ADAPTIVE DIFFERENTIAL EVOLUTION AND ITS APPLICATION TO MACHINE VISION

A Dissertation
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Deepak Dawar

In Partial Fulfillment of the Requirements
for the Degree of
DOCTOR OF PHILOSOPHY

Major Department:
Computer Science

July 2016

Fargo, North Dakota

NORTH DAKOTA STATE UNIVERSITY

Graduate School

Title

ADAPTIVE DIFFERENTIAL EVOLUTION AND ITS APPLICATION TO MACHINE VISION

By

Deepak Dawar

The supervisory committee certifies that this dissertation complies with North Dakota State University's regulations and meets the accepted standards for the degree of

DOCTOR OF PHILOSOPHY

SUPERVISORY COMMITTEE:

Dr. Simone A. Ludwig
Chair

Dr. Dean Knudson

Dr. Anne Denton

Dr. Cheryl J. Wachenheim

Approved:

July 7, 2016
Date

Brian Slator
Department Chair

ABSTRACT

Over recent years, Evolutionary Algorithms (EA) have emerged as a practical approach for solving hard optimization problems ubiquitously presented in real life. The inherent advantage of EA over other types of numerical optimization methods lies in the fact that they require much less or no prior knowledge of the objective function. Differential Evolution (DE) has emerged as a highly competitive and powerful real parameter optimizer in the diverse community of evolutionary algorithms.

The study of this dissertation is focused on two main approaches. The first approach focuses on studying and improving DE by creating its variants that aim at altering/adapting its control parameters and mutation strategies during the course of the search. The performance of DE depends largely upon the mutation strategy used, its control parameters namely the scale factor F, the crossover rate Cr, and the population size NP, and is quite sensitive to their appropriate settings. A simple and effective technique that alters F in stages, first through random perturbations and then through the application of an annealing schedule, is proposed. After that, the impact and efficacy of adapting mutation strategies with or without adapting the control parameters is investigated.

The second approach is concerned with the application side of DE which is used as an optimizer either as the primary algorithm or as a surrogate to improve the performance of the overall system. The focus area is video based vehicle classification. A DE based vehicle classification system is proposed. The system in its essence, aims to classify a vehicle, based on the number of circles (axles) in an image using Hough Transform which is a popular parameter based feature detection method. Differential Evolution (DE) is coupled with Hough Transform to improve the overall accuracy of the classification system. DE is further employed as an optimizer in an extension of the previous vehicle detector and classifier. This system has a novel appearance based model utilizing pixel color information and is capable of classifying multi-lane moving vehicles into seven different classes. Five different variants of DE on varied videos are tested, and a performance profile of all the variants is provided.

ACKNOWLEDGEMENTS

I would like to convey my earnestness to my primary advisor and mentor, Dr. Simone A. Ludwig for her enthusiastic and encouraging support during the pursuit of my research. This work is as much as a product of her guidance as of my own research. Not only did she guide me through the dissertation process, but also taught me the overall process of performing quality research. I cannot thank her enough for the supervision, encouragement and inspiration in developing the theory, conducting the experiment, publishing papers, and writing this dissertation. She has shown me the joys of research, and importance of life, and for this I am indebted to her.

I would like to also thank Dr. Dean Knudson and Dr. Anne Denton for serving on my graduate committee. I enjoyed discussing many aspects of this research with them and look forward to future discussions. They dedicated their valuable time on supporting my graduate study, and helping on my professional developments.

I would also like to thank Dr. Cheryl Wachenheim. With all of the other commitments to the university, she dedicated time to take part in my dissertation research, and provided valuable insights into the completion of my dissertation. She is one of the most helpful human beings I have ever known. It has been a privilege knowing her.

Finally, I would like to thank my family and friends for their support, understanding, and encouragement. I certainly wasn't the easiest person to be around during the last year while I worked on this research, and my dissertation.

DEDICATION

To everyone who is yearning to learn and grow

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

Challenging real world optimization problems are ubiquitous in scientific and engineering domains. Complexity of the problem notwithstanding, the problem's objective function may also be non-continuous, and non-differentiable adding to the overall difficulty and negotiability of the search space. Darwinian inspired evolutionary theories like social group behavior and foraging strategies, to name a few, have attracted the attention of researchers for tackling hard and complex optimization problems. The outcome of this research effort are nature inspired algorithms. These algorithms can be broadly classified into two categories: evolutionary computing methods, and swarm intelligence algorithms, both of which employ their own set of control parameters.

The underlying idea behind evolutionary algorithms is the iterative fitness improvement of a population of individuals (solutions), through natural selection. An iteration generally involves, producing new individuals through a series of mutations and recombinations, gradually removing lesser fit individuals from the population and replacing them with newly generated individuals if their fitness proves to be better than the individuals they were generated to replace [1].

The operation of swarm intelligence algorithms may be behaviorally characterized as a decentralized swarm searching for optimal food sources (solutions) [2]. The direction of individual search is influenced by the current location of the individual, its best location ever, and the location of the best individual in the whole swarm.

The performance of both these classes of algorithms is quite sensitive to their respective control parameters settings, good values of which are problem dependent. Unless the user has quite an experience in parameter tuning, finding out the best parameter settings for a given problem through trial and error may prove, at best, an arduous, and sometimes an infeasible task. A way out of this conundrum lies in an arrangement that may alter or adapt these parameters during the course of the algorithm. Much attention has been paid to this problem and many adaptive schemes have been proposed [3]-[8].

One of the main motivations of this work is to intensively investigate one of the popular evolutionary algorithms namely Differential Evolution (DE), and propose mechanisms to alter-adapt its control parameters to improve its performance. This motivation stems from the fact that performance of DE is highly dependent on its control parameters and setting up these parameters for a given problem is a non-trivial task.

The other part of this work focuses on the application side of DE. Lately evolutionary algorithms have found applications in the image processing and object tracking domains. One of the sub-domains of these problems is automatic vehicle classification. Vehicle classification is a difficult problem to tackle. Categorizing vehicles comprehensively using a video is quite an arduous task given the variety of vehicles and similarities between them at the same time. Different shapes and sizes within a single vehicle category adds to the dilemma. On top of this we have drastically changing weather conditions, shadows, camera noise, occlusions, etc., which make the task even more challenging. DE is employed as a parameter optimizer for vehicle classification and real time object tracking tasks.

The following sections briefly describe the research conducted for this dissertation. Brief descriptions of the background are presented in Sections 1.1-1.4. The motivation of the work is discussed in Section 1.5. The contributions of the work is listed in Section 1.6, and the structure of the dissertation is described in Section 1.7.

1.1. Differential Evolution

Differential Evolution (DE) [9], proposed by Storn and Price in 1995, has emerged as a robust real parameter optimizer in the field of evolutionary computation. The power and popularity of DE can be gauged from the heightened research activity on the subject in the past decade. Numerous studies have been conducted to ascertain DE's efficacy on a broad range of problems ranging from benchmark to real life scientific and engineering problems [10]. An year after its introduction, the power of DE was on display at the First International Contest on Evolutionary Optimization in May 1996 [11], where it secured first place among evolutionary algorithms. Since then, DE and its variants have performed exceedingly well in the optimization contests such as The IEEE Congress on Evolutionary Computation and alike.

DE is simple and employs few control parameters namely scale factor (F), crossover rate (Cr), and population size (NP). The performance of DE is very sensitive to the proper settings of these parameters [1], [12], [13].

An unfavorable combination of these parameters can seriously degrade the algorithm's efficacy. At the same time, choosing effective control parameter settings can be quite cumbersome. A good combination of these parameters depends upon the problem at hand and requires a good amount of experience of the user. The bottom-line is, “better and more informed values of control parameters yield better results for a given problem.”

1.2. Adaptive Differential Evolution

Since it may be difficult and time consuming to generalize a set of control parameter values, there is always a motivation to alter or adapt them during the course of the algorithm as defined by certain rules. Intensive research activity has been reported in the area of finding good values of control parameters. In [1], the authors grouped this change into three broad classes:

- **Deterministic -** the parameters are altered based on some user defined rules [9], [14].
- **Adaptive -** the parameters are allowed to adapt based on some feedback from the algorithm [15].
- **Self-Adaptive -** the parameters are encoded into the solution itself and they evolve as a part of the general population [16], [17].

During the search process a particular combination of control parameters and mutation strategy may prove more favorable than the others [18]. As a result, many partially [19]-[21] adaptive schemes that adapt one or more control parameters, and fully adaptive schemes [22] that adapt mutation strategy and control parameters, have been proposed in the past.

The efficacy of employing an adaptive mutation strategy module both in the presence and the absence of a control parameter adaptation scheme is investigated. Based on empirical results we create a pool of mutation strategies. A memory based fully adaptive version of Differential Evolution, SA-SHADE, is then proposed that adapts the control parameters to their appropriate values and chooses the best suited mutation strategy from the pool.

1.3. Video Based Vehicle Classification

Automatic vehicle classification has emerged as a significantly important element in the myriad web of traffic data collection and statistics. Regulations on road side construction for pertinent reasons, increasing vehicle density, and cost of overlaying roads are some of the factors calling for ever more efficient utilization of our existing transportation networks. A part of the solution to these pressures lies in vehicle classification systems that compute the number and type of vehicles passing a particular street or highway.

This information has an evident impact on the cost and efficiency of the transportation system; road thickness decision being one of the many advantages this system has to offer. Many video based classification systems have been proposed in the past with their own advantages and disadvantages. These systems can be primarily distinguished by the type of sensors they use, most common of which are magnetic, laser, pressure, single or multiple cameras, etc. Magnetic and laser sensors tend to have a higher classification accuracy but at the same time have high equipment and installation costs, and are intrusive techniques. Computer vision based vehicle classification systems are generally attributed with low cost and accuracy, and are an active area of research. A video based vehicle classification system is proposed that determines the type of vehicle based on the number of axles and distance between them. Hough Transform [23], a parameter based feature detection method, is employed to detect the axles. The quality of the detected circles is sensitive to appropriate settings of these parameters. Since the process is time consuming and it may not be fruitful to adjust these parameters manually every time, there is always a motivation to do a parameter search by attaching a machine learning algorithm to discover an optimized set. This parameter search is done using DE.

1.4. A Differential Evolution Based Multiclass Vehicle Classifier for Urban Environments

This chapter rebuilds and further enhances the capabilities of video based vehicle classifier proposed previously. Commercial vehicle classifiers bank upon sophisticated appearance models and employ a multitude of tracking techniques for vehicle detection and classification [139]. The main idea behind vehicle detection and tracking is to estimate the motion state of an object as accurately as possible given an image sequence. A large body of research has been conducted on this topic as it finds important applications in multitude of real life areas like video surveillance, human computer interaction, and traffic flow monitoring, to name a few. Examples include a visual traffic flow monitoring system proposed in [25], pedestrian counting system in [26], accident detection system in [27] etc. As a part of this dissertation, a DE based multi lane vehicle detector and tracker with a novel appearance based model system is proposed.

1.5. Motivation and Problem Statement

Generally, an optimization process/algorithm, given a set of constraints, tries to find the best solution among all feasible solutions for a given problem. In real life though, there remain a large number of problems in class NP [28] for which finding the best solution is not possible in polynomial time, at least as of now. In such cases, it is more plausible to find a good enough solution (sub-optimal) instead of spending a great deal of computational time to find the best solution. Moreover, optimization is not always about finding the perfect solution. Sometimes, it is more about finding a good solution given a set of constraints or environment. As the constraints change the solution also needs to change accordingly. This has many parallels in the Evolutionary Process.

Evolution in a way is also an optimization process wherein the solution or adaptive behavior depends upon the constraints posed by the environment of an organism. The behavior of an organism is optimized and changes according to the environment. Since evolution has been able to produce organisms of high perfection over a long period of time, there is always a big motivation behind application of evolutionary principles in the optimization process to solve hard real world engineering problems. It was due to this primary motivation that Evolutionary Algorithms (EA) came on the horizon of optimization.

Differential Evolution (DE) is a simple yet powerful evolutionary algorithm (EA) for global optimization introduced by Price and Storn [9]. The main motivations of this research can be summarized as follows.

- The DE algorithm has gradually become more popular among other EA's, and has been used in many practical cases, mainly because it has demonstrated good convergence properties, and is easy to understand [12]. Its success notwithstanding, DE has its own drawbacks. Its very high performance sensitivity to its control parameter settings sometimes turns out to be more of a hindrance than an asset. Improper settings of these parameters seriously degrades the DE's performance and renders it ineffective on the problem being solved. Moreover, determining good control parameter values for every problem would either require repetitive trial & error or a good amount of user experience. These approaches in most cases are either time consuming or infeasible. Allowing the control parameters to alter/adapt themselves during the search process is, therefore, an important task. For this purpose, an adaptive control parameter mechanism for control parameters are developed.

- DE's ability to find a good solution, apart from its control parameters, is also dependent on the mutation strategy it employs. A mutation strategy determines how the chosen vectors will be differentially mutated to create a donor. Even in the presence of favorable control parameter settings, an unfavorable mutation strategy can seriously degrade the quality of the final solution. Allowing the mutation strategy to adapt with the control parameters may help in alleviating this problem. For this purpose, adaptive mechanisms for mutation strategies are developed that are useful both in presence, and absence of adaptive control parameter mechanisms.

- On the application side, DE has been utilized in many real world domains for optimization. Video based classification of vehicles is an important task in intelligent transportation systems. Categorizing vehicles comprehensively is quite an arduous task given the variety of vehicles and similarities between them at the same time. Different shapes and sizes within a single vehicle category adds to the dilemma. On top of this, drastically changing weather conditions, shadows, camera noise, occlusions, etc., make the task even more challenging. A DE based vehicle classifier is proposed wherein DE is used as an optimizer to improve the overall classification accuracy. DE acts as a accuracy improvement sub system. The primary axle detection mechanism is the Hough Transform which after detection feeds the DE sub system. DE further validates the input and attempts to improve the accuracy of the classification system.

- Due to variety and complexity of scenes, and external noise, vehicle detection on multilane traffic is a challenging task. The performance of such a system is largely dependent upon the appearance model and tracker it uses. A number of approaches have been proposed both for appearance models and trackers. A bio-inspired multilane vehicle detection and classification system is proposed that uses a novel pixel-to-pixel color cue comparison approach for appearance modeling and DE as a discrete optimizer.

1.6. Contributions

This dissertation makes several contributions towards improving the convergence properties of differential evolution and reducing its dependency on trial & error methods to select a good set of control parameters.

On the application sides, a novel DE based vehicle classification system is proposed and later DE is is applied as real time object tracker with gaussian mixture object models. The contributions are:

1. Differential Evolution with Dither and Annealed Scale Factor (DEDASF) was devised and applied to twenty difficult benchmark functions. During the search process, the scale factor, F, was initially randomized to sample diverse areas of the search landscape, and then allowed to be non-linearly an-

nealed. This allows for random step sizes during initial exploratory stages of the search, and a gradual step size reduction during the exploitative stages. The performance of DEDASF was compared with state of the art algorithms proposed in the past. DEDASF proved to be highly competitive amongst the algorithms compared.

2. A fully adaptive version of DE, Strategy Adaptive Success History Based Differential Evolution (SA-SHADE) was proposed that is capable of adapting control parameters as well as the mutation strategy. A pool of mutation strategies was created based on empirical evidence, and successful mutation strategies were added to the success history. The mutation strategy to be applied in a given situation was chosen from this success history. This significantly improved the performance of DE, and reduced human intervention in setting up these parameters. SA-SHADE was compared with several state of the art adaptive variants, and was shown to be superior to most and competitive to the remainder.

3. A novel DE based vehicle classification system is proposed that utilizes the axle count, their corresponding distance from each other and other parameters of a vehicle to classify the type of vehicles. Hough transform is used as a primary axle or circle detector, and DE is then subsequently used to improve the classification accuracy. Hough transform is a parameter based method that requires the parameters to be set before its operation. This requires human intervention. To alleviate this drawback, DE is used as an automatic parameter search method. The classification system is shown to perform vastly superior in presence of DE as an optimizer when compared with the system that does not employ DE and sets the parameters manually.

4. A DE based multilane vehicle detection, tracking and classification system is proposed wherein color cues are utilized to model the appearance of the object. Multiple variants of DE are compared to ascertain robust behavior. Extensive experimental results show that DE based object tracker is robust and shows satisfactory performance for multilane vehicle detection in presence of noise, occlusion, and target deformation.

1.7. Dissertation Overview

This dissertation is a paper-based version, where each chapter has been derived from the papers published during the Ph.D. work. This is an overview of the remaining chapters of this dissertation.

In Chapter 2, a adaptive version of DE, named Differential Evolution with Dither and Annealed Scale Factor or DEDASF, is discussed. This chapter is derived from the publication:

- Deepak Dawar and Simone A. Ludwig, "Differential Evolution with Dither and Annealed Scale Factor." *IEEE Symposium on computational intelligence*: 9-12 Dec. 2014.

In Chapter 3, a fully adaptive version of DE namely SA-SHADE is presented. This chapter is derived from the submitted work:

- Deepak Dawar and Simone A. Ludwig, "Effect of Strategy Adaptation on Differential Evolution in Presence and Absence of Parameter Adaptation: An Investigation." *Applied Soft Computing*, Submitted.

In Chapter 4, a novel DE based vehicle classification system is discussed. This chapter is derived from the publication:

- Deepak Dawar and Simone A. Ludwig, "A Differential Evolution Based Axle Detector for Robust Vehicle Classification." *IEEE Congress on Evolutionary Computation*, May 25-28, 2015, Sendai, Japan.

In Chapter 5, a DE based multilane vehicle detection, tracking and classification system is discussed. This chapter is derived from the submitted work:

- Deepak Dawar and Simone A. Ludwig, "A Differential Evolution Based Multiclass Vehicle Classifier for Urban Environments." *International Journal of Swarm Intelligence Research*, Submitted.

2. DIFFERENTIAL EVOLUTION WITH DITHER AND ANNEALED SCALE FACTOR

Differential Evolution (DE) is a highly competitive and powerful real parameter optimizer in the diverse community of evolutionary algorithms. The performance of DE depends largely upon its control parameters and is quite sensitive to their appropriate settings. One of those parameters commonly known as scale factor or F, controls the step size of the vector differentials during the search. During the exploration stage of the search, large step sizes may prove more conducive while during the exploitation stage, smaller step sizes might become favorable. This work proposes a simple and effective technique that alters F in stages, first through random perturbations and then through the application of an annealing schedule. The performance of the new variant on 20 benchmark functions of varying complexity is reported, and compared with the classic DE algorithm (DE/Rand/1/bin), two other scale factor altering variants, and state of the art, SaDE.

The rest of this chapter is structured as follows. Section 2.1 presents an introduction to the chapter. In Section 2.2, related work is presented. Section 2.3 describes and explains the new algorithm with all its features. In Section 2.4, results and their analysis are presented, and finally, conclusions and future work are discussed in Section 2.5.

2.1. Introduction

Dither is a deterministic parameter control technique wherein F is allowed to take on random values between a specific range represented by F_{low} and F_{high}. Combining dither with an annealing based cooling schedule to alter F is proposed, and a two stage technique, DEDASF, is introduced based on this concept. In the first stage, dither is applied and in the second stage, F is allowed to be reduced by a randomized factor. Every stage runs for a fixed number of function evaluations, a count of which has to be set beforehand. The idea is to scatter the population to diverse and favorable areas first and then reduce the step size to take advantage of the notion that during early stages of exploration of the search space by DE, large step sizes may prove beneficial for investigating the maximum area of the problem landscape, and when the exploitation stage kicks in, small step sizes may become more advantageous. Though it is possible to alternate between exploration-exploitation during the course of the search, which may be beneficial for many landscapes, this investigation is limited to a single stage exploration-exploitation model.

2.2. Related Work

There have been many attempts to improve the performance of DE by varying the scale factor during the search process and multiple methods have been suggested to achieve this goal. One of them is Dither [14]. It is a deterministic scheme of randomization of scale factor, F. Many different ways of randomizing F are possible. For example in [29], F was randomized on a generational basis as:

$$F_{dither} = F_l + rand_G(0,1) \times (F_h - F_l) \tag{2.1}$$

where F_l and F_h are the predefined lowest and highest values of F, respectively, and $rand_G(0,1)$ is a uniformly distributed real number generated anew for every generation G. In [15], convergence of DE was reported to have been improved while using dither, though the authors applied dither on an individual basis rather than generational basis. Therefore, preliminary results make a good case for using this technique.

There is a similar technique proposed in [14] called *Jitter* wherein F is randomized for every difference of the parameters involved in the differential operation. The operation can be represented as:

$$F_{jitter} = F \times (1 + \gamma \times (rand_j[0,1] - 0.5)) \tag{2.2}$$

where it is imperative that γ be small. In [30], the author mixed both *dither* and *jitter*.

Apart from randomization schemes, another technique that appears to be useful while negotiating the search space is step-size reduction. Step size may be considered as the distance between position of the current vector and the newly generated vector, in a D dimensional space. In DE, the step size is controlled by F. In [31], authors describe a step size reduction technique for their one point direct search algorithm. The algorithm starts with a point x_0 in a D dimensional space. The nearby space is explored and a new point x_1 is selected for evaluation. If x_1 is found to be worse than x_0, then the step size is assumed to have been too large and is reduced by a certain factor. An obvious drawback of this scheme is that the technique only contracts the step size and never expands it thereby increasing the chances of the solution getting stuck in a local optimum.

In [15], authors proposed a step size reduction scheme, DETVSF, wherein they reduced the step size with every generation. Mathematically, this scheme is described as:

$$F_{curr} = (F_{max} - F_{min}) \times (G_{max} - G_{curr})/G_{max} \tag{2.3}$$

where F_{curr} is the current value of the scale factor, F_{max} and F_{min} are the predefined maximum and minimum values of the scale factor, respectively. G_{curr} and G_{max} are the current and maximum generation number, respectively. This technique was reported to have improved the performance of DE in a statistically meaningful way [15].

The step size reduction by a constant factor may well be juxtaposed with the concept of the metallurgical technique of annealing, which is the process of treating a metal by first heating it above its critical temperature and then cooling it at a certain rate.

The work in this chapter is motivated by the encouraging results reported in [15] and [29], which clearly insinuate the use of dither and step size reduction techniques. Though the individual results of these schemes are promising, a closer look might reveal some shortcomings of the individual use of these techniques. Dither offers randomized step sizes throughout the search process. During the initial phase of the search, this may prove useful but during the later phase, when focus shifts to a particular area of landscape, arbitrary and occasional large step sizes may prove detrimental to convergence. Thus, dither may be avoided during the later part of the search. During the exploration stage of the search, large step sizes are advantageous, while during the exploitation stage, small values prove more effective. While step size reduction techniques make a lot of sense, solely contracting step size run the risk of getting stuck in local minimum if there is not enough diversity in the population.

With these points in mind we decided to hybridize these techniques to take advantage of their individual strengths. Our hybrid is stage based. Dither is applied in the first and step size reduction in the second stage. This sequence of stages proves more conducive and effective, as we shall present in the results, than employing the dither or step size reduction technique alone.

2.3. DEDASF

This work proposes DEDASF, Differential Evolution with Dither and Annealed Scale Factor, an algorithm that applies dither and annealing to the scale factor, F. DEDASF is summarized in Algorithm 1.

Algorithm 1 PSEUDO-CODE FOR DEDASF

```
Set values of $NP$, $Cr$
Set Dither range $F_l$, $F_h$. Set no. of generations, $G_d$, for which dither would be applied
Set Annealing constants $F_0$, $\alpha_l$, $\alpha_h$
Initialize a population of $NP$ individuals $P = [X^1, X^2, ...X^{NP}]$ where every $i^{th}$ individual is a $D$ dimensional vector represented as $X_j^i$=[$x_1^i$, $x_2^i$ ... $x_D^i$].
While stopping criteria is not met do
  For every target vector $X^{target}$ in $P$ do
    Select three vectors $X^{r_1}, X^{r_2}, X^{r_3}$ where $r_1, r_2$, and $r_3$ are three mutually exclusive indices and different from the index of target vector
    Produce a donor vector through mutation as
    $X^{donor} = X^{r_1} + F \times (X^{r_2} - X^{r_3})$ where $F$ is calculated as:
        $F = \begin{cases} F_l + rand(0,1] \times (F_h - F_l) & \text{if } G_c \leq G_d \\ \alpha^{G_c - G_d} \times F_0 & \text{otherwise} \end{cases}$
    Produce a trial vector, $X^{trial}$ = ($x_1^{trial}, \ldots, x_D^{trial}$), through crossover as:
        $x_j^{trial} = \begin{cases} x_j^{donor} & \text{if } rand_j(0,1] \leq Cr \text{ or } j = j_{rand} \\ x_j^{target} & \text{otherwise.} \end{cases}$
    Select either the target vector or the trial vector based on their fitness values as:
        $X_{G+1}^{survivor} = \begin{cases} X_G^{trial} & \text{if } F(X_G^{trial}) \leq F(X_G^{target}) \\ X_G^{target} & \text{otherwise.} \end{cases}$
  end For
end While
```

The algorithm makes changes to F in two stages. The duration of the first stage is a pre-specified number of generations or function evaluations (FEs). In the first stage, dither is applied to F within a pre-specified range. For our experiments we chose the range [0.1,0.9] to promote a multitude of step sizes that would help sample different and wide areas of the search landscape. Also, loss of diversity is a known problem in DE and dither may help improve it as argued in [32].

After the first stage, when the population has scattered sufficiently to seemingly favorable areas of the landscape, F is allowed to cool slowly and the cooling rate is controlled by the randomized factor α. During the exploration stage a high value of F is advantageous, while during the exploitation stage small values of F are desirable. Thus, while in general it is difficult to specify the step size for different stages of the search, this scheme may help the search make a smooth transition from the exploration to the exploitation stage. The two stages of this scheme can be outlined as:

$$F_c = \begin{cases} F_l + rand(0,1] \times (F_h - F_l) & \text{if } G_c < G_d \\ \\ \alpha^{G_c - G_d} \times F_0 & \text{otherwise} \end{cases} \tag{2.4}$$

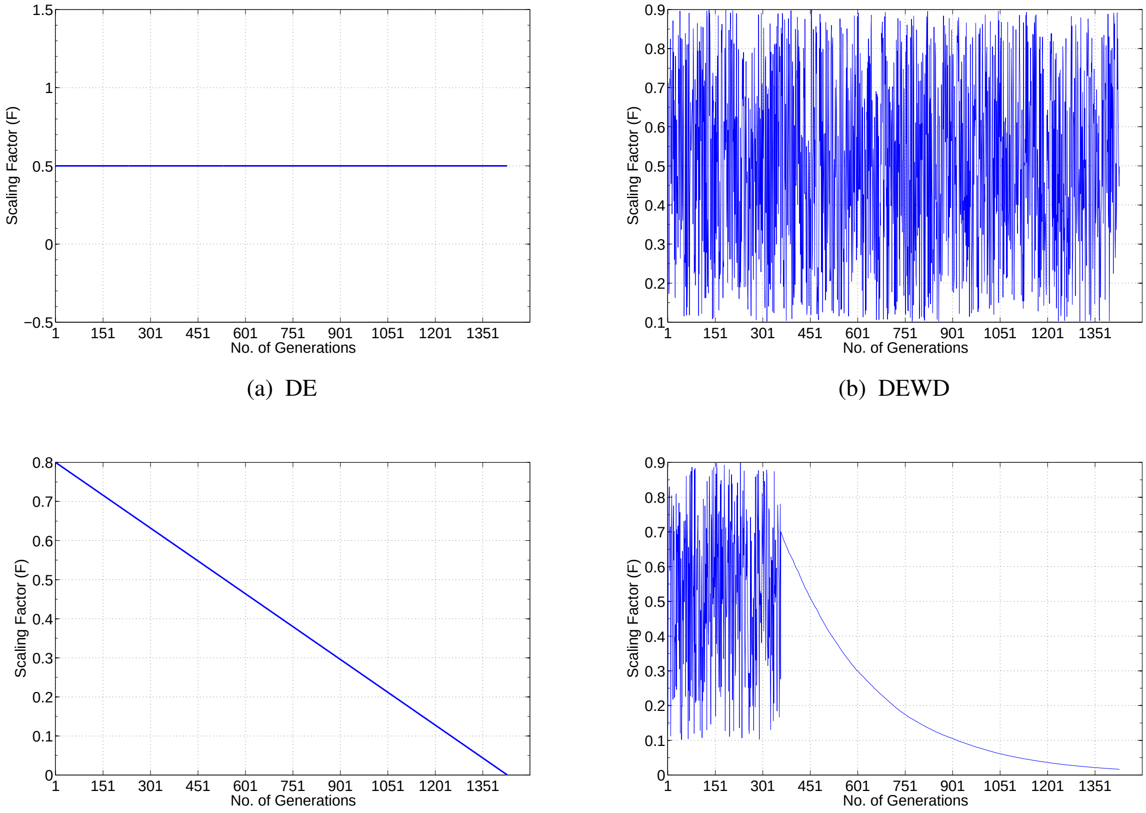
where F_c is the current value of scale factor, F_l and F_h, the predefined lowest and highest values of scale factor. G_c is the current generation, and G_d is the number of generations allotted to the dither stage.

