Design Optimization Of A Parallel Hybrid Powertrain Using Derivative-Free Algorithms

Sachin Kumar Porandla

DESIGN OPTIMIZATION OF A PARALLEL HYBRID POWERTRAIN USING DERIVATIVE-FREE ALGORITHMS

By

Sachin Kumar Porandla

A Thesis
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Electrical Engineering
in the Department of Electrical and Computer Engineering

Mississippi State, Mississippi

December 2005

DESIGN OPTIMIZATION OF A PARALLEL HYBRID POWERTRAIN USING

DERIVATIVE-FREE ALGORITHMS

By

Sachin Kumar Porandla

Approved:

Wenzhong Gao Assistant Research Professor of Electrical and Computer Engineering (Director of Thesis)	Michael Mazzola Professor of Electrical and Computer Engineering (Committee Member)
Herb Ginn Assistant Professor of Electrical and Computer Engineering (Committee Member)	Robert Reese Associate Professor of Electrical and Computer Engineering (Committee Member)
Nicholas Younan Professor of Electrical and Computer Engineering (Graduate Coordinator)	Roger L. King Associate Dean of the Bagley College of Engineering

Name: Sachin Kumar Porandla

Date of Degree: December 9, 2005

Institution: Mississippi State University

Major Field: Electrical Engineering

Major Professor: Dr. Wenzhong Gao

Title of Study: DESIGN OPTIMIZATION OF A PARALLEL HYBRID POWERTRAIN USING DERIVATIVE-FREE ALGORITHMS.

Pages in Study: 66

Candidate for Degree of Master of Science

A Hybrid Electric Vehicle (HEV) is a complex electro-mechanical-chemical system that involves two or more energy sources. The inherent advantages of HEVs are their increased fuel economy, reduced harmful emissions and better vehicle performance. The extent of improvement in fuel economy and vehicle performance greatly depends on selecting optimal component sizes. The complex interaction between the various components makes it difficult to size specific components manually or analytically. So, simulation-based multi-variable design optimization is a possible solution for such kind of system level design problems. The multi-modal, noisy and discontinuous nature of the HEV design requires the use of derivative-free global algorithms because the derivative-based local algorithms work poorly with such design problems.

In this thesis, a Hybrid Vehicle is optimized using various Global Algorithms – DIviding RECTangles (DIRECT), Simulated Annealing (SA), Genetic Algorithm (GA), and Particle Swarm Optimization (PSO). The objective of this study is to increase the overall fuel economy on a composite of city and highway driving cycle and to improve

the vehicle performance. The performance of each algorithm is compared on a six variable hybrid electric vehicle design problem. Powertrain System Analysis Tool (PSAT), a state-of-the-art powertrain simulator is used as the vehicle simulator. Further, a Hybrid algorithm that is a combination of global and local algorithm is developed to improve the convergence of the global algorithms.

DEDICATION

I would like to dedicate this research to my parents and my brothers.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my major professor Dr. Wenzhong Gao for his invaluable assistance during my thesis. His constant guidance, suggestions and innovative ideas helped me very much in completing this thesis.

I would like to extend my special gratitude to Dr. Michael Mazzola, Dr. Herb Ginn and Dr. Robert Reese for their willingness to serve on my graduate committee and more importantly for their support and valuable suggestions.

I would like to thank Aymeric Rousseau at the Center for Transportation Research of Argonne National Laboratory for his technical assistance.

I would also like to thank Center for Advanced Vehicular Systems (CAVS) for giving me a GRA and extending financial support during my research work.

Finally, I would like to thank my family for providing me an opportunity to pursue higher education and all my friends for their valuable support during my graduate study.

TABLE OF CONTENTS

	Page
DEDICATION	ii
ACKNOWLEDGMENTS	iii
LIST OF TABLES	vi
LIST OF FIGURES	vii
CHAPTER	
I. INTRODUCTION	1
1.1 Introduction	1
1.2 Hybrid Electric Vehicle	2
1.3 HEV Architectures	3
1.3.1 Series Hybrid Electric Vehicle	4
1.3.2 Parallel Hybrid Electric Vehicle	4
1.3.3 Series – Parallel Hybrid Electric Vehicle	5
1.3.4 Complex Hybrid Electric Vehicle	6
1.4 PSAT	6
1.5 Optimization	7
1.6 Thesis Scope and Organization	8
II. OPTIMIZATION	10
2.1 Background	10
2.2 Parametric Study	12
2.3 Optimization Algorithms	16
III. OPTIMIZATION ALGORITHMS	17
3.1 DIRECT	17
3.1.1 Algorithm Description	17
3.1.2 Flowchart	23
3.1.3 Advantages/Disadvantages	24
3.2 Simulated Annealing	25
3.2.1 Algorithm Description	25
3.2.2 Tunable Parameters	27

CHAPTER Page

3.2.3 Flowchart 27
3.2.4 Advantages/Disadvantages 30
3.3 Genetic Algorithm........ 30
3.3.1 Flowchart 31
3.3.2 Operators and Selection Method 32
3.3.3 Tunable Parameters 33
3.3.4 Advantages/Disadvantages 33
3.4 Particle Swarm Optimization 34
3.4.1 Algorithm description........ 34
3.4.2 Flowchart 36
3.4.3 Advantages/Disadvantages 38
3.5 Summary of Global Algorithms........ 38

IV. SIMULATION AND ANALYSIS 41

4.1 Problem Statement 41
4.2 Running a Simulation........ 45
4.3 HEV Optimization Results........ 47

V. CONCLUSIONS AND FUTURE WORK 54

5.1 Conclusions 54
5.2 Future Work 55

REFERENCES 58

APPENDIX

A. HYBRID ALGORITHM 61

LIST OF TABLES

TABLE		Page
2.1	Design variables for parametric study	12
3.1	Summary of global algorithms	40
4.1	Selected Parallel HEV configuration	44
4.2	Lower and upper bounds of design variables	44
4.3	Initial design variable values	48
4.4	Initial vehicle performance	48
4.5	Comparison of fuel economy	49
4.6	Design variables final values	50
4.7	Comparison of the vehicle performance	51
4.8	Comparison of the mass of the vehicle	51
A.1	Hybrid Algorithm on Rosenbrook's banana function	66
A.2	Hybrid algorithm on camel function	66

LIST OF FIGURES

FIGURE		Page
1.1	HEV Configurations	3
1.2	PSAT Graphical User Interface	7
2.1	Optimization process	11
2.2	Fuel Economy dependence on fuel converter power rating	13
2.3	Fuel Economy dependence on the motor controller power rating	14
2.4	Fuel Economy dependence on battery number cells	15
2.5	Fuel economy dependence on final drive ratio	15
3.1	Lower bound of a function $f(x)$ using Lipschitz Constant	18
3.2	Rectangle selection using all possible K	20
3.3	First three iterations of a two dimensional problem	21
3.4	Selection of optimal rectangles in each iteration	22
3.5	Flowchart showing the DIRECT algorithm	23
3.6	Flowchart showing the Simulated Annealing Algorithm	28
3.7	Flowchart showing the Genetic Algorithm	31
3.8	Updating velocity and position in PSO	35
3.9	Flowchart of Particle Swarm Optimization	37
4.1	FET -75 drive cycle	42
4.2	HWFET drive cycle	42

4.3 Configuration of the selected HEV 43

4.4 File interaction in HEV optimization 46

4.5 Algorithm performance comparison... 52

A.1 Flowchart showing the hybrid algorithm implementation 64

A.2 Two test functions: (a) Rosenbrook's banana (b) 3-hump Camel back 65

CHAPTER I

INTRODUCTION

1.1 Introduction

Conventional automobiles are the major source of energy consumption and airborne pollutants all over the world. The low efficiency of the conventional Internal Combustion Engine (ICE) vehicles results in lower fuel economy and high emissions [1]. To address the environmental concerns, California Air Regulatory Board (CARB) proposed regulations that would require a progressively increasing percentage of automobiles to be zero emissions vehicles beginning in the year 1998. Therefore the need for alternative vehicles improving the fuel economy and reducing emissions is growing. Electric Vehicle (EV) is the best possible solution for an efficient, environmental friendly and sustainable vehicle for urban transportation. The higher efficiency of Electric Vehicle automatically results in a higher fuel economy.

A Hybrid Electric Vehicle, powered by Internal Combustion Engine (ICE) and energy storage, is being given more attention because of the advantages associated with it. The inherent advantages of HEVs are their increased fuel economy, reduced harmful emissions and better vehicle performance. Various vehicle simulators are available for the analysis of Hybrid Electric Vehicles. Some of the most widely tools are Powertrain System Analysis Tool (PSAT), Advanced Vehicle Simulator (ADVISOR), and Versatile Electrically Peaking Hybrid Vehicle (V-Elph) tool etc. The fuel economy and the performance of a hybrid vehicle greatly depend on the component selection. In the modeling process, a component is modeled using many design variables. An optimal selection of the design variables is required for a better performance of the hybrid powertrain. This is achieved by Design Optimization. Design Optimization searches for various combinations of design variables and selects the components for greatest improvement. Since this design problem is a multi-modal, noisy and discontinuous problem, Global optimization algorithms best fits this problem.

1.2 Hybrid Electric Vehicle

Electric vehicle (EV) is a road vehicle which involves electric propulsion. With this broad definition in mind, EVs can be classified depending on the source of the propulsion into Battery Electric Vehicles (BEV), Hybrid Electric Vehicles (HEV), and Fuel Cell Electric Vehicle (FCEV). The battery is the only source of propulsion in BEVs, supplying the needed electrical energy to the electrical components and accessories. The source of propulsion in a HEV comes from two or more sources, usually a combination of combustion engine and energy storage. Therefore, it combines the range advantage of a conventional vehicle with the environmental benefits of a pure electric vehicle. FCEV uses fuel cell as its source of energy. Fuel cell vehicles turn hydrogen fuel and oxygen into electricity. The electricity generated then powers an electric motor.

A hybrid vehicle is a vehicle with multiple distinct power sources that can be separately or simultaneously used to propel the vehicle. The energy can come from a number of different sources like batteries, gasoline, solar energy, fuel-cell, ultracapacitor, or flywheels. The common and the most promising hybrid vehicle today is the one using an internal combustion engine and a battery powering a motor. The electric motor is used to improve the efficiency and the vehicular emissions while the ICE provides extended range capability. Thus, HEV makes the best use of existing technology by providing the benefits of both electric and conventional vehicles, while minimizing the shortcomings of each. One of the interesting features of HEV is that it uses the regenerative braking to make sure that the decelerating kinetic energy is stored back to the onboard battery. In a conventional vehicle, this kinetic energy is lost as heat. The availability of two energy sources allows for different configurations, but generally, they can be classified into series, parallel, series-parallel, and complex hybrid systems. These HEV systems are discussed in section 1.3.

1.3 HEV Architectures

The HEV configurations are shown in Figure 1.1

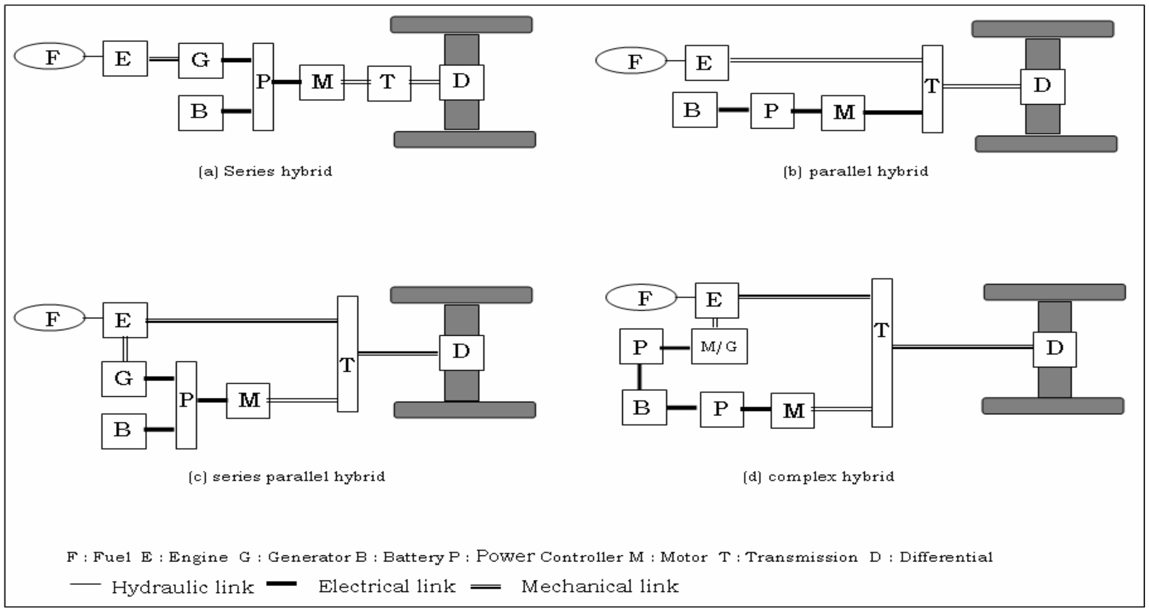

F
E
G
B
P
M
T
D
(a) Series hybrid
(b) parallel hybrid
(c) series parallel hybrid
M/G
(d) complex hybrid
F : Fuel E : Engine G : Generator B : Battery P : Power Controller M : Motor T : Transmission D : Differential
Hydraulic link
Electrical link
Mechanical link

Figure 1.1 HEV Configurations

1.3.1 Series Hybrid Electric Vehicle

In a series hybrid system shown in Figure 1.1a, the mechanical output from the ICE is converted to electrical energy using a generator and the electrical energy is either used to charge the battery or is bypassed from the battery to the motor that propels the wheels.

The advantage of a series configuration is that since the engine never idles, there are less emissions; therefore, it is better for the environment. Some other advantages of series hybrid are flexibility of location of engine-generator set, simplicity in design and stability for short trips. The disadvantage of series hybrids is that it needs three propulsion components: ICE, generator and motor. Therefore, the efficiency of series hybrid is generally lower. The motor must be designed for the maximum sustained power that the vehicle may require, such as when climbing a high grade. However, the vehicle operates below the maximum power most of the time. All three drive train components need to be sized for maximum power for long-distance, sustained, and high speed driving. Otherwise, the batteries will exhaust fairly quickly, leaving ICE to supply all the power through the generator [2].

1.3.2 Parallel Hybrid Electric Vehicle

In a parallel configuration shown in Figure 1.1b, there is a direct mechanical connection from both the electric power unit and the gasoline engine to the wheels. Thus, the propulsion may be supplied by the ICE alone, or by the electric motor alone, or both. The power delivered to the wheels is determined by the control strategy. In case of a high power demand such as for high acceleration, both the ICE and electric motor deliver power to the wheels. In less demanding situations, the ICE can be operated in the efficient mode to deliver a higher power than the power required at the wheels; the excess power is stored in the batteries for later use. The other option will be to operate the electric motor alone to drive the vehicle. This has the advantage of operating the ICE in a more efficient mode or shutting off during low efficiency operation. During long and steady state cruises, the ICE engine can alone drive the wheels avoiding the inherent inefficiency of the battery. The electric motor can be used as a generator to charge the battery by regenerative braking or absorbing the power from the ICE when the output power from the ICE is greater than the power required at the wheels.

