Graduate School Form 30
(Revised 08/14)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Ali Gholamjafari

Entitled
A GENETIC ALGORITHM APPROACH TO BEST SCENARIO SELECTION FOR PERFORMANCE EVALUATION OF VEHICLE ACTIVE SAFETY SYSTEMS

For the degree of Master of Science in Electrical and Computer Engineering

Is approved by the final examining committee:

Lingxi Li

Maher Rizkalla

Brian King

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification/Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

Approved by Major Professor(s): Lingxi Li

Approved by: Brian King 07/18/2014

Head of the Department Graduate Program Date

A GENETIC ALGORITHM APPROACH TO BEST SCENARIOS SELECTION FOR PERFORMANCE EVALUATION OF VEHICLE ACTIVE SAFETY SYSTEMS

A Thesis

Submitted to the Faculty

of

Purdue University

by

Ali Gholamjafari

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Electrical and Computer Engineering

May 2015

Purdue University

Indianapolis, Indiana

First and foremost, I have to thank my parents for their love and support throughout my life. Thank you both for giving me strength to reach for the stars and chase my dreams. To all my friends, thank you for your understanding and encouragement in my many, many moments of crisis. Your friendship makes my life a wonderful experience. I cannot list all the names here, but you are always on my mind.

ACKNOWLEDGMENTS

I would like to thank Dr. Lingxi Li for his endless academic and financial support during my research and study, and provisions above the academic contexts. I would like to thank my committee members Dr. Rizkalla and Dr. King for their assistance and supervision in preparation of this thesis. Finally, I reserve special thanks for my family, for their boundless mental, emotional, and financial support during my hard work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Gholamjafari, Ali. MSECE, Purdue University, May 2015. A Genetic Algorithm Approach to Best Scenarios Selection for Performance Evaluation of Vehicle Active Safety Systems. Major Professor: Dr. Lingxi Li.

One of the most crucial tasks for Intelligent Transportation Systems is to enhance driving safety. During the past several years, active safety systems have been broadly studied and they have been playing a significant role in vehicular safety. Pedestrian Pre- Collision System (PCS) is a type of active safety systems which is used toward pedestrian safety. Such system utilizes camera, radar or a combination of both to detect the relative position of the pedestrians towards the vehicle. Based on the speed and direction of the car, position of the pedestrian, and other useful information, the systems can anticipate the collision/near-collision events and take proper actions to reduce the damage due to the potential accidents. The actions could be triggering the braking system to stop the car automatically or could be simply sending a warning signal to the driver depending on the type of the events.

We need to design proper testing scenarios, perform the vehicle testing, collect and analyze data to evaluate the performance of PCS systems. It is impossible though to test all possible accident scenarios due to the high cost of the experiments and the time limit. Therefore, a subset of complete testing scenarios (which is critical due to the different types of cost such as fatalities, social costs, the numbers of crashes, etc.) need to be considered instead. Note that selecting a subset of testing scenarios is equivalent to an optimization problem which is maximizing a cost function while satisfying a set of constraints. In this thesis, we develop an approach based on Genetic Algorithm to solve such optimization problems. We then utilize crash and field database to validate the accuracy of our algorithm. We show that our method is effective and robust, and runs much faster than exhaustive search algorithms. We also present some crucial testing scenarios as the result of our approach, which can be used in PCS field testing.

1. INTRODUCTION

1.1 Choosing a Subset of Accident Scenarios to Evaluate the Performance of Pre-collision System

Recently, the science, engineering, and technology areas have given a great deal of attention to vehicular active safety systems [1], [2], [3]. These systems are able to predict potential crash/near-crash events by taking advantage of advanced sensing/ communication/ control techniques. The systems then take appropriate actions by sending a warning signal to the driver and/or automatically braking, thus preventing and/or reducing the cost of damage caused by potential accidents. Some of these systems are specifically designed for enhancing pedestrian safety. They achieve this goal by using the data collected by radar and camera to determine the position of the pedestrian and use an advanced algorithm to estimate the crash and near-crash events. Much research has been done in this area (see, for instance, [4], [5], [6], [7], [8], [9], [10], [11] and references therein) and many pedestrian safety system-related products have been on market and occupied in vehicles such as the pre-collision system [12], the crash imminent braking system [13], the pedestrian and cyclist detection system [14], and others.

These advanced pedestrian safety systems have undoubtedly improved driving safety. However, the performance of these systems, even for the same accident scenarios, are significantly different. Currently, there is no common standard in order to evaluate, compare, and improve the performance of these systems. In general, in order to evaluate their performance, we need to design and implement accident test scenarios, perform the experiment in the testing field, collect the data, and analyze it to see how the system performs for certain accident scenarios.

Currently, the Transportation Active Safety Institute (TASI) at Indiana University-Purdue University Indianapolis (IUPUI), with the support of Collaborative Safety Research Center (CSRC) of Toyota, is conducting independent research on generating test scenarios and procedures for evaluating pedestrian Pre-Collision System (PCS). The general approach adopted in this study is to use the recent statistical US pedestrian crash data for 2010 and 2011, State pedestrian crash data, and the data from the IUPUI TASI 110-Car Naturalistic Driving Study [15] to create test scenarios. Understandably, it is impossible to test every scenario for performance evaluation due to the time and cost of vehicle field testing. Therefore, a critical subset of test scenarios (that capture key crash parameters) must be obtained for vehicle testing.

1.2 Different Methods of Choosing a Subset of Accident Scenarios

Based on different criteria, different scenarios can be chosen to form the suitable subset of accident scenarios to study. One way is based on the cost of the accidents. From the crash databases, there are three different types of costs, namely, the crashes, fatalities, and social costs. Clearly, different types of costs might lead to different testing scenarios to be the candidates of the cost-based subset. For instance, the scenarios shown in Table 1.1 are the highest-cost scenarios based on the costs of the crashes and social costs. According to these scenarios, when the cost is in terms of crashes, maximum cost is for the case that the light condition is Day-light whereas when the cost is in terms of social costs, it is maximum when the light condition is Dark-Lit.

Table 1.1.: Top Scenarios Based on different costs

	Pedestrian Action Standing	Vehicle Action	Light	P_ mann	P_Speed	Crashes	SoCost
Top Scenario Based on Crash	Crossing	Straight	Daylight	fit	1.8	8062.11	2211.9
Top Scenario Based on Fatalities	Crossing	Straight	Dark Lit	fit	1.8	6114.41	5776.3

Besides the cost, each attribute for a certain variable of scenarios might also be important to determine the testing scenarios. Hence, the subset of top scenarios can be obtained based on the collection of important attributes other than costs. This gives us an optimization problem which can be solved using different methods. One way to approach this problem is using exhaustive search algorithm, which gives us the exact solutions. However, it turns out that finding the exact solutions is not always possible due to time constraints. Therefore, we will also solve this problem using approximation methods (e.g., Genetic Algorithm). Other than finding the exact solutions, approximation algorithms are able to find the approximated optimal solutions in a much faster time.

In this thesis, we will introduce some related works using approximation methods, formulate the problem to be studied, present our exhaustive search algorithm and show some results. We then show our Genetic Algorithm and customize it to solve the formulated problem and finally show some results and compare the efficiency of the Genetic Algorithm and exhaustive search approach.

2. RELATED WORK

A Genetic Algorithm (GA) is a very strong approximation method that has been used widely to solve complex engineering problems [16-23]. In [16], Parallel GA has been used to determine the common areas formed by arbitrarily shaped beam. In finding the exact solution the amplitude and phase angle of each element need to be calculated which require lots of calculation and it is not affordable. Thus the authors used GA as an approximation method to reduce the number of calculations. In [17], GA has been introduced and used in 1D and 2D antenna design. The authors argued that GA is more efficient than conventional methods that are based on greedy algorithm or random walk approach. In [18], A Genetic Algorithm has been used to identify the blind channel. GA has been used instead of gradient search method. The challenge of the gradient search method is to come up with a good initial population to prevent from trapping in local minimum. The authors claimed that GA is very robust and efficient. In [19], The author illustrates that the combinatorial problems are the types of the problems that can be solved through GA. Eventually, the application of GA on electromagnetic problems was discussed. In [20], a hybrid of partial swarm optimization (PSO) and GA has been used for recurrent network design. The author compared the results and efficiency calculated by PSO, GA, and the hybrid method. In [21], a combination of Taguchi and Genetic Algorithm method (TCGA) have been used to solve an optimization problem. The optimization problem is defined for PID controller and the goal is to minimize the maximum percentage overshoot, the rise time, the set up time, and the steady state error. In [22], GA has been modified, redesigned, and applied to two different problems. The first problem is a two-dimensional quadratic function and the second is the optimal design of a permanent magnet motor. The author changed the policy of fitness function in terms of giving scores to chromosomes and elimination and also the rate of cross over and mutation. The authors have shown that their method is robust and efficient for both problems. In [23], GA has been used in image and video processing. The author showed that for real-time constraint cases, GA is more efficient than conventional methods.

In addition to engineering problems, GA has been used to solve a variety of science and technology problems [24-38]. In [24], a combination of GA and Taguchi algorithm has been utilized to a neural network. The author showed that in the combination method, Taguchi method chooses the appropriate chromosomes for cross over and consequently the chromosomes approach to the solution faster. The authors have applied the hybrid method to three different real problems and showed that their algorithm is more efficient compared to the previous methods. In [25], GA has been applied to the design of large power distribution system. The authors have utilized and examined GA to determine the solution of real distribution systems. In [26], GA has been utilized and applied to solve CPU scheduling problems. These types of problems are in general NP-hard. The author compared the optimal scheduling with the results calculated by GA for different cases with different numbers of tasks. In [27], GA has been used for nonlinear systems for active noise control. The author showed that GA can present solutions from being trapped into local minimum and results obtained by GA are satisfactory. In [28], GA has been applied to energy management to minimize the cost and maximize the efficiency. The author compared the results calculated by GA and by the classical methods. In [29], GA and ant colony algorithms have been applied to robot path integration. The results of both algorithms have been compared and the pros and cons have been discussed. In [30], GA has been used to the local-area networks with the purpose of minimizing the delay of the system. The authors compared the delays calculated by GA and the lower bound delays and showed the results are satisfactory. In [31], GA has been used to approximate the optimal values of resistors, inductors, etc. for the design of digital circuits. The author compared the results of other optimization methods with GA and discussed the benefits of GA. In [32], a hybrid of GA and simulated annealing approach has been used to predict the software reliability. The author argued that the hybrid algorithm is superior compared with GA. The authors have shown that this method is more efficient and has smaller errors. In [33], GA has been applied to the image segmentation problem. In this problem, the segmentation should change as the variable conditions of the environment (such as the light, season, etc.) change. The authors have shown that this method has improved the performance of the system over 30 percent. In [34], GA has been applied to multi-level inverters to determine the optimal switching angles for certain harmonics elimination. The authors have proved the accuracy of their algorithm and claimed that their approach can be applied to all similar optimization problems. In [35], GA has been discussed and used to model and optimize microwave devices. The impacts of several different sizes of initial population besides several different rates of cross over and mutation have been analyzed. In [36], a hybrid of Petri Net and extended GA has been used to minimize the scheduling methods for wafer fabrication. Throughput and mean cycle obtained by GA have been used to compare with results from two other methods. In [37], a hybrid of GA and simplex method has been used to model metabolic systems. The hybrid method then was used to be compared with five other methods. The author showed that this approach is the fastest method to solve such optimization problems. In [38], GA has been applied to aerodynamic design of cascade airfoil. The authors showed that this approach leads to high pressure rise, high tuning angle, and low total pressure loss, which are satisfactory for such systems.

3. PROBLEM FORMULATION

3.1 The General Problem

The general form of the problem we analyze in this thesis is as the following: we are given a table that contains a set of N scenarios of accidents where N ranges from 100 to several thousands. We are also given a set of constraints and P where P can range from 2 to 25. We are looking for a subset of P scenarios out of N scenarios that satisfy the following conditions:

1. Satisfy all constraints
2. Maximize the total cost

Note that for each problem, N and P are constants that are given.