F_0 is the value at which the reduction of scale factor starts or critical temperature in metallurgical terms. After an empirical study, we fixed F_0 at 0.7 as this value should not be too high or too low for the search would have progressed towards favorable regions by the time the annealing stage kicks in. Keeping F_0 high would slow down the convergence and a low value might result in a failure to explore prospective areas. In Equation 2.4, α represents the cooling rate, a value which is randomized between α_l and α_h.

After an empirical study, a part of which is explained in the Results section, the values of α_l and α_h were fixed at 0.995 and 0.998, respectively, and α calculated as:

$$\alpha = rand(0,1] \times (\alpha_h - \alpha_l) \tag{2.5}$$

where $rand(0,1]$ is a uniformly distributed random number between 0 and 1.



(a) DE

(b) DEWD

(c) DETVSF

(d) DEDASF

Figure 2.1. Variation in scale factor with number of generations executed for DE, DEWD, DETVSF, and DEDASF.

2.4. Experimentation and Results

2.4.1. Benchmark Functions

Experiments were performed on 20 benchmark functions of varying properties and geometric orientations. The first five functions (f1 to f5) are unimodal, the next fifteen (f6 to f20) are multimodal. Due to paucity of space in this chapter, the details about the test functions are difficult to provide here, but a detailed insight into the functions can be found in [33].

2.4.2. DEDASF vs Other Methods

Algorithm comparison is performed in two ways. First we compare DEDASF with other deterministic parameter control techniques to determine its rank. A comparative performance test of DEDASF is performed with: (a) Classic DE, (b) DETVSF, as proposed in [15], and (c) DEWD, classic DE with dither as proposed in [14]. DETVSF and DEWD were described in Section III. Three of the compared algorithms namely DEDASF, DETVSF, and DEWD alter F as the search progresses, while in DE, F is kept constant. For these compared algorithms, the variation in F with the number of generations is presented in Fig 2.1.

After the initial comparative evaluations, DEDASF is then compared with Self Adaptive Differential Evolution algorithm, SaDE [34], to contrast the performance of with our deterministic parameter control method with the adaptive one.

2.4.3. Control Parameter Set Up

A large body of research on control parameter settings is available, which while not being able to provide a panacea for the control parameter setting problem, provides suitable guidelines for their use.

Authors in [13] suggest that the population size NP be between 3D to 8D, where D is the dimensionality of the problem. Using their guidelines coupled with our own experience, we fixed NP to be seven times the problem dimensionality. Storn and Price in [14] suggest that Cr be either between [0.0,0.2] or [0.9,1]. The reason for such division is that separable functions are solved quite well at low values of Cr, and non-separable at high values. However, to maintain uniformity, we fixed Cr at 0.9. Another good reason for this choice is to increase the diversity and minimize the orthogonal movements of vectors. A high value of Cr is also recommended in [35].

DEDASF, DTVSF, DEWD alter the scale factor with their own mechanisms. The scale factor for classic DE is fixed at 0.5 as suggested in [35].

According to the guidelines laid down in [33], the maximum number of function evaluations (MaxFEs) has been restricted to 10^4 times the dimensionality of the problem, every benchmark function is evaluated 51 times, and evaluation is terminated once MaxFEs is reached or the difference between the global optimum and current best reaches a value of 10^{-8}.

2.4.4. Results

Table 2.1 reports the performance of the deterministic parameter control algorithms namely DEDASF, DETVSF, DEWD, and the classic DE, at problem dimensionality 10, 30, and 50, respectively.

With a mere glance at the Table 2.1, it can be inferred that none of the algorithms perform significantly better than its competitors at problem dimensionality 10 where DEDASF and DE score two wins each, DETVSF eight, and DEWD four wins. Table 2.1 also revels that at problem dimensionality 30, DEDASF wins 10 times, DETVSF 3 times, and DEWD wins 4 times while classic DE does not record any win. At 50D, DEDASF scores 12 wins, DETVSF 3, and DEWD 4, while DE again scores no wins. To decipher the statistical difference between the algorithms, we compare them with the Friedman test and later on with the Hochberg post-hoc procedure.

The Friedman test [36], is a multiple comparisons procedure that aims to detect significant performance differences between the compared algorithms. It calculates the relative ranks of the algorithms through an average ranking procedure and computes the Friedman statistic, which is further used to calculate the *p* value.

Table 2.2 presents the relative ranks, and Table 2.3 reports the Friedman statistic and *p* values obtained by the algorithms at problem dimensionality 10, 30, and 50, respectively.

It is observed that the Friedman test did not detect a significant difference between the algorithms at problem dimensionality 10. DETVSF emerges as the best ranked algorithm, but the difference is not statistically significant when compared to other algorithms either at 0.05 or 0.1 level of significance.

At problem dimensionality 30 and 50, however, the Friedman test reports a significant difference between the algorithms. The difference is significant at the 0.05 level of significance, as clearly shown in Table 2.3, and DEDASF clearly emerges as the best ranked algorithm.

The Friedman test is capable of detecting significant differences between algorithms, but is unable to perform comparisons between some of the algorithms, for example, when a particular control algorithm

Table 2.1. Performance of DEDASF, DETVSF, DEWD, and DE at 10D, 30D, and 50D, respectively. Reported values are the averages of 51 independent runs for each function. Error values reaching within 10^{-8} of the global optimum of the function are reported as 0.00+E00.

10D				
Function	**DEDASF**	**DETVSF**	**DEWD**	**DE**
1	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00
2	2.63E+03±6.38E+03	**1.31E-02**±5.73E-02	7.81E+02±1.97E+03	2.01E+01±9.04E+01
3	4.21E+00±1.38E+01	1.95E-01±2.10E-01	1.74E+00±6.15E+00	**2.25E-02**±6.32E-02
4	4.03E+01±9.05E+01	**2.62E-04**±9.37E-04	1.96E+00±5.94E+00	3.56E-02±1.70E-01
5	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00
6	**0.00E+00**±3.65E-08	2.00E-01±1.70E-01	2.48E+00±0.00E+00	2.97E-04±1.06E-03
7	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00
8	2.05E+01±5.54E-02	2.05E+01±5.09E-02	2.05E+01±6.00E-02	2.05E+01±5.13E-02
9	3.00E-02±6.14E-01	1.80E-01±3.29E-01	9.72E-02±3.22E-01	**1.92E-05**±9.05E-05
10	4.63E-02±2.06E-01	8.77E-02±6.31E-02	**4.35E-02**±3.27E-02	7.13E-02±4.88E-02
11	8.95E-01±5.51E-01	9.12E-01±1.16E+00	**7.06E-01**±7.66E-01	6.46E+00±5.48E+00
12	6.15E+00±3.95E+00	7.03E+00±2.84E+00	**5.52E+00**±2.66E+00	1.67E+01±8.40E+00
13	9.32E+00±2.09E-01	**8.66E+00**±3.74E+00	1.16E+01±5.34E+00	1.46E+01±8.26E+00
14	5.65E+01±3.41E-01	**1.75E+01**±1.10E+01	1.92E+02±1.63E+02	9.21E+02±2.69E+02
15	2.42E+02±3.09E-02	**1.59E+02**±9.27E+01	1.05E+03±3.05E+02	1.38E+03±1.08E+02
16	1.11E+00±7.47E-01	**1.04E+00**±1.67E-01	1.08E+00±2.14E-01	1.04E+00±1.80E-01
17	**1.15E+01**±2.64E+00	1.16E+01±7.99E-01	2.38E+01±5.03E+00	2.52E+01±3.27E+00
18	2.21E+01±4.44E+00	**1.90E+01**±2.03E+00	3.27E+01±6.32E+00	3.46E+01±4.80E+00
19	5.69E-01±5.98E+01	**5.38E-01**±7.21E-02	5.57E-01±1.25E-01	5.38E-01±1.69E-01
20	2.27E+00±2.13E+02	1.65E+00±4.57E-01	**1.07E+00**±5.52E-01	2.61E+00±2.00E-01

30D				
Function	**DEDASF**	**DETVSF**	**DEWD**	**DE**
1	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00
2	1.09E+06±4.17E+05	1.53E+06±5.52E+05	**9.38E+05**±4.61E+05	3.91E+06±1.11E+06
3	**2.41E+06**±2.97E+06	2.49E+06±2.80E+06	4.80E+06±3.14E+06	7.90E+06±5.01E+06
4	9.76E+03±2.23E+03	1.44E+04±2.96E+03	**4.64E+03**±1.44E+03	3.03E+04±5.69E+03
5	2.87E-06±4.72E-06	2.32E-06±9.25E-06	**0.00E+00**±0.00E+00	1.01E-05±2.84E-06
6	9.48E+01±2.01E+01	**2.93E+00**±2.29E-01	1.40E+01±2.86E+01	8.17E+01±9.88E+00
7	**2.87E-02**±1.42E-01	5.33E-01±6.29E-01	3.79E-01±8.72E-01	3.08E+00±1.57E+00
8	2.10E+01±4.87E-02	2.10E+01±4.80E-02	2.10E+01±3.38E-02	2.10E+01±4.32E-02
9	3.79E+00±2.48E+00	6.39E+00±2.00E+00	**1.42E+00**±1.33E+00	3.75E+01±1.19E+00
10	7.66E-02±9.02E-02	2.10E-01±2.88E-01	**2.86E-02**±1.36E-02	8.07E-01±2.03E-01
11	5.69E+00±2.17E+00	**3.84E+00**±1.56E+00	6.16E+01±1.61E+01	1.82E+02±1.09E+01
12	3.14E+01±6.82E+00	**2.84E+01**±7.25E+00	1.38E+02±3.63E+01	1.98E+02±1.15E+01
13	**5.50E+01**±1.85E+01	5.57E+01±1.57E+01	1.56E+02±2.37E+01	1.97E+02±8.63E+00
14	**1.90E+02**±1.09E+02	6.80E+02±1.76E+02	6.56E+03±3.11E+02	7.08E+03±2.50E+02
15	**3.91E+03**±1.02E+03	5.84E+03±3.79E+02	7.10E+03±2.14E+02	7.33E+03±2.33E+02
16	**2.43E+00**±2.97E-01	2.54E+00±3.10E-01	2.47E+00±2.45E-01	2.46E+00±2.56E-01
17	**4.06E+01**±3.20E+00	4.89E+01±4.05E+00	1.86E+02±8.03E+00	2.16E+02±8.35E+00
18	**1.43E+02**±2.17E+01	1.82E+02±8.98E+00	2.01E+02±1.07E+01	2.27E+02±1.07E+01
19	**1.26E+00**±2.16E-01	1.86E+00±1.16E-01	2.01E+00±3.35E-01	1.90E+00±4.75E-01
20	**1.09E+01**±8.08E-01	1.20E+01±2.81E-01	1.18E+01±2.99E-01	1.33E+01±1.41E-01

50D				
Function	**DEDASF**	**DETVSF**	**DEWD**	**DE**
1	8.72E-07±2.38E-06	1.18E-06±3.87E-06	**0.00E+00**±0.00E+00	1.64E+00±3.90E-01
2	**7.55E+06**±2.06E+06	1.08E+07±3.04E+06	1.56E+07±3.29E+06	1.11E+08±1.86E+07
3	**1.31E+07**±2.71E+07	2.94E+07±2.79E+07	2.76E+07±3.11E+07	8.97E+07±2.74E+07
4	**5.00E+04**±7.41E+03	7.02E+04±7.01E+03	9.82E+04±9.46E+03	1.25E+05±8.85E+03
5	3.88E-03±4.51E-03	1.29E-03±2.48E-03	**2.41E-06**±1.56E-06	7.96E-01±1.34E-01
6	**1.20E+02**±1.11E+01	1.68E+02±2.60E+01	1.31E+02±3.17E+01	1.55E+02±1.37E+01
7	7.46E+00±1.95E+00	**4.40E+00**±1.93E+00	8.62E+00±3.42E+00	8.53E+01±1.00E+01
8	2.11E+01±3.84E-02	2.11E+01±3.06E-02	2.11E+01±2.78E-02	2.11E+01±4.67E-02
9	**1.33E+01**±4.77E+00	2.16E+01±4.03E+00	1.52E+01±1.42E+01	7.02E+01±1.68E+00
10	4.13E+00±1.21E+00	6.06E+00±2.08E+00	**1.03E+00**±1.54E-01	6.88E+01±1.31E+01
11	8.13E+00±2.69E+00	**8.10E+00**±2.73E+00	2.12E+02±2.39E+01	3.91E+02±1.55E+01
12	**6.80E+01**±1.32E+01	1.18E+02±4.22E+01	3.38E+02±1.25E+01	4.12E+02±1.91E+01
13	**1.33E+02**±2.68E+01	1.89E+02±4.96E+01	3.49E+02±1.56E+01	4.18E+02±1.38E+01
14	**4.74E+02**±2.17E+02	3.01E+03±4.77E+02	1.31E+04±4.01E+02	1.33E+04±2.71E+02
15	**1.27E+04**±6.77E+02	1.35E+04±4.29E+02	1.36E+04±4.03E+02	1.34E+04±4.13E+02
16	3.31E+00±2.58E-01	3.42E+00±2.65E-01	**3.26E+00**±3.92E-01	3.29E+00±2.10E-01
17	**8.20E+01**±9.57E+00	1.07E+02±8.20E+00	3.71E+02±1.49E+01	4.44E+02±1.58E+01
18	**3.62E+02**±1.29E+01	3.81E+02±1.13E+01	3.97E+02±1.49E+01	4.58E+02±1.48E+01
19	**2.56E+00**±3.72E-01	2.70E+00±3.52E-01	2.65E+00±4.55E-01	1.47E+01±1.89E+00
20	2.25E+01±9.87E-01	**2.16E+01**±8.75E-01	2.18E+01±2.51E-01	2.50E+01±2.14E-01

Table 2.2. Relative Ranks obtained by DEDASF, DETVSF, DEWD, and DE at 10D, 30D, and 50D.

Algorithm	Rank-10D	Rank-30D	Rank-50D
DEDASF	2.475	**1.625**	**1.6**
DETVSF	**2.075**	2.325	2.375
DEWD	2.575	2.325	2.35
DE	2.875	3.725	3.675

Table 2.3. Friedman statistic (distributed according to chi-square with 3 degrees of freedom) and *p* value computed by Friedman Test at 10D, 30D, and 50D.

Dimension	Friedman Statistic	*p* Value
10	3.93	0.269123
30	27.93	**0.000004**
50	26.745	**0.000007**

is to be compared with other algorithms. To do this, a family of hypotheses must be defined and then a post-hoc analysis should be conducted to find a *p* value indicating rejection or acceptance of the family of hypotheses. To compute this *p* value, we performed a post-hoc analysis using the Hochberg procedure [37].

Tables 2.5 and 2.6 present the unadjusted and adjusted *p* values obtained by the Hochberg post-hoc procedure at problem dimensionality 30 and 50, respectively. The Hochberg post-hoc procedure was not applied to the Friedman test results obtained at 10D as there was not any significant difference reported between the compared algorithms. The Hochberg post-hoc test suggests that DEDASF is significant at the 0.1 level of significance for problem dimensionality 30 and 50.

It can be inferred from Figure 2.2 that DEDASF is not relatively effective at lower dimensions but its performance improves as the dimensionality of the problem increases, though more research needs to be conducted to firmly confirm this observation.

Another aspect of the results that demands attention is the effect of the cooling rate of DEDASF on the search process. It is found that some of the benchmark functions respond well to a slow cooling rate, while others lend themselves well to a quick rate of cooling. There are a few test functions that respond to cooling rates in an arbitrary manner. Hence, after extensive experimentation with several cooling rates, we concluded that a single cooling rate would not be suitable for all the test functions. Through this experimentation we found the approximate range between which the cooling rate is effective, which we

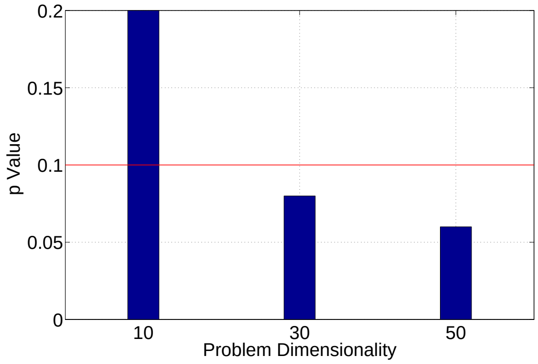
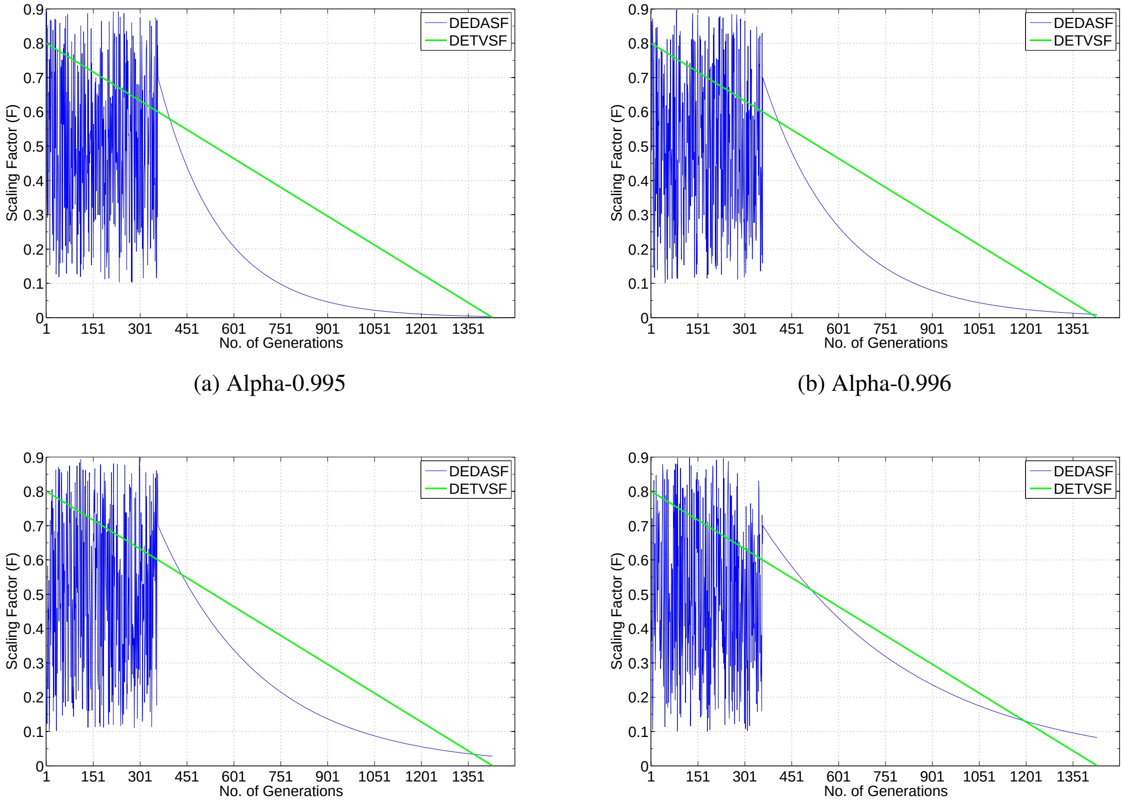


Figure 2.2. Adjusted p values obtained by Hochberg procedure for DEDASF at problem dimentionality 10, 30, and 50. DEDASF is significant at the 0.1 level of significance at problem dimensionality 30 and 50.



(a) Alpha-0.995

(b) Alpha-0.996

(c) Alpha-0.997

(d) Alpha-0.998

Figure 2.3. Comparison of slope of DEDASF and DETVSF at different cooling rates.

denote as $\alpha_{low}(0.995)$ and $\alpha_{high}(0.998)$, the highest and lowest cooling rate, respectively. The higher the value of α, the lower the cooling rate.

A division of test functions based on their response to cooling rates is shown in Table 2.4, where Type I represents the functions that show good results with slow cooling rates (high α), Type II represents the functions that show good results with fast cooling rates (low α), and functions in Type III do not follow a specific pattern.

Table 2.4. Division of benchmark functions based on their response to cooling rates.

Type I	Type II	Type III
2	11	1
3	12	8
4	13	16
5	14	19
6	15	-
7	17	-
9	18	-
10	20	-

Figure 2.3 shows the slope of DEDASF with different values of α as compared to DETVSF. Since a single cooling rate would not yield good results for all the test functions, we decided to randomize it between the range α_{high} and α_{low}. Randomization indeed proved useful and resulted in significant performance improvements at problem dimensionality 30 and 50 as is clear from Tables 2.5 and 2.6.

Table 2.5. Adjusted p-values-30D.

Algorithm	unadjusted p	$p_{Hochberg}$
DE	0	**0.000001**
DETVSF	0.086411	**0.086411**
DEWD	0.086411	**0.086411**

We also conducted a one-on-one comparison between the algorithms that reduce the scale factor during the course of execution, i.e., DEDASF and DETVSF since comparing multiple algorithms may run

Table 2.6. Adjusted p-values-50D.

Algorithm	unadjusted p	$p_{Hochberg}$
DE	0	**0.000001**
DETVSF	0.057649	**0.066193**
DEWD	0.066193	**0.066193**

the risk of accumulating the Family Wise Error Rate (FWER), even when it is controlled. We used the well known Wilcoxon test for the comparison. The result is reported in Figure 2.4.

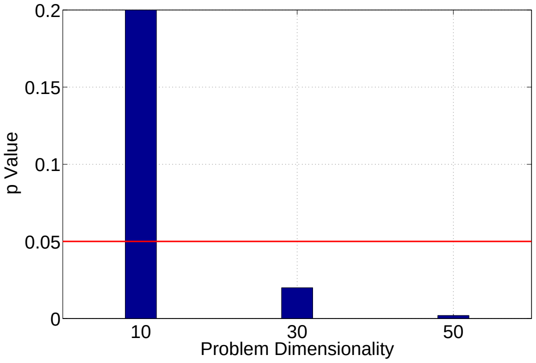


Figure 2.4. Exact p values obtained by Wilcoxon test for DEDASF when compared to DETVSF, at problem dimensionality 10, 30, and 50. DEDASF is significant at the 0.05 level of significance at problem dimensionality 30 and 50.

It is clear from Figure 2.4 that again there is no significant difference between DEDASF and DETVSF at problem dimensionality 10. But at problem dimensionality 30 and 50, DEDASF is statistically significant compared to DETVSF at the significance level 0.05. Also, the performance of DEDASF improves as the problem dimensionality increases, which was also the case during the comparison of multiple algorithms.

To contrast the performance of DEDASF with SaDE (results sourced from [38]), we first performed the sign test, and then the Wilcoxon test. The reason for choosing a double test measure is the fact that while sign test being very crude and insensitive, provides a general idea about the performance difference, and Wilcoxon test provides a more sensitive overall performance report. Table 2.7 indicates that DEDASF is competitive at problem dimensionality 10 and 30 but does not perform well at 50 dimensions. Part of the

Table 2.7. Performance of DEDASF when compared to SaDE.

Dimensions	Wins	Loses	p Val (Sign Test)	p Val (Wilcoxon)
10	8	12	0.50	0.42
30	9	11	0.82	0.73
50	7	13	0.26	0.21

reason for less than convincing performance of DEDASF at 50 dimensions may be attributed to the lack of adaptive capabilities of DEFASF.

2.5. Summary

This work presents DEDASF, a variation of the classic DE algorithm wherein the scale factor, F, is altered first with dither and then reduced at a certain rate. We report the performance of DEDASF at three different problem dimensionalities, 10, 30 and 50, on the twenty benchmark functions. We compare DEDASF with DETVSF (another algorithm that reduces F with time with a constant factor), DEWD (an algorithm that randomizes F), the classic DE, and SaDE.

We conduct a post-hoc analysis using the Hochberg procedure to determine the best performing deterministic parameter control algorithm. The results indicate that though there is no significant difference between the compared algorithms at problem dimensionality 10, DEDASF is significant at significance level 0.1 at problem dimensionality 30 and 50. Also, DEDASF is significant at significance level 0.05 when only scale factor reduction algorithms, namely DEDASF and DETVSF, are compared. DEDASF also fairs well at problem dimensionality 10 and 30 when compared with SaDE but is outperformed by SaDE at 50 dimensions, though the difference is not significant. Moreover, an interesting finding from this work is the observation that different functions respond differently to various cooling rates. Future work includes testing the scheme at higher dimensions and on a variety of test problems, and performing further analysis to contrast the behavior of the functions when subjected to different step sizes.

3. EFFECT OF STRATEGY ADAPTATION ON DIFFERENTIAL EVOLUTION IN PRESENCE AND ABSENCE OF PARAMETER ADAPTATION: AN INVESTIGATION

Differential Evolution (DE) is a simple, yet highly competitive real parameter optimizer in the family of evolutionary algorithms. A significant contribution of its robust performance is attributed to its control parameters, and mutation strategy employed, proper settings of which, generally lead to good solutions. Finding the best parameters for a given problem through the trial and error method is time consuming, and sometimes impractical. This calls for the development of adaptive parameter control mechanisms. In this work, we investigate the impact and efficacy of adapting mutation strategies with or without adapting the control parameters, and report the plausibility of this scheme. Backed with empirical evidence from this and previous works, first a case is build for strategy adaptation in the presence as well as in the absence of parameter adaptation. Afterwards, a new mutation strategy, and an adaptive variant SA-SHADE is proposed which is based on a recently proposed self-adaptive memory based variant of Differential evolution, SHADE. The performance of SA-SHADE on 28 benchmark functions of varying complexity is reported, and compared with the classic DE algorithm (DE/Rand/1/bin), and other state-of-the-art adaptive DE variants including CoDE, EPSDE, JADE, and SHADE itself. The results show that adaptation of mutation strategy improves the performance of DE in both presence, and absence of control parameter adaptation, and should thus be employed frequently.

The rest of this paper is structured as follows. Section 3.1 presents an introduction to the chapter. In Section 3.2, related work is presented. Section 3.3 presents the empirical results for building a case for strategy adaptation irrespective of parameter adaptation. In Section 3.4, SA-SHADE is described with all its features and then compared with state-of-the-art adaptive DE variants. Section 3.5 concludes this paper.

3.1. Introduction

Challenging real world optimization problems are ubiquitous in scientific, and engineering domains. Complexity of the problem notwithstanding, its objective function may also be non-continuous, and non-differentiable adding to the overall difficulty, and negotiability of the search space. Researchers have been looking towards Darwinian inspired evolutionary theories like social group behavior, and foraging strategies, to name a few, for tackling hard, and complex optimization problems. Nature inspired algorithms are the outcomes of such research activity. These algorithms can be broadly classified into two categories: evolutionary computing methods, and swarm intelligence algorithms, both of which employ their own set of control parameters.

The underlying idea behind evolutionary algorithms is the iterative fitness improvement of a population of individuals (solutions), through natural selection. An iteration generally involves, producing new individuals through a series of mutations and recombinations, gradually removing lesser fit individuals from the population, and replacing them with newly generated individuals if their fitness proves to be better than the individuals they were generated to replace [1]. The operation of swarm intelligence algorithms may be behaviorally characterized as a decentralized swarm searching for optimal food sources (solutions) [2]. The direction of individual search is influenced by the current location of the individual, its best location ever, and the location of the best individual in the whole swarm. The performance of both these classes of algorithms is quite sensitive to their respective control parameter settings, good values of which are problem dependent. Unless the user has quite an experience in parameter tuning, finding the best parameter settings for a given problem through trial and error may prove, at best, an arduous, and sometimes an infeasible task. A way out of this conundrum lies in an arrangement that may alter or adapt these parameters during the course of the algorithm. Much attention has been paid to this problem and many adaptive schemes have been proposed in the past [3]-[8].