The main advantage of parallel HEVs is improved dynamic performance due to the direct coupling between the ICE, electric motor, and the wheels. The disadvantage with the ICE being directly coupled to the wheels is that there is more transient speed operation than in a series vehicle. This tends to result in poorer efficiency and increased emissions.

1.3.3 Series – Parallel Hybrid Electric Vehicle

In these systems as shown in Figure 1.1c, the ICE is also used to charge the battery. This makes sure that the battery is properly charged even in the long drive cycles. Although possessing the advantageous features of both series and parallel HEVs, the series-parallel HEVs are relatively more complicated and costly. Nevertheless, with the advances in control and manufacturing technologies, some modern HEVs like Toyota Prius adopt this configuration [3].

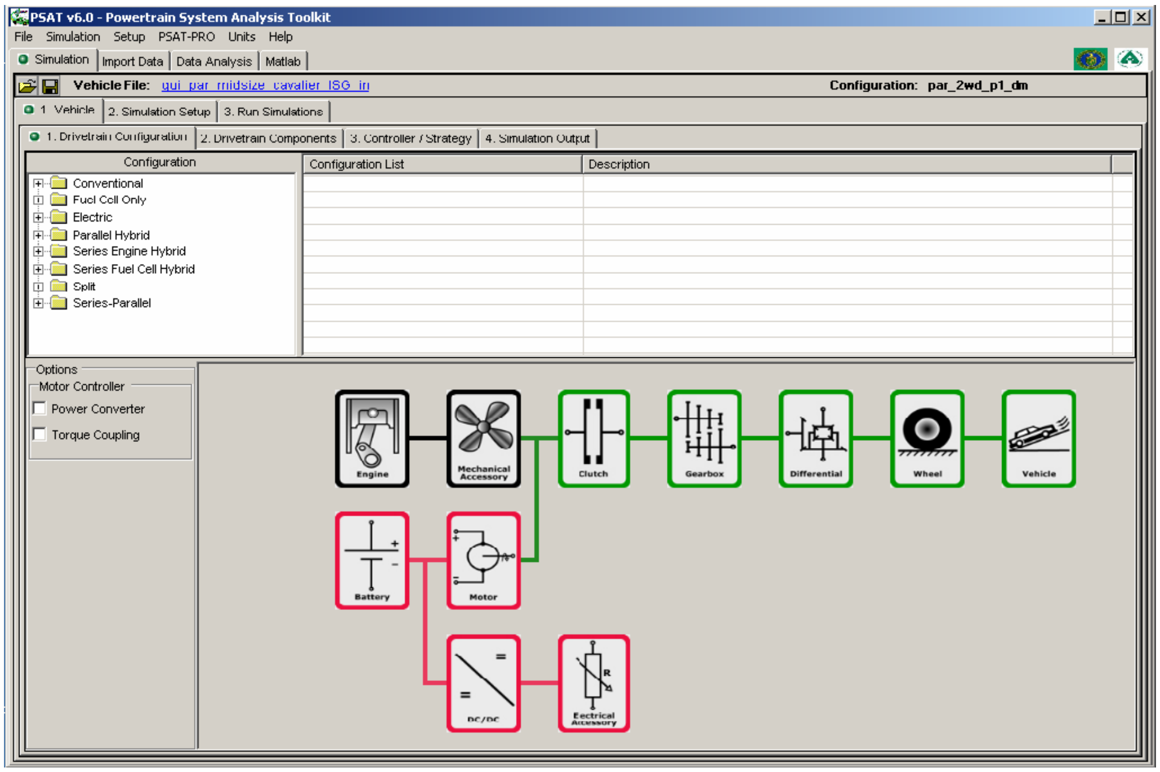
1.3.4 Complex Hybrid Electric Vehicle

The Complex hybrid systems shown in Figure 1.1d involve a complex configuration. It has all the features of the above discussed hybrid systems and a unique feature of bidirectional power flow of the electric motor. This bidirectional power flow can allow for versatile operating modes, especially for the system involving the ICE and two motors. The inherent advantage of complex hybrid systems is the configurability and the main disadvantage is the complexity and cost [4].

1.4 PSAT

The Powertrain System Analysis Toolkit (PSAT) is a state-of-the-art flexible simulation package developed by Argonne National Laboratory and sponsored by the US Department of Energy (DOE). PSAT was developed in MATLAB/Simulink environment and is set up with a graphical user interface (GUI), which makes it user friendly and easy to learn. A screen capture of the PSAT GUI is shown in Figure 1.2. PSAT is a forward looking model and allows users to simulate more than 200 predefined configurations, including conventional, electric, fuel cell, and hybrids (parallel, series, power split, series-parallel). The large library of component models and data allows users to simulate light, medium, and heavy-duty vehicles [5].

The presence of quasi-steady models and control strategies in PSAT sets it apart from other steady state simulation tools like ADVISOR. This feature makes it predict fuel economy and performance of a vehicle more accurately. PSAT is designed to co-simulate with other environments and is capable of running optimization routines. Hardware-in-the-loop (HIL) testing is made possible in PSAT with the help of PSAT-PRO, a control code to support the component and vehicle control. The main drawbacks of PSAT are it does not support any component calibration and runs with a too large sampling time [6].



Vehicle File: gui_par_midsize_cavalier_ISG_in

Configuration: par_2wd_p1_dm

Figure 1.2 PSAT Graphical User Interface

1.5 Optimization

Hybrid Electric Vehicles are recently given more attention because of their ability to increase the fuel economy and performance while reducing the emissions. The extent of improvement greatly depends on selection of each component and control strategy parameters. The complex interaction between the various components makes it difficult to size specific components manually or analytically. So, simulation-based optimization is a possible solution for such kind of system level design problems. Optimization tries to minimize or maximize an objective function by searching the multi-dimensional design space for the various combinations of component sizes (design variables) and selecting the best combination at each iteration. In other words, it eliminates the bad designs while keeping the good designs.

The choice of optimization algorithm is also an important issue. Local algorithms use derivative information to find the local minima and they do not search the entire design space. On the other hand, global algorithms search the entire design space and find the global optimum. Also, global algorithms do not require the derivative information. With this in mind and also taking in to consideration the noisy, discontinuous, and multi-modal nature of the design problem, derivative-free global algorithms best suits this design problem. Various optimization algorithms already exist to solve such complex design problems. Global algorithms like DIRECT, Simulated Annealing (SA), Genetic Algorithm (GA), and Particle Swarm Optimization (PSO) etc., aptly suit this problem. These optimization algorithms are looped with the simulation tool which feedbacks the objective value and the vehicle performance values. The optimization searches the multi-dimensional design and reaches a better design point using some heuristic (deterministic) or random (stochastic) way depending upon the algorithm used.

1.6 Thesis Scope and Organization

Optimization of a Parallel Hybrid Electric Vehicle is the primary focus of this thesis. Powertrain System Analysis Tool is used as the vehicle simulator. A parametric study is done to see the effect of each component size on the final objective. Local optimization routines suffer in finding the global optimum and miserably fail when faced with discontinuous responses. In this thesis, a Parallel Hybrid Vehicle is optimized using various global algorithms – DIRECT (DIviding RECTangles), Simulated Annealing (SA), Genetic Algorithm (GA), and Particle Swarm Optimization (PSO). Further, a Hybrid Algorithm, which is a combination of global and local algorithm, is used to optimize two simple test functions. DIRECT is a deterministic algorithm whereas the other three are stochastic methods. Deterministic and Stochastic algorithms are taken intentionally to see the performance of different kinds of algorithms on the design problem. A comparison of the four optimization algorithms is done based on the improvement in the fuel economy. These optimization routines can be embedded in to the simulation tools for a better performance. Global algorithms can be coupled with local algorithms for better convergence.

The organization of this thesis is as follows. Chapter 2 gives the background information related to optimization. The optimization process is explained and a brief description about the local and global based algorithms is presented. A parametric study is provided to see the effect of various component sizes on the fuel economy. Chapter 3 explains in detail about the algorithms used in this study – DIRECT, SA, GA, and PSO. Each algorithm is provided with a flowchart for easy understanding. The advantages and disadvantages of each algorithm are also provided. Chapter 5 is the main section of this thesis, stating the design problem and the results associated with it. Chapter 6 deals with the conclusions of this study and the future work.

CHAPTER II

OPTIMIZATION

2.1 Background

Optimization is the process of minimizing or maximizing an objective function while satisfying the prevailing constraints [7]. A general non-linear constrained optimization problem is illustrated in the following example:

$$\begin{aligned}
&\min \quad f(x) = weight \quad of \quad the \quad vehicle \\
&w.r.t \quad x = \{engine \quad size, motor \quad size\} \\
&s.t. \quad 0-60\,mph \le 12\,s \\
&\qquad 40-60\,mph \le 5.3\,s \\
&\qquad 0-85\,mph \le 23\,s \\
&\qquad \max \quad speed \ge 85\,mph
\end{aligned}$$

In the above design problem, the objective is the minimization of the vehicle weight, design variables are the engine and motor sizes, and the constraints are the performance limits on the vehicle. Here, the engine and motor size doesn't mean the actual physical sizes or dimensions of the components but instead relate to the power ratings of the respective components. The optimization algorithm searches the design space defined by the bounds on the design variables and identifies a point that minimizes the objective function while satisfying the constraints. The objective function is evaluated by the vehicle analysis/simulation tool. Various computer programs like SIMPLEV [8], MARVEL[9], V-Elph [10], Carsim 2.5.4 [11], ADVISOR [12], PSAT, etc. are available for the analysis of the hybrid vehicles. An illustration of the optimization process is given in Figure 2.1.

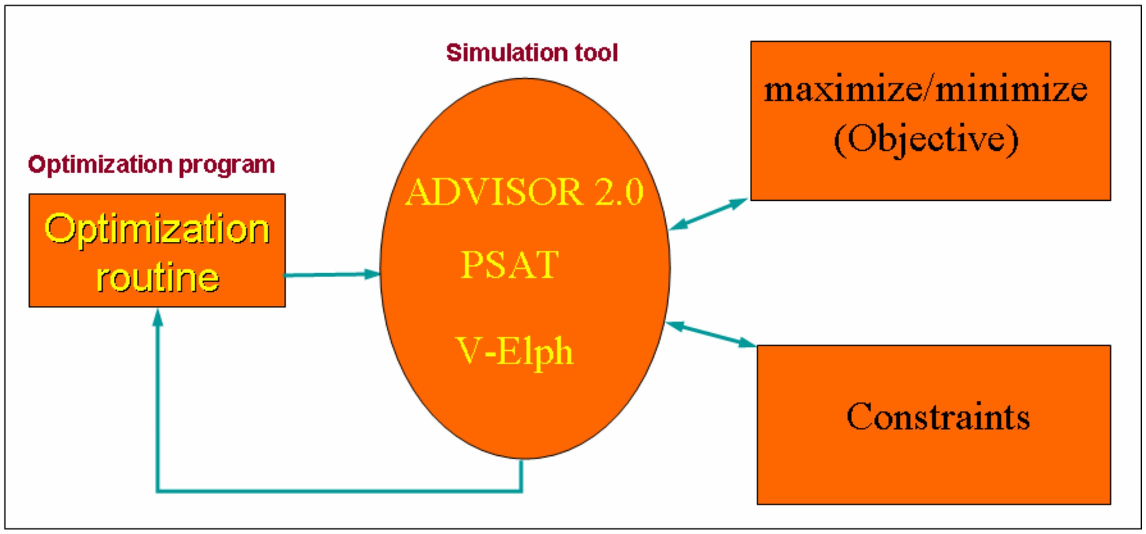


Figure 2.1 Optimization process

It can be seen from Figure 2.1 that the vehicle simulator is looped with the optimization program. The optimization process solves the design problem by iteratively calling the optimization routine to modify the design variables and then calling the vehicle simulator to calculate the response. This process is continued until the number of iterations gets exhausted or a predefined accuracy is obtained.

A hybrid electric vehicle has the potential to improve the fuel economy compared to conventional vehicles without sacrificing performance, but the extent of fuel savings is highly dependent on component optimization. The components include energy sources, transmission, control strategy, vehicle body, etc. The variables associated with these components are taken as design variables. For a Hybrid Electric Vehicle design there would be hundreds of design variables related to energy sources, control strategy, transmission, vehicle, etc. Optimizing all the design variables requires very high computation time and cost. Thus, in this thesis, limited number of design variables is considered.

2.2 Parametric Study

The effect of various design variables on the objective function is studied here. The fuel economy is taken as the objective function and its variation is studied subject to changes in design variables related to hybrid vehicle component sizes. The design variables include the power ratings of the fuel converter (ICE), motor controller, number of battery cells, and the final drive ratio. The description of the design variables is given in Table 2.1. The vehicle model gui_par_midsize_cavalier_ISG_in.m (available in the PSAT model library) is taken for the parametric study. The study is basically limited to four design variables and can be extended to more number of variables.

Table 2.1 Design variables for parametric study

Variable	Description
eng.pwr_max_des	Fuel Converter (ICE) power rating
mc.pwr_max_des	Motor Controller power rating
ess.init.num_module	Battery number of cells
fd.init.ratio	Final drive ratio

The selected HEV is simulated on a composite of city and highway driving cycles in PSAT to get the fuel economy. Note here the best values of the design variables obtained from this study can only maximize the fuel economy but may not improve the vehicle performance.

Figure 2.2 shows the plot for the fuel economy obtained for different power rating values of fuel converter. The size of the fuel converter is varied between 60 kW to 100 kW. This plot indicates that the fuel economy is higher for fuel converter power ratings between 60 kW and 70 kW. This is because of the lesser weight of the fuel converter at these power ratings. The maximum fuel economy is achieved at a fuel converter power rating of 66 kW. After 70 kW there is a sharp decline in the fuel economy because of the additional weight of the fuel converter to the vehicle weight.