3.1.1 Accident Scenarios

Each scenario is a combination of different attributes and a cost (such as social costs, fatalities, and crashes). Scenarios come as a table that contains V variables and one cost where V ranges from 4 to 8. Each variable can have different attributes where the numbers of attributes range from 2 to 8. For each problem, V and the numbers of attributes are fixed. Table 3.1 shows a small sample of a scenario table with 20 scenarios. In this case, the table has five variables as pedestrian action, vehicle action, light condition, pedestrian type, pedestrian speed, and a cost that is social cost. Each variable in this case can have up to four attributes. For example the fourth variable (p_mann) has three attributes shown in Table 3.2.

Table 3.1.: A Sample of 20 Accident Scenarios

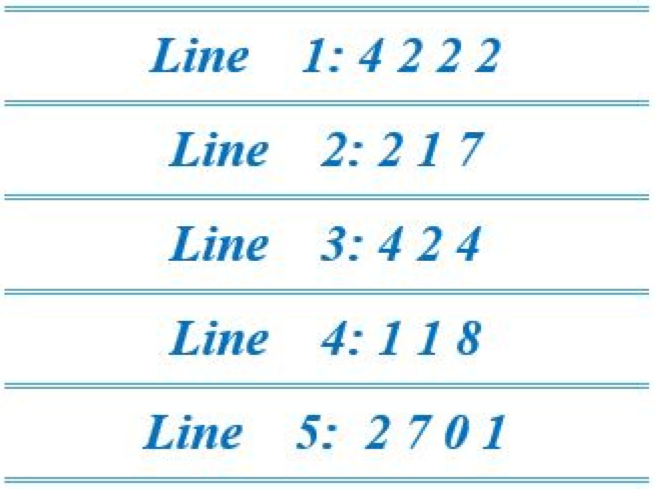
PedAction	**VehAction**	**light**	**p_mann**	**P_Speed**	**Social cost**
Crossing	Straight	Dark Lit	fit	1.8	5776.342
Crossing	Straight	Dark Unlit	fit	1.8	3462.494
Crossing	Straight	Dark Lit	Obese	1.8	2594.5
Crossing	Straight	Daylight	fit	1.8	2211.911
Standing	Straight	Dark Unlit	fit	0	1591.939
Crossing	Straight	Dark Unlit	Obese	1.8	1504.528
Walk/Run with Vehicle	Straight	Dark Unlit	fit	1.5	1425.011
Crossing	Straight	Dark Lit	fit	1.5	1358.357
Crossing	Straight	Daylight	fit	1.5	1070.68
Standing	Straight	Dark Lit	fit	0	1010.4
Crossing	Straight	Daylight	Obese	1.8	831.9123
Crossing	Turning Left	Daylight	fit	1.5	727.803
Standing	Straight	Dark Unlit	Obese	0	648.6777
Walk/Run with Vehicle	Straight	Dark Lit	fit	1.5	611.7579
Walk/Run with Vehicle	Straight	Dark Unlit	Obese	1.5	587.8934
Crossing	Straight	Dark Lit	Obese	1.5	577.9891
Crossing	Straight	Daylight	kid	1.8	469.6301
Standing	Straight	Daylight	fit	0	431.507
Standing	Straight	Dark Lit	Obese	0	427.1868
Crossing	Straight	Daylight	kid	1.5	408.1051

Table 3.2.: Attributes of Valuable 4

Attribute 1	Obese
Attribute 2	Kid
Attribute 3	Fit

3.1.2 The Set of Constraints

The set of constraints is a set of V lines where V is the number of variables. Each line of constraints specifies the desired frequencies of attributes of one of the variables for P desired scenarios. Line 2 for instance, specifies the frequencies of attributes of variable 2 for P desired scenarios. Figure 3.1 shows a sample of a set of constraints for the case where V = 5 and P = 10. Assuming that the set in Figure 3.1 corresponds to one of the scenarios from Table 3.3, Line 4, for example, means that out of 10 selected scenarios, we want one scenario with obese, one scenario with kid and 8 scenarios with fit. Note that each line should sum to P where for this case P is equal to 10.



Line 1: 4 2 2 2

Line 2: 2 1 7

Line 3: 4 2 4

Line 4: 1 1 8

Line 5: 2 7 0 1

Fig. 3.1.: A sample of set of constraints for the case that the scenario table has five variables

3.2 Model Verification with Experimental Data

Before working on the table we should trim the scenario table and get rid of duplicate scenarios.

3.2.1 Trimming the Scenario Table by Cleaning and Combining Scenario

The first step of cleaning data is to drop the scenarios that miss certain information. Table 3.3 shows such scenarios that have some unknown attributes for one or several variables. These scenarios will cause inconsistency in our approach. Thus we simply delete all of them. The next step is to sort the scenarios and combine the ones that have the same attributes. Table 3.4 shows some scenarios that are exactly the same, except for the cost part. Such scenarios should be combined in a sense that they have only one representative with a cost which comes from the sum of the individual costs. Table 3.5 shows the scenarios given in Table 3.4 after being combined. After combining the scenarios, all scenarios in the table become unique. After cleaning the redundant scenarios and combining the scenarios we call the scenarios table as trimmed table.

3.2.2 Mapping the Scenarios

For simplicity, the attributes can be mapped into positive integers. Table 3.6 is a mapping table that is used to map the scenarios in Table 3.1. Table 3.7 shows the mapped scenario table for the scenarios of Table 3.1 using the mapping table in Table 3.6.

3.3 Generating the Set of Constraints

Considering the percentages of individual attributes is one criterion that the constraints can be built based upon. Table 3.9 shows the percentages of individual attributes in terms of social cost, crashes, and fatalities. Note that the assumption is

Table 3.3.: Scenarios with some missing information

PedAction	**VAction**	**Light**	**p_mann**	**P_speed**	**Crashes**
Unknown	Turning Right	3 Dark Unlit	Fit	0	31.39056
Unknown	Turning Right	4 Twilight	Fat	2.2	62.16197
Unknown	Turning Right	4 Twilight	Fit	0	218.7786
Unknown	Turning Right	4 Twilight	Kid	0	1.127399
Playing/ Lying/Standing	Unknown	1 Daylight	Fat	2.2	54.85905
Playing/ Lying/Standing	Unknown	1 Daylight	Fit	2.2	107.1003

Table 3.4.: Scenario table with several repeated accident scenarios

PedAction2	**VehAction**	**Light**	**p_mann**	**P_Speed**	**Crashes**
Crossing	Straight	2 Dark Lit	Fat	2.2	25.71934
Crossing	Straight	2 Dark Lit	Fat	2.2	45.93587
Crossing	Turning Right	2 Dark Lit	Fat	2.2	6.23494
Crossing	Turning Right	2 Dark Lit	Fat	2.2	5.334997
Crossing	Straight	2 Dark Lit	Fat	1.5	837.273
Crossing	Straight	2 Dark Lit	Fat	1.5	97.65648
Crossing	Straight	2 Dark Lit	Fat	1.5	121.168
Crossing	Straight	2 Dark Lit	Fat	1.5	87.61506

that variables are uncorrelated. Table 3.10 and table 3.11 show the desired frequencies of attributes calculated based on such percentages given in Table 3.9. The frequencies shown in Table 3.10 and Table 3.11 are for the cases that P=2, 3 up to 10 and the cost

Table 3.5.: Scenario table after combining the scenarios

PedAction2	VehAction	Light	p_mann	P_Speed	Crashes
Crossing	Straight	2 Dark Lit	Fat	2.2	71.65521
Crossing	Turning Right	2 Dark Lit	Fat	2.2	11.56994
Crossing	Straight	2 Dark Lit	Fat	1.5	1143.713

is fatality. Determining the frequencies of individual attributes based on percentages can be subjective most of the time. Thus, we introduce and explain a systematic way to determine these frequencies.

3.3.1 A systematic Way to Determine the Frequencies of the Individual Attributes

We define remaining frequencies needed (RFN) as a variable with initial value of P. We also define minimum percentage needed (MPN) as another variable with initial value of 100/P where P is the number of desired scenarios. For each variable, we first consider the percentages of individual attributes. We select the attribute with the highest cost percentage and we call it AHCP. If AHCP is less than MPN we renew the value of MPN by the value of AHCP. We find the floor of AHCP/MPN as determined frequencies DF. We then allocate DF frequencies to the attribute of the selected one. We renew RFN as (RFN-DF) and the percentage of the selected attribute as the remainder of AHCP/MPN. We keep following above steps until RFN=0. Example 3.1 illustrates the steps of the above procedure for one of the variables (P_Speed) and the case that P=9 and the cost is Social Cost. Note that these frequencies can be calculated based on the percentages given in Table 3.9.

Table 3.6.: A mapping table for the scenarios in table

Ped action		Veh Action	
Attribute name	Mapped Value	Attribute name	Mapped Value
Crossing	1	Straight	1
Standing	2	Turning Left	2
Walk/Run against Vehicle	3	Turning Right	3
Walk/Run with Vehicle	4		
Light		P_man	
Attribute name	Mapped Value	Attribute name	Mapped Value
1 Daylight	1	Fat	1
2 Dark Lit	2	Fit	2
3 Dark Unlit	3	Kid	3
P_speed			
Attribute name	Mapped Value		
1.8	1		
1.5	2		
2.2	3		
0	4		

Table 3.7.: Mapped Scenario Table

PedAction2	**VehAction**	**Light**	**p_mann**	**P_Speed**	**Soccost**
1	1	2	2	1	5776.342
1	1	3	2	1	3462.494
1	1	2	1	1	2594.5
1	1	1	2	1	2211.911
2	1	3	2	4	1591.939
1	1	3	1	1	1504.528
4	1	3	2	2	1425.011
1	1	2	2	2	1358.357
1	1	1	2	2	1070.68
2	1	2	2	4	1010.4
1	1	1	1	1	831.9123
1	2	1	2	2	727.803
2	1	3	1	4	648.6777
4	1	2	2	2	611.7579
4	1	3	1	2	587.8934
1	1	2	1	2	577.9891
1	1	1	3	1	469.6301
2	1	1	2	4	431.507
2	1	2	1	4	427.1868
1	1	1	3	2	408.1051

Example 3.1: Determining the desired frequencies of the attributes for P_Speed based on the percentages given in Table 3.9 for the case that P = 9 and the cost is social cost.

Initializing variables: RFN = P =9; MPN = 100/P = 100/9   11.1;
Is AHCP smaller than MPN? No

Table 3.8.: Initialized Variables

	Initial percentages		Initial values		Allocated frequencies
P_Speed=1	54.80163	RFN	9	P_Speed=1	0
P_Speed=2	29.97826	MPN	11.1	P_Speed=2	0
P_Speed=3	0.655744	AHCP	54.80163	P_Speed=3	0
P_Speed=4	14.56437			P_Speed=4	0

Allocating frequencies to the attributes:
DF = Floor of (AHCP/MPN) = 4;
Renewing the contents of the variables:
Remainder of (AHCP/MPN) = 10.40; RFN = RFN DF = 5;
AHCP = 29.97826;
Is AHCP smaller than MPN? No

Table 3.9.: The percentages of individual attributes

	Crashes	Falities	Soccost
PedAction2=1	83.94721	72.199	74.0321
PedAction2=2	9.154159	15.76087	14.56437
PedAction2=3	1.165246	1.04515	1.05045
PedAction2=4	5.733382	10.99498	10.35308
VehAction=1	65.63668	94.83696	91.7486
VehAction=2	27.95616	4.264214	6.694718
VehAction=3	6.407161	0.898829	1.556682
light=1	61.46597	21.65552	26.37567
light=2	30.69221	43.7291	42.68684
light=3	7.841822	34.61538	30.93749
p_mann=1	24.04242	28.83608	28.09089
p_mann=2	65.42596	67.02488	66.83074
p_mann=3	10.53162	4.139044	5.07837
P_Speed=1	37.1527	56.98161	54.80163
P_Speed=2	51.84289	26.77676	29.97826
P_Speed=3	1.850248	0.480769	0.655744
P_Speed=4	9.154159	15.76087	14.56437

Table 3.10.: Desired frequencies of individual attributes for the cases that P ranges from 2 to 5 and the cost is fatality

Individual attributes	Desired frequency for P=2	Desired frequency for P=3	Desired frequency for P=4	Desired frequency for P=5
PedAction2=1	2	2	3	4
PedAction2=2	0	1	1	1
PedAction2=3	0	0	0	0
PedAction2=4	0	0	0	0
VehAction=1	2	3	4	5
VehAction=2	0	0	0	0
VehAction=3	0	0	0	0
light=1	0	1	1	1
light=2	1	1	2	2
light=3	1	1	1	2
p_mann=1	1	1	1	2
p_mann=2	1	2	3	3
p_mann=3	0	0	0	0
P_Speed=1	1	2	2	3
P_Speed=2	1	1	1	1
P_Speed=3	0	0	0	0
P_Speed=4	0	0	1	1