Lately, Differential Evolution (DE) [9], an evolutionary algorithm, has established itself as a robust real parameter optimizer. Intensive research activity on the subject in the past decade speaks volumes of its power and popularity. DE has been rigorously evaluated on a broad range of benchmark problems, and has been extensively applied to real life scientific and engineering problems [10]. It also secured first position in the First International Contest on Evolutionary Optimization in May 1996 [11].

DE is simple and operates with only a few control parameters namely scale factor (F), crossover rate (Cr), and population size (NP). The performance of DE, as with any evolutionary algorithm, is quite sensitive to the appropriate settings of these parameters as reported in [1], [12], [13]. A good setting can improve both the convergence speed, and the quality of the solution. Conversely, a poorly chosen setting of these parameters can seriously deteriorate the algorithm's efficacy. Given the importance the parameter setting carries, choosing effective control parameter values, at the same time, can be quite a tedious task.

Generally, an effective combination of these parameters depends upon the problem being tackled, and necessitates a good amount of user experience. It would not be inappropriate to remark that the more informed values of these control parameters are, the better the results. While the role of good parameter settings in DE's performance may be unequivocal, there is no single accepted scheme to ascertain their universally applicable or effective values.

As a result, a good deal of research effort has been spent to devise and further improve the alter/adapt schemes to automatically find good, and acceptable values of these control parameters.

The efficacy of employing an adaptive mutation strategy module both in presence and absence of a control parameter adaptation scheme is investigated in this work. Based on empirical results obtained through this investigation on 28 benchmark functions, a pool of successful mutation strategies is created. Then a memory based fully adaptive version of Differential Evolution, SA-SHADE, is proposed that adapts the control parameters to their appropriate values and chooses the best suited mutation strategy from the pool.

3.2. Related Work

It is an established notion that the performance of DE depends greatly on the mutation strategy employed, and the corresponding control parameters [12]. As the complexity of the problem increases, this dependence becomes even more profound [13]. A good choice of mutation strategy and control parameters can lead to better results, and at the same time, an unfavorable choice may seriously degrade DE's performance [14], [20], [40]. Choosing a good mutation strategy and associated control parameters is not an easy task and requires quite a bit of user experience. A good amount of research has been conducted in the area of determining good values of the control parameters. Authors in [41] suggested that good values of F lie between 0.4 and 0.95. For Cr, they ascribed the range (0,0.2) for separable functions and (0.9,1) for non-separable functions. On the other side of the spectrum, the authors in [13] suggested good value of F to be 0.6 and Cr ranging between [0.3,0.9]. As can be seen, these suggestions differ, and sometimes, are conflicting at best. This situation naturally calls for adaptive mechanisms that would require little or no user intervention in setting up the control parameters while optimizing with DE.

Much work has been reported on this problem of automating mutation strategy and control parameters [18], [42], [43], [44]. A fuzzy adaptive differential evolution with fuzzy logic controllers was presented in [42] where F and Cr are adapted based on the relative fitness values and individuals of subsequent generations. Authors developed linguistic fuzzy sets to encode knowledge by taking into consideration the noise and non-linearity of the objective function. Qin $et\ al.$ [34] proposed a memory based self adaptive differential evolution (SaDE) algorithm. They adapted the mutation strategy depending upon its success history. The mutation strategy is chosen from a pool and successful strategies and Cr values are recorded. The subsequent mutation strategy is selected probabilistically based on its ability to produce successful trials. The scale factor, F, is not adapted and instead randomly sampled from the normal distribution (0.5,0.3). The idea was to employ exploration (large F values) and exploitation (small F values) throughout the search process. The successful values of the crossover rate, Cr, on the other hand, are stored in a memory bank, and new values of Cr are generated from the normal distribution $N(Cr_m$,0.1), where Cr_m is the median Cr value in the memory bank m. A small value of standard deviation 0.1 was chosen to guarantee that most of the Cr values generated by $N(Cr_m$,0.1) are between [0,1], even when Cr_m is close to 0 or 1.

In the self-adaptive scheme, jDE, proposed by Brest $et\ al.$ [17], F and Cr are encoded directly into the individuals so that individuals with better values of these parameters are more likely to survive, thus automatically retaining good parameter values, increasing the length of the vector. Two new parameters τ_1 and τ_2 are introduced to control the values of F and Cr as

$$F^i_{G+1} = \begin{cases} F_l + rand_l \times F_u & \text{with probability } \tau_1 \\ \\ F^i_G & \text{otherwise} \end{cases} \tag{3.1}$$

$$Cr^i_{G+1} = \begin{cases} rand_2 & \text{with probability } \tau_2 \\ \\ Cr^i_G & \text{otherwise} \end{cases} \tag{3.2}$$

where F_l and F_u are the lower and upper limits of F restricted to the range [0,1]. The authors used τ_1=τ_2=0.1 with F_l=0.1 and F_u=0.9. Thus, essentially F (0.1,0.9) and CR (0,1) are restricted to their respective ranges. It should be noted that this scheme has four extra parameters to be set namely F_l, F_h, τ_1, and τ_2, which might pose a problem in itself. The authors, in this case, opined to have used only a single setting for them and kept them constant throughout the search.

A fitness based adaptation of F was proposed in [45]. Cr was fixed at 0.5. The mechanism comprised of two evolving populations. After every generation, F was updated as

$$F = \begin{cases} \max(l_{min}, 1 - \frac{f_{max}}{f_{min}}) & \text{if } \frac{f_{max}}{f_{min}} < 1 \\ \\ \max(l_{min}, 1 - \frac{f_{min}}{f_{max}}) & \text{otherwise} \end{cases} \tag{3.3}$$

where f_{min} and f_{max} are the generational minimum and maximum objective function values obtained by the individuals over the populations and l_{min} is the lower bound on F.

In [32], authors proposed a scheme wherein F was reduced linearly with an increase in the number of function evaluations. The idea was to use high values of F during the exploration stage and small values during the exploitation state in the later part of the search. Dawar $et\ al.$ in [46] proposed a similar technique with the difference that they used random perturbation of F in the initial stages of the search, and reduced F non-linearly afterwards. Both of the above approaches demonstrated favorable performance over conventional DE.

Authors in [19] proposed another adaptive version of DE named SDE, in which F and population size NP were adapted but Cr was sampled from a normal distribution N(0.5,0.15). SDE was reported to have outperformed other basic versions of DE described in [9]. On similar lines, DESAP (Differential evolution with self adaptive population size) was proposed by Teo [47] in which the population size, NP was adapted alongside F and Cr. Population size reduction has also been reported to have a favorable effect on the performance of DE as argued in [48]. Authors of the same work reported an improvement in both efficiency and robustness of DE when NP is gradually reduced.

Another novel adaptive mechanism proposed in [22] uses three different pools of values, one for each mutation strategy, F, and Cr, respectively. The F pool contained the values in the range [0.4,0.9] with an increment of 0.1, and the CR pool had values in the range [0.1,0.9] with 0.1 increments. The mutation strategy pool contained three strategies namely rand/1/bin, best/2/bin, and current-to-rand/1/bin. Initially every individual is randomly assigned a set of [F, Cr, Ms] and during the search successful sets are carried forward to the next generation while unsuccessful sets are re-initialized. The parameter Ms in the set denotes a mutation strategy.

In [49], the authors presented an adaptive scheme and also proposed a new mutation strategy current-to-$p$best/1. The scheme also included a diversity maintenance mechanism by keeping an optional external archive of unsuccessful parents that were unable to move to the next generation owing to their worse fitness. The mutation strategy current-to-$p$best/1 that the authors employed is different from the basic current-to-$p$best/1 strategy in the sense that the individual $p$best can be selected from a user controlled set of top individuals instead of representing just the top individual. The search mechanism of current-to-$p$best/1 is quite greedy in nature and experimentally, it has been shown that this greediness often leads to poor performance on multimodal functions [50]. In other words, the greediness of this basic mutation strategy can be controlled to some extent in the new version. The donor from current-to-$p$best/1 is obtained as

$$D_{i,G} = X_{i,G} + F_i \times (X_{pbest,G} - X_{i,G}) + F_i \times (X_{r1,G} - X_{r2,G}) \tag{3.4}$$

where the individual $x_{pbest,G}$ is randomly selected from the top $NP \times n$ ($n \in [0,1]$) members in the G-th generation. Here, n may be regarded as the greediness control operator. The authors adopted a memory based control parameter adaptation scheme. F and Cr were drawn from a normal N (μ_F,0.1) and a cauchy C (μ_{Cr},0.1) distribution respectively where μ_F and μ_{Cr} are the respective mean values of the distributions. At the beginning of the search, μ_F and μ_{Cr} are initialized to 0.5 and adapted thereafter as

$$\mu_{Cr} = (1-c)\mu_{Cr} + c.A(S_{Cr}) \tag{3.5}$$

$$\mu_F = (1-c)\mu_F + c.L(S_F) \tag{3.6}$$

where c is the learning rate which was suggested to be set to 0.1. S_F and S_{Cr} are the successful values stored in the memory during the generation. A and L are the arithmetic and Lehmer means, respectively.

Several extensions to JADE have been proposed. Authors in [51] propose a restart strategy for JADE and also suggest replacing the arithmetic mean in Equation 3.5 by a weighted mean, where higher weights are assigned to Cr values that achieve a higher fitness difference. A co-evolutionary extension to JADE was proposed in [52]. In [53], authors adaptively select the mutation strategy to be applied among current-to-$pbest$/1 with/without the external archive, and rand-to-$pbest$/1 with/without the external archive. JADE has also been successfully applied to combinatorial and multi-objective optimization problems [54], [55].

In another memory based parameter adaptation scheme called success history based adaptive DE (SHADE) [56], authors improve upon the robustness of JADE. They argue that the continuous mean update mechanism used in JADE may allow unfavorable values of F and CR to impact their mean value thereby allowing the possibility of a degraded search performance. They maintain a historical memory of means as M_F, and M_{Cr} which are the successful values of means calculated from S_F and S_{Cr}. In essence, SHADE maintains a pool of successful pairwise means in contrast with JADE, which works with a single pair of means. In case an unfavorable set of μ_F and μ_{Cr} are recorded, its impact would be far less profound as there may be other successful and favorable means in the pool to offset this disadvantage. SHADE was shown to have outperformed JADE in [57].

Apart from F and Cr, adaptation of population size NP has also received much attention. The population size significantly impacts the convergence rate of any evolutionary algorithm, with DE being no exception. A smaller value of population size, NP, tends to favor exploitation and the solution converges faster while always breaming with the possibility of getting stuck in a local minima. Large population sizes favor exploration of the landscape thereby slowing down the convergence rate. Many population resizing methods have been proposed that have shown to be effective in improving the performance of evolutionary algorithms [48], [58], [59], [60], [61]. These methods of population size methods are essentially deterministic instead of adaptive, as they increase or decrease the population size based on some predefined rules.

One of such techniques named Linear Population Size Reduction (LPSR) was incorporated by authors in [62] to enhance the performance of SHADE. Authors in [63] proposed jDEdynNP, a self adaptive version of DE, in which F and Cr are self adapted and a population size reduction technique is used. Population size adaptation has not been investigated in this work.

3.3. Experimentation And Results

To gauge the impact of strategy adaptation with and without adapting F and Cr, five basic mutation strategies are evaluated on 28 benchmark functions listed in [33] at problem dimensionality 10, 30, and 50, results of which are tabulated in Tables 3.1, 3.2, and 3.3, respectively, and are discussed in the next section.

3.3.1. Relative performance of basic strategies

Initially experiments were performed with five basic mutation strategies described below.

1. DE/rand/1/

$$X_i = X_{r1} + F \times (X_{r2} - X_{r3}) \tag{3.7}$$

2. DE/rand/2/

$$X_i = X_{r1} + F \times (X_{r2} - X_{r3}) + F \times (X_{r4} - X_{r5}) \tag{3.8}$$

3. DE/best/2/

$$X_i = X_{best} + F \times (X_{r1} - X_{r2}) + F \times (X_{r3} - X_{r4}) \tag{3.9}$$

4. DE/current-To-best/

$$X_i = X_{target} + F \times (X_{best} - X_{target}) + F \times (X_{r1} - X_{r2}) \tag{3.10}$$

5. DE/rand-to-best/

$$X_i = X_{r1} + F \times (X_{best} - X_{r2}) + F \times (X_{r3} - X_{r4}) \tag{3.11}$$

The first experiment led to a generally accepted result that different mutation strategies perform differently on different problems. Table 3.4 summarizes the relative performance of the basic mutation strategies as the ranks obtained by applying Friedman test [36] at problem dimensionality 10, 30, and 50.

It is clear that Rand/1 is relatively the most consistent strategy across problem dimensionality. The next question one might ask - "Is the performance of Rand/1 statistically significant?". The experimental results pertaining to this question are shown in Table 3.5 which contains the p-values (for α = 0.05) obtained by applying the Hochberg post hoc method [37] over the results of Table 3.1, 3.2, 3.3, respectively.

Table 3.1. Performance of Rand/1, Rand/2, Best/2, RandToBest, and CurrToBest at 10. Reported values are the averages of 51 independent runs for each function. Error values reaching within 10^{-8} of the global optimum of the function are reported as 0.00+E00.

Function	Rand/1	Rand/2	Best/2	RandToBest	CurrToBest
1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
2	**0.00E+00**	3.54E+00	3.40E-06	3.36E+04	9.03E+02
3	5.21E-01	5.23E+03	**1.38E-01**	1.30E+06	2.76E+05
4	**0.00E+00**	1.87E-02	2.26E-08	2.33E+02	3.04E+01
5	0.00E+00	0.00E+00	0.00E+00	6.33E-01	3.25E-04
6	4.50E-03	1.64E-06	**0.00E+00**	6.37E+00	7.66E+00
7	4.80E-04	1.29E+00	7.71E-02	5.07E-02	**7.39E-05**
8	2.04E+01	2.04E+01	2.04E+01	2.04E+01	2.04E+01
9	**1.93E-01**	6.75E+00	4.88E+00	8.65E-01	5.57E-01
10	3.60E-01	5.39E-01	5.47E-01	**5.16E-03**	1.07E-02
11	1.73E+01	2.49E+01	2.22E+01	**2.03E+00**	8.24E+00
12	2.59E+01	3.23E+01	3.21E+01	**1.02E+01**	1.54E+01
13	2.59E+01	3.14E+01	2.96E+01	**1.01E+01**	1.70E+01
14	1.02E+03	1.28E+03	1.24E+03	**7.88E+02**	1.01E+03
15	1.25E+03	1.36E+03	1.37E+03	**1.04E+03**	1.16E+03
16	9.99E-01	1.14E+00	1.15E+00	**9.48E-01**	9.69E-01
17	3.07E+01	3.99E+01	3.54E+01	**1.76E+01**	1.80E+01
18	3.58E+01	4.60E+01	4.19E+01	2.56E+01	**2.46E+01**
19	1.84E+00	2.62E+00	2.45E+00	**1.41E+00**	1.64E+00
20	2.54E+00	2.95E+00	2.65E+00	**1.95E+00**	2.09E+00
21	3.72E+02	**3.05E+02**	3.81E+02	4.00E+02	4.00E+02
22	9.47E+02	1.44E+03	1.29E+03	**3.39E+02**	9.11E+02
23	1.16E+03	1.39E+03	1.33E+03	**5.83E+02**	9.39E+02
24	**1.98E+02**	2.05E+02	2.05E+02	2.01E+02	2.01E+02
25	2.00E+02	2.00E+02	2.02E+02	2.00E+02	2.00E+02
26	1.26E+02	1.39E+02	1.46E+02	1.23E+02	**1.18E+02**
27	3.00E+02	3.07E+02	3.00E+02	3.05E+02	3.00E+02
28	**2.52E+02**	2.90E+02	2.52E+02	3.00E+02	3.00E+02

Table 3.2. Performance of Rand/1, Rand/2, Best/2, RandToBest, and CurrToBest at 30D. Reported values are the averages of 51 independent runs for each function. Error values reaching within 10^{-8} of the global optimum of the function are reported as 0.00+E00.

Function	Rand/1	Rand/2	Best/2	RandToBest	CurrToBest
1	0.00E+00	2.40E-01	0.00E+00	1.18E+02	5.95E+02
2	**5.09E+05**	3.72E+07	1.41E+06	4.21E+06	3.16E+06
3	**2.29E-02**	1.71E+09	5.26E+04	1.97E+09	1.91E+09
4	**9.98E+02**	3.54E+04	7.30E+03	4.56E+03	2.63E+03
5	**0.00E+00**	5.06E-01	4.83E-10	2.43E+02	4.21E+02
6	9.21E+00	1.80E+01	**6.26E+00**	1.04E+02	1.08E+02
7	**1.00E-01**	5.82E+01	9.91E+00	1.79E+01	1.15E+01
8	2.09E+01	2.09E+01	2.10E+01	2.09E+01	2.09E+01
9	2.25E+01	3.89E+01	3.84E+01	**8.68E+00**	1.04E+01
10	5.40E-03	2.55E+01	**1.39E-04**	5.80E+01	9.98E+01
11	1.23E+02	2.10E+02	1.87E+02	**2.06E+01**	8.89E+01
12	1.77E+02	2.26E+02	1.98E+02	**7.71E+01**	1.60E+02
13	1.72E+02	2.28E+02	1.98E+02	**1.37E+02**	1.66E+02
14	6.25E+03	6.81E+03	6.85E+03	**6.22E+03**	6.43E+03
15	7.12E+03	7.31E+03	7.22E+03	**6.57E+03**	6.84E+03
16	2.49E+00	**2.45E+00**	2.57E+00	2.48E+00	2.51E+00
17	1.83E+02	2.69E+02	2.23E+02	**1.59E+02**	1.67E+02
18	2.12E+02	2.83E+02	2.31E+02	**1.78E+02**	1.86E+02
19	**1.50E+01**	2.04E+01	1.73E+01	3.44E+01	3.43E+01
20	**1.21E+01**	1.27E+01	1.26E+01	1.21E+01	1.24E+01
21	**2.91E+02**	3.12E+02	3.10E+02	6.69E+02	6.67E+02
22	6.45E+03	6.95E+03	6.90E+03	**4.52E+03**	5.89E+03
23	7.14E+03	7.23E+03	7.08E+03	**6.10E+03**	6.68E+03
24	**2.00E+02**	2.70E+02	2.06E+02	2.17E+02	2.21E+02
25	**2.39E+02**	2.82E+02	2.51E+02	2.47E+02	2.45E+02
26	2.00E+02	2.03E+02	2.00E+02	2.00E+02	2.11E+02
27	**3.37E+02**	1.17E+03	5.93E+02	5.13E+02	4.64E+02
28	3.00E+02	3.25E+02	3.00E+02	3.25E+02	9.93E+02

Table 3.3. Performance of Rand/1, Rand/2, Best/2, RandToBest, and CurrToBest at 50D. Reported values are the averages of 51 independent runs for each function. Error values reaching within 10^{-8} of the global optimum of the function are reported as 0.00+E00.

Function	Rand/1	Rand/2	Best/2	RandToBest	CurrToBest
1	0.00E+00	7.13E+01	0.00E+00	1.44E+03	3.55E+03
2	**2.70E+06**	2.30E+08	2.23E+07	1.76E+07	1.07E+07
3	**3.36E+05**	3.15E+10	3.90E+06	8.89E+09	1.07E+10
4	2.10E+04	7.38E+04	4.32E+04	4.50E+03	**3.47E+03**
5	**0.00E+00**	1.84E+01	8.11E-09	5.56E+02	9.32E+02
6	**4.34E+01**	6.04E+01	4.34E+01	2.17E+02	2.54E+02
7	**1.03E+00**	1.24E+02	2.03E+01	2.86E+01	2.98E+01
8	**2.11E+01**	2.11E+01	2.11E+01	2.11E+01	2.11E+01
9	7.04E+01	7.23E+01	7.23E+01	**2.36E+01**	2.39E+01
10	4.06E-02	4.63E+02	**1.42E-02**	3.08E+02	4.17E+02
11	2.16E+02	4.32E+02	3.65E+02	**6.69E+01**	8.27E+01
12	3.61E+02	4.77E+02	3.86E+02	**7.87E+01**	2.34E+02
13	3.51E+02	4.79E+02	3.83E+02	**3.14E+02**	3.68E+02
14	**1.13E+04**	1.30E+04	1.30E+04	1.21E+04	1.26E+04
15	1.39E+04	1.39E+04	1.39E+04	**1.31E+04**	1.35E+04
16	3.33E+00	**3.17E+00**	3.32E+00	3.31E+00	3.36E+00
17	**3.30E+02**	5.45E+02	4.19E+02	3.48E+02	3.81E+02
18	4.01E+02	5.60E+02	4.43E+02	**3.75E+02**	3.97E+02
19	**2.97E+01**	4.93E+01	3.36E+01	4.54E+02	1.34E+03
20	2.21E+01	2.27E+01	2.24E+01	**2.06E+01**	2.07E+01
21	4.06E+02	4.31E+02	**2.74E+02**	2.06E+03	2.30E+03
22	1.08E+04	1.34E+04	1.32E+04	**3.88E+03**	1.19E+04
23	1.37E+04	1.39E+04	1.39E+04	**1.23E+04**	1.30E+04
24	**2.07E+02**	3.61E+02	2.14E+02	2.60E+02	2.68E+02
25	**2.78E+02**	3.81E+02	3.13E+02	3.31E+02	3.31E+02
26	**2.45E+02**	3.45E+02	3.76E+02	3.15E+02	2.90E+02
27	**5.71E+02**	2.04E+03	1.22E+03	8.96E+02	9.57E+02
28	4.00E+02	4.59E+02	4.00E+02	1.26E+03	1.53E+03

Table 3.4. Relative ranks obtained by Rand/1, Rand/2, Best/2, RandToBest, and CurrToBest at 10D, 30D, and 50D respectively.

Strategy	Rank-10D	Rank-30D	Rank-50D
Rand/1	**2.46**	**2.01**	**2.00**
Rand/2	3.97	4.14	4.23
Best/2	3.5	3.08	3.08
RandToBest	2.5	2.58	2.41
CurrToBest	2.57	3.16	3.26

Table 3.5. p values obtained using Hochberg procedure by mutation strategies Rand/2, Best/2, RandToBest and CurrToBest when compared to Rand/1 at 10D, 30D, and 50D respectively at α level 0.05.

Strategy	p_{Hoc}-10D	p_{Hoc}-30D	p_{Hoc}-50D
Rand/2	0.001	0.000	0.000
Best/2	0.042	0.020	0.019
RandToBest	**0.932**	**0.176**	**0.331**
CurrToBest	**0.932**	0.022	0.008

The results in Table 3.5 show that Rand/1 significantly outperforms Rand/2 and Best/2 at every problem dimensionality. At 10 dimensions there is not much of a performance difference between Rand/1, RandToBest, and CurrToBest. At 30 dimensions though, Rand/1 significantly outperforms CurrToBest, and RandToBest remains the only competitive strategy against Rand/1, and this observation is repeated at 50 dimensions.

The first inference that can be drawn from these results is that for the given number of function evaluations and in the absence of control parameter adaptation, Rand/1 remains the most competitive strategy across problem dimensionality, and the relative competitiveness of Rand/1 against Rand/2, Best/2, and CurrToBest improves with an increase in problem dimensionality.

The performance of Best2 relative to CurrToBest improves as problem dimensionality increases as it is ranked 3 at 50 dimensions as compared to 4 at 30 dimensions. Rand2 turns out to be the worst performing strategy at every problem dimensionality.

To ascertain the relative competitiveness of RandToBest and CurrToBest at 30D and 50D, the Wilcoxon test was performed, and the results are presented in Table 3.6.

Table 3.6. Results obtained by the Wilcoxon test for strategy Rand2Best against CurrToBest

Problem Dimensionality	Asymptotic P-value
30	0.070897
50	0.001324

Table 3.6 shows that Rand2Best turns out to be a better performing strategy, significantly outperforming CurrToBest at 50 (α=0.05) and 30 (α=0.01) dimensions.

3.3.2. A case for strategy adaptation irrespective of parameter adaptation

Looking at the above results, one may surmise that Rand1 is relatively the most robust strategy, and can be used with confidence while optimizing with DE. This observation, however, requires greater scrutiny. Table 3.7 shows the number of wins scored by all the mutation strategies at 10, 30, and 50 dimensions. Functions on which multiple strategies score equally are not counted as wins.

Table 3.7. Number of wins scored, out of 28, by all mutation strategies at 10, 30, and 50 dimensions, respectively.

Dimensions	Rand1	Rand2	Best2	RandToBest	CurrToBest
10	5	2	1	10	5
30	10	1	2	10	0
50	11	1	2	9	2

Cumulative results presented in Tables 3.5 and 3.7 coupled with works reported in [18], [34] are indicative that no single strategy has the ability to perform relatively better on all the problems. This is evidence that calls for automating the selection of mutation strategies during the course of the search.

It must be noted that several works in the past [50], [64], [65] have evaluated multiple variants of DE on various real life and benchmark functions, and have arrived at seemingly contrasting results, possibly due to varying test subjects and problems. For example, authors in [64] reported that DE/Best/* variants perform much better than DE/Rand/* variants on the problem of optimal design of shell-and-heat tube exchangers. This of-course is the opposite of what is reported in this work. On similar lines, authors in [65] report superior performance of DE/Rand/1/bin.

All in all, to circumvent these contrasting claims, it would be prudent to let the mutation strategy adapt during the search operation. Strategy adaptive variants proposed in the past [18], [34], [66], [67], [68], have reported favorable results.

3.3.3. Impact of mutation strategy on adaptive control parameter models

To ascertain the importance of mutation strategy employed while using adaptive control parameters (F and Cr) models, experiments were performed on the same test suite, but this time a control parameter adaptive model proposed in [56] was used. The choice of this model was based on the superior performance this demonstrated over other adaptive models as is evident from the work in [56]. After making this choice, the following questions were asked.

1. What impact does a mutation strategy has on the search direction when the control parameters are adapted?

2. Is the impact profound enough to necessitate automated strategy selection?

To answer the first question the adaptive model was tested with multiple mutation strategies at problem dimensionality 10, 30, and 50, the results of which are presented in Tables 3.8, 3.9, and 3.10, respectively. Table 3.11 and 3.12 show the ranks and p values obtained by the Friedman and Hochberg test respectively, for the tested mutation strategies at problem dimensionality 10, 30, and 50, respectively.

It is worth noting that SHADE used CurrentTopBest as the mutation strategy that used an external archive of inferior solutions, and draws the $p$best vector from the top $x\%$ individuals in the population. Results from Tables 3.11 and 3.12 show that when a control parameter adaptation model is used, the strategy used in SHADE is ranked at the top of the strategies tested, and proves vastly superior to Rand1, Rand2, and Best2 at every problem dimensionality. It, however, is not significant when compared to CurrentToBest and RandToBest as is clear from the p values shown in Table 3.12.

There are two important inferences that can be drawn from these results.

- With or without a given parameter adaptation model, the choice of mutation strategy plays an important role in determining the quality of solutions.

- RandToBest and CurrentToBest tend to perform relatively better that Rand1, which according to the results tabulated earlier, was the most robust strategy in absence of parameter adaptation.

Table 3.8. Performance of Rand/1, Rand/2, Best/2, RandToBest, CurrentToBest, and SHADE at 10D when employed with adaptive control parameter model used in SHADE. Reported values are the averages of 51 independent runs for each function. Error values reaching within 10^{-8} of the global optimum of the function are reported as 0.00+E00.