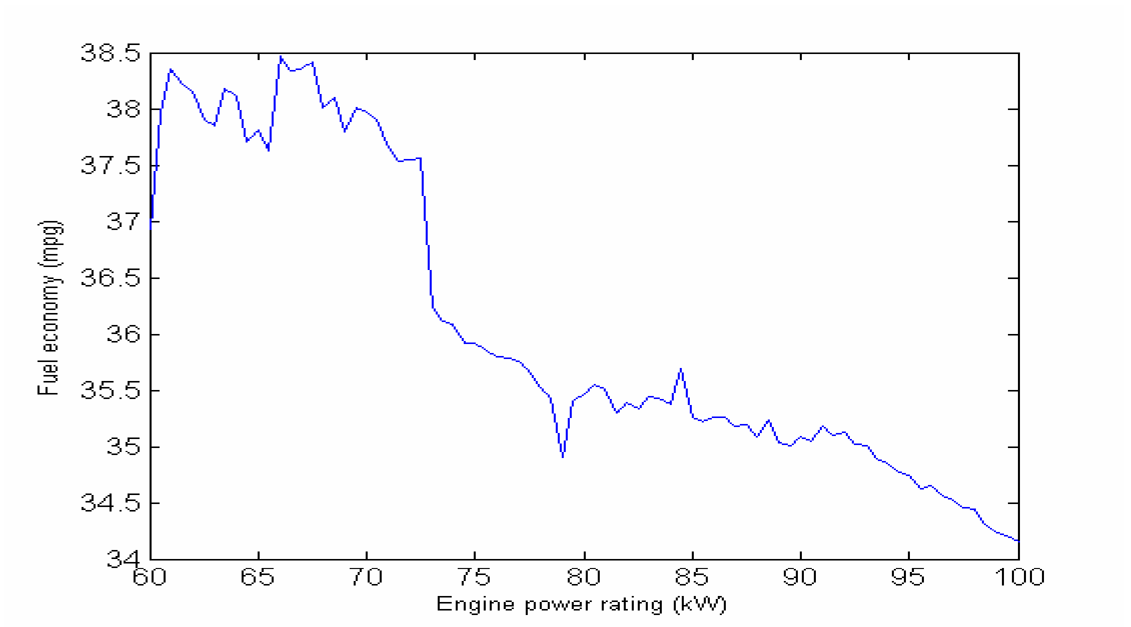


Figure 2.2 Fuel Economy dependence on fuel converter power rating

Figure 2.3 shows the dependence of the fuel economy with respect to motor power rating. The power rating of the motor converter is varied between 10 kW to 80 kW. The fuel economy reached a maximum value at a motor power rating of 20 kW and decreased for higher motor power rating values. Even in this case, the motor power rating of 20 kW is only designed to achieve maximum fuel economy.

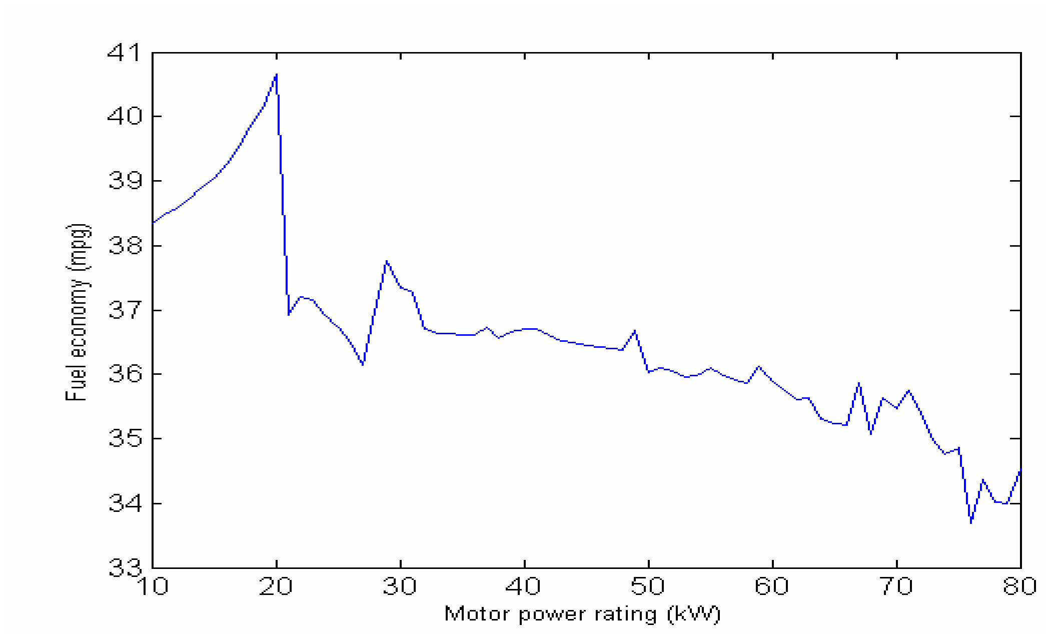


Figure 2.3 Fuel Economy dependence on the motor controller power rating

The dependence of the composite fuel economy on the number of battery cells is shown in Figure 2.4. The fuel economy is higher for battery cells between 260 and 270. Similarly, Figure 2.5 shows the dependence of the fuel economy on the final drive ratio. The fuel economy reached the optimal value for a final drive ratio of 3.9.

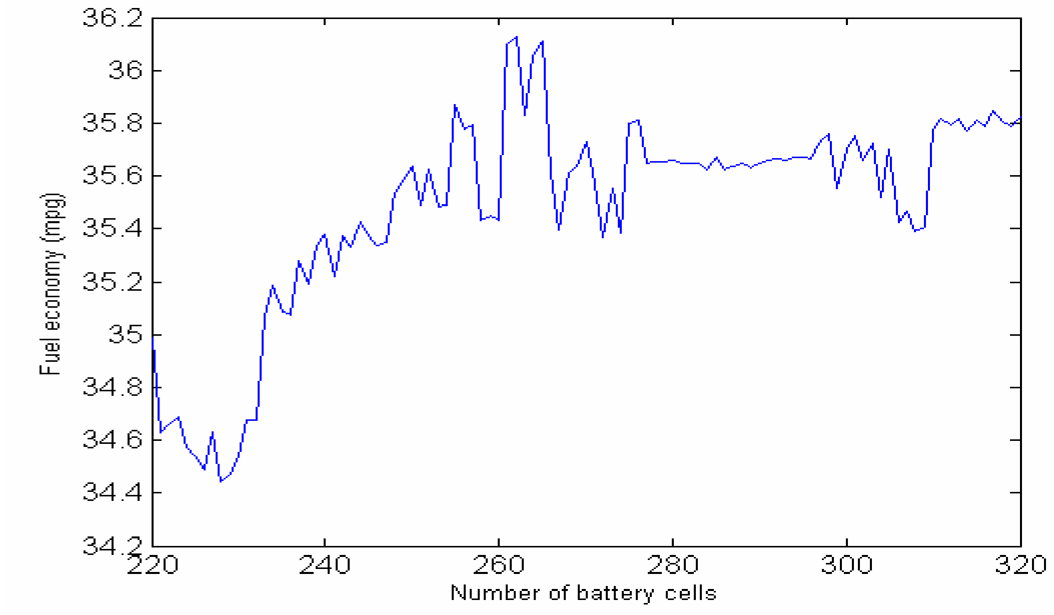


Figure 2.4 Fuel Economy dependence on battery number cells

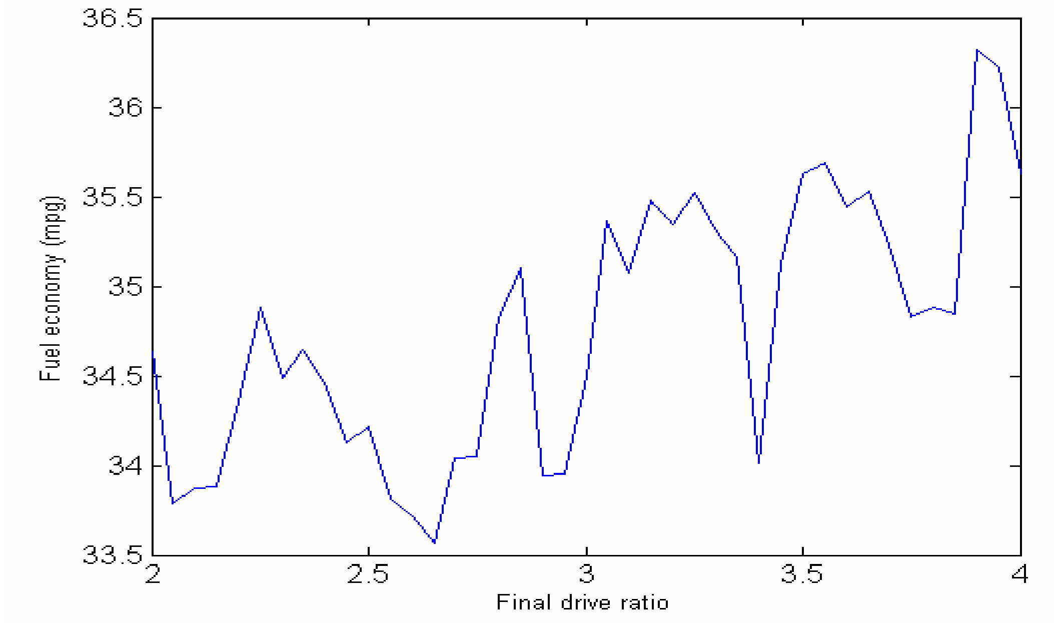


Figure 2.5 Fuel economy dependence on final drive ratio

2.3 Optimization Algorithms

Optimization methods can be divided into derivative and non-derivative methods. Gradient based algorithms like Sequential Quadratic Programming (SQP) [13] are good at finding local minima. SQP uses the derivatives of the objective function to find the path of greatest improvement to quickly find a minimum. Gradient based methods works fine for smooth, continuous functions but often fails miserably when faced with noisy and discontinuous functions. The major disadvantage of local optimizers is that they do not search the entire design space and so cannot find the global minimum.

This thesis focuses on non-derivative methods such as DIRECT, SA, GA, and PSO. Non-gradient methods are more robust in locating the global optima and are applicable in a broader set of problem areas. Another advantage of these methods is that they do not require any derivatives of the objective function in order to find the optimum. Hence, they are also known as blackbox methods. Here the objective function values are the results of complex computer simulations.

However the disadvantages are that we cannot prove that we have found the actual optima. This also applies partly to gradient methods as they might get caught in local optima. By conducting several optimizations with different initial conditions, it could be argued that the global optimum is truly found. Another disadvantage with non-gradient methods is that they usually require more function evaluations than gradient methods, and are thus more computationally expensive. However, as the computing power of the computers are increasing this disadvantage is diminishing. Furthermore, most non-gradient methods are well suited for implementation on parallel processors.

CHAPTER III

OPTIMIZATION ALGORITHMS

3.1 DIRECT

DIRECT is a global optimization algorithm developed by Donald R. Jones [14]. This algorithm is a modification of the standard Lipschitzian approach that eliminates the need to specify the Lipschitz constant [15]. Lipschitz constant is a weighing parameter, which decides the emphasis on the global and the local search. The use of Lipschitz constant is eliminated in [14] by searching all possible values for the Lipchitz constant, thus putting a balanced emphasis on both the global and local search.

3.1.1 Algorithm Description

DIRECT is a modification of a one dimensional Lipschitzian algorithm by Shubert [16] and extending it to multi-dimensional problem. The Lipschitzian approach by Shubert is given as follows:

A function $f(x)$ defined in the closed interval $[l,u]$ is said to have a lower bound such that there exists a positive Lipschitz constant K and satisfies the following condition:

$$\left|f(x)-f\left(x'\right)\right| \leq K\left|x-x'\right| \quad for \quad all \quad x,x' \in [a,b] \tag{1}$$

The above equation states that the rate of change of the function $f(x)$ should be less than the change between x and x' multiplied by K. If we substitute a for x' and b for x' in equation (1), we see that $f(x)$ must satisfy the following inequalities:

$$f(x) \leq f(a) - K(x-a) \tag{2}$$

$$f(x) \leq f(b) - K(x-b) \tag{3}$$

These equations represent two lines with slopes $-K$ and $+K$ as shown in Figure 3.1. These two lines forms a V shape and the lowest value of $f(x)$ can attain at the bottom of the V. This minimum point is denoted by $X(a,b,f,K)$ and the corresponding lower bound of f is denoted by $f_{\min}(a,b,f,K)$

$$X(a,b,f,K) = (a+b)/2 + [f(a) - f(b)]/2K \tag{4}$$

$$f_{\min}(a,b,f,K) = [f(a) + f(b)]/2 - K(b-a) \tag{5}$$

where $f(a)$ and $f(b)$ are the function values at a and b.

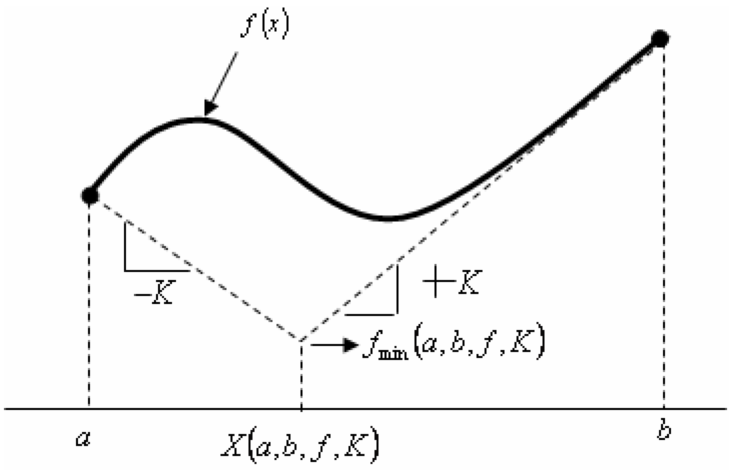


Figure 3.1 Lower bound of a function $f(x)$ using Lipschitz Constant

This minimum point is taken as x_1. This new point divides the search space in to two intervals. Then one of the two intervals with the least $f_{\min}$ is selected for division. This division is continued until some prespecified tolerance of the final solution is met. Equation (5) explains the local and global search of the algorithm. The first term leads to the local search and the second term leads to the global search. And the Lipschitz constant serves as a relative weight between local and global search. The larger the value of K, the higher the emphasis on the global search.

The Lipschitzian approach followed by Shubert has two main disadvantages:

1) Need to specify the correct value of Lipschitz constant K

2) Need 2^n function evaluations for *n*-dimensional design space.

These problems are fixed in the DIRECT algorithm proposed by D. Jones in which the sampling is done at the center point of the interval rather than at the endpoints to reduce the number of function evaluations. The balance between the local and global search in the DIRECT algorithm is made by selecting the optimal intervals (optimal rectangles) assuming all possible values for the Lipschitz constant. For example, assume that Figure 3.1 is divided in to 10 intervals (10 center points) and the function values at the center points of the intervals are evaluated. A plot showing these 10 points with the width of the interval on the x-axis and the corresponding function value is shown in Figure 3.2. If a line with a slope given by Lipschitz constant K is drawn from a point, then the y-intercept is the local bound for the function. Instead of fixing one value of K, various possible values of K are taken. This gives the lowest lower bound intervals represented by the lower convex hull of points shown in Figure 3.2. The same procedure will be followed to select the optimal rectangles in DIRECT algorithm.

