [7].
- Secondly, we have employed exploratory agent-based modeling level of CABC framework to model and implement a clonal selection algorithm to make an efficient collision avoidance decision making for the AVs.

The rest of the paper is organized as follows. Section 2 elaborates the research methodology. The literature review is discussed in Section 3. The proposed solution is given in the 4th Section. Section 5 and 6 covers the algorithms and the UML design. The simulation results are given in Section 7. While, Section 8 concludes the work.

2. Method

The detailed literature review of the computational algorithm has been performed to avoid collision in AVs. Later, we design the architecture of the goal-based agent. Moreover, the UML design for the goal-based agent has been provided for the AV. The simulations are performed in C#. To evaluate the performance of the proposed scheme, the results of the clonal selection algorithm (CSA) are compared with GA. Finally, the conclusions are drawn from the observation of results.

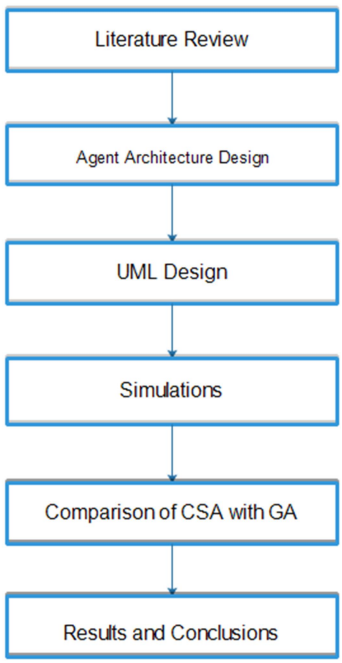

Figure 3. The Proposed Methodology

3. Literature Review

During the last couple of decades, the researchers from robotics and automotive engineering have done a lot of research to avoid collision between AVs. We categorize the literature review into two main threads. In the first thread, we discuss the application of GA for a collision free path planning. In this regard, Ahmed et al. (Ahmed and Deb 2013) have proposed an evolutionary multi-objective optimization (EMO) algorithm for multi-objective collision free path planning of an AV. The primary objective of the proposed path planning technique is to avoid collision between vehicles in the shortest path while the secondary objective is the smooth path. The smooth path objective helps in making decision when there is more than one better solution. EMO is the modification of multi-objective NSGA-II algorithm. For the path planning, a three-path representation scheme named as binary coded, mixed code and integer code has been applied. The performance of EMO has been compared with simple GA in a difference scenario. The performance of GA decreases as the number of obstacles increases. The authors noticed that EMO performs well in dense obstacle environment with multiple objectives. The results show that the proposed algorithm performs efficiently and robustly in 90% dense grid. The multi-objective evolutionary technique with the integer code scheme can be useful for non-monotonic path planning in the real world. Mahmoud Zadeh and A. Yazdani (MahmoudZadeh, et al. 2016) have proposed a differential evolutionary (DE) algorithm for collision-free path planning in underwater environments. The proposed Differential Evolutionary (DE) algorithm is the improved version GA as it allows to set floating point as a parameter. Experiment analysis proved that DE provides collison free path in less computational time for autonomous underwater vehicles (AUV) with kinematic variables. Riaz et al. (Riaz, et al. 2015), have employed the GA to avoid collision between AVs. A chromosome consists of four genes named as speed, brake, angle, and time of avoidance (TOA). To initiate the value of genes the following distance values have been given to the algorithm such as rear end distance, forward end distance, left and right distances between vehicles. For fitness calculation, the fitness of each gene is subtracted from the desired solution. They have used Roulette Wheel selection method to select the fittest gene. The crossover and mutation operations have been used to produce offsprings which represent the fittest solution of the problem. For UAVs, the conflict detection and resolution technique for safe path planning has been presented by Alejo et al. (Alejo, et al. 2016). They identified the problem that the UAVs can collide with other UAVs and static objects in the airspace. To address this problem, the authors have employed GA which helps to resolve the conflict. They have used distance, speed and heading as parameters for a safe waypoint. After applying crossover and mutation, the GA selects the best collision free path for UAV. This algorithm has been used for different wind conditions. The execution time of GA depends on various wind condition. However, it produces an optimal collision-free path in less computational time.

While in the second thread, we discuss the application of CSA for a collision free path planning of AVs. It has been discussed in (Lee, et al. 2007), that the clonal selection has been applied for a smooth path selection for underwater AVs. They state that this selection method optimizes the coefficient for proportional-integral derivative (PID) controller. This controller intelligently controls the vehicle movement in an unpredictable water environment. A technique for path planning of airship in three directions has been presented in (Liu, et al. 2008). For the optimal flight and performance parameter selection of airship, the CSA has been employed by the authors. This algorithm has been applied in three cases and provides optimal solution for flight planning in 30-50 iterations. Furthermore, the CSA has been employed for multi-combat dynamic ariel weapon-target assignment game model (Wang, et al. 2016). The combat is divided into two parties named as red and blue. This algorithm identifies the weapon constraints, the number of aerial combat steps, damage probability and prediction strategies. It takes into account both the diversity and convergence speed. To check the efficiency of the proposed scheme, it has been compared with GA. It has been noticed that the GA fails to provide an optimal solution in 100 iterations while CSA provides an efficient and real-time solution. Another work in (Das, et al. 2016), employs the same algorithm for cooperative motion control of AUVs. The model of AUVs was designed on the loader

follower strategy. The leading AUV estimates the position and announces to another vehicle. Basically, this model consists of two parts, one is the team controller while the other is the AUV controller. The former implements the clonal selection to generate waypoints while the later generates a possible trajectory for AUVs. To check the efficiency of this technique, its results have been compared with other search methods named as multi-directional, stochastic approximation and simple techniques.

