Evolutionary Algorithms for Solving Multi-Objective Shortest Path Problem

- Case Study of Vehicle Navigation Problems

PhDの論文

学生: Umair Farooq Siddiqi (ウメル　ファロク　セディキ)
学生番号: 10802273

先生: Prof. Wei Shu Gang and Prof. Yoichi Shiraishi
ウェイ先生、 白石先生

群馬大学
GUNMA UNIVERSITY
工学学部
群馬大学、日本

Evolutionary Algorithms for Solving Multi-Objective Shortest Path Problem - Case Study of Vehicle Navigation Problems

Student: Umair Farooq Siddiqi
Student Id: 10802273

Supervisors: Prof. Wei Shu Gang and Prof. Yoichi Shiraishi
Faculty of Engineering
Gunma University, Japan

November 2012

Contents

1 Introduction **2**

2 Existing Algorithms to Solve the MOSP Problem **6**
2.1 Polynomial Time Approximation Algorithms (PTAAs) 6
2.2 Evolutionary Algorithms 7
2.2.1 Single Solution based Algorithms 7
2.2.2 Population based Algorithms 8

3 Problem Description and Performance Measurement **10**
3.1 Problem Description 10
3.2 Performance Measurement 11

4 Proposed Algorithms **14**
4.1 Method to generate a random paths 14
4.2 Proposed StocE based Algorithm 14
4.2.1 Perturb Operation 17
4.2.2 Mutation 19
4.2.3 Store pareto-optimal solutions 20
4.3 Proposed Off-Storing Non-Storing GA 20
4.3.1 Initialization 21
4.3.2 Mark Pareto-Optimal Solutions 21
4.3.3 Selection of the GA Operation 21
4.3.4 GA Operations 23
4.3.5 Crossover Operation 26
4.3.6 Mutation Operation 26
4.4 Estimation of the Memory Requirements 26
4.5 Comparison of Memory Requirements with Some Typical Values 27
4.6 Summary 27

5 Application of the Proposed Algorithms to the Vehicle Navigation Problem of Conventional Vehicles **29**
5.1 Introduction 29
5.2 Implementation of the existing algorithms 30
5.3 Proposed StocE based Algorithm 31
5.4 Proposed Off-Spring Non-Storing GA 32
5.5 Summary 33

6 Application of the Proposed Algorithms to the Vehicle Navigation Problem of Battery Electric Vehicles 42
6.1 Introduction . . . 42
6.2 Proposed StocE Algorithm . . . 44
6.3 Proposed Off-Spring Non-Storing GA . . . 45
6.4 Summary . . . 46

7 Generalization of the Proposed Algorithms 55
7.1 Proposed StocE-based Algorithms . . . 56
7.1.1 Perturb Operation . . . 56
7.1.2 Mutation . . . 58
7.2 Off-Spring Non-Storing GA . . . 58
7.2.1 Selection of the GA Operation . . . 58
7.3 Simulations . . . 59
7.3.1 Test Problems . . . 59
7.3.2 Algorithm Parameters . . . 60
7.3.3 Results of the Test Problems . . . 61
7.4 Effect of Memory Size on the Solution Quality . . . 63
7.5 Summary . . . 68

8 Congestion Awareness in case of Multi-Vehicles Problem 71
8.1 Problem Description . . . 72
8.2 Proposed Algorithm . . . 73
8.3 Simulation Results . . . 75
8.4 Summary . . . 77

9 Conclusion 78

List of Figures

1.1 Illustration of types of components in EAs, PTAAs and PTEAs . 3
1.2 Illustration of types of EAs 4

3.1 Illustration of the HV metric. 12

4.1 Method to find a random path: $y = form\_path(s, d)$. 15
4.2 Proposed StocE based Algorithm 16
4.3 Function for the selection of a sub-path $y = select\_subpath(S, Num, P_b)$. 18
4.4 Proposed Perturb operation $S' = Perturb(S, Num, P_b)$. 19
4.5 Mutation Operation. 19
4.6 Proposed GA based Algorithm 22
4.7 Method used to find if P_i has a feasible pair in the population $y = CheckCn(P_i, Population, s)$. 23
4.8 Method used to select a GA operation for the chromosome P_j, $OPER = SelectOperation(P_j)$. 24
4.9 Procedure to select a feasible pair for P_j, i.e., $y = findpair(P_j, Population, s)$ 24
4.10 Procedure to apply the crossover operation to particle P_j, i.e., $c' = Crossover(P_j, Population)$ 25
4.11 Mutation operation, i.e., $c' = mutation(P_j)$ 25

5.1 Results of the HV ratios for the proposed StocE-based algorithm on the BAY road network. 34
5.2 Results of the HV ratios for the proposed StocE-based algorithm on the COL road network. 35
5.3 Results of the HV ratios for the proposed StocE-based algorithm on the NY road network. 36
5.4 Summary of the HV ratio results of the Proposed StocE-based Algorithm. 37
5.5 Results of the HV ratios for the proposed Off-Spring Non-Storing GA algorithm on the BAY road network. 38
5.6 Results of the HV ratios for the proposed Off-Spring Non-Storing GA algorithm on the COL road network. 39
5.7 Results of the HV ratios for the proposed Off-Spring Non-Storing GA algorithm on the NY road network. 40
5.8 Summary of the HV ratio results of the Proposed Off-Spring Non-Storing GA Algorithm. 41

6.1 Function $g_1(B_{ini}, P)$. 44

6.2 Results of the HV ratios for the proposed StocE-based algorithm on the BAY road network. 47
6.3 Results of the HV ratios for the proposed StocE-based algorithm on the COL road network. 48
6.4 Results of the HV ratios for the proposed StocE-based algorithm on the NY road network. 49
6.5 Summary of the HV ratio results of the Proposed StocE-based Algorithm. 50
6.6 Results of the HV ratios for the Proposed Off-Spring Non-Storing GA on the BAY road network. 51
6.7 Results of the HV ratios for the Proposed Off-Spring Non-Storing GA on the COL road network. 52
6.8 Results of the HV ratios for the Proposed Off-Spring Non-Storing GA on the NY road network. 53
6.9 Summary of the HV ratio results of the Proposed Off-Spring Non-Storing GA Algorithm. 54

7.1 Proposed StocE-based algorithm for the general MOP problems. 57
7.2 Perturb Operation for the general MOP, $SA = Perturb(S, Num)$ 58
7.3 Method to select a GA operation for the chromosome P_i, $GAOper = SelectGAOper(P_i)$ 59
7.4 Results of the calculation of HVR, GD and IGD metrics on the test problems for the experiments in Case I. 64
7.5 Summary of the results in Case I. 65
7.6 Results of the calculation of HVR, GD and IGD metrics on the test problems for the experiments in Case II. 66
7.7 Summary of the results in Case II. 67
7.8 HVR, GD and IGD metric values of the Proposed StocE-based algorithm at different memory sizes. 69
7.9 HVR, GD and IGD metric values of the Proposed Off-Spring Non-Storing GA algorithm at different memory sizes. 70

8.1 Illustration of the proposed congestion minimization method. . . 74
8.2 Proposed algorithm for congestion minimization. 74
8.3 Method to initialize a random solution in U 75
8.4 Method to perform mutation operation on a strategy profile U . 75
8.5 Illustration of the gradual decrease in the congestion cost as iterations proceeds. 76

List of Tables

3.1 Details of the Road Networks 12
3.2 Variables representing the HV of the algorithms 12

4.1 Memory requirements of the algorithms 26
4.2 Memory requirements based on typical values 27

6.1 Charging rate of the BEVs using different types of CPTs 43

7.1 Description of test problems 60
7.2 Number of variables (i.e., value of n) used in the experiments. . . 61
7.3 Values of parameters. 62
7.4 Parameter values and memory sizes. 68

8.1 Characteristics of graphs. 76
8.2 Results of the congestion reduction algorithm 76

Abstract

Finding Multi-objective shortest paths (MOSP) is an important problem in computer and transportation networks. MOSP is an NP-hard problem when it contains more than two objectives. MOSP problem can be efficiently solved using the evolutionary algorithms (EAs). The existing EAs are of two types: Population-based and single-solution-based. Population-based EAs are memory-intensive and single-solution-based EAs cannot yield good quality solutions within a small amount of time. We proposed two new EAs to solve the MOSP problem and overcome the shortcomings of the existing EAs. The proposed EAs require lesser memory and at the same time can also yield good quality solutions. The first algorithm is based on Stochastic Evolution (StocE) and works on a single solution. It considers different sub-paths in the solution as its characteristics and eliminates bad sub-paths from generation to generation. The second proposed algorithm is an off-spring non-storing GA which is memory-efficient than the existing GAs and its variants. Unlike existing GA-based algorithms it does not store children chromosomes in the memory. In the proposed GA-based algorithm, the children chromosomes conditionally replace their parent chromosomes and thus do not need to be stored at new memory locations. The quality of the pareto-optimal sets of the proposed algorithms is determined by using the Hypervolume metric. This works considers two applications in which the MOSP problem occurs. The first problem is the selection of optimal paths in the conventional vehicles and the second problem is the selection of optimal paths in the electric vehicles. The proposed algorithm outperforms the existing single-solution-based EAs in solution quality and requires lesser memory than the population-based algorithms. The proposed algorithms can also be generalized to solve any multi-objective optimization problems. The proposed algorithm can solve complicated test problems of multi-objective optimization with a quality which is competitive to the existing popular EAs. The effect of memory size on the solution quality is also studied. It is found that excessive increase in the memory size does not improve the solution quality. The experimental results show that the proposed StocE and GA based algorithms are highly suitable to solve the MOSP problem in embedded systems.

Acknowledgements

First and foremost, I would like to praise Almighty Allah for His blessings and help throughout my life.

I would like to express my sincere gratitude to my advisor Prof. Yoichi Shiraishi for his kind support and guidance throughout my PhD program. I would also like to express my gratitude to the committee members: Prof. Wei Shu Gang, Prof. Yoshikuni Onozato, Prof. Koichi Yamazaki and Prof. S. Matsumura. I would also like to thanks my MS thesis advisor Prof. Sadiq Sait, Department of Computer Engineering, King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia for his help in my research work.

I would also like to thanks other PhD and MS students in our laboratory, specially, Dr. Mona Dahb for her guidance in different steps of the PhD course. Acknowledgements are also due to KDDI Foundation, Japan for providing the financial support.

I would also like to acknowledge my mother for her motivation and guidance at every stage of my life. I would also like to thanks my wife, Amber and daughter, Sireen for their patience. I would also like to acknowledge the moral support of my siblings and friends in Pakistan.

Lastly, I would like to appreciate everyone who contributed in my PhD studies and research.

Chapter 1

Introduction

The shortest path problem in its simplest form refers to finding a path between any two nodes in a network, such that the sum of the weights of its edges is minimized and the constraint on the sum of the weights of its edges is also satisfied. When the edges have only one weight associated with them, then the shortest path problem is called a single objective shortest path or simply a shortest path problem and can be accurately solved using polynomial time algorithms like Dijkstra's Algorithm [1], etc. However, when the edges contain two or more weights and the problem contains two or more objectives then the problem of shortest path is called as Multi-Objective Shortest Path (MOSP) problem.

MOSP problem is an important operation in transportation networks and computer networks. Vehicles use the shortest paths to reach their destinations in lesser amount of time and/or using lesser fuel. In computer networks, the data transfer rate can be increased dramatically by using shortest paths between the source and destination nodes.

MOSP is an NP-hard problem [8, 3] and its approximate solutions should be determined using heuristics. MOSP problem can be solved using three different types of algorithms:

* Evolutionary Algorithms (EAs)
* Polynomial Time Approximation Algorithms (PTAAs)
* Polynomial Time Exact Algorithms (PTEAs)

The EAs mimics the biological process of evolution in search for the optimum solution. First and the most important advantage of solving any multi-objective optimization problems (MOPs) with the EAs is that an EA that is developed to solve any particular MOP remains useful for the other MOPs. The MOPs can be entirely different from each other. The EAs are defined in terms of evolutionary operators to produce useful results. The evolutionary operators consists of functions or operations that are problem-specific. Therefore, the design of the EA is not dependent on the problem. For instance, Non-dominated sorting genetic algorithm-II (NSGA-II) [4] which is a popular EA was used to solve many MOPs that include: Multi-objective Electromagnetic Optimization [5], Service restoration in distribution systems [6], and optimal application mapping on NoC infrastructure[7]. Therefore, many problems or different versions

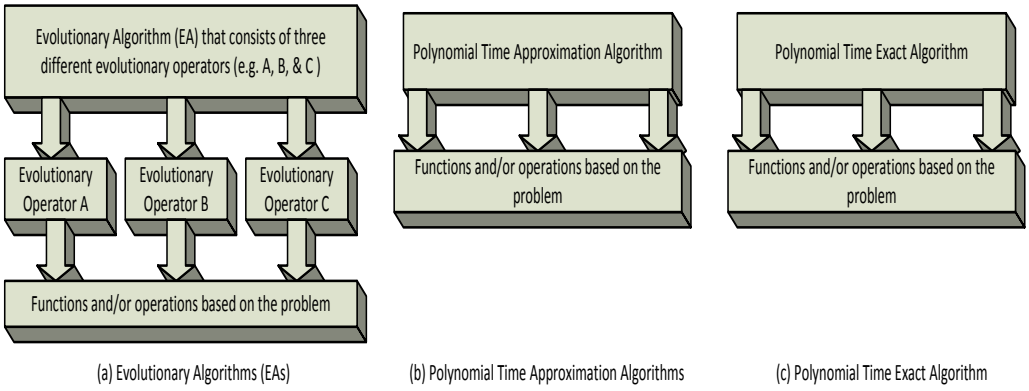


Figure 1.1: Illustration of types of components in EAs, PTAAs and PTEAs

of the same problem can be efficiently solved using a same EA. The efforts required to developed optimization algorithms for different problems also reduces significantly. EAs also do not require any pre-computation and calculation in any generation are independent from the previous generations. Therefore, they are robust to dynamic changes in the network.

PTAAs aim to approximately solve the MOSP problem within the bounds of the polynomial amount of time. PTAAs guaranteed to solve any MOP with some solution quality. Most of the PTAAs are described in terms of functions or operations that are based on the problem [8, 9]. Therefore, the FTAAs are specific to one or certain problems. FTAAs require pre-computation of some values. In case of dynamic changes in the network, the computation of the optimal paths should be restarted. Therefore, they are not robust to dynamic changes in the network.

PTEAs are suitable for small size networks only. In huge size networks, their time complexity becomes impractical for most real-time operations. Martin's algorithm [13] is a PTEA to solve the MOSP problem. The Dijkstra's Algorithm can solve the shortest path problem in networks and has a time complexity which is better than the other polynomial time exact algorithms. The authors implemented the Dijkstras Algorithm on the nVIDA CUDA graphics processing unit (GPU) and its execution time came out to be around 8 minutes on the road network of the New York City. Gunichev et al. [12] also found the execution time of the Dijkstra's Algorithm to be eight minutes. Eight minutes is not suitable for most of the real-time operations. Ahn, at al. [14] reported that the time complexity of the Dijkstra's Algorithm increases with the number of nodes in the network. In huge size networks, Genetic Algorithm (GA) becomes faster than the Dijkstra's Algorithm.

The structure of the EAs, PTAAs and PTEAs is shown in Fig. 1.1. The EAs comprises of evolutionary operators and therefore, their design is independent from the specific properties of the problem. PTAAs and PTEAs on the other hand, contains functions or operations that are based on the problem and can change from problem to problem.

The MOSP problem occurs in many applications and may take different

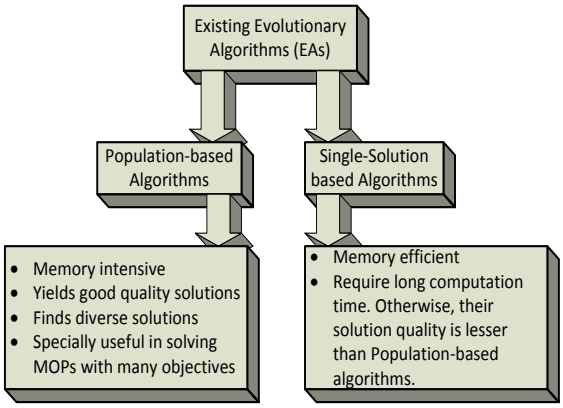


Figure 1.2: Illustration of types of EAs

forms. Therefore, the objective functions and constraints can take different forms. MOSP problem also occurs in emerging fields like electric vehicles, where new enhancements are continue to happen. Therefore, the MOSP problem is subject to further modification in the future. The EA approach is specially suitable for such kind of problems because it can incorporate changes in the objective function or constraints without effecting the top-level algorithm. The EA approach also enables that the optimization algorithms to remain useful for the other and unrelated optimization problems.

Many EAs exists to solve the MOPs. The existing EAs can be divided into two types: The first type of EAs works on a single solution and are memory efficient. Simulated Evolution (SimE) and Stochastic Evolution (StocE) [16, 17] are examples of single-solution based EAs. Single solution EAs are suitable for use in embedded systems that have limited memory and computational power. The second type of EAs work on a population of solutions. They are memory intensive but generally yields results better than the single solution based algorithms. Genetic Algorithm (GA) [16] is an example of population based EA.

This thesis presents two new EAs to solve the MOSP problem. The new EAs aims to be memory-efficient as well as yield quality solutions to the MOSP problem. The two important properties of the proposed algorithms are as follows:

* Memory-efficient than the existing population-based EAs

* Achieves better quality than the existing single-solution based algorithms.

The first algorithm is based on Stochastic Evolution (StocE) algorithm. StocE was first proposed by Youssef G. Saab and Vasant B. Rao [17] in 1990 for solving combinatorial optimization problems. StocE algorithm works on a single solution and resembles a biological evolutionary process in which species eliminate some of the bad characteristics of the older generation in order to produce a better new generation. The proposed StocE based algorithm considers

different sub-paths in a solution as its different characteristics and removes bad sub-paths with new sub-paths from generation to generation.

The second algorithm is an off-spring non-storing GA algorithm. GAs was first proposed by John Holland and his colleagues in the early 1970s [18]. GA simulates the process of natural evolution based on Darwinian principles. The conventional GA algorithms store children chromosomes in the memory and require a total memory of size about twice the population size. The proposed off-spring non-storing GA algorithm, on the other hand, conditionally store children chromosomes in place of the parent chromosome. Therefore, it requires memory almost equal to the size of the population. The memory-efficiency in the proposed algorithms makes them suitable to solve the MOSP problem in embedded systems.

The performances of the proposed algorithms were compared with some famous MOO algorithms: (i) (1-1)-Pareto-Archived Evolution Strategy ((1-1)- PAES) algorithm [20], (ii) Non-dominated Sorting Genetic Algorithm-II (NSGA-II) [4], (iii) Micro-GA [21], (iv) Multi-objective Simulated Annealing (MOSA) [22], and (v) A straight-forward StocE (std-StocE). NSGA-II was successfully used in many applications like: multi-objective electromagnetic problem that consists of many conflicting objectives [5] and optimization problem in distribution systems [6]. Micro-GA was also used to solve important problems like optimization problem in Network-on-Chip (NoC) [7]. PAES was also successfully used to perform optimization required by the AI (artificial intelligence) in video games [23]. The comparison with Std-StocE shows the benefit of using the proposed design of StocE over the standard StocE algorithm.

This dissertation is organized as follows: The second chapter describes some popular multi-objective optimization algorithms. The third chapter describes the MOSP problem in detail and a method of calculating and comparing the performances of different multi-objective optimization algorithms. The fourth chapter contains the description of the proposed algorithms. The fifth and sixth chapters show the application of the proposed algorithms to solve the navigation problem in conventional vehicles and battery electric vehicles. The seventh chapter shows the general forms of the proposed StocE and GA-based algorithms and their comparison with the existing algorithms on several test problems. The eighth chapter discusses the problem of congestion that occurs due to a large number of vehicles and a simple method to minimize the congestion. The last chapter contains the conclusion.

Chapter 2

Existing Algorithms to Solve the MOSP Problem

This chapter describes some existing algorithms to solve the MOSP problem. Two types of algorithms are discussed that includes: evolutionary algorithms and polynomial time approximation algorithms.

2.1 Polynomial Time Approximation Algorithms (PTAAs)

Mandow et al. [9] presented initial results of extending the A* search algorithm to the solution of MOSPs. The new algorithm is named MOA*, which is a heuristic search algorithm used to find non-dominated solutions. The search process in MOA* is guided by heuristic functions. When the guiding heuristic does not meet a certain bounding test, MOA* becomes unreliable and cannot produce any useful solution. However, it is reliable when used with a proper set of heuristics. Tsaggouris and Zaroliagis [10] proposed an improved Fully Polynomial Time Approximation Scheme (FPTAS) algorithm for solving MOSPs. Their algorithm resembles the multi-objective Bellman-Ford algorithm. Among FTPASs, it has the best time complexity. Horoba [11] performed an analysis of a simple evolutionary algorithm that consists of a fitness function and mutation operation and found that it met the requirements of a Fully Polynomial Time Randomized Approximation Scheme and its runtime was comparable to that of Tsaggouris and Zaroliagis's algorithm [10]. The conventional FTPAS requires pre-computation of some values (e.g., node labels which includes cost of links) before determining the optimal path. In case of dynamic changes, the cost of links can change to reflect dynamic changes in the network. Therefore, some or many of the node labels becomes invalid. The conventional FTPAS does not robustly accommodate dynamic changes and the calculation of the shortest path must be restarted several times whenever there are dynamic changes in the network. Because simple EC algorithms perform well and robustly accommodate dynamic changes in the network, they are often used to solve MOSPs with dynamic changes.

2.2 Evolutionary Algorithms

Elitist Evolutionary Multi-objective Optimization (EMO) algorithms are the most recent EMOs used for finding Pareto-optimal solutions. In elitist EMO algorithms, good solutions are preserved during iterations. Recent EMO algorithms can be classified into two classes: (i) single solution based algorithms, and (ii) population based algorithms. The single solution based EMOs work on a single solution and therefore, are quite memory efficient. The population based EMOs work on a population of solutions. They require more memory than the single solution based algorithms but also yield good quality solutions. In this chapter, some popular single solution based and population based algorithms are described.

2.2.1 Single Solution based Algorithms

(1-1)- PAES

Knowles and Corne [20] proposed a multi-objective local search algorithm which they named as: (1-1)-Pareto Archived Evolution Strategy ((1-1)-PAES) algorithm. (1-1)-PAES is capable of generating diverse Pareto-optimal solutions. The (1-1)-PAES maintains an archive of Paret-optimal solutions of size m solutions. Each solution in the archive has a position in the n-dimensional grid (where n is the number of objectives). The PAES has three portions: (1) the candidate solution generator; (2) the candidate solution acceptance; and (3) the non-dominated-solutions (NDS) archive. The candidate solution generator consists of a simple mutation operation and in each iteration it creates a new solution (called as mutant solution) through performing random mutation in the solution. The design of the candidate solution acceptance function is as follows: If the mutant solution dominates the existing solutions then it is accepted and the current solution is updated. If the mutant solution does not dominate the existing solution then the following rules are applied: If the mutant solution lies in a less crowded region of the archive then the existing solution is updated to the mutant solution. The NDS archive of m solutions is maintained. However, when the archive becomes full then the following method is used to remove a solution from the archive to make space for the new solution (c). The grid position of c in the archive is determined. If c lies in a less crowded region than any solution in the archive of pareto-optimal solutions then a solution that lies in the most crowded region is removed and c is inserted to the archive. The experimental results show that despite being simple, it is very effective in solving multi-objective optimization problems.

MOSA

Smith, et al. [22] proposed a multi-objective simulated annealing (MOSA) algorithm. The algorithm keeps an archive of pareto-optimal solutions. They proposed a transformation function to transform a multi-objective solution into an energy value. A solution can be compared with another solution based on its energy value and a solution can be better, equal or worse than the other solution. The energy of any solution is determined by calculating the number of solutions in the archive of pareto-optimal set that dominates it. Therefore, the solutions having higher energy values are considered inferior. All solutions

in the archive of pareto-optimal solutions are mutually non-dominating. A new solution is accepted to replace the existing solution according to an exponential equation of energy difference. When the archive of pareto-optimal solutions contains too few solutions then the efficiency of the energy difference equation becomes low; therefore, new solutions (real and/or artificial) are considered to be a part of the archive of pareto-optimal solutions. Two methods can be used to increase the number of solutions in the archive of pareto-optimal solutions: (i) Conditional removal of the dominated solutions from the archive, and (ii) Creation of artificial solutions through linear interpolation. In the first method, the dominated solutions whose removal reduces the size of archive below some specified limit are retained in the archive. In the second method, linear interpolation is used to generate new points. The new hypothetical points are evenly spread along the pareto-optimal front and dominated by at-least one actual solution in the archive of the pareto-optimal solutions.

2.2.2 Population based Algorithms

Genetic Algorithm (GA) is a famous population based algorithm. This section shows some famous variants of GA to solve the MOO problem.

Micro-GA

Coello and Pulido [21] proposed Micro-GA for high quality multi-objective optimization. It stores solutions in two types of memories. The first type is the population memory and the second type is an external memory to store pareto-optimal solutions. The population memory is further divided into a replaceable portion and a non-replaceable portion. The replaceable portion contains the solutions that can be altered during the optimization; whereas, the non-replaceable portion contains the solutions that cannot be changed after being initialized in the initialization step. The Micro-GA contains a micro-GA cycle that works on a separate population of solutions in which the elements are selected from the replaceable and non-replaceable portions of the population memory. The micro-GA cycle consists of the conventional GA operators (crossover and mutation) and executes for a smaller number of iterations. When the micro-GA cycle finishes, up to two new pareto optimal solutions are copied from its population to the replaceable portion of the population memory and to the external archive of pareto-optimal solutions. In the next micro-GA cycle, the elements of the population of the micro-GA cycle are again selected from the population memory. When the stopping criteria of the Micro-GA is reached then algorithm goes to the termination instead of initiating another micro-GA cycle.

NSGA-II

Deb, et al. [4] proposed Non-dominated Sorting Genetic Algorithm-II (NSGA-II) to perform multi-objective optimization. NSGA-II has low computational complexity. During its iterations, it preserves two types of solutions: non-dominated solutions and solutions that are most distinct in the population. By doing so, the algorithm maintains both quality and diversity among its solutions. An iteration cycle consists of the following steps: (i) The population (including children chromosomes) is sorted according to the non-domination count of

the solutions. The chromosomes are assigned pareto front values such that the non-dominated solutions are selected in the first front. The solutions that are non-dominated after removing the chromosomes of the first front are placed in the second front. The pareto-front values to the remaining solutions are also assigned in the same way. (ii) The crowding distances of the chromosomes are calculated based on the distances between the objective function values of the chromosomes. The crowding distance helps in developing a uniformly spread pareto optimal front. (iii) The chromosomes for the population of the next iteration are selected from the combined population (i.e., population and children chromosomes). The solutions having lesser pareto-front values are first selected and the solutions of the last front are selected based on the crowding distance. The children chromosomes are created using crossover and mutation operations. Experimental results have shown that this algorithm is very successful in finding diverse Pareto-optimal sets of solutions for multi-objective optimization problems. Bora, et al. [24] added greedy reinforcement learning to NSGA-II for self-tuning its parameters. Their new algorithm, NSGA-RL, tunes four NSGA-II parameters—the probabilities of crossover and mutation operations and the distribution indices in crossover and mutation operations—on the basis of the results of previous generations. NSGA-RL is slower than NSGA-II; however, its results are closer to the NSGA-II results that have the best possible parameter values.

Chapter 3

Problem Description and Performance Measurement

This chapter describes the MOSP problem in detail. It also describes the methods of measuring the solution of different algorithms. The result of any MOSP problem is a set of pareto-optimal solutions. The quality of the pareto-optimal solutions obtained by different algorithm can vary, therefore, calculating the performance of the pareto-optimal sets is important.