F	Rand/1	Rand/2	Best/2	RandToBest	CurrToBest	SHADE
1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
2	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
3	2.30E+01	1.39E+04	**2.08E-05**	6.11E-01	3.13E-01	1.27E-01
4	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
5	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
6	8.88E+00	8.41E+00	9.81E+00	9.35E+00	8.41E+00	**7.89E+00**
7	3.18E-01	9.09E-01	1.16E-04	**2.60E-05**	4.26E-02	3.26E-03
8	2.04E+01	2.04E+01	2.04E+01	2.03E+01	2.03E+01	2.04E+01
9	4.16E+00	4.81E+00	4.01E+00	3.64E+00	3.48E+00	**3.39E+00**
10	4.74E-02	6.42E-02	6.42E-02	1.70E-02	**1.02E-02**	1.20E-02
11	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
12	8.48E+00	1.05E+01	9.00E+00	5.27E+00	3.55E+00	**3.14E+00**
13	1.20E+01	1.31E+01	1.50E+01	7.48E+00	4.98E+00	**3.77E+00**
14	0.00E+00	2.73E-02	2.83E-04	5.95E-03	0.00E+00	4.90E-03
15	7.21E+02	7.19E+02	7.01E+02	6.27E+02	4.57E+02	**4.21E+02**
16	1.01E+00	1.16E+00	1.05E+00	8.92E-01	**4.99E-01**	7.08E-01
17	1.01E+01	**1.00E+01**	1.01E+01	1.01E+01	1.01E+01	1.01E+01
18	2.29E+01	2.17E+01	1.88E+01	1.78E+01	1.77E+01	**1.69E+01**
19	4.11E-01	4.66E-01	4.30E-01	3.65E-01	**3.32E-01**	3.44E-01
20	2.54E+00	2.80E+00	2.32E+00	2.24E+00	2.38E+00	**2.16E+00**
21	4.00E+02	**3.53E+02**	3.81E+02	4.00E+02	4.00E+02	4.00E+02
22	1.70E+01	5.86E+01	3.80E+01	5.58E+00	1.06E+01	**4.84E+00**
23	7.71E+02	8.13E+02	6.94E+02	5.98E+02	5.42E+02	**4.61E+02**
24	2.06E+02	1.98E+02	2.06E+02	2.00E+02	1.93E+02	**1.93E+02**
25	2.01E+02	2.00E+02	2.03E+02	2.00E+02	2.00E+02	2.00E+02
26	1.12E+02	1.22E+02	1.24E+02	1.07E+02	**1.05E+02**	1.33E+02
27	3.00E+02	3.30E+02	3.00E+02	3.00E+02	3.00E+02	3.00E+02
28	3.00E+02	3.00E+02	3.00E+02	3.00E+02	3.00E+02	3.00E+02

Table 3.9. Performance of Rand/1, Rand/2, Best/2, RandToBest, and CurrentToBest at 30D when employed with adaptive control parameter model used in SHADE. Reported values are the averages of 51 independent runs for each function. Error values reaching within 10^{-8} of the global optimum of the function are reported as 0.00+E00.

F	Rand/1	Rand/2	Best/2	RandToBest	CurrToBest	SHADE
1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
2	6.89E+06	1.72E+07	1.11E+07	3.78E+04	1.77E+04	**9.00E+03**
3	4.36E+06	7.06E+07	5.89E+05	1.37E+05	1.31E+05	**4.02E+01**
4	2.55E+04	2.25E+04	1.61E+04	2.45E+03	1.64E-01	**1.92E-04**
5	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
6	1.42E+01	1.49E+01	1.23E+01	1.47E+01	8.42E+00	**5.96E-01**
7	2.43E+01	3.59E+01	5.09E+00	**1.97E+00**	3.18E+00	4.60E+00
8	2.09E+01	2.10E+01	2.09E+01	2.06E+01	2.06E+01	2.07E+01
9	2.79E+01	2.81E+01	2.80E+01	**2.74E+01**	2.74E+01	2.75E+01
10	1.23E+00	1.00E+01	8.69E-02	1.72E-01	1.07E-01	**7.69E-02**
11	0.00E+00	0.00E+00	3.79E-01	0.00E+00	0.00E+00	0.00E+00
12	6.28E+01	6.13E+01	4.54E+01	2.52E+01	2.44E+01	**2.30E+01**
13	9.42E+01	9.67E+01	8.57E+01	5.47E+01	5.20E+01	**5.03E+01**
14	1.09E-02	3.60E+00	1.55E+00	2.08E-02	**8.92E-03**	3.18E-02
15	4.72E+03	4.96E+03	4.57E+03	4.38E+03	3.22E+03	3.22E+03
16	1.43E+00	1.96E+00	1.61E+00	**6.09E-01**	9.22E-01	9.13E-01
17	3.04E+01	3.04E+01	3.04E+01	3.04E+01	3.04E+01	3.04E+01
18	1.11E+02	1.42E+02	8.99E+01	**6.89E+01**	7.36E+01	7.25E+01
19	1.79E+00	1.92E+00	1.80E+00	1.52E+00	1.38E+00	**1.36E+00**
20	1.17E+01	1.19E+01	1.10E+01	**1.04E+01**	1.07E+01	1.05E+01
21	**2.48E+02**	2.74E+02	2.81E+02	2.95E+02	2.97E+02	3.09E+02
22	**8.19E+01**	2.07E+02	1.52E+02	1.08E+02	9.79E+01	9.81E+01
23	4.83E+03	5.16E+03	4.44E+03	4.57E+03	3.74E+03	**3.51E+03**
24	2.38E+02	2.66E+02	2.28E+02	**2.02E+02**	2.05E+02	2.05E+02
25	2.84E+02	2.87E+02	2.85E+02	2.69E+02	2.82E+02	**2.59E+02**
26	2.06E+02	2.01E+02	2.01E+02	2.15E+02	2.02E+02	2.02E+02
27	9.85E+02	1.01E+03	9.62E+02	**3.26E+02**	4.60E+02	3.88E+02
28	3.00E+02	3.00E+02	3.00E+02	3.00E+02	3.00E+02	3.00E+02

Table 3.10. Performance of Rand/1, Rand/2, Best/2, RandToBest, and CurrentToBest at 50D when employed with adaptive control parameter model used in SHADE. Reported values are the averages of 51 independent runs for each function. Error values reaching within 10^{-8} of the global optimum of the function are reported as 0.00+E00.

F	Rand/1	Rand/2	Best/2	RandToBest	CurrToBest	SHADE
1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
2	3.13E+07	4.93E+07	2.71E+07	1.79E+05	8.70E+04	**2.66E+04**
3	5.14E+08	2.87E+09	2.85E+06	2.67E+06	**6.84E+05**	8.80E+05
4	6.14E+04	6.13E+04	3.02E+04	1.24E+04	5.21E-01	**1.61E-03**
5	0.00E+00	0.00E+00	0.00E+00	6.92E-09	0.00E+00	0.00E+00
6	4.35E+01	4.35E+01	4.34E+01	4.38E+01	4.36E+01	**4.28E+01**
7	6.15E+01	8.33E+01	2.43E+01	**1.62E+01**	2.58E+01	2.33E+01
8	2.11E+01	2.11E+01	2.11E+01	2.09E+01	**2.08E+01**	2.09E+01
9	5.57E+01	5.63E+01	5.52E+01	**5.47E+01**	5.56E+01	5.54E+01
10	1.27E+01	5.63E+01	1.71E-01	2.50E-01	1.28E-01	**7.37E-02**
11	0.00E+00	4.74E-02	4.03E+00	0.00E+00	0.00E+00	0.00E+00
12	1.24E+02	1.48E+02	8.25E+01	6.46E+01	6.09E+01	**5.86E+01**
13	2.03E+02	2.07E+02	1.53E+02	**1.40E+02**	1.50E+02	1.45E+02
14	**8.33E-03**	1.76E+01	1.25E+01	3.69E-02	1.61E-02	3.45E-02
15	9.22E+03	9.03E+03	9.22E+03	8.79E+03	7.04E+03	**6.82E+03**
16	2.04E+00	2.12E+00	1.81E+00	1.42E+00	**1.22E+00**	1.28E+00
17	5.08E+01	5.08E+01	5.08E+01	5.08E+01	5.08E+01	5.08E+01
18	2.33E+02	2.64E+02	1.49E+02	**1.16E+02**	1.33E+02	1.37E+02
19	3.26E+00	3.59E+00	3.24E+00	2.77E+00	2.74E+00	**2.64E+00**
20	2.11E+01	2.14E+01	2.07E+01	**1.92E+01**	1.97E+01	1.93E+01
21	4.94E+02	**3.57E+02**	5.54E+02	8.15E+02	9.66E+02	8.45E+02
22	3.28E+01	2.35E+02	9.92E+01	**1.24E+01**	1.50E+01	1.33E+01
23	9.79E+03	1.04E+04	9.28E+03	8.58E+03	8.07E+03	**7.63E+03**
24	3.22E+02	3.40E+02	2.93E+02	**2.21E+02**	2.30E+02	2.34E+02
25	3.72E+02	3.74E+02	3.71E+02	3.54E+02	3.74E+02	**3.40E+02**
26	2.70E+02	2.28E+02	3.07E+02	3.08E+02	**2.06E+02**	2.58E+02
27	1.67E+03	1.71E+03	1.65E+03	**6.99E+02**	9.57E+02	9.36E+02
28	4.00E+02	4.00E+02	5.42E+02	4.00E+02	4.00E+02	4.58E+02

Table 3.11. Relative ranks obtained by Rand/1, Rand/2, Best/2, RandToBest, CurrToBest, and SHADE at 10D, 30D, and 50D.

Strategy	Rank-10D	Rank-30D	Rank-50D
Rand/1	4.17	4.16	4.30
Rand/2	4.50	5.19	5.00
Best/2	4.05	4.03	4.00
RandToBest	3.16	2.82	2.71
CurrToBest	2.57	2.46	2.71
SHADE	**2.53**	**2.32**	**2.26**

Table 3.12. p values obtained using Hochberg procedure by Rand1/1, Rand/2, Best/2, RandToBest, and CurrToBest when compared with SHADE at 10D, 30D, and 50D at α level 0.05.

Strategy	p_{Hoc}-10D	p_{Hoc}-30D	p_{Hoc}-50D
Rand/1	0.004	0.000	0.000
Rand/2	0.000	0.000	0.000
Best/2	0.000	0.000	0.019
RandToBest	**0.422**	**0.317**	**0.371**
CurrToBest	**0.943**	**0.775**	**0.371**

Table 3.13 shows the number of wins scored by each strategy and provides some insights into investigating the plausibility of automated strategy selection vis-a-vis control parameter adaptation.

Table 3.13. Number of wins scored, out of 28, by all mutation strategies and SHADE at 10, 30, and 50 dimensions, respectively.

D	Rand1	Rand2	Best2	RandToBest	CurrToBest	SHADE
10	0	2	0	2	5	7
30	2	0	0	6	3	11
50	1	1	0	7	4	8

As is clear from the results in Table 3.12 and 3.13, even though the strategy used in SHADE is highest ranked, there is a possibility of further improving SHADE by automating the selection of mutation strategy as the cumulative wins scored by all strategies at every problem dimensionality is greater than the wins recorded by SHADE.

3.4. SA-SHADE Algorithm

3.4.1. SA-SHADE and its characteristic differences compared to SHADE

Motivated by the results presented in the previous section, we investigated the benefits of plugging a strategy adaptation in the original SHADE algorithm.

SA-SHADE, a memory based adaptive version of Differential Evolution is proposed wherein F, Cr, and mutation strategy are adapted during the search process. The basis of this chapter is the SHADE algorithm [56], which adapts F and Cr but uses a fixed mutation strategy which is current-to-$p$best/1 with an optional external archive that was originally used in [49].

SA-SHADE and SHADE differ on three aspects.

- SA-SHADE adapts the mutation strategy during the search process, while SHADE uses a single mutation strategy throughout the search process.
- SA-SHADE uses the mode of successful mutation strategies to update its memory, which is slightly different from the way F and Cr are adapted in SHADE.
- The learned memory of successful strategies in SA-SHADE is wiped out after a certain number of function evaluations which is determined by the reset rate R. In SHADE on the other hand, such reset is not performed for updating the memory of successful F and Cr values.

The operation of SA-SHADE is described as follows. First, the mutation strategies to be included in the pool P are selected. Then an integer vector M_{Ms}, the memory containing successful mutation strategies, is initialized, length of which is user controlled, with randomly selected mutation strategies from the pool, P. All of the mutation strategies in P are included in M_{Ms} at least once. Then, every vector is allowed to randomly choose a mutation strategy from the memory M_{Ms}. For every successful individual, just like F and Cr, we record, in another integer vector S_{Ms}, the successful mutation strategies over a generation. Then, the most successful strategy of the generation, given by the mode of S_{Ms}, is stored in the memoryM_{Ms}. This operation is repeated until a number of function evaluations are reached after which the memory M_{Ms} is re-initialized. This resetting of the memory is done to disallow any probabilistic bias created by the system towards a particular mutation strategy, and we show that this indeed proves useful. At the same time, it can be argued that the same reset scheme can be applied to F and Cr but our experiments show that this proves

Algorithm 2 PSEUDO-CODE FOR SA-SHADE

```
Set generation number, G=0
Set memory size $M$, reset rate $R$=0.2, counter $k$=1, and initialize external archive $A = \emptyset$
Initialize the mutation strategy pool $P_{Ms}$
Initialize a population of $NP$ individuals $P = [X_1, X_2, ...X_{NP}]$ where every $i^{th}$ individual is a $D$ dimensional vector represented as $X_i^j$=[$x_i^1$, $x_i^2$ ... $x_i^D$] where $1 \leq j \leq D$. Restrict $x_i^j$ to its minimum and maximum bounds as $x_{i,min}^j$ and $x_{i,max}^j$
Set all values in memory $M_{Cr}, M_F$ to 0.5 and randomly initialize $M_{Ms}$ with mutation strategies from the pool $P_{Ms}$
while stopping criteria is not met do
    $S_{Cr} = \emptyset$ $S_F = \emptyset$; $S_{Ms} = \emptyset$;
    for every target vector $X_i$ in $P$ do
        Select a random integer $r$ from [1,$M$]
        Draw $F$ from a cauchy distribution as $C(M_{F,r}, 0.1)$
        Draw $Cr$ from a normal distribution as $N(M_{Cr,r}, 0.1)$
        Choose a mutation strategy from $M_{Ms,r}$
        Produce a trial vector $V_i^G$, using the control parameters generated and the strategy selected above
        Select either the target vector or the trial vector based on their fitness values as:
            $$X_i^{G+1} = \begin{cases} V_i^G & \text{if } f(V_i^G) \leq f(X_i^G) \\ X_i^G & \text{otherwise} \end{cases}$$
        //update external archive $A$
        if ($f(V_i^G) \leq f(X_i^G)$) then
            Add $X_i^G$ to external archive $A$
        end if
    end for
    //check for mutation memory reset
    if $G = k \times (R \times G_{max})$ then
        Randomly initialize $M_{Ms}$ with mutation strategies from the pool $P_{Ms}$
    end if
    Increase the generation count $G$ to $G + 1$
    Increase the counter $k$ to $k + 1$
end while
```

counterproductive in most of the cases. Hence, this work has steered clear of using this reset method on F and Cr. SA-SHADE is summarized in Algorithm 2.

3.4.2. Choice of mutation strategies used in SA-SHADE

As proven in Section III-B, the choice of mutation strategy has a significant impact on the solution quality. A good mutation strategy is problem dependent, i.e., for the one which is successful on one landscape may prove adverse on others. There are some characteristics associated with every mutation strategy that may justify its use or otherwise. For example, double difference vector strategies like DE/rand/2/bin and DE/best/2/bin exhibit better diversity than DE/rand/1/bin and DE/best/1/bin [13], [15], [39], [40], making them more suitable on landscapes riddled with local minima. Strategies that use the best individual to generate mutant like DE/best/1/bin and DE/rand-to-best/1/bin tend to be greedy and score well on unimodal problems but their performance worsens on difficult and highly multimodal problems. A rotationally invariant strategy, DE/current-to-rand/1/, tends to do better on rotated problems [69]. The scheme DE/target-to-best/1/bin with neighborhood search proposed in [70], provides a good balance between exploration and exploitation.

Our pool P_{Ms} was obviously designed to contain the mutation strategies with diverse capabilities. The following strategies were chosen for the listed reasons.

1. DE/rand/1/: Most widely used, less greedy but robust.

2. DE/rand/2/: Even though it has a poor record of achieving good solutions, it has the ability to improve the diversity of population as it is capable of generating more trial vectors due to presence of two difference vectors [9], [13].

3. DE/best/2/: Greedy but also has the ability of diversity improvement as it utilizes two different vectors [9], [13].

4. DE/current-To-$pbest$WithArchive/: Proposed in [49] and used in [56] which is the basis of SA-SHADE.

5. DE/current-rand-to-$pbest$/: A new mutation strategy that we experimented with, that uses the target vector as the base vector, a difference of one of the top 20% of best vectors and a randomly chosen

vector, and another difference vector of two randomly chosen vectors. It has proven to be unstable sometimes but has the capability to negotiate local minima.

$$X_i = X_{target} + F \times (X_{pbest} - X_{r1}) + F \times (X_{r2} - X_{r3}) \tag{3.12}$$

This works incorporates a memory based adaptation mechanism into SA-SHADE on similar lines as the memory based adaptation of F and Cr. SHADE does not adapt the mutation strategy but only F and Cr.

3.4.3. Results

The comparative results of SA-SHADE with other variants at 30 dimensions are shown in Table 3.14. Table 3.15 lists the rank and p values obtained by SA-SHADE. It is clear that SA-SHADE is the top ranked algorithm among all the algorithms compared. SA-SHADE is proven to be statistically significant compared to AD-Rand/1 (α = 0.05), AD-Rand2 (α = 0.05), AD-Best2 (α = 0.05), AD-RandToBest (α = 0.05), and AD-CurrToBest (α = 0.1) while being highly competitive compared to SHADE. The prefix AD denotes that the algorithm uses the adaptive control parameter mechanism. While in previous results, SHADE was not found to be better than Ad-RandToBest and AD-CurrToBest at any significance level, SA-SHADE improves upon SHADE and shows superior results.

Table 3.17 shows the relative performance of SHADE with recently proposed state-of-the-art adaptive DE mechanisms. It is clear that SHADE, apart from EPSDE, is not statistically significant when compared to other algorithms when compared at α = 0.05 or 0.1. SA-SHADE, with the results listed in Table 3.18, improves over SHADE and shows statistically significant performance against EPSDE, CoDE, and dynNP-jDE while being highly competitive against JADE and SHADE.

The superior performance of SA-SHADE can be attributed to the strategy adaptation module as the underlying control parameter adaptation mechanism remains the same as SHADE. Determining the contribution of memory reset used in strategy adaptation is a matter of further investigation.

3.5. Summary

In this chapter, the plausibility of integrating a strategy adaptation mechanism with a parameter adaptation mechanism is investigated. Given fixed control parameter settings, it is demonstrated, on a test suite of 28 benchmark functions, that no single mutation strategy performs significantly better than all other

Table 3.14. Performance of parameter adaptive Rand/1, Rand/2, Best/2, RandToBest, and CurrentToBest against SHADE, and SA-SHADE at 30D. Reported values are the averages of 51 independent runs for each function. Error values reaching within 10^{-8} of the global optimum of the function are reported as 0.00+E00.

F	Rand/1	Rand/2	Best/2	RandToBest	CurrToBest	SHADE	SA-SHADE
1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
2	6.89E+06	1.72E+07	1.11E+07	3.78E+04	1.77E+04	9.00E+03	**8.15E+03**
3	4.36E+06	7.06E+07	5.89E+05	1.37E+05	1.31E+05	**4.02E+01**	1.19E+05
4	2.55E+04	2.25E+04	1.61E+04	2.45E+03	1.64E-01	**1.92E-04**	3.10E-02
5	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
6	1.42E+01	1.49E+01	1.23E+01	1.47E+01	8.42E+00	5.96E-01	**0.00E+00**
7	2.43E+01	3.59E+01	5.09E+00	**1.97E+00**	3.18E+00	4.60E+00	3.06E+00
8	2.09E+01	2.10E+01	2.09E+01	2.06E+01	2.06E+01	2.07E+01	2.07E+01
9	2.79E+01	2.81E+01	2.80E+01	2.74E+01	2.74E+01	2.75E+01	**2.69E+01**
10	1.23E+00	1.00E+01	8.69E-02	1.72E-01	1.07E-01	7.69E-02	**6.13E-02**
11	0.00E+00	0.00E+00	3.79E-01	0.00E+00	0.00E+00	0.00E+00	0.00E+00
12	6.28E+01	6.13E+01	4.54E+01	2.52E+01	2.44E+01	2.30E+01	**1.82E+01**
13	9.42E+01	9.67E+01	8.57E+01	5.47E+01	5.20E+01	5.03E+01	**3.84E+01**
14	1.09E-02	3.60E+00	1.55E+00	2.08E-02	**8.92E-03**	3.18E-02	6.34E-01
15	4.72E+03	4.96E+03	4.57E+03	4.38E+03	3.22E+03	**3.22E+03**	3.24E+03
16	1.43E+00	1.96E+00	1.61E+00	**6.09E-01**	9.22E-01	9.13E-01	1.01E+00
17	3.04E+01	3.04E+01	3.04E+01	3.04E+01	3.04E+01	3.04E+01	3.04E+01
18	1.11E+02	1.42E+02	8.99E+01	**6.89E+01**	7.36E+01	7.25E+01	7.17E+01
19	1.79E+00	1.92E+00	1.80E+00	1.52E+00	1.38E+00	1.36E+00	**1.33E+00**
20	1.17E+01	1.19E+01	1.10E+01	1.04E+01	1.07E+01	1.05E+01	**1.03E+01**
21	**2.48E+02**	2.74E+02	2.81E+02	2.95E+02	2.97E+02	3.09E+02	2.81E+02
22	**8.19E+01**	2.07E+02	1.52E+02	1.08E+02	9.79E+01	9.81E+01	9.75E+01
23	4.83E+03	5.16E+03	4.44E+03	4.57E+03	3.74E+03	**3.51E+03**	3.58E+03
24	2.38E+02	2.66E+02	2.28E+02	2.02E+02	2.05E+02	2.05E+02	**2.01E+02**
25	2.84E+02	2.87E+02	2.85E+02	2.69E+02	2.82E+02	**2.59E+02**	2.80E+02
26	2.06E+02	2.01E+02	2.01E+02	2.15E+02	2.00E+02	2.02E+02	2.00E+02
27	9.85E+02	1.01E+03	9.62E+02	**3.26E+02**	4.60E+02	3.88E+02	4.11E+02
28	3.00E+02	3.00E+02	3.00E+02	3.00E+02	3.00E+02	3.00E+02	3.00E+02

Table 3.15. Relative ranks obtained by Rand/1, Rand/2, Best/2, RandToBest, CurrToBest, SHADE, and SA-SHADE at 30D.

Strategy	Rank	p_{Hoc}
AD-Rand/1	4.17	0.00
AD-Rand/2	4.50	0.00
AD-Best/2	4.05	0.00
AD-RandToBest	3.16	0.03
AD-CurrToBest	2.57	0.06
SHADE	2.53	0.26
SASHADE	2.28	-

mutation strategies. Then, it is shown that similar results are observed in the presence of control parameter adaption. This built the case of automating the mutation strategies with or without control parameter adaptation. After that, a strategy adaptation mechanism is incorporated into a well known history based parameter adaptation mechanism, SHADE. The enhanced version SA-SHADE is then compared with other well known adaptive mechanisms and the superior results obtained by SA-SHADE are shown. SA-SHADE performs significantly better than well known adaptive variants, i.e., CoDE, EPSDE, and dynNP-jDE and is highly competitive compared to SHADE, and JADE. SHADE, though being superior, was not found to be statistically significantly different when compared with CoDE and dynNP-jDE. Thus, SA-SHADE improves upon SHADE in this regard. Another important conclusion that can be drawn from the results is that strategy adaptation is a useful mechanism both in presence and absence of control parameter adaptation, and we propose that it should be used while optimizing with DE.

Future work includes investigating the impact of different adaptive strategies on multiple classes of benchmark functions and classifying strengths and weaknesses of each mechanism accordingly. Further to that, the utility of a population size adaptation mechanism into SA-SHADE is also proposed as future work.

Table 3.16. Relative performance of SA-SHADE against state-of-the-art adaptive variants of DE at 30D. Reported values are the averages of 51 independent runs for each function. Error values reaching within 10^{-8} of the global optimum of the function are reported as 0.00+E00.

F	SA-SHADE	SHADE	CoDE	EPSDE	JADE	dynNP-jDE
F1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F2	8.15E+03	9.00E+03	9.78E+04	1.37E+06	**7.67E+03**	9.52E+04
F3	1.19E+05	**4.02E+01**	1.08E+06	1.75E+08	4.71E+05	1.71E+06
F4	3.10E-02	**1.92E-04**	8.18E-02	8.08E+03	6.09E+03	4.76E+01
F5	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F6	**0.00E+00**	5.96E-01	4.16E+00	9.27E+00	2.07E+00	1.19E+01
F7	3.06E+00	4.60E+00	9.32E+00	5.88E+01	3.16E+00	**2.62E+00**
F8	2.07E+01	2.07E+01	2.08E+01	2.09E+01	2.09E+01	2.10E+01
F9	2.69E+01	2.75E+01	**1.45E+01**	3.50E+01	2.65E+01	2.20E+01
F10	6.13E-02	7.69E-02	**2.71E-02**	1.02E-01	4.04E-02	3.63E-02
F11	0.00E+00	0.00E+00	0.00E+00	1.95E-02	0.00E+00	0.00E+00
F12	**1.82E+01**	2.30E+01	3.98E+01	4.94E+01	2.29E+01	4.07E+01
F13	**3.84E+01**	5.03E+01	8.04E+01	7.68E+01	4.67E+01	7.10E+01
F14	6.34E-02	3.18E-02	3.60E+00	3.99E-01	2.86E-02	**9.39E-03**
F15	3.24E+03	**3.22E+03**	3.36E+03	6.75E+03	3.24E+03	4.39E+03
F16	1.01E+00	9.13E-01	**3.38E-01**	2.48E+00	1.84E+00	2.32E+00
F17	3.04E+01	3.04E+01	3.04E+01	3.04E+01	3.04E+01	3.04E+01
F18	7.17E+01	7.25E+01	**6.69E+01**	1.37E+02	7.76E+01	1.35E+02
F19	1.33E+00	1.36E+00	1.61E+00	1.84E+00	1.44E+00	**1.27E+00**
F20	**1.03E+01**	1.05E+01	1.06E+01	1.30E+01	1.04E+01	1.13E+01
F21	**2.81E+02**	3.09E+02	3.02E+02	3.05E+02	3.04E+02	2.94E+02
F22	9.75E+01	9.81E+01	1.17E+02	3.09E+02	**9.39E+01**	1.03E+02
F23	3.58E+03	3.51E+03	3.56E+03	6.74E+03	**3.36E+03**	4.36E+03
F24	**2.01E+02**	2.05E+02	2.21E+02	2.91E+02	2.17E+02	2.04E+02
F25	2.80E+02	2.59E+02	2.57E+02	2.99E+02	2.74E+02	**2.55E+02**
F26	2.00E+02	2.02E+02	2.18E+02	3.56E+02	2.15E+02	2.00E+02
F27	4.11E+02	**3.88E+02**	6.20E+02	1.21E+03	6.70E+02	3.90E+02
F28	3.00E+02	3.00E+02	3.00E+02	3.00E+02	3.00E+02	3.00E+02

Table 3.17. Relative ranks and p values obtained by SHADE against CoDE, EPSDE, JADE, and dynNP-jDE at 30D.

Algorithm	Rank-10D	p value
SHADE	**2.30**	–
CoDE	2.91	**0.15**
JADE	2.50	**0.64**
dynNP-jDE	2.76	**0.27**
EPSDE	4.51	0.00

Table 3.18. Relative ranks and p values obtained by SA-SHADE against SHADE, JADE, dynNP-jDE, CoDE, and EPSDE at 30D.