4. Proposed Solution

In this section, a goal-based agent design and the formulation for action selection has been proposed for collision-free path planning of an AV.

4.1. Goal-based agent architecture

The goal-based agent architecture has three components as shown in figure 4. It takes different environment situations from the sensor and input them to the CSA optimizer. In order to avoid collision, this optimizer runs the clonal selection algorithm to generate optimized action for the motor module.

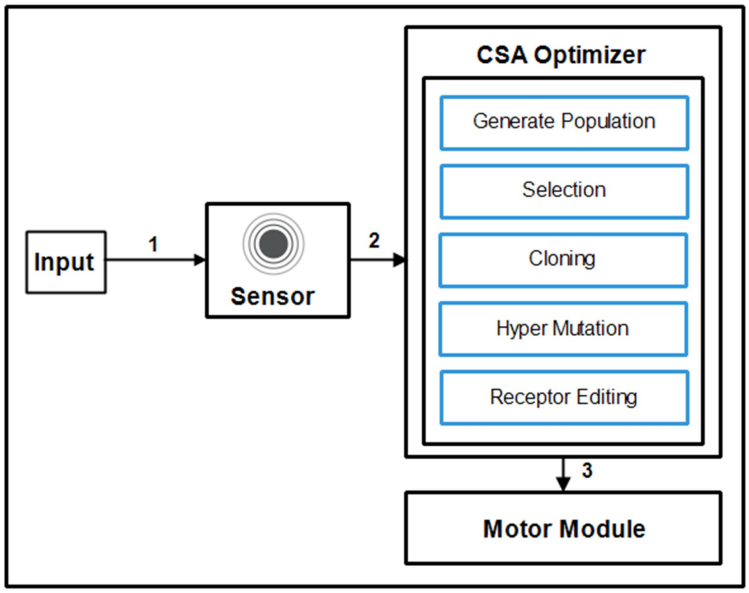

Figure 4. Agent Architecture

Components Description:

1. Sensor: It senses left, right, front rear distance from the environment.
2. CSA optimizer: It selects optimal action by running clonal selection algorithm and passes the optimized action to the motor module.
3. Motor module: It executes the optimized action for collision avoidance.

4.2. Clonal selection and Affinity function

Initially, the population of 100 antibodies is generated. The value of antibodies is randomly generated according to the congestion rate. We have divided road congestion into three categories named low, average and high congestion. In order to produce a good solution, we apply affinity function. The antibodies with the high affinity are added to the next generation. For the selection of the best antibodies to the next generation, we use the roulette wheel selection method. Through this method, the antibodies are selected according to their probability. While, the probability of each antibody is calculated using equation 1:

$$p(i) = \frac{A(i)}{\sum_{j=1}^{N} A(i)} \quad (1)$$

Where p(i) represents the probability of each antibody while A(i) represents the affinity of the individual antibodies in the population. After the process of selection, cloning is performed on the selected antibodies. At last, the clones are mutated. By applying the roulette wheel there is a chance of high-affinity antibodies to add to the next population.

4.2.1. Affinity function

Each antibody constitutes four cells named as speed, brake, angle, and time to avoidance (TTA).

Speed:

The affinity of speed is calculated using the following equation:

Difference (d)= $Required\ speeed - Speed\ in\ antibody$ (2)

Equation (2) provides the absolute value of the difference between the required speed and the speed in the antibody. After calculating the difference, a variable 'e' is declared whose value is between the ranges of (60-120) speed. When the value of speed is less than variable 'e' the affinity will be calculated using equation (3)

$$.A(S) = x * \left(\frac{D}{e}\right) \quad (3)$$

Where A(S) represents the affinity of speed and x is the weight of parameters in antibodies. To provide equal weights 25 value are assigned when d>e then the following equation will be used.

$$A(S) = x \quad (4)$$

Angle

The affinity of angle is calculated using the following equations:

$Difference(d) = Required\ inner\ tire\ angle -\ inner\ tire\ angle\ in\ antibody$ (5)

In angle, consider a variable 'n' which should lie within the limit of the angle which is (60-120). When 'd' is smaller than 'p' the affinity of inner tire angle is calculated using equation (6).

$$A(athe\ ) = x * \left(\frac{D}{e}\right) \quad (6)$$

Otherwise, it will be calculated using equation (7).

$$A(a) = x \quad (7)$$

Break:

The affinity function for a break is zero, and one.