Table 3.11.: Desired frequencies of individual attributes for the cases that P ranges from 6 to 10 and the cost is fatality

Individual attributes	Desired frequency for P = 6	Desired frequency for P=7	Desired frequency for P = 8	Desired frequency for P=9	Desired frequency for P = 10
PedAction2=1	4	5	6	7	7
PedAction2=2	1	1	1	1	2
PedAction2=3	0	0	0	0	0
PedAction2=4	1	1	1	1	1
VehAction=1	6	7	8	9	10
VehAction=2	0	0	0	0	0
VehAction=3	0	0	0	0	0
light=1	1	2	2	2	2
light=2	3	3	3	4	4
light=3	2	2	3	3	4
p_mann=1	2	2	2	3	3
p_mann=2	4	5	6	6	7
p_mann=3	0	0	0	0	0
P_Speed=1	3	4	5	5	6
P_Speed=2	2	2	2	2	3
P_Speed=3	0	0	0	0	0
P_Speed=4	1	1	1	2	1

Table 3.12.: First round of finding the frequencies

	new percentages		new values		Allocated frequencies
P_Speed=1	10.4	RFN	5	P_Speed=1	4
P_Speed=2	29.97826	MPN	11.1	P_Speed=2	0
P_Speed=3	0.655744	AHCP	29.97826	P_Speed=3	0
P_Speed=4	14.56437			P_Speed=4	0

Does RFN equal to 0? No

Allocating frequencies to the attributes:

DF = Floor of (AHCP/MPN) =2;

Renewing the contents of the variables:

Remainder of (AHCP/MPN) = 7.77; RFN = RFN DF = 3;

AHCP = 14.56437;

Is AHCP smaller than MPN? No

Table 3.13.: Second round of finding the frequencies

	new percentages		new values		Allocated frequencies
P_Speed=1	10.4	RFN	3	P_Speed=1	4
P_Speed=2	7.77	MPN	11.1	P_Speed=2	2
P_Speed=3	0.655744	AHCP	14.56437	P_Speed=3	0
P_Speed=4	14.56437			P_Speed=4	0

Does RFN equal to 0? No

Allocating frequencies to the attributes:

DF = Floor of (AHCP/MPN) =1;

Renewing the contents of the variables:

Remainder of (AHCP/MPN) = 3.46; RFN = RFN DF = 2;

AHCP = 10.40;

Is AHCP smaller than MPN? Yes $\rightarrow$ MPN = AHCP = 10.40;

Table 3.14.: Third round of finding the frequencies

	new percentages		new values		Allocated frequencies
P_Speed=1	10.4	RFN	2	P_Speed=1	4
P_Speed=2	7.77	MPN	10.4	P_Speed=2	2
P_Speed=3	0.655744	AHCP	10.4	P_Speed=3	0
P_Speed=4	3.46			P_Speed=4	1

Does RFN equal to 0? No

Allocating frequencies to the attributes:

DF = Floor of (AHCP/MPN) =1;

Renewing the contents of the variables:

Remainder of (AHCP/MPN) = 0; RFN = RFN DF = 1;

AHCP = 7.77;

Is AHCP smaller than MPN? Yes $\rightarrow$ MPN = AHCP = 7.77;

Table 3.15.: Fourth round of finding the frequencies

	new percentages		new values		Allocated frequencies
P_Speed=1	0	RFN	1	P_Speed=1	5
P_Speed=2	7.77	MPN	7.77	P_Speed=2	2
P_Speed=3	0.655744	AHCP	7.77	P_Speed=3	0
P_Speed=4	3.46			P_Speed=4	1

Does RFN equal to 0? No

Allocating frequencies to the attributes:

DF = Floor of (AHCP/MPN) =1;

Renewing the contents of the variables:

Remainder of (AHCP/MPN) = 0; RFN = RFN DF = 0;

AHCP = 3.46;

Is AHCP smaller than MPN? Yes → MPN = AHCP = 3.46;

Table 3.16.: Fifth round of finding the frequencies

	new percentages		new values		Allocated frequencies
P_Speed=1	0	RFN	0	P_Speed=1	5
P_Speed=2	0	MPN	3.46	P_Speed=2	3
P_Speed=3	0.655744	AHCP	3.46	P_Speed=3	0
P_Speed=4	3.46			P_Speed=4	1

Does RFN equal to 0? Yes $\rightarrow$ Quit.

3.4 The System as a Black Box

The main problem is defined based on a given table of N accident scenarios and a set of constraint as discussed in this chapter. Figure 3.2 shows the problem as a black box with inputs and outputs. In next chapter, we introduce an algorithm based on exhaustive search method to solve this problem. We also show some results and discuss the performance and limitation of this algorithm.

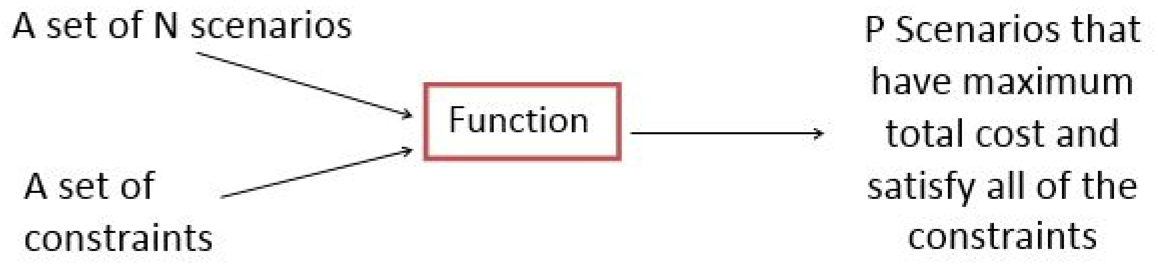

A set of N scenarios
A set of constraints
Function
P Scenarios that have maximum total cost and satisfy all of the constraints

Fig. 3.2.: The main problem as a black box.

3.5 Problem Formulation in Mathematical Form

Considering a discrete function as $C(X_i) = C_i$ where X_i is a v dimensional vector defined on Z^{+^v} and belongs to the set X=$\{X_1, X_2, X_3, \ldots, X_n\}$ (Set of all attribute vectors) and C is the cost function where $C_i \in$ Set C=$\{C_1, C_2, C_3, \ldots, C_n\}$ (Set of costs) and Set C is a set of n members that each belongs to $\Re^+$ and $i \in \{1, 2, \ldots, n\}$. In general, v is a variable that belongs to $\{4, 5, 6, 7, 8\}$ and n is a variable that belongs to $\{170, 171, \ldots, 3000\}$. However, in each case that we consider, v has a constant value equals to V and n has a fixed value equals to N.

Let us define cost vector $VC_j \in \Re^n$ as any vector with the following condition: The ith element of VC_j is either C_i or 0. For each VC_j let us also define a corresponding sum vector (binary vector) as $I_j \in Z^{(+n)}$ such that the ith element of I_j equals to 0 if ith element of VC_j equals to 0 and the ith element of I_j equals to 1, otherwise. We also define set VC and set I as the set of all VC_j and all I_j, respectively.

Consider a VC_j with K non zero elements. There are total of K attribute vectors (X_i) such that each corresponds to one of the none-zero elements of VC_j. For instance, if the 3rd, 5th and 6th elements of VC_j are the only none-zero elements (which are C_3, C_5 and C_6), then X_3,X_5 and X_6 are the corresponding X_i vectors. For each VC_j and its corresponding X_i vectors we define a frequency function Freq$_{VC_j}$ where Freq$_{VC_j}$ (y,z) is a number that shows, out of K corresponding X_i vectors how many of them have value z in their yth elements. For instance, if X_3, X_5 and X_6 are the corresponding X_i vectors to VC_7 and X$_3$ = $[1\ 1\ 2\ 1\ 1]^T$, X$_5$ = $[2\ 3\ 1\ 2\ 2]^T$, X$_6$ = $[1\ 1\ 2\ 1\ 1]^T$ then Freq$_{VC_7}$(3,2) = 2.

Considering all definitions above, our problem is formulated as follows.
Given sets C and X and a constraint matrix

$$Q = \begin{bmatrix} q_{11} & \dots & q_{1m} \\ \vdots & \ddots & \vdots \\ q_{v1} & \dots & q_{vm} \end{bmatrix}_{vXm}$$

where each row of matrix Q sums to p (1≤ p ≤ N), we would like to find a specific vector $T_j \in$ VC that satisfies the following two conditions.

1: $$\begin{bmatrix} Freq_{T_j}(1,1) & \dots & Freq_{T_j}(1,m) \\ \vdots & \ddots & \vdots \\ Freq_{T_j}(v,1) & \dots & Freq_{T_j}(v,m) \end{bmatrix}_{vXm} = \begin{bmatrix} q_{11} & \dots & q_{1m} \\ \vdots & \ddots & \vdots \\ q_{v1} & \dots & q_{vm} \end{bmatrix}_{vXm}$$

2: $IT_j \cdot T_j^T \geq I_j \cdot VC_j^T$

$\forall$ VC$_j$ $\in$ VC that satisfies the first condition.

Note that Q is a v = V by m= M matrix where v is the dimension of X_i and m is a variable that belongs to $\{4, 5, 6, 7, 8\}$. It is defined as the maximum of maximum value that an element of X_i has for all X_i vectors that belong to set X. For instance, m = $Max\{Max(x_{i1}), Max(x_{i2}), Max(x_{i3}), \ldots, Max(x_{iv})\}$ for all $i \in \{1, 2, \ldots, n = N\}$.

4. EXHAUSTIVE SEARCH METHOD

One way to approach this problem is using the exhaustive search algorithm. Figure 4.1 shows the flow chart on how this algorithm works. This algorithm first finds all P elements subsets out of the N elements universal set. The algorithm then checks if each of these subsets satisfies all constraints by counting the frequencies of the attributes of the P elements selected and comparing them with the desired frequencies that are specified in the constraints set. We mark all subsets that satisfy all constraints. Eventually, the algorithm chooses the marked subset with the maximum total cost as the solution.

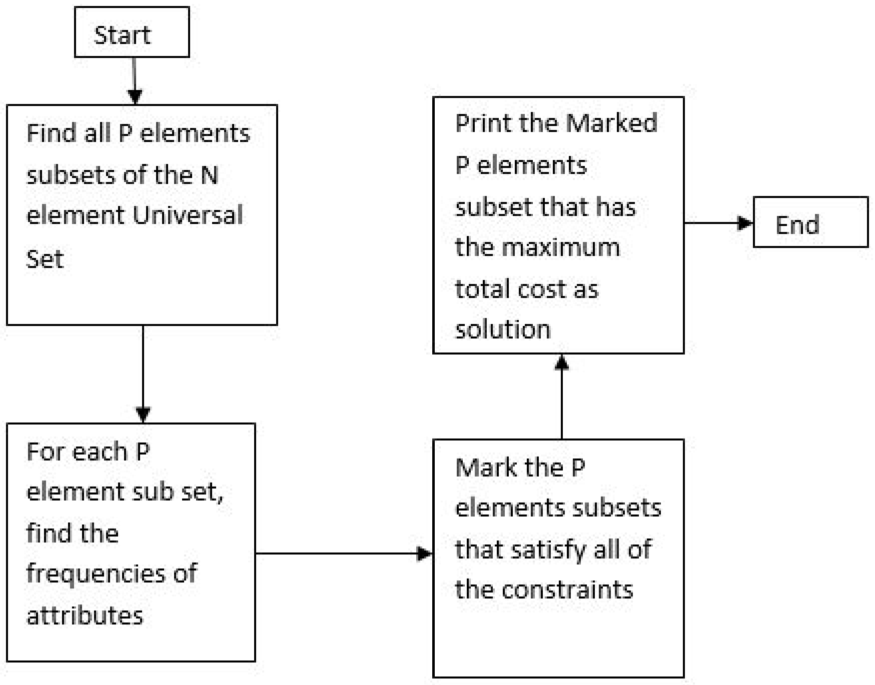

Start
Find all P elements subsets of the N element Universal Set
For each P element sub set, find the frequencies of attributes
Mark the P elements subsets that satisfy all of the constraints
Print the Marked P elements subset that has the maximum total cost as solution
End

Fig. 4.1.: Flow chart of exhaustive search algorithm

4.1 Results

Table 4.1, Table 4.2, and Table 4.3 contain the final results calculated by exhaustive search algorithm for the cases that the costs are social costs, crashes, fatalities, and P equals to 5.