Algorithm	Rank-10D	p value
SA-SHADE	**2.46**	–
SHADE	2.92	**0.35**
JADE	3.14	**0.15**
dynNP-jDE	3.41	0.05
CoDE	3.60	0.02
EPSDE	5.44	0.00

4. A DIFFERENTIAL EVOLUTION BASED AXLE DETECTOR FOR ROBUST VEHICLE CLASSIFICATION

Video based vehicle classification is gaining huge grounds due to its low cost and satisfactory accuracy. This paper presents a robust vehicle classification system. The system in its essence, aims to classify a vehicle based on the number of circles (axles) in an image using Hough Transform which is a popular parameter based feature detection method. The system consists of four modules whereby the output of one module feeds the next in line. We test our system on single lane highway and street traffic. When the information about the problem at hand (changing weather conditions, camera calibration parameters etc.) is limited or is dynamic, determining the Hough Transform set-up parameters manually becomes time consuming, challenging, and may often lead to false detections. This calls for finding the appropriate parameter-set dynamically according to the situation, which inherently is a global optimization problem. Differential Evolution has emerged as a simple and efficient global optimizer, and we couple it with Hough Transform to improve the overall accuracy of the classification system. Five different variants of DE are tested on varied videos, and a performance profile of all the variants is provided. The results demonstrate that employing DE indeed improves the system's classification accuracy (at the expense of extra compute cycles) making the system more reliable and robust.

The rest of this chapter is structured as follows. Section 4.1 presents an introduction to the chapter. Section 4.2 describes the related work. Section 4.3 outlines and explains the proposed classification system with all its features. In Section 4.4, results and their analysis are presented, and Section 4.5 concludes the chapter.

4.1. Introduction

Automatic vehicle classification has emerged as a significantly important element in the myriad web of traffic data collection and statistics. Regulations on road side construction for pertinent reasons, increasing vehicle density, and cost of overlaying roads are some of the factors calling for ever more efficient utilization of our existing transportation networks. A part of the solution to these pressures lies in vehicle classification systems that compute the number and type of vehicles passing a particular street or highway.

This information has an evident impact on the cost and efficiency of the transportation system; road thickness decision being one of the many advantages this system has to offer. Many video based classification systems have been proposed in the past with their own advantages and disadvantages. These systems can be primarily distinguished by the type of sensors they use, most common of which are magnetic, laser, pressure, single or multiple cameras, etc. Magnetic and laser sensors tend to have a higher classification accuracy but at the same time have high equipment and installation costs, and are intrusive techniques. Computer vision based vehicle classification systems are generally attributed with low cost and accuracy, and are an active area of research. We propose a video based vehicle classification system that determines the type of vehicle based on the number of axles and distance between them. We use Hough Transform, a parameter based feature detection method, to detect the axles. The quality of the detected circles is sensitive to appropriate settings of these parameters. Since the process is time consuming and it may not be fruitful to adjust these parameters manually every time, there is always a motivation to do a parameter search by attaching a machine learning algorithm to discover an optimized set.

We employ DE as the real parameter optimizer to find the best suited parameters for accurate circle detection, which is a crucial part of our vehicle classification system. We show that the use of DE, apart from removing the need for setting Hough Transform parameters manually, also has the added advantage of improving the accuracy of the axle detection module, thereby improving the robustness of the overall classification system. On the other side, the process of finding the optimal Hough Transform parameters does add an extra computational cost making it a obvious case of trade-off between speed and accuracy.

The focus of this work is to propose a new system of classifying vehicles, and investigate the utility of DE to improve its classification accuracy. Due to limited space, this work keeps the former part succinct and describes the later in detail. To the best of our knowledge, no axle based vehicle classification system in the current form has been proposed before.

4.2. Related Work

Vehicle classification is a difficult problem to tackle. Categorizing vehicles comprehensively is quite an arduous task given the variety of vehicles and similarities between them at the same time. Different shapes and sizes within a single vehicle category adds to the dilemma. On top of this we have drastically changing weather conditions, shadows, camera noise, occlusions, etc., which make the task even more challenging. Many attempts have been made to solve this classification problem using real time (online) and recorded (offline) video. In [72], the authors describe a vehicle tracking and classification system that could classify moving objects as humans or vehicles without classifying vehicles into further subcategories. A parameterized three dimensional model for vehicle classification was presented in [140]. The model was based on the shape of a common sedan, the assumption being that in regular traffic conditions, cars are more likely to be encountered than trucks or other vehicles. In [74] and [75], the authors developed three dimensional models for various vehicles like sedans, wagons, etc., and then compared the projections of these models with features of the detected object in the image. This model was parameterized and improved in [76]. In their award winning paper [114], the authors proposed a video based detection and classification system that modeled vehicles as rectangular patches with dynamic behavior. They used vehicle dimensions, i.e. length and height, to classify vehicles into two categories: cars and non-cars. Camera orientation played a big role in determining the height of the vehicle in this case. For example, the vehicle's height was computed as a combination of width and height as it was not possible to separate the two using only the vehicle boundaries and camera parameters.

Vehicle detection, which is an indispensable part of the classification system, has been generally approached through background subtraction models. In [78]-[80], the authors used background subtraction models for the vehicle detection task. An approximated background is subtracted from the current frame to extract the foreground object, and the background is updated over time. The important challenge for the background subtraction scheme, apart from being relatively computationally expensive, is the determination of the background, which may change with changing environmental conditions, and which may affect the heuristic thresholding that the scheme utilizes.

Lately, for vehicle counting and to circumvent the problems associated with background subtraction models to some extent, time-spatial image generation models have been proposed [81]-[83]. These models aim to detect a moving object that crosses a virtual line on the video frame. For the moving objects that pass this virtual line, a time-spatial image is generated and a count of the vehicles is approximated by the number of blobs detected in that image.

Feature based techniques [72], [114] are quite popular for classifying the detected objects. These methods make use of direct or indirect geometric and statistical features extracted from the pertinent frame which is usually constructed using background subtraction models discussed above. The larger the number of features used for classification, the smaller the misclassification error, but at the same time the higher the computational load. The classification performance of these models highly depend upon the chosen background model and its adaptation through the thresholding measure used. Moreover, the performance may start to degrade if the data statistics of the dynamically updated background inches closer to the detected objects.

This work proposes an axle based vehicle classification system. The main emphasis of this work is to investigate the feasibility of using axles to classify vehicles. Identifying axles in an image is essentially a circle detection problem. Circle detection holds high significance in image analysis as is evident from its vast applications in the manufacturing goods industry, military, etc. [71]. This problem has been tackled with different approaches most common of which are:

- *Deterministic* - Hough Transform based methods [23].
- *Geometric Hashing and template matching* [84], [85].
- *Stochastic* - Simulated annealing [86], Genetic Algorithms (GA) [87], etc.

The listed methods have shown important results with some limitations. For example, template matching has shown much promise [88], but it struggles to deal with pose invariance generated from complex models. Hough Transform based methods are the most common and popularly used [89], but are relatively computationally expensive. A number of methods have been proposed to overcome this shortcoming [90]-[92]. A GA based circle detector was presented in [93], which could detect multiple circles on real images, but failed to detect the ones with less than perfect configurations. The authors in [94] proposed an optimization method as an automatic circle detector, which was a combination of DE and simulated annealing. It could detect only one circle on synthetic images and also had the drawback of converging to sub-optimal solutions.

After weighing the pros and cons of all these methods we choose Hough Transform for our investigation. The main reasons for this choice, apart from its good success rate and popularity, was its relative ease of use, simple setup, and open availability of relevant APIs for testing.

The choice of Hough Transform as the circle detection method brings another challenge to the front. It is a parameterized method that works on thresholds. The quality and number of detected circles depend largely upon the parameter thresholds, which may vary given changing intensities, illumination of pixels and other relevant features of the image. Manual settings of these parameters could prove difficult as these settings will have to be adjusted for different scenarios of traffic. To solve this problem, we use DE as the parameter optimizer and attach it to the circle detection method. We describe the details in the next section.

4.3. The Proposed System

We present a video based vehicle classification system that categorizes vehicles based upon the number of axles and distance between them. As already mentioned, the focus of this work is to propose the idea of a different way of vehicle classification and test Differential Evolution's utility as a parameter optimizer in the process. The process essentially entails extracting relevant frames from a given video sequence, detecting axles as circles, computing distance between the farthest axles, and then classifying the detected vehicles. The proposed system, for now, works for a single traffic lane with the camera mounted on the sideways that captures the side view of the moving vehicle. A black-box description of the system is represented by Figure 4.1.

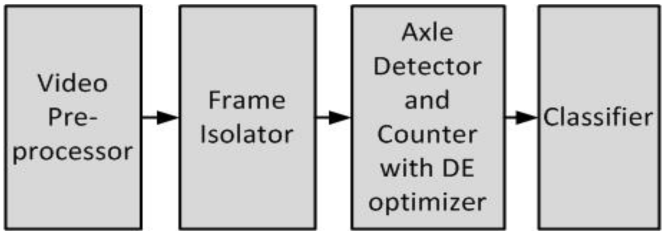


Figure 4.1. Modular overview of the axle count based vehicle classification system.

4.3.1. Video Pre-processor

The video pre-processor is an optional sub-system. The main utility of this module is to reduce the video size (frame size) from the recorded/captured resolution to the one set by the user. The higher the resolution of the video, the greater is the computational cost. A low resolution video, however, will be detrimental in achieving good detection accuracy. So the frame resolution should be kept within an acceptable range.

4.3.2. Frame Isolator

This module is responsible for locating the important frames within the captured video, i.e. the frames that contain the potential vehicles in them. This module assumes fairly high importance in the sense that the more potential frames this module misses to isolate from the video, the less the number it sends to the next module for axle detection thereby reducing the overall accuracy of the system. The important frames are extracted using the background subtraction model with the background being learned and updated dynamically. Background subtraction is a relatively popular technique for frame differencing. The idea is simple. The image data of the two frames isolated at different times is compared and the difference is converted into a useful metric. The resultant metric is then evaluated against a threshold. There are many popular methods for representing image data in terms of metrics. We use the histogram representation of image data. We compare the histograms of the current background and current frame, and then apply the Chi-Square metric [95], to compare the similarity of the frames. The Chi-Square metric is calculated as:

$$D(H_B, H_C) = \sum_{I} \frac{(H_B(I) - H_C(I))^2}{H_B(I)} \tag{4.1}$$

where I is the pixel intensity, H_B and H_C are the histograms for the background and current frames, respectively, and D is the resultant Chi-Square score. A low value of this score represents a better match and vice-versa. Another well-known metric that we experimented with in conjunction with Chi-Square metric is the Bhattacharyya distance [96]. Using a combination of these two metrics added robustness to this module, but at the same time slowed down the frame isolation process to some extent. Thus, for now, we employ only the Chi-Square metric to make a decision of either discarding or sending the current frame to the next module. The data statistics of the frames, and in some cases their difference or both, are fed to a model which returns a float value. If this value is above a certain threshold, a significant difference between the frames has been detected, and the current frame is sent to the axle detector and counter module, which is described in the next section. Details of this and the next module are kept succinct due to paucity of space.

4.3.3. Axle Detector and Counter with DE optimizer

This module is responsible for counting the number of vehicles in a given frame, their axles and distance between the axles. Hough Transform is used to detect the circles. Being a parameter based detection method, Hough Transform requires that the user provides some information about the circles that need to be detected. For example, the edge detector component requires a threshold to be set for the quality of edges detected. The higher this threshold, the fewer the number of circles that are detected. The important parameters for Hough Circle detection are:

- Accumulator threshold
- Edge detection threshold
- Inverse ratio of resolution
- Minimum distance between detected centers
- Minimum radius of detected circles
- Maximum radius of detected circles

4.3.4. DE optimizer

This section is the primary focus of the work presented in this chapter. All the parameters mentioned in the previous section are integers. These parameters can be tuned manually for a given scenario but the same set may show less than satisfactory performance on other test subjects. Thus, there is always a motivation to automate the process, and for that reason we employ DE to perform the parameter search. This, of course will require more compute cycles but would, at the same, improve the accuracy and robustness of the system as a whole. We test 5 DE variants to gauge their ability to perform this task effectively, and suggest the one which performs the best in terms of number of function evaluations used.

DE, being a real parameter optimizer, has to be modified to work with integer values. This essentially makes the task a combinatorial optimization problem. Truncating the real values to integer values seems a straight forward solution to this problem, but it has shown to be characteristically unstable in some cases [97]. Many novel approaches have been proposed to make DE perform the combinatorial optimization tasks and have yielded good results [98]-[99]. We utilize the approach suggested in [99] to convert integer values to float values and vise-versa, keeping all other properties of the DE variants unchanged.

After a potential frame is selected from the video, it is sent to the axle detection module. In real world applications, in general, apart from the distance between the camera and the road, other calibration parameters are usually known to the designer. This may help in determining a region of interest of the image where the vehicles are most likely to be detected. It would be computationally prudent to perform the detection and analysis on this region instead of the whole frame. As this work is primarily focused on testing the axle detection and counting approach (examining DE's effectiveness at the same time), we have steered clear of having to specify the calibration parameters of the camera and the captured scene. Instead, we have used video sequences where the distance between camera and the road is not fixed. This approach, though being relatively computationally expensive, tests the robustness of the system, and DE in particular by expanding its search space.

The fitness function for DE to optimize is kept simple. There is a cost associated with circles which are detected but are not aligned horizontally within a certain threshold. This addition of cost is based on the assumption that all the axles of the vehicle are likely to be horizontally aligned. The special case of raised axles is not considered here. Another cost is added if the radii of the detected circles differ more than a certain set threshold. This again is based on the assumption that all the axles of a vehicle are more likely to be of the same radius. There is a minimum distance between the centers that is specified and a cost is added if some circles are found to be closer than that distance. This is done to discourage DE from finding circles which are very close to each other. In mathematical form our model is represented as:

$$f(x) = (C_M)^2 \times (g(x) + h(x) + r(x)) \tag{4.2}$$

where

$$g(x) = (\frac{1}{C_A + \epsilon} + (C_T - C_A)) \tag{4.3}$$

$$h(x) = (\frac{1}{C_R + \epsilon} + (C_T - C_R)) \tag{4.4}$$

$$r(x) = (\frac{1}{C_D + \epsilon} + (C_T - C_F)) \tag{4.5}$$

and

C_M - maximum number of axles/circles to be detected in a frame; in our case we have fixed it to 10

C_T - total number of circles detected in a frame

C_A - number of horizontally aligned circles detected

C_R - number of detected circles having almost same radius

C_D - number of circles having their centroids satisfactorily distant from each other

ϵ - a very small number to avoid divide by zero error

There can certainly be many more sophisticated ways to improve this model but for our purposes we have kept it simple.

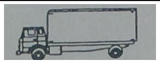
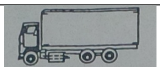
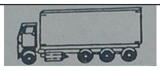
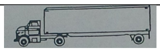
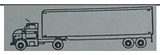
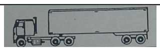
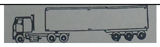
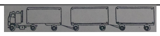
S. No.	Vehicle	Vehicle Class
1		Passenger Vehicle
2		Truck Type I
3		Truck Type II
4		Truck Type III
5		Truck Type IV
6		Truck Type V
7		Truck Type VI
8		Truck Type VII

Figure 4.2. Vehicle outlines and their associated classes.

4.3.5. Classifier

Classifying vehicles based on the number of axles and distance between them does away with the need to compute other attributes of the vehicle like height, width, area, solidity, etc. Computing these additional features may improve the classification accuracy but not without increasing the computational cost. Also, the length of a vehicle can be fairly approximated as the distance between the farthest axles. Our approach also does away with the need for employing a specialized classification algorithm, for now, as there are only two features involved. We use a simple Decision Tree classifier. In the future if the need arises, we might consider using a more sophisticated classifier. The current decision classes that we have experimented on, are shown in Figure 4.2. It should be noted that for this scheme to be fruitful, the distance between the camera and the road need to be fixed beforehand which should be considered a part of the camera calibration process. We, however, have experimented with varying distances as already mentioned and for the reasons stated in the previous section.

Table 4.1. A comparison of five variants of DE in detecting the number axles and their centers in 18 frames isolated from multiple video sequences. The values presented indicate the best/minimum value obtained by the variant along with a binary number (successful detection is represented as 1 and 0 otherwise).

Vehicle No.	No. of Axles	Manual Setting	DE/Best/1/	DE/Rand/1/	DE/RandToBest/1/	DE/Best/2/	DE/Rand/2/
1	2	1	52.00 (1)	52.00 (1)	52.00 (1)	52.00 (1)	52.00 (1)
2	2	1	52.00 (1)	52.00 (1)	36.33 (0)	52.00 (1)	36.33 (0)
3	2	1	52.00 (1)	52.00 (1)	52.00 (1)	52.00 (1)	52.00 (1)
4	2	1	52.00 (1)	52.00 (1)	52.00 (1)	52.00 (1)	52.00 (1)
5	2	1	52.00 (1)	52.00 (1)	52.00 (1)	52.00 (1)	52.00 (1)
6	2	1	52.00 (1)	52.00 (1)	52.00 (1)	52.00 (1)	52.00 (1)
7	2	1	52.00 (1)	52.00 (1)	52.00 (1)	52.00 (1)	52.00 (1)
8	4	1	36.33 (0)	29.00 (1)	29.00 (1)	29.00 (1)	29.00 (1)
9	5	1	29.00 (0)	25.00 (1)	25.00 (1)	25.00 (1)	25.00 (1)
10	5	0	25.00 (1)	29.00 (0)	29.00 (0)	29.00 (0)	36.33 (0)
11	2	0	52.00 (1)	52.00 (1)	52.00 (1)	52.00 (1)	52.00 (1)
12	2	1	52.00 (1)	52.00 (1)	52.00 (1)	154.00 (0)	154.00 (0)
13	2	0	52.00 (1)	52.00 (1)	154.00 (0)	154.00 (0)	203.00 (1)
14	2	0	52.00 (1)	52.00 (1)	52.00 (1)	52.00 (1)	52.00 (1)
15	2	0	154.00 (0)	203.00 (0)	52.00 (0)	52.00 (1)	52.00 (1)
16	2	0	136.33 (0)	52.00 (1)	52.00 (1)	154.00 (0)	152.00 (0)
17	2	0	52.00 (1)	52.00 (1)	52.00 (1)	52.00 (1)	52.00 (1)
18	2	0	152.00 (0)	203.00 (0)	154.00 (0)	52.00 (1)	203.00 (0)
–	Wins	10	13	15	13	14	13
–	Loses	8	5	3	5	4	5
–	Suc. Rate (%)	55	72	83	72	77	72

Table 4.2. Wins, Loses, and Success Rate of multiple NP-FEs combinations for DE/Rand/1/bin tested on 18 vehicular frames isolated from multiple video sequences.

Vehicle No.	10-10	10-20	10-30	10-40	20-20	20-30	20-40	20-50	30-30	30-40	30-50	30-60	40-40	40-50	40-60	40-70	50-50	50-60	50-70	50-80	60-60	60-70
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	1	1
9	0	0	0	0	0	1	1	1	0	0	1	1	0	0	0	0	0	0	1	1	1	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
11	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1	1	1	1	1
12	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
13	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
14	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
15	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	1	1	0	1
16	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0
17	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Wins	11	12	12	12	12	13	13	13	12	12	13	13	12	12	14	15	13	13	16	16	15	14
Loses	7	6	6	6	6	5	5	5	6	6	5	5	6	6	6	3	5	5	2	2	3	4
Suc. Rate (%)	61	66	66	66	66	72	72	72	66	66	72	72	66	66	77	83	72	72	88	88	72	77

4.4. Experimentation and Results

The performance of the system with and without the DE optimizer is presented. The videos captured were of single lane highways and streets. The traffic flow was chosen to be moderate. Table 4.1 presents the performance of five variants of DE on 18 test frames isolated from multiple video sequences, which are the outputs of the frame isolator module. The respective column value includes the best value achieved by the DE variant alongside a binary number which is shown as 1 if the DE variant was able to find the equivalent number of axles with the same centroids in the frame, i.e., excluding the false positives. The binary number is substituted as 0 otherwise. The results of a superior manually tuned parameter setting is also presented. We fixed the crossover rate (Cr) to 0.9 and scaling factor (F) to 0.5 as suggested in [35]. The maximum

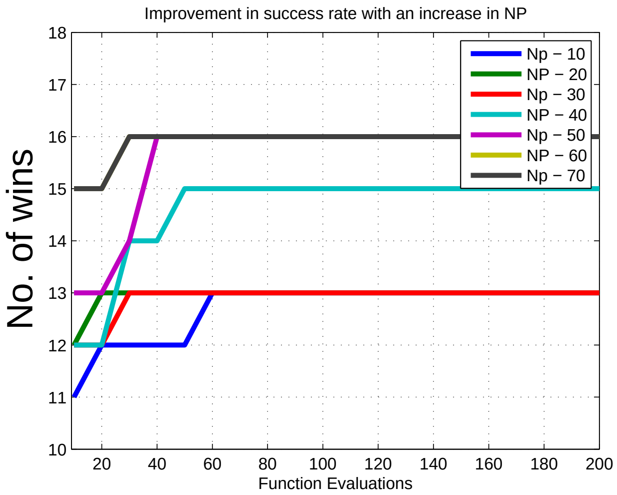


Figure 4.3. Performance of DE/Rand1/bin using multiple population sizes with increasing function evaluations.

function evaluations was set to 300.

It is clear that DE/Rand/1/bin emerges as the best strategy among the DE variants. To improve upon the accuracy and speed of detection, we further experimented with multiple population sizes to see if that actually impacts the system's performance. The motivation is to investigate if a lower value of the population size *NP*, and for that matter fewer function evaluations, produces the same results as shown in Table 4.1, or would a higher *NP* produce better results. *NP* cannot be too high so as to exacerbate the performance making the system untenable. At the same time, it cannot be too low as this might seriously degrade the accuracy. In essence, this problem presents the classical accuracy versus speed dilemma and we try to find the critical and harmonious set of parameters that lead to acceptable performance on this particular problem. The results are enumerated in Table 5.2.

We tested multiple *NP-FEs* combinations leading to a generally expected result, i.e., performance improves with an increase in *NP* and subsequent increase in *FEs*. Table 5.2 also points out an interesting observation, i.e., an increase in the number of solutions after a certain number does not necessarily lead to an improvement in performance of DE. This phenomenon has also been corroborated by some recent publications [35], [100], though, their domain was real parameter optimization on benchmark functions. This result insinuates that there is a critical value of *NP* after which an increase in the number of solutions

Table 4.3. Effect of increasing NP and function evaluations on success rate. Saturation point is reported at 50-70 combination.

Population Size (NP)	Success Rate %	Saturation FEs
10	66	60
20	72	40
30	72	50
40	83	70
50	88	70
60	88	100
70	88	90

does not necessarily lead to an improved performance.

Figure 4.3 summarizes the results presented in Table 5.2. We performed the parameter search with population size ranging between 10 and 70 with an increment of 10. Figure 4.3 shows that the success rate of a population size improves with an increase in function evaluations. This is an expected outcome. But after a point, increasing the population size does not improve the success rate. On similar lines, an increase in function evaluations does not offer an added advantage after a certain limit as the success rate saturates. We found that the best set of control parameters that lead to the highest accuracy (88%) among the combinations compared is: F=0.5, Cr=0.9, *NP*=50 with 70 *FEs*. Increasing *NP* above this value does not yield better results. Figure 4.4 visually depicts the results obtained for manual settings (left aligned in the sub-figures) as compared to DE/Rand/1/bin optimized set (right aligned in the sub-figures) discovered. Due to space constraints, only 16 of total frames are presented. Axles detected by both methods are represented by red circles.

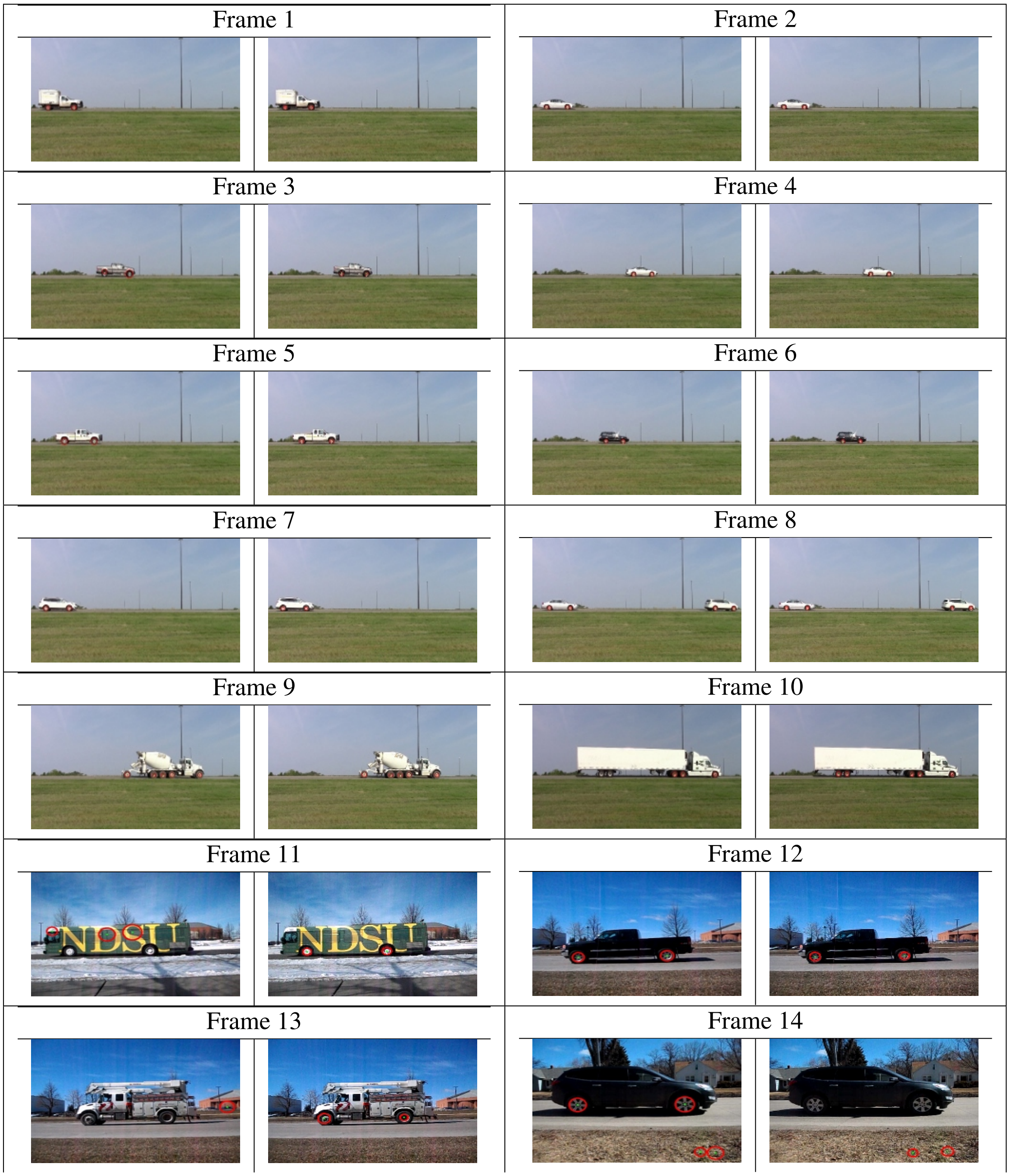
It is imperative to note that manually setting circle detection parameters can be tedious and depends upon the scenario at hand. At the same time it is quick. Our results show that manually setting the parameters leads to a low success rate (55%) if the weather conditions and other factor are changed. At the same time, attaching an DE optimizer to the circle detection system can slow down the detection process but yields a much higher success rate (88%). The system of classification that we propose, therefore, may be suited more for off-line detection and classification of vehicles where the video is pre-processed to some extent to reduce its size etc.

4.5. Summary

This work presents an axle count based vehicle classifier. Our system consists of four modules namely video preprocessor, frame isolator, axle detector, and classifier. The output of one module feeds the other in the same sequence. We used the background subtraction technique in our frame isolator module to extract pertinent frames from a video sequence. Axle detection is performed with Hough Transform, which is a well-known feature detection method in the image analysis domain. Hough Transform for circle detection works on parameters that are dependent on the image data and type of problem that is being addressed. Manually setting these parameters can be tricky, tedious, and often produces less than satisfactory results (as shown in this chapter) if the weather conditions and related circumstances change. We therefore use a combinatorial version of Differential Evolution to optimize the parameter set.