Difference (d) = $Required\ break - break\ in\ antibody$ (8)

When the difference is not equal to zero it will be set as 25. In case, the difference match the required break then it will be zero.

Time to avoidance (TTA)

Difference (d)= $Required\ TTA - TTA\ in\ antibody$ (9)

Consider a variable 't' whose value can be any number within the limit (20-90). When the value of 'd' is smaller than 't', the affinity will be calculated as follows.

$$A(TTA) = x * \left(\frac{D}{t}\right) \quad (10)$$

Otherwise, it will be calculated as follows:

$A(TTA) = \text{x}$ (11)

The value of x is 25 which represents the weight for the antibody.

Total Affinity

The total affinity is calculated by using equation (12):

$$[100 - [Affinity[speed] + Affinity[break] + Affienity\ [Angle] + Affinity[TTA]]]]$$

(12)

5. CSA for Collision Avoidance

As discussed earlier, we use CSA to get optimized actions to avoid collision. The CSA is based on the immune system and it describes how the immune system responds to the antigen (Burnet 1959).

Algorithm 1: Action Optimization

```
Set random values of Vehicle Distances front, rear, lef, and right (to
generate real-life scenario).

          IF (VehicalDis_front> VehicalDis_rear)
                Vehical_Speed=Vechical_Speed + 20.
                Vehical_Break=0.
          Else

               IF (VehicalDis_rear>20)
                  Vehical_Break= 1.
                  Vehical_Speed=Vechical_Speed - 20.
              Else
               Vehical_Speed=Vechical_Speed - 10.
               Vehical_Break=0;

        IF (VehicalDis_left> VehicalDis_right)
           Vehical_Angle= Vehical_Angle + 20.
            Vehical_tta= Vehical_tta+10.
        Else
           Vehical_Angle= Vehical_Angle - 20.
           Vehical_tta= Vehical_tta - 10.}

Randomly initiate population of antibodies.
Calculate Affinity of each element of population.
Select elements with High Affinity.
Perform cloning by generating copies of these.
Mutate all copies.
Add mutated individual to the population and reselect from mutating.
Repeat step 20 to 25 until termination criteria meet.
```

6. The UML Design of Simulations:

This section covers the UML design of our simulations. First use case diagram is shown in figure 5. It consists of five use cases. In the first use case, the agent initiates perception to interpret the environment. After integrating the current scenario, it searches the best action from the search space. While in the last use case, the agent forwards the optimized actions to a motor module.

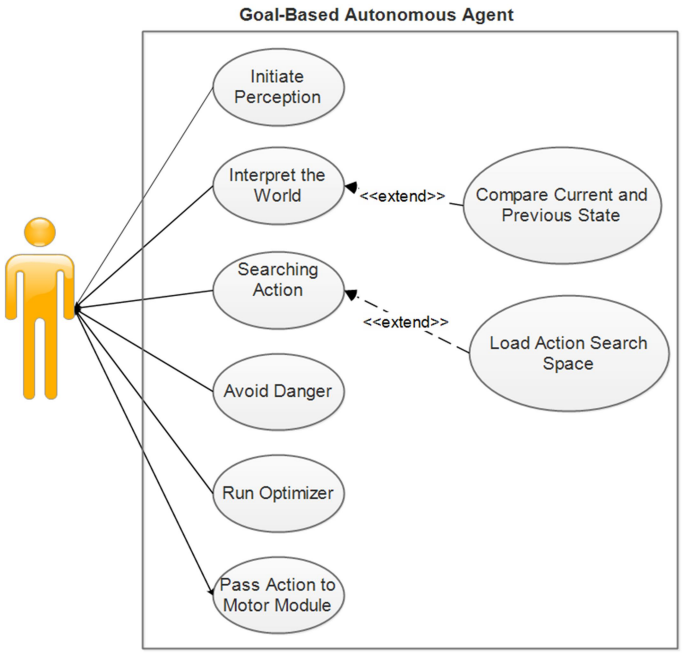

Figure 5. Use case diagram of simulation

A sequence diagram is shown in figure 6 which represents the actions that are performed in sequence. It has six modules named as sensors, goal-based AV, action selector, optimizer, controller and motor.