Table 4.1.: Final result for the case that P=5 and the cost is social cost

PedAction2	VehAction	Light	p_mann	P_Speed	Soccost
Crossing	Straight	Dark Lit	Obese	1.8	2594.5
Crossing	Straight	Dark Lit	Fit	1.8	5776.342
Crossing	Straight	Dark Unlit	Fit	1.8	3462.494
Crossing	Straight	Daylight	Obese	1.5	370.2002
Standing	Straight	Dark Unlit	Fit	0	1591.939

Table 4.2.: Final result for the case that P=5 and the cost is Crash

PedAction2	VehAction	Light	p_mann	P_Speed	Crashes
Crossing	Straight	Daylight	Fit	1.8	8062.02
Crossing	Straight	Dark Lit	Fit	1.8	6114.414
Crossing	Straight	Daylight	Fit	1.5	7019.07
Crossing	Turning Left	Daylight	Fit	1.5	7970.328
Standing	Turning Left	Dark Lit	Obese	0	63.39208

Table 4.3.: Final result for the case that P=5 and the cost is fatality

PedAction2	VehAction	Light	p_mann	P_Speed	Fatality
Crossing	Straight	1 Daylight	Obese	1.5	44.52323
Crossing	Straight	2 Dark Lit	Obese	1.8	414.5695
Crossing	Straight	2 Dark Lit	Fit	1.8	910.493
Standing	Straight	3 Dark Unlit	Fit	0	268.4724
Crossing	Straight	3 Dark Unlit	Fit	1.8	571.0684

4.2 Performance and Limitations of Exhaustive Search Algorithm

The advantage of exhaustive search is that results are exact and reliable. However, due to the complexity of the algorithm, calculating the results takes long time for some cases. As N and P grow the time needed to calculate the result grows with a combinatorial rate. For the case that N=124 and P=5 it takes approximately 40 minutes for a normal computer to calculate the result. This is due to the complexity of the algorithm. The program finds all P elements subsets of an N elements universal set which are totally C (N,P). The result of this combination can be a very large number. For instance for the case that N=170 and P=10, C(170,10) = 4.2419e+015 which is large. Note that N can range from 100 to several thousands and P can range from 2 to 25. Also note that C (N,P) is just the numbers of subsets that the program finds. For each subsets then the program needs to check if the subset satisfies the constraints. Depending on the outcome then the program might need to add the selected subset to the set of marked subsets. So the complexity of the algorithm is $O(C(N,P) \times T(\textit{checking the constraints and possibly marking them}))$. Thus the numbers of instructions are much higher than C(N,P). Table 4.4 shows a lower bound time estimation for several different cases based on the performance of the program on a Quad 2 Gigahertz computer.

For most of the cases that P>7 and N>100, using the exhaustive search algorithm to calculate results is impossible for a normal computer with an 8 Gigahertz CPU. Since the result of the problem is important, we can compensate for the speed by using some approximation methods. In approximation methods we may lose the accuracy of the result but we may find some results that are satisfactory. In other words, we might not find the best solution of the problem. However, we may find something that is close enough to the solution that we can accept it as an approximation solution. Our need to such methods takes us to the next chapter. In the next chapter we introduce a very powerful approximation algorithm named Genetic Algorithm. We then customize and design the Genetic Algorithm for our problem and analyze the results.

Table 4.4.: A lower bound time estimation for several different cases with different values of N and P

N	P	Lower bound time
80	5	5 min
100	5	15 min
120	5	40 min
170	5	4 hours
30	10	6 min
80	10	228 days
100	10	6.5 years
120	10	44 years
170	10	16 centuries
30	15	31 min
80	15	25 centuries
100	15	963 centuries
40	30	3 hours
50	30	17 years

5. GENETIC ALGORITHM

Genetic Algorithm (GA) is an approximation method based on natural selection theory that starts with a world of Initial Population of species where each individual is called a chromosome. Each chromosome is composed of a fixed number of genes. Once the initial chromosomes are generated they start to produce new chromosomes through two processes called Mutation and Cross over. The chromosomes with weaker genes then get eliminated by a function called evaluation function. Mutation, Cross Over and Elimination processes keep happening until the desired chromosome is generated or the set up time expires. The desired chromosome is the one that is either the solution or is relatively close enough to the solution of the problem that it can be accepted as an approximation solution. Figure 5.1 is a simple flow chart that indicates how GA works in general:

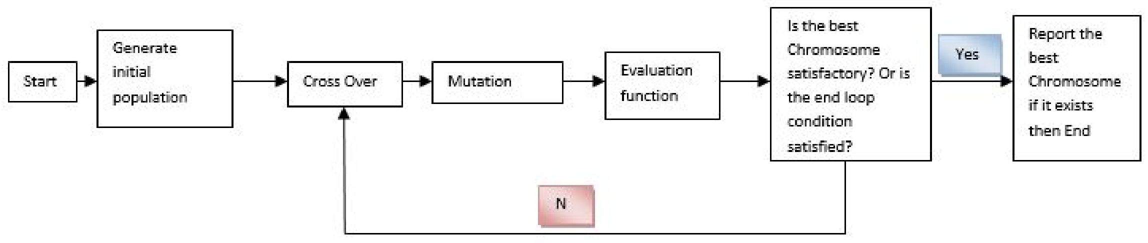

Start
Generate initial population
Cross Over
Mutation
Evaluation function
Is the best Chromosome satisfactory? Or is the end loop condition satisfied?
Yes
Report the best Chromosome if it exists then End
N

Fig. 5.1.: A simple flow chart of the Genetic Algorithm

5.1 Fundamentals of Genetic Algorithm

5.1.1 Initial Population

Initial Population is the first generation of the chromosomes. Generating an appropriate Initial Population is one of the key points that guarantee GA converges to the solution.

5.1.2 Mutation

During the process of Mutation first a group of chromosomes are randomly selected and then for each selected chromosome one of its genes are randomly selected, arbitrarily turns into a new gene and the process produces new chromosomes.

5.1.3 Cross Over

During the process of Cross Over first a group of chromosomes are randomly selected. For the selected chromosomes every two arbitrarily selected chromosomes mate and produce a new chromosome as a child. The child has half of its genes from its mother and the other half from its father.

5.1.4 Evaluation Function

Evaluation function is a process in which the chromosomes with weaker genes get eliminated. The strength of the genes depends on the nature of the problem. For example if in our design we know that chromosomes with certain genes would not help in evolving towards the solution they should be eliminated. In general evaluation function first assigns scores to all of the chromosomes based on the strength of their genes. The function then eliminates the chromosomes that their scores are under a defined threshold. The threshold can be defined based on the overall scores of the current population. As the program progresses the threshold can be fixed or can change depending on the nature of the problem and the overall performance of the scores.

5.1.5 Tuning

Tuning is a very important part of a GA. The size of the Initial Population and the rates of Cross Over Mutation and Elimination are the factors that need to be tuned in a way that results in convergence. Usually tuning is a trial and error method and requires lots of time. However statistical analysis can be done to accelerate the process of tuning.

5.1.6 Time and the Result

In general proving that the approximation solution equals to the exact solution is impossible unless we already know the exact solution using other methods. All we can claim about GA is that as the time of running the program approaches to infinity or is significantly large enough the approximation solution approaches to the exact solution and GA may finally hit the exact solution.

5.2 Developing GA for the Scenario Selection Problem

5.2.1 Defining chromosomes

The first step of customizing a GA to a certain problem is defining chromosome. A chromosome is a data structure that can potentially contain the solution of the problem. Recall that our problem was to find P scenarios of accident out of total N scenarios while satisfying several constraints. In that sense
pagebreak we define a chromosome as an array of P integers where each element of this array is an integer between 1 to N. We know that the solution is P scenarios.

Therefore, the defined chromosome can be potentially the solution of our problem. Figure 5.2 are three samples of three valid chromosomes for the case that P equals to 5 and N equals to 420.

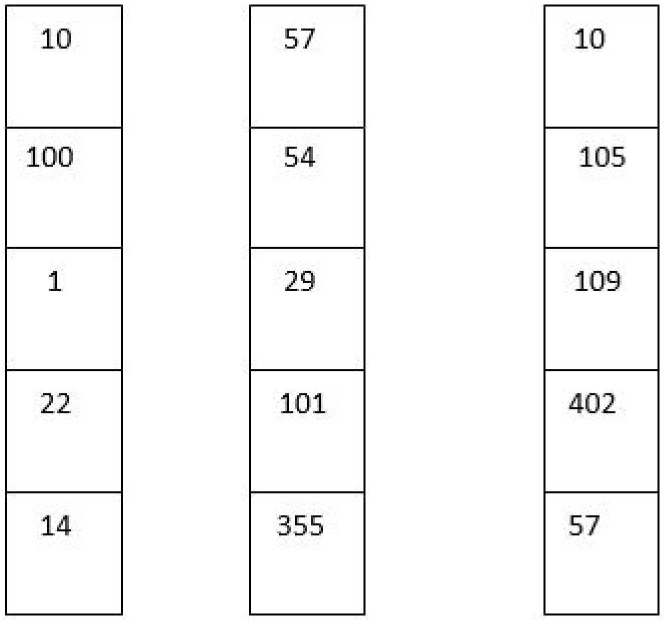


Fig. 5.2.: Three samples of three valid chromosomes

5.2.2 Initial Population

The next step after defining chromosomes is to generate a suitable Initial Population. As we mentioned before, an appropriate Initial Population is one of the important elements of a GA that guarantees the convergence. In terms of tuning, the size of Initial Population is one of the parameters that is flexible and may need to be tuned. We name this parameter as tune1 which can be initially set as 10^4. Also we define our chromosome space as an array of chromosomes with the size of MaxChromIndex where it is initially 5×10^6, however, tune1 and MaxChromIndex can change based on the size of P and N. Having tune1, chromosome space and MaxChromIndex in mind we may start defining the Initial Population generator.

5.2.2.1 Initial Population Generator Function

This function generates V (the number of variables) sets of chromosome that each set of chromosomes has tune1 chromosomes where tune1 is 10^4 as our initial approach. Recall that each chromosome is composed of P integer numbers where P is the number of desired scenarios and integer numbers are the row numbers of each accident scenario. The V sets of chromosomes would be generated in such a way that the chromosomes in each set would satisfy at least one of the constraints. Obviously, if one of the V sets remains empty the problem does not have any solution. In order to generate theses sets we define another function Partitioner to partition the accident scenarios such that looking at the constraints we can simply choose P scenarios randomly from the corresponding partitions and we can guarantee that at least one of the constraints is satisfied.