3.1 Problem Description

Let us consider an undirected graph $G(V, E)$, where V is the set of all vertices in the network and E is the set of all edges in the network. An edge $e_x \in E$ has a starting node and an ending node. For any edge e_i, the starting node is represented as $e_i.st$ and the ending node is represented as $e_i.en$, where $st, en \in V$. Any edge $e_i \in E$ has up to K weights associated with it, which are represented as $\{e_i.w_1, e_i.w_2, ..., e_i.w_K\}$. The weights of the edges should be non-negative real numbers. One or more point-to-point (P2P) paths exist between the nodes in the graph. If s and d (where $s, d \in V$) are two distinct nodes in G, which are selected as the source and destination nodes. Then many paths exist between s and d. A path is represented by P and is a sub-set of E, i.e., $P \subseteq E$. If e_x is the first edge in P and e_y is the last edge in P, then $e_x.st$ should be s and $e_y.en$ should be d. For any two consecutive edges e_m and e_n edges in P (s.t., e_n lies after e_m in P), $e_m.en = e_n.st$.

In the multi-objective shortest path problem (MOSP), the path P is associated with up to K_1 (where $1 < K_1 \leq K$) objective functions and up to K_2 constraints (where $0 \leq K_2 \leq K$). The objective functions can be represented as f_1, f_2, ..., f_K. The value of any objective function can be calculated as:

$$f_k(P) = \sum_{e_x \in P} e_x.w_k, \text{for } k\text{= 1 to } K_1 \tag{3.1}$$

The set of constraints are represented as $g_1(P)$, $g_2(P)$, ..., $g_M(P)$. The value of any constraint $g_k(P)$ can be determined as:

$$g_k(P) = \sum_{e_x \in P} e_x.w_k - C \geq 0, \text{where } C \geq 0, C \in R^+ \text{ and } k\text{= 1 to } K_2. \tag{3.2}$$

The MOSP problem can be defined as:

$$Minimize(f_1(P), f_2(P), ..., f_k(P)), \text{s.t.}, \{g_1(P), g_2(P), ..., g_k(P)\} \text{ can be satisfied.} \tag{3.3}$$

The objective functions and constraints can have any other form based on the actual problem. When the number of objectives is greater than one, i.e., $K_1 > 1$ and $K > 1$, then the MOSP problem becomes an NP-hard problem [8, 3] and heuristics should be used to solve it. The multiple objectives can be in contradiction to each other and no single solution is said to be optimum in all objectives. Therefore, a set of pareto-optimal solutions should be obtained for the MOSP problem. A pareto-optimal set contains all solutions that are not dominated by any other solution. A solution A dominates another solution B (which is represented as $A \succ B$), if A is better than B in at-least one objective function value and A is not inferior than B in any objective function value.

3.2 Performance Measurement

EAs are used to approximately solve the multi-objective optimization problems. Therefore, the quality of pareto-optimal set obtained from any EA should be calculated. Hypervolume (HV) metric [25, 26] is a popular method for finding and comparing the quality of pareto-optimal solutions obtained from different algorithms. The HV metric calculates the space covered by the solutions in the pareto-optimal set of any algorithm in the solution space. The HV indicator measures both quality and diversity of solutions. The algorithms that yield higher values of the HV metric are considered good. The choice of performance metric depends on the information available about the problem. Many test problems were designed to do experiments with the multi-objective optimization algorithms. In actual or true pareto-front of the test problems is known. Therefore, many performance metrics exist for the test problems. The real-world problems, on the other hand have unknown true pareto-fronts. Therefore, very few performance metrics exist for real-world problems. HV metric has the advantage that it can works well in problems with unknown true pareto-front.

The HV metric is illustrated in Fig. 3.1 for a minimization problem of two objectives. In Fig. 3.1, the pareto-optimal set has two solution a_1 and a_2 and b is the maximum bounding point. The shaded portion shows the HV value of the pareto-optimal set. The bounding point (b) has coordinates equal to the number of objectives in the multi-objective problem. The value of b can be determined by using the method proposed by Knowles [27]. He selected the bounding point as $b_j = max_j + \delta(max_j - min_j)$, where b_j is the bounding value of the j^{th} coordinate and max_j and min_j are the maximum and minimum values, respectively, of the j^{th} coordinate in the Pareto-optimal solutions. The value of δ can be taken as 0.01. The HV of the Pareto-optimal sets were calculated using the tool proposed by Fonseca, et al. [28]. The tool uses an improved version of the Hypervolume by Slicing Objectives algorithm [29], which accurately computes the HV. This algorithm is among the fastest methods to compute HV.

When two algorithms (A and B) are compared against each other. If HV_A is the HV of algorithm A and HV_B is the HV of the algorithm B. The ratio $\frac{HV_A}{HV_B}$ has value equal to greater than 1 when $HV_A \geq HV_B$. The ratio can relate the HVs of any two algorithms. If A represents any of the proposed algorithm

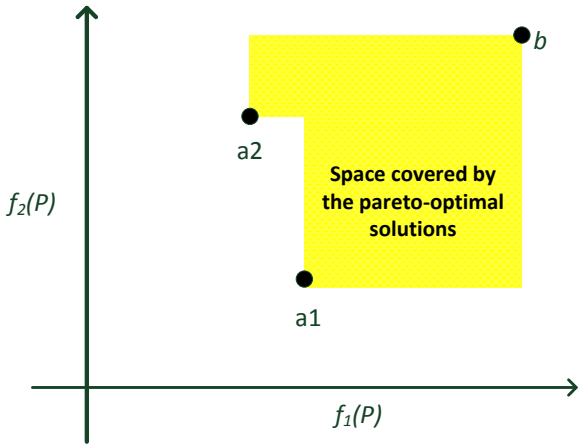


Figure 3.1: Illustration of the HV metric.

Table 3.1: Details of the Road Networks

Road Network	Description	Number of nodes	Number of edges
BAY	Road network of the San Francisco Bay Area	321,270	800,172
COL	Road network of the Colorado state	435,666	1,057,066
NY	Road network of the New York City	264,346	733,846

Table 3.2: Variables representing the HV of the algorithms

Symbols	Description
HV_{StocE}	HV of the Proposed StocE-based algorithm
HV_{GA}	HV of the Proposed Off-Spring Non-Storing GA algorithm
$HV_{NSGA-II}$	HV of the existing NSGA-II algorithm
$HV_{Micro-GA}$	HV of the existing Micro-GA algorithm
HV_{PAES}	HV of the existing PAES algorithm
HV_{MOSA}	HV of the existing MOSA algorithm
$HV_{Std-StocE}$	HV of the straight-forward StocE algorithm
$HV(NSGA-II)$	Ratio of the HV of the Proposed algorithm to that of the NSGA-II
$HV(Micro-GA)$	Ratio of the HV of the Proposed algorithm to that of the Micro-GA
$HV(PAES)$	Ratio of the HV of the Proposed algorithm to that of the PAES
$HV(MOSA)$	Ratio of the HV of the Proposed algorithm to that of the MOSA
$HV(Std-StocE)$	Ratio of the HV of the Proposed algorithm to that of the std-StocE

and B represents any of the existing algorithm. Then, the ratio $\frac{HV_A}{HV_B}$ gives the following information: (i) If $\frac{HV_A}{HV_B} \geq 1$, then the proposed algorithm is equal to or better than the existing algorithm, (ii) If $\frac{HV_A}{HV_B} = x < 1$, then the HV of the proposed algorithm is about $x\%$ of the existing algorithm. When x value is closer to 1 like 0.9, 0.8 or 0.7 then the performance of the two algorithms is considered to be very close.

The MOSP problem is solved on the road networks of some cities and states of the United States of America. The DIMACS design challenge website [30] contains huge size road networks of some cities and states of United States of America. The selected road networks are shown in Table 3.1. Table 3.1 shows the symbol, description of the road network and the number of nodes and edges that it contains. All the road networks are of huge sizes. An experiment consists

of randomly selecting the source and destination nodes in the road network and finding MOSPs between them. The EAs are non-deterministic algorithms and they yield a different solution every time. Therefore, a test-case consists of up-to five trials of an experiment. The average HV of the five trials is the HV of that test-case. This works consists of two proposed algorithms and five existing algorithms. The symbols used to represent the HV of a test-case (i.e., average HV of its five trails) of different algorithms is shown in Table 3.2. In the ratios of the HVs, the numerator contains the HV of the proposed algorithm. The proposed algorithm refers either to the Proposed StocE-based algorithm or Proposed Off-spring Non-storing GA-based algorithm based on the context. in the ratios $HV_{NSGA-II}$, $HV_{Micro-GA}$, HV_{PAES}, HV_{MOSA}, and $HV_{Std-StocE}$, when their value is greater than 1, then the proposed algorithm performs better than the existing algorithm. When their value is 1, then the proposed and the existing algorithm performs equal to each other. When their value is lesser than 1, then the existing algorithm performs better than the proposed algorithm.

Chapter 4

Proposed Algorithms

This section describes two new EAs for solving the MOSP problem. Our first proposed algorithm is based on Stochastic Evolution ($StocE$); and our second algorithm is based on Genetic Algorithm (GA).

4.1 Method to generate a random paths

In the proposed algorithms, a random path needs to be built between any source and destination nodes in the network. Fig. 4.1 shows a method to generate a random path between the nodes s and d. The method can be invoked by calling the function $form\_path(s, d)$. The first four lines initialize the variables. Each node has two attributes π and sel associated with it, π is used to store the node that is preceding it in the path from source to destination and sel is used to indicate that the corresponding node is entered in the queue Q at least once. The $while$ loop between lines 6 and 15, retrieves an element from Q i.e. η then inspects the nodes that are adjacent to η and inserts new elements in Q according to the conditions shown. The second $while$ loop between 17 and 20, forms a complete path between the source and destination node by following the π attribute of the nodes. $Reverse\_Order$ function in line 21, corrects the order to path from source to destination.

4.2 Proposed StocE based Algorithm

StocE is a general purpose iterative stochastic algorithm [16, 17]. It resembles a biological evolutionary process in which the solution eliminates its bad characteristics from generation to generation. The perturb operation in the StocE performs the task of elimination of the bad characteristics from the solution. This paper proposes a StocE algorithm for the MOSP problem. In the proposed algorithm, sub-paths in the solution are considered as its characteristics and the solution replaces the bad sub-paths with better sub-paths from generation to generation. The proposed StocE based algorithm is shown in Fig. 4.2. The inputs are: source and destination nodes (s & d); $iter$, which is the maximum number of iterations in the Perturb-cycle; m_t, which is the maximum iteration in the Mutation-cycle; Num, which is the number of sub-paths (i.e. the number of characteristics) to be considered in the Perturb operation; P_b,

Input: nodes: s, & d
Output: y: A path from s to d nodes.-

1: Q= $\emptyset$, $done$= 0, y =$\emptyset$
2: **for** (each node $n_i \in V$) **do**
3: $n_i.sel$= false, $n_i.\pi$= 0
4: **end for**
5: $Q = Q \cup \{s\}$
6: **while** (!$done$) **do**
7: η= A randomly selected element from Q, $Q = Q - \{\eta\}$
8: **for** (each node $n_x \in Adj(\eta)$) **do**
9: **if** ($n_x == d$) **then**
10: $n_x.\pi = \eta$, $done = 1$
11: **else if** (!$n_x.sel$ and $Q \cap \{n_x\} == \emptyset$) **then**
12: $n_x.\pi = \eta$, $Q = Q \cup \{n_x\}$, $n_x.sel = true$
13: **end if**
14: **end for**
15: **end while**
16: $n = d$, $y = y \cup \{n\}$
17: **while** ($n! = src$) **do**
18: $n = n.\pi$
19: $y = y \cup \{n\}$
20: **end while**
21: $Reverse\_Order(y)$
22: **return** y

Figure 4.1: Method to find a random path: $y = form\_path(s, d)$.

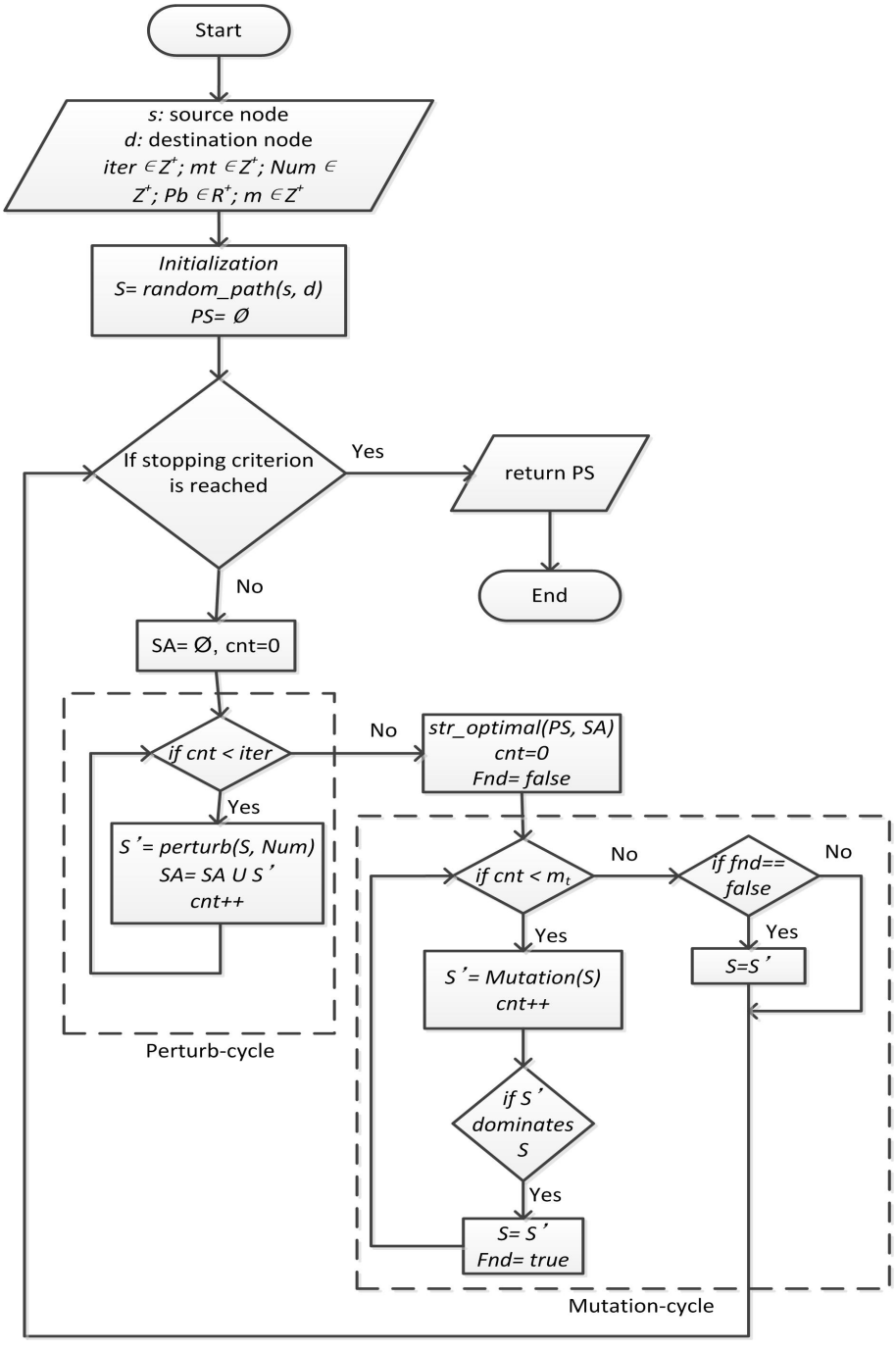


Figure 4.2: Proposed StocE based Algorithm

which is the probability value used in the Perturb operation; and m is the maximum number of solutions in the archive of pareto-optimal solutions. The value of P_b should be kept large, (e.g. 0.7). The first step is initialization in which the solution S is initialized with a random solution or path between the nodes s and d. The random path can be built using the function $form\_path$. A loop which contains two cycles (i.e. Perturb-cycle and Mutation-cycle) starts after the initialization step. The Perturb-cycle creates new solutions through the perturb operation and store the new solutions in the set SA. The set SA is initialized to null at the start of the Perturb-cycle and stores all solutions created through the perturb operation. After the completion of the Perturb-cycle, the function $str\_optimal()$ is called. Using the $str\_optimal()$ function, the solutions in SA that are not dominated by any other solution in both sets PS (where PS is the archive of pareto-optimal solutions) and SA are inserted into PS. The mutation cycle also contains up to m_t number of iterations. The Mutation-cycle contains small number of cycles so as to increase the possibility that the mutant solution dominates (i.e. better than) the existing solution and also the mutation cycle should not consume much execution time. Due to the mutation operation, the next iteration of the outer-most loop executes on a different solution then the previous iteration. The stopping criterion can be the maximum number of iterations or maximum execution time. After stopping criteria is reached, the solutions in PS are returned by the algorithm. The different steps are described in detail in the following sub-sections.

4.2.1 Perturb Operation

In StocE, each characteristic of the solution should prove its suitability to remain in the next generation. The perturb operation implements this feature of the StocE. This work considers different sub-paths in the solution as its characteristics. Therefore, the sub-paths should calculate their suitability to remain in the next generation. Fig. 4.2 shows that the Perturb cycle has $iter$ number of iterations. In each iteration, the Perturb operation is applied to form a new solution S' from S. S' is stored in SA and at the end of the Perturb cycle, up to $iter$ number of solutions are stored in SA.

The proposed perturb operation consists of two parts. The first part consists of finding a sub-path in S which is minimally suitable to remain in the next generation. The suitability of any sub-path (p) is determined based on two values: (i) the values of its objective function values $(f_1(p), f_2(p), ..., f_K(p))$, and (ii) the number of elements in the sub-path. The sub-paths that have higher values of the objective functions and lesser elements are considered to be less suitable for the next generation. The second part consists of finding an alternative sub-path to replace the selected sub-path from S. Alternative sub-path replaces the selected sub-path if it dominates the selected sub-path, or is better than the selected sub-path in any objective function value.

Fig. 4.3 shows the first part of the perturb operation and Fig. 4.4 shows the second part of the perturb operation. Fig. 4.3 shows the $select\_subpath()$ function, the inputs are the current solution S, positive integer (Num), and a positive real number (P_b). The Num is used to specify the number of sub-paths to be considered in the current solution. The for loop between lines 5 and 15 stores the starting and ending nodes of the selected sub-paths in the arrays st and en. NN stores the number of nodes in each sub-path. The for loop

Input: S: current solution, $Num \in Z^+$, $P_b \in R^+, Pb \in \{0 \leq Pb \leq 1\}$
Output: $y = \{y_1, y_2\}$, starting (y_1) and ending (y_2) indices of a sub-path in S

1: st= null, en= null, NN= null, $rank$= null, n= number of elements in S
2: **for** k= 1 to K **do**
3: $F_k = null$
4: **end for**
5: **for** i = 0 to $Num - 1$ **do**
6: $i_1 = 0$, $i_2 = 0$
7: **while** $i_1 \geq i_2$ **do**
8: i_1= random integer between 0 and $n - 1$
9: i_2= random integer between 0 and $n - 1$
10: **end while**
11: **for** k=1 to K **do**
12: $F_k[i] = f_k(S[i_1....i_2])$
13: **end for**
14: $st[i] = i_1$, $en[i] = i_2$, $NN[i] = i_2 - i_1 + 1$
15: **end for**
16: **for** i=0 to $Num - 1$ **do**
17: $r_1 = 0$, $r_2 = 0$, ..., $r_{K+1} = 0$
18: **for** j=0 to $Num - 1$ **do**
19: **for** k=1 to K **do**
20: **if** $i! = j$ and $F_k[i] > F_k[j]$ **then**
21: r_k++
22: **end if**
23: **end for**
24: **if** $i! = j$ and $NN[i] < NN[j]$ **then**
25: $r_{K+1}++$
26: **end if**
27: **end for**
28: $rank[i] = \sum_{i=1}^{K+1} r_k$
29: **end for**
30: I_r: a random integer between 0 and $Num - 1$
31: I_m: index of the maximum element in $rank$
32: RN= a random real number between 0 and 1
33: **if** $RN \leq P_b$ **then**
34: $y_1 = st[I_m]$, $y_2 = en[I_m]$, $y = \{y_1, y_2\}$
35: **else**
36: $y_1 = st[I_r]$, $y_2 = en[I_r]$, $y = \{y_1, y_2\}$
37: **end if**
38: **return** y

Figure 4.3: Function for the selection of a sub-path $y = select\_subpath(S, Num, P_b)$.

Input: S: current solution, $Num \in Z^+$, $P_b \in R^+, Pb \in \{0 \leq Pb \leq 1\}$, $G = (V, E)$
Output: S: solution after the perturb operation
1: $I = \{i_1, i_2\} = select\_subpath(S, Num)$
2: $e_1 = (n_a, n_b) = S[i_1]$, $e_2 = (n_c, n_d) = S[i_2]$
3: $t = form\_path(n_a, n_d)$
4: $S' = Concatenate\{S[0...i_1 - 1], t, S[i_2 + 1....]\}$
5: **return** S'

Figure 4.4: Proposed Perturb operation $S' = Perturb(S, Num, P_b)$.

between lines 16 and 29 finds the ranks of all sub-paths. The sub-paths that are less suitable are assigned higher rank values. The sub-paths that have higher objective function values and lesser number of nodes are assigned higher ranks. In line 30, I_m is the index of the sub-path that has the highest rank value. In line 31, I_r is the index of a randomly selected sub-path. In lines 33 to 37, the starting and ending indices of the sub-path that has maximum rank value is stored in y_1 with probability P_b. With probability $1 - P_b$, the starting and ending indices of a random sub-path is stored in y_1. The P_b is assigned a higher value, therefore, generally the algorithm in Fig. 4.3 returns a sub-path that is least suitable to be kept in the next iteration as compared to the several other randomly selected sub-paths.

Fig. 4.4 shows the second part of the perturb operation. The inputs are the current solution (S), Num and P_b, which are already described. The function returns a new solution which is obtained after applying the perturb operation. In line 1, a sub-path is chosen in S by calling the function $select\_subpath()$. e_1 and e_2 represent the edges that contains the starting and ending nodes of the sub-path. In line 3, a new sub-path t is formed between the starting (n_a) and ending nodes (n_b) of the selected sub-path. In line 4, a new path S' by inserting t between the upper and lower portions of the current solution. In lines 5 and 6, S is updated to S', if S' is not dominated by S or S' is better than S in any objective function value.

4.2.2 Mutation

Input: S: current solution, d destination node
Output: S': solution after the mutation operation
1: n= number of elements in S, $S' = S$
2: r_n= random integer between 0 and $n - 1$
3: $e_{r_n} = (n_a, n_b) = S'[r_n]$
4: $t = form\_path(n_a, d)$
5: $S' = concatenate\{S'[0...r_n - 1], t\}$
6: **return** S'

Figure 4.5: Mutation Operation.

Fig. 4.2 shows the Mutation-cycle that comprises of up to m_t number of iterations. The Mutation-cycle aims to find a new solution through mutation that dominates the existing solution. However, after m_t iterations are completed and

no new solution can dominate the existing solution, then the existing solution is updated to the solution that is created in the last iteration of the Mutation-cycle. The purpose of the mutation operation is to move the current solution to another random location in the solution space. This operations aims to increase the diversity of solutions and escaping from local minima. The mutation operation is shown in Fig. 4.5. It first selects a random edge e_{r_n} in the current solution, then forms a new random path from the starting node (i.e. n_a) of selected edge to the destination node (d). The mutant solution S' is returned at the end.

4.2.3 Store pareto-optimal solutions

Fig. 4.2 shows that the function *str_optimal*() is executed after the completion of the Perturb-cycle. Using the function *str_optimal*(), the solutions in the set SA are stored in the archive of pareto-optimal solutions (i.e., PS) if they are feasible as well as not dominated by any other solution in the sets SA or PS. The solutions in SA are inserted into PS one by one. The solution from SA is compared with the solutions that already exist in SA. If the solution from SA is feasible and not dominated by any solution in PS, then it is inserted into PS. The solutions in PS that are dominated by the solution from SA are removed from PS. The maximum size of the archive of the pareto-optimal solutions (PS) is m solutions. When the archive becomes full, then one of the following two approaches can be used to replace an existing solution with the new solution in the archive of pareto-optimal solutions. The first approach is to randomly select a solution in the archive and replace it with the new solution. The second approach is the adaptive grid method proposed by Knowles, at al. [20]. It is used to decide if the solution should be accepted by replacing another already existing member of the archive. In this method, each solution in the pareto-optimal archive has a grid location. If the new solution lies in a less crowded location in the grid, then it is accepted in place of a solution that lies in a crowded location of the grid. The grid has dimensions equal to the number of objectives, and the range of each dimension is known from the solutions in the archive. The grid location of any solution is determined by recursively bisecting each dimension and finding the half in which the solution lies. The maximum number of sub-divisions in each dimension is set by the user and an example value is 15 or 25.

4.3 Proposed Off-Storing Non-Storing GA

The GA and its different variants (like NSGA-II, Micro-GA, etc) create children chromosomes equal to the size of the population. Therefore, if the population size is N_1, then the GA stores a total of $2 \times N_1$ solutions in the population. In our proposed GA-based algorithm, a child chromosome after its creation through the crossover or mutation operation is not stored at a new memory location but conditionally stored at the place of its parent chromosome. The child chromosome replaces its parent chromosome under the following conditions: (i) If the child chromosome dominates the parent chromosome, (ii) If the parent chromosome is not pareto-optimal solution in the existing population then the child chromosome always replaces its parent chromosome. Therefore,

additional amount of memory is not required to store the children chromosomes. The algorithm consists of two GA operations: crossover and mutation. In each iteration, all chromosomes undergo any one GA operation. Dominated chromosomes prefer the crossover operation and non-dominated chromosomes prefer the mutation operation. The second parent in the crossover operation is always a non-dominated chromosome that has some common genes with the first parent. The first parent in the crossover operation is considered as the actual parent chromosome.

The proposed GA-based algorithm is shown in Fig. 4.6. The inputs to the algorithm are: Population size (N), source and destination nodes (s and d), a probability (P_b) which is used in selecting a GA operator for the chromosome. The first step is initialization. After initialization, the outer-most loop performs the optimization until the stopping criterion which can be the maximum number of iterations or maximum execution time is reached. In the optimization, the first step is to mark the chromosomes in the population that are pareto-optimal and feasible in the current population. In that way, the pareto-optimal solutions can be preserved in the GA operations. The next step consists of a loop that has N iterations. In any iteration, P_i represents the chromosome that exists at the i^{th} position in the Population. First, a GA operator is selected for the chromosome P_i and then the selected GA operator is applied to P_i. The new chromosome that is created after applying the GA operator is called c'. If P_i is a non-dominated chromosome and c' dominates P_i then the existing value of P_i is replaced by c'. If P_i is a not a non-dominated chromosome then c' replaces the existing value of P_i. After the stopping criterion is reached, then the pareto-optimal and feasible solutions in the Population are returned by the algorithm. The remaining part of this section describes the main operations in the proposed algorithm.

4.3.1 Initialization

In this step, the Population is initialized with up to N distinct and random chromosomes. Each chromosome is a complete solution or path between the nodes s and d. The random paths are generated using the function $form\_path()$.

4.3.2 Mark Pareto-Optimal Solutions

The chromosomes are associated with an attribute *sel* which is initially set to the false value. When its value is true then the chromosome is called as *selected*. In this step, all chromosomes in the Population that are feasible as well as not dominated by any other solution in the Population are selected and the values of their *sel* attributes are set to true.