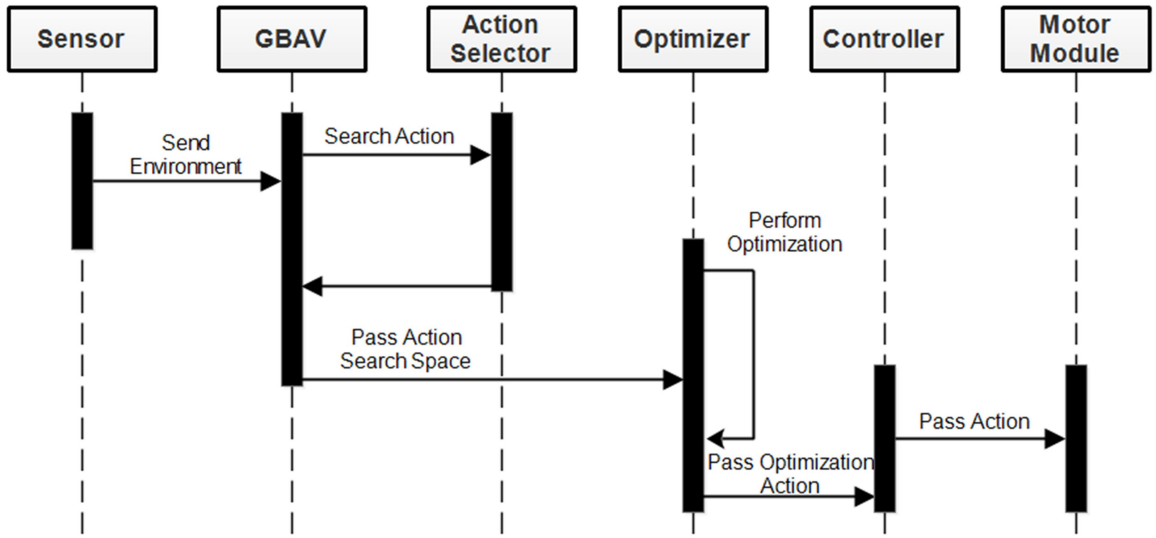

Figure 6. Sequence diagram of simulation

7. Simulations and Results

This section provides a detailed discussion of the simulations and their results. However, first we discuss the simulation environment. We have performed our simulations in C# with a different type of road congestion. After the simulation design, we discuss the simulation parameters. These parameters represent the cell structure of antibodies with their ranges. At the end of this section, we provide the test case design of our methodology.

7.1. Simulation Environment

Our scheme is implemented in C# .net platform as shown in figure 7. Three different types of congestion rates are presented as low, average and high congestion. The number of vehicles

varies on different congestion rates. After selecting the congestion rate, the simulator runs CSA and GA against ideal discussion. The simulator gives an optimized result computed from CSA and GA. It also computes the computation time taken by CSA and GA.

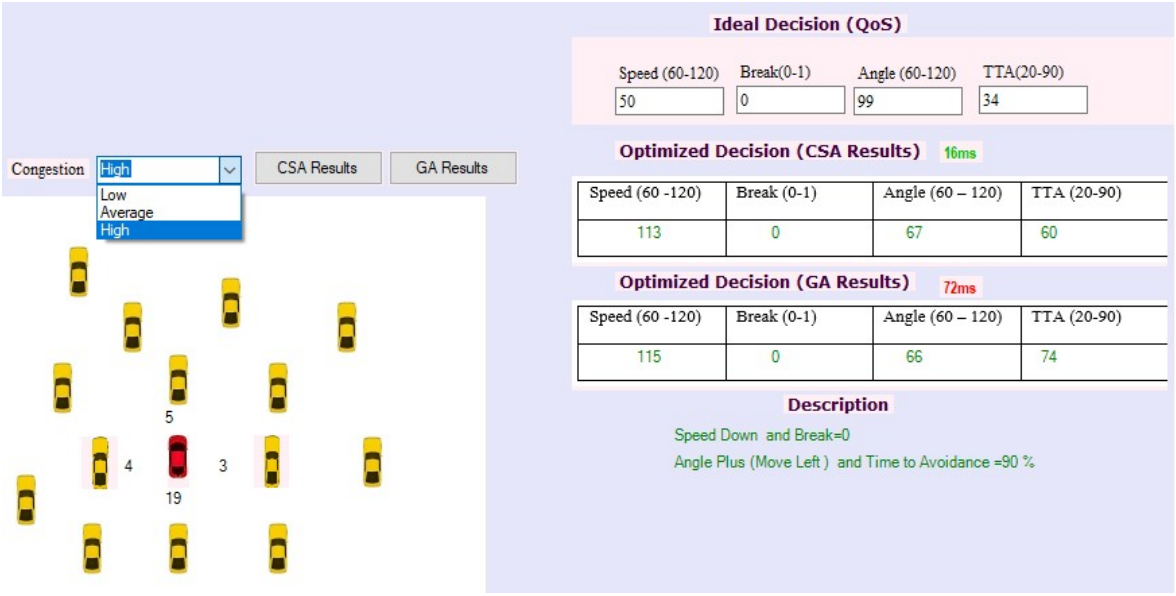

Figure 7. Simulation interface

7.2. Simulation Parameters

Our simulation is designed for motorway traffic. The antibodies in our research are divided into four cells where each cell has a different set of values. The decision making depends on the values of these cells. For simplicity, we assign the break value 0 and 1. The value zero means deceleration and 1 means acceleration**.**

Table 1. Structure of antibody

Series	**Cells**	**Range**
1	Speed	60-120
2	Angle	60-120
3	Break	0-1
4	Time To Avoidance (TTA)	20_90%

7.3. Test Case Design:

We design the test cases for three types of congestions on the road. These congestions are names as low, average and high congestion. The number of cars varies with different congestion rates. For each scenario, we have performed 100 tests resulting in 600 tests in total. We apply CSA and GA and compare their performance to check which algorithms performs the best.