This approach yields much higher accuracy as shown by the results we achieved. We initially tested five different variants of DE, and concluded that DE/Rand/1/bin is most suitable for this task reaching a steady success rate of 83% while excluding the false positives. We further investigated the plausibility of DE/Rand/1/bin to further its accuracy and speed. For this we tested this variant with multiple population sizes (*NP*) - *FEs* combinations. We found that F=0.5, Cr=0.9, and *NP*=50 with 70 *FEs* yields an accuracy of around 88%, and increasing *NP* further does not yield any better results. Our current system is designed to be used as an offline vehicle classifier.

To make the system perform as an online classifier, a few changes need to be made. For example, by careful camera calibration, it is possible to specify a region of interest in the test frame where the probability of finding the axles is quite high given various assumptions about inclination of the road. This will reduce the computing load considerably. If there is enough information available about the scene, it is possible to initialize DE with good values to begin with. These and other modifications are planned as future work. In addition, future work includes developing the system further by employing a parallel version of DE to increase its overall speed to make it work online, and extending it to classify multiple lane traffic.



Frame 1

Frame 2

Frame 3

Frame 4

Frame 5

Frame 6

Frame 7

Frame 8

Frame 9

Frame 10

Frame 11

Frame 12

Frame 13

Frame 14

Figure 4.4. Results obtained through manual settings (left aligned) of Hough Transform parameters vs the best settings obtained for DE/Rand/1/bin (right aligned).

5. A DIFFERENTIAL EVOLUTION BASED MULTICLASS VEHICLE CLASSIFIER FOR URBAN ENVIRONMENTS

Video analytics is emerging as a high potential area supplementing intelligent transportation systems (ITSs) with wide ranging applications from traffic flow analysis to surveillance. Object detection and classification, as a sub part of a video analytical system, could potentially help transportation agencies to analyze and respond to traffic incidents in real time, plan for possible future cascading events, or use the classification data to design better roads. This work presents a specialized vehicle classification system for urban environments. The system is targeted at the analysis of vehicles, especially trucks, in urban two lane traffic, to empower local transportation agencies to decide on the road width and thickness. A hybrid appearance model specifically designed for speedy foreground extraction in the given context is presented. A simple motion cue based tracking algorithm is used, and this work stays clear of using probabilistic trackers. The main thrust is on the accurate classification of the detected objects using an evolutionary algorithm. The classifier is backed by a differential evolution (DE) based discrete parameter optimizer. It is shown that, though employing DE proves expensive in terms of computational cycles, it measurably improves the accuracy of the classification system. The system was tested on multiple real video footage during varied weather conditions from a camera mounted in urban areas, achieving a peak classification accuracy of approximately 90%.

The rest of this chapter is structured as follows. Section 5.1 presents an introduction to the chapter. Section 5.2 describes the related work. In Section 5.3 describes and explains the proposed system with all its features. In Section 5.4, results and their analysis are presented, and Section 5.5 concludes the chapter.

5.1. Introduction

Vehicle detection and classification is currently a hot focus area in the myriad web of intelligent transportation systems (ITSs) with immense potential for traffic flow control, security, and surveillance to name a few. ITSs are embracing data driven techniques [101] wherein the data related to traffic density, incidents, etc. are relayed back to users of the traffic systems thereby empowering them to make real-time decisions about routes thereby promoting efficiency. Road side regulations, increasing density of vehicles on roads, and costs of overlaying the roads are some of the supplementary and rather critical reasons calling for ever more efficient utilization of our transportation networks. Robust vehicle classification systems that are able to compute the number and type of vehicles plying on a particular road or highway, provide a part of the solution.

Various vehicle detection and classification systems such as digital wave radars [102], amplitude modulated laser radars [103], lidars [104], magnetometers [105] etc., have been proposed and some have been in continual use commercially with their inherent advantages and disadvantages. These systems can be broadly classified depending upon the type of sensors they use or on the basis of their installation vis a vis intrusive and non-intrusive. Most commonly used systems, use one or a combination of laser, piezoelectric, microwave, or video cameras. For example, inductive loop detectors are one of the most accurate and widely used systems for vehicle detection. But due to their intrusive nature, high equipment and installation costs, they are not often being applied [106].

Video based vehicle classification have recently emerged as a low cost alternative to conventional intrusive systems primarily owing to their low cost, non-intrusive nature of installation, and operation. For example, there have been extensive use of video based vehicle detection systems in surveillance [107], [108], [109]. Other benefits include [110]:

- Freedom from extensive sawing residue and extensive cleaning after installation and continual maintenance.
- Installation can be done year round.
- No need for road closure for installation and maintenance thereby reducing the impact on traffic flow.
- Ease of operation.
- Provides rich data through which additional information and context can be approximated.
- Vison based systems integrate well with other object detection systems.

Their advantages notwithstanding, video based detection and classification systems have their own challenges, and pose many difficulties for researchers. Some of them may be categorized as [107]:

- Occlusion: Vehicles and other objects on roads can block each other in the camera view leading to a false count.

- Rapid illumination changes: Weather changes may range from periods of very high brightness to very low intensities posing a problem for detection algorithms.

- Vehicle edge/contour deformation: Due to various continual changes in scene, the vehicle may not be detected with a perfect boundary leading to mis-classifications.

- Multiplicity of vehicle types: There are highly varied types of vehicles around ranging from a mid-sized sedan to transport trucks with different lengths, heights, and axles in between. Extracting individual features and correctly classifying vehicles becomes a challenge in this scenario. Add to that, pedestrian, bicycles and other objects, the problem becomes even more difficult.

There are other barrages of problems that specific stages of the classification systems try to solve, and will be discussed later in this work.

A novel DE based vehicle detector and classifier (DEVEC) capable of classifying vehicles on highway and urban areas is presented. DEVEC has a conventional vehicle detection architecture with the difference that an evolutionary algorithm (DE) is used for classification of vehicles using multiple cues including the axle count. Hough Transform, a parameter based feature detection method is employed to detect the vehicle axles. The quality of the detected circles is sensitive to appropriate settings of these parameters. Since the process is time consuming, it is not viable to adjust these parameters manually every time, thus, there is always a motivation to do a parameter search by attaching a machine learning algorithm to discover an optimized set.

DE is modified and used as a discrete parameter optimizer to find the best suited parameters for accurate axle detection, which is a crucial part of this vehicle classification system. The use of DE, apart from removing the need for setting Hough Transform parameters manually, also has the added advantage of improving the accuracy of the axle detection module, thereby improving the robustness of the overall classification system. On the other side, the process of finding the optimal Hough Transform parameters does add an extra computational cost making leading to the trade-off between speed and accuracy.

The motivation of this work is three-fold:

- As a proof of concept, this works classifies vehicles primarily based on the number of axles and distance between by mounting the camera sideways instead of a lamppost where the camera is conventionally mounted on the top.

- Assessing the utility of including axle count as a feature for classification aids in discrimination of trucks having similar geometric features but have distinct axle count.

- To maximize the model function representing the scene under consideration, an evolutionary algorithm namely Differential Evolution (DE) is attached .

To the best of my knowledge, no other vehicle detection and classification algorithm makes use of an evolutionary algorithm detecting axles for vehicle classification.

5.2. Related Work

A typical video based vehicle detection system is shown in Fig 5.1. First a review of different techniques proposed to tackle problems associated with each stage is provided.

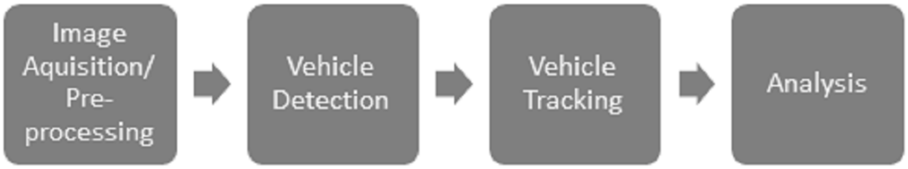


Figure 5.1. A typical vehicle detection system

5.2.1. Vehicle Detection

Detection is the primary step towards analysis of videos in an intelligent transportation system. The robustness of this step is quite critical as it feeds the higher sub systems like vehicle tracking and classification. The vehicle detection problem, in particular, has been approached through many different ways, and multiple methods have been proposed to achieve detection. These methods can be broadly classified into two classes: motion based and appearance based [107].

Motion based detection methods aim to extract the vehicle information based on a comparison of present the pixel state with an assumed stationary (background) state of the system. The easiest way to perform motion based detection is frame differencing [111], [112], [113] wherein the pixel wise thresholded difference is calculated between two consecutive frames furnishing the output in terms of motion pixels, or what is generally referred to as the foreground. Although simple, this technique needs supplementation with other methods for dynamic motion and use of more information apart from just the difference of pixel is desirable.

Background subtraction is another common motion based segmentation technique and one of the more widely used [114]. This method tries to build a background model of the scene based on the accumulated information. This background is then compared with the current video frame giving the motion information. In its simplest form, called as the background averaging method [115], a background is built by averaging the pixel attributes of some n previous frames. This may not be suitable for highly dynamic traffic scenes where the background changes too rapidly. There are other methods of background construction that do not assume a fixed background beforehand. For example, background was constructed assuming a single Gaussian distribution in [116], [117]. Every pixel is either classified as background or foreground based on its distribution. This model may be perceived as an equivalent to computation of a dynamic threshold [117]. The advantages of this method are improved robustness and reduced memory requirement.

Gaussian mixture models (GMM) were introduced by [118], and since then have been used quite liberally to model the background scene [119], [120], [121], [122], [123], [124]. Each pixel is modeled as a mixture of two or more Gaussians, with each distribution being estimated as either background or foreground. The advantage of modeling the scene as GMM lies in the fact that it can handle multi-model background distributions. A sudden change in illumination affects the performance considerably [125]. Other motion segmentation based techniques include median filter [126], [127], kernel density estimation [128], kalman filtering [129], [130] and optical flow [131], [132].

Appearance based detectors, in contrast with motion based detectors, use appearance features like color, texture, shape, etc., to extract the object of interest, in this case vehicles from the image or video directly. Coded descriptors based on features are utilized to model the appearance of vehicles. Local edge operators have been used in [133], [134].

Recently, more sophisticated and robust feature descriptors have been proposed in the form of Scale Invariant Feature Transform (SIFT) [135], Histogram of oriented gradients (HOG) [136], Speeded up Robust Features (SURF) [137], and Haar-like features [138], to name a few. A detailed explanation of these and more such techniques can be found in the recent survey paper [139].

5.2.2. Vehicle Tracking

Vehicle tracking is essentially a state prediction and data association problem. The idea is to recognize the vehicle in subsequent frames, locate its position, and ultimately obtain its trajectory. Vehicle tracking is sometimes merged with the detection task but may also be performed separately. There are many methods available in this literature that cater to this task and these methods can be broadly classified into three categories: model based, region based, and feature based tracking.

Model based methods presume a predetermined 2-D or 3-D vehicle appearance model matching the presumed model with regions of motion in the sequence [131]. A multi view 3-D model that builds a 3-D model based on 2-D geometrical information, was constructed using edge features by [140]. Other 3-D modeling techniques were proposed in [141] and [142].

Region based tracing on the other hand, aims to detect a vehicle's silhouette contained within a geometric shape represented by multiple features lie area, length, width, centroid, etc. The vehicle may be represented in terms of a feature vector with continuous updation. This vector is tracked through shape matching or data association in subsequent frames. To track highway vehicles, a graph based matching approach was used in [115]. The information such as length and height of the convex hull was used in [142] to track the vehicles while in [117], centroid and velocity information was used.

Feature based tracking is based on the combined use of simple features like edges with feature descriptors like SIFT, SURF, HOG, etc. For example, region based tracking was combined with SIFT in [143]. Main advantages of feature based tracking is its ability to perform well in crowded areas but the challenge lies in choosing effective features.

5.2.3. Vehicle Classification

Classification requires vehicles to be associated with a particular class. This has been achieved in the literature either by using shape features (height, aspect ratio, etc.) or appearance of the vehicle.

The number of features used has a direct impact on the number of discriminant classes, and nature of the classifier needed.

Authors in [144] classify vehicles into car, SUV, and minibus by making use of the curve information associated with 3-D ridges. Their classifier achieved an accuracy of 88%. A similar 3-D model based classifier was constructed in [145] capable of discrimination between car, bus, van, and motorcycles with a classification accuracy of approximately 96%. Based on the contour information a classifier based on a combination a voting algorithm and Euclidian distance was proposed in [146] achieving a classification rate of 93%.

Appearance based classifiers use information based on gradients, corners, etc. In [147], a 2-D LDA technique was employed capable of classifying 25 vehicle types with an accuracy of approximately 91%.

5.3. Proposed Approach: Differential Evolution Based Vehicle Classifier (DEVEC)

This work presents a Differential Evolution based vehicle classifier (DEVEC) primarily designed for urban two lane traffic but may be extended for multi lanes with a few modifications. A black box view of DEVEC is presented in Fig 5.2. A detailed description of the individual stages is presented below.

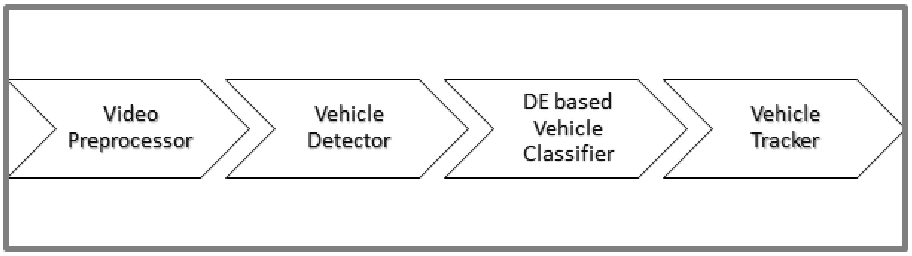


Figure 5.2. A typical vehicle detection system

5.3.1. Video Preprocessor

This is an optional sub-system. The main utility of this module is to reduce the video size (frame size) from the recorded/captured resolution to the one set by the user. The higher the resolution of the video, the greater is the computational cost. A low resolution video, however, will be detrimental in achieving good detection accuracy. Thus, the frame resolution should be kept within an acceptable range. Another optional sub-feature of this module to automatically identify the road in consideration thereby extracting the region of interest (ROI) containing the potential vehicles. The subsequent steps operate on ROI extracted from the video frames.

5.3.2. Vehicle Detector

This component constitutes the backbone of the whole system. The reason for this claim is because correct vehicle classification is as good as vehicle detection. Vehicle detection, in this system, primarily consists of two basic steps: background construction/modeling and detection.

In motion based segmentation and detection models, as in ours, a robust background plays a vital role in appropriate detection, as the motion pixels are extracted from the scene by subtracting it with the modeled background. There are many different background models for vehicle detection available in literature based on different techniques. [148], [149], [150] used single Gaussian to model the background while recently authors in [151] used a mixture of Gaussians. A consolidated review of such research was presented in a recent survey paper [139].

A pixel-to-pixel based adaptive background construction model is proposed. Without assuming any distribution information about the background, this model compares the color RGB information of every pixel in the background with the current scene, and calculates a three dimensional Chi-Square metric. This metric is then compared with a threshold, which is determined adaptively and is continuously updated. This process is described in Algorithm 1.

To calculate the initial threshold T, n training frames are required. For every two subsequent frames I and I-1 in the training set, the Chi-Square distances of R, G, and B components are calculated as:

$$R_T = \Sigma_N \frac{(R_I - R_{I-1})^2}{R_I} \tag{5.1}$$

$$G_T = \Sigma_N \frac{(G_I - G_{I-1})^2}{G_I} \tag{5.2}$$

$$B_T = \Sigma_N \frac{(B_I - B_{I-1})^2}{B_I} \tag{5.3}$$

Algorithm 3 DEVEC ADAPTIVE BACKGROUND CONSTRUCTION

```
Choose initial n training frames
Initialize distance vector V, with size n
Initialize threshold T as 0
for every frame i till n-1 frames do
    Compute mean M, and standard deviation SD, for RGB components of frame i
    Update vector V as V[i] = M + SD
end for
Compute T as:
     T = mean(V) + standarddev(V)
Update frame n as current background Bcurrent
Update frame n as previous background Bprev
while video has frames do
    for very pixel p in frame I and background Bcurrent do
        Compute Chi-Square distance CS
        if C thenS $\geq$ T
            Classify p as foreground
        else
            Classify p as background
        end if
    end for
    Update foreground pixel locations in Bcurrent from Bprev
    if no vehicle detected for 3 consecutive frames then
        Recalculate T and update current frame as Bcurrent
    end if
end while
```

Since this investigation presumes no information about the background color, all the differences are equally weighted to calculate the gradient of pixel differences G as:

$$G = w_R \times R_T + w_G \times G_T + w_B \times B_T \tag{5.4}$$

with $w_R = w_G = w_B = 0.33$. Based on the assumptions of scene color, different weights may be assigned to different color components but this work presumes no such information.

After the initial gradient threshold G is determined, a new background is adaptively constructed. Figure 5.3 presents the adaptive determination of G for the first 72 frames of a particular video sequence. A gradient threshold G is then determined for every subsequent frame. Every current frame I is compared with the previous frame I-1 pixel by pixel, and differences between RGB components for every pixel p are calculated as:

$$RGB_{diff}(p) = w_R \times (R_I - R_{I-1}) + w_G \times (G_I - G_{I-1}) + w_B \times (B_I - B_{I-1}) \tag{5.5}$$

If $RGB_{diff}(p)$ for a pixel p exceeds the pixel gradient G, p is classified as a foreground pixel else it is considered as a background pixel. In the current form every pixel difference is compared with G, and a decision is made about the pixel's class. Another version of this model can be constructed in which every pixel's own gradient can be calculated and stored but this would require extra computation and storage. For the purposes of this investigation the current form will suffice.

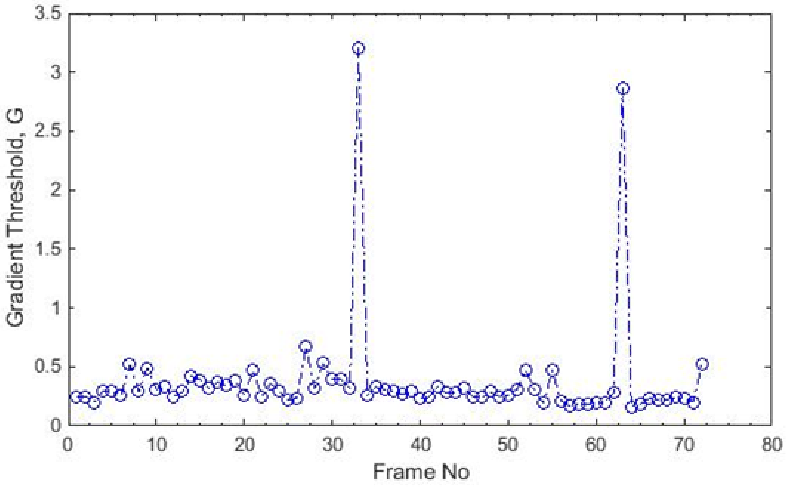


Figure 5.3. Change of gradient threshold during adaptive threshold calculation and updation

A previous background is maintained in the memory, which provides the pixels to current background under construction at locations where foreground pixels are detected. In this way the current background is updated continuously, and describes the current scene robustly. Figure 5.3 depicts this process of adaptive background construction while classifying the vehicles as foreground and background.

The pixel gradient threshold G is updated after x_{update} number of frames, which is a user defined number. As a word of caution, x_{update} should be under acceptable limits, as a low value of x_{update} would mean very high frequency of threshold calculation and updation that may degrade the performance of the system.

After the background construction, the next step is to detect vehicles from the scene using motion segmentation. Background subtraction is used to identify motion pixels as contours. With motion pixels, undesired noise is most certainly always detected. A simple noise suppressor algorithm is used to remove the unwanted noise, which increases the cost marginally in terms of computation cycles but has a desirable impact on system's fidelity. After that a bounding box is fit around the detected contours.

In essence the bounding box creation is a hypothesis generation step. A dual hypothesis verification process with soft and hard matches is employed. As an initial soft match, this bounding box is matched with a basic car/truck template. If this criteria is satisfied, this patch is sent to a DE enabled axle detector for a hard match. A soft match is defined as a positive if only the basic feature matching threshold requirements are met which in this case are length, breadth, and contour area. A hard match involves determining the number of axles in the vehicle, and minimizing a vehicle model fitness function using DE. This two-step process improves robustness of the detection module though at the expense of extra compute cycles. Algorithm 2 depicts this detection process. During the detection and tracking process, due to dynamic change in scene and vehicle characteristics, there is always a possibility that the vehicle being tracked appear shape-wise deformed in subsequent frames or it may split up into different parts. For example, a single vehicle may split up into two unrelated and distinct parts, or a sub area, capable of being identified as a new vehicle, or is formed inside the tracked vehicle itself etc. This algorithm is capable of resolving such anomalies, and boost accurate detection.

Algorithm 4 VEHICLE DETECTION PROCESS IN DEVEC

```
while video has frames do
    Perform background subtraction
    Extract foreground pixels
    Clean the background with a noise suppressor
    Find contours in the cleaned background
    Fit bounding boxes around contours
    Soft match the boxes with a basic vehicle template
    if soft match successful then
        Send the object for a DE enabled axle detector
    end if
end while
```

5.3.3. Classification

After the soft detection of the vehicle that only checks for a minimum contour area, length and breadth of the bounding box, the relevant region is extracted from the current color frame, and sent to the DE based vehicle classifier module. The main emphasis of using this novel module is to investigate the feasibility of using axles to classify vehicles. Identifying axles in an image is essentially a circle detection problem. Circle detection holds high significance in image analysis as is evident from its vast applications in the manufacturing goods industry, military, etc. [71]. The approach used here is similar to the used in Chapter 4.

After weighing the pros and cons of the methods used for circle detection Hough Transform is chosen for the investigation. The main reason for this choice, apart from its good success rate and popularity, was its relative ease of use, simple setup, and open availability of relevant APIs for testing. The choice of Hough Transform as the circle detection method brings another challenge to the front. It is a parameterized method that works on thresholds. The quality and number of detected circles depend largely upon the parameter thresholds, which may vary given changing intensities, illumination of pixels, and other relevant features of the image. Manual settings of these parameters could prove difficult as these settings will have to be adjusted for different scenarios of traffic. To solve this problem, this work uses DE as the parameter optimizer, and attach it to the circle detection method.

5.3.4. DE Optimizer

DE, being a real parameter optimizer, has to be modified to work with integer values. The approach suggested in [97] to convert integer values to float values and vise-versa is utilized, keeping all other properties of the DE variants unchanged.

In real world applications, in general, apart from the distance between the camera and the road, other calibration parameters are usually known to the designer. This may help in determining a region of interest of the image where the vehicles are most likely to be detected. It would be computationally prudent to perform the detection and analysis on this region instead of the whole frame. As this work is primarily focused on testing the axle detection, and counting approach (examining DE's effectiveness at the same time), this work has steered clear of having to specify the calibration parameters of the camera and the captured scene. Instead, we have used video sequences where the distance between camera and the road is not fixed. This approach, though being relatively computationally expensive, tests the robustness of the system, and DE in particular by expanding its search space.

The fitness function for DE to optimize is kept simple. There is a cost associated with circles, which are detected but are not aligned horizontally within a certain threshold. This addition of cost is based on the assumption that all the axles of the vehicle are likely to be horizontally aligned. The special case of raised axles is not considered here. Another cost is added if the radii of the detected circles differ more than a certain set threshold. This again is based on the assumption that all the axles of a vehicle are more likely to be of the same radius. There is a minimum distance between the centers that is specified and a cost is added if some circles are found to be closer than that distance. This is done to discourage DE from finding circles that are very close to each other. Chapter 4 can be referred for the mathematical formulation of the model.

There certainly can be many more sophisticated ways to improve this model but for work's purpose it is kept it simple. Another reason for keeping the fitness function quick to compute is to make the classification process time efficient.

Classifying vehicles based on the number of axles, and distance between them does away with the need to compute other attributes of the vehicle like area, solidity, depth, etc. Computing these additional features may improve the classification accuracy but not without increasing the computational cost. Also, the length of a vehicle can be fairly approximated as the distance between the farthest axles.

The current approach also does away with the need for employing a specialized classification algorithm as there are only few features involved. A simple Decision Tree classifier is used. The current decision classes that are the target of this experiment are shown in Figure 5.7. As is clear from the figure, employing axle count information is crucial, in fact necessary to correctly distinguish between Truck Type I and II, Truck Type III and IV, Truck Type VI and VII, as they have similar geometric features but different axle counts.

5.3.5. Tracking

After the vehicles are detected, tracking is performed on them to ascertain their positions in the next frame. In other words, the trajectory of the vehicle during the course of its existence in the video is determined through tracking.

This is an important operation if the correct count of the vehicles is to be obtained. If the vehicles are not tracked properly, the re-detection of the same vehicle in the next frame may be surmised as a new vehicle, which in turn would result in over-counting.

Detection and tracking are performed separately where first vehicles are detected in every frame, and then this detected vehicle data is associated with data from the previous frame, and then the vehicle's location and features are updated iteratively. This technique takes advantage of the fact that any detected vehicle is not likely to be present in only one frame. This method is an example of region based tracking mechanism [152], [153] where the vehicle contour is detected and fitted inside a rectangular box. This box and contour, in the current system, is characterized by edges, contour area, and box coordinates. Data association and template matching is then performed to track the vehicles in consecutive frames to track them. A local memory containing the features, and other relevant data is maintained for every vehicle that is detected. The features are updated during tracking as the vehicle moves from frame to frame. This way the vehicle appearance in the memory stays in sync with the current state of the vehicle. This continuous updation is vital since the vehicle appearance is prone to edge, and area deformation while in motion.

5.4. Experimentation and Results

In this section, the performance of DEVEC is presented when tested on multiple real time , primarily urban two lane traffic scenarios. Apart from evaluating the system for its speed, correct vehicle count, and classification accuracy, the major thrust of the evaluation is on the performance of Differential Evolution (DE) as an optimizer applied to this scenario. Moreover, given the current setting and to gauge the performance of DE, the system was tested with and without the DE optimizer.

As discussed in the previous section, the main utilization of DE is to find and locate the number of correct axles of the vehicle by minimizing a fitness function. The information on the number of axles, distance between the farthest axles supplemented with height, width, and contour area of the vehicle was used to classify the vehicle in one of the listed classes. The system was initially tested with manual settings of hough transform parameters and then with five popular variants of DE, on vehicles isolated from multiple video sequences.

The videos were recorded on a two lane road, and the traffic flow was chosen to be moderate. Table 5.1 presents the first set of results comparing the system's classification performance with and without the

DE optimizer. It lists the performance on 20 different vehicles isolated from a single video sequence with the number of axles in each vehicle. A single fine-tuned manual setting of hough parameters was used all along the tested videos. The number 1 denotes the correct identification of number of axles by the system using the manual settings, and a 0 means that the system failed to identify the correct number of axles with the given parameters leading to a misclassification.

It is important to mention that after a vehicle is detected, the rectangular region around it is extracted, and may be examined either by a pre-determined circle detector or the DE optimizer module. This extracted vehicle region from the frame may measure as less as 40×30 pixels. This presents a peculiar problem when it comes to detecting circles in such a confined image space. To maintain the practicality of the context and achieve good circle detection, hough parameter thresholds are highly relaxed. This has the effect of detecting many more circles and in cases a lot of false positives are generated. It is this challenge that DE is supposed to overcome with correct circle detection, and remove unwanted circles by minimizing the fitness function.

As for DE, it was allowed to search the space for optimal hough parameters thereby minimizing the energy/fitness function. For all the five variants, the fitness function value obtained is listed in the corresponding columns. If the DE variant was able to identify the correct number of axles, it is signified with a 1 alongside the fitness value, and 0 otherwise. As for the control parameter settings of DE, this work fixes the crossover rate (Cr) to 0.9 and scaling factor (F) to 0.5, and population size (NP) to 50 as suggested in [35]. The maximum number of function evaluations was set to 200.