In order to create the Initial Population we need a Partitioner function which creates a three dimensional array name PartList. PartList is a V X MaxVal X N array. Where V is the number of variables MaxVal is the maximum value that a variable can have and N is the number of total scenarios. PartList[i][j] contains all accident scenarios row numbers that has value j in their variable i. For instance, PartList[2][3] contains scenario row numbers that have value 3 in their variable 2. In order to have a better understating, let us consider a simple example in Table 5.1:

Table 5.1.: A simple example

Row Number	Variable 1	Variable 2	Variable 3	Variable 4	Variable 5	Cost 1
1	1	1	3	2	1	1000
2	2	2	2	2	4	2000
3	1	1	1	2	2	400
4	3	2	2	1	1	500
5	2	1	1	1	1	202
6	3	2	2	2	2	952
7	4	1	2	3	4	272
8	2	2	2	1	3	22
9	2	1	1	1	3	33
10	2	4	4	3	3	77
11	1	2	3	3	1	780

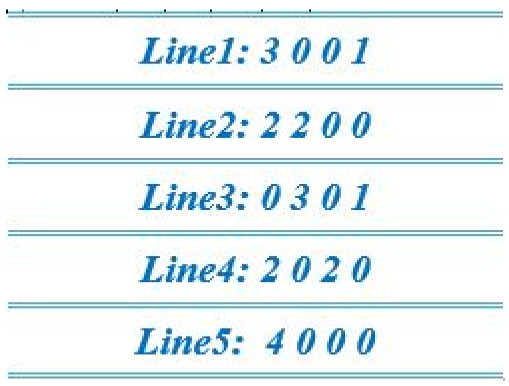
For this case PartList[3][2] for example contains the highlighted rows. The content of this data structure shown in Table 5.2:

Table 5.2.: Contents of data structure

PartList[3][2]
2
4
6
7
8
-
-
-
-
-
-

Once the Partioner function is called and PartList array is built, we can generate the Initial Population. Recall that the Initial Population contained V sets where each set satisfied at least one of the constraints. To generate the chromosomes of each of these sets we should get help from PartList. For more intuition let us consider the set of constraints shown in Figure 5.3 for the sets of scenarios of Table 5.1:

Our goal is to generate chromosomes to satisfy at least one of the constraints. For instance let us assume that we would like to generate a chromosome that satisfies at least the third constraint: Line 3: 0 3 0 1



Line1: 3 0 0 1

Line2: 2 2 0 0

Line3: 0 3 0 1

Line4: 2 0 2 0

Line5: 4 0 0 0

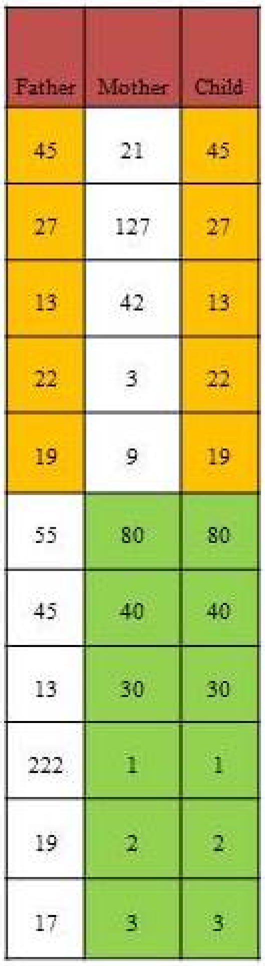
Fig. 5.3.: Set of constraints

From this set of constraints, we know that variable 3 should have 1 scenario with value 4 and 3 scenarios with value 2. If we choose 4 unrepeated scenarios, 3 from PartList[3][2] and 1 from PartList[3][4] the four selected scenarios would at least satisfy the third line constraint. Using this method we can generate many chromosomes that satisfy the third constraint. Therefore we can generate all V sets which form the Initial Population.

5.2.3 Cross Over

Simple Cross Over is a process in which two chromosomes would be randomly selected as the parents. The chromosomes would mute then and produce a new chromosome as their child which half of the genes of the child come from the mother and the other half come from the father. For the case that P is odd, (P-1)/2 of the genes of the child come from the father and (P+1)/2 come from the mother. Where the father is the first selected chromosome and the mother is the second one. Figure 5.4 is a sample of a simple Cross Over for the case that P=11.

The rate of Cross Over is another factor that may need to be tuned. It determines how many Simple Cross Over occur during an iteration. We call this rate as Tune2 and initially we set it as 20 percent of the current population. We shall now define the Cross Over as putting the Simple Cross Over in a loop with the size of Tune2.



Father	Mother	Child
45	21	45
27	127	27
13	42	13
22	3	22
19	9	19
55	80	80
45	40	40
13	30	30
222	1	1
19	2	2
17	3	3

Fig. 5.4.: A sample of a simple Cross Over

5.2.4 Mutation

A simple Mutation is a process in which one of the current chromosomes would be randomly selected, one of its genes would then be randomly selected and randomly changes into a valid number which result in a new muted chromosome. Here a valid number is an integer between 1 to N. Example 5.1 shows all the steps of a simple Mutation.

Example 5.1: Steps of a simple Mutation:

Step 1) Randomly select a chromosome:

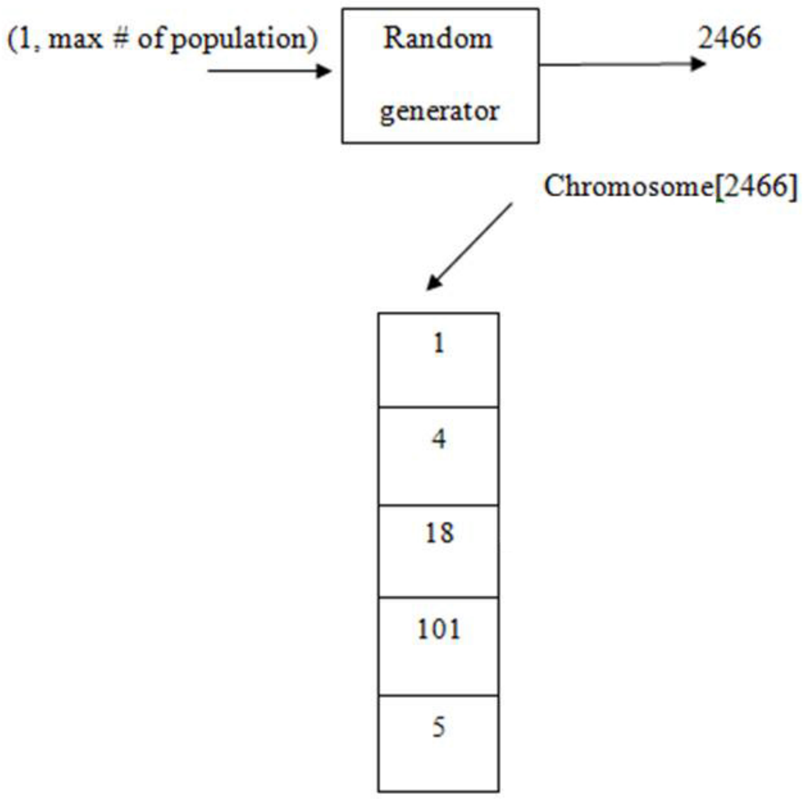


Fig. 5.5.: Step 1

Step 2) Randomly choose a gene:

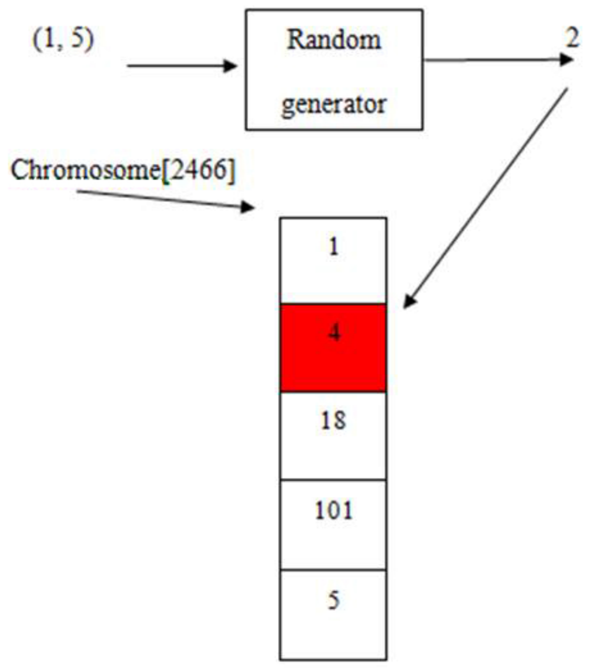


Fig. 5.6.: Step 2

Step 3) Randomly generate a valid number:

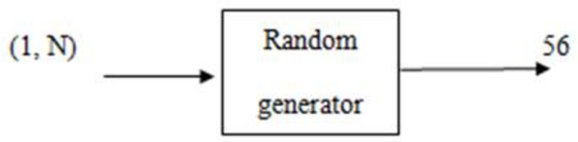


Fig. 5.7.: Step 3

Step 4) Produce a new chromosome with a muted gene:

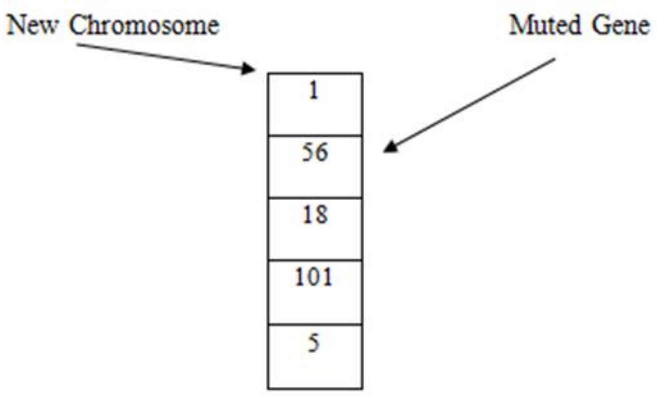


Fig. 5.8.: Step 4

The Mutation rate is the next factor that may need to be tuned we call this rate as Tune3 and initially we set it as two percent of the size of current population. The Mutation is defined as putting Simple Mutation Function in a loop with the size of tune3.

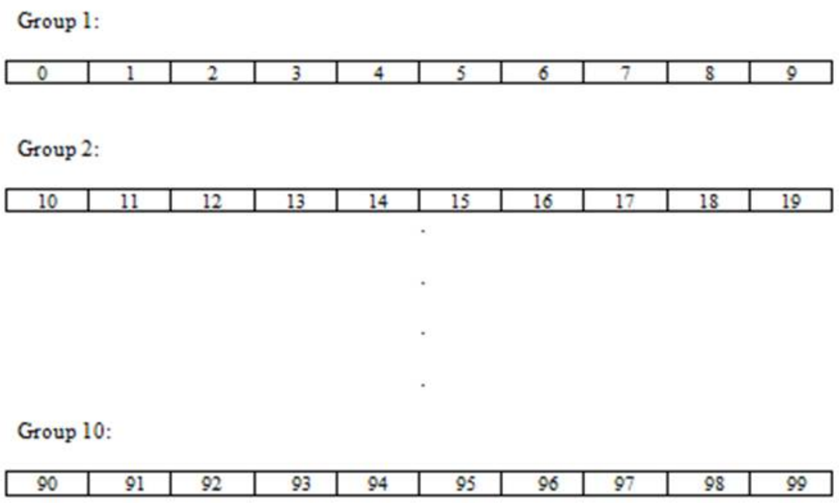
Initial Population, Cross Over and Mutation all need a random function. Since the random generator in C language was not satisfactory in terms of uniformly distribution and the output range, we had to expand that function and make it suitable for our task. Next we would discuss the limitations of the rand function in C language and how we made this function usable for our purpose.

5.2.4.1 Random Function

Rand Function in C generates an integer number between 0 to RAND_MAX where RAND_MAX is 32767 and the distribution is uniform. If the goal is to generate a number in a different range the function should be modified or a new function needs to be written. In our case we need to generate an integer between two arbitrary integers which both are less than a predefined MAX where we initially set the Max as nine hundred million. We decided to write a new function which uses the Rand function in C and expand and control it in a way that is suitable for our task. We would like our function to receive two integers (both smaller than MAX) and randomly generates an integer between the two given inputs. The first and easy step to change the range is to use modulo operator which finds the remainder. Simply using modulo operation for an integer, T, we can guarantee that the max is definitely less than T. we can also use modulo operator to force the output to be in a range between min and max. To do so we first generate a number and then find the remainder of that number for max-min+1. There is an issue with this method regarding the distribution. Using the modulo operator corrupts the distribution of some numbers. This method cause some numbers have more likelihood than the others. To illustrate this issue let us consider example 5.2.

Example 5.2) For simplicity, Assume that there is a rand function that generates number between 0 to 105. Now assume we would like to change the range to 0 to 9 rather than 0 to 105.

Using modulo operator will guarantee that any generated number belongs to the group 1 to group 10 will be mapped to 0 to 9.



Group 1:

0	1	2	3	4	5	6	7	8	9

Group 2:

10	11	12	13	14	15	16	17	18	19

.

.

.

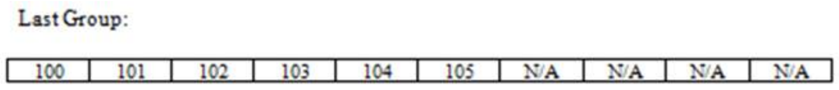
.

Group 10:

90	91	92	93	94	95	96	97	98	99

Fig. 5.9.: Groups 1 through 10

The tricky part is that the last group will mapped only to 0 to 5.



Last Group:

100	101	102	103	104	105	N/A	N/A	N/A	N/A

Fig. 5.10.: Groups 1 through 10

Which makes the likely hood of 0,1,2,3,4,5 higher than 6,7,8,9.