4.3.3 Selection of the GA Operation

The proposed algorithm contains a set of GA operations, which is represented as GA_{set}. The set consists of two elements, i.e., $GA_{set} = \{Crossover, Mutation\}$. All chromosomes must go through any one of the operations from the set GA_{set}. The crossover operation requires that a common node should exist between the two parents [14]. In a conventional GA, parents are selected by methods such as roulette-wheel or tournament selection. This work proposes that the crossover

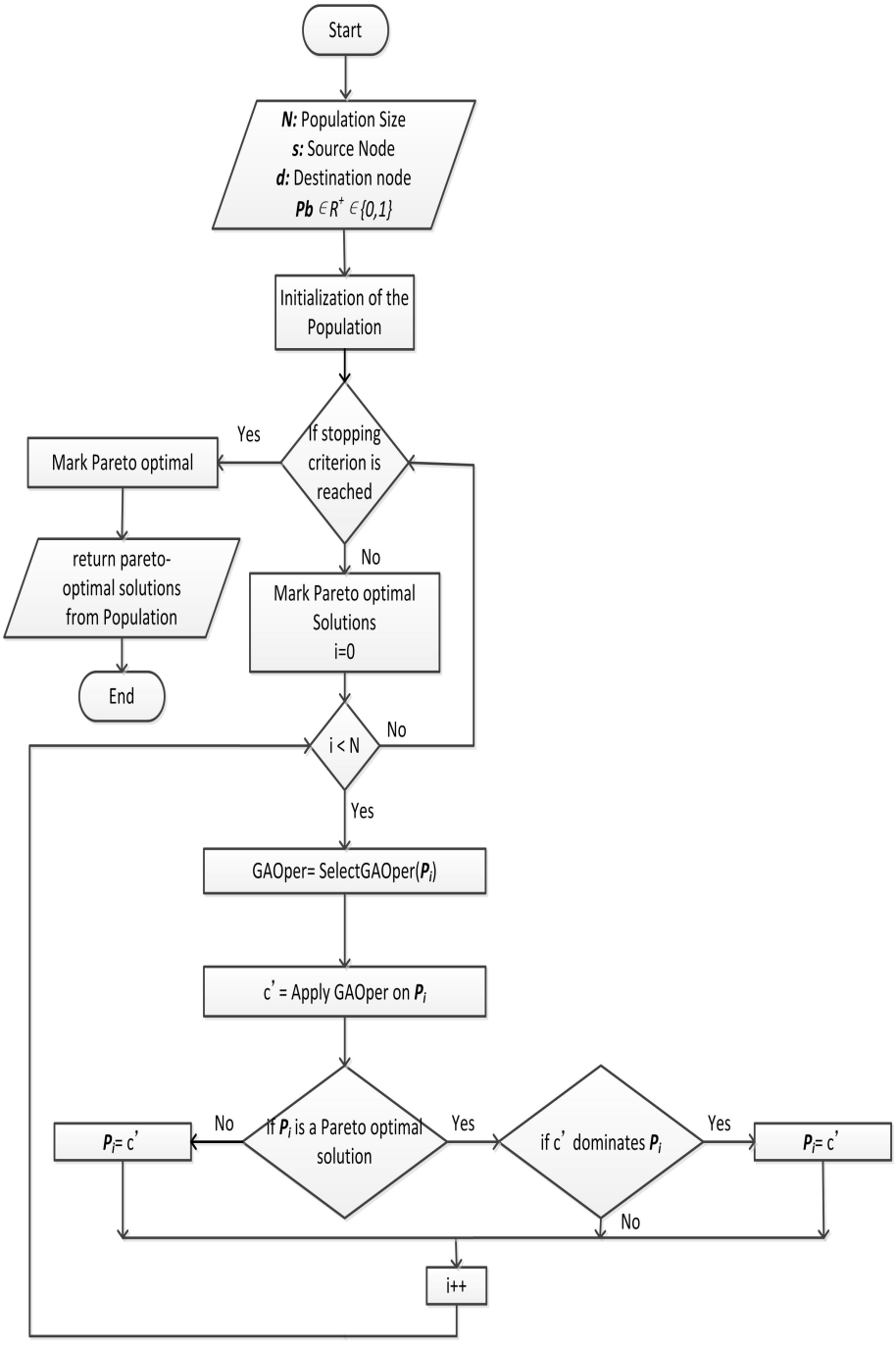


Figure 4.6: Proposed GA based Algorithm

Input: $P_i \in Population$, s: source node, $Population$
Output: y=1, if P_i has a feasible pair in $Population$. .

```
1: cnt=0
2: for each chromosome P_j ∈ Population and P_i ≠ P_j do
3:    for each edge e_x = (n_a, n_b) ∈ P_j do
4:       for each edge e_y = (n_u, n_v) ∈ P_i do
5:          if n_a == n_u ≠ s then
6:             cnt++
7:             Exit from the nested for loops
8:          end if
9:       end for
10:   end for
11: end for
12: if cnt > 0 then
13:   return 1
14: else
15:   return 0
16: end if
```

Figure 4.7: Method used to find if P_i has a feasible pair in the population $y = CheckCn(P_i, Population, s)$.

operation can be applied to any chromosome P_i such that the second parent is any feasible pair for P_i. A feasible pair for P_i is a non-dominated chromosome that has at least one common node with P_i (excluding source and destination nodes). The procedure to check if any feasible pair exists for a chromosome P_i is shown in Fig. 4.7 and is represented as $CheckCn()$. The function $CheckCn(P_i)$ returns 1 if at least one pair exists for P_i, otherwise it returns 0.

The method to select a GA operation for the chromosome is shown in Fig. 4.8. The basic idea behind the proposed assignment of GA operations is the following. The chromosomes are distinguished into two classes: non-dominated and dominated. The crossover operation is selected for the dominated chromosomes with a high probability (P_b). As previously mentioned, the second parent in the crossover operation should be a non-dominated chromosome. Therefore, the dominated chromosomes have a high probability of producing a better offspring by exchanging genes with a non-dominated chromosome. On the other hand, a mutation operation is selected for the non-dominated chromosomes with a higher probability (P_b) so that they try to produce a better quality offspring by introducing new genes. The inputs to the method in Fig. 4.8 are as follows: chromosomes P_j and P_b which is a real number between 0.5 and 1. At the end of the method, a GA operation is selected for the input chromosome P_j.

4.3.4 GA Operations

GA_{set} in the proposed algorithm consists of two operations: crossover and mutation. Therefore, $GA_{set} = \{crossover, mutation\}$ and new operations can be added into the set GA_{set} at any time. The function $SelectOperation()$ should be modified if new operations are added into GA_{set}. The next two subsections describe the operations which currently exists in GA_{set}.

Input: $P_j \in Population$, $P_b \in \{x \in \mathbb{R} | 0.5 < x \leq 1\}$
Output: $OPER$: A GA operation for P_j

v= $CheckCn(P_j)$
r: random real number between $[0, 1]$
if $v == 0$ **then**
  return Mutation
else if $P_j.sel == true$ and $r \leq P_b$ **then**
  return Mutation
else if $P_j.sel == true$ **then**
  return Crossover
else if $r \leq P_b$ **then**
  return Crossover
else
  return Mutation
end if

Figure 4.8: Method used to select a GA operation for the chromosome P_j, $OPER = SelectOperation(P_j)$.

Input: P_j, $Population$, s: source node
Output: Index of the feasible pair

for each particle $P_k \in Population$ **do**
  if $P_k.marked == true$ and $P_j \neq P_k$ **then**
    for each edge $e_x = (n_a, n_b) \in P_j$ **do**
      for each edge $e_y = (n_u, n_v) \in P_k$ **do**
        if $n_a == n_u \neq s$ **then**
          I= index of P_k in the $Population$
          $P_c = P_c \cup I$
          exit to the outermost for loop
        end if
      end for
    end for
  end if
end for
$selP$= randomly select an element from P_c
return $selP$

Figure 4.9: Procedure to select a feasible pair for P_j, i.e., $y = findpair(P_j, Population, s)$

Input: P_j, $Population$.
Output: c': offspring
1: $c' = Population[findgbest(P_j)]$
2: $C_n = null$
3: **for** each edge $e_x = (n_a, n_b) \in P_j$ **do**
4: **for** each edge $e_y = (n_u, n_v) \in c'$ **do**
5: **if** $(n_a == n_u)$ **then**
6: $C_n = C_n \cup n_a$
7: **end if**
8: **end for**
9: **end for**
10: r= a randomly selected node from C_n
11: $c' = concatenate(c'(s..., r), P_j(r, ...d))$
12: **return** (c')

Figure 4.10: Procedure to apply the crossover operation to particle P_j, i.e., $c' = Crossover(P_j, Population)$

Input: $P_j \in Population$, d: destination node
Output: c': offspring
1: $e_r = (n_a, n_b)$: a randomly selected edge in P_j
2: c'= $form\_path(n_a, d)$
3: c'= $concatenate(P_j(e_0, ..., e_x), c')$ (s.t. $e_x = (n_x, n_a)$)
4: **return** (c')

Figure 4.11: Mutation operation, i.e., $c' = mutation(P_j)$

Table 4.1: Memory requirements of the algorithms

Component	Number of Paths in different Algorithms					
	Prop. StocE	Prop. GA	NSGA-II	Micro-GA	(1-1)-PAES	MOSA
Population	1	N	N	N	1	1
Children	1	1	N	N	1	1
Pareto-optimal set	m	0	0	0	m	m
others	$iter$	0	0	m_1	0	0
total paths	$2 + iter + m$	$N + 1$	$2N$	$2N + m_1$	$2 + m$	$2 + m$

4.3.5 Crossover Operation

The crossover operation in this algorithm is different from conventional single-point crossover. Before the crossover operation can be applied on the chromosome $P_j \in Population$, the second parent should be determined by calling the function $findpair(P_j)$, which is shown in Fig. 4.9. The procedure finds a feasible pair for P_j. As shown in Fig. 4.9, the indexes of all feasible pairs for P_j are stored in P_c. Then, an element is randomly selected from P_c and is returned. The crossover operation which is quite similar to the one proposed by Ahn, et al. [14] is shown in Fig. 4.10. The crossover operation stores the second parent in variable c'. The common nodes between C and P_j are stored in C_n. Then, an element is randomly selected from C_n. In the second to last row, c' is formed by combining the upper portion of c' with the lower portion of P_j. Finally, c' is returned

4.3.6 Mutation Operation

The proposed mutation operation is shown in Fig. 4.11. The inputs are: a chromosome P_j and destination node d. An edge is randomly selected in P_j that is represented as $e_r = (n_a, n_b)$, where n_a and n_b are its starting and ending nodes. Then a random sub-path is formed between the nodes n_a and d. In the next step, c' is updated to store the newly created path between the nodes s and d.

4.4 Estimation of the Memory Requirements

This section presents an estimation of the memory requirements of the proposed algorithms in terms of maximum number of solutions which they store in the memory at any time during their execution. For comparison purposes, the memory required by some famous multi-objective optimization algorithms including NSGA-II, Micro-GA, (1-1)-PAES and MOSA are also shown.

The memory requirements in terms of the number of paths that should be stored in the memory by the algorithms are shown in Table 4.1. The first column mentions the component of the algorithm that store the specified number of paths; the second column mentions the number of paths for different algorithms. The "Prop." acts as short form of "Proposed". The number of paths are calculated based on the input parameters of the algorithms. In that way, different input parameters have different memory requirements. The description of values in the first column are as follows: (i) "Population" refers to the population of solutions in the algorithms; (ii) "Children" refers to the number of chromosomes which are created through the GA operators (crossover and/or

Table 4.2: Memory requirements based on typical values

Algorithm	Parameter values	Number of Paths
Proposed StocE	$m = 10, iter = 3$	15
Proposed GA	$N = 20$	21
NSGA-II	$N = 20$	40
Micro-GA	$N = 10, m_1 = 20$	40
(1-1)-PAES	$m = 10$	12
MOSA	$m = 10$	12

mutation); (iii) "Pareto-optimal set" mentions the size of the pareto-optimal set; (iv) "others" include the number of paths to be stored in the memory by the remaining components of the algorithm. In the proposed StocE based algorithm, "others" include number of paths to be stored in the Perturb cycle. In the Micro-GA algorithm, "others" include the paths that needs to be stored in the memory by the replaceable and non-replaceable memories and their value is represented as m_1. .

4.5 Comparison of Memory Requirements with Some Typical Values

This section shows the comparison of the memory requirements of the proposed algorithms with the existing algorithms. The authors of NSGA-II used a population size of 20 [4] in their experiments. Therefore, the typical values of parameters have a population size of 20. The experiments in the next three chapters use the same parameter values to solve the vehicles' navigation problem. The typical values are selected such that the population size in all population-based algorithms is same. The NSGA-II and Micro-GA require additional memory to store the children chromosomes, therefore, their overall memory requirement is higher than the proposed algorithms.

In Table 4.2, the first column mentions the algorithm. The second column mentions parameter values that are relevant with respect to determining the memory requirements. The last column mentions the memory requirements of the algorithms in terms of the number of paths. The results show that the proposed algorithms are memory-efficient than the existing population based multi-objective algorithms. The solution quality of the algorithms will be shown in the next two chapters. The next chapters show that based on typical values The memory requirement of an algorithm should be combined by its solutions quality The solution quality will be analysed in the next two chapters.

4.6 Summary

This chapter presents two new EAs to solve the MOSP problem. The first algorithm is based on StocE algorithm and treats the random sub-paths in a solution as its characteristics. During optimization it replaces the bad sub-paths with good sub-paths. The second algorithm is an Off-Spring Non-Storing GA. The GA generally creates children chromosomes equal to the population size. The proposed GA is different from the conventional GA because it minimizes the memory requirements of the algorithm by not storing children chromosomes. The children chromosomes either replace their parent chromosomes or being dis-

carded soon after their creation. An estimation and comparison of the memory requirements of the proposed algorithms with the existing algorithms shows that the proposed algorithm uses lesser memory than the famous population-based algorithms.

Chapter 5

Application of the Proposed Algorithms to the Vehicle Navigation Problem of Conventional Vehicles

5.1 Introduction

This chapter shows the application of the proposed algorithms to solve the navigation problem of internal combustion engine based vehicles (ICEVs). The navigation problem in ICEVs aims to find one or more point-to-point (P2P) MOSPs in a road network. The objectives in the MOSP problem are generally minimizing the length of the paths and the travelling time on the paths. The length of a path is equal to the sum of the lengths of the individual edges that are included in it and is generally expressed in km. The travelling time on a path is based on many factors such as the traffic, condition of the roads, road width, maximum allowed speed of the vehicles, etc. The travelling time is generally not related with just length of the path. Therefore, length of the path and their travelling time can be in contradiction to each other. Therefore, a set of pareto-optimal solution should be determined for the MOSP problem for ICEVs.

Let us consider a road network $G = (V, E)$, where V contains all intersections in the road network and E contains all roads that join the intersections in the road network. Any edge $e_i \in E$ has a starting node u and an ending node v and is associated with up to two attributes $e_i.l$ and $e_i.S_A$. $e_i.l$ represents the length of the edge and $e_i.S_A$ represents the average travelling time on the edge for the ICEVs vehicles. The driver selects two distinct nodes s and d in the road network and wants to find one or more paths from s to d. If P represents a path between nodes s and d, then $P \subseteq E$. The first and last elements in P can be represented as e_s and e_d such that the starting node of e_s is s and the ending node of e_d is d.

The MOSP problem consists of two objectives which are: $f_1(P)$ and $f_2(P)$. $f_1(P)$ represents the total distance of the path and $f_2(P)$ represents the total

travelling time of the path. The values of $f_1(P)$ and $f_2(P)$ can be determined as follows.

$$f_1(P) = \sum_{e_i \in P} e_i.l \tag{5.1}$$

$$f_2(P) = \sum_{e_i \in P} e_i.S_A \tag{5.2}$$

The objective function of the MOSP problem can be defined as follows:

$$Minimize(f_1(P), f_2(P)) \tag{5.3}$$

The pareto-optimal set of solutions that is obtained from the optimization algorithms can be represented as S and contains one or more paths (or solutions).

As already mentioned in the chapter 3, the experiments are conducted using the road networks of some cities and states of the USA. The road networks that are shown in Table 3.1 contain the following information: length and travelling time of the edges (i.e., $e_i.l$ and $e_i.S_A$), and geographical location of the nodes.

5.2 Implementation of the existing algorithms

The proposed algorithms were compared with some existing Evolutionary or Stochastic Algorithms in order to show their performances. We selected NSGA-II, Mirco GA, (1-1)-PAES and MOSA algorithms for the comparison because they showed very good performance on many different problems and somewhat can act as benchmark algorithms. The algorithms were implemented using C# in Visual Studio 2010 and executed on an Intel iCore 5, 2.27 MHz based laptop computer that has 3 GB of RAM. The details of the implementations of the existing algorithms are shown in the following:

The stopping criterion in all algorithms was set to 30 seconds. The NSGA-II was implemented with population size of 20 solutions. The authors of NSGA-II experimented their algorithm with a population size of 20 [4], therefore, this work also uses the same population size. It used tournament selection based on crowding distance to select the parents for the crossover operation. The crossover and mutation operations were used for the shortest path problem, as proposed by Ahn, et al. [14]. Crossover probability was set to 0.90 and mutation probability was set to 0.15. During each iteration, the elements for the population in the next iteration were selected from the population and children sets. The selection was made based on the non-domination count and diversity of solutions.

The Micro-GA was implemented with population memory size of 20 solutions in which 50% is non-replaceable memory and the remaining is replaceable memory. The size of the population for the micro-GA cycle is 10. The nominal convergence of the micro-GA cycle is reached in 5 iterations. In the micro-GA cycle, the parents for the crossover operation are selected using tournament selection based on non-domination count of the solutions. The crossover rate and mutation probability were set to 0.90 and 0.15. The crossover and mutation operations were used for the shortest path problem, as proposed by Ahn, et al. [14]. In the micro-GA cycle, up to one non-dominated solution is retained from

one generation to the next. When nominal convergence of the micro-GA cycle is reached then up to two non-dominated solutions are copied to the external memory and replaceable portion of the population memory.

The MOSA was implemented with parameter values as follows, initial temperature (T_o) was set to 100 and its value is decreased by 10% in each iteration. The threshold for minimum solutions in the pareto optimal solutions archive was set to 4, i.e., if the archive of pareto optimal solutions contains less than four solutions then interpolation is applied to find hypothetical solutions that are dominated by at least one solution in the archive of pareto optimal solutions. The perturb operation in MOSA is similar to the GA mutation operation [14].

The (1-1)-PAES was also implemented and contains mutation operation as proposed by Ahn, et al. [14]. The size of the archive of pareto-optimal solutions was set to 10. The value of parameter l, i.e., the number of bisections of each dimension in the adaptive grid was set to 100.

A straight-forward implementation of StocE was also performed. The StocE is not previously applied to the MOSP problem, however, a straight-forward implementation is performed to show the advantage of using the proposed algorithm. In the perturb operation, each edge in the solution is considered as a movable element [17]. The perturb operation consists of the following steps: (i) A loop is executed that selects a different edge from the current solution in each of its iterations, (ii) A new sub-path is built using the function $form\_path$ from the starting node of the selected edge to the destination node. The new sub-path is integrated in the original solution and a new solution is formed. With a probability of 0.80, the new solution replaces the original solution only if it dominates the original solution. With a probability of 0.2, the new solution replaces the original solution without any condition. As soon as the original solution is updated the outer for loop is terminated and the perturb operation finishes.

5.3 Proposed StocE based Algorithm

The proposed StocE based algorithm was implemented with typical parameter values, such that sufficient number of iterations can be executed by the algorithm. The fine tuning or analysis of the parameter values is left for future study. The proposed StocE based algorithm was implemented with the following parameter values: (i)$iter = 3$, (ii) $m_t = 3$, (iii) $m = 10$, (iv) number of bisections in the adaptive grid = 100, and (v) stopping criterion was 30 seconds, which is equal to the other algorithms.

The performance of the proposed StocE based algorithm is compared with the existing population and single solution-based algorithms. The existing algorithms include: (i) NSGA-II, (ii) Micro-GA, (iii) (1-1)-PAES, (iv) MOSA, and (v) Std-StocE. The ratios of the HVs between the proposed StocE-based algorithm to the existing algorithms are determined. The results are shown in Figs. 5.1 to 5.3. Fig. 5.1 shows the results of the BAY road network. Fig. 5.2 shows the results of the COL road network. Fig. 5.3 shows the results of the NY road network. In Fig. 5.1, the Proposed StocE-based algorithm obtains average HV values that are 55% to 85% than that of the NSGA-II. It also obtains HV values better than Micro-GA in up-to 20% test cases. It obtains average HV values better than (1-1)-PAES and Std-StocE in 90% test cases. It obtains av-

erage HV values better than MOSA in all test cases. In Fig. 5.2, the Proposed StocE-based algorithm obtains average HV values that are better than that of the NSGA-II in 40% test cases. It also obtains average HV values which are better than Micro-GA in 50% test cases. Its average HV value remains better than that of (1-1)-PAES, MOSA and Std-StocE in all test cases. In Fig. 5.3, the Proposed StocE-based algorithm obtains average HV values that are better than that of the NSGA-II and Micro-GA in 20% test cases. It obtains an average HV value which is better than (1-1)-PAES in 70% test cases. It obtains an average HV which is better than that of the MOSA and Std-StocE in all test cases.

The results are summarized in Fig. 5.4. The graph breaks the HV ratio values into four classes. The interval $\{x|x > 1\}$ includes test cases in which the proposed StocE-based Algorithm obtains an average HV which is better than the other algorithm. The interval $\{x|x = 1\}$ includes test cases in which the proposed and the existing algorithm obtains equal average HV value. The third interval $\{x|0.8 \geq x < 1\}$ includes the test cases in which the existing algorithm obtains a better average HV value than the proposed algorithm but the average HV of the proposed algorithm is not lesser than 80% of the average HV value of the existing algorithm. The last interval $\{x|0.5 \geq x < 0.8\}$ includes the test cases in which the proposed algorithm obtains an average HV value which is equal to or more than 50% but lesser than 80% of the average HV of the existing algorithm. The graph in Fig. 5.4 shows that the Proposed StocE-based Algorithm dominates the existing single-solution based algorithms while using just 25% more memory than them. It also occasionally performs better than the existing population based algorithms while using memory which is about 2.65 times lesser than them. The Proposed StocE-based algorithm also performs better than NSGA-II and Micro-GA in 17% and 30% test cases, respectively. In that way, the Proposed StocE-based Algorithm is successful in achieving its goals by outperforming existing single-solution based algorithms and remains memory efficient.

5.4 Proposed Off-Spring Non-Storing GA

The proposed off-spring non-storing GA was implemented with the following parameter values:(i) $N = 20$, (ii) $M_b = 0.75$, and (iii) stopping criterion was set to 30 seconds. The proposed algorithm was compared with NSGA-II, Micro-GA, (1-1)-PAES and MOSA. The implementation of the existing algorithms is described in Section 5.2. The proposed GA based algorithm requires memory which is almost half of the NSGA-II and Micro-GA.

The performance of the proposed GA based algorithm is compared with the existing population and single solution-based algorithms. The existing algorithms include: (i) NSGA-II, (ii) Micro-GA, (iii) (1-1)-PAES, (iv) MOSA, and (v) Std-StocE. The ratios of the HVs between the proposed GA algorithm to the existing algorithms are determined. The results are shown in Figs. 5.5 to 5.7. Fig. 5.5 shows the results of the BAY road network. Fig. 5.6 shows the results of the COL road network. Fig. 5.7 shows the results of the NY road network. In Fig. 5.5, the Proposed GA-based algorithm obtains an average HV which is better then NSGA-II in 30% test cases, better than Micro-GA in 20% test cases, better than (1-1)-PAES in 90% test cases, and better than MOSA

in 80% test cases. In Fig. 5.6, the Proposed GA-based algorithm obtains an average HV which is better then NSGA-II in 30% test cases, better than Micro-GA in 50% test cases, better than (1-1)-PAES in 60% test cases, and better than MOSA in 70% test cases. In Fig. 5.7, the Proposed GA-based algorithm obtains an average HV which is better then NSGA-II in 80% test cases, better than Micro-GA in 50% test cases, better than (1-1)-PAES in 90% test cases and better than MOSA in all test cases.

The results are summarized in Fig. 5.8. The graph breaks the HV ratio values into four classes as also performed for the Proposed StocE-based algorithm. The graph in Fig. 5.8 shows that the Proposed StocE-based Algorithm dominates the existing single-solution based algorithms while using just 75% more memory than them. It also performs better than the NSGA-II and Micro-GA in 47% and 40% test cases, respectively. It uses memory which is about half of the NSGA-II and Micro-GA. Its performance is also slightly better than the Proposed StocE-based algorithm, however, the Proposed StocE-based algorithm requires lesser memory. Based on the obtained results, the proposed-GA based algorithm successfully outperformed the existing single-solution based algorithms and also uses lesser memory than the Population-based algorithms. Therefore, it is suitable to be utilized for solving MOSP problem in embedded systems.

5.5 Summary

This chapter showed the application of the proposed algorithms to solve the MOSP problem of ICEVs. The MOSP problem of ICEVs has two objectives that are: (i) Minimize the total distance of the path, and (ii) Minimize the total travelling time of the path. The algorithms were implemented using C# in Visual Studio 2010. The experiments were conducted on real road networks of some cities and states in the USA. The performances of algorithms are measured in terms of HV metric. The algorithms having larger HV values are considered better. The HV values of the proposed and the existing algorithm are compared by finding the ratio between the HV of the proposed algorithm to the HV of the existing algorithm. Each experiment consists of up to six trials and the comparisons were performed on the average HV values of the six trials.

The experiments were performed under the condition in which the proposed algorithms can use memory between the single-solution and population-based algorithms. The Proposed StocE-based algorithm outperformed several existing single-solution based algorithms ((1-1)-PAES, MOSA, and Std-StocE) while using only 25% more memory. The Proposed StocE-based algorithm also performs better than the population-based algorithms (NSGA-II and Micro-GA) in around 17% test cases. The Proposed Off-Spring Non-Storing GA also outperforms several existing single-solution-based algorithms ((1-1)-PAES and MOSA) and performs better than the population-based algorithms (NSGA-II and Micro-GA) in 40% test cases.

The experimental results show that the Proposed StocE-based and Off-Spring Non-Storing GA algorithms should be preferred to be solve MOSP problem in embedded systems than the single-solution-based algorithms. They have solution quality better than the single-solution-based algorithms and memory requirement which is lesser than the population-based algorithms.

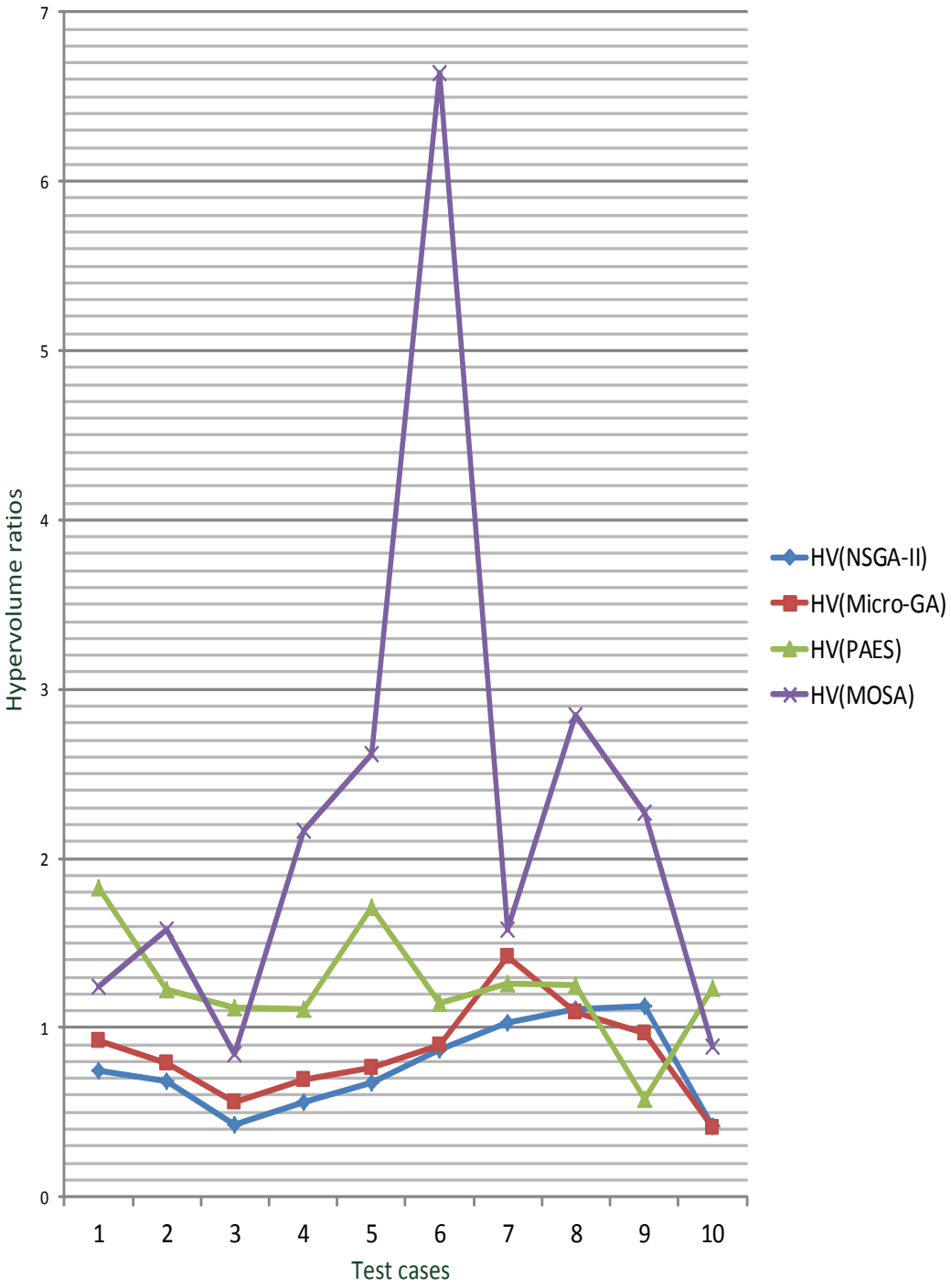


Figure 5.1: Results of the HV ratios for the proposed StocE-based algorithm on the BAY road network.