The results show that DE/Rand/1/Bin strategy emerges as the best DE strategy among the ones tested, reaching an accuracy of approximately 90% in all video sequences tested. All the five different variants tested in this paper have their own distinct characteristics with which they affect the generation of new solutions and impact the overall search process. For example, DE/Best/1/Bin tries to search around the best solution achieved so far thereby moving towards the solution very quickly and in many cases converging prematurely. This strategy losses diversity of the population quite fast as compared to other variants. DE/Rand/2/Bin, on the other hand is known to create diverse solutions due to the presence of two differential vectors. This diversity, many a times, leads to very slow convergence. All in all, no single strategy is perfect for all problems, and their success is partially dependent on the nature of problem they intend to solve. In this case, DE/Rand/1/Bin proves to be a better strategy, primarily due to its balanced and simple approach towards the search process. It is also relatively slow but manages to maintain good diversity of solutions throughout the search process.

Apart from the mutation strategy and as is true with any evolutionary algorithm, the control parameter settings play an important part in the performance of DE. Conducting a large scale control parameter analysis for this scenario is out of the scope of this work. This work nevertheless experimented with multiple population sizes to see if that actually impacts the system's performance. The motivation is to investigate if a lower value of the population size NP, and for that matter fewer function evaluations, produces the same results as shown in Table 4.1, or would a higher NP produce better results. NP cannot be too high so as to exacerbate the performance making the system untenable. At the same time, it cannot be too low as this might seriously degrade the accuracy. In essence, this problem presents the classical accuracy versus speed dilemma and this work tries to find the critical and harmonious set of parameters that lead to acceptable performance on this particular problem. The results are enumerated in Table 5.2.

An optimal parameter search was performed with population size ranging between 10 and 80 with increments of 10. It is shown that the success rate of a population size improves with an increase in function evaluations. But after a certain point, increasing the population size does not improve the success rate. On similar lines, an increase in function evaluations does not offer an added advantage after a certain limit as the success rate saturates. It is observed that the best set of control parameters that lead to the highest accuracy (90%), among the combinations compared, is: F=0.5, Cr=0.9, NP=40 with 60 FEs. Increasing NP above this value does not yield better results. If this critical point can be deduced theoretically, it may be used as an effective indicator of the extra computational budget that the DE optimizer module may consume in the system.

Figure 5.8, visually depicts a part of the results obtained for manual settings (left aligned in the sub-figures) as compared to DE/Rand/1/bin optimized set (right aligned in the sub-figures) discovered, for 9 different vehicles sourced from multiple video sequences. Axles detected by both methods are represented by solid white circles.

Table 5.3 and 5.4 summarize the cumulative results of the DEVEC as a confusion matrix. The system was tested on three videos of approximate duration of 10 minutes each. This by no means is an

Table 5.1. A comparison of five variants of DE in detecting the number axles and their centers in 20 frames isolated from multiple video sequences. The values presented indicate the best/minimum value obtained by the variant along with a binary number (successful detection is represented as 1 and 0 otherwise).

Vehicle No.	No. of Axles	Manual Setting	DE/Best/1/bin	DE/Rand/1/bin	DE/RandToBest/1/bin	DE/Best/2/bin	DE/Rand/2/bin
1	2	0	52.00(1)	52.00(1)	52.00(1)	52.00(1)	52.00(1)
2	2	1	52.00(1)	52.00(1)	36.33(0)	52.00(1)	52.00(1)
3	2	1	52.00(1)	52.00(1)	52.00(1)	52.00(1)	36.33(0)
4	2	1	52.00(1)	52.00(1)	52.00(1)	52.00(1)	52.00(1)
5	5	0	36.33(0)	25.00(1)	25.00(1)	36.33(0)	29.00(0)
6	2	1	52.00(1)	52.00(1)	52.00(1)	52.00(1)	52.00(1)
7	2	0	52.00(1)	52.00(1)	52.00(1)	52.00(1)	52.00(1)
8	4	0	36.33(0)	36.33(0)	29.00(1)	29.00(1)	29.00(1)
9	2	1	52.00(1)	52.00(1)	36.33(0)	52.00(1)	36.33(0)
10	2	1	52.00(1)	52.00(1)	52.00(1)	52.00(1)	52.00(1)
11	5	0	36.33(0)	25.00(1)	25.00(1)	25.00(1)	29.00(0)
12	2	0	52.00(1)	52.00(1)	52.00(1)	52.00(1)	52.00(1)
13	2	1	154.00(0)	203.00(0)	52.00(1)	154.00(0)	154.00(0)
14	2	1	52.00(1)	52.00(1)	52.00(1)	52.00(1)	52.00(1)
15	4	0	36.33(0)	29.00(1)	36.33(0)	29.00(1)	29.00(1)
16	5	1	36.33(0)	25.00(1)	25.00(1)	36.33(0)	29.00(1)
17	4	0	36.33(0)	29.00(1)	36.33(0)	29.00(1)	29.00(1)
18	2	1	52.00(1)	52.00(1)	36.33(0)	36.33(0)	52.00(1)
19	2	1	52.00(1)	52.00(1)	36.33(0)	52.00(1)	52.00(1)
20	2	0	52.00(1)	52.00(1)	36.33(0)	36.33(0)	52.00(1)
-	Wins	10	13	18	13	15	14
-	Suc. Rate(%)	50	65	90	65	75	70

exhaustive test for this system but as a proof of concept the sample size is assumed to be good enough. In total, the video sequences contained about 302 vehicles to be classified into seven classes. It must be recorded that, since the feature set is small (length, width, contour area, axles), the classification performance is less a test of the decision tree classifier, and more of a test of correct feature detection.

DEVEC shows excellent performance while detecting the passenger vehicles achieving a precision of 0.90 for this class of vehicles. The precision for truck types I, II, III, and IV is also decent with all classes having a value above 0.80. The system however shows a less than satisfactory performance on large trucks having five or more axles. One reason as to why DEVEC does not accurately distinguishes between these two classes is the immense edge discrepancies on and along the close axles of these trucks, which prohibits them from being detected. As a consequence, being of the same height, and width, truck type VI is classified as truck type V. Another reason of low precision and recall might be the limited number of truck type V, and VI samples present in the corresponding videos.

Table 5.2. Effect of increasing NP and function evaluations on success rate. Saturation point is reported at 40-60 combination.

Population Size (NP)	Success Rate %	Saturation FEs
10	70	60
20	75	50
30	85	50
40	90	60
50	90	70
60	90	70
70	90	85
80	90	90

Table 5.3. Confusion matrix for cumulative vehicle count across three video sequences. PV stands for passenger vehicle.

-	PV/Truck Type 0	Truck Type I	Truck Type II	Truck Type III	Truck Type IV	Truck Type V	Truck Type VI
PV/ Truck Type 0	134	5	0	0	0	0	0
Truck Type I	14	60	3	0	0	0	0
Truck Type II	0	8	31	2	0	0	0
Truck Type III	0	0	4	23	2	0	0
Truck Type IV	0	0	0	5	18	0	0
Truck Type V	0	0	0	0	2	9	2
Truck Type VI	0	0	0	0	0	3	6

On a manual inspection of the test video sequences, a total of 326 vehicles were counted. DEVEC was able to detect 302 vehicles achieving a decent detection accuracy of 92%. One reason for misdetection of vehicles was the almost similar color configuration of the background and the passing vehicle, though this was a rare observation. In some circumstances, vehicles that moved very slowly, became a part of the background itself thereby generating no motion pixels through the background subtraction method. One method of improving the detection further would be use more scene cues for background construction that just the color information.

5.5. Summary

This work presents DEVEC, an axle count based vehicle classifier capable of detecting and classifying vehicles on a two lane urban environment. This work uses an adaptive threshold updation technique to compare two subsequent frames, and then employ background subtraction method to extract motion pixels from a video sequence. Axle information is utilized to perform classification of vehicles. Axle detection is

Table 5.4. Confusion matrix for individual classes.

Vehicle Type	Precision	Recall
PV/Truck Type 0	0.90	0.96
Truck Type I	0.82	0.77
Truck Type II	0.81	0.75
Truck Type III	0.82	0.79
Truck Type IV	0.81	0.78
Truck Type V	0.75	0.69
Truck Type VI	0.75	0.66

performed with Hough Transform, a parameterized feature detection method. The working parameters set of Hough Transform is dependent on the image data, and type of problem being addressed. With changing weather conditions and scene outlook, manually setting these parameters is tedious, and as shown in this work often produces less than satisfactory results. To circumvent this problem, a combinatorial version of Differential Evolution was used to optimize the parameter set yielding much higher accuracy as shown by the results this work achieved. Five different variants of DE were tested initially, and it was observed that DE/Rand/1/bin is most suitable for this task reaching a steady success rate of 90% while excluding the false positives. To further improve the speed and accuracy of the system, DE/Rand/1/bin was further investigated. For this, this variant was tested with multiple population sizes (NP) – FEs combinations. It is observed that F=0.5, Cr=0.9, and NP=40 with 60 FEs yields an accuracy of around 90%, and increasing NP further does not yield any better results. Due to speed considerations, as of now, this current system is suited to be used as an offline vehicle classifier. To make the system perform as an online classifier, I will in the future work, consider making some changes. For example, by careful camera calibration, it is possible to specify a region of interest in the test frame where the probability of finding the axles is quite high given various assumptions about inclination of the road. This will reduce the computing load considerably. If there is enough information available about the scene, it is possible to initialize DE with good values to begin with. These and other modifications are planned as future work. Moreover, background construction that uses most of the compute cycle need to be refined further, like using a Gaussian mixture model type pixel classification instead of per-pixel comparison to classify, to speed up the process.

In addition, the novel DEVEC presents some advantages as well as challenges. For example, it does away the need for a shadow removal module, as the camera is mounted sideways. This camera view also restricts the field of view. Occlusion has been handled successfully in moderate traffic. The speed of the system remains an issue as the frame processing rate currently achieved stands at 13 fps on average. This is not yet suitable for online classification. Future work, in this regard, includes developing an improved and speedy background construction algorithm to improve the frame rate. A parallel version of DE may also be investigated to identify the axles quickly.

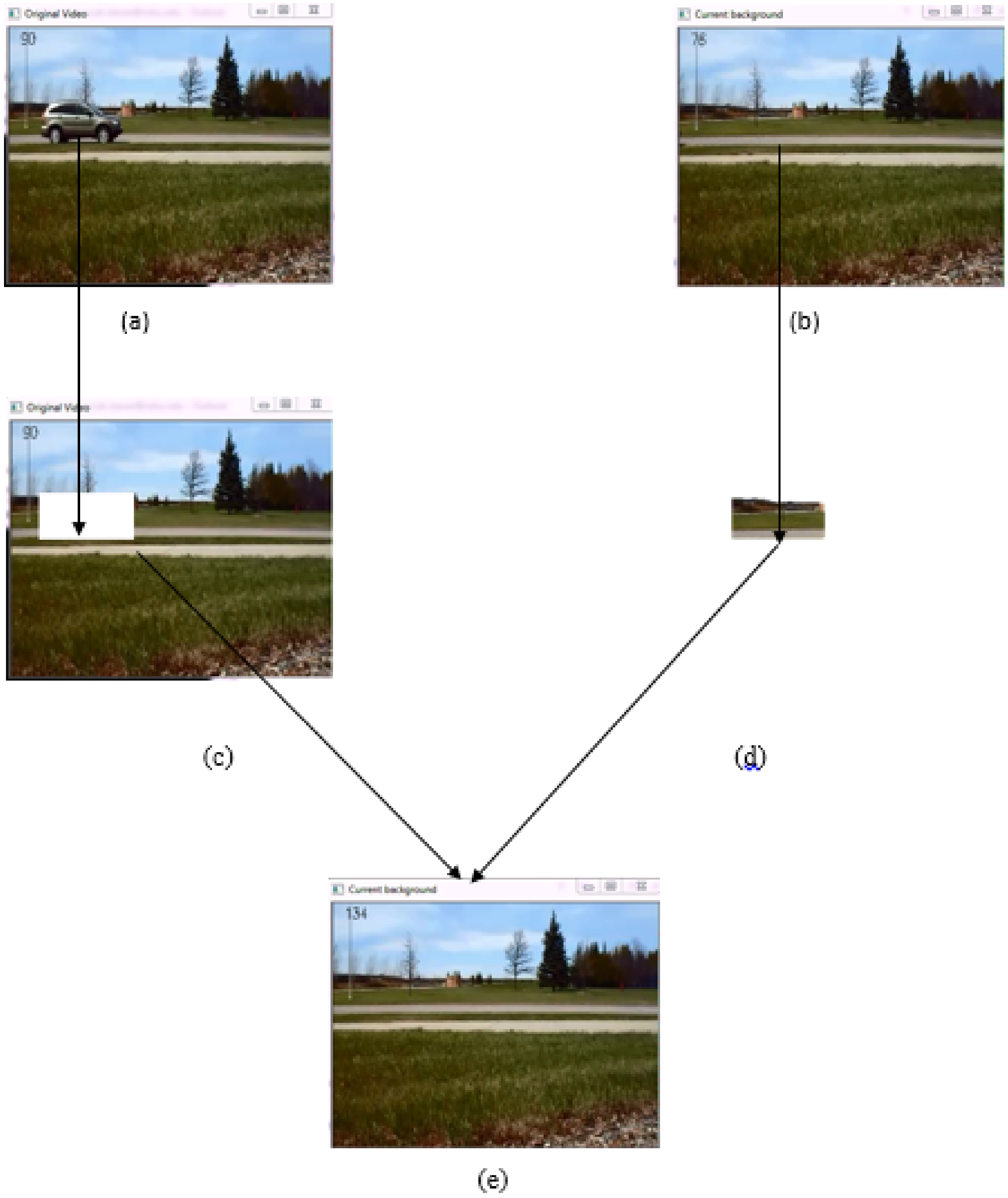


Figure 5.4. A step by step description of adaptive background construction

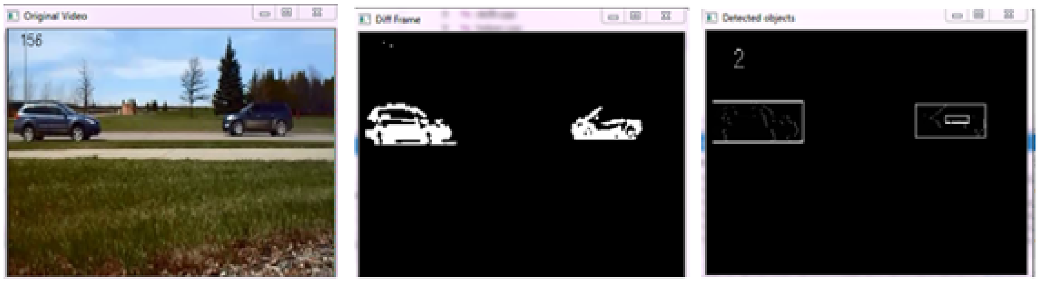
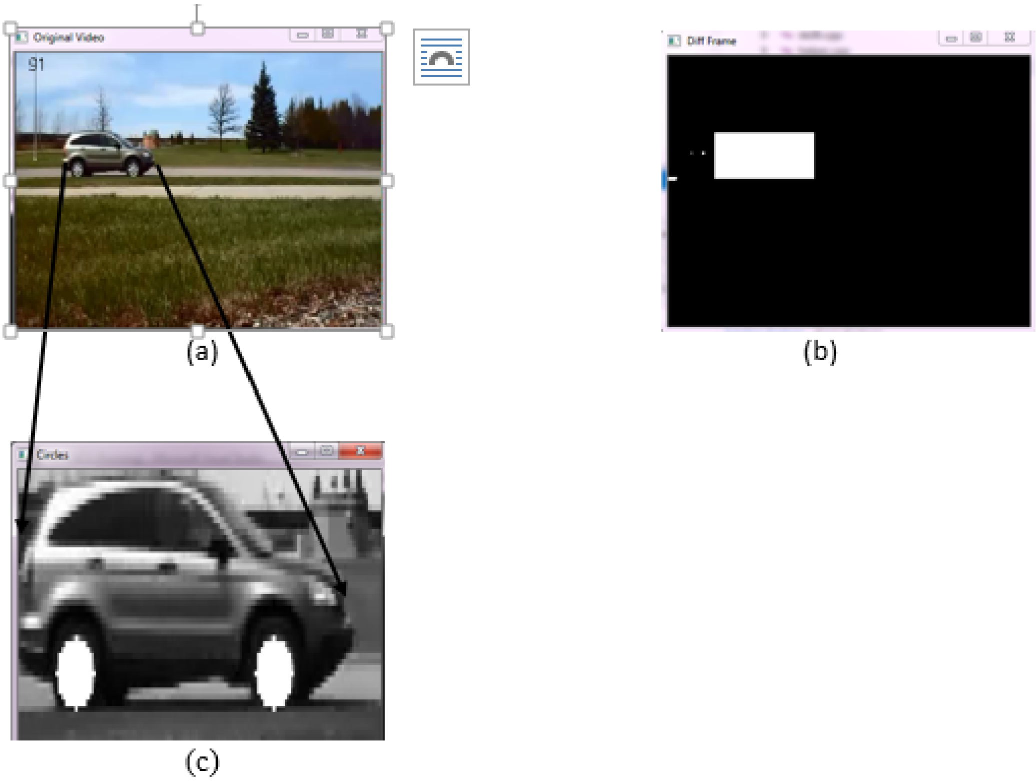


Figure 5.5. Vehicle detection process



(a)

(b)

(c)

Figure 5.6. Axle detection using DE

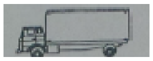
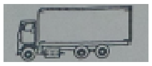
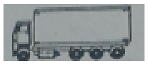
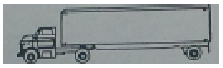
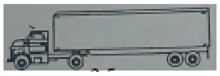
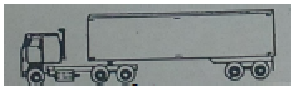
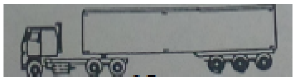
S. No.	Vehicle	No. of Axles	Vehicle Class
1		2	Passenger Vehicle/Truck Type 0
2		3	Truck Type I
3		4	Truck Type II
4		3	Truck Type III
5		4	Truck Type IV
6		5	Truck Type V
7		6	Truck Type 1

Figure 5.7. Vehicle outlines and their associated classes. Axle count information is necessary to distinguish between Truck Type I and II, Truck Type III and IV, and Truck Type V and VI

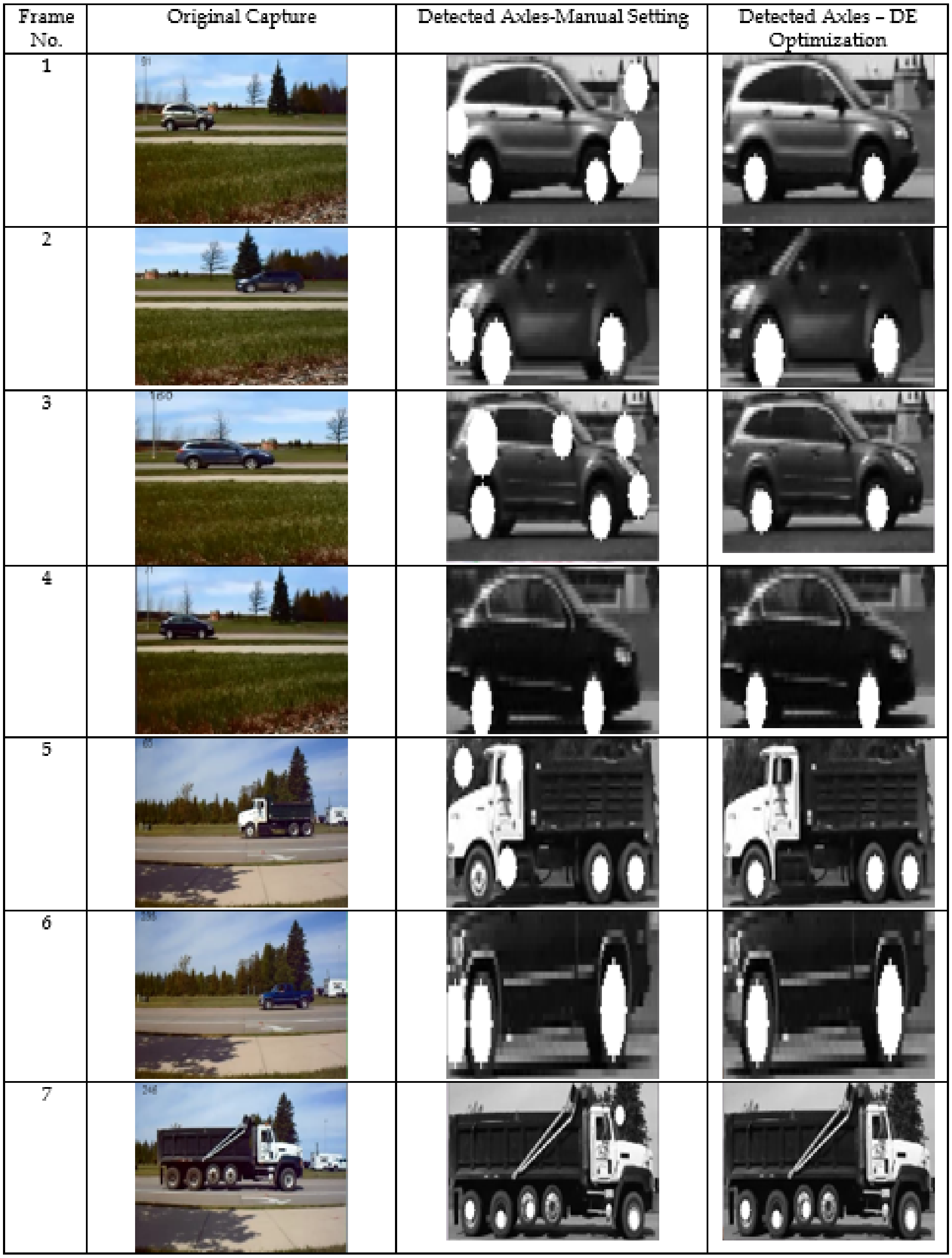

Frame No.
Original Capture
Detected Axles-Manual Setting
Detected Axles – DE Optimization
1
2
3
4
5
6
7

Figure 5.8. Results obtained through manual settings (third column) of Hough Transform parameters vs the best settings obtained for DE/Rand/1/bin (fourth column)

6. CONCLUSION

This study was set out to explore the suitability of Differential Evolution (DE), a control parameter dependent evolutionary algorithm, to be applied as a real and discrete parameter optimizer on hard benchmark optimization problems, and real life scenarios primarily vehicle detection and classification. The study has also sought to improve the classic DE algorithm comparing a modified DE variant with state-of-the-art evolutionary algorithms to judge its overall applicability and robustness. Adding to the general literature on DE, this study sought to answer the following research questions:

- What is the impact of linear and non-linear scale factor reduction on the performance of DE?
- What is the impact of adaptation of mutation strategy on the performance of DE both in presence and absence of control parameter adaptation?
- What is the suitability of DE in parameter based vehicle detection and classification systems as a real or discrete parameter optimizer?

The main empirical observations, results and findings are specific to the chapters and were summarized separately. A unified summary of the findings is the following. It is observed that scale factor reduction has a significant impact on the performance of DE. A statistical study conducted on twenty benchmark problems suggests that a combination of Dither and non-linear scale factor reduction significantly outperforms the classical DE algorithm and this technique stands competitive when compared with the state-of-the-art algorithm SaDE.

It is also reported that the adaptation of mutation strategy is a highly useful mechanism to enhance the performance of DE. This mechanism is useful regardless of the presence of control parameter adaptation. This study therefore, after exhaustive experimentation on 28 benchmark problems and statistical comparison with the state-of-the-art algorithms, conclude that strategy adaptation should be used liberally when applying DE to hard optimization problems.

As for the last question this work set out to answer, it was found that DE is particularly useful for vehicle classification systems that would want to make use of number of axles as a cue for robust vehicle classification. It is observed that the classic DE algorithm, in some cases, is able to achieve an average accuracy improvement of 30%-40% over a manually initialized and parameterized vehicle classification system. This observation, of course, is then restricted to axle based vehicle classification systems.

From the future work's perspective, the scope and scale of this study is extensive, and may be expanded upon broadly. To integrate DE into commercial vehicle classifiers, which tend to be online, more experimentation is required particularly to improve its speed and accuracy. The results and conclusions obtained on scale factor alterations and strategy adaptation may be used to create a variant of DE that may match the requirements of a robust online axle based vehicle classifier.

REFERENCES

[1] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, ser. Natural Computing. Berlin, Germany: Springer-Verlag, 2003.

[2] G. Beni, and J. Wang *Swarm Intelligence in Cellular Robotic Systems*. In Proceedings of the NATO Advanced Workshop on Robots and Biological Systems, Tuscany, Italy, 1993.

[3] P. J. Angeline, Adaptive and self-adaptive evolutionary computation, in: M. Palaniswami, Y. Attikiouzel, R.J. Marks, D.B. Fogel, T. Fukuda (Eds.), Computational Intelligence: A Dynamic System Perspective, IEEE Press, 1995, pp. 152-161.

[4] A. E. Eiben, R. Hinterding, Z. Michalewicz, Parameter control in evolutionary algorithms, IEEE Transactions on Evolutionary Computation 3 (1999) 124-141.

[5] J. Gomez, D. Dasgupta, F. Gonazalez, Using adaptive operators in genetic search, in: Proceedings of the Genetic and Evolutionary Computation Conference 2003 (GECCO'03), Chicago, Illinois, USA, 2003, pp. 1580-1581.

[6] B. R. Julstrom, What have you done for me lately? Adapting operator probabilities in a steady-state genetic algorithm, in: Proceedings of the 6th International Conference on Genetic Algorithms, Pittsburgh, PA, USA, 1995, pp. 81-87.

[7] J. E. Smith, T.C. Fogarty, Operator and parameter adaptation in genetic algorithms, Soft Computing 1 (June) (1997) 81-87.

[8] A. Tuson, P. Ross, Adapting operator settings in genetic algorithms, Evolutionary Computation 6 (1998) 161-184.

[9] R. M. Storn and K. V. Price, “Differential evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces,” International Computer Science Institute, Berkeley, CA, USA, ICSI Technical Report 95-012, Mar. 1995.

[10] S. Das and P. N. Suganthan, “Differential evolution - A survey of the state-of-the-art,” IEEE Transactions on Evolutionary Computation, vol. 15, no. 1, pp. 4-31, Feb. 2011.

[11] R. M. Storn and K. V. Price, “Minimizing the real functions of the ICEC 1996 contest by differential evolution,” in Proc. IEEE Int. Conf. Evol. Comput., 1996, pp. 842-844.

[12] J. Liu and J. Lampinen, “On setting the control parameter of the differential evolution method” in Proc. 8th Int. Conf. Soft Computing (MENDEL), 2002, pp. 11-18 .

[13] R. Gamperle, S. D. Muller, and P. Koumoutsakos, “A parameter study for differential evolution” NNA-FSFS-EC 2002. Interlaken, Switzerland, WSEAS, Feb. 11-15, 2002.

[14] K. Price, R. Storn, and J. Lampinen, “Differential Evolution - A Practical Approach to Global Optimization” Berlin, Germany: Springer, 2005.

[15] S. Das, A. Konar, and U. K. Chakraborty, “Two improved Differential Evolution schemes for faster global search,” in Proc. ACM-SIGEVO GECCO, Jun. 2005, pp. 991-998.

[16] H. A. Abbass, “The self-adaptive pareto differential evolution algorithm,” in Proceedings of the 2002 IEEE Congress on Evolutionary Computation (CEC’02), Honolulu, Hawaii, USA, vol. 1, pp. 831-836, 2002.

[17] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, “Self adapting control parameters in differential evolution: A comparative study on numerical benchmark problems,” IEEE Transactions on Evolutionary Computation, vol. 10, no. 6, pp. 646-657, Dec. 2006.

[18] A. K. Qin, V. L. Huang, P. N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Transactions on Evolutionary Computation 13, April 2009, pp. 398-417.

[19] M. G. H. Omran, A. Salman, A. P. Engelbrecht, Self-adaptive differential evolution, in: Computational Intelligence and Security, PT 1, Proceedings Lecture Notes in Artificial Intelligence, 2005, pp. 192-199.

[20] D. Zaharie, Control of population diversity and adaptation in differential evolution algorithms, in: Proceedings of the 9th International Conference on Soft Computing, Brno, 2003, pp. 41-46.