To deal with this issue we can ignore the last group which guarantees that the distribution would remain uniform. Thus once the rand function is performed before using the modulo operation, the outcome needs to be checked. If the outcome belongs to the Last group the rand function should be recalled. Otherwise the modulo operator can be used.

Having a significantly small max which is 32767 is the next issue. Extending the max into RAND_MAX ^{2} which is greater than 9×10^8 will solve our problem. To extend the max we can try two independent random experiments. To illustrate this technique let us consider Example 5.3.

Example 5.3)
Let us assume that the RAND_MAX is 4 and a max random number higher than that is needed. Let us assume the MAX needed number is 14.

The normal random experience can generate 5 different numbers between 0 to 4. If we perform this random experiment two times independently the sample space of the joint has 25 different combinations which can be map to integers in the range of 0 to 24 and form a random variable. If the max number needed is 14 we shall first perform the experiment twice independently and then find the modulo operator for 15. Of course if just in case the outcome belong to the last group we can simply reform the independent experiments for the sake of the uniform distribution. Utilizing this technique for our problem we can expand the RAND_MAX from 32767 to a number greater than 9×10^8.

To check how the function works we did some test. We generated 3×10^8 random numbers in a fixed range and found the max and min frequency of the generated numbers and compare it with expected frequency for a uniform distribution. Table 5.3 contains the results of our three different tests.

Table 5.3.: Three different random tests to confirm the uniform distribution of the random fuction

Title	Min	Max	Expected Value	Min Freq	Max Freq
1st Test	0	49999	6000	5687	6338
2nd Test	0	9999	30000	29310	30585
3rd Test	0	999	300000	298492	301500

In one of these cases we set the max as 49999 which is greater than 32767. Our result shows that our function can generate random numbers for higher ranges. Also according the Table 5.3 as the expected value grows the Min_freq and Max_freq are getting close. This indicates that the random function has a uniform distribution.

5.2.5 Evaluation Function

The last major function that we need to define for GA is evaluation function. This function first needs to assign values of worth to chromosomes and then it needs to kill all chromosomes that their values of worth are under a pre set threshold. Setting the right values of worth and the right threshold are the tricky parts. Designing an appropriate evaluation function besides a good Initial Population can guarantee that the chromosomes evolve and approach to the solution in an asymptotic way. Instead of setting certain values of worth we decided to categorize our chromosomes based on the strength of their genes. The categories are defined as follows.

5.2.5.1 Defining Chromosome Types

The strongest Chromosome is Chro-God which is defined as a chromosome that satisfies all of the constraints. We also name the solution of our problem as Chro-God*. The second and third strongest chromosomes are King and Lord. In order to define king and Lord first we define the concepts of a perfect father & a perfect mother. A chromosome is a perfect father if all its top genes are correct (Compared to the top part of a Chro-God). A chromosome is a perfect mother if all its bottom genes are correct (Compared to the bottom part of a Chro-God). We shall now define Lord as a chromosome that is either a perfect father or a perfect mother and not both. We also define a chromosome as a King if it acts as a perfect mother and a perfect father but it is not a Chro-God. We define a Chromosome as Middle Class if it can potentially turn into a Lord due to one step simple Mutation.We simply define the rest of the Chromosomes as poor. The level of the strength of the genes is defined as follows:
Chro-God* $\geq$ Chro-God $>$ King $>$ Middle Class $>$ Poor

5.2.5.2 How to Determine the Type of a Chromosome

In order to determine the type of a chromosome we need to determine how many genes are potentially correct in the top, bottom and the whole part of a chromosome. To achieve this we need to check how many attributes a chromosome or part of the chromosome is missing (considering the set of constraints). To illustrate this let us consider Example 5.4:

Example 5.4) Assume that the line 3 of the given set of constraint is as follows:
Line 3: 3 0 0 2

Obviously, this constraint is defined on the third variable and it means that out of 5 selected scenarios 3 of them should have variable 3 with value 1 and 2 of them should have variable 3 with value 4. Let us assume that we are considering the chromosome 127 which corresponds to the scenarios given in the Table 5.4 The selected scenarios have variable 3 with 2 value 1, one value 2, one value 3 and 1 value 4. Considering line 3 of the constraint this chromosome is missing one value 1 and one value 4 which we say that it lacks two. Considering just this constraint we can say for sure that at least 2 of the selected scenarios are incorrect.

Table 5.4.: Chromosome 127 with its corresponding scenarios

Genes of the Chromosome 127 (Scenario row numbers)	Var1	Var2	Var3	Var4	Var5	Cost
5	1	2	4	2	2	5000
7	2	1	1	1	2	200
13	4	3	1	2	1	300
22	5	2	3	1	1	202
19	5	2	2	1	1	209

To determine at least how many selected scenarios are incorrect we need to consider all of the constraints and see how many each lacks. To do so let us define Max-L. We consider all of the variables of the certain selected (genes) scenarios and check how many each variable lacks. We define L(i) as the numbers of values that variable i of a set of genes lacks. We also define Max-L, as the max of L(i)s.

For the case that P is even, if the Max-L equals to 0 for all selected P genes then the chromosome is a Chro-God. If Max-L equals to P/2 for the top P/2 genes and Max-L equals to P/2 for the bottom P/2 genes of a chromosomes and the chromosome is Not a Chro-God then it is a King. If Max-L equals to P/2 for either top P/2 genes or bottom P/2 genes and not for both part of a chromosome then the chromosome is a Lord. If Max-L equals to (P/2)+1 for either top P/2 genes or bottom P/2 genes or both parts of a chromosome then the chromosome is a Middle Class. If a Chromosome is none of the defined chromosomes then it is a Poor Chromosome.

For the case that P is odd if the Max-L equals to 0 for all selected P genes then the chromosome is a Chro-God. If Max-L equals to (P-1)/2 for the top (P-1)/2 genes and Max-L equals to (P+1)/2 for the bottom (P+1)/2 genes of a chromosomes and the chromosome is Not a Chro-God then it is a King. If Max-L equals to (P-1)/2 for the top (P-1)/2 genes or Max-L equals to (P+1)/2 for the bottom (P+1)/2 genes and the chromosome is not a King then the chromosome is a Lord. If Max-L equals to ((P-1)/2)+1 for the top (P-1)/2 genes or Max-L equals to (P+1)/2)+1 for the bottom (P+1)/2 genes of a chromosome then the chromosome is a Middle Class. If a Chromosome is none of the defined chromosomes then it is a Poor Chromosome.

After categorizing the chromosomes for each generation, the evaluation function should kill (eliminate with no further consideration) the chromosomes with weaker genes. The evaluation function of our program starts eliminating Poor Chromosomes in age1 and depending on the progress of the program it will change the threshold from Poor to middle class in age2 and continue eliminating both middle class and poor in the second age. We could also consider changing the threshold to Lord however since we got a reasonable and satisfactory result we did not take our function that far.

5.3 Intuition of our Approach

Our goal is to complete the Chro-God set. Clearly, if Chro-God set is not empty Chro-God* exists. Our goal is to use GA to generate Chro-Gods randomly. Since GA hits the Chro-Gods randomly finding Chro-God* is just the matter of time and if the program runs long enough based on randomization hitting Chro-God* is guaranteed. Let us assume rows 7,11,3,101 and 5 of the accident scenarios satisfy all of the constaraint. Therefore the chromosome in Figure 5.11 is a Chro-God. If this Chromosomes does not exist in Initial Population it can only get generated in one iteration either by a one step simple Cross Over or Mutation. Figure 5.12 and Figure 5.13 shows how the generation of the Chro-God through one step Cross Over and Mutation.

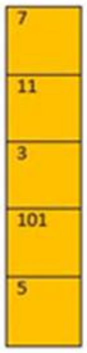


Fig. 5.11.: Chro-God

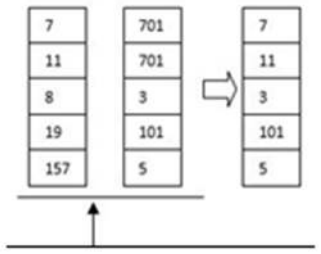


Fig. 5.12.: Generation of Chro-God through Cross Over

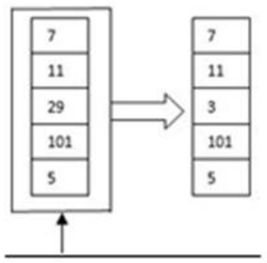


Fig. 5.13.: Generation of Chro-God trough Mutation

In the case of Cross Over the God can get generated when the parents are either a Lords or Kings. So completing the sets of Lords and Kings would lead to hitting the Chro-Gods. Note that by completing the sets we do not need all combinations of Lords or Kings. As long as the right genes (the gene of the half Chro-God) are generated hitting the Chro-God is guaranteed. To have decent sets of Kings and Lords we need an almost complete set of middle class. Middle class then can turn into Lords through one step simple Mutations. In the Initial Population we need to make sure that we have enough middle class chromosomes before we run the next steps of the algorithm. Middle Class belongs to the complete set of Initial Population and the likelihood of middle class is very high for the cases that P is small. When P=5 either one or maximum two selected lines must be correct in order for a chromosome to be a middle class. Clearly as P grows, the likelihood of middle class in the Initial Population decreases. So we can keep generating the Initial Population until we make sure that we have enough numbers of middle class compared to a pre set threshold and then we can proceed with the next steps of the algorithm. The idea of our algorithm is to first generate a decent amount of different middle class chromosomes through the process of Initial Population, we then generate Lords through Mutations of Middle classes then increase the population of Kings through Cross Over of the Lords and eventually hit the Chro-God through Cross Over of the Kings and Lords. Figure **??** shows the flow chart of our algorithm. As it is shown in Figure 5.14 in the first age we kill only Poor Chromosomes. When we have enough numbers of Lord and Kings we start killing the middle class besides Poor in the second age since clearly we will no longer need middle class.

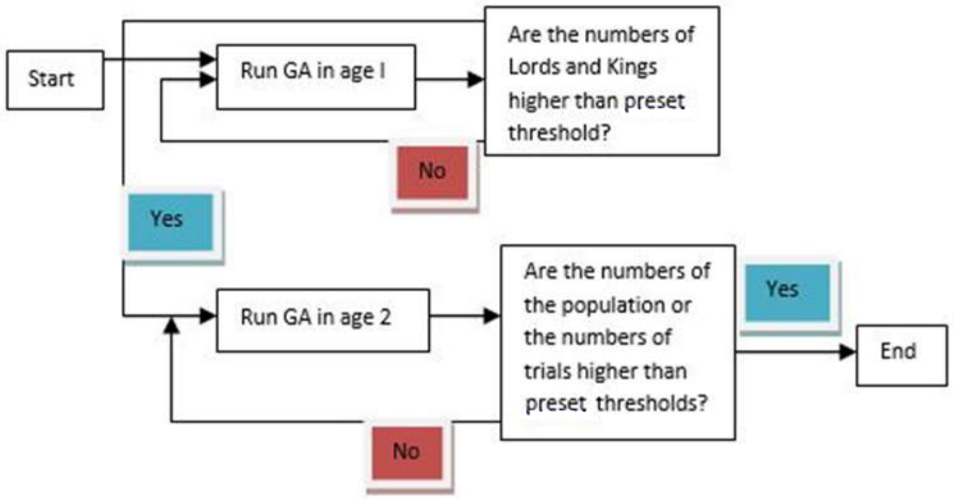

Start
Run GA in age I
Are the numbers of Lords and Kings higher than preset threshold?
No
Yes
Run GA in age 2
Are the numbers of the population or the numbers of trials higher than preset thresholds?
Yes
End
No

Fig. 5.14.: The Flow chart of our Algorithm

5.4 Experimental Results

5.4.1 Sample of Top Scenarios

Table 5.5 and Table 5.6 show the top 5 and 10 selected scenarios. For the top 5 the result maximized based on Social Cost and for top 10 it is maximized based on Crashes. Both of the tables are for the case that the total number of the scenarios were initially about 300 lines and reduced to 124 lines. In the case of 5 selected scenarios the solution is exact and we can verify that by comparing the result of the brute force. In the case of 10 selected scenarios however, the result is an approximation and we cannot verify that it is the best solution.