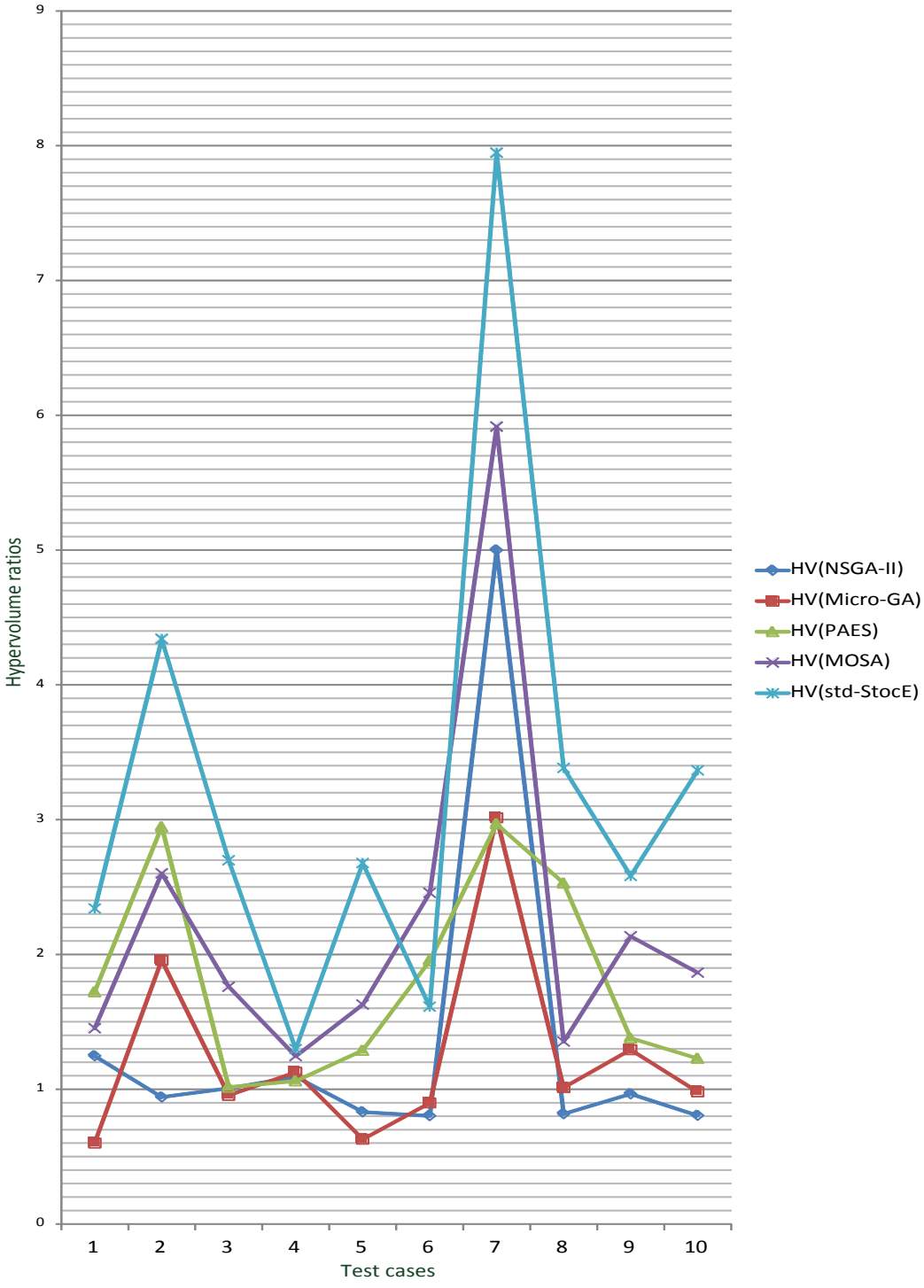


Figure 5.2: Results of the HV ratios for the proposed StocE-based algorithm on the COL road network.

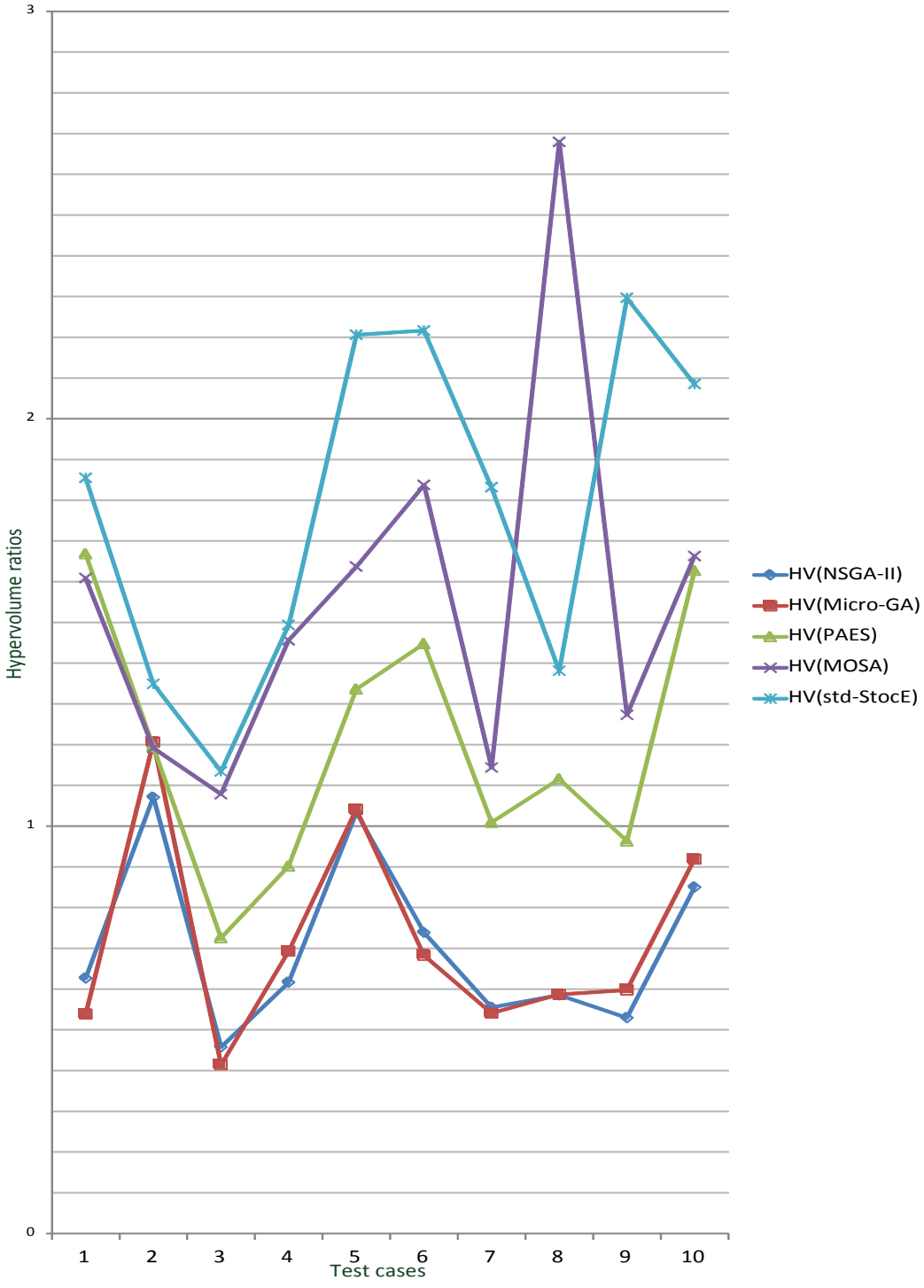


Figure 5.3: Results of the HV ratios for the proposed StocE-based algorithm on the NY road network.

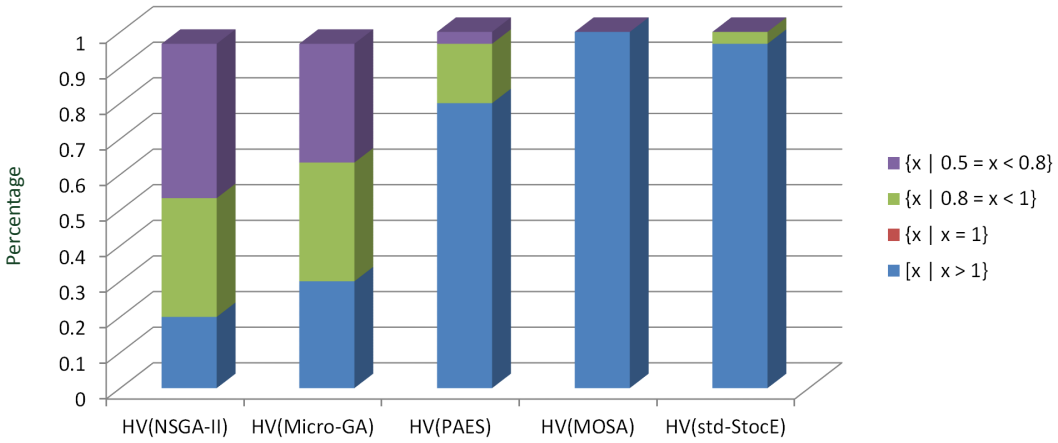


Figure 5.4: Summary of the HV ratio results of the Proposed StocE-based Algorithm.

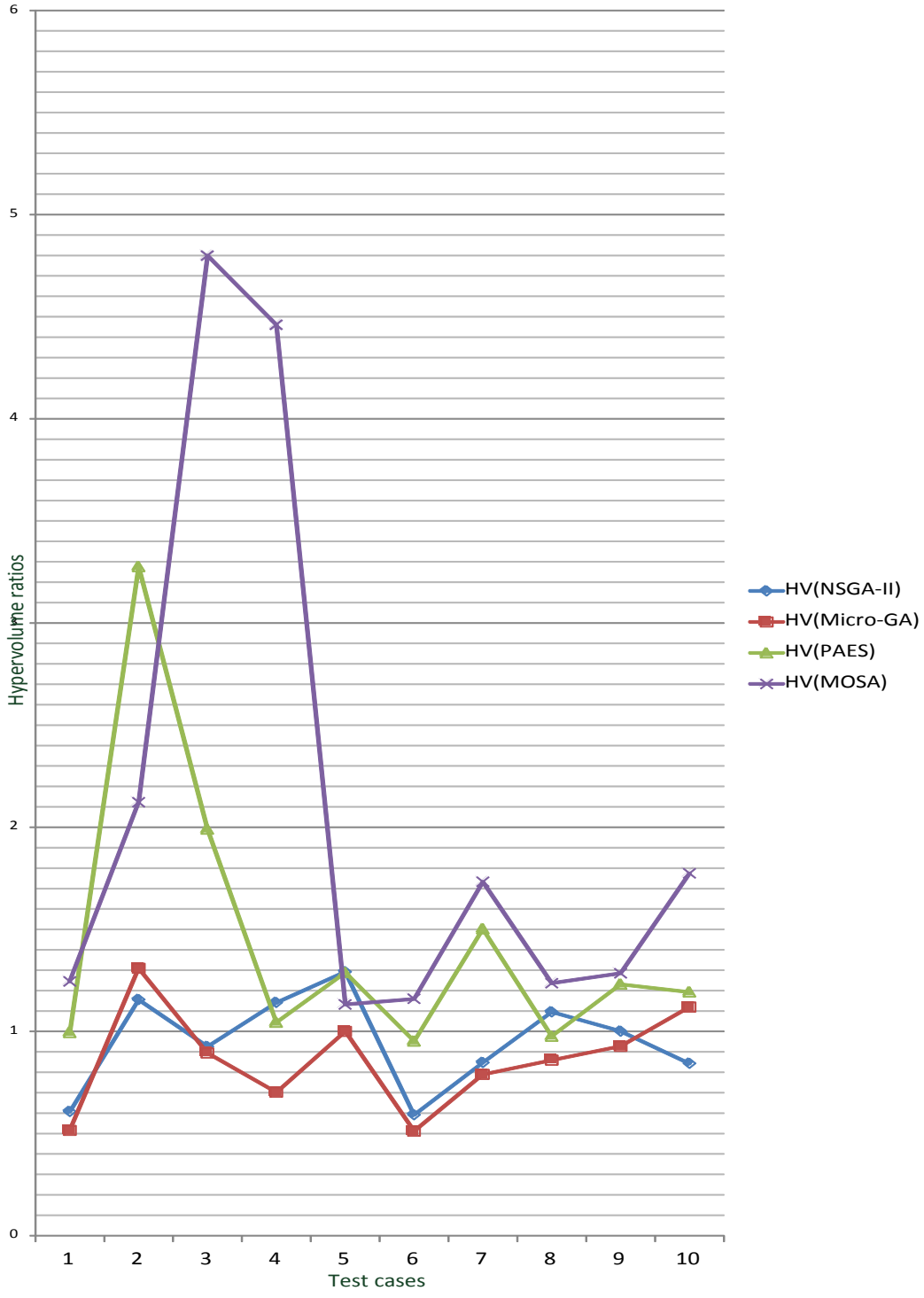


Figure 5.5: Results of the HV ratios for the proposed Off-Spring Non-Storing GA algorithm on the BAY road network.

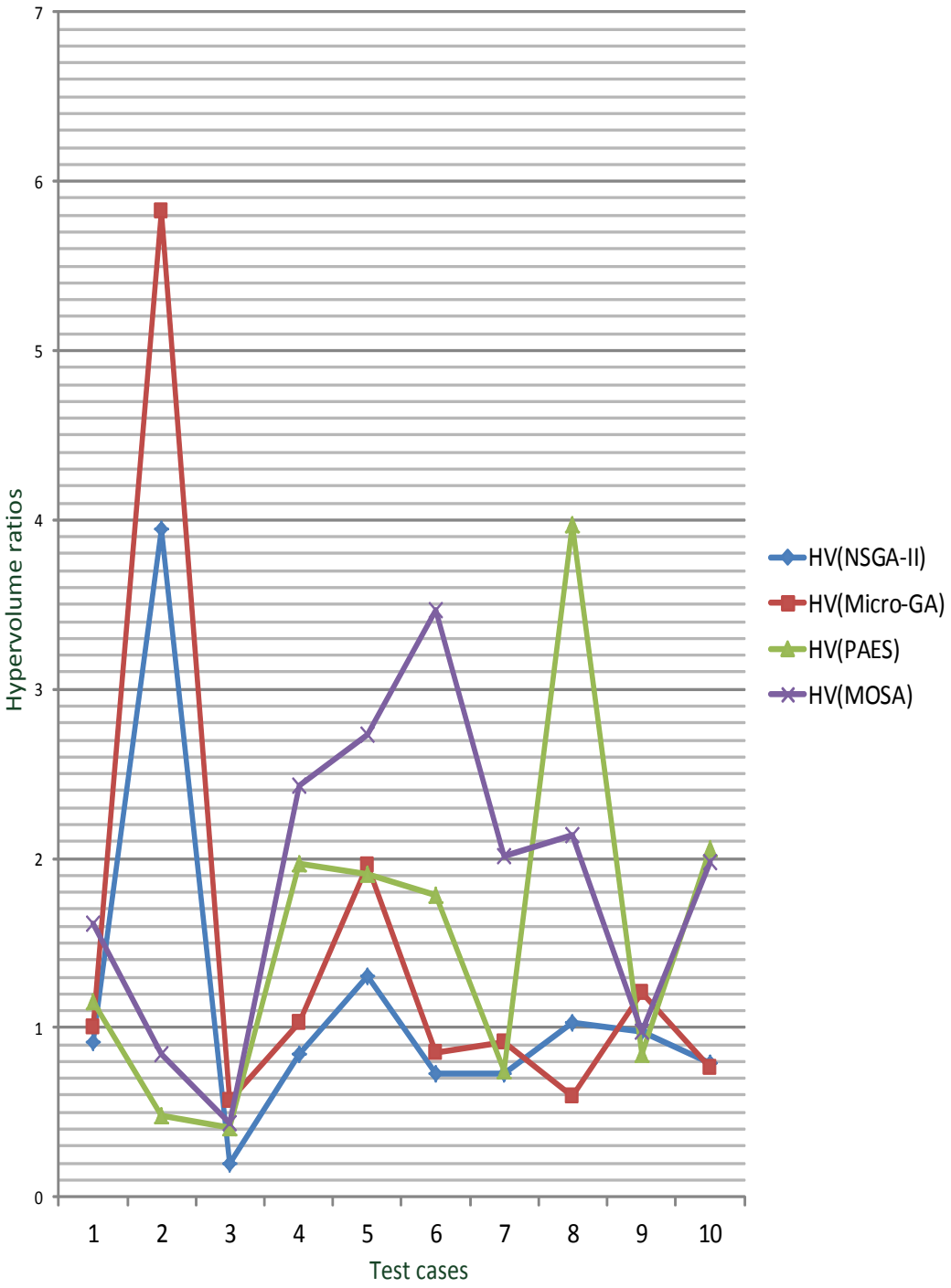


Figure 5.6: Results of the HV ratios for the proposed Off-Spring Non-Storing GA algorithm on the COL road network.

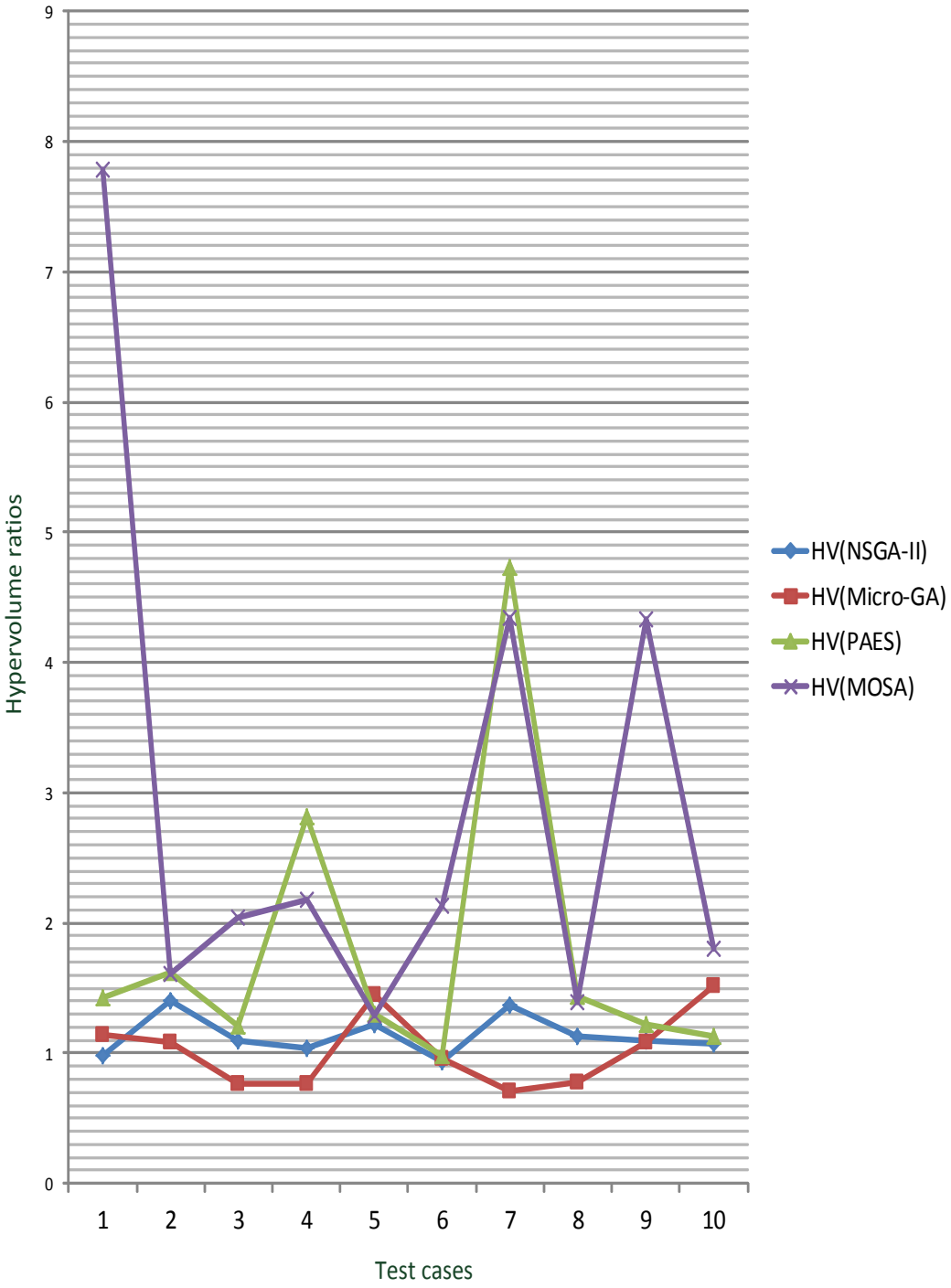


Figure 5.7: Results of the HV ratios for the proposed Off-Spring Non-Storing GA algorithm on the NY road network.

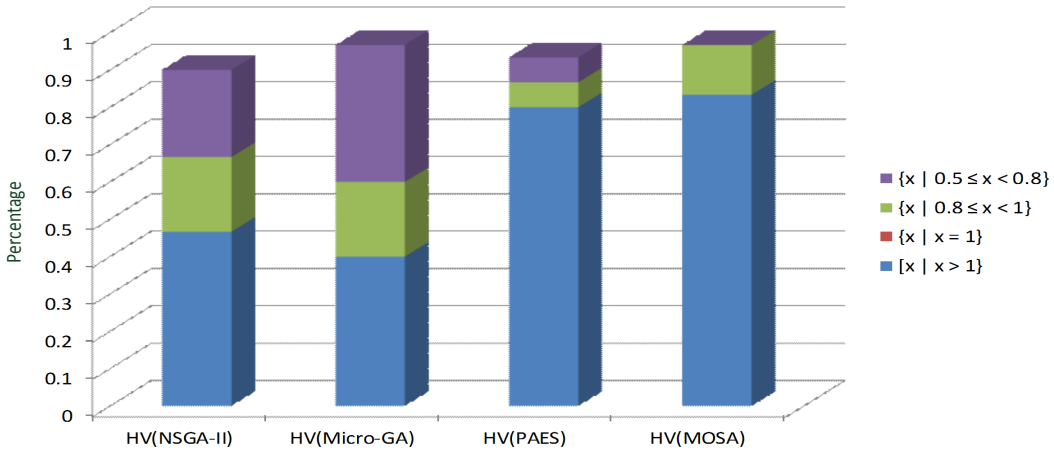


Figure 5.8: Summary of the HV ratio results of the Proposed Off-Spring Non-Storing GA Algorithm.

Chapter 6

Application of the Proposed Algorithms to the Vehicle Navigation Problem of Battery Electric Vehicles

6.1 Introduction

This chapter shows the application of the proposed algorithms to solve the navigation problem of battery electric vehicles (BEVs). The navigation problem in a BEV aims to find Point-to-Point (P2P) multi-objective shortest paths (MOSP) in the selected road network.

The intersections in a road network for BEVs are assumed to contain Contactless Power Transfer (CPT) units [31]. The CPTs are installed in the roads and use inductive wireless recharging technology to recharge the BEVs while they are waiting on the traffic signal at the intersection. The recharging capability of a CPT depends on its power rating. CPTs of four different power ratings are available: (a) 10 kW, (b) 20 kW, (c) 30 kW, and (d) 40 kW. The capability of a CPT to recharge BEVs per unit time increases with its power rating. Therefore, the intersections that are equipped with high power CPTs can provide more recharging than the CPTs that are equipped with less powerful recharging stations.

Chopra, et al. [32] investigated the rate of charging or discharging of the BEVs when the road network contains CPTs. They assumed that BEVs are running according to one of the following standard driving cycles: (a) U.S. standard FTP 72 (Federal Test Procedure) cycle also called Urban Dynamometer Driving Schedule (UDDS); (b) European standard ECE-EUDC combined urban test cycle; (c) Japanese standard JC08 urban test cycle. Each driving cycle specifies the characteristics such as: (a) the total distance of the path; (b) total duration; and (c)average speed of the vehicle, etc. Chopra, et al. determined charging rates for the BEVs when the intersections contain different types of CPTs. In our experiments, we used the road networks of the United States of America (as described in Chapter 5). Therefore, the results of the UDDC cycle

Table 6.1: Charging rate of the BEVs using different types of CPTs

Power rating of the CPT (kW)	total percentage charging (%)	total travelling time (seconds)	% charging per minute (%/minute)
0 kW	-7.8	1396	-0.3352
10 kW	-5.6	1396	-0.2454
20 kW	-2.9	1396	- 0.1271
30 kW	-0.2	1396	- 0.0088
40 kW	+2	1396	+0.0887

give best approximate values for the charging rate of the BEVs.

Table 6.1 shows the results which were presented by Chopra, et al. for a UDDC driving cycle. In Table 6.1, the first column mentions the power rating of the CPT. The 0 kW CPT means that no CPTs exist at the intersection. The second column mentions the percentage charge of the BEV from its initial value after completing a UDDC driving cycle. For example, if the initial battery level of any BEV is 80%, then on a road network that has 10 kW CPTs installed at the intersections, the battery level of the BEV at the end of an UDDC driving cycle would be 74.2% (because, $80 + (-5.6) = 74.4$). The negative sign refers to discharging and the positive sign refers to charging. The third column mentions the time for which the BEV travelled. The fourth column mentions the charging rate per minute. In a strict sense, the values in the fourth column are only valid for the BEVs that are following the requirements of the UDDC driving cycle. However, the UDDC cycle was designed considering most common driving conditions in the United States of America. Therefore, same results can be used for the other BEVs that are travelling in the road networks of the United States of America and not following any particular driving cycle. The charging rate per minute is used in our experiments to determine the charging rates of the edges in the network. The networks can also contain more than one type of CPTs. When the CPTs at the starting and ending nodes of an edge are different then the average charging rates of the two nodes is considered as the charging rate of that edge. The percentage charge per minute for any power rating of CPT can be obtained by calling the function; $y = Rat(x)$, where x is the power rating of the CPT and $x \in \{0, 10, 20, 30, 40\}$. The value of y is obtained from Table 6.1.

Let us consider a road network $G(V, E)$. V contains all nodes or intersections in the road network and E contains the edges or roads that join the intersections. Any node $v_i \in V$ is associated with a property $v_i.r$ that contains a value from the set $\{0, 10kW, 20kW, 30kW, 40kW\}$ that specifies the power rating of the CPT that is present at the node v_i. In the experiments, the type of CPTs is randomly chosen. Any edge $e_i \in E$ has a starting node v_x and an ending node v_y and is associated with up to three attributes $e_i.l$, $e_i.S_A$, and $e_i.s$. $e_i.l$ represents the length of the edge; $e_i.S_A$ represents the average travelling time on the edge; and $e_i.s$ represents the charging rate on e_i. If v_x and v_y are the starting and ending nodes of e_i, respectively, then $e_i.s = \frac{Rat(v_x.r)+Rat(v_y.r)}{2}$.