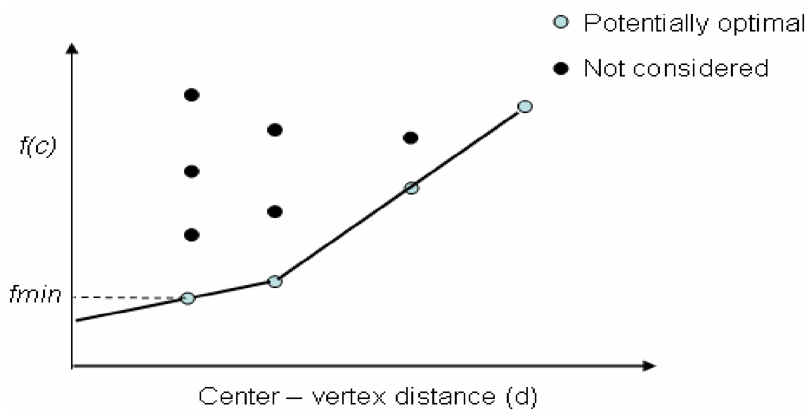
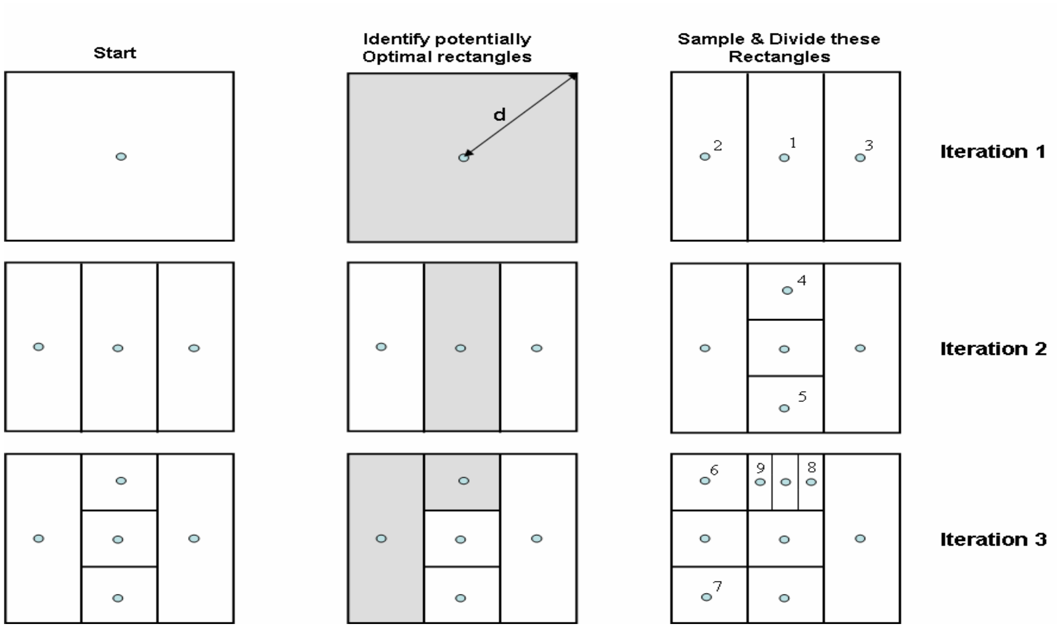


Figure 3.2 Rectangle selection using all possible K

DIRECT algorithm is based on the above theoretical background. A brief introduction of the DIRECT algorithm is presented here. A detailed explanation of the DIRECT algorithm can be found in [14].

DIRECT algorithm is basically a sampling algorithm. The algorithm begins by scaling the design box to an n dimensional unit hypercube for a n dimensional design problem. DIRECT initiates its search by evaluating the objective function at the center point of the hypercube. DIRECT then trisects this hyperrectangle and samples the center points of the two resulting hyperrectangles. From here, DIRECT selects the optimal hyperrectangles using various values of Lipschitz constant and trisects them. An example selection of the optimal hyperrectangles is shown by the lower convex hull of dots in Figure 3.2. The selection of optimal hyperrectangles selects the larger rectangles (global search) as well as smaller rectangles (local search). This division process continues until pre-specified function evaluations are reached or convergence is achieved. The division of rectangles in first three iterations of a two dimensional problem is illustrated in Figure 3.3. In this figure *d* represents the center to vertex distance and the numerals located above the center points represent the label for the corresponding rectangles.



Start
Identify potentially Optimal rectangles
Sample & Divide these Rectangles
d
2
1
3
Iteration 1
4
5
Iteration 2
6
9
8
7
Iteration 3

Figure 3.3 First three iterations of a two dimensional problem

In the first iteration, the unit hypercube is trisected into three rectangles. The objective function value is evaluated at the center points of the three resulted rectangles. The objective function values are plotted against the center – vertex distance as shown in Figure 3.4(a). Then the rectangle with least objective value in each column of dots is selected as the optimal rectangle. In the first iteration there is only one column of dots; therefore rectangle 1 is selected as the optimal rectangle and trisected in the second iteration. Similarly in the second iteration, rectangle 4 and rectangle 2 are the least objective valued rectangles as shown in Figure 3.4(b). These two rectangles are selected as optimal rectangles and trisected in the third iteration. This process is continued until the maximum number of objective function evaluations are exhausted.

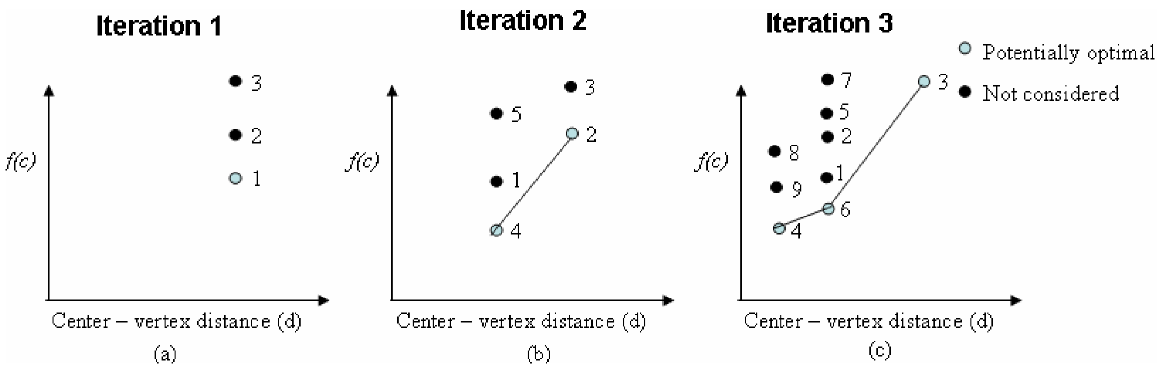


Figure 3.4 Selection of optimal rectangles in each iteration

The inequality constraints are handled by an auxiliary function given in [14] that combines the information of the objective and constraint functions. The auxiliary equation is given in equation (6) and is a weighed sum of the violated constraints and the deviation of the objective function value from a projected global minimum.

$$\max\left(f_r - f^*, 0\right) + \sum_{j=1}^{m} c_j \max\left(g_{rj}, 0\right) \tag{6}$$

In the above equation, f_r is the objective function value at the center point of the rectangle r, f^* is the assumed global minimum, m is the number of inequality constraints, c_j are the positive weighing coefficients, and g_{rj} is the constraint violation of the j^{th} constraint at the midpoint of rectangle r.

3.1.2 Flowchart

A flowchart explaining the DIRECT algorithm is shown in Figure 3.5.

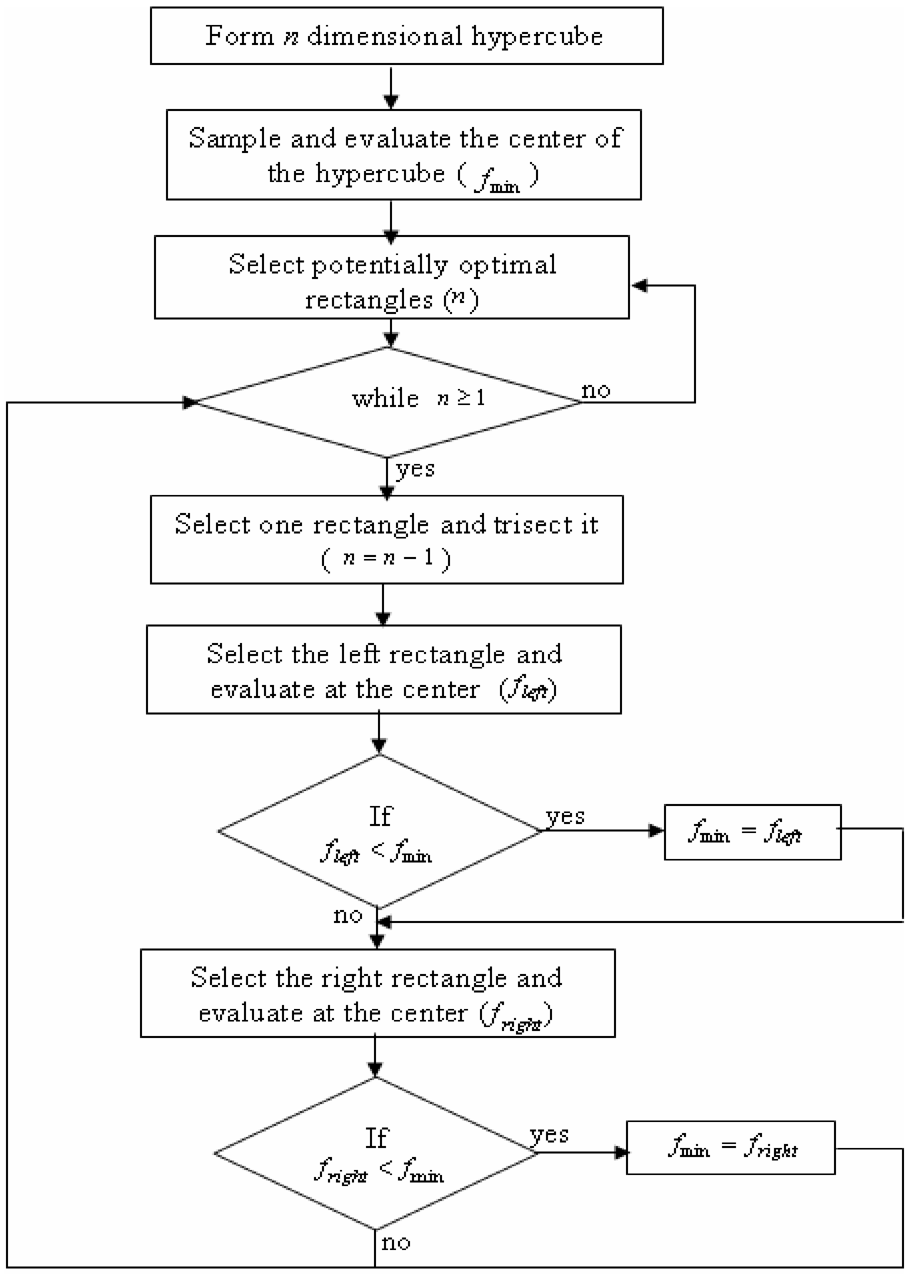

Form n dimensional hypercube
Sample and evaluate the center of the hypercube ($f_{\min}$)
Select potentially optimal rectangles (n)
while $n \geq 1$
no
yes
Select one rectangle and trisect it ($n = n - 1$)
Select the left rectangle and evaluate at the center (f_{left})
If $f_{left} < f_{\min}$
yes
$f_{\min} = f_{left}$
no
Select the right rectangle and evaluate at the center (f_{right})
If $f_{right} < f_{\min}$
yes
$f_{\min} = f_{right}$
no

Figure 3.5 Flowchart showing the DIRECT algorithm

Initially, DIRECT converts the n dimensional design space in to a n dimensional unit hypercube. It samples and evaluates at the center point of this rectangle. This function value is assigned to $f_{\min}$ which holds the minimum function objective value. Then a set of optimal rectangles is selected assuming various possible values for the Lipschitz constant. These rectangles form the lower convex hull of dots as shown in Figure 3.2. However, in the first iteration, the only present rectangle is selected as the optimal rectangle. Each rectangle in the optimal rectangle set is trisected to give two more rectangles (left and right rectangles). Objective function is evaluated at the center points of the left and right rectangles and $f_{\min}$ is updated if there is an improvement in the objective function. The whole process is continued until a pre-specified number of function evaluations.

3.1.3 Advantages/Disadvantages

DIRECT algorithm has no tuning parameters in order to get good algorithm performance. Also, the user is not required to specify the starting point since DIRECT starts at the center point of the design space as its initial point. Therefore, it eliminates the problem of choosing a good starting point. Note here that the local optimizers require a good initial point to reach the optimum value. Another advantage is that it covers the entire design space, avoiding any chance of missing the global optimum.

DIRECT converges to the global optimal region with few function evaluations but needs more number of function evaluations to actually reach the global optimum.

3.2 Simulated Annealing

Simulated Annealing comes under the class of stochastic algorithms, which means that they follow a random path in every searching process for global optimum. Simulated Annealing has been presented by large number of authors, but Kirkpatrick *et* al. [17] started using the Simulated Annealing method in various combinatorial problems. Simulated Annealing is based on the Metropolis Monte Carlo Simulation proposed in [18].

3.2.1 Algorithm Description

Simulated Annealing algorithm is an analogy between the annealing process occurring in metals and the function minimization. This analogy is explained briefly here. When metals are at high temperatures, the atoms can move relatively freely in the higher energy states; but as the temperature is decreased slowly, the atoms can move freely enough and begin adopting the most stable orientation by taking the lowest possible energy state. If the temperature is decreased rapidly, the atoms become frozen at a high energy state. Attaining the lowest possible state can be thought as reaching the global minimum in the optimization process. The temperature is decreased slowly (cooling) so that the design will find the global minimum (lowest energy state).

The algorithm starts by evaluating the objective function at a random design point. From this design point, the algorithm jumps to new random design point and evaluates the objective function value and feasibility. If the current point is better than the previous, then the current point is accepted to be optimal point; if the current point is worse than the previous point then its acceptance or rejection depends on the Metropolis probability criterion given below:

$$P(f,T) = e^{[\frac{f_{new} - f_{current}}{T}]} \tag{7}$$

From the above equation, it can be seen that the new point is more likely to be accepted, if the new design point function value is close to the current design point function value. And also, the probability of accepting a design point is more when the temperature is high. Note here that the algorithm may accept a new design point even when it is worse (has a higher function value) than the current one. It is this feature that prevents the method from getting stuck in a local minimum. So, initially when the temperature is high, the simulating algorithm does a global search where even worse design points are more likely to be accepted and switches to local search when the temperature is decreased where worse design points are less likely to be selected. Thus, the switching from the global search to local search depends on the value of the temperature. Another parameter which is responsible for the switching from the global to local search is the maximum step size. This variable is reduced as the algorithm progresses forcing the algorithm to search more locally. The algorithm is terminated when the temperature reduction cycles reach the specified number or until the number of function evaluations are exhausted.