7.4. Results

As discussed earlier, 100 test cases are designed for low, average and high congestion. To check the performance of CSA, several scenarios are generated. These test cases run at different speeds as shown in table 2. A number of ideal decisions are given to the simulator and the simulator optimized them using the CSA. In table 2, ideal decisions are given for low congestion.

Table 2. Ideal decision in low congestion

Ideal test cases				
	SPEED (Km/h)	BREAK	ANGLE	TTA
1	96	1	96	72
2	87	1	86	67
3	135	0	121	88
4	128	0	133	67

5	96	1	131	83
6	84	1	97	72
7	84	1	131	83
8	90	1	129	97
9	126	0	134	90
10	84	1	82	61

Table 3. Optimized result by CSA in low congestion

	CSA optimized result			
	SPEED (Km/h)	BREAK	ANGLE	TTA
1	81 ± 17	1 ± 0	91 ± 18	62 ± 21
2	86 ± 16	1 ± 0	80 ± 18	55 ± 22
3	89 ± 15	0 ± 0	81 ± 18	54 ± 17
4	99 ± 11	0 ± 0	88 ± 14	40 ± 25
5	94 ± 16	1 ± 0	81 ± 23	52 ± 16
6	94 ± 16	1 ± 0	86 ± 21	59 ± 29
7	87 ± 16	1 ± 0	96 ± 18	55 ± 17
8	83 ± 14	1 ± 0	9 4 ± 16	48 ± 19
9	88 ± 16	0 ± 0	93 ± 17	56 ± 14
10	88 ± 16	1 ± 0	88 ± 23	52 ± 23

Table 3 shows the optimized result of CSA in case of low congestion. The test case was performed at a maximum speed of 135 and a minimum of 84. Results in table 3 validate the functionality of CSA by giving maximum variation in speed of 17 and minimum of 14, maximum variation in angle is 23 and the minimum variation is 14, maximum variation is 29 and the minimum variation is 14 in TTA. Further, table 4 provides the ideal decision for average congestion.

Table 4. Ideal decision in average congestion

Ideal test cases				
	SPEED (Km/h)	BREAK	ANGLE	TTA
1	89	0	65	28
2	71	1	119	43
3	77	0	101	68
4	72	1	118	72
5	83	0	65	52
6	106	0	68	61
7	89	0	69	63
8	77	1	64	62
9	62	1	75	34
10	109	0	108	43

Table 5. Optimized result by CSA in average congestion

	SPEED (Km/h)	BREAK	ANGLE	TTA
1	9.1 ± 8.1	0 ± 0	67.2 ± 6.1	28.1 ± 4.17
2	71.1 ± 6.0	1 ± 0	115 ± 8.15	41.2 ± 9
3	76.3 ± 4.52	0 ± 0	90.1 ± 5.91	67.5 ± 3.86
4	74.3 ± 5.56	1 ± 0	111.2 ± 6.64	73.5 ± 9.22

5	82.3 ± 6.6	0 ± 0	65.8 ± 3.7	51 ± 7.1
6	103 ± 4	0 ± 0	65 ± 3.4	63.6 ± 4.7
7	90.2 ± 6.4	0 ± 0	68± 8	64.9 ± 4.6
8	77.8 ± 5.8	1 ± 0	67.7 ± 5.35	65.1 ± 8.7
9	63.8 ± 4.89	1 ± 0	73.3 ± 6.4	32.1 ± 9
10	106.6 ± 5.3	0 ± 0	110 ± 5.72	41.9 ± 5.7

Table 5 shows the optimized results of CSA in case of average congestion. The test case was performed at the maximum speed of 109 and a minimum of 62. Results in table 5 validate the functionality of CSA by giving maximum variation in speed of 8 and a minimum of 4, maximum variation in angle is 8.15 while the minimum variation 3.4, maximum variation 9.22 and the minimum variation 4.1 in TTA.

Table 6. Ideal decision in high congestion

. Ideal test cases				
	SPEED (Km/h)	BREAK	ANGLE	TTA
1	57	0	87	41
2	44	1	53	15
3	83	0	90	30
4	57	0	94	33
5	99	0	84	35
6	96	0	87	47
7	99	0	82	43
8	63	0	94	43
9	84	0	81	45
10	89	0	86	30

Table 7. Optimized result by CSA in high congestion

	CSA optimized result			
	SPEED (Km/h)	BREAK	ANGLE	TTA
1	62.5 ± 3.5	0 ± 0	87 ± 7.8	40.1 ± 6.5
2	60 ± 13	1 ± 0	66.5 ± 3.5	24.5 ± 6.5
3	83.9 ± 17	0 ± 0	62.5 ± 11.6	53 ± 13
4	90 ±14	0 ± 0	81.8 ± 14	51.8 ± 20
5	89 ± 17	0 ± 0	41 ± 16	51.6 ± 27
6	83 ± 18	0 ± 0	81.7 ± 11	53.5 ± 26
7	90.9 ±13	0 ± 0	83.3 ± 15	55.7 ± 18
8	89 ± 14	0 ± 0	49.9 ± 14	47 ± 4
9	99 ±19	0 ± 0	87 ±20	44.5 ± 18.5
10	87.4 ±21	0 ± 0	82.6 ± 29	12 ± .6

Table 7 shows the optimized result of CSA in case of high congestion. Test cases were performed at a maximum speed of 99 and a minimum of 44. Results in table 7 validate the functionality of CSA by giving maximum variation in speed of 21 and a minimum of 3.5, maximum variation in angle is 7.8 and minimum 3.5, maximum 27 and minimum 0.6 in TTA.