A driver selects two distinct nodes s and d in the road network and wants to find one or more paths from s to d. A BEV starts its journey with some battery level that is represented as B_{ini} and its value can vary between 0 and 1 and represents the percentage of battery which is charged at the source node (s). If P represents a path between nodes s and d, then $P \subseteq E$. The first and last elements in P can be represented as e_s and e_d such that the starting node

Input: B_{ini}: initial battery level of the BEV, P: a solution or path
Output: *result*: true or false, depends on if the constraint is satisfied or not
1: $B_c = B_{ini}$, $result = true$
2: **for** each edge $e_x \in P$ **do**
3: $B_c = B_c + (e_x.s \times e_x.S_A)$
4: **if** $B_c < 0$ **then**
5: $result = false$
6: **end if**
7: **end for**
8: **return** *result*

Figure 6.1: Function $g_1(B_{ini}, P)$.

of e_s is s and the ending node of e_d is d. The starting node of all edges in P except the first edge in P should be same as the ending node of the preceding edge. The MOSP problem contains two objective functions and one constraint. The objective functions can be defined as: (i) $f_1(P)$, which is the total length of P; and (ii) $f_2(P)$, which is the total travelling time on P. Mathematically:

$$f_1(P) = \sum_{e_i \in P} e_i.l \tag{6.1}$$

$$f_2(P) = \sum_{e_i \in P} e_i.S_A \tag{6.2}$$

The constraint is represented as $g_1(P)$ and is shown in Fig. 6.1. It is used to ensure that the BEV do not run out of battery during its journey. In Fig. 6.1, the inputs are the BEV's initial battery level (i.e. B_{ini}), and the solution P. In line 1, B_c initializes to B_{ini} and *result* to true. The for loop in lines 2 to 7, adds the percentage charging of each edge in P into B_c. If the B_c value becomes negative at any time then it means that the BEV cannot travel on the path P. The path or solution P is a feasible solution, if and only if $g_1(P)$ is true. The objective functions, constraints and the MOSP problem can be mathematically expressed as follows:. The objective function of the MOSP problem for the BEVs can be defined as follows:

$$\begin{aligned} &Minimize(f_1(P), f_2(P)), \\ &\text{such that } g_1(P) = true \end{aligned} \tag{6.3}$$

The pareto-optimal set is represented as S and contains only feasible pareto-optimal solutions.

6.2 Proposed StocE Algorithm

The implementation and parameter values of the algorithms remains same as already discussed in the Section 5.3 of the previous chapter. The performance of the proposed StocE based algorithm is compared with the existing population and single-solution-based algorithms. The existing algorithms include: (i) NSGA-II, (ii) Micro-GA, (iii) (1-1)-PAES, (iv) MOSA, and (v) Std-StocE. The ratios of the HVs between the proposed StocE-based algorithm to the existing algorithms were determined.

Figs. 6.2 to 6.4 shows the results of the HVs ratios of the experiments. Fig. 6.2 shows the results of the BAY road network. Fig. 6.7 shows the results of the COL road network. Fig. 6.4 shows the results of the NY road network. In Fig. 6.2, the Proposed StocE-based algorithm obtains average HV values that are better than NSGA-II in 70% test cases, it also average HV values better than Micro-GA in 80% test cases. The proposed GA-based algorithm obtains also performs better than the (1-1)-PAES and MOSA in most of the test cases. In Fig. 6.3, the proposed algorithm obtains a better value than the other algorithms as follows: (a) better than NSGA-II in 50% test cases, (ii) better than Micro-GA in 80% test cases, (iii) better than (1-1)-PAES and MOSA in all test cases.

The results for all road networks are summarized and the plot in Fig. 6.5 shows the summary of all test cases. The intervals $\{x|x > 1\}$ and $\{x|x = 1\}$ indicates percentage of test cases in which the proposed algorithm obtains an average HV value that is better than or equal to the existing algorithms. The Proposed StocE-based algorithm performs better than NSGA-II in 30% test cases; it also performs better than Micro-GA in 40% experiments; it also performs better than (1-1)-PAES in 67% test cases and; it performs better than MOSA in 94% test cases. The main reason for the better performance of the proposed algorithm is its ability to explore more paths in the network through the use of the innovative Perturb operation. .

6.3 Proposed Off-Spring Non-Storing GA

The implementation of the algorithms and parameters values was same as discussed in Section 5.4 of the previous chapter. The proposed algorithm was compared with NSGA-II, Micro-GA, (1-1)-PAES and MOSA. The proposed GA based algorithm requires memory which is almost half of the NSGA-II and Micro-GA.

Figs. 6.6 to 6.8 shows the results of the HVs ratios of the experiments. Fig. 6.6 shows the results of the BAY road network. Fig. 6.7 shows the results of the COL road network. Fig. 6.8 shows the results of the NY road network. In Fig. 6.6, the Proposed Off-Spring Non-Storing algorithm obtains average HV values that are better than NSGA-II in 70% test cases, it also obtain average HV values that are better than that of Micro-GA in 80% test cases. The proposed GA-based algorithm also performs better than that of (1-1)-PAES in 90% test cases. It obtains average HV values better than MOSA in all test cases. In Fig. 6.7, the proposed algorithm obtains HV values as follows: (a) better than NSGA-II in 50% test cases, (ii) better than Micro-GA in 80% test cases, (iii) better than (1-1)-PAES and MOSA in all test cases. In Fig. 6.8, the proposed algorithm obtains HV values as follows: (a) better than NSGA-II and Micro-GA in 60% test cases, (ii) better than Micro-GA in 40% test cases, (iii) better than (1-1)-PAES and MOSA in all test cases.

The results for all road networks are summarized and the plot in Fig. 6.9 shows the summary of all test cases. The intervals $\{x|x > 1\}$ and $\{x|x = 1\}$ indicates percentage of test cases in which the proposed algorithm obtains an average HV value that is better than or equal to the existing algorithms. The Proposed GA-based algorithm performs better than NSGA-II in 60% test-cases. It also performs better than Micro-GA in 67% test cases. It performs better

than $(1-1)-PAES$ in 97% test cases. It performs better than MOSA in all test cases. Therefore, the Proposed Off-Spring Non-Storing GA outperformed the existing single-solution algorithms in terms of solution quality. It performs better than the population-based algorithms in 60% test cases while using a memory which is about half of the population-based algorithm. The innovative methods for selecting the GA operation for the chromosomes and the conditional replacement of the parent chromosomes by their children are found to be very successful in improving the solution quality and keeping the population size small.

6.4 Summary

The proposed algorithms are also applied to solve the MOSP problem of BEVs. The experiments were conducted on real road networks and assume CPTs exist at the intersections. The MOSP problem consists of two minimization objectives and one constraint. The minimization objectives consist of minimizing the length and travelling time of the path. On any path P, the BEV starts its travel with an initial battery level and on each edge in P the battery level of the BEV decreases up to some percent. The total percentage decrease in the battery level of the BEV should not exceed the initial battery level of the BEV. The proposed algorithms were implemented to find the pareto-optimal feasible solutions. The performances of the proposed algorithms were compared with some famous existing algorithms (NSGA-II, Micro-GA, PAES, MOSA and straightforward StocE). The comparison results showed that the proposed StocE based algorithm outperforms the existing single solution-based algorithms while using 25% more memory. It also performs better than the population-based algorithms in 17% test cases. The Proposed Off-Spring Non-Storing GA also outperforms existing single-solution-based algorithms. It performs better than the population-based algorithms in 60% test cases while using about half amount of memory. Therefore, the proposed algorithms are suitable to solve the MOSP problem in embedded systems and other platforms in which memory-efficiency is very important.

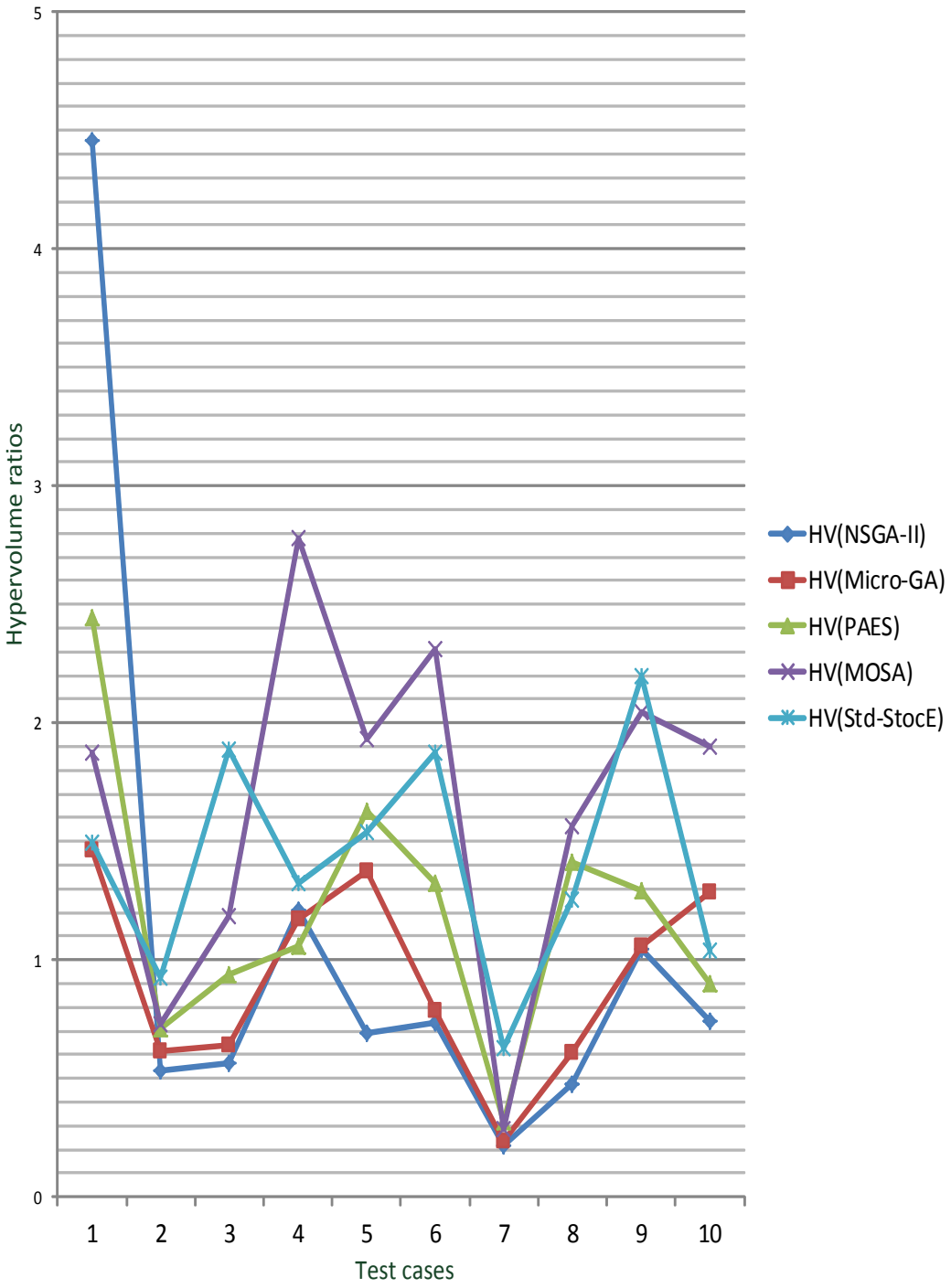


Figure 6.2: Results of the HV ratios for the proposed StocE-based algorithm on the BAY road network.

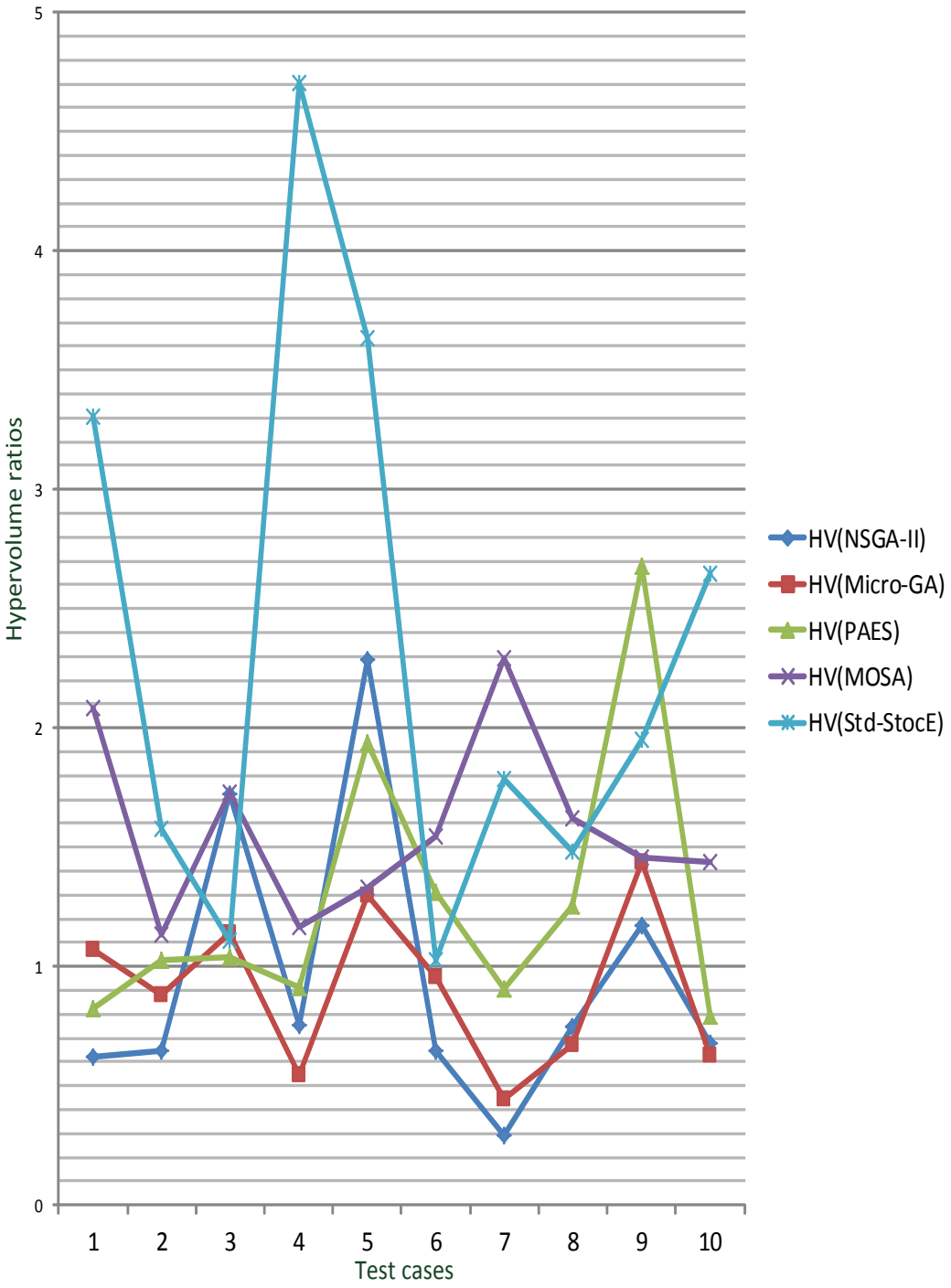


Figure 6.3: Results of the HV ratios for the proposed StocE-based algorithm on the COL road network.

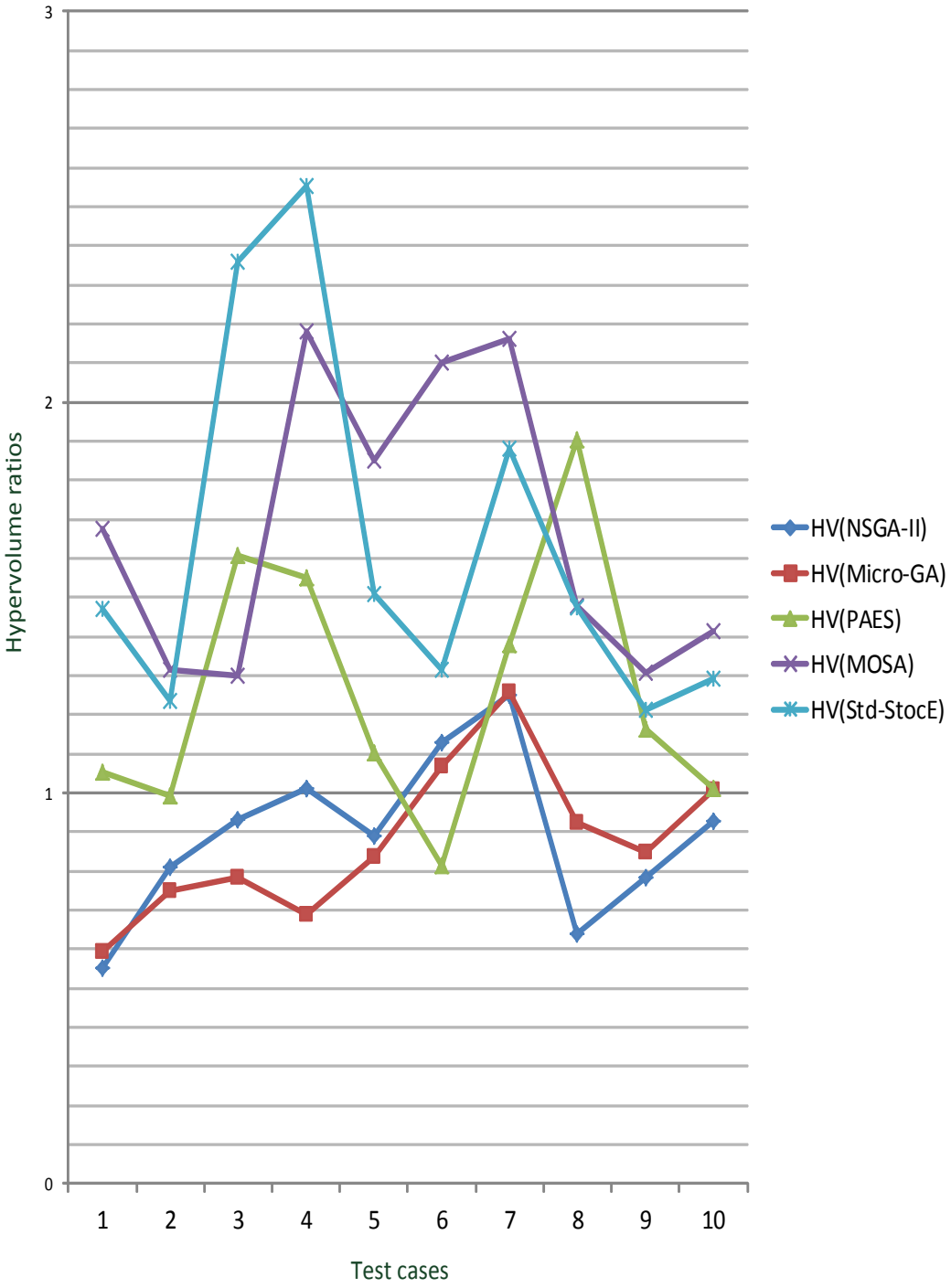


Figure 6.4: Results of the HV ratios for the proposed StocE-based algorithm on the NY road network.

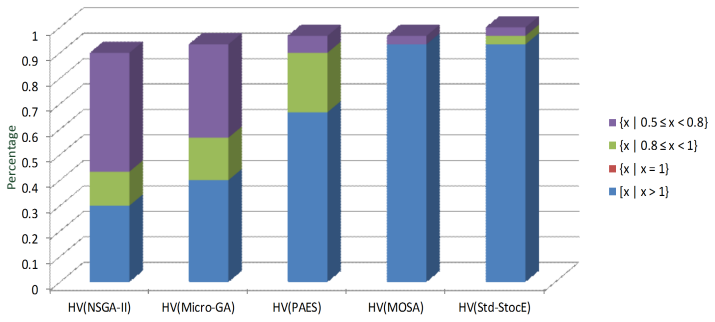


Figure 6.5: Summary of the HV ratio results of the Proposed StocE-based Algorithm.

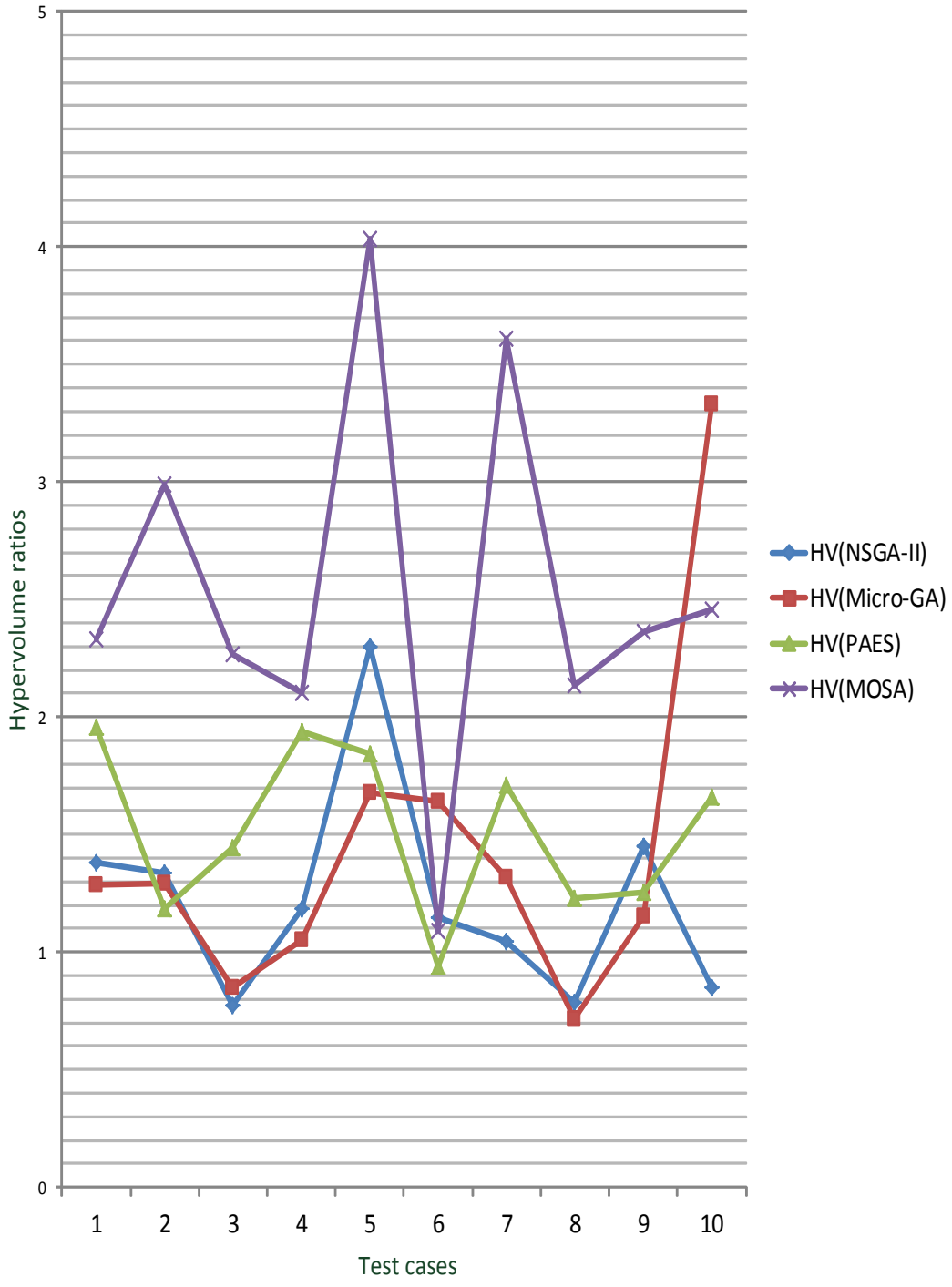


Figure 6.6: Results of the HV ratios for the Proposed Off-Spring Non-Storing GA on the BAY road network.

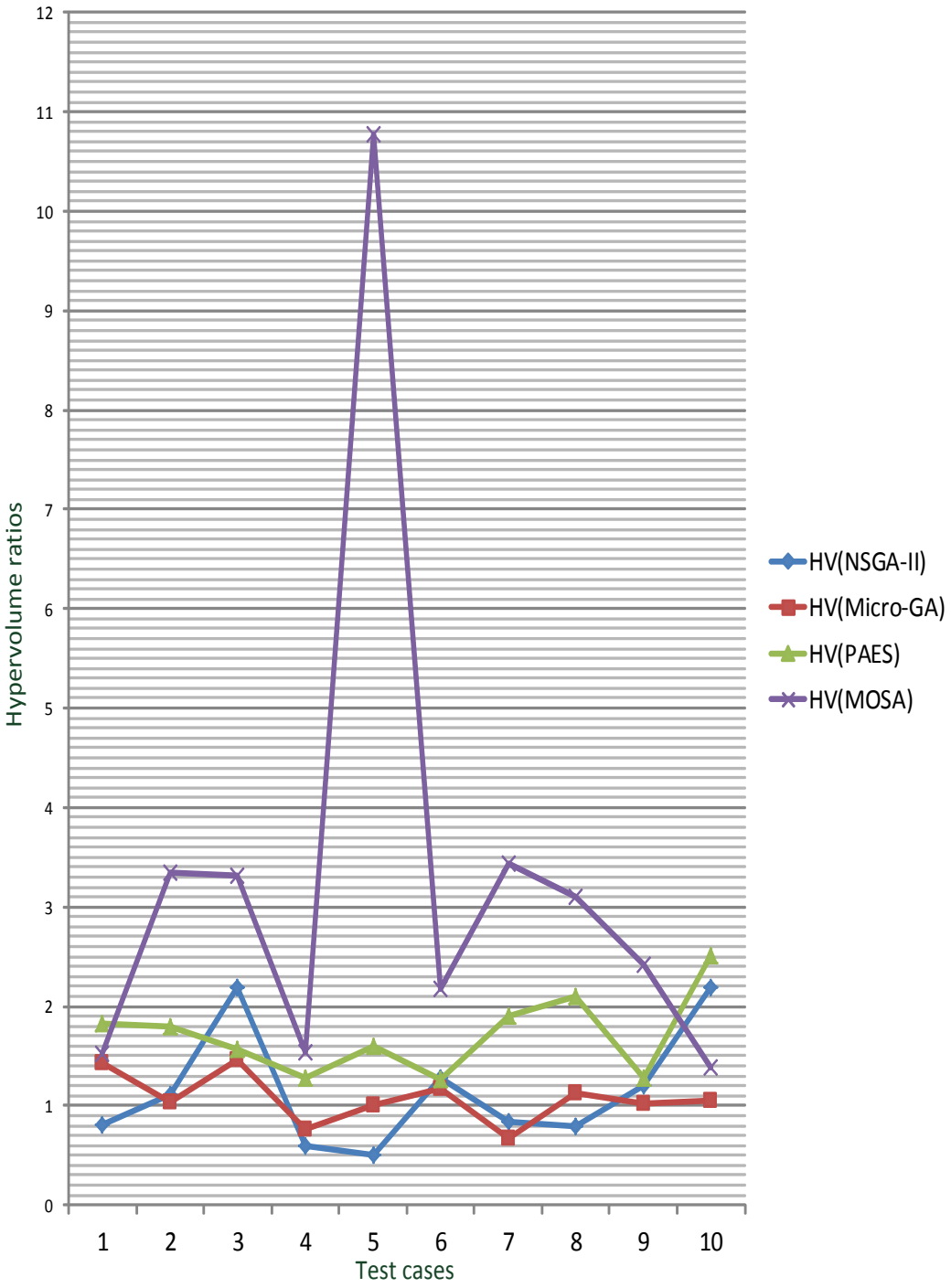


Figure 6.7: Results of the HV ratios for the Proposed Off-Spring Non-Storing GA on the COL road network.