Table 5.5.: Approximated top 5 scenarios for social cost calculated by GA

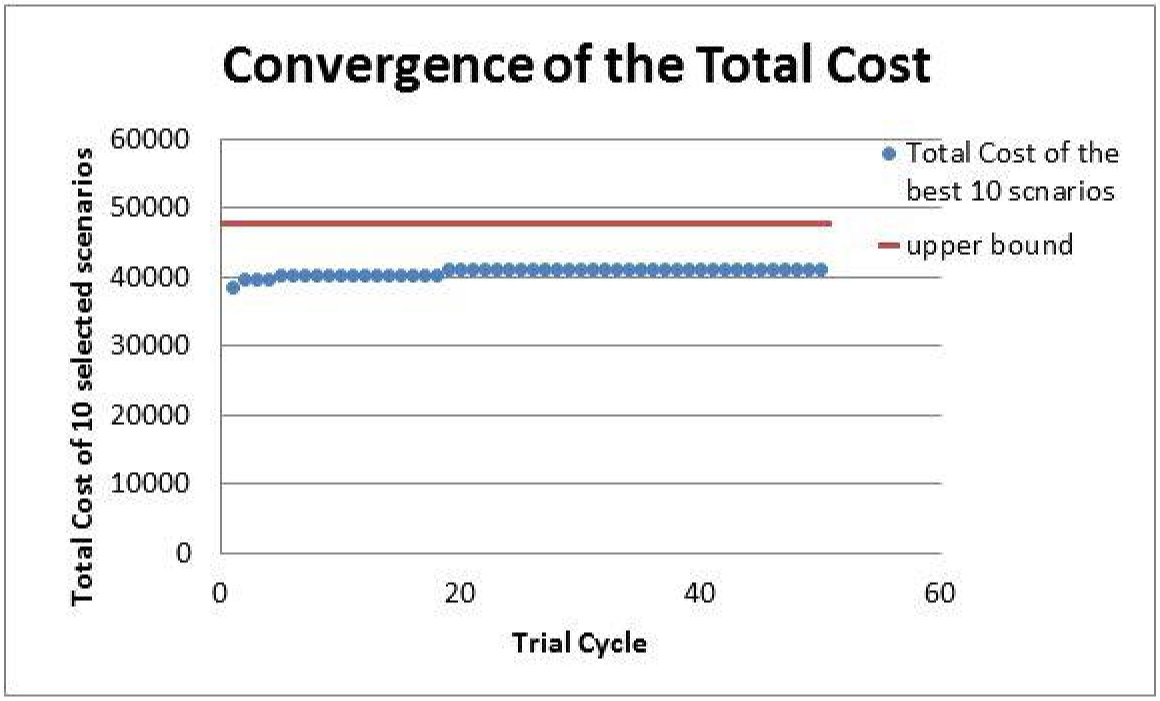
Ped Action2	Veh Action	Light	p_mann	P_Speed	Soccost
Crossing	Straight	Dark Lit	Obese	1.8	2594.5
Crossing	Straight	Dark Lit	Fit	1.8	5776.342
Crossing	Straight	Dark Unlit	Fit	1.8	3462.494
Crossing	Straight	Daylight	Obese	1.5	370.2002
Standing	Straight	Dark Unlit	Fit	0	1591.939

Table 5.6.: Approximated top 10 scenarios for crash calculated by GA

PedActionStanding	VehAction	Light	p_mann	P_Speed	Crashes
Crossing	Straight	1 Daylight	Fit	1.8	8062.02
Crossing	Turning Left	1 Daylight	Fit	1.5	7970.328
Crossing	Straight	1 Daylight	Fit	1.5	7019.07
Crossing	Straight	2 Dark Lit	Fit	1.8	6114.414
Crossing	Turning Left	2 Dark Lit	Fit	1.5	3509.129
Crossing	Straight	1 Daylight	Kid	1.5	3187.769
Crossing	Straight	1 Daylight	Fat	1.8	2756.603
Crossing	Straight	2 Dark Lit	Fat	1.8	2368.723
Walk/Run against Vehicle	Turning Right	1 Daylight	Fit	1.5	22.40009
Standing	Turning Left	3 Dark Unlit	Fit	0	17.07412

5.4.2 Convergence Table and The Accuracy

One of the things we need to show is that the GA is convergent to the optimal solution. Figure 5.15 shows the performance of our program in terms of convergence. The horizontal axis is the time line and it contained of approximately 50 trials where each trial takes about 3 minutes. The vertical axis is the total cost axis and shows the total cost of the best Chro-God. The top diagram contains a red line which shows the total cost of top ten scenarios while we are not considering any of the constraint. Clearly, the red line is a supremum for our problem and in the case that the corresponding scenarios to the red line do not satisfy all of the constraint we can call the red line an unreachable supremum. In Figure 5.15 the top diagram indicates that the total cost of best God is around 42000 while the unreachable supremum is around 48000. Even though we cannot prove that our solution is the best solution, the total cost of our solution is high enough compared to the unreachable supremum that we can accept it as a satisfactory approximation solution. To show the progress of the diagram in terms of total cost we connected the mid points of each horizontal line in the bottom diagram of the Figure 5.15. As the bottom diagram shows the total cost starts from 38500 in the first trial and saturated in midpoint 30 which is approximately 90 minutes. Clearly if the longer we run the program the more accurate the approximation result is compared to the exact one.



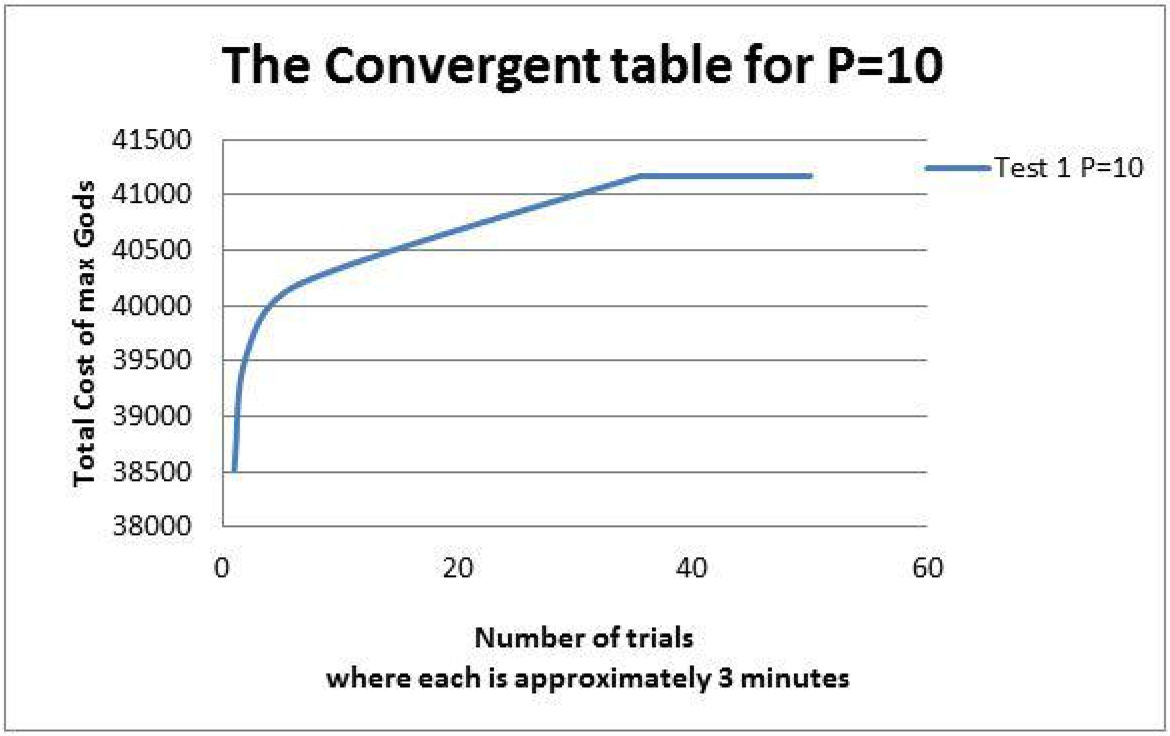


Fig. 5.15.: Convergence Figure

5.4.3 Time of GA Compared to Brute Force Search

GA performs much faster than Brute force algorithm. Table 5.7 summarizes the performance of GA compared to the exhaustive search algorithm. In the case of selecting 5 scenarios, GA takes seconds to hit the exact results, which is much faster compared to 40 minutes needed for exhaustive search algorithm. In case of N = 124 and P = 10, even though we cannot guarantee that GA-approximated results are the same as exact results found by exhaustive search, we can confirm that all constraints are met and the total cost is very close to the theoretical upper bound. For this case (P = 10), if we are using exhaustive search algorithm, obtaining exact solutions will be un-affordable due to the time needed to run the program.

Table 5.7.: Comparing the performance of GA and exhaustive search

N	P	Running time of exhaustive exhaustive search algorithm	Running time of our GA Approach	Does GA hit the exact solution?
124	5	40 minutes	30 seconds	Yes
124	10	Not affordable	150 minutes	Unknown

6. CONCLUSION AND FUTURE WORK

In this thesis, we introduced methods of choosing a subset of testing scenarios from a complete set of testing scenarios based on the importance of the cost and individual attributes. We designed an exhaustive search algorithm to find the results and discussed the complexity and limitations of this algorithm. We then introduced Genetic Algorithm and developed a GA-based method to approximate the optimal results of the problem. We illustrated that our GA-based approach is very fast and efficient compared to the exhaustive search method and showed that, for some cases it takes several seconds to approximate the results and for some other cases with larger values of N and P, it takes several hours to approximate satisfactory results while due to the complexity of exhaustive search algorithm, these cases are impossible to solve using exhaustive search algorithm.

Table 6.1 shows the selected scenario lines from a sorted table for a case that there are totally 64 scenarios. The scenarios are sorted from large to small based on the cost of the crashes and the results are for the cases that P ranges from 2 to 8. By comparing the total cost of the selected scenarios shown in Table 6.1 with the total costs of the top scenarios without considering the constraints we can conclude that as the numbers of the desired scenarios grow the percentage coverage of the two methods becomes tight. Table 6.2 shows the total percentage cost coverage by the selected scenarios for both of the methods and the cases that P ranges from 2 to 8. Table 6.1 indicates that for several of the cases the difference between both of the methods is very small. As the numbers of the desired scenarios increase, the percentage cost covered by the selected scenarios approaches to one hundred percent. Thus from a certain number of selected scenarios we can choose the top scenarios without considering the constraints which gives us enough variation of different attributes.

Table 6.1.: The selected scenarios for the cases that P ranges from 2 to 8 and the cost is crash

PedAction2	VehAction	light	p_mann	P_Speed	Crashes	2	3	4	5	6	7	8
Crossing	Straight	1_Daylight	Obese&fit	1.8&1.5	19824.7		x	x	X	x	x	X
Crossing	Turning_Left	1_Daylight	Obese&fit	1.8&1.5	13981.3	X	x	x	X	x	x	X
Crossing	Straight	2_Dark_Lit	Obese&fit	1.8&1.5	12749.2	X	x	x	X	x	x	X
Crossing	Turning_Left	2_Dark_Lit	Obese&fit	1.8&1.5	6763.66				X	x	x	X
Crossing	Straight	1_Daylight	Kid	1.8&1.5	5757.95					x	x	X
Crossing	Straight	3_Dark_Unlit	Obese&fit	1.8&1.5	3362.42						x	X
Crossing	Turning_Right	1_Daylight	Obese&fit	1.8&1.5	3200.67							X
Standing	Straight	1_Daylight	Obese&fit	0	2920.13			x		x	x	X
Walk/Run	Straight	2_Dark_Lit	Obese&fit	1.8&1.5	1735.32							
_with_Vehicle												
Walk/Run	Straight	1_Daylight	Obese&fit	1.8&1.5	1474.45							
_with_Vehicle												
Walk/Run	Straight	3_Dark_Unlit	Obese&fit	1.8&1.5	1431.67							
_with_Vehicle												
Standing	Straight	2_Dark_Lit	Obese&fit	0	1251.88							
Crossing	Turning_Right	2_Dark_Lit	Obese&fit	1.8&1.5	1113.52							
Standing	Straight	3_Dark_Unlit	Obese&fit	0	957.76							
Crossing	Straight	1_Daylight	Obese&fit	2.2	836.62							
Standing	Turning_Left	1_Daylight	Obese&fit	0	835.69							
Standing	Straight	1_Daylight	Kid	0	731.27				X			
Crossing	Straight	2_Dark_Lit	Kid	1.8&1.5	667.35644							
Walk or Run_against_Veh.	Straight	3_Dark_Unlit	Kid	1.8&1.5	0.14							
Crossing	Turning_Right	3_Dark_Unlit	Kid	1.8&1.5	0.13							
Walk or Run_against_Veh.	Turning Right	1 Daylight	Kid	1.8&1.5	0.09							