In order to check the computational time of CSA, we run both CSA and GA on core i3 processor with 4GB RAM as shown in table 8. We have observed that the CSA takes average maximum time of 28.9 millisecond and minimum average time 14.3 milliseconds. However, GA takes maximum average time of 85.5 millisecond and minimum average time 71.4 in case of low congestion. In case of average congestion, CSA takes a maximum of 15.4 and a minimum of 13 milliseconds for computation. While, GA takes a maximum of 72.2 ms and minimum

computational time of 65.4ms. On the other hand, CSA takes a maximum of 18.4ms for providing an optimized decision for collision avoidance. GA takes a maximum of 82.6ms and a minimum of 55.4ms for providing an optimal decision in high congestion.

Table 8. The computational time of CSA and GA in low, average and high congestion.

Low Congestion			**Average Congestion**			**High Congestion**		
	Time in millisecond			**Time in ms**			**Time in ms**	
Test case	**CSA**	**GA**	Test case	**CSA**	**GA**	Test case	**CSA**	**GA**
1	28.9	77.9	1	15.4	65.8	1	18.4	82.6
2	16.8	75.2	2	13.1	64.6	2	14.8	74.7
3	15.1	78.3	3	13.7	65.4	3	13	56.5
4	14.7	71.4	4	12.9	69.8	4	12.6	54.9
5	14.3	71.4	5	13.4	72.2	5	12.5	54.9
6	14.3	77.8	6	13	67.8	6	12.4	54.9
7	17.4	76.8	7	13.3	67	7	12.3	56.4
8	16.8	85.3	8	13.1	69.4	8	14.8	55.7
9	21.7	83.7	9	13.1	66.5	9	12.7	54.5
10	16.4	72	10	13.2	65.9	10	12.2	55.4