3.2.2 Tunable Parameters

Simulated Annealing has many parameters that needs to be tuned to improve the efficiency of the algorithm. So, particular attention should be given to these parameters. The parameters used in this algorithm are described below:

- Num_steps : Number of steps before reducing the temperature and maximum step size.
- T_0_init : Initial temperature.
- V_0_init : Initial step size (maximum).
- $Temp\_red$: Temperature reduction factor. The temperature for the next cycle is reduced by this factor.
- V_0_red : This factor is multiplied with the initial step size to get the step size for the next design point.

3.2.3 Flowchart

A flowchart showing the core of the Simulated Annealing is shown in Figure 3.6.

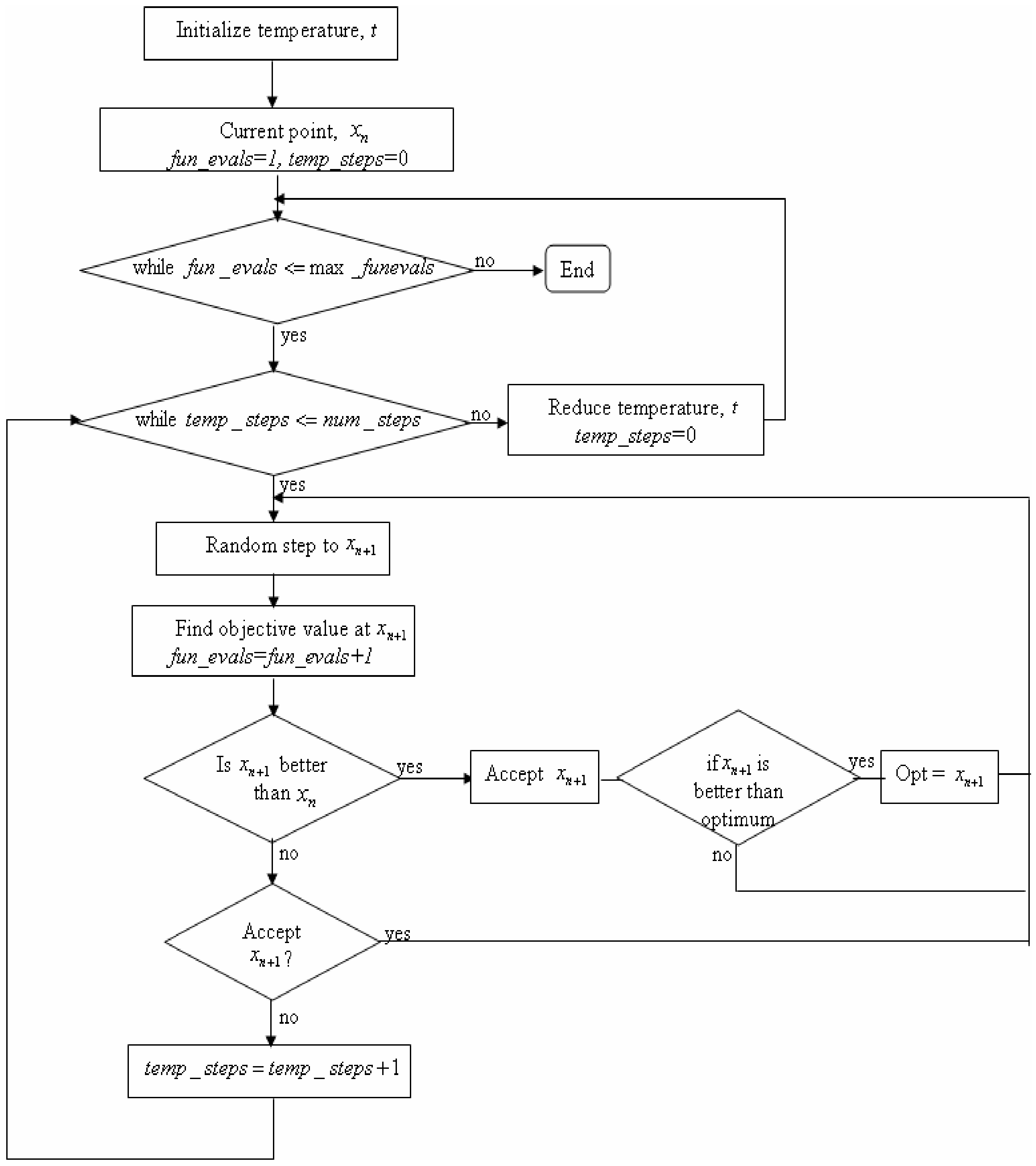

Initialize temperature, t
Current point, x_n
$fun\_evals=1$, $temp\_steps=0$
while $fun\_evals <= \max\_funevals$
no
End
yes
while $temp\_steps <= num\_steps$
no
Reduce temperature, t
$temp\_steps=0$
yes
Random step to x_{n+1}
Find objective value at x_{n+1}
$fun\_evals=fun\_evals+1$
Is x_{n+1} better than x_n
yes
Accept x_{n+1}
if x_{n+1} is better than optimum
yes
Opt = x_{n+1}
no
no
Accept x_{n+1}?
yes
no
$temp\_steps = temp\_steps + 1$

Figure 3.6 Flowchart showing the Simulated Annealing Algorithm

Simulated Annealing starts by initializing a temperature t. Next, a random design point x_n is selected such that it satisfies all the constraints (feasible point). This point is passed as the current point to the algorithm core. Simulated Annealing is carried out in two loops. The outer loop defines the number of function evaluations that the algorithm must run before terminating. This parameter is defined by *max_funevals* parameter. The inner loop defines the number of steps after which the temperature and the step size is reduced. This parameter is defined by *num_steps* parameter. As discussed above, this parameter is responsible for the switching from the global to local search. As shown in the flowchart, the feasible point found initially is made as the current point x_n. Then, it makes a random step to x_{n+1} and the objective function value and the constraints are calculated in the next step. The current step size is used in finding the new design point. In the next step, the function value and the constraints at this design point are evaluated. Then the new design point is compared with the old design point to see whether it is better or not. This comparison is done by penalty method. A quadratic penalty function given in equation (8) is used.

$$Penalty(x_i) = f(x_i) + \sum_{j=1}^{n} \left\{ f(x_i) \left[\frac{\max(0, g_j(x_i))}{boundingvalue_j} \right] \right\}^2 \tag{8}$$

This function gives a higher penalty to the design points which violates the constraints more. So, if the penalty of the current point x_{n+1} is less than the penalty for the design point x_n then it is accepted and tested if it is better than the current optimum. If it is better, then it is assigned as the current optimum and this point is fed back to generate a new design point. If the penalty of the current point x_{n+1} is more than the penalty for the design point x_n then the decision of its acceptance is taken using Metropolis criterion given in equation (7). If the current point is accepted then a new design point is generated based on design point x_{n+1}. If it is not accepted then a new design point is generated from the previous design point x_n.

3.2.4 Advantages/Disadvantages

The main advantage of SA is that it is a very efficient algorithm for finding the global minima. It accepts some worse points during the process in hope of finding global minima. Another advantage is that it doesn't cover the entire design space to find the global minima.

The disadvantage of the SA is the tuning parameters discussed above. Right set of parameters are needed to improve the efficiency of the algorithm. This tuning in turn becomes an optimization problem. SA is efficient in finding the region of global minima but it may take more number of function evaluations to find the true global minima.

3.3 Genetic Algorithm

Genetic algorithms [19, 20] are based on evolutionary processes and Darwin's concept of natural selection. In this selection, only the fittest populations survive while the bad populations are weeded out. During the process, several natural processes like crossover, mutation, and natural selection are used to select the bestfit population. The same concept is extended to the mathematical optimization problems where only good design points are selected while the bad design points are neglected. In this context, the objective function is usually referred to as a fitness function, and the process of "survival of the fittest" implies a maximization procedure.

3.3.1 Flowchart

The flowchart for the Genetic Algorithm is given in Figure 3.7.

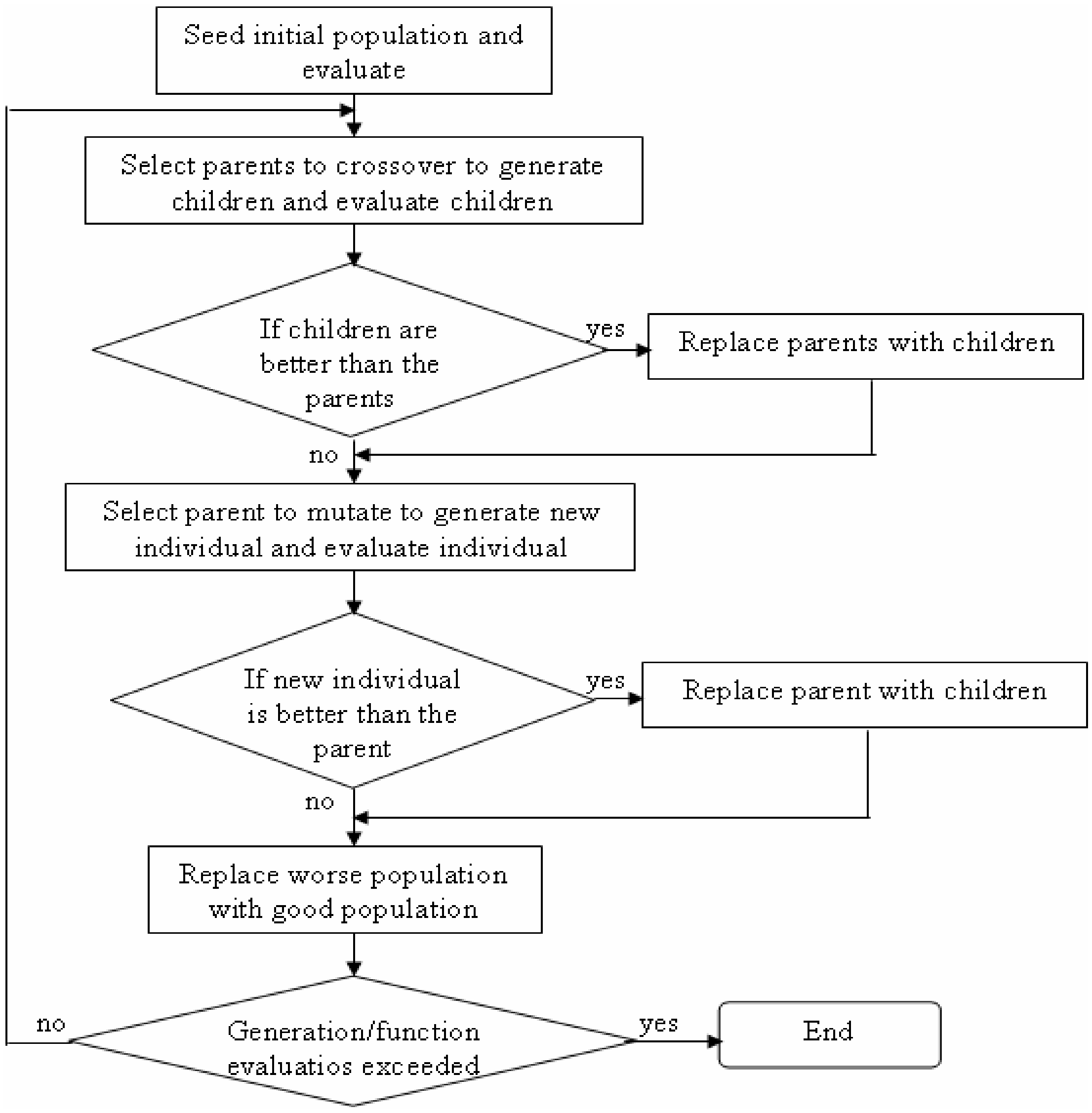

Seed initial population and evaluate
Select parents to crossover to generate children and evaluate children
If children are better than the parents
yes
Replace parents with children
no
Select parent to mutate to generate new individual and evaluate individual
If new individual is better than the parent
yes
Replace parent with children
no
Replace worse population with good population
no
Generation/function evaluatios exceeded
yes
End

Figure 3.7 Flowchart showing the Genetic Algorithm

Genetic Algorithm begins by randomly generating or seeding an initial population of candidate solutions (design points). Starting with the initial random population, GA then applies a sequence of operations like the design crossover where two individuals (parents) from the initial population are selected randomly and are reproduced to get two new individuals (children) and mutation where one individual from the initial population is slightly changed to get a new individual. If the newly generated individuals created by the crossover and the mutation operators are better than the parents used, then the parents are replaced by the newly created individuals. Again at the end, the worst design points are weeded out from the population in order to improve the fitness function. The above entire process can be termed as one generation and is continued for several generations or until the maximum number of function evaluations are exhausted in order to further improve the fitness function.

3.3.2 Operators and Selection Method

Arithmetic crossover operator is implemented in this study. Two parents reproduce to generate two new individuals (children). The parent individuals are selected randomly. The newly generated individuals can be represented as a linear combination of the parents as shown in equation (9)

$$\begin{aligned} X_{ind1} &= r.X_{par1} + (1-r).X_{par2} \\ X_{ind2} &= (1-r).X_{par1} + r.X_{par2} \end{aligned} \tag{9}$$

In equation (9), $X_{ind1,}$ X_{ind2} represent the two new created individuals, X_{par1}, X_{par2} represent the parents and r represents a random variable between 0 and 1.

In mutation a parent is selected and is altered to get a new individual. *Uniform mutation* is implemented in this study.

A selection method is needed to choose the bestfit individuals. A *normalized geometric* selection method which is a ranking type method is used as the selection method. The ranking method was chosen because of the presence of negative fitness. With this selection method, a probability is assigned to each individual of the population given by equation (10), where q is the probability of selecting the best design, r is the rank of the individual, and P is the population size.

$$\text{Each individual probability} = \frac{q}{1-(1-q)^P}(1-q)^{r-1} \qquad (10)$$

3.3.3 Tunable Parameters

Like Simulated Annealing, GA has many parameters that need to be tuned for a better performance although GA has less number of tunable parameters compared to SA. The tunable parameters and their description are given below:

- *Pop_size*: number of individuals in a generation.
- *xoverFNs*: number of times the crossover operation is to be done.
- *mutFNs*: number of times the mutation operation is to be done.

3.3.4 Advantages/Disadvantages

The initial population generated primarily determines a good starting point for GA. So, it can be seeded by some good design points to improve its efficiency. Different types of selection methods, crossover and mutation operators can be specified based on the design problem.