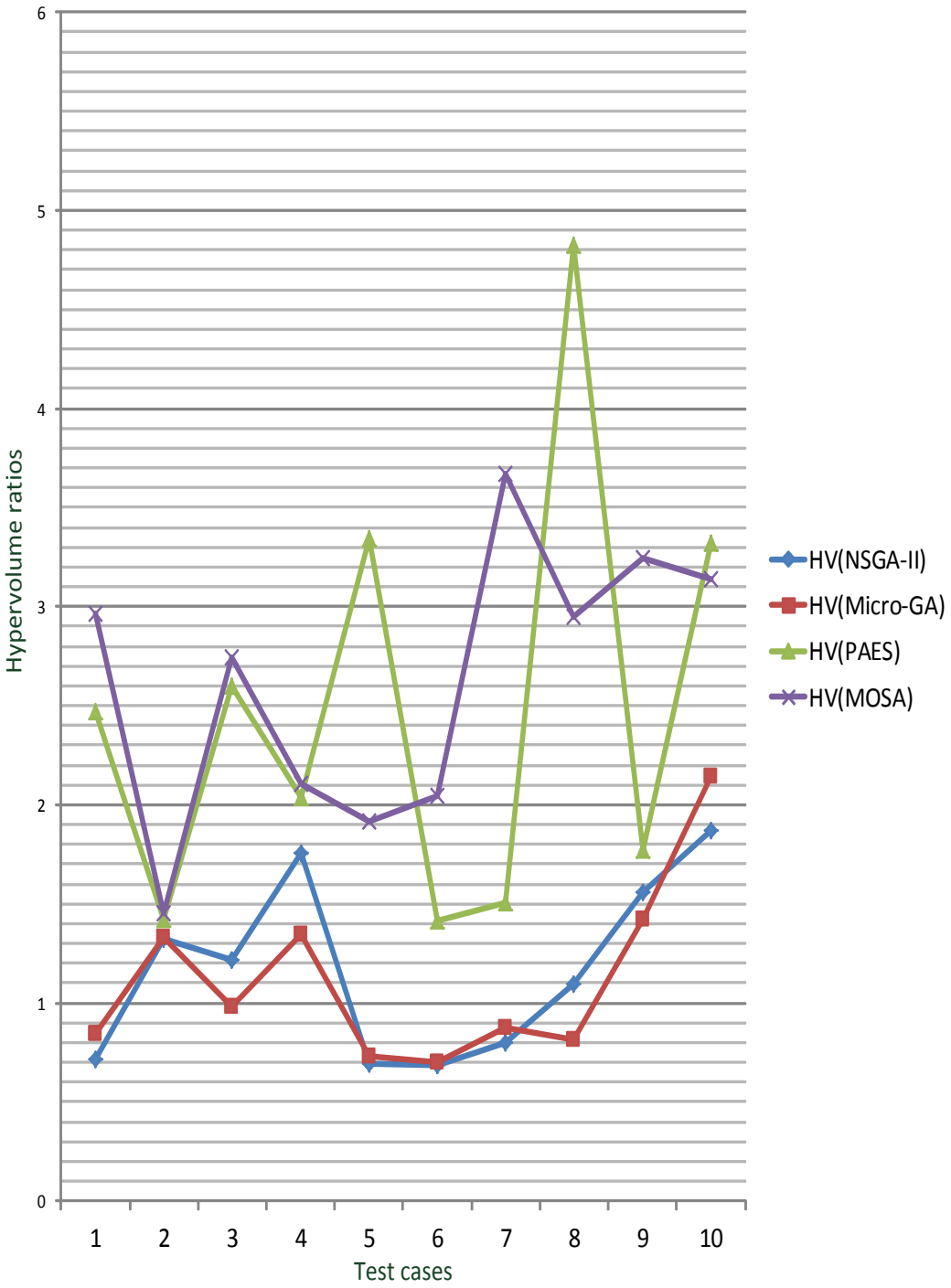


Figure 6.8: Results of the HV ratios for the Proposed Off-Spring Non-Storing GA on the NY road network.

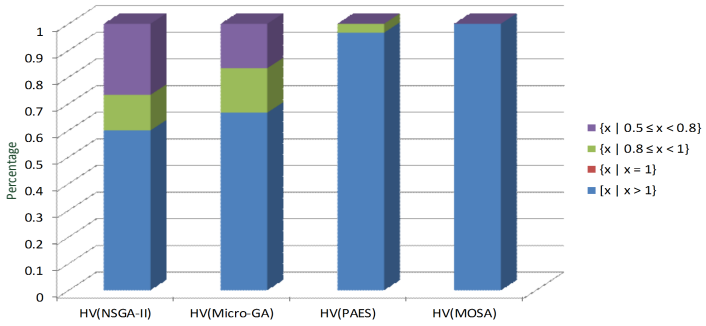


Figure 6.9: Summary of the HV ratio results of the Proposed Off-Spring Non-Storing GA Algorithm.

Chapter 7

Generalization of the Proposed Algorithms

An important feature of the Evolutionary Algorithms (EAs) is that they are not specific to any problem. Therefore, the proposed algorithms remain useful for applications other than the multi-objective shortest path (MOSP) problem. This chapter discusses the generalized designs of the proposed algorithms and their performance comparison with the existing algorithms. The proposed StocE-based algorithm and off-spring non-storing GA-based algorithms that are shown in Figs. 4.2 and 4.6 can be applied to solve any multi-objective optimization problem. However, the evolutionary operators that constitutes the proposed algorithms should be redefined based on their basic principles for a general multi-objective optimization problem. Several test problems exist for the multi-objective optimization problems. The test problems have known actual pareto-fronts. Many performance metrics exist for the test problems with known actual pareto-front. Therefore, the performance of the proposed algorithms can be evaluated and compared with the exiting algorithms in detail.

The continuous multi-objective optimization (MOP) problem having m objectives can be described as follows:

$$\begin{aligned} &\text{minimize } F(x) = (f_1(x), ..., f_m(x)) \\ &\text{subject to } x \in \Omega \end{aligned} \tag{7.1}$$

where Ω is the decision (variable) space, R^m is the objective space, and $F : \Omega \rightarrow R^m$ consists of m real-valued objective functions. A point $x \in \Omega$ is represented by a set of m elements, i.e., $x = (x_1, x_2, ..., x_m)$. A point x^* is called pareto-optimal if there is no $x \in \Omega$ such that $F(x)$ dominates $F(x^*)$. Any point $x \in \Omega$ is a possible solution and is used as a solution or chromosome in the proposed algorithms.

This chapter is organized as follows: The first two sections describe the generalized versions of the proposed StocE-based and off-spring non-storing GA-based algorithms. The third section shows the simulation results.

7.1 Proposed StocE-based Algorithms

The proposed StocE-based algorithm is introduced in chapter 4. Fig. 7.1 shows the generalized form of the proposed StocE-based algorithm. The generalized form of the proposed StocE-based algorithm is slightly different from its version to solve the MOSP problem. The Perturb operation in the generalized StocE-based algorithm returns a set SA that contains a maximum of Num number of solutions, instead of a single solution. Therefore, the step $str\_optimal(..)$ that is used to store the solutions in SA into the archive of pareto-optimal solutions is present inside the Perturb cycle. The proposed StocE-based algorithm contains two evolutionary operators that are: Perturb operation and Mutation. The following two sub-sections describe the general designs of the Perturb and mutation operations.

7.1.1 Perturb Operation

Chapter 4 described the Perturb operation for the MOSP problem. It selects different sub-paths in the solution, determine their suitability to remain in the next generation and then finally replace a least suitable sub-path with another sub-path. A solution in the MOSP problem is a set of interconnected edges and the value of each element is dependent on its predecessor. If we want to replace just one element in the solution without effecting the remaining elements than very few alternate edges exist. Therefore, in-order to increase the possible choices we used sub-paths instead of just one edge. The sub-paths are created by using a method in which the destination node is searched from the source node in a fashion similar to the breadth-first-search. When the source and destination nodes are too far then it can be slow. Therefore, the creation of new sub-paths is reduced by first computing the suitability of the existing sub-paths and then generating just one new sub-path to replace an existing least suitable sub-path.

In a general continuous MOP, a solution contains up to m elements or variables such that each element can select its value independently from its set of possible values. Alternate choices for any element can be generated by randomly selecting values from its set of possible values. Therefore, generating alternate solutions is not an expensive operation. The Perturb operation is shown in Fig. 7.2. The inputs are: the current solution S and positive integer Num. The Perturb operation creates up-to Num number of new solutions and store in the set SA. It returns the set SA at the end. The steps performed in the Perturb operation are as follows: An element (x_r) is randomly selected in the solution. A loop is executed for Num number of times. In each iteration, it first generates a random number R that belongs to the set of possible values for x_r. Then the value of the element x_r is assigned equal to R to form a new solution S'. S' is similar to S in all elements except x_r. S' is added into SA. If S' dominates S then the existing current solution S is updated to S'. After the Perturb operation is completed, then the current solution S may be updated to a new value and several solutions that are not dominated by S are contained in the set SA. All solutions in SA are inserted into the archive of pareto-optimal solutions, if they remains non-dominated in comparison with the solutions that already present in the archive.

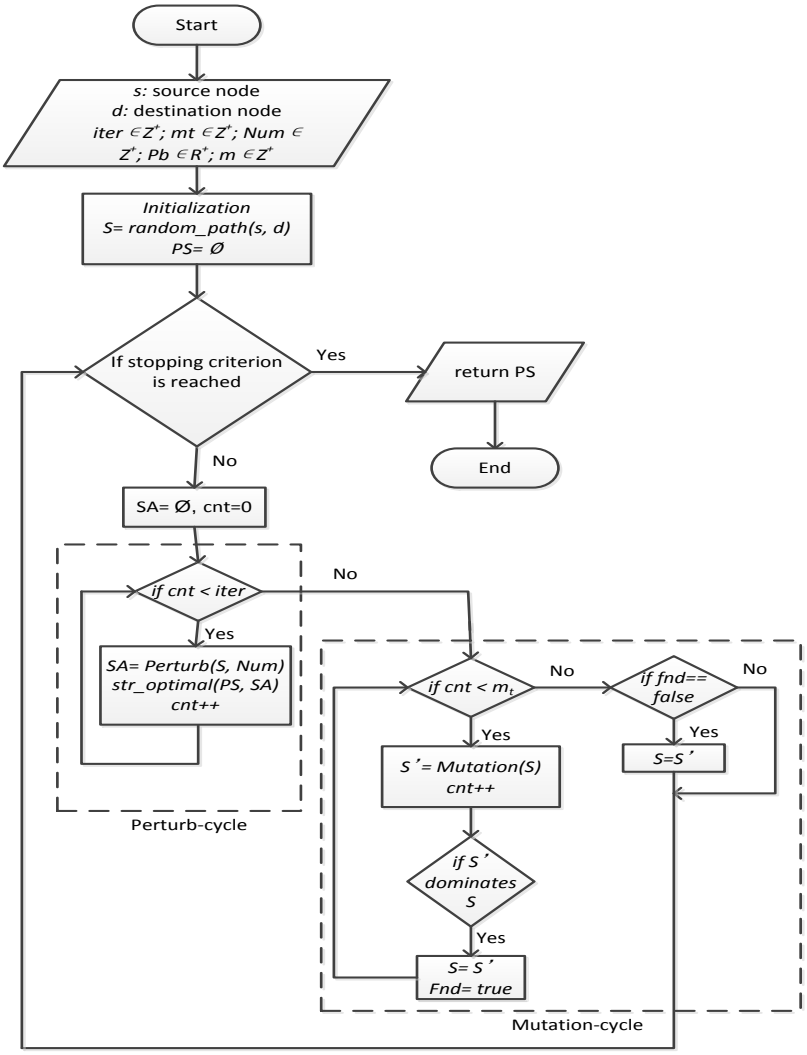


Figure 7.1: Proposed StocE-based algorithm for the general MOP problems.

Input: $S = \{x_1, x_2, ..., x_m\}$: current solution, $Num \in Z^+$: Number of iterations in the Perturb Operation.
Output: SA: set of solutions created in the Perturb operation.

```
1: SA = null; x_r: a randomly selected element from S
2: for i= 1 to Num do
3:    generate R (where R is a randomly selected value for the x_r element from
      its set of possible values.)
4:    S' = S, S'(x_r) = R
5:    SA = SA ∪ S'
6:    if S' dominates S then
7:       S = S'
8:    end if
9: end for
10: return SA
```

Figure 7.2: Perturb Operation for the general MOP, $SA = Perturb(S, Num)$

7.1.2 Mutation

The proposed mutation operation forms a new solution from scratch. In the mutated solution, the values of all elements in the solution are randomly selected from their sets of possible values.

7.2 Off-Spring Non-Storing GA

The proposed Off-spring Non-Storing GA-based algorithm is shown in Fig. 4.6 and is described in chapter 4. The algorithm uses a novel procedure that selects a GA operator for the chromosome. The algorithm consists of two evolutionary operators: Crossover and Mutation. The procedure to select a GA operator for the chromosomes and the evolutionary operators should be defined for a general MOP. The population is represented as *Population* and contains up to N chromosomes.

7.2.1 Selection of the GA Operation

In the MOSP problem, the crossover operation can be applied to the chromosomes that have at-least one common node with another non-dominated chromosome. In the general MOP, there is no such type of restriction. Both the crossover and mutation operations can be applied to any chromosome. If the crossover operation is selected for the chromosome $P_i \in Population$, then the second parent is any non-dominated chromosome $P_k^* \in Population$. The procedure to select a GA operator for the chromosome $P_i \in Population$ is shown in Fig. 7.3.

For the non-dominated chromosomes in the *Population*, the mutation operation is selected with probability P_b and the crossover operation can be selected with probability $1 - P_b$. For the remaining chromosomes, the crossover operation is selected with probability P_b and the mutation operation can be selected with probability $1 - P_b$.

Input: $P_i = \{x_1, x_2, ..., x_m\}$: a chromosome from the *Population*, P_b: Probability value
Output: *GAOper*: the selected GA operator for the chromosome P_i

```
1: r: a randomly generated real number between [0, 1]
2: if P_i is a non-dominated solution in the Population then
3:    if r ≤ P_b then
4:       GAOper = Mutation
5:    else
6:       GAOper = Crossover
7:    end if
8: else
9:    if r ≤ P_b then
10:      GAOper = Crossover
11:   else
12:      GAOper = Mutation
13:   end if
14: end if
15: return GAOper
```

Figure 7.3: Method to select a GA operation for the chromosome P_i, $GAOper = SelectGAOper(P_i)$

Different types of crossover operators have been proposed that can be used in the proposed algorithm. Some examples of the crossover operators include: Simulated Binary Crossover (SBX) [33], Partially Matched Crossover (PMX) [16], Single Point Crossover and Two Point Crossover [16]. Different types of mutation operations are also proposed that includes: Polynomial mutation [34], uniform mutation and non-uniform mutation [35].

7.3 Simulations

The Proposed StocE and Off-Spring Non-Storing GA-based algorithms were implemented using Java programing language on the jMetal toolkit [36, 37]. jMetal was developed for the purpose of promoting the research of evolutionary algorithms. It contains implementations of several multi-objective optimization algorithms, evolutionary operators, and performance metrics. The implementations of the proposed algorithms contain new implementation of their novel components and the existing implementations of the standard GA functions like crossover, mutation, finding non-dominance, etc. The jMetal also contains several test problems that have known pareto-fronts. The following sections describes the test problems on which the algorithms were evaluated; parameters of the algorithms; and the simulation results on the test problems.

7.3.1 Test Problems

Hui Li and Qingfu Zhang [38] have introduced a suite of nine test problems that are named as: $LZ09\_F1$ to $LZ09\_F9$. The test problems have complicated shape pareto-fronts that are difficult to achieve by any algorithm. Table 7.1 shows the test problems for multi-objective optimization as proposed by Hui Li

Table 7.1: Description of test problems

Problem	Objective Functions	Variable Bounds
$LZ09\_F1$	$f_1 = x_1 + \frac{2}{J_1}\sum_{j\in J_1}(x_j - x_1^{0.5(1.0+\frac{3(j-2)}{n-2})^2})$ $f_2 = 1 - \sqrt{x_1} + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2}(x_j - x_1^{0.5(1.0+\frac{3(j-2)}{n-2})})^2$ where $J_1 = \{j \mid j \text{ is odd and } 2 \le j \le n\}$ and $J_2 = \{j \mid j \text{ is even and } 2 \le j \le n\}$	$[0,1]^n$
$LZ09\_F2$	$f_1 = x_1 + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}(x_j - sin(6\pi x_1 + \frac{j\pi}{n}))^2$ $f_2 = 1 - \sqrt{x_1} + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2}(x_j - sin(6\pi x_1 + \frac{j\pi}{n}))^2$ where J_1 and J_2 are same as those of $LZ09\_F1$	$[0,1]\times[-1,1]^{n-1}$
$LZ09\_F3$	$f_1 = x_1 + \frac{2}{\sqrt{J_1}}\sum_{j\in J_1}(x_j - 0.8x_1 cos(6\pi x_1) + \frac{j\pi}{n})^2$ $f_2 = 1 - \sqrt{x_1} + \frac{2}{\lvert\sqrt{J_2}\rvert}\sum_{i\in J_2}(x_j - 0.8x_1 sin(6\pi x_1 + \frac{j\pi}{n}))^2$ where J_1 and J_2 are same as those of $LZ09\_F1$	$[0,1]\times[-1,1]^{n-1}$
$LZ09\_F4$	$f_1 = x_1 + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}(x_j - 0.8x_1 cos(\frac{6\pi x_1 + \frac{j\pi}{n}}{3}))^2$ $f_2 = 1 - \sqrt{x_1} + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2}(x_j - 0.8x_1 sin(6\pi x_1 + \frac{j\pi}{n}))^2$ where J_1 and J_2 are same as those of $LZ09\_F1$	$[0,1]\times[-1,1]^{n-1}$
$LZ09\_F5$	$f_1 = x_1 + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}\{x_j - [0.3x_1^2 cos(24\pi x_1 + \frac{4j\pi}{n}) + 0.6x_1]cos(6\pi x_1 + \frac{j\pi}{n})\}^2$ $f_2 = 1 - \sqrt{x_1} + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2}\{x_j - [0.3x_1^2 cos(24\pi x_1 + \frac{4j\pi}{n}) + 0.6x_1]sin(6\pi x_1 + \frac{j\pi}{n})\}^2$ where J_1 and J_2 are same as those of $LZ09\_F1$	$[0,1]\times[-1,1]^{n-1}$
$LZ09\_F6$	$f_1 = cos(0.5x_1\pi)cos(0.5x_2\pi) + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}(x_j - 2x_2 sin(2\pi x_1 + \frac{j\pi}{n}))^2$ $f_2 = cos(0.5x_1\pi)sin(0.5x_2\pi) + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2}(x_j - 2x_2 sin(2\pi x_1 + \frac{j\pi}{n}))^2$ $f_3 = sin(0.5x_1\pi) + \frac{2}{\lvert J_3\rvert}\sum_{j\in J_3}(x_j - 2x_2 sin(2\pi x_1 + \frac{j\pi}{n}))^2$ where $J_1 = \{j \mid 3 \le j \le n,$ and j-1 is a multiplication of 3 $\}$ $J_2 = \{j \mid 3 \le j \le n,$ and j-2 is a multiplication of 3 $\}$ $J_3 = \{j \mid 3 \le j \le n,$ and j is a multiplication of 3 $\}$	$[0,1]^2\times[-2,2]^{n-2}$
$LZ09\_F7$	$f_1 = x_1 + \frac{2}{\lvert J_1\rvert}(4\sum_{j\in J_1} y_j^2 - 2\prod_{j\in J_1} cos(\frac{20y_j\pi}{\sqrt{j}} + 2)$ $f_2 = 1 - \sqrt{x_1} + \frac{2}{\lvert J_2\rvert}(4\sum_{j\in J_2} y_j^2 - 2\prod_{j\in J_2} cos(\frac{20y_j\pi}{\sqrt{j}} + 2)$ where J_1 and J_2 are same as those of $LZ09\_F1$ and $y_j = x_j - x_1^{0.5(1.0+\frac{3(j-2)}{n-2})}$, $j = 2, .., n.$	$[0,1]^n$
$LZ09\_F8$	$f_1 = x_1 + \frac{2}{\lvert J_1\rvert}(4\sum_{j\in J_1} y_j^2 - 2\prod_{j\in J_1} cos(\frac{20y_j\pi}{\sqrt{j}}) + 2)$ $f_2 = 1 - \sqrt{x_1} + \frac{2}{\lvert J_2\rvert}(4\sum_{j\in J_2} y_j^2 - 2\prod_{j\in J_2} cos(\frac{20y_j\pi}{\sqrt{j}}) + 2)$ where J_1 and J_2 are same as those of $LZ09\_F1$ and $y_i = x_j - x_1^{0.5(1.0+\frac{3(j-2)}{n-2}}, j = 2, ..., n$	$[0,1]^n$
$LZ09\_F9$	$f_1 = x_1 + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}(x_j - sin(6\pi x_1 + \frac{j\pi}{n}))^2$ $f_2 = 1 - x_1^2 + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2}(x_j - sin(6\pi x_1 + \frac{j\pi}{n}))^2$ where J_1 and J_2 are same as those of $LZ09\_F1$	$[0,1]\times[-1,1]^{n-1}$

and Qingfu Zhang. In Table 7.1, the first column shows the problem name, the second column describes the objective functions (f_1, f_2 and f_3) of the problem. The third column mentions the possible set of values for the variables. The number of variables (i.e., the values of n) which were used in the experiments are shown in Table 7.2.

7.3.2 Algorithm Parameters

The jMetal tool-kit contains implementations of several existing algorithms that includes: NSGA-II, Strength Pareto Evolutionary Algorithm 2 (SPEA2) [39], Pareto Archived Evolution Strategy -2 (PAES-2), (1-1)-PAES, etc. The proposed algorithms were compared with NSGA-II, SPEA2 and (1-1)-PAES. NSGA-II and SPEA2 are population-based algorithms, whereas, (1-1)-PAES is a single solution based algorithm. NSGA-II requires to store number of paths equal to the twice of the population size. If the Population size is $N_{NSGA-II}$, then NSGA-II requires to store a total of $2N_{NSGA-II}$ paths in the memory.

Table 7.2: Number of variables (i.e., value of n) used in the experiments.

Test Problem	Value of n
$LZ09\_F1$	10
$LZ09\_F2$	30
$LZ09\_F3$	30
$LZ09\_F4$	30
$LZ09\_F5$	30
$LZ09\_F6$	10
$LZ09\_F7$	10
$LZ09\_F8$	10
$LZ09\_F9$	30

SPEA2 maintains a population of solutions as well as an archive of Pareto-optimal solutions. Therefore, if the population size is N_{SPEA2} and archive size is A_{SPEA2}, then it requires to store $2N_{SPEA2} + A_{SPEA2}$ paths in the memory. The (1-1)-PAES requires to store number of solutions equal to $A_{PAES}+2$ in the memory, where A_{PAES} is the size of the archive to store pareto-optimal solutions. The proposed StocE-based algorithm needs to store a total of $2+m+Num$ number of paths in the memory, where m is the size of the archive of pareto-optimal solutions and Num is the number of iterations in the Perturb cycle. The proposed off-spring non-storing GA requires to store a total of $2 + N_{GA}$ solutions in the memory, where N_{GA} is the population size. The solution quality of the different algorithms were compared at an equal execution time and memory requirement. The execution time (or stopping criteria) of the algorithms was set to 20 seconds. The experiments were divided into two cases: case I and case II. In case I, the total memory requirement of the algorithms was set to 300 solutions. In case II, the total memory requirement of the algorithms was set to 100 solutions. The parameter values of the algorithms are shown in Table 7.3. The algorithms uses the SBX Crossover with probability equal to 0.90. The algorithms used Polynomial mutation operator with probability equal to $\frac{1}{\text{Number of variables}}$. NSGA-II used Binary tournament selection for selecting the parents for the crossover operation.

7.3.3 Results of the Test Problems

The nine test problems were solved using the proposed StocE and GA-based algorithms as well as using the NSGA-II and SPEA2. Each test problem was solved by any particular algorithm for up-to ten times. In each trial, the values of the performance metrics were calculated and the average value of the performance metrics in the ten trials is reported in this subsection. The performance of the algorithms was measured in terms of the following performance metrics: Hypervolume Ratio (HVR), Generational Distance (GD), and Inverse Generational Distance (IGD). HVR is the ratio between the pareto-front obtained from the algorithm to the actual pareto-front of the problem. Its value lies between 0 and 1 and a value closer to 1 is considered better. GD is the average distance of the solutions in the obtained paret-front from nearest member of the actual pareto-front. A smaller value of GD is considered better and the obtained pareto-front is considered to be closer to the actual front. IGD is the distance of the actual pareto-front from the obtained front. A smaller value of IGD shows that the obtained pareto-front is closer to the actual pareto-front and is also uniformly spread along the actual pareto-front.

The results of case I are shown in Figs. 7.4. Fig. 7.4 contains three graphs

Table 7.3: Values of parameters.

Proposed StocE-based Algorithm			
Case I		Case II	
Parameter	Value	Parameter	Value
m	278	m	88
$iter$	10	$iter$	10
Num	20	Num	10
m_t	10	m_t	10
Proposed GA-based Algorithm			
Case I		Case II	
Parameter	Value	Parameter	Value
N_{GA}	298	m	98
P_b	0.95	P_b	0.95
NSGA-II			
Case I		Case II	
Parameter	Value	Parameter	Value
$N_{NSGA-II}$	150	$N_{NSGA-II}$	50
SPEA2			
Case I		Case II	
Parameter	Value	Parameter	Value
N_{SPEA2}	100	N_{SPEA2}	25
A_{SPEA2}	100	A_{SPEA2}	50
(1-1)-PAES			
Case I		Case II	
Parameter	Value	Parameter	Value
A_{PAES}	298	A_{PAES}	98

(a), (b) and (c). The graph (a) shows the average HVR values of the ten trials for each test problem. The graph (b) shows the average GD values for each test problem and the graph (c) shows the average IGD values for the test problems. The graphs show that in many test problems, the proposed algorithms obtains a better results than the existing algorithms. The comparison is quantitatively shown in Fig. 7.5. The y-axis mentions the percentage of test problems in which the proposed algorithms obtains a better value than that of the algorithm which is mentioned on the x-axis.

The results are summarized in Fig. 7.5. Fig. 7.5 shows that the Proposed StocE-based algorithms performance is as follows. In terms of HVR values: (i) It performs better than NSGA-II in 67% test problems, (ii) it performs better than SPEA2 in 56% test problems, (iii) it performs better than (1-1)-PAES in 78% test problems. In terms of GD values: (i) It performs better than NSGA-II in 22%, (ii) it performs better than SPEA2 in 22% test problems, (iii) it performs better than (1-1)-PAES in 100% test problems. In terms of IGD values: (i) It performs better than NSGA-II in 67%, (ii) it performs better than SPEA2 in 89% test problems, (iii) it performs better than (1-1)-PAES in 89% test problems.

Fig. 7.5 shows that the Proposed GA-based algorithm is as follows. In terms of HVR values: (i) It performs better than NSGA-II in 56% test problems, (ii) it performs better than SPEA2 in 33% test problems, (iii) it performs better than (1-1)-PAES in 44% test problems. In terms of GD values: (i) It performs better than NSGA-II in 100%, (ii) it performs better than SPEA2 in 89% test problems, (iii) it performs better than (1-1)-PAES in 79% test problems. In terms of IGD values: (i) It performs better than NSGA-II in 22%, (ii) it performs better than SPEA2 in 44% test problems, (iii) it performs better than (1-1)-PAES in 67% test problems.

The results of case II are shown in Fig. 7.6. Similar to Fig. 7.4, it also has three graphs. The results show that in many test problems the proposed

algorithms performs better than the existing algorithms. The results are summarized in Fig. 7.7. Fig. 7.7 shows that the Proposed StocE-based algorithms performance is as follows. In terms of HVR values: (i) It performs better than NSGA-II in 67% test problems, (ii) it performs better than SPEA2 in 56% test problems, (iii) it performs better than (1-1)-PAES in 56% test problems. In terms of GD values: (i) It performs better than NSGA-II in 44% test problem, (ii) it performs better than SPEA2 in 22% test problems, (iii) it performs better than (1-1)-PAES in 22% test problems. In terms of IGD values: (i) It performs better than NSGA-II in 89% test problems, (ii) it performs better than SPEA2 in 89% test problems, (iii) it performs better than (1-1)-PAES in 100% test problems.