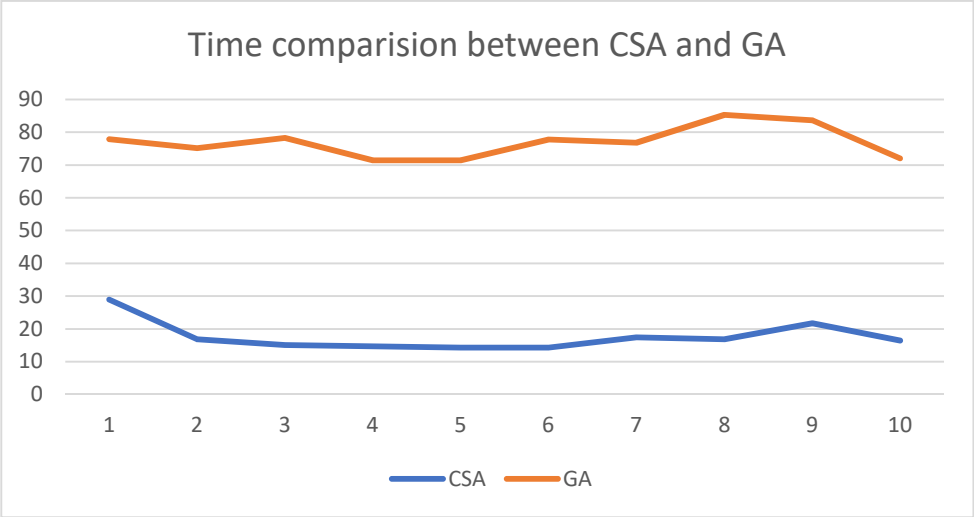

Figure 8. Comparing CSA and GA in low congestion

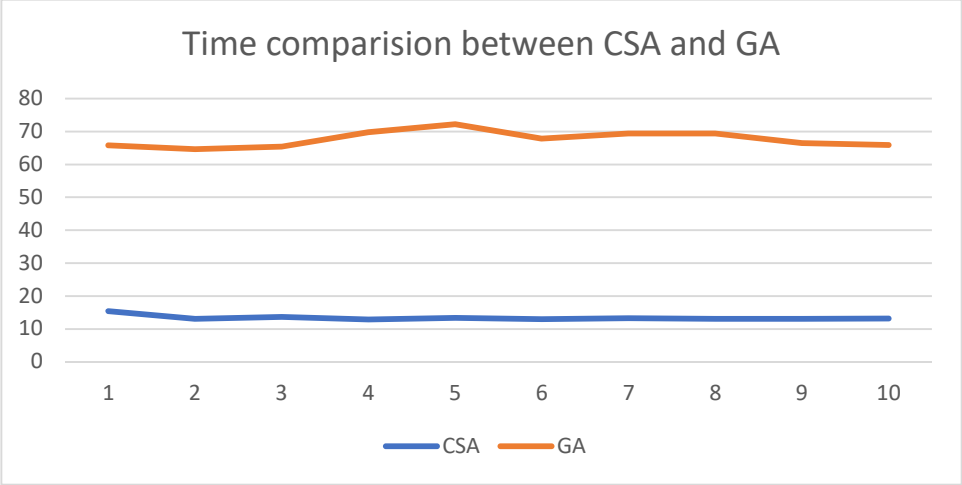

Figure 9. Comparing CSA and GA in average congestion

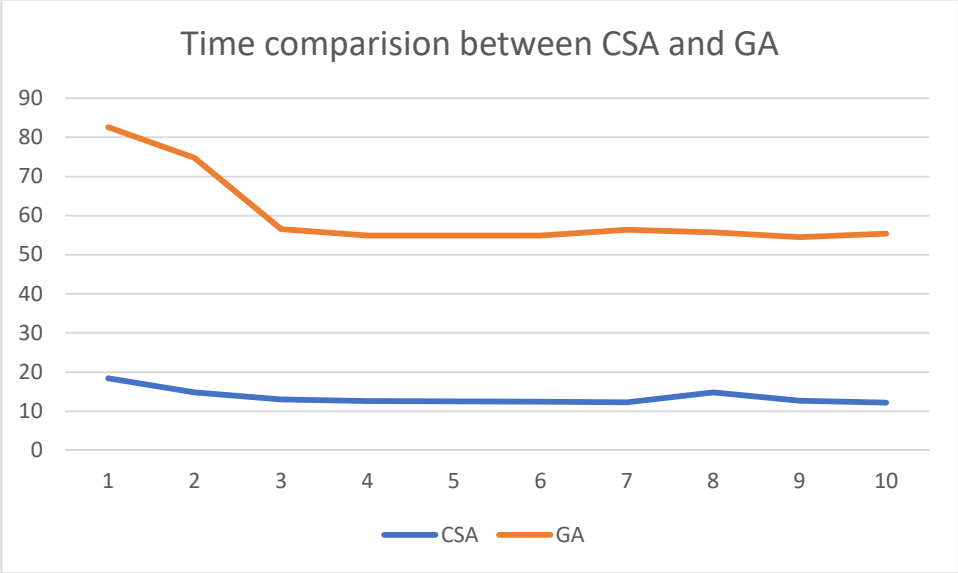

Figure 10. Comparing CSA and GA in high congestion

Thus, the computational time of CSA is less than the computational time of GA as shown in the figures 8-10. Hence, CSA performs better than GA in terms of computational time.

Now, we check whether CSA avoids accident within required TTA. As shown in table 8, 9 and 10 CSA finds an optimized solution within the required TTA. GA fails to provide a solution in the required TTA. GA provides a better solution than CSA but it fails to provide it with the required TTA time. Thus, an accident can occur. The CSA is better than GA in terms of computational time.

Table 9. Comparing GA and CSA in term of TTA for Low congestion

Test case	**Required TTA (ms)**	**CSA**		**GA**	
1	72	28.9	Avoided	77.9	Not avoided
2	67	16.8	Avoided	75.2	Not avoided
3	88	15.1	Avoided	78.3	Not avoided
4	67	14.7	Avoided	71.4	Not avoided
5	83	14.3	Avoided	71.4	Avoided
6	72	14.3	Avoided	77.8	Not avoided
7	83	17.4	Avoided	76.8	Avoided

8	97	16.8	Avoided	85.3	Avoided
9	90	21.7	Avoided	83.7	Avoided
10	61	16.4	Avoided	72	Not avoided

Table 10. Comparing GA and CSA in term of TTA for average congestion

Test case	**Required TTA (ms)**	**CSA**		**GA**	
1	28	15.4	Avoided	65.8	Not avoided
2	43	13.1	Avoided	64.6	Not avoided
3	68	13.7	Avoided	65.4	Not avoided
4	72	12.9	Avoided	69.8	Not avoided
5	52	13.4	Avoided	72.2	Not avoided
6	61	13	Avoided	67.8	Not avoided
7	63	13.3	Avoided	69.4	Not avoided
8	62	13.1	Avoided	69.4	Not avoided
9	34	13.1	Avoided	66.5	Not avoided
10	43	13.2	Avoided	65.9	Not avoided

Table 11. Comparing GA and CSA in term of TTA for High congestion

Test case	**Required TTA (ms)**	**CSA**		**GA**	
1	41	18.4	Avoided	82.6	Not avoided
2	15	14.8	Avoided	74.7	Not avoided
3	30	13	Avoided	56.5	Not avoided
4	33	12.6	Avoided	54.9	Not avoided
5	35	12.5	Avoided	54.9	Not avoided
6	47	12.4	Avoided	54.9	Not avoided
7	43	12.3	Avoided	56.4	Not avoided
8	43	14.8	Avoided	55.7	Not avoided
9	45	12.7	Avoided	54.5	Not avoided
10	30	12.2	Avoided	55.4	Not avoided