The major disadvantage is the tuning of the parameters. And also, the initial population is randomly chosen. This random initial population may not cover the entire design space uniformly. It was observed that most of the design points generated by the crossover and mutation operators were not able to meet the constraints (infeasible points) because the operators have no knowledge of the constraints. This resulted in many infeasible points. This was the reason why the algorithm had difficulty in improving the best objective function value found in the initial random population.

3.4 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is an evolution-based stochastic global optimization technique developed by Kennedy and Eberhart [21]. PSO is based on the Swarm Intelligence found in the natural systems. Such systems are typically made up of a population (*swarm*) of simple *agents or particles* interacting locally with one another and with their environment. Bird flocking, ant colonies, animal herding etc. are a few of the examples for such natural systems. In these systems, the local interactions between the agents such as changing the position and velocity lead to the global behavior. The same technique can be applied in the optimization problems to find global maxima.

3.4.1 Algorithm description

Just like GA, PSO is a population-based search procedure. It starts by initializing random solutions called *particles* in the multi-dimensional design space. In a PSO system, each particle flies in the multi-dimensional design space looking for the global maximum. Each particle in the PSO is defined by a point in the design space called *position* and its flight speed called *velocity.* And also, each particle is aware of its best position reached so far (*pbest*) and the best position of the group so far (*gbest*). During flight, each particle adjusts its position according to its own experience (*pbest* value), and according to the experience of its neighboring particles (*gbest* value). The position is modified using the concept of velocity. The velocity of each particle is updated as follows:

$$v_i^{n+1} = kv_i^n + \alpha_1 rand_1 \left(pbest_i - p_i^{\,n}\right) + \alpha_2 rand_2 \left(gbest - p_i^{\,n}\right) \tag{11}$$

where v_i^{n+1} is the velocity of the particle i at iteration $n+1$, k is the weighing function, α_1 and α_2 are the weighing factors, $rand_1$ and $rand_2$ are two random numbers between 0 and 1, p_i^n is the position of the particle i at iteration n, $pbest_i$ is the best position of the particle i, $gbest$ is the best position of the group (best of all $pbests$). In this study k=0.6 and α_1=α_2=1.7 are taken for better convergence [22]. Similarly, the position is updated as follows:

$$p_i^{n+1} = p_i^n + v_i^{n+1} \tag{12}$$

The velocity and position updating for particle i is illustrated in Figure 3.8. Note here that Figure 3.8 represents a two dimensional problem.

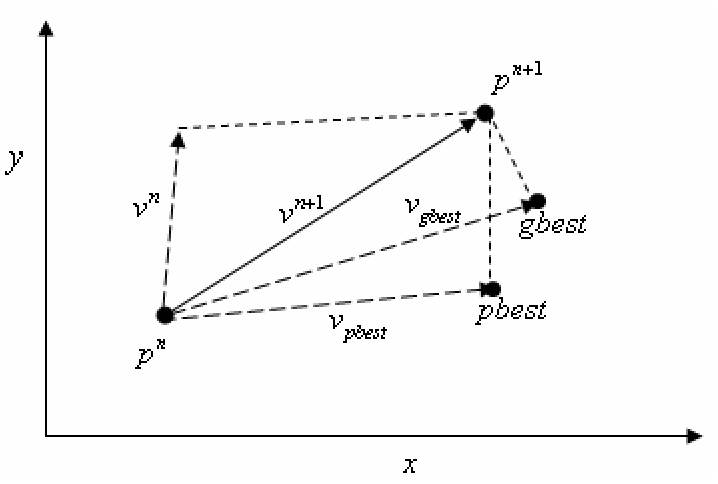


Figure 3.8 Updating velocity and position in PSO

From Figure 3.8, it can be seen that the position of the particle is adjusting itself towards the *gbest* position. This is because the velocity has changed its direction towards the *pbest* and *gbest* values.

The constraints in PSO are treated the same way as in SA and GA. A penalty is assigned to each design point using equation (8). The penalty is used to update the pbest and gbest for each particle. For a particle i, *pbest* value is updated if the penalty of the particle is less than the previous best penalty. And the same is done when the *gbest* is updated. This makes sure that the objective function is maximized.

3.4.2 Flowchart

The flowchart representing the PSO code is shown in Figure 3.9. The algorithm starts by initializing a population (particles) of random design points. In the case of PSO, the random solutions are normalized for a better performance. The population size (n) is user defined. The random design points are evaluated to give in the next step. Initially, the best position values ($pbest$) and the best objective values ($pbestval$) of each particle are assigned to staring position values (pos) and the best starting values (out).

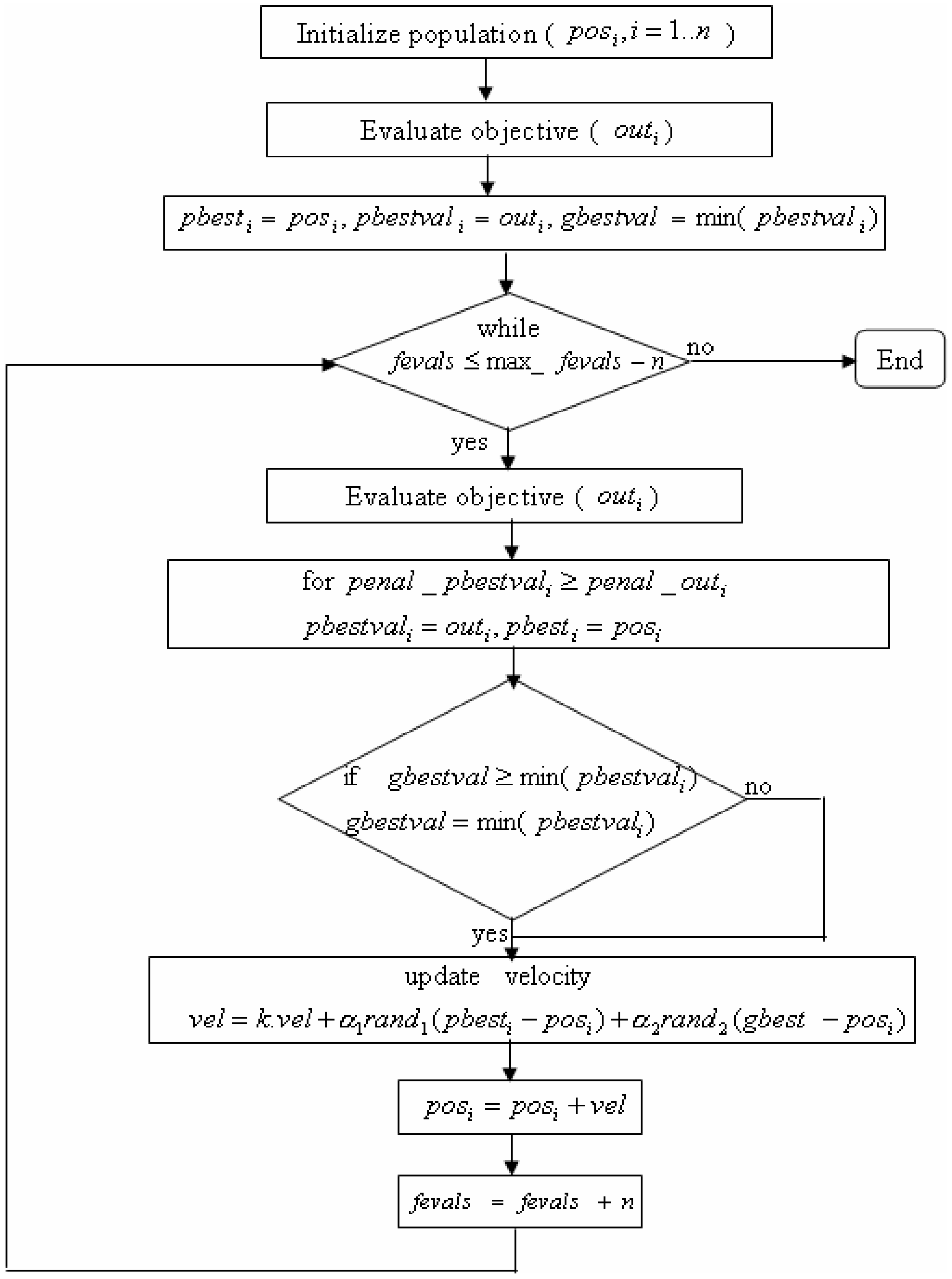

Initialize population ($pos_i, i = 1..n$)
Evaluate objective (out_i)
$pbest_i = pos_i, pbestval_i = out_i, gbestval = \min(pbestval_i)$
while $fevals \leq \max\_fevals - n$
no
End
yes
Evaluate objective (out_i)
for $penal\_pbestval_i \geq penal\_out_i$
$pbestval_i = out_i, pbest_i = pos_i$
if $gbestval \geq \min(pbestval_i)$
$gbestval = \min(pbestval_i)$
no
yes
update velocity
$vel = k.vel + \alpha_1 rand_1(pbest_i - pos_i) + \alpha_2 rand_2(gbest - pos_i)$
$pos_i = pos_i + vel$
$fevals = fevals + n$

Figure 3.9 Flowchart of Particle Swarm Optimization

The best value of all the particles ($gbestval$) is the least value of all $pbest's$. The variable $gbestval$ holds the current best global maximum of the objective function. Next, a while loop is run for specified number of function evaluations. The while loop updates the *pbest* and *pbestval* values if there is any improvement in the current *pbestval* value. It also updates *gbestval* if there is an improvement. The updated *pbest* and *gbest* are used in determining the new velocity (*vel*) and position (*pos*). The function evaluations (*fevals*) are incremented by the population size every iteration.

3.4.3 Advantages/Disadvantages

The major advantage of the PSO algorithm is that fewer parameters must be adjusted compared to SA and GA. The constants in the updating velocity are very critical in obtaining better performance. Many sets of constants are available suitable for specific problems. Moreover, no natural operators like crossover, mutation and selection are present in PSO. PSO is easier to understand because it involves simple equations.

The disadvantage lies in the selection of the constants in the velocity updating. If inappropriate constants are taken then the problem may not converge to the optimum.

3.5 Summary of Global Algorithms

A summary of global algorithms is given in Table 3.1. The thesis focuses on four global algorithms – DIRECT, Simulated Annealing, Genetic Algorithm, and Particle Swarm Optimization.

Of the four optimization algorithms, DIRECT is a deterministic algorithm. Deterministic algorithm follows a fixed way towards the global optimum in every searching process. The other three algorithms are stochastic algorithms. Stochastic algorithms start at a random point and follow a random path towards the global optimum in every searching process.

DIRECT moves towards the global optimum by trisecting the multi-dimension design space and selecting the optimal rectangles. It begins at the center point of the multi-dimensional design space in the searching process. The constraints in DIRECT are handled using an auxiliary function given in equation 6. The main advantages of DIRECT are it covers the entire design space in search of global optimum and it does not have tuning parameters.

Simulated Annealing is based on the annealing process in metals. As the temperature is decreased, SA accepts better points that improve the objective function. The constraints in SA are handled using a penalty function given in equation 8. SA is able to find the global optimum without covering the entire design space. The disadvantage of SA is that its performance depends on the tuning parameters.

Genetic algorithm is based on the evolutionary processes and Darwin's concept of natural selection. It begins by seeding a random population and it moves towards the global optimum by selecting best populations while weeding out bad populations. During the process, GA uses operators like crossover, mutation and natural selection to find the optimal solution. A number of different types of operators can be selected which may suit the design problem.

Particle Swarm optimization is based on the swarm intelligence found in the natural systems. In a swarm, the agents or design points fly in a multi-dimensional space towards the optimal solution using the experience of itself and the experience of its neighbors. Like GA, PSO also begins by seeding a population and updates the velocity and position of each design point until it reaches the global solution. The constraints in PSO are handled using the same penalty function as in the case of GA and SA. PSO is simple to understand because of its simple equations. The disadvantage of PSO is in selecting the optimal parameter values.

Table 3.1 Summary of global algorithms

Algorithm	Type	Starting point	Tuning parameters	Initialize population	Handling constraints
DIRECT	Deterministic	Center point	No	No	Auxiliary function given in equation 6
Simulated Annealing	Stochastic	Random	Yes	No	Penalty function given in equation 8
Genetic Algorithm	Stochastic	Random	Yes	Yes	Penalty function given in equation 8
Particle Swarm optimization	Stochastic	Random	Yes	Yes	Penalty function given in equation 8

CHAPTER IV

SIMULATION AND ANALYSIS

4.1 Problem Statement

Hybrid Electric Vehicle has the potential to substantially improve fuel economy due to its efficient engine loading. The extent of this improvement greatly depends on the component optimization. The input-output relationships of these components are modeled using the design variables in the modeling process. Each component is modeled using several design variables. For instance, the energy storage system has variables representing the number of energy modules, maximum capacity of each module, nominal voltage of each module, weight, etc. Because the HEV model involves hundreds of design variables, it is difficult to optimize all the design variables. So, only those design variables that have a maximum impact on the final objective are considered for the optimization. The power ratings of the energy sources are definitely the first choice in the design variables.

The objective of this optimization problem is to maximize fuel economy on a composite driving cycle and to improve the vehicle performance. The driving cycle is composed of city driving represented by FTP-75 (Federal Test Procedure) and the Highway driving represented by HEFET (Highway Fuel Economy Test). The two drive cycles are shown in Figure 4.1 and Figure 4.2.