Fig. 7.7 shows that the Proposed GA-based algorithm is as follows. In terms of HVR values: (i) It performs better than NSGA-II in 100%, (ii) it performs better than SPEA2 in 78% test problems, (iii) it performs better than (1-1)-PAES in 67% test problems. In terms of GD values: (i) It performs better than NSGA-II in 89% test problems, (ii) it performs better than SPEA2 in 89% test problems, (iii) it performs better than (1-1)-PAES in 89% test problems. In terms of IGD values: (i) It performs better than NSGA-II in 67% test problems, (ii) it performs better than SPEA2 in 67% test problems, (iii) it performs better than (1-1)-PAES in 56% test problems.

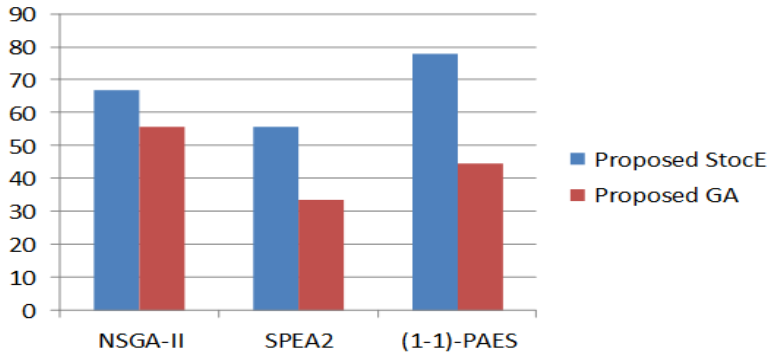
The comparison results show that given equal amount of memory and execution time, the Proposed StocE and GA-based algorithms are competitive in performance to the existing well-known algorithms on a set of difficult test problems. In many test problems, the proposed algorithms obtains a better results than one or more of the existing algorithms. Therefore, they should be utilized in solving practical problems in the field of multi-objective optimization.

7.4 Effect of Memory Size on the Solution Quality

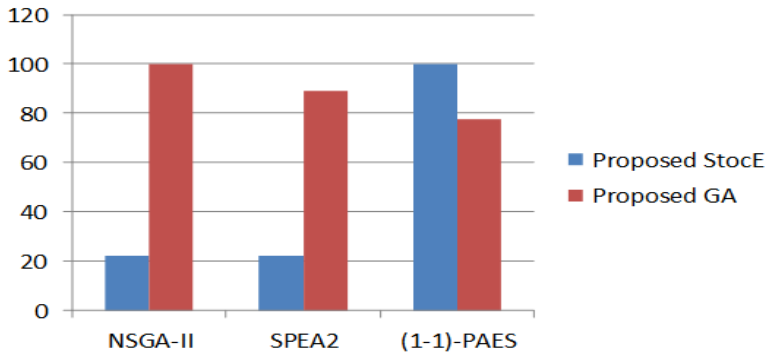
In proposed algorithms, the impact of memory size on the solution quality can be determined by executing the algorithms at a constant execution time but with different memory sizes. The proposed StocE and GA-based algorithms are implemented with the following parameter values: (i) execution time is set constant to 20 sec, (ii)$iter$= 5, (iii) m_t= 10, and (iv) num= 10. The value of m (i.e., size of the archive of pareto-optimal solutions) varies in different experiments. The proposed Off-Spring Non-Storing GA was implemented with the following values: (i) P_b was set to 0.75. The value of population (i.e., N_{GA}) varies in different experiments. In each experiment, the nine test problems are solved by using the proposed algorithms. Each test problem is solved for 10 times and the average values of the 10 trials is used in the analysis. The total memory requirement of the Proposed StocE-based algorithm is given by: $m + Num + 2$, since $Num = 10$, therefore, the expression becomes $m + 12$. The total memory requirement of the Proposed GA-based algorithm is equal to $2 + N_{GA}$. The values of the m and N_{GA} in different experiments in shown in Table 7.4.

The performance of the algorithms is measured using the same three performance metrics: (i) HVR, (ii) GD, and (iii) IGD. The nine test problems in the

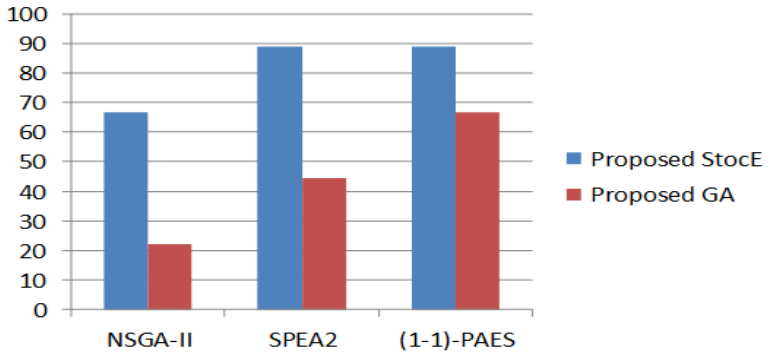
Figure 7.4: Results of the calculation of HVR, GD and IGD metrics on the test problems for the experiments in Case I.



(a) Percentage of experiments in which the proposed algorithms obtain a better HVR value than the exiting algorithms

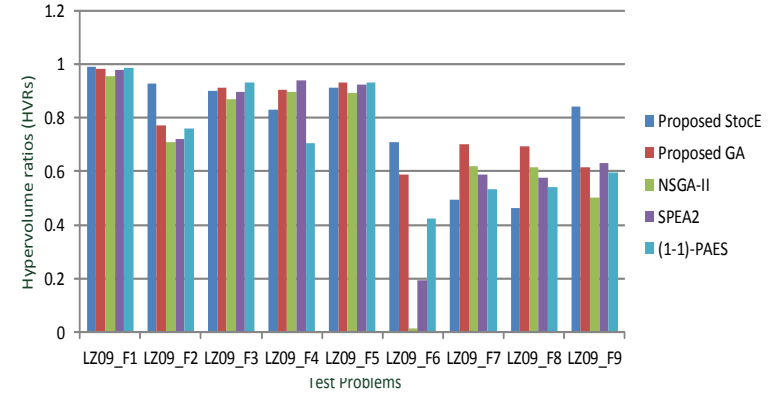


(b) Percentage of experiments in which the proposed algorithms obtain a better GD value than the exiting algorithms

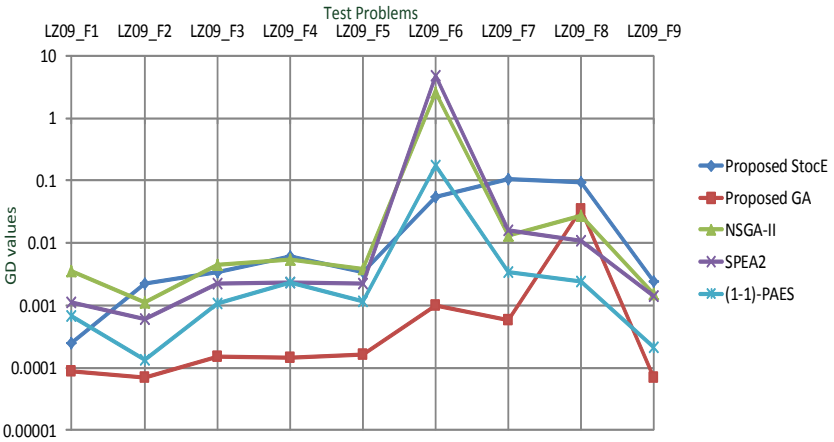


(c) Percentage of experiments in which the proposed algorithms obtain a better IGD value than the exiting algorithms

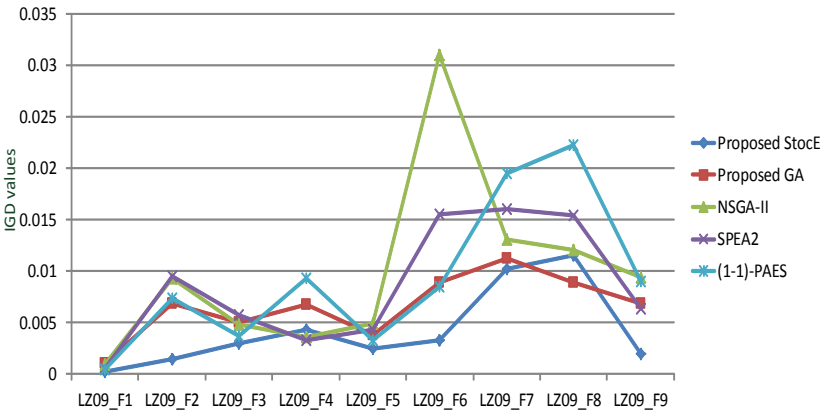
Figure 7.5: Summary of the results in Case I.



(a) Average HVR values.

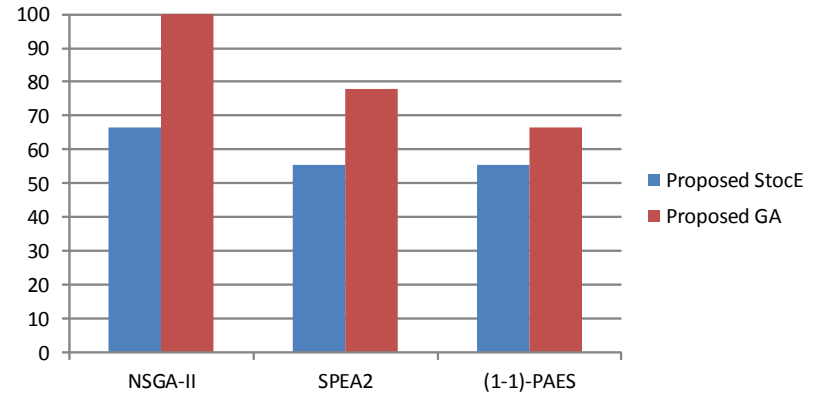


(b) Average GD values

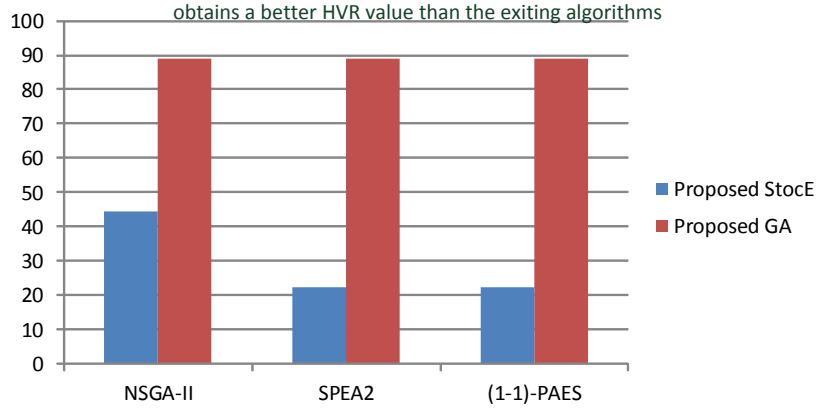


(c) Average IGD values

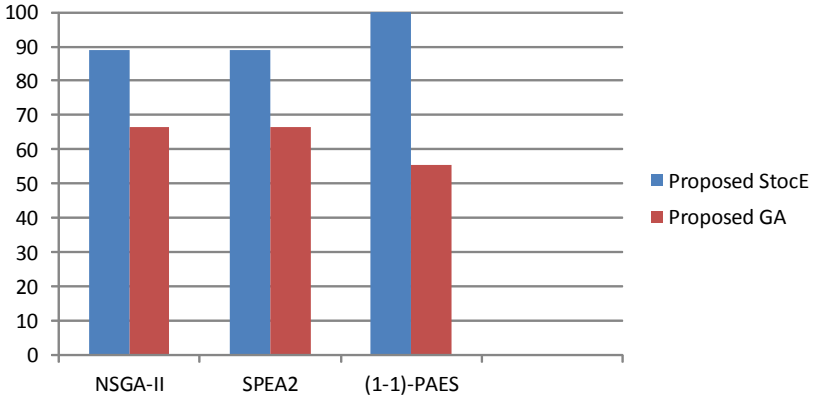
Figure 7.6: Results of the calculation of HVR, GD and IGD metrics on the test problems for the experiments in Case II.



(a) Percentage of experiments in which the proposed algorithms obtains a better HVR value than the exiting algorithms



(b) Percentage of experiments in which the proposed algorithms obtains a better GD value than the exiting algorithms



(c) Percentage of experiments in which the proposed algorithms obtains a better IGD value than the exiting algorithms

Figure 7.7: Summary of the results in Case II.

Table 7.4: Parameter values and memory sizes.

Algorithm	Parameter values	Total memory size
Proposed StocE	m= 38	50 solutions
Proposed GA	N_{GA}= 48	50 solutions
Proposed StocE	m= 88	100 solutions
Proposed GA	N_{GA}= 98	100 solutions
Proposed StocE	m= 188	200 solutions
Proposed GA	N_{GA}= 198	200 solutions
Proposed StocE	m= 288	300 solutions
Proposed GA	N_{GA}= 298	300 solutions
Proposed StocE	m= 388	400 solutions
Proposed GA	N_{GA}= 298	400 solutions

suite $LZ09$ were used. The performance of the Proposed StocE-based algorithm is shown in Fig. 7.8. The results show there at very small memory size (i.e. 50 solutions), the performance of the proposed algorithm is not good. However, in memory sizes of 100 solutions and onwards, the performance of the proposed algorithm is good and also remains nearly constant.

The performance of the Proposed Off-Spring Non-Storing GA is shown in Fig. 7.9. The results show that for the small size memory (i.e. 50 solutions) the performance of the algorithm is not good. But for memory sizes of 100 solutions and onwards, the performance of the proposed algorithm remains good. Therefore, the performance analysis shows that the proposed algorithms excessively memory sizes are not necessary to obtain goof quality solutions from the proposed algorithms.

7.5 Summary

This chapter shows the general forms of the proposed StocE and Off-spring non-storing GA-based algorithms. The general forms are not restricted to any particular MOP (Multi-Objective Optimization) problem, but can solve any type of MOP problem. The performance of the algorithms are shown in nine test MOP problems. Each test problem has a different and difficult pareto-optimal front. Therefore, the experiments on the test problems demonstrates the algorithm's capability to solve different types of MOP problems. The proposed algorithms were compared with NSGA-II and SPEA2. The proposed algorithms were found to be competitive with the existing algorithms at an equal memory requirement and execution time. The relationship between the memory size and solution quality is also analysed in detail. In most of the test problems, the solution quality increases with the memory size but excessive increase in the memory size does not improve performance.

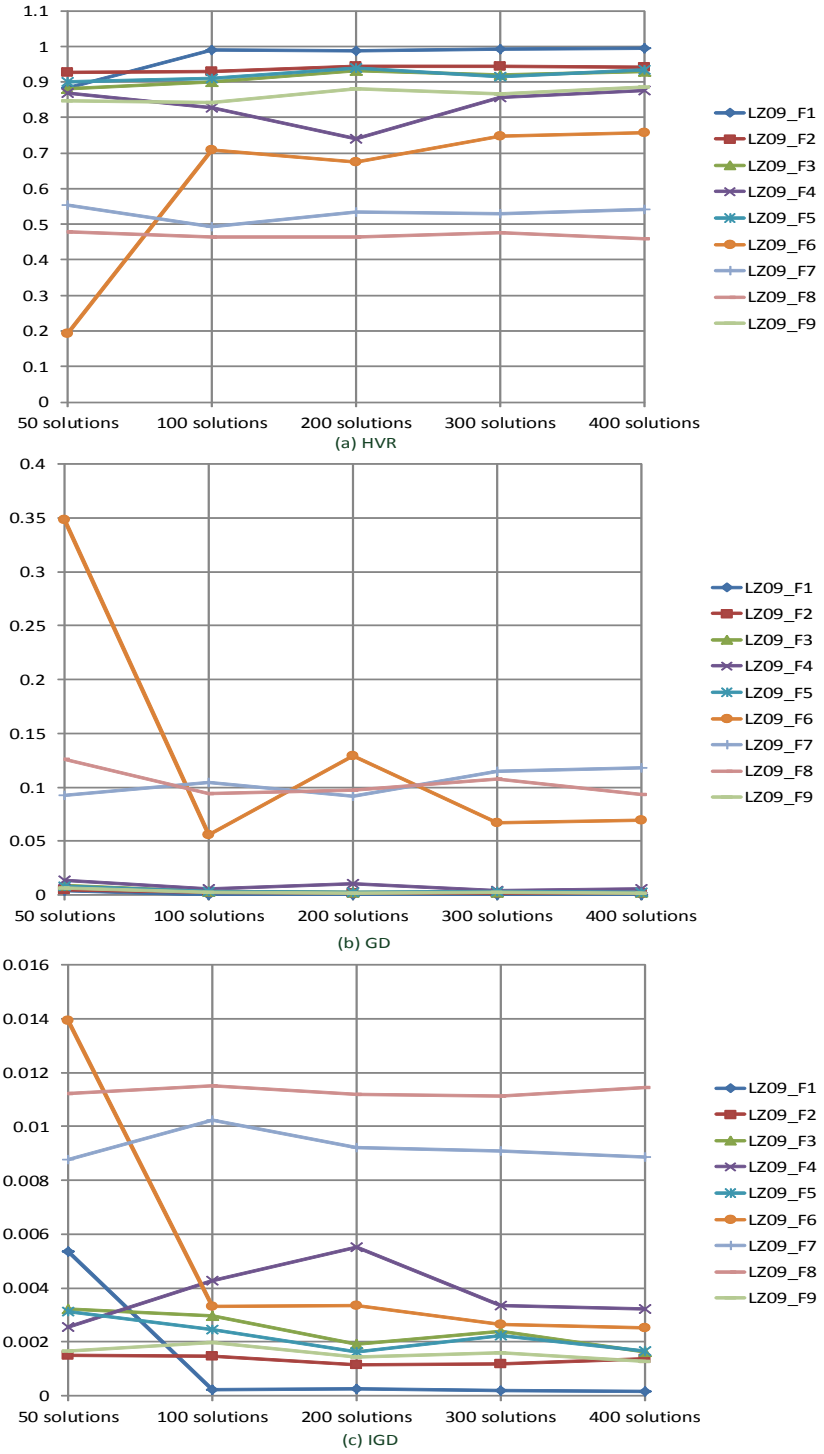
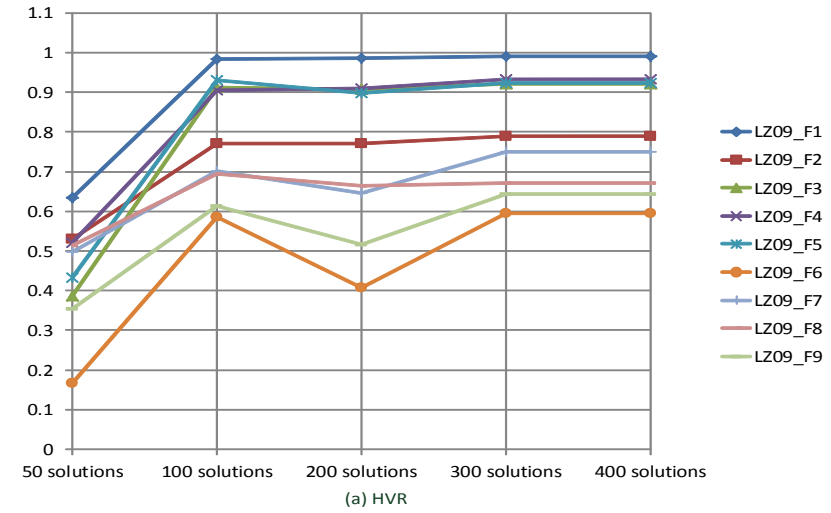
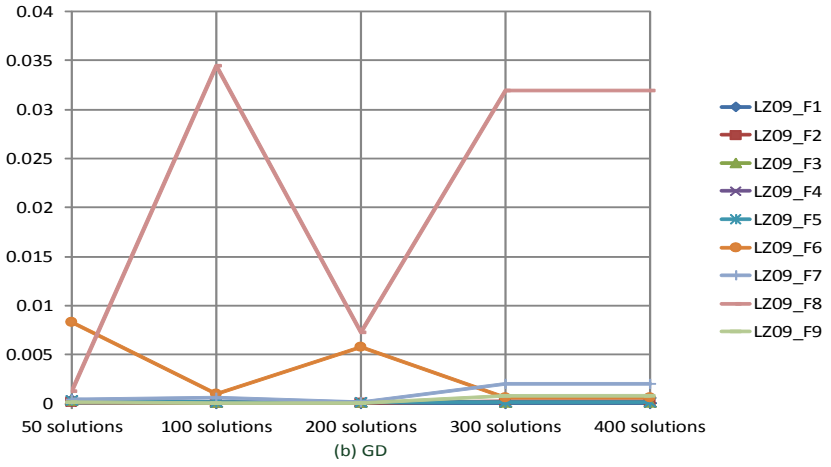


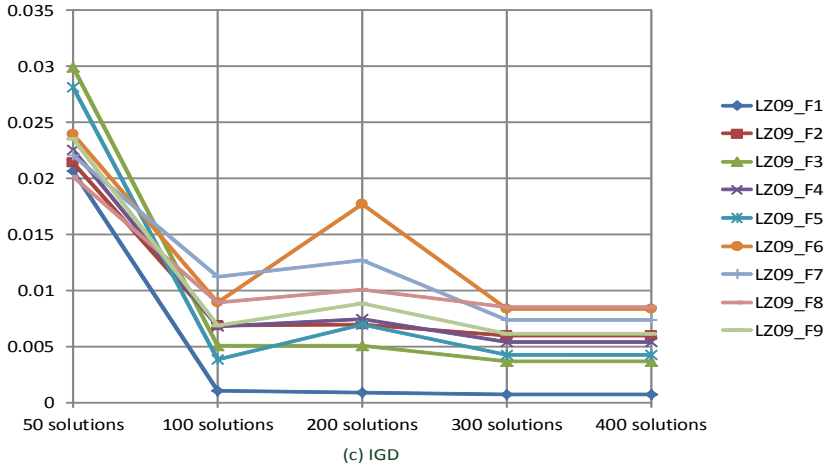
Figure 7.8: HVR, GD and IGD metric values of the Proposed StocE-based algorithm at different memory sizes.



(a) HVR



(b) GD



(c) IGD

Figure 7.9: HVR, GD and IGD metric values of the Proposed Off-Spring Non-Storing GA algorithm at different memory sizes.

Chapter 8

Congestion Awareness in case of Multi-Vehicles Problem

The vehicles can independently use the proposed multi-objective optimization algorithm to find most suitable paths to the destination. The values of parameters of the road network which are stored in the vehicle significantly impact the quality of the optimal paths. The advent of intelligent vehicles enable the vehicles to communicate with the other vehicles and with the road infrastructure (i.e., Road Side Units (RSUs)) using V2V and V2I types of communications [41, 42]. The detectors situated alongside the roads can monitor the traffic situation [43] and broadcast the latest traffic information to the vehicles using V2I communications. Therefore, the intelligent vehicles can frequently update the values of their network parameters and therefore, can find good quality optimal paths. The independent computation of optimal paths in the vehicles forms a sort of distributed system for determination of optimal paths.

A centralized system for traffic management is also proposed in recent past. In a centralized system, the road network information is maintained at a central control station. The vehicles send their requests to find optimal paths between their source and destination nodes to the central control station. The central control station finds the optimal paths for the vehicles and then sends the results back to the vehicles [44]. One major advantage of using the centralized system as compared to the distributed systems is that the vehicles can easily exchange information among each other in search of the optimal paths. A cooperation (or exchange of information) between the vehicles help in increasing the congestion awareness [42] of the vehicles and potential congestions in the road network can be avoided. The V2I communications enable the vehicles to communicate with their nearest control station. The range of V2I communication in any road network is restricted. The V2I communication uses IEEE 802.11p standard that has range of 1000 meters [42], but its range can be increased using multi-hop transmission through interconnected RSUs at expense of increase in communication time and cost. Therefore, a huge size road network can contain a large number of control stations. Each control station can perform minimization of potential congestion of the vehicles in its range.

Considering the benefits of using the control stations. The step to perform minimization of potential congestion in the road network can be also be added into the proposed method of optimal path selection. The vehicles can compute their optimal paths independently and then use the nearest or any reachable control station to reduce the potential congestion that could occur due to the optimal paths of the vehicles. Traffic flow (Q) on any edge is defined as the number of vehicles which pass through it within a certain amount of time. The traffic is also equal $Q = K \times V$, where K is the vehicles per Km and V is the maximum speed of the vehicles in terms of Km per hour [45]. However, until a certain threshold value of the density value, the flow increases. But, after the density passes the threshold value then the traffic flow start decreasing. Through, traffic management we want to keep the density of all edges below their threshold values.

This chapter is focussed on showing that a single EA can be used at the control station to serve the vehicles in its range. Small size random road networks are generated and then a first a large number of vehicles independently compute their optimal paths using the proposed algorithm. Then, the potential congestion is reduced by using a simple GA-based algorithm.

This chapter first describes the congestion minimization problem in a road network which is shared by a large number of vehicles, then shows the algorithm which is proposed to minimize the congestion.

8.1 Problem Description

Let us consider a road network $G(V, E)$, where V contains all intersections or nodes in the road network and E contains all edges that join the intersections. Any edge $e_i \in E$ has a starting node u and an ending node v and is associated with up to three properties: (i) $e_i.l$: which is the length of the edge, (ii) $e_i.S$: which is the average speed of the vehicles on it, (iii) $e_i.d$: is the average density of vehicles in terms of vehicles per km on the edge, and (iv) $e_i.D_{max}$ is the threshold on the maximum density value and the traffic flow on e_i start decreasing after this value. The edges on the actual density exceeds the D_{max} are called as congested. The average travelling time on any edge e_i can be computed as $T(e_i) = \frac{e_i.l}{e_i.S}$.

The road network is being shared by up to m vehicles which are represented as: $R = \{r_0, r_1, ..., r_{m-1}\}$. Any vehicle $r_j \in R$ has a pair of source and destination nodes (i.e., $r_j.s$ and $r_j.d$). The vehicles independently determine optimal paths between their source and destination nodes using any one of the algorithms which were described in Chapters 5 and 6. After the vehicles determine their optimal, then they should select an optimal path from their set of pareto-optimal solutions such that the congestion in the road network is minimized.

A vehicle $r_j \in R$ stores its pareto-optimal paths in the set $r_j.S$. Let U be a set in which all vehicles have selected up to one path, then $U = \{p_0, p_1, ..., p_{m-1}\}$, where $p_0 \in r_0.S$, $p_1 \in r_1.S$, and so on. The set U contains many different combinations, however, we want to select a combination in U that yields minimum congestion in the road network. The congestion in a road network can be determined as follows:

The potential increase in the density of any edge e_i is considered as the number of times e_i occurs in the paths which are selected by different vehicles

in U divided by the length of the edge. Therefore, the increases in the density of any edge is equal to its frequency in the set U divided by $e_i.l$ (i.e., $\frac{freq(U,e_i)}{e_i.l}$). The new density of the edge e_i becomes $e_i.d' = e_i.d + \frac{freq(U,e_i)}{e_i.l}$. If $e_i.d' \geq e_i.D_{max}$, then the edge e_i is called congested.

$$CE(U) = \{e_x \in U \text{ if } e_i.d' \geq e_i.D_{max}\} \tag{8.1}$$

The congestion in a network due to the paths in a set U is represented as $Congestion(U)$ and can be determined by computing the number of elements in $CE(U)$ (or cardinal number of $CE(U)$). The function $n(A)$ yields the cardinal number of the set A and the congestion can be determined as follows:

$$Congestion(U) = n(CE(U)) \tag{8.2}$$

8.2 Proposed Algorithm

The road network $G(V, E)$ is assumed to have a control station C which is accessible by all vehicles in the set R. The vehicles can use Vehicle to Infrastructure (V2I) communications [41] to access the central control station. The proposed congestion minimization algorithm is illustrated in Fig. 8.1. The first step shows that the set R contains a large number of vehicles (m) that want to travel between any two nodes in the network. In Step 2, each vehicle computes its MOSP paths and populate its set of pareto-optimal solutions using any one of the algorithm for solving the MOSP problem. In Step 3, all vehicles that belong to the set R send their pareto-optimal paths to the central control station. The proposed algorithm executes at the central control station is based on the game theoretic approach [47] such that each vehicle act as a player. The aim of the proposed algorithm is to select a path for vehicle such that the congestion in the road network is minimized. In the game theoretic approach, a strategy profile contains a pareto-optimal path selected for each vehicle in the set R. A strategy profile is represented as U. The game theoretic approach consists of an iterative process. In each iteration, each vehicle selects a path from its set of pareto-optimal solutions which minimizes the congestion while considering the paths selected by the other players. The iterative process continues until a Nash Equilibrium is achieved in which all players are satisfied with their selection and no player wants to switch from its previously selected path. When number of vehicles is equal to m, and the number of paths in the pareto-optimal set of any vehicle r_j is equal to n_j (where $n_j \geq 0$), then the total number of possible strategy profiles is equal to $\prod_{i=0}^{m-1} n_j$. For large values of m and n_j, the total number of possible strategy profile becomes very large and it may take very long time to determined the Nash Equilibrium. Therefore, a simple EA that consists of a mutation operation is used to find an approximate solution. In Step 5, the central control station informs the vehicles the path that should use from their set of pareto-optimal solutions in order to minimize the congestion in the road network.