[21] J. Tvrdik, Adaptation in differential evolution: a numerical comparison, Applied Soft Computing 9 (June) (2009) 1149-1155.

[22] R. Mallipeddi, P.N. Suganthana, Q.K. Pan, M.F. Tasgetiren, "“Differential evolution algorithm with ensemble of parameters and mutation strategies,” Appl. Soft Comput.,” Volume 11 Issue 2, March, 2011 Pages 1679-1696.

[23] H. K. Yuen, J. Princen, J. Illingworth, and J. Kittler, “Comparative study of Hough transform methods for circle finding,” Image Vision Comput. vol. 8, no. 1, pp. 71-77, 1989.

[24] I. Haritaoglu and M. Flickner, *Real-time surveillance of people and their activities,* IEEE Trans. Pattern Anal. Mach. Intell. 22, 8, 809-830, 2000.

[25] B. Coifman, D. Beymer, P. Mclauchlan and J. Malik, *A real-time computer vision system for vehicle tracking and traffic surveillance,* Transport. Res. Part C 6, 4, 271-288.

[26] O. Masoud, N. P. Papanikolupoulos, *A novel method for tracking and counting pedestrians in real-time using a single camera,* IEEE Trans. Vehicul. Technol. 50, 5, 1267-1278.

[27] J. Tai, S. Tsang, C. Lin, AND K. Song, *Real-time image tracking for automatic traffic monitoring and enforcement application,* Image Vision Comput. 22, 6, 485-501, 2004.

[28] G. W. Greenwood, *Finding Solutions to NP Problems,*, Published in proceedings CEC 2001 , pp. 815-822, 2001.

[29] D. Karabogam and S. Okdem, “A Simple and Global Optimization Algorithm for Engineering Problems: Differential Evolution Algorithm,” in Turkish Journal of Electrical Engineers, vol.12(1), pp. 53-60, 2004.

[30] R. Storn, “Digital Filter Design Program” FIWIZ 2000 [Online]. Available: http://www.icsi.berkeley.edu/~storn/fiwiz.html.

[31] R. Hooke And T. A. Jeeves, “Direct Search Solution of Numerical and Statistical Problems,” Journal of the ACM (JACM), vol. 8, Issue 2, pp. 212-229, April 1961.

[32] U. K. Chakraborty, “Advances in Differential Evolution,” in Differential Evolution Research-Trends and Open Questions, Springer, 2008, pp. 11-12.

[33] J. J. Liang, B.Y. Qu, P. N. Suganthan, and A. G. Hernandez-Dıaz, “Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization,” Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang, 2013.

[34] A. K. Qin and P. N. Suganthan, ”Self-adaptive Differential Evolution Algorithm for Numerical Optimization”, Proc. IEEE Congress on Evolutionary Computation, Sept. 2005.

[35] A. K. Qin, X. Li, “Differential Evolution on the CEC-2013 Single-Objective Continuous Optimization Testbed,” IEEE Congress on Evolutionary Computation, Cancun, Mexico, June 20-23, 2013.

[36] M. Friedman, “The use of ranks to avoid the assumption of normality implicit in the analysis of variance,” Journal of the American Statistical Association 32, 1937, pp. 674-701.

[37] Y. Hochberg, “A sharper Bonferroni procedure for multiple tests of significance,” Biometrika, 1988, pp. 800-803.

[38] A. K. Qin, X. Li, H. Pany, S. Xiay, ”Investigation of Self-adaptive Differential Evolution on the CEC-2013 Real-Parameter Single-Objective Optimization Testbed,” IEEE Congress on Evolutionary Computation, Cancun, Mexico, June 20-23, 2013.

[39] R. Storn, K. Price, Differential evolution— “A simple and efficient heuristic for global optimization over continuous spaces,” Journal of Global Optimization, 11 (1997) 341-359.

[40] J. Lampinen, I. Zelinka, On stagnation of the differential evolution algorithm, in: Proceedings of MENDEL 2000, 6th International Mendel Conference on Soft Computing, (2000) 76-83.

[41] J. Ronkkonen, S. Kukkonen, K. V. Price, Real parameter optimization with differential evolution, in Proceedings of IEEE Congress on Evolutionary Computation, 1 (2005) 506-513.

[42] J. Liu, J. Lampinen, A Fuzzy Adaptive Differential Evolution Algorithm, in: Soft Computing, A Fusion of Foundations, Methodologies and Applications, 9 (6) (2005) 448-462.

[43] F. Neri, V. Tirronen, Recent advances in differential evolution: a survey and experimental analysis Artificial Intelligence Review, 33 (1-2) (2010) 61-106.

[44] J. Ronkkonen, J. Lampinen, On using normally distributed mutation step length for the differential evolution algorithm, in: Proceedings of the 9th Int. Conf. on Soft Computuing MENDEL, Brno, Czech Republic (2003) 11-18.

[45] M. M. Ali, A. Torn, Population set based global optimization algorithms: Some modifications and numerical studies, Journal of Computers and Operations Research, 31 (10) (2004) 1703-1725.

[46] D. Dawar, S. A. Ludwig, Differential evolution with dither and annealed scale factor, in: Proceedings of the IEEE Symposium Series on Computational Intelligence, Orlando, Florida, U.S.A., (2014) 1-8.

[47] J. Teo, Exploring dynamic self-adaptive populations in differential evolution, Soft Computing - A Fusion of Foundations, Methodologies and Applications, 10 (8) (2006) 673-686.

[48] J. Brest, M. S. Mauèec, Population size reduction for the differential evolution algorithm, Applied Intelligence, 29 (3) (2008) 228-247.

[49] J. Zhang, A. C. Sanderson, JADE: Adaptive differential evolution with optional external archive, IEEE Transaction on Evolutionary Computation, 13 (5) (2009) 945-958.

[50] E. Mezura-Montes, J. Velazquez-Reyes, C. A. Coello Coello, A comparative study of differential evolution variants for global optimization, in GECCO (2006) 485-492.

[51] F. Peng, K. Tang, G. Chen, X. Yao, Multi-start JADE with knowledge transfer for numerical optimization, in: Proceedings of the IEEE CEC (2009) 1889-1895.

[52] Z. Yang, J. Zhang, K. Tang, X. Yao, A. C. Sanderson, An adaptive coevolutionary differential evolution algorithm for large-scale optimization, in: Proceedings of the IEEE CEC (2009) 102-109.

[53] W. Gong, Z. Cai, C. X. Ling, H. Li, Enhanced differential evolution with adaptive strategies for numerical optimization, IEEE Transactions on Systems, Man, and Cybernetics, PartB, 41 (2) (2011) 397-413.

[54] J. Zhang, V. Avasarala, A. C. Sanderson, T. Mullen, Differential evolution for discrete optimization: An experimental study on combinatorial auction problems, in: Proceedings of the IEEE CEC (2008) 2794-2800.

[55] J. Zhang, A. C. Sanderson, Self-adaptive multi-objective differential evolution with direction information provided by archived inferior solutions, in: Proceedings of the IEEE CEC (2008) 2801-2810.

[56] R. Tanabe, A. Fukunaga, Success-History Based Parameter Adaptation for Differential Evolution, in: Proceedings of the IEEE CEC (2013) 71-78.

[57] R. Tanabe, A. Fukunaga, Evaluating the performance of SHADE on CEC 2013 benchmark problems, in: Proceedings of the IEEE CEC (2013) 1952-1959.

[58] A. Auger, N. Hansen, A Restart CMA Evolution Strategy With Increasing Population Size, in: Proceedings of the IEEE CEC (2005) 1769-1776.

[59] C. Garcia-Martınez, M. Lozano, F. Herrera, D. Molina, A. M. Sanchez, Global and local real-coded genetic algorithms based on parent-centric crossover operators, European Journal of Operations Research, 185 (3) (2008) 1088-1113.

[60] M. A. M. de Oca, T. Stutzle, K. V. den Enden, M. Dorigo, Incremental Social Learning in Particle Swarms, IEEE Transactions on Systems, Man, and Cybernetics, PartB, 41 (2) (2011) 368-384.

[61] J. L. J. Laredo, C. Fernandes, J. J. M. Guervos, C. Gagne, Improving Genetic Algorithms Performance via Deterministic Population Shrinkage, in: Proceedings of the GECCO (2009) 819-826.

[62] R. Tanabe, A. Fukunaga, Improving the Search Performance of SHADE Using Linear Population Size Reduction, in: Proceedings of the IEEE CEC (2014) 1658-1665.

[63] J. Brest, A. Zamuda, B. Boskovic, M. S. Maucec, V. Zumer, Highdimensional real-parameter optimization using self-adaptive differential evolution algorithm with population size reduction, in: Proceedings of the IEEE Congress on Evolutionary Computation (2008) 2032-2039.

[64] B. V. Babu, S. A. Munawar, Optimal design of shell-and-tube heat exchangers bu different strategies of Differential Evolution, Technical Report PILANI -333 031, Department of chemical engineering, BITS, Rajasthan, India (2001).

[65] J. Vesterstrom, R. A. Thomson, Comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems, in: Proceedings of the IEEE Congress on Evolutionary Computation (2004) 1980-1987.

[66] A. Zamuda, J. Brest, B. Boškovic, V. Žumer. Large scale global optimization using differential evolution with self-adaptation and cooperative co-evolution, in: Proceedings of the 2008 IEEE World Congress on Computational Intelligence (2008) 3719-3726.

[67] Z. Yang, K. Tang, X. Yao. Self-adaptive differential evolution with neighborhood search. In Proceedings of the IEEE Congress on Evolutionary Computation (2008) 1110-1116.

[68] L. Xie, G. Zhu, Y. Wang, H. Xu, and Z. Zhang, “Robust vehicles extraction in a video-based intelligent transportation system,” in Proc. IEEE International Conference on Communications, Circuits and Systems, vol. 2, pp. 887-890, 2005.

[69] A. Iorio, X. Li, Solving rotated multi-objective optimization problems using differential evolution, in: Australian Conference on Artificial Intelligence, Cairns, Australia (2004) 861-872.

[70] S. Das, A. Abraham, U.K. Chakraborthy, Differential evolution using a neighborhood-based mutation operator, IEEE Transactions on Evolutionary Computation 13 (June) (2009) 526-553.

[71] L. F. D. Costa and R. M. Cesar Jr., “Shape Analysis and Classification,” CRC Press, Inc. Boca Raton FL, U.S.A, 2000.

[72] A. J. Lipton, H. Fujiyoshi, and R. S. Patil, “Moving target classification and tracking from real-time video,” in Proc. IEEE Workshop Applications of Computer Vision, 1998, pp. 8-14.

[73] D. Koller, “Moving object recognition and classification based on recursive shape parameter estimation,” in Proc. 12th Israel Conf. Artificial Intelligence, Computer Vision, and Neural Networks Dec. 27-28, 1993, pp. 359-368.

[74] K. D. Baker and G. D. Sullivan, “Performance assessment of model based tracking,” in Proc. IEEE Workshop Applications of Computer Vision, Palm Springs, CA, 1992, pp. 28-35.

[75] G. D. Sullivan, “Model-based vision for traffic scenes using the ground plane constraint,” Phil. Trans. Roy. Soc. (B), vol. 337, pp. 361-370, 1992.

[76] G. D. Sullivan, A. D.Worrall, and J. M. Ferryman, “Visual object recognition using deformable models of vehicles,” in Proc. Workshop on Context-Based Vision, Cambridge, MA, Jun. 1995, pp. 75-86.

[77] S. Gupte, O. Masoud, R. F. K. Martin, and N. P. Papanikolopoulos, “Detection and Classification of Vehicles,” in IEEE Transactions on Intelligent Transportation Syatems, vol. 3, no. 1, pp. 37-47, Mar. 2002.

[78] E. Rivlin, M. Rudzsky, M. Goldenberg, U. Bogomolov, and S. Lapchev, “A real-time system for classification of moving objects,” in Proc. International Conference on Pattern Recognition, vol. 3, pp. 668-691, 2002.

[79] J. Cao and L. Li, “Vehicle objects detection of video images based on gray-scale characteristics,” in Proc. IEEE International Conference on Education Technology and Computer Science, 2009, pp. 936-940.

[80] B. T. Morris and M. M. Trivedi, “Learning, modeling, and classification of vehicle track patterns from live video,” IEEE Transactions on Intelligent Transportation Systems, vol. 9, no. 3, pp. 425-437, 2008.

[81] L. Anan, Y. Zhaoxuan, and L. Jintao, “Video vehicle detection algorithm based on virtual-line group,” in Proc. IEEE Asia Pacific Conference on Circuits and Systems, 2006, pp. 1148-1151.

[82] J. Wu, Z. Yang, J. Wu, and A. Liu, “Virtual line group based video vehicle detection algorithm utilizing both luminance and chrominance,” in Proc. IEEE Conference on Industrial Electronics and Applications, 2007, pp. 2854-2858.

[83] Y. Hue, “A traffic-flow parameters evaluation approach based on urban road video,” International Journal of Intelligent Engineering and Systems, vol.2, no.1, pp. 33-39, 2009.

[84] J. Iivarinen, M. Peura, J. Särelä, and A. Visa, “Comparison of combined shape descriptors for irregular objects,” in 8th Proc. British Machine Vision Conf., Cochester, UK, pp. 430-439, 1997.

[85] G. A. Jones, J. Princen, J. Illingworth, and J. Kittler, “Robust estimation of shape parameters,” in Proc. British Machine Vision Conf., 1990, pp. 43-48.

[86] G. Bongiovanni, P. Crescenzi, and C. Guerra, “Parallel Simulated Annealing for Shape Detection,” Computer Vision and Image Understanding, vol. 61, no. 1, pp. 60-69, 1995.

[87] G. Roth and M.D. Levine, “Geometric primitive extraction using a genetic algorithm,” IEEE Trans. Pattern Anal. Machine Intell. vol. 16, no 9, pp. 901–905, 1994.

[88] M. Peura and J. Iivarinen,“Efficiency of simple shape descriptors”, Advances in Visual Form Analysis. World Scientific, Singapore, pp. 443–451, 1997.

[89] H. Muammar, M. Nixon, “Approaches to extending the Hough transform,” in Proc. Int. Conf. on Acoustics, Speech and Signal Processing ICASSP 89, vol. 3, pp. 1556-1559, 1989.

[90] D. Shaked, O. Yaron, and N. Kiryati, “Deriving stopping rules for the probabilistic Hough transform by sequential analysis,” Comput. Vision Image Understanding vol 63, pp 512-526, 1996.

[91] L. Xu, E. Oja, and P. Kultanen, “A new curve detection method: Randomized Hough transform (RHT),” Pattern Recognition Lett. vol. 11, no 5, pp. 331-338, 1990.

[92] J. Becker, S. Grousson, and D. Coltuc, “From Hough transforms to integral transforms,” in Proc. Int. Geoscience and Remote Sensing Symp., IGARSS 02, vol. 3, pp. 1444-1446, 2002.

[93] V. Ayala-Ramirez, C. H. Garcia-Capulin, A. Perez-Garcia, and R. E. Sanchez-Yanez, “Circle detection on images using genetic algorithms,” Pattern Recognition Letters, vol. 27, 652-657, 2006.

[94] S. Das, S. Dasgupta, A. Biswas, and A. Abraham, “Automatic Circle Detection on Images with Annealed Differential Evolution,” in 8th International Conference on Hybrid Intelligent Systems, pp. 684-689, 2008.

[95] G. W. Corder and D. I. Foreman, “Nonparametric Statistics: A Step-by-Step Approach,” Wiley, New York, 2014.

[96] F. Goudail, P. Réfrégier, and P. Delyon, “Bhattacharyya distance as a contrast parameter for statistical processing of noisy optical images,” JOSA A, vol. 21, no. 7, pp. 1231-1240, 2004.

[97] G. Onwubolu, and D. Davendra, “Differential Evolution: A Handbook for Global Permutation-Based Combinatorial Optimization,” Springer-Verlag, Heidelberg, 2009.

[98] G. Onwubolu, and D. Davendra, “Scheduling flow shops using differential evolution algorithm,” Eur. J. Oper. Res. vol. 171, no. 2, pp. 674-692, 2006.

[99] D. Davendra, and G. Onwubolu, “Forward Backward Trasformation,” Differential Evolution A Handbook for Global Permutation-Based Combinatorial Optimization, pp. 37-78, Springer, Heidelberg, 2009.

[100] A. P. Engelbrecht, “Fitness Function Evaluations: A Fair Stopping Condition?,” IEEE Symposium Series on Computational Intelligence, Orlando, Florida, U.S.A, Dec 9-12, 2014.

[101] J. Zhang, F. Y. Wang, K. Wang, W. H. Lin, X. Xu, and C. Chen, “Data-driven intelligent transportation systems: a survey.” IEEE Transactions on Intelligent Transportation Systems, vol. 12 no. 4, pp. 1624-1639, 2011.

[102] A. Sharma, M. Harding, B. Giles, D. Bullock, J. Sturdevant, and S. Peeta, “Performance Requirements and Evaluation Procedures for Advance Wide Area Detectors.” Submitted to the Transportation Research Board for publication and presentation at the 87th Annual Meeting, Washington, D.C., 2008.

[103] X. Mao, D. Inoue, S. Kato, and M. Kagami. “Amplitude-modulated laser radar for range and speed measurement in car applications.” IEEE Transactions on Intelligent Transportation Systems vol. 13, no. 1, pp.408-413, 2012.

[104] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Kolter, D. Langer, O. Pink, V. Pratt, M. Sokolsky, G. Stanek, D. Stavens, A. Teichman, M. Werling, and S. Thrun, “Towards fully au-

tonomous driving: Systems and algorithms." Proceedings of the IEEE Intelligent Vehicle Symposium, pp. 163–168, 2011.

[105] 3M Canoga Application Note. "Canoga Vehicle Detection System." Accurate Vehicle Detection at Intersections Using 3M Canoga M702 Non-Invasive Microloops TM-2003-11, 2003.

[106] D. Middleton, H. Chara, and R. Longmire. "Alternative vehicle detection technologies for traffic signal systems." Technical Report 2009. Retrieved May 5th, 2016, from http://d2dtl5nnlpfr0r.cloudfront.net/tti.tamu.edu/documents/0-5845-1.pdf

[107] B. Tian, B. T. Morris, M. Tang, Y. Liu, Y. Yao, C. Gou, D. Shen and S. Tang, "Hierarchical and Networked Vehicle Surveillance in ITS: A Survey". IEEE Transactions on Intelligent Transportation Systems, vol. 16, no. 2, pp. 557-580, Apr 2015.

[108] S. Sivaraman, and M. M. Trivedi, "Looking at vehicles on the road: A survey of vision based detection, tracking, and behavior analysis." IEEE Transactions on Intelligent Transportation Systems, vol. 14, no. 4, pp. 1773-1795, 2015.

[109] X. Wang, "Intelligent multi-camera video surveillance: A review." Pattern Recognition Letters vol. 34, pp. 3–19, 2013.

[110] S. Linda, and M. T. Volling, "Paradigm is shifting in vehicle detection." IMSA journal featured article, 2003. Retrieved May 6th, 2016, from http://www.imsasafety.org/journal/jf03/janfeb3.htm

[111] Q. Li, and J. F. He, "Vehicles detection based on three-frame-difference method and cross-entropy threshold method," Computer Engineering, vol. 37, no. 4, pp. 172-174, 2011.

[112] K. Park, D. Lee, and Y. Park, "Video-based detection of street parking violation", Proceedings of International Conference on Image Processing, Computer Vision, and Pattern Recognition(IPCV), pp. 152-156, 2007.

[113] P. V. Nguyen, and H. B. Le, "A multi-modal particle filter based motorcycle tracking system", In PRICAI 2008: Trends in Artificial Intelligence, pp. 819-828. Springer Berlin Heidelberg, 2008.

[114] S. Gupte, O. Masoud, R. F. K. Martin, and N. P. Papanikolopoulos, “Detection and classification of vehicles,” IEEE Trans. Intell. Transp. Syst., vol. 3, no. 1, pp. 37–47, Mar. 2002.

[115] A. Lai and N. H. C. Yung, “A fast and accurate scoreboard algorithm for estimating stationary backgrounds in an image sequence,” in Proc. IEEE Int. Symp. Circuits Syst., vol. 4, pp. 241–244, May 1998.

[116] P. Kumar, S. Ranganath, and H.Weimin, “Bayesian network based computer vision algorithm for traffic monitoring using video,” in Proc. Int. IEEE Conf. Intell. Transp. Syst., 2003, vol. 1, pp. 897–902.

[117] B. T. Morris and M. M. Trivedi, “Learning, modeling, and classification of vehicle track patterns from live video,” IEEE Trans. Intell. Transp. Syst., vol. 9, no. 3, pp. 425–437, Sep. 2008.

[118] C. Stauffer andW. E. L. Grimson, “Adaptive background mixture models for real-time tracking,” in Proc. IEEE Conf. Comput. Vis. Pattern Recog., vol. 2, pp. 252, 1999.

[119] J. Zheng, Y. Wang, N. L. Nihan, and M. E. Hallenbeck, “Extracting roadway background image: Mode-based approach,” J. Transp. Res. Board, vol. 1944, no. 1, pp. 82–88, 2006.

[120] S. C. Sen-Ching, and C. Kamath, “Robust techniques for background subtraction in urban traffic video,” International Society for Optics and Photonics in Electronic Imaging, pp. 881-892, Jan. 2004.

[121] M. Haque, M. M. Murshed, and M. Paul, “Improved Gaussian mixtures for robust object detection by adaptive multi-background generation,” IEEE 19th International Conference on Pattern Recognition (ICPR), pp. 1-4, Dec. 2008.

[122] Z. Zivkovic, and F. van der Heijden, “Efficient adaptive density estimation per image pixel for the task of background subtraction,” Pattern recognition letters, vol. 27, no. 7, pp. 773-780, 2006.

[123] N. Greggio, A. Bernardino, C. Laschi, P. Dario, and J. Santos-Victor, “Self-adaptive Gaussian mixture models for real-time video segmentation and background subtraction,” in Proc. IEEE 10th International Conference on Intelligent Systems Design and Applications(ISDA), Nov. 2010, pp. 983-989.

[124] S. L. Zhao, and H. J. Lee, “A spatial-extended background model for moving blobs extraction in indoor environments,” Journal of Information Science and Engineering, vol. 25, no. 6, pp. 1819-1837, 2009.

[125] B. Johansson, J. Wiklund, P.E. Forssén, and G. Granlund, “Combining shadow detection and simulation for estimation of vehicle size and position,” Pattern Recognition Letters, vol. 30, no. 8, pp. 751-759, 2009.

[126] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, “Detecting moving objects, ghosts, and shadows in video streams,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 25, no. 10, pp. 1337-1342, 2003.

[127] N. J. McFarlane, and C .P. Schofield, “Segmentation and tracking of piglets in images,” Machine vision and applications, vol. 8, no. 3, pp. 187-193, 1995.

[128] A. Elgammal, D. Harwood, L. S. Davis, “Non-parametric Model for Background Subtraction”, 6th European Conference on Computer Vision. Dublin, Ireland, July 2000.

[129] R. E. Kalman, “A new approach to linear filtering and prediction problems,” Journal of Fluids Engineering, vol. 82, no. 1, pp. 35-45, 1960.

[130] K. P. Karmann, A. V. Brandt, and R. Gerl, “Moving object segmentation based on adaptive reference images,” in Proc. of 5th European Signal Processing Conference, 1990, vol. 2, pp. 951-954.

[131] A. Ottlik, and H. H. Nagel, “Initialization of model-based vehicle tracking in video sequences of inner-city intersections,” International Journal of Computer Vision, vol. 80, no. 2, pp. 211-225, 2008.

[132] S. Indu, M. Gupta, and A. Bhattacharyya, “Vehicle tracking and speed estimation using optical flow method,” Int. J. Engineering Science and Technology, vol. 3, no. 1, pp. 429-434, 2011.

[133] S. Agarwal, A. Awan, and D. Roth, “Learning to detect objects in images via a sparse, part-based representation,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 11, pp. 1475-1490, 2004.

[134] X. Ma, and W.E.L. Grimson, “Edge-based rich representation for vehicle classification,” in Proc. IEEE 10th International Conference on Computer Vision, vol. 2, Oct. 2005, pp. 1185-1192.

[135] D. G. Lowe, “Object recognition from local scale-invariant features”, in proc. IEEE 7th international conference on Computer vision, Jan. 1999, vol. 2, pp. 1150-1157.

[136] N. Dalal, and B. Triggs, “Histograms of oriented gradients for human detection,” in proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, Jun. 2005, pp. 886-893.

[137] H. Bay, T. Tuytelaars, and L. Van Gool, “Surf: Speeded up robust features,” in Computer vision (ECCV), Jan. 2006, pp. 404-417. Springer Berlin Heidelberg.

[138] P. Viola, and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in proc. of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001, vol. 1, pp. 511-518.

[139] M. Al-Smadi, K. Abdulrahim, R. A. Salam, “Traffic Surveillance: A Review of Vision Based Vehicle Detection, Recognition and Tracking,” International Journal of Applied Engineering Research ISSN 0973-4562 vol. 11, no. 1 pp 713-726, 2016.

[140] D. Koller, J. Weber, and J. Malik, “Towards realtime visual based tracking in cluttered traffic scenes,” in Proc. of the IEEE Intelligent Vehicles’ 94 Symposium, Oct. 1994, pp. 201-206.

[141] J. Lou, T. Tan, W. Hu, H. Yang, and S.J. Maybank, “3-D model-based vehicle tracking,” IEEE Transactions on Image Processing, vol. 14, no. 10, pp. 1561-1569, 2005.

[142] N. Buch, J. Orwell, and S.A. Velastin, “Detection and classification of vehicles for urban traffic scenes,” in 5th International Conference on Visual Information Engineering, Jul. 2008, pp. 182-187.

[143] Z. Wang, and K. Hong, “A new method for robust object tracking system based on scale invariant feature transform and camshift,” in Proc. 2012 ACM Research in Applied Computation Symposium, Oct. 2012, pp. 132-136.

[144] D. Han, M.J. Leotta, D.B. Cooper, and J.L. Mundy, "Vehicle class recognition from video-based on 3d curve probes," in 2ed Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, , Oct. 2005, pp. 285-292.

[145] Z. Chen, T. Ellis, and S. Velastin, "Vehicle detection, tracking and classification in urban traffic," in 15th International IEEE Conference on Intelligent Transportation Systems (ITSC), Sep. 2012, pp. 951-956.

[146] P. Negri, X. Clady, M. Milgram, and R. Poulenard, "An oriented-contour point based voting algorithm for vehicle type classification," In 18th International Conference on Pattern Recognition, Aug. 2006, vol. 1, pp. 574-577.

[147] I. Zafar, E.A. Edirisinghe, S. Acar, and H.E. Bez, "Two-dimensional statistical linear discriminant analysis for real-time robust vehicle-type recognition," in International Society for Optics and Photonics Electronic Imaging 2007.

[148] N.K. Kanhere, and S.T. Birchfield, "Real-time incremental segmentation and tracking of vehicles at low camera angles using stable features," IEEE Transactions on Intelligent Transportation Systems, vol. 9, no. 1, pp. 148-160, 2008.

[149] X. Chen, and C. Zhang, "Vehicle classification from traffic surveillance videos at a finer granularity," Advances in Multimedia Modeling, pp. 772-781, 2006. Springer Berlin Heidelberg.

[150] C.R. Wren, A. Azarbayejani, T. Darrell, and A.P. Pentland, "Pfinder: Real-time tracking of the human body," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, no. 7, pp. 780-785, 1997.

[151] P. Barcellos, C. Bouvié, F.L. Escouto, and J. Scharcanski, "A novel video based system for detecting and counting vehicles at user-defined virtual loops," Expert Systems with Applications, vol. 42, no. 4, pp. 1845-1856, 2015.

[152] N. A. Mandellos, I. Keramitsoglou, and C. T. Kiranoudis, "A background subtraction algorithm for detecting and tracking vehicles," Expert Systems with Applications, vol. 38, no. 3, pp. 1619-1631, 2011.

[153] J. C. Lai, S. S. Huang, and C. C. Tseng, "Image-based vehicle tracking and classification on the highway," in International Conference on Green Circuits and Systems (ICGCS), Jun. 2010, pp. 666-670.