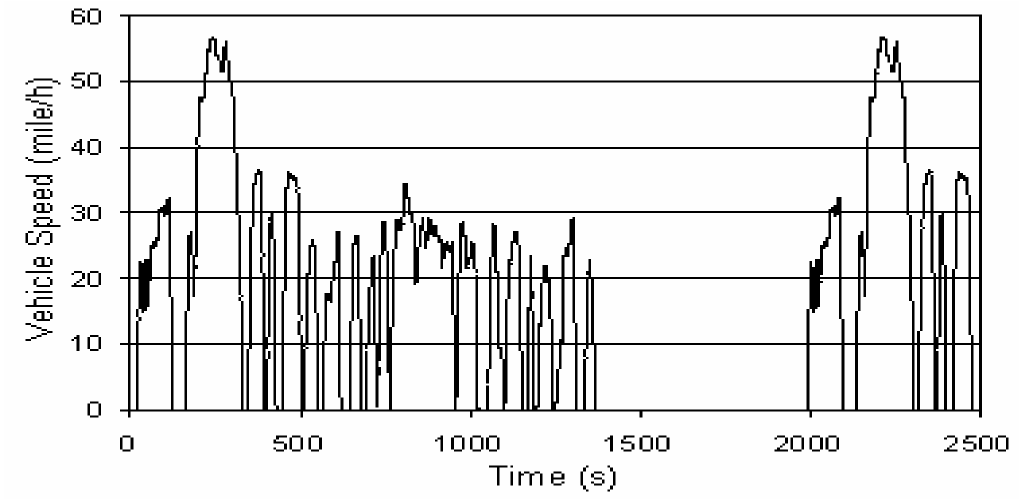


Figure 4.1 FET -75 drive cycle

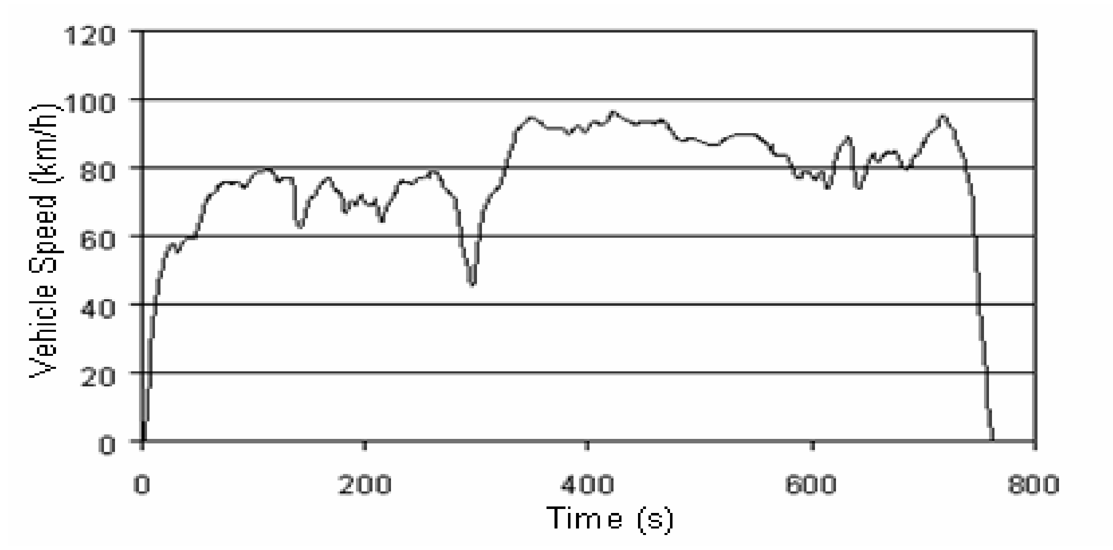


Figure 4.2 HWFET drive cycle

The fuel economy from each of these drive cycles is combined to get the composite fuel economy. By definition, composite fuel economy is the harmonic average of the SOC-balanced fuel economy values during the two separate drive cycles. The composite fuel economy can be calculated as follows:

$$CompositeFuelEconomy = \frac{1}{\frac{0.55}{City\_FE} + \frac{0.45}{Hwy\_FE}}$$

where *City_FE* and *Hwy_FE* represents the city and highway fuel economy values respectively.

In this study, the vehicle model gui_par_midsize_cavalier_ISG_in.m (available in the PSAT model library) is taken for the optimization study. This vehicle is a 2 wheel-drive starter-alternator parallel configuration with manual transmission. The basic configuration of the parallel HEV used for optimization is illustrated in Figure 4.3 and main components of the vehicle are listed in Table 4.1.

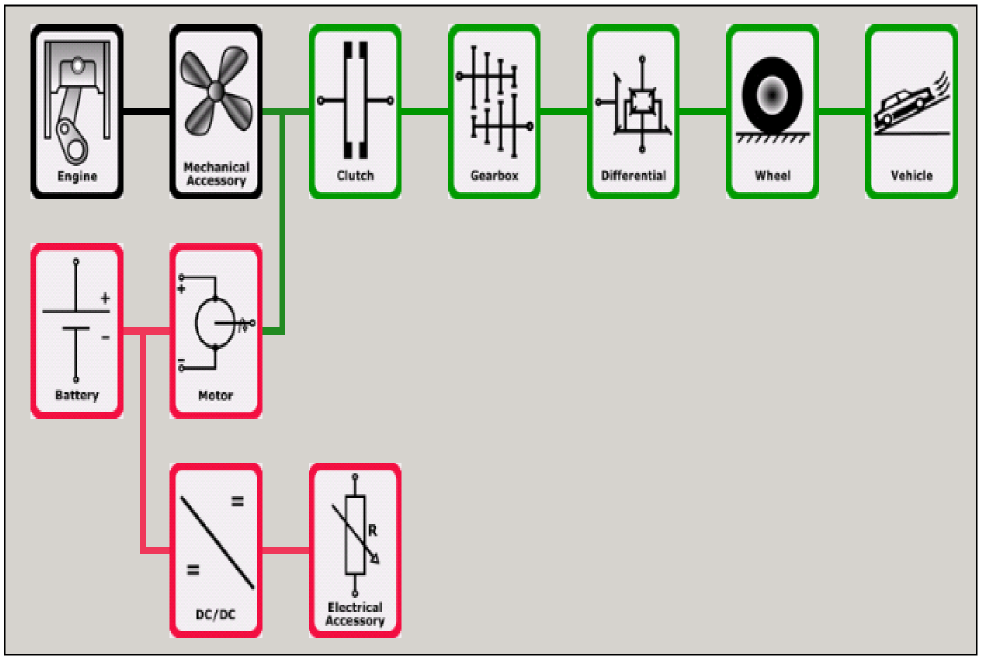


Figure 4.3 Configuration of the selected HEV

Table 4.1 Selected Parallel HEV configuration

Component	Description
Fuel Converter	84 kW and 2.2 L Cavalier Gasoline Engine
Motor	ECOSTAR motor model with continuous power of 33 kW and peak power of 66 kW
Battery	NiMH Panasonic Battery with capacity 6.5 Ah and 240 cells
Transmission	4 speed manual gearbox with final drive ratio 3.63
Control strategy	Propelling, Shifting and Braking

Table 4.2 shows the six design variables and their description. The first two define the power ratings of the fuel converter and motor controller. The third, fourth and fifth variables define the number of battery modules, minimum battery State of Charge (SOC) allowed and maximum battery SOC allowed. The sixth design variable defines final drive ratio. The lower and upper bounds for the design variables are given in third and the fourth column respectively.

Table 4.2 Lower and upper bounds of design variables

Design Variable	Description	Lower Bound	Upper Bound
eng.scale.pwr_max_des	Fuel converter power rating	40 kW	100 kW
mc.scale.pwr_max_des	Motor Controller power rating	10 kW	80 kW
ess.init.num_module	Battery number of cells	150	350
ess.init.soc_min	Minimum SOC allowed	0.2	0.4
ess.init.soc_max	Maximum SOC allowed	0.6	0.9
fd.init.ratio	Final drive ratio	2	4

Vehicle performance constraints are imposed on the design problem to ensure the requirement on performance of the vehicle is met. The following constraints are imposed on the design problem:

Acceleration time	0 - 60 mph <= 18.1 s
Acceleration time	40 - 60 mph <= 7 s
Acceleration time	0 - 85 mph <= 35.1 s

Maximum Acceleration >= 3.583 m/s^2

The general optimization procedure followed for the HEV design problem is described here. The optimization procedure can be thought as a two-step process. In the first step, the default vehicle model is run for the fuel economy and vehicle performance values. In the second step, the same vehicle model with bounds given in Table 4.2 is looped with the optimization routines and run for some predefined number of function evaluations. Then, the fuel economy and the vehicle performance values for each optimization routine are recorded. The stochastic algorithms are run twice and the best results are considered. A comparison of the fuel economy and the vehicle performance is done at the end for each optimization algorithm. Also, a comparison of the algorithm efficiency is also done.

4.2 Running a Simulation

In section 2.1, a brief overview is given about the optimization process. The optimization process is explained in detail here with the Matlab files (.m files) used for the DIRECT algorithm. However, the process is same for the other three algorithms. The optimization is carried out using five Matlab files. The interaction between the five files is illustrated in Figure 4.4 and a brief description of each file is given below:

- *psat_opt.m*: Main script file to run the optimization.
- *gclSolve.m*: Matlab implementation of the DIRECT algorithm.
- *run_func.m*: Loads objective function value (*f*) by calling *run_psat.m*.
- *run_psat.m*: Invokes PSAT and evaluates objective function value (*f*) and performance values (*g*).
- *run_con.m*: Loads the vehicle performance values (*g*).

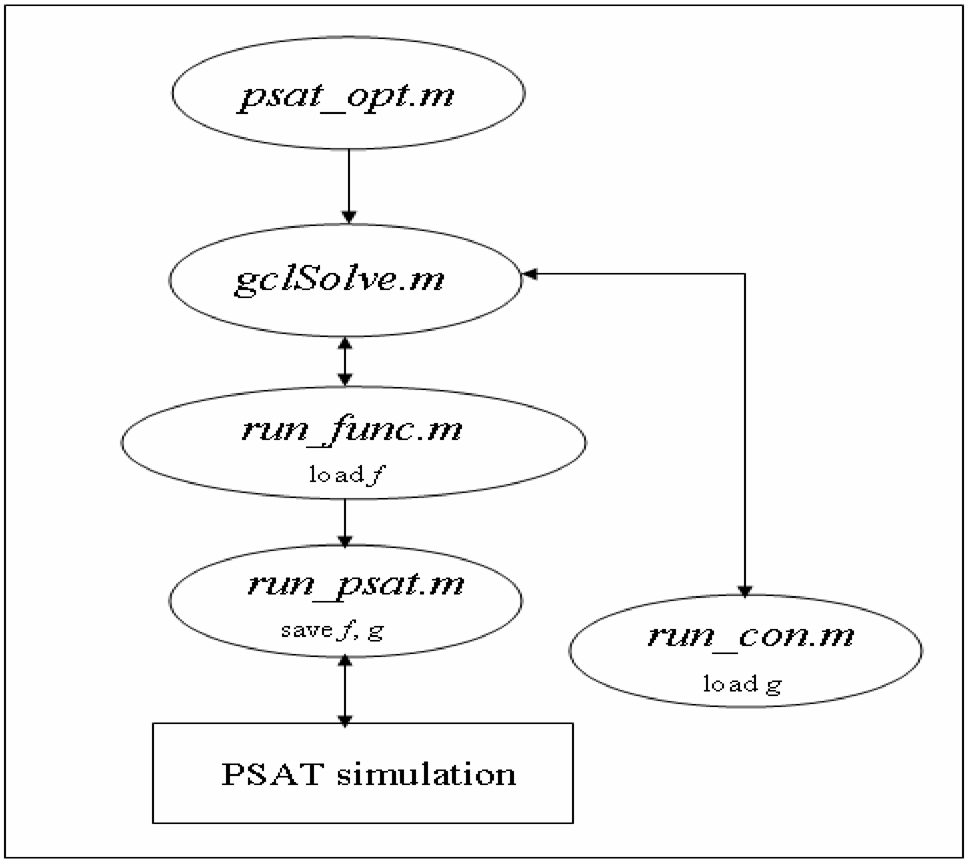


Figure 4.4 File interaction in HEV optimization

From the above figure, it can be seen that the optimization is started by running the *psat_opt.m* file. The design variable bounds and maximum number of function evaluations are also assigned in this file. It then calls *gclSolve.m* (Matlab implementation of DIRECT algorithm). Then the DIRECT algorithm iterates until the maximum number of function evaluations are exhausted. For each function evaluation, DIRECT feeds the *run_func.m* with a design point for an improvement in the objective function. The *run_func.m* in turn calls *run_psat.m.* In the *run_psat.m*, the PSAT is invoked and the objective function (*f*) and vehicle performance values (*g*) are evaluated. Now the process returns to the *run_func.m* where the objective function value (*f*) is loaded. Next, the process returns to *gclSolve.m.* Later, *gclSolve.m* calls *run_con.m* where the performance values are loaded.

4.3 HEV Optimization Results

Step (1): In this step, the default vehicle model gui_par_midsize_cavalier_ISG_in.m is run to get the fuel economy and the vehicle performance values. This is done in the *simulation setup* in the PSAT GUI. The design variables are given in Table 4.3. The fuel economy was observed to be 35.1 mpg and the vehicle performance values are given in Table 4.4. Note here that the constraints of the HEV design problem are actually the performance of the initial vehicle.

Table 4.3 Initial design variable values

Design variable	Initial Value
eng.pwr_max_des	86 kW
mc.pwr_max_des	65.9 kW
ess.init.num_module	240
ess.init.soc_min	0
ess.init.soc_max	1
fd.init.ratio	3.63

Table 4.4 Initial vehicle performance

Performance parameter	Performance value
Time for 0 – 60 mph	18.1 s
Time for 40 – 60 mph	7 s
Time for 0 – 85 mph	35.1 s
Maximum acceleration	3.583 m/s^2

Step (2) In the second step, the optimization algorithms DIRECT, Simulated Annealing, Genetic Algorithm, and PSO discussed in chapter III are looped with the PSAT vehicle simulator and the optimization is carried on. For this step, the same default vehicle configuration given in Table 4.1 is taken and the bounds for the design variables are taken as given in Table 4.2. All the four algorithms are allowed to run for 400 function evaluations. Using the same number of function evaluations will allow us to compare the performance of the different algorithms. The entire optimization process for each algorithm took approximately 80 - 90 hours to complete. This huge process time is not due to the algorithm but because of the large time PSAT takes to evaluate each design point. For each design point, PSAT takes approximately 10 minutes to evaluate and the algorithm takes approximately 2 minutes for the calculations.

A comparison of the fuel economy before and after the optimization is given in Table 4.5. A significant improvement in the fuel economy is seen due to optimization although the improvement is less in the case of GA and PSO. Of all the four algorithms, SA performed extremely well with an improvement of 5.27 mpg, followed closely by DIRECT algorithm with an improvement of 4.54 mpg. This shows that SA is more efficient in finding a global optimal solution for this HEV design problem. It was also observed that rough initial populations in the case of GA and PSO were responsible for their not-so-good performance.

Table 4.5 Comparison of fuel economy

Fuel Economy				
Before Optimization	After Optimization			
	DIRECT	SA	GA	PSO
35.1 mpg	39.64 mpg	40.37 mpg	37.6 mpg	37.1 mpg

A comparison of the initial design variables and the optimum design variables found by the four optimization algorithms is given in Table 4.6. We see that DIRECT, SA and PSO found an almost identical optimum design point (except the variations in the number of energy modules), suggesting that these solutions are almost the true global optimum. The global optimum is however not validated since PSAT is a quasi-steady model and so it is difficult to arrive at an optimal design point in PSAT just by using PSAT models. For the GA case, the lack of improvement in the fuel economy is justified by the design point found to be distant from the global optimum, suggesting this is highly likely a local minimum. It was also noticed that DIRECT and SA algorithms reduced the power ratings of the engine and the motor significantly.