In the aforementioned discussion, we have examined the performance of CSA and GA. From the results, we conclude that the clonal selection algorithm is less robust than GA. By comparison, it is clear that the standard deviation of CSA is higher than the standard deviation of GA. But CSA beat GA in computation time. We have only examined CSA with GA, because to the best of our knowledge, CI-based algorithms have not been employed for collision avoidance in a land vehicle. There may be a possibility that other CI-based algorithms will perform better than GA or we can also create a hybrid algorithm which can combine the best features of CI-based algorithms to timely avoid collision with better-optimized results. In the future, we will extend our research work by hard wiring the CSA to check its performance in the real life. We also explore other CI-based algorithms and try to negotiate why their application has not been done for collision avoidance in the land vehicles.

8. Conclusion

From an extensive literature review of computational intelligence CI, we found that very few researchers use CI-based algorithm for land autonomous vehicles. We have implemented CI based clonal selection algorithm with a CABC framework for collision avoidance in the AVs. We have implemented CSA and GA in different types of congestions. To validate the performance of CSA it is compared with GA. GA provides better results than CSA. However, the computation time of CSA is less than the GA. Moreover, CSA performed better results than GA in terms of TTA because CSA has producesd optimal results with the required TTA. Whereas, GA failed to provide

optimal results within the required TTA. Thus, we conclude that CSA provides the required action with TTA and avoids a collsion.

References

Ahmed, F., & Deb, K. (2012). Multi-objective optimal path planning using elitist non-dominated sorting genetic algorithms. *Soft Computing*, *17*(7), 1283-1299.

Alejo, D., Cobano, J. A., Heredia, G., & Ollero, A. (2013). Particle swarm optimization for collision-free 4d trajectory planning in unmanned aerial vehicles. *International Conference on Unmanned Aircraft Systems,* 298-307.

Alejo, D., Cobano, J. A., Heredia, G., & Ollero, A. (2016). An efficient method for multi-UAV conflict detection and resolution under uncertainties. *Second Iberian Robotics Conference,* 635-647.

Baskar, L. D., De Schutter, B., Hellendoorn, J., & Papp, Z. (2011). Traffic control and intelligent vehicle highway systems: a survey. *IET Intelligent Transport Systems*, 5(1), 38-52.

Connelly, L. B., & Supangan, R. (2006). The economic costs of road traffic crashes: Australia, states and territories. *Accident Analysis & Prevention*, *38*(6), 1087-1093.

Emmanuel, I. (2017). Fuzzy Logic-Based Control for Autonomous Vehicle: A Survey. *International Journal of Education and Management Engineering*, *7*(2), 41.

Hodge, N. E., & Trabia, M. B. (1999). Steering fuzzy logic controller for an autonomous vehicle. In Proceedings 1999 IEEE International Conference on Robotics and Automation, 3, 2482-2488.

Koopmans, J., Friedman, L., Kwon, S., & Sheehan, K. (2015). Urban crash-related child pedestrian injury incidence and characteristics associated with injury severity. *Accident Analysis & Prevention*, *77*, 127-136.

Lee, J., Roh, M., Lee, J., & Lee, D. (2007). Clonal selection algorithms for 6-DOF PID control of autonomous underwater vehicles. In International Conference on Artificial Immune Systems (pp. 182-190). Springer, Berlin, Heidelberg.

Xin, Y., Liang, H., Mei, T., Huang, R., Chen, J., Zhao, P., ... & Wu, Y. (2015). A New Dynamic obstacle Collision Avoidance System for Autonomous Vehicles. *International Journal of Robotics & Automation*, *30*(3), 278-288.

Liu, Y., Zhang, Y., & Hu, Y. (2008, April). Optimal path planning for autonomous airship based on clonal selection and direct collocation algorithm. In *2008 IEEE International Conference on Networking, Sensing and Control* (pp. 1828-1832).

MahmoudZadeh, S., Powers, D., Sammut, K., & Yazdani, A. M. (2016). Differential evolution for efficient AUV path planning in time variant uncertain underwater environment. *Robotics (cs. RO)*.

Niazi, M. A., & Hussain, A. (2012). *Cognitive agent-based computing-I: a unified framework for modeling complex adaptive systems using agent-based & complex network-based methods*. Springer Science & Business Media.

Riaz, F., Niazi, M. A., Sajid, M., Amin, S., Ratyal, N. I., & Butt, F. (2015). An efficient collision avoidance scheme for autonomous vehicles using genetic algorithm. *J Appl Environ Biol Sci*, *5*(8), 70-76.

Schultz, A., & Grefenstette, J. (1992, August). Using a genetic algorithm to learn behaviors for autonomous vehicles. In *Guidance, Navigation and Control Conference* (p. 4463).

Wang, Y., Zhang, W., & Li, Y. (2016, July). An efficient clonal selection lgorithm to solve dynamicweapon-target assignment game model in UAV cooperative aerial combat. In 2016 35th Chinese Control Conference (CCC) (pp. 9578-9581).