Fig. 8.2 shows the proposed algorithm to find the strategy profile that minimizes the congestion. The inputs to the algorithm are the pareto-optimal paths of all vehicles and the output is a strategy profile (U) which minimizes the congestion in the road network. In line 1, U is initialized to null. In line 2,

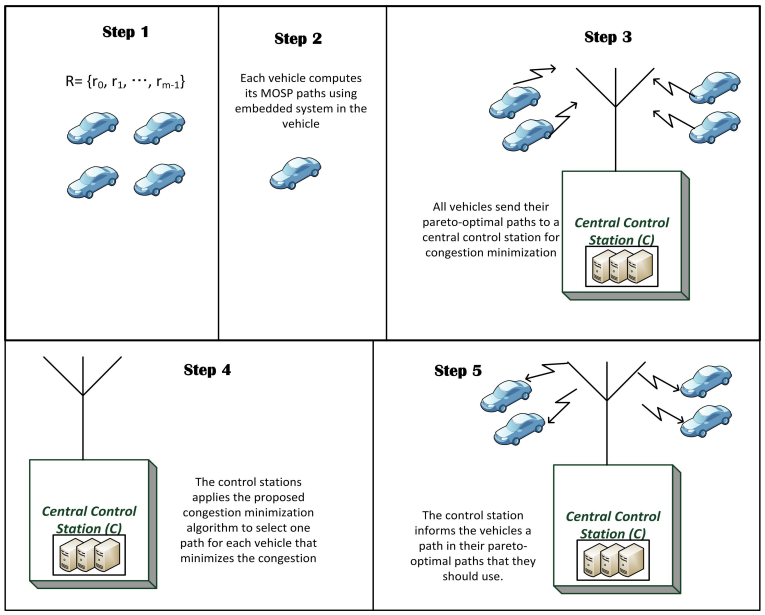


Figure 8.1: Illustration of the proposed congestion minimization method.

Input: $All_{paths} = \{r_0.S, r_1.S, ..., r_{m-1}.S\}$
Output: $U = \{p_0, p_1, ..., p_{m-1}\}$, s.t. $p_j \in r_j.S$.

U= null
Initialize(U)
while stopping criteria not reached **do**
  $c_1 = Congestion(U)$;
  U'= Mutation(U)
  $c_2 = Congestion(U')$
  if $c_2 < c_1$ **then**
    $U = U'$
  end if
end while
return U

Figure 8.2: Proposed algorithm for congestion minimization.

Input: $All_{paths} = \{r_0.S, r_1.S, ..., r_{m-1}.S\}$
Output: U
1: U= null
2: **for** i= 0 to m-1 **do**
3: p'= randomly select a path from $r_i.S$
4: $U = U \cup p'$
5: **end for**
6: **return** U

Figure 8.3: Method to initialize a random solution in U

Input: $All_{paths} = \{r_0.S, r_1.S, ..., r_{m-1}.S\}$, U, $z(s.t.z \in 0, 1 \in R^+)$
Output: U: after the mutation operation
1: M_{num} : a random number between 0 and $z \times m$.
2: **for** i= 0 to M_{num} **do**
3: I= a random number between 0 and $m - 1$
4: p'= randomly select a path from $r_I.S$
5: $U[I] = p'$
6: **end for**
7: **return** U

Figure 8.4: Method to perform mutation operation on a strategy profile U

U is initialize to a random strategy profile. Lines 3 to 10 contain a while loop, in which first mutation operation is applied to U to form a new strategy profile U'. The costs of U and U' are determined and if the cost of U' is lesser than the cost of U, then U is replaced by U. The stopping criterion of the algorithm can be the maximum elapsed time or maximum iterations.

Fig. 8.3 shows the method to initialize a random solution in U. In line 1, U is initialized to null. In the for loop between lines 2 and 5, p' is a randomly selected path from the set of pareto-optimal paths of the vehicle which is selected in that iteration (i.e. r_i). The selected path is inserted in the set U. Using the for loop, the paths for all vehicles are selected and stored in U.

Fig. 8.4 shows the mutation operation that is used to make random changes in the strategy profile U. The inputs are the strategy profiles of all vehicles; m, which is the number of vehicles in the set network; z, which is a positive real number between 0 and 1 and is used to decide the number of vehicles whose paths should be altered in the mutation operation (M_{num}). Using the for loop, the paths of up to M_{num} vehicles are replaced by another randomly selected paths from their set of pareto-optimal paths.

8.3 Simulation Results

The proposed congestion minimization algorithm was implemented using C# in Visual Studio 2010. Several small size random graphs were generated using a random graph generation tool [46]. Table 8.1 shows the details of the random graphs. The values of network parameters are assigned as follows: (i) $e_i.l$, i.e., length of the edge is assigned between $[20, 50]$ Km, (ii) $e_i.S$, i.e., average speed of the vehicle is assigned randomly between $[50, 100] km/hour$. (iii) $e_i.d$

Table 8.1: Characteristics of graphs.

Graph	Number of nodes	Number of edges
$G0$	600	2000
$G1$	650	2000
$G2$	750	2500

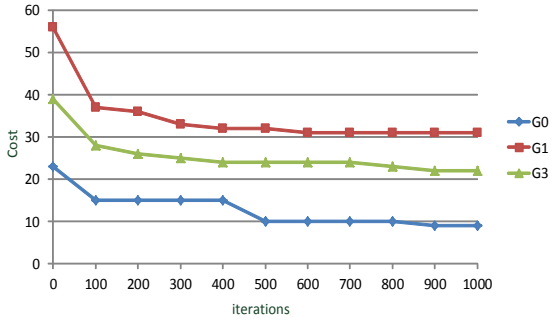


Figure 8.5: Illustration of the gradual decrease in the congestion cost as iterations proceeds.

and $e_i.D_{max}$ which are the average density of vehicles and maximum density of vehicles on the edge e_i are assigned as follows. The average traffic flow capacity of a one lane of motorway is 1400 $\frac{vehicles}{hour}$ with an average speed of 60 $\frac{Km}{h}$. The roads are assumed to be carry traffic which is 90% of their maximum capacity. Therefore,the value of $e_i.d$ are assigned equal to $0.90 \times e_i.D_{max}$.

The vehicles obtained their sets of possible strategies through successively applying the random path generation method which is shown in Fig. 4.1. Each vehicle generates as many alternate paths for its set of possible strategies as possible within a time interval of 800 msec. In small size networks, many pareto-optimal paths may not occur between the source and destination nodes. Therefore, method of generation of alternate random solutions is used instead of the optimization algorithms. The actual road networks should use the optimization algorithms. The congestion due to the paths selected by different vehicles and due to the overlapping among them was reduced using our proposed algorithm. The number of vehicles in the road network i.e., m is assigned a value equal to 5000. The source and destination nodes for the vehicles were randomly chosen in the network.

Our proposed algorithm for minimizing the congestion was implemented with the following parameter values: The stopping criterion was set to 1000 iterations; the value of z in the mutation operation was set to 0.15. Figs. 8.5 shows the performance of the proposed algorithm and the congestion cost values

Table 8.2: Results of the congestion reduction algorithm

Graph	Cost w/o optimization	Cost w/ optimization	% decrease in value
G0	23	9	61%
G1	56	31	45%
G2	39	22	44%

are plotted after every 100 iterations. The plot shows gradual decrease in the cost value in all graphs.

Table 8.2 shows the results of the congestion minimization algorithm. The first column indicates the network, the second column indicates the Cost or number of edges that can potentially go into congestion if no cooperation and optimization is performed in selecting the paths for the vehicles. The second last column shows the number of edges on which congestion may still occur after applying the proposed method. The last column shows the percentage reduction in the number of edges from w/o optimization case to w/ optimization case.

Therefore, based on the results, we can suggest that the proposed optimization algorithms can be used in a traffic management system. The vehicles can find their pareto-optimal paths using their embedded systems then use a nearby control station to minimize the potential congestion due to the other vehicles.

8.4 Summary

This chapter considers the problem of congestion awareness. Congestion awareness refers to reducing the potential congestion which may occur due to the vehicles which are planning to start their journeys. in the proposed method, the vehicles first finds their optimal-paths using their embedded systems. The vehicles after finding their optimal-paths sends their optimal paths to a reachable control station through V2I communications. The control station collects optimal paths of many vehicle and perform minimization of potential congestion which may occur due to the paths selected by different vehicles. The control station selects one path for each vehicle such that minimum number of edges in the road network can go into congestion by the vehicles which are about to their journeys. The control station uses a simple GA to optimization the congestion cost.

The experiments were performed on several randomly generated small size networks. The results show that in all networks, the congestion costs are reduced over their initial value. Therefore, the simulations show that it is benefical to use a control station to reduce potential occurrence of congestion in the network.

Chapter 9

Conclusion

Multi-Objective Shortest Path (MOSP) problem often occurs in computer and transportation networks. When the MOSP problem has two or more objectives then it becomes an NP-hard problem. Different approaches can be used to solve the MOSP problem that includes: (1) Evolutionary algorithms (EAs), (ii) Polynomial time approximation algorithms (PTAAs), and (iii) Polynomial time algorithms (PTEAs). PTEAs can exactly solve the MOSP problem but they are useful in case of small size networks only. In huge size networks, the MOSP problem should be solved using EAs or PTAAs. EAs have the advantage that they can be described independently to any particular problem. Therefore, an EA that can solve the MOSP problem remains useful for the other optimization problems. Generally, PTAAs are specific to one or several classes of problems. The existing EAs can be divided into two classes: (i) Population-based algorithms, and (ii) Single-solution based algorithm. The population-based EAs can yield good quality solutions but they require large memory sizes. Single-solution based EAs, on the other-hand, are memory-efficient but require long computation time to yield good quality solutions. Therefore, this work aimed to propose two new EAs that can overcome the problems of memory-size and solution quality of the existing EAs. The ability of the proposed EAs to solve general multi-optimization problem was also demonstrated through evaluating them on several test problems that are quite different from the MOSP problem.

We proposed an Stochastic Evolution (StocE)-based EA and an Off-Spring Non-Storing Genetic Algorithm (GA). The standard StocE algorithm improves the bad characteristics in a solution through applying the Perturb operation. The Perturb operation determines the suitability to remain in the next generation of different characteristics and improves a least suitable characteristic in the solution. The Proposed StocE-based algorithm consists of an innovative Perturb operation. It considers different sub-paths (or sub-portions) in a solution as its characteristics. The suitability of the sub-paths is measured in terms of their objective function values and their sizes. The sub-path that has higher values of objective functions and smaller size than the other sub-paths is considered to be least suitable to remain in the next generation. One of the least suitable sub-path is replaced by another randomly generated sub-path. The algorithm maintains an archive of pareto-optimal solutions. It requires memory which is slightly more than the existing single-solution based algorithms and much lesser than the population-based algorithms.

The proposed Off-Spring Non-Storing GA is memory efficient than the existing GA and its different variants (e.g. NSGA-II, MicroGA, etc). In the existing GAs, if the population size is N solutions, then they also create N children chromosomes. Therefore, their total memory size is $2N$ solutions. In the proposed GA-based algorithm, the children chromosomes are conditionally stored at the memory locations of their parent chromosomes. Therefore, additional memory is not required to store the children. The proposed GA-based algorithm requires a total memory of $N+1$ solutions instead on $2N$. In the proposed GA-based, a GA operation (i.e., crossover or mutation) should be selected for each chromosome in the population. The children which is created after applying the GA operator replaces the parent chromosome based on the following conditions: (i) If the parent is a non-dominated solution in the population and the children dominates the parent chromosome, or (ii) if the parent is dominated by another solution in the population, and the children chromosome is not dominated by the parent chromosome.

The performance of the Proposed algorithms was evaluated through comparing their performance against some well-known multi-objective optimization algorithms. The existing algorithms included Non-dominated Sorting Genetic Algorithm-II (NSGA-II), Micro-Genetic Algorithm (Micro-GA), (1-1)-Pareto Archived Evolutionary Strategy ((1-1)-PAES) algorithm, Multi-objective Simulated Annealing (MOSA), and a straight-forward StocE. The NSGA-II and Micro-GA are population-based algorithms and the remaining algorithms are single-solution-based algorithms. The experiments were conducted on the road networks of some states and cities in the United States. The road networks have sizes. The quality of the pareto-optimal set of any algorithm is measured by calculating its Hypervolume (HV) value. The HV is the space occupied by the pareto-optimal set in the solution space and measures the quality and diversity of solutions in the pareto-optimal set. The algorithms that obtain a higher HV value have better performance than the other algorithms. Each experiments is repeated for size time or in other words each experiments has six trials. The average HV of the six trials was used in the comparison. The comparison results show that the proposed algorithms in all road networks outperformed the existing single-solution-based algorithms and also performs better than the population-based algorithms in some experiments. Therefore, the proposed algorithms have successfully solved the MOSP problem with the solution quality which is better than the existing single-solution-based algorithms and with the memory-size which is lesser than the existing population-based algorithms.

The proposed algorithms were also generalized to solve the general multi-objective optimization problems. Complex test problems with known true pareto-fronts were selected from recent literature. The test problems were solved using the proposed algorithms as well as using existing algorithms. The existing algorithms include: NSGA-II, Strength Pareto Evolutionary Algorithm 2 (SPEA2), and (1-1)-PAES. The NSGA-II and SPEA2 are population-based algorithms and (1-1)-PAES is a single-solution-based algorithm. The performance of the algorithms was measured by using up-to three different performance metrics. The performance metrics include: (i) Hypervolume Ratio (HVR), which is the ratio between the HV of the pareto-front obtained from the algorithm to the HV of the true pareto-front. (ii) Generational Distance (GD), which is the average distance of the pareto-front of any algorithm from the actual pareto-front. (iii) Inverse Generational Distance (IGD), which is the distance of the actual

pareto-front from the pareto-front of any algorithm. The GD value indicate the closeness of the obtained pareto-front from the actual pareto-front. A better IGD value indicates that the obtained pareto-front is closer and uniformly spread along the true pareto-front. The results show that given equal amount of memory and execution time, the proposed algorithms are competitive to the existing algorithms. The effect of memory size on the proposed algorithms was also studied. It is found that at a constant execution time, the proposed algorithms start yielding good quality solutions after a certain memory size (which was not a large value). It was also noticed that a continued increase in the memory size do not improve the solution quality.

At any time, in a road network, many vehicles are about to start their journeys and are finding optimal paths between their source and destination nodes. Congestion awareness refers to detecting the potential congestion which may occur due to the paths selected by the vehicles that are about to start their journeys. The control stations can be built in the road network that can communicate with the vehicles that lie within their range using V2I communications. The control station can perform minimization of potential congestion which could occur due to the paths selected by the different vehicles. The vehicles send their sets of pareto-optimal solutions to a reachable control station. A control station accepts input from a large number of vehicles and then use a simple EA to find one path per vehicle, such that the selected paths minimizes the potential congestion in the road network. The experiments were formed on small size random networks and the results show that a simple EA is efficient is reducing the potential congestion in a road network.

The experimental results show that the proposed optimization algorithms are suitable to solve the MOSP problem in embedded systems. They can yield a solution quality which is better than the existing single-solution-based algorithms. The proposed optimization algorithms can also become part of a traffic management system. The proposed algorithm can also be used to solve any other multi-objective optimization problem.

Published Articles

From the mentioned research work, the authors have published several articles in different international journals and conferences.

Journal Articles

1. **Umair F. Siddiqi**, Yoichi Shiraishi, Sadiq M. Sait, "Multi-Objective Optimal Path Selection in the Electric Vehicles," Artificial Life and Robotics: Vol. 17, No. 1, pp. 113-122, 2012.

2. **Umair F. Siddiqi**, Yoichi Shiraishi, and Sadiq M. Sait, "Memory Efficient Genetic Algorithm for Path Optimization in Embedded Systems," Transactions on Mathematical Modeling and Applications, Information Processing Society of Japan, 2013. (to appear)

Conference Articles

1. **Umair F. Siddiqi**, Yoichi Shiraishi, Mona Dahb, Sadiq M. Sait, "Finding Multi-Objective Shortest Paths using Memory Efficient Stochastic Evolution based Algorithm," The Third International Conference on Networking and Computing, Okinawa, Japan, December 5-7, 2012.

2. **Umair F. Siddiqi**, Yoichi Shiraishi, Sadiq M. Sait, "Multi-Objective Optimal Path Selection in the Electric Vehicles," 17th International Conference on Artificial Life and Robotics, Beppu, Oita, Japan, 19-21 Jan. 2012,

3. **Umair F. Siddiqi**, Yoichi Shirashi, Sadiq M. Sait, "Multi-Constrained Route Optimization for Electric Vehicles (EVs) using Particle Swarm Optimization (PSO)," International Conference on Intelligent Systems Design and Applications (ISDA) 2011, pp. 391-396, Cordoba, Spain, 22-24 Nov. 2011,.

4. **Umair F. Siddiqi**, Yoichi Shirashi, Sadiq M. Sait, "Multi-Constrained Route Optimization for Electric Vehicles using SimE," International Conference on Soft Computing and Pattern Recognition (SoCPaR) 2011, pp.

376-383, 14-16 Oct. 2011, Dalian, China.

5. **Umair F. Siddiqi**, Yoichi Shiraishi, Mona A. El. Dahb, and Sadiq M. Sait, "Simulated Evolution (SimE) based Embedded System Synthesis Algorithm for Electric Circuit Units (ECUs)," 10th International Conference on Adaptive and Natural Computing Algorithms (ICCANGA) 2011, Part 1, pp. 400-409, 2011.

6. **Umair F. Siddiqi**, Yoichi Shiraishi, Mona A. El-Dahb and Sadiq M. Sait, "Embedded Systems Synthesis Using Simulated Evolution (SimE) with ECU-Specific Optimization," International Conference on Computer Mathematics and Natural Computing, ICCMNC 2011, Year 7, Issue 74, pp. 42-46, Penang, Malaysia, 22-24 February 2011.

Bibliography

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, Introduction to Algorithms, 2^{nd} edition, MIT Press, 2001.

[2] Zbigniew Tarapata, "Selected Multicriteria Shortest Path Problems: An Analysis of Complexity, Models and Adoption of Standard Algorithms," Int. J. Appl. Math. Comput. Sci., Vol. 17, No. 2, pp. 269-287, 2007.

[3] M.R. Garey and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman and Co., 1997.

[4] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," IEEE Trans. Evolutionary Computation, Vol. 6, No. 2, pp. 182- 197, 2002.

[5] L. dos Santos Coelho and P. Alotto, "Multiobjective Electromagnetic Optimization Based on a Nondominated Sorting Genetic Approach with a Chaotic Crossover Operator," IEEE Trans. Magnetics, Vol. 44, No. 6, pp. 1078-1081, 2008.

[6] Y. Kumar, B. Das, and J. Sharma, "Service restortation in distribution system using non-dominated sorting genetic algorithm," Electric Power Syst. Res., vol. 76, pp. 768-777, 2006.

[7] Marcus Vinicius Carvalho da Silva, Nadia Nedjah and Luiza de Macedo Mourelle, "Optimal application mapping on NoC infrastructure using NSGA-II and Micro-GA," International Conference on Intelligent Engineering Systems (INES), pp. 83-88, 2009.

[8] Zbigniew Tarapata, Selected Multicriteria Shortest Path Problems: An Analysis of Complexity, Models and Adoption of Standard Algorithms, Int. J. Appl. Math. Comput. Sci., Vol. 17, No. 2, pp. 269-287, 2007.

[9] Christina Hallam, K. J. Harrison, and J. A. Ward, "A Multiobjective Optimal Path Algorithm, Digital Signal Processing," vol. 11, pp. 133-143, 2001.

[10] George Tsaggouris and Christos Zaroliagis, "Multiobjective Optimization: Improved FPTAS for Shortest Paths and Non-Linear Objectives with Applications," Journal Theory of Computer Systems, Vol. 45, No. 1, pp. 162-186, 2009.

[11] Christian Horoba, "Exploring the Runtime of an Evolutionary Algorithm for the Multi-Objective Shortest Path Problem," Evolutionary Computation, Vol. 18, No. 3, pp. 357-381, 2010

[12] Andrey Gunichev, Srikanta Bedathur, Stephan Seufert, and Gerhard Weikum, "Fast and Accurate Estimation of Shortest Paths in Large Graphs," Proc. 19^{th} ACM International Conference on Information and Knowledge Management, Toronto, Canada, pp. 499-508, 2010.

[13] E. Martins and J. Santos, "The labelling algorithm for the multiobjective shortest path problem," Departamento de Matematica, Universidade de Coimbra, TR-99/005, Portugal, 1999.

[14] Chang Wook Ahn, and R. S. Ramakrishna, "A Genetic Algorithm for Shortest Path Routing Problem and the Sizing of Populations," IEEE Trans. Evolutionary Computation, Vol. 6. No. 6, (2002)

[15] Sai Ho Yeung and Kim Fung Man, "Multiobjective Optimization," IEEE microwave magazine, pp. 120-133, October, 2011.

[16] S.M. Sait and H. Youssef, Iterative Computer Algorithms with Applications in Engineering, IEEE Computer Society Press, 1999.

[17] Youssef G. Saab and Vasant B. Rao, "Stochastic Evolution: A Fast Effective Heuristic for Some Generic Layout Problems," 27^{th} Design Automation Conference, pp. 36-31, 24-28 June 1990.

[18] J. H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, Michigan, 1975.

[19] Ralph Michael Kling and Prithviraj Banerjee, "Concurrent ESP: A placement algorithm for execution on distributed processors," Proc. IEEE International Conference on Computer-Aided Design, pp. 354-357, 1987.

[20] Joshua D. Knowles and David W. Corne, "Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy," Evolutionary Computation, vol. 8, No. 2, pp. 149-172, 2000.

[21] Carlos A. Coello Coello and Gregorio Toscano Pulido, "A Micro-Genetic Algorithm for Multiobjective Optimization," Departamento de Matematica, Universidade de Coimbra, TR-99/005, Portugal, 1999.

[22] Kevin I. Smith, Richard M. Everson, Jonathan E. Fieldsend, Chris Murphy and Rashmi Misra, "Dominance-Based Multiobjective Simulated Annealing," IEEE Trans. Evolutionary Computation, Vol. 12, No. 3, pp. 323- 342, 2008.

[23] J. Teo, P. Anthony, Jia Hui Ong, "Neural network ensembles for video game AI using evolutionary multi-objective optimization," 11^{th} International Conference on Hybrid Intelligent Systems (HIS) 2011, pp. 605-510, Malaysia, pp. 605-510, 5-8 Dec. 2011.

[24] Teodoro C. Bora, Luiz Lebensztajn, and Leandro Dos S. Coelho, "Non-Dominated Sorting Genetic Algorithm Based on Reinforcement Learning to Optimization of Broad-Band Reflector Antennas Satellite," IEEE Trans. Magnetics, Vol. 48, No. 2, pp. 767-770, 2012.

[25] P. Ngatchou, Anahita Zarei, M.A. El-Sharkawi, "Pareto Multi Objective Optimization," Proc. 13^{th} Intelligent Systems Application to Power System, pp. 84-91, 2005..

[26] Johannes M. Bader, "Hypervolume-Based Search for Multiobjective Optimization: Theory and Methods," Ph.D. dissertation, Swiss Federal Inst. Technology (ETH) Zurich, Switzerland, (2009).

[27] Joshua Knowles, "ParEGO: A Hybrid Algorithm With On-Line Landscape Approximation for Expensive Multiobjective Optimization Problem," IEEE Trans. Evolutionary Computation, Vol. 10 No. 1, pp. 50-66, 2005.

[28] Carlos M. Fonseca, Lus Paquete, and Manuel Lpez-Ibez, "An Improved Dimension - Sweep Algorithm for the Hypervolume Indicator," 2006 IEEE Congress on Evolutionary Computation (CEC'06), pp. 1157-1163, 2006.

[29] L. White, P. Hingston, L. Barone, and S. Husband, "A Faster Algorithm for Calculating Hypervolume," IEEE Trans. Evolutionary Computation, Vol. 10. No. 1, pp. 29-38, 2006.

[30] http://www.dis.uniroma1.it/challenge9/download.shtml

[31] Murat Yilmaz, Veysel T. Buyukdegirmenci, and Philip T. Krein, " General Design Requirements and Analysis of Roadbed Inductive Power Transfer System for Dynamic Electric Vehicle Charging," 2012 IEEE Transportation Electrification Conference and Expo, pp. 1-6, 18-20 June 2012.

[32] Swagat Chopra and Pavol Bauer, "Driving Range Extension of EV with On-Road Contactless Power Transfer- A Case Study," IEEE Trans. Industrial Electronics, vol. 60, No. 1, January, 2013.

[33] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," Complex Systems, No. 9, pp. 115-148, 1995.

[34] K. Deb and M. Goyal, "A combined genetic adaptive search (GeneAS) for engineering design," Computer Sciences and Informatics, Vol. 26, No. 4, pp. 30-45, 1996.

[35] http://www.nd.com/products/genetic/mutation.htm

[36] Juan J. Durillo and Antonio J. Nebro, "jMetal: A Java framework for multi-objective optimization," Advances in Engineering Software, vol. 42, pp. 760-771, 2011.

[37] http://jmetal.sourceforge.net/

[38] H. Li and Q. Zhang. Multiobjective Optimization Problems with Complicated Pareto Sets, MOEA/D and NSGA-II, IEEE Trans on Evolutionary Computation, vol. 2, No. 12, pp. 284-302, 2009.

[39] E. Zitzler and M. Laumanns and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm," Technical Report, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, no. 103, 2001.

[40] http://www.nvidia.com/object/cuda_home_new.html

[41] Erik Strom, Hannes Hartenstein, Paolo Santi, Werner Wielsbeck, "Vehicular Communications: Ubiquitous Networks for Sustainable Mobility," Proc. of the IEEE, Vol. 98, No. 7, pp. 1111-1112, July 2010.

[42] Miguel Sepulcre, Jens Mittag, Paolo Santi, Hannes Hartenstein, and Javier Gozalvez, "Congestion and Awareness Control in Cooperative Vehicular Systems," Proceedings of IEEE, vol. 99, no. 7, pp. 1260-1279, 2011.

[43] David H. Roper and Goro Endo, "Advanced Traffic Management in California," IEEE Trans. Vehicular Technology, vol. 40, no. 1, pp. 152-158, 1991.

[44] Yanyan Chen, Michael G. H. Bell, and Klaus Bogenberger, "Reliable Pretrip Multipath Planning and Dynamic Adaptation for a Centralized Road Navigation System," IEEE Trans. Intelligent Transportation Systems, vol. 8, no. 1, pp. 14-20, 2007.

[45] Wikipedia $http://en.wikipedia.org/wiki/Traffic\_engineering\_(transportation)$.

[46] Fabien Viger, Matthieu Latapy, Efficient and Simple Generation of Random Simple Connected Graphs with Prescribed Degree Sequence, 11^{th} Conference of Computing and Combinatorics (COCOON 2005), pp. 440-449, (2005).

[47] Jorgen W. Weibull, Evolutionary Game Theory, The MIT Press, 1995.