Table 6.2.: Total cost percentage covered by the selected scenarios for the cases that the constraints are considered / not considered

Numbers of desired scenarios (P)	Total percentage of cost covered by P scenarios selected based on satisfying the constraints while maximizing the total cost	Total percentage of cost covered by P scenarios selected based on maximizing the total cost without considering the constraints	Difference of percentage coverage between two methods
2(% Coverage)	31.76269878	40.17018626	8.407487475
3(% Coverage)	55.31950461	55.31950577	1.16E-06
4(% Coverage)	58.78936066	63.35644635	4.567085691
5(% Coverage)	64.22538523	70.19835155	5.972966321
6(% Coverage)	73.66821224	74.19375796	0.525545717
7(% Coverage)	77.66361488	77.99696505	0.333350176
8(% Coverage)	81.46682691	81.46682691	0

6.1 Future Work

Even though we have shown that our GA approach is reliable, more data analysis can be done to improve the algorithm. The size of the Initial Population (tune1), the rate of Cross Over and Mutation (tune2 & tune3), and the total size of the chromosomes space can be tuned based on the values of N and P and the nature of the data in a way that the program works more efficient and faster. Figure 6.1 shows the percentage of hitting the best solution using GA-based approach for different sizes of initial population for the case that P is equal to 5. The result clearly indicates that, as the size of initial population grows, the hit percentage grows and for the case that tune1 equals to 500,000 the percentage of success is 93.7 percent. Of course for P=5, since the size of the memory we allocated is huge compared to the size of the problem, we can increase the likelihood of success very close to one hundred percent whereas this is not possible for larger problems. Also the bigger the size of the initial population and the size of the chromosome space, the slower the program will be. For the case corresponding to Figure 6.1, we do not need to make the size any bigger than 5,000,000. Thus it is always a tradeoff between the size and the speed of the program which can be optimized by data analysis for different cases. Also for larger values of P and N, we can keep the size of the initial population relatively large and, by repeating the experiment and choosing the best of N trials, we can increase the likelihood of success. Additional data analysis can be done for different size of the tuning variables to find the numbers of God-Chros, Kings, Lords, Middle-Class and find relations to find the best tuning values.

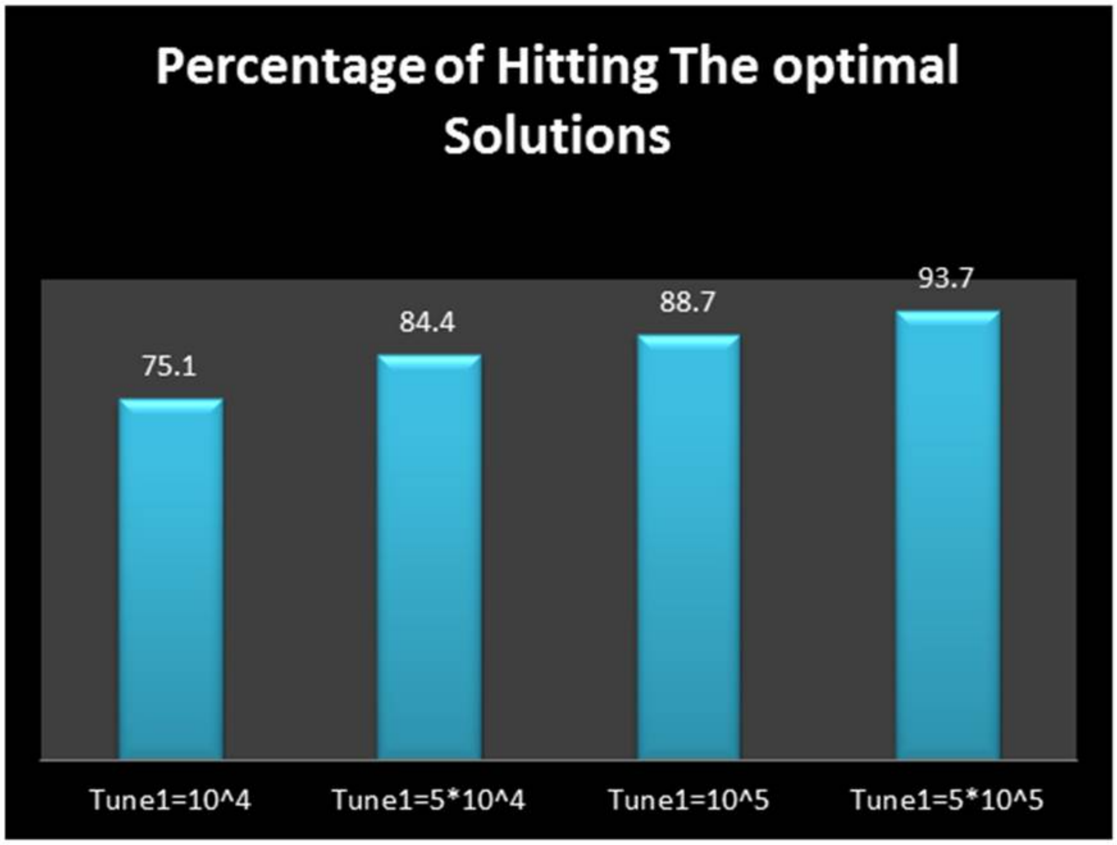


Fig. 6.1.: Percentage of hitting the optimal solution for P=5 and different sizes of initial population

LIST OF REFERENCES

LIST OF REFERENCES

[1] J. Clanton, “A low-cost solution for an integrated multisensor lane departure warning system,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, pp. 47–59, 2009.

[2] C. Desjardins and B. Chaib-draa, “Cooperative adaptive cruise control: A reinforcement learning approach,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, pp. 1248–1260, 2011.

[3] H. Xiong and L. N. Boyle, “Drivers adaptation to adaptive cruise control: Examination of automatic and manual braking,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, pp. 1469–1473, 2012.

[4] S. Nedevschi, “Stereo-based pedestrian detection for collision-avoidance applications,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, pp. 380–391, 2009.

[5] T. Gandhi and M. M. Trivedi, “Pedestrian protection systems: Issues, survey, and challenges,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, pp. 413–430, 2007.

[6] C. Keller, “Active pedestrian safety by automatic braking and evasive steering,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, pp. 1292–1304, 2011.

[7] D. Geronimo, “Survey of pedestrian detection for advanced driver assistance systems,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 1239–1258, 2010.

[8] R. Tian, “Study on the display positions for the haptic rotary device-based integrated in-vehicle infotainment interface,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, pp. 1234–1245, 2014.

[9] L. Lingxi, “Studying the effects of driver distraction and traffic density on the probability of crash and near-crash events in naturalistic driving environment,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, pp. 1547–1555, 2013.

[10] S. Chien, “A novel evaluation methodology for combined performance of warning and braking in crash imminent braking systems,” *IEEE Intelligent Transportation Systems Magazine*, vol. 5, pp. 62–72, 2013.

[11] A. Ari, “A new scoring mechanism for vehicle crash imminent braking systems,” *IEEE Intelligent Transportation Systems Magazine*, vol. 4, pp. 17–29, 2012.

[12] [Online]. Available: http://www.autoblog.com/2013/10/13/toyota-pre-collision-system-steering-assist/ Last Date Accessed: 12/01/2014

[13] [Online]. Available: http://en.wikipedia.org/wiki/Pre-CollisionSystem Last Date Accessed: 12/01/2014

[14] [Online]. Available: http://www.kbb.com/car-news/all-the-latest/2014-volvos-will-offer-new-cyclist-detection-safety-system-option/2000009157/ Last Date Accessed: 12/01/2014

[15] R. Tian, "Estimation of the vehicle/pedestrian encounter/conflict risk on the road based on tasi-110 car naturalistic driving data collection," in *Proc. 2014 IEEE Intelligent Vehicles Symposium*, 2014.

[16] F. J. Villegas, "Parallel genetic-algorithm optimization of shaped beam coverage areas using planar 2-d phased arrays," *IEEE Transactions on Antennas and Propagation*, vol. 55, 2007.

[17] J. M. Johnson and Y. Rahmat-samii, "Genetic algorithm optimization and its application to antenna design," *Antennas and Propagation Society International Symposium*, vol. 1, 1994.

[18] S. Chen, "Genetic algorithm optimization for blind channel identification with higher order cumulant fitting," *IEEE Transactions on Evolutionary Computation*, vol. 1, 1997.

[19] J. M. Johnson and Y. Rahmat-Samii, "Genetic algorithms in engineering electromagnetics," *IEEE Antennas and Propagation Magazine*, vol. 39, 1997.

[20] C.-F. Juang, "A hybrid of genetic algorithm and particle swarm optimization for recurrent network design," *IEEE Transactions On Systems, Man, And CyberneticsPart B: Cybernetics*, vol. 34, 2004.

[21] H. M. Hasanien, "Design optimization of pid controller in automatic voltage regulator system using taguchi combined genetic algorithm method," *IEEE Systems Journal*, vol. 7, 2013.

[22] D.-J. Sim, "Application of vector optimization employing modified genetic algorithm to permanent magnet motor design," *IEEE Transactions On Magnetics*, vol. 33, 1997.

[23] D. R. Bull, "Optimization of image coding algorithms and architectures using genetic algorithms," *IEEE Transactions On Industrial Electronics*, vol. 43, 1996.

[24] J.-T. Tsai, "Tuning the structure and parameters of a neural network by using hybrid taguchi-genetic algorithm," *IEEE Transactions On Neural Networks*, vol. 17, 2006.

[25] L. Bernal-Agustin, "Genetic algorithms applied to the design of large power distribution systems," *IEEE Transactions on Power Systems*, vol. 13, 1998.

[26] H. Hou, "A genetic algorithm for multiprocessor scheduling," *IEEE Transactions On Parallel And Distributed Systems*, vol. 5, 1994.

[27] "Accuracy and performance evaluation in the genetic optimization of nonlinear systems for active noise control," *IEEE Transactions On Instrumentation And Measurement*, vol. 56, 2007.

[28] A. Arabali, “Genetic-algorithm-based optimization approach for energy management,” *IEEE Transactions On Power Delivery*, vol. 28, 2013.

[29] G. S. Tewolde, “Robot path integration in manufacturing processes: Genetic algorithm versus ant colony optimization,” *IEEE Transactions On Systems, Man, And CyberneticsPart A: Systems And Humans*, vol. 38, 2008.

[30] R. Elbaum and M. Sidi, “Topological design of local-area networks using genetic algorithms,” *IEEE Transactions On Networking*, vol. 4, 1996.

[31] C. J. Fourie and W. J. Perold, “Comparison of genetic algorithms to other optimization techniques for raising circuit yield in superconducting digital circuits,” *IEEE Transactions On Applied Superconductivity*, vol. 13, 2003.

[32] C. Jin, “Software reliability prediction based on support vector regression using a hybrid genetic algorithm and simulated annealing algorithm,” *IET Software*, vol. 5, pp. 398–405, 2011.

[33] B. Bhanu, “Adaptive image segmentation using a genetic algorithm,” *IEEE Transactions On Systems, Man, And Cybernetics*, vol. 25, 1995.

[34] B. Ozpineci, “Harmonic optimization of multilevel converters using genetic algorithms,” *IEEE Power Electronics Letters*, vol. 3, 2005.

[35] Y. A. Hussein, “Modeling and optimization of microwave devices and circuits using genetic algorithms,” *IEEE Transactions On Microwave Theory And Techniques*, vol. 52, 2004.

[36] F. Qiao, “A petri net and extended genetic algorithm combined scheduling method for wafer fabrication,” *IEEE Transactions On Automation Science And Engineering*, vol. 10, 2013.

[37] J. Yen, “A hybrid approach to modeling metabolic systems using a genetic algorithm and simplex method,” *IEEE Transactions On Systems, Man, And CyberneticsPart B: Cybernetics*, vol. 28, 1998.

[38] S. Obayashi, “Multiobjective genetic algorithm applied to aerodynamic design of cascade airfoils,” *IEEE Transactions On Industrial Electronics*, vol. 47, 2000.