Table 4.6 Design variables final values

Design variable	Initial value	Final value			
		DIRECT	SA	GA	PSO
eng.pwr_max_des	86 kW	83.1 kW	82.4 kW	95.5 kW	87.1 kW
mc.pwr_max_des	65.9 kW	20.2 kW	21.9 kW	24.2 kW	14.8 kW
ess.init.num_module	240	245	311	300	238
ess.init.soc_min	0	0.25	0.22	0.34	0.26
ess.init.soc_max	1	0.84	0.78	0.89	0.78
fd.init.ratio	3.63	3.9	4.0	3.49	3.42

All the four optimization algorithms resulted in improved vehicle performance. The performance comparison of the Hybrid Electric Vehicle before and after the optimization is given in Table 4.7. It shows that the optimized vehicle performance is greatly improved compared to the unoptmized vehicle performance. The performance improvement by SA is far better compared to the other three algorithms.

Table 4.7 Comparison of the vehicle performance

Constraint	Constraint value	Before Optimization	After Optimzation			
			DIRECT	SA	GA	PSO
0 – 60 mph	<=18.1 s	18.1 s	15.5 s	10.8 s	11.9 s	11.1 s
40 – 60 mph	<=7 s	7 s	6.8 s	5 s	4.4 s	4.9 s
0 – 85 mph	<=35.1 s	35.1 s	30.6 s	20.7 s	21.2 s	22 s
Max. Acceleration	>=3.583 m/s^2	3.583 m/s^2	3.97 m/s^2	4.07 m/s^2	3.94 m/s^2	3.99 m/s^2

The mass of the vehicle changes as the design variables change because the mass of the vehicle depends on the design variables. In particular, of the chosen six design variables, four design variables (power ratings of engine and motor, energy modules and final drive ratio) affect the mass of the vehicle. The mass of the vehicle before and after the optimization is given in Table 4.8. The mass of the vehicle decreased with DIRECT and SA algorithms while the mass of the vehicle increased with GA and PSO algorithms.

Table 4.8 Comparison of the mass of the vehicle

Mass of the vehicle				
Before Optimization	After Optimization			
	DIRECT	SA	GA	PSO
1683 kg	1635 kg	1656 kg	1694 kg	1690 kg

The algorithm performance comparison of the global optimization algorithms is shown in Figure 4.5. It shows the best objective function value plotted against the number of function evaluations.

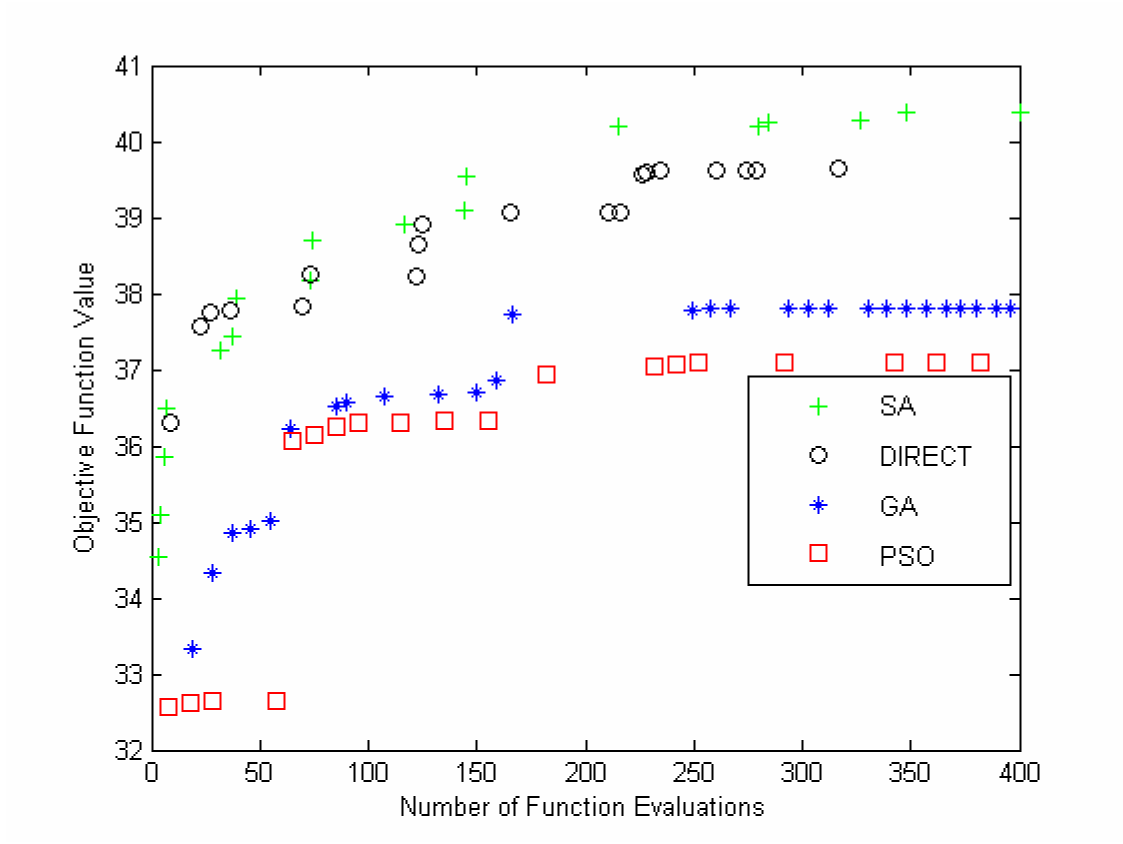


Figure 4.5 Algorithm performance comparison

We can see that fuel economy improvement with the SA and DIRECT algorithms is very close until 125 function evaluations, after which SA leaped ahead of DIRECT. GA and PSO are slow to catch SA and DIRECT initially because they take some function evaluations to generate the initial population. The improvement in fuel economy with GA and PSO algorithms is similar. After 225 function evaluations, GA and PSO did not find any good design point to get further improvement in the fuel economy.

Figure 4.5 also shows that the rate of improvement in the fuel economy reduced considerably after 200 function evaluations. Infact, there is very small improvement for large number of function evaluations. This means that the algorithms have reached the plateau region (local) in the objective response and requires large number of function evaluations to reach the true global optimum point. The rate of fuel economy improvement in the local region can be improved by using a local or derivative-based algorithm. The local based algorithms use the derivatives of the objective function to find the path of greatest improvement towards the global optimum. A hybrid algorithm which is a combination of a global and a local algorithm is developed and tested on two simple mathematical functions. The hybrid algorithm results are presented in Appendix A.

The results in this study can be implemented physically to obtain real performing vehicle because PSAT is a quasi-steady simulation tool using models and data from the vehicles in the real world. Here, the vehicle 'gui_par_midsize_cavalier_ISG_in.m' used in this study is based on a midsize Cavalier vehicle. So, the optimization can be viewed as scaling the models up or down to increase the fuel economy and improve the vehicle performance. And also, the scaling is defined using the bounds which are feasible in the real world. For example, in the case of the DIRECT algorithm, it can be seen from Table 4.6, the engine power rating is scaled down from 86 kW to 83.1 kW, motor power rating is scaled down from 65.9 kW to 20.2 kW, number of battery cells are increased from 240 to 245, minimum soc allowed is increased from 0 to 0.25, maximum soc is decreased from 1 to 0.84, and the final drive ratio is increased from 3.63 to 3.9.

CHAPTER V

CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

The objectives of this thesis are two-fold. The first objective is to increase the composite fuel economy along with an improvement in the vehicle performance of the selected Hybrid Electric Vehicle (HEV). The second objective is to see the effect of different optimization algorithms on the HEV design problem. Global optimization algorithms like DIRECT, Simulated Annealing, Genetic Algorithm and Particle Swarm optimization are used. Out of the four algorithms, DIRECT is a deterministic algorithm and the other three are stochastic algorithms.

Results showed that Simulated Annealing found the best optimal solution with a fuel economy improvement of approximately 5 mpg compared to the other three algorithms. This also shows that SA has the best overall local convergence ability. Also, Simulated Annealing performed extremely well out of the four algorithms with best rate of objective function value improvement followed closely by the DIRECT algorithm. The GA and PSO algorithms were observed to be slow in objective function improvement. The slow improvement in the objective function value may be due to the bad initial population generated by these two algorithms.

In the case of vehicle performance, the improvement in performance value is more using SA when compared to the other three algorithms except the 40 – 60mph performance value. Correct tuning of the parameters in the SA was observed to be the reason for this performance improvement. The other three algorithms also performed well to get the vehicle performance close to the vehicle performance achieved by SA.

The algorithms SA, GA and PSO are stochastic-based methods. These algorithms follow a random path in every searching process for finding the global optimum. So, these algorithms are run many times to find the best optimal solution. In this design problem, the SA is run once while the GA and PSO are run twice before finding the optimal results given in this thesis. Therefore, if total simulation time is a concern then DIRECT (follows fixed path in every search process) is the best algorithm of all the four optimization algorithms.

The drawback of the global optimization algorithms is the slow convergence to the true global optimum once they reach the global optimum region. For the purpose of improving convergence, a hybrid algorithm is developed in which a fast converging and derivative-based local algorithm (SQP) is combined with the global algorithm (DIRECT). Results showed that the hybrid algorithm converged quickly in the case of banana function to the global optimum taking less number of function evaluations.

5.2 Future Work

In this thesis, a hybrid electric vehicle is optimized for better fuel economy and vehicle performance. The HEV design problem involves six design variables and four vehicle performance constraints. The number of design variables can be increased between 10 – 15 in the design problem for further improvement in the objective function value. However, increasing the number of design variables beyond 20 increases the complexity of the design problem and may give diminishing results. The scalability of each algorithm can be studied. In the case of stochastic algorithms, the study of scalability may be difficult because of the randomness involved in the algorithm performance.

The constraints in the HEV design problem are actually limited to the performance values of the initial vehicle. More stringent constraints can be assigned to get a better vehicle performance but one should take care that stringent constraints result in fewer feasible design points. With fewer feasible points, the algorithm needs more number of function evaluations to reach the global optimum because some function evaluations will be wasted in finding infeasible points.

Except for the DIRECT, the performance of the other three algorithms depends on the parameter tuning. Since this is a simulation based optimization, there is no theoretical way of deciding the optimal parameter values. A trial and error method can be employed to arrive at the best values for the parameters.

The developed hybrid algorithms works well in the case of simple functions like Banana and Camel functions, but when it comes to the case of complex problems like the HEV design problem it failed to work. The reason for this lies in the calculation of gradients for the search direction and second order derivatives for updating the Hessian matrix in the SQP algorithm. SQP finds these derivatives using numerical approximations. So, it is highly possible that it can get in to trouble in calculating the first and second order derivatives. Also, many technical papers and user-groups say that algorithms get in to trouble if the algorithms have to find both first and second order of derivatives numerically unless first order is provided analytically [23, 24]. One of the possible solutions for this problem could be to find a local algorithm which can calculate two orders of derivatives numerically for this complex problem. Other solution is to use local algorithms like nonlinear conjugate gradient methods which uses single order of derivatives to find the search direction [25].

This thesis focuses on using four global optimization algorithms for the HEV design problem. Tabu Search, a meta-heuristic search technique is also used successfully in optimizing several complex problems. It avoids revisiting the design points already visited in the design space and in this process the algorithm may accept inferior points. This approach can lead to exploring new regions in the design space, with the goal of finding a solution by globalize search. This approach reduces the number of function evaluations. This algorithm can be applied to the HEV design problem.

The long simulation time is a concern in the HEV problem. It takes 80 – 90 hours to complete 400 function evaluations. So, distributed or parallel computing can be employed to reduce this huge simulation time.

REFERENCES

[1] G. Zorpette, “The Smart Hybrid,” *IEEE Spectrum*, vol. 41 issue 1, pp. 44 - 47, Jan. 2004.

[2] I. Husain, “Electric and hybrid vehicles: Design Fundamentals,” CRC Press, New York, 2003.

[3] C. C. Chan and K. T. Chau, “Modern Electric Vehicle Technology,” Oxford University Press, 2001.

[4] C. C. Chan, “The state of the Art of Electric and Hybrid vehicles,” *Proc. of the IEEE*, vol. 90, no. 2, Feb. 2002.

[5] PSAT 2005 Brochure [Online]. Available: http://www.transportation.anl.gov/pdfs/MC/338.pdf

[6] PSAT Technical Information [Online]. Available: http://www.transportation.anl.gov

[7] A. D. Belegundu, and T. R. Chandrupatla, “Optimization Concepts and Applications in Engineering,” Prentice Hall, New Jersey 1999.

[8] G. H. Cole, “SIMPLEV: A Simple Electric Vehicle Simulation Program, Version 2.0 ,” EG&G Idaho, Inc., April 1993.

[9] A. Campbell, A.Rengan, J. Steffey, and J. Ormiston, “The Simulation of 42-Volt Hybrid Electric Vehicle” [online] http://www.math.msu.edu/Graduate/

[10] K. L. Butler, M. Ehsani, and P. Kamath, “A Matlab-Based Modeling and Simulation Package for Electric and Hybrid Electric Vehicle Design,” *IEEE Trans. on Veh. Tech.*, vol. 48, no. 6, pp. 1770-1778, Nov. 1999.

[11] R. D. Senger, “Validation of ADVISOR as a Simulation Tool for a Series Hybrid Electric Vehicle Using the Virginia Tech FutureCar Lumina,” Thesis submitted to Virginia Polytechnic Institute and State University, 1997.

[12] K. B. Wipke, M. R. Cuddy, and S. D. Burch, “ADVISOR 2.1: A User-Friendly Advanced Powertrain Simulation Using a Combined Backward/Forward Approach,” NREL/JA-540-26839, Sep. 1999.

[13] K. Schittkowski, "NLQPL: A FORTRAN-Subroutine Solving Constrained Nonlinear Programming Problems," *Annals of Operations Research*, vol. 5, pp 485-500, 1985.

[14] D. R. Jones, “DIRECT Global Optimization Algorithm,” Encyclopedia of Optimization, Kluwer Academic Publishers, 2001.

[15] D. R. Jones, C. D. Perttunen, and B. E. Stuckman, “Lipschitz optimization without Lipschitz constant,” *Journal of Optimization Theory and Applications*, vol. 79, no. 1, Oct. 1993.

[16] B. Shubert, “A Sequential Method Seeking the Global Maximum of a Function,” *SIAM Journal on Numerical Analysis*, vol. 9, pp. 379-388, 1972.

[17] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by simulated annealing,” *Science*, vol. 220, pp. 671-680, 1983.

[18] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and, E. Teller, “Equations of State Calculations by Fast Computing Machines,” *Journal Chem. Phys.*, vol. 21, pp. 1087- 1092, 1958

[19] D. E. Goldberg, Genetic Algorithms in Search and Machine Learning, Addison Wesley, Reading, 1989.

[20] H. J. Holland, Adaptation in Natural and Artificial Systems, an introductory analysis with application to biology, control and artificial intelligence, The University of Michigan Press, Ann Harbor, USA, 1975.

[21] J. Kennedy, and R. Eberhart, “Particle Swarm Optimization,” *Proc. IEEE Intl. Conference on Neural Networks*, vol. IV, pp.1942-1948, 1995.

[22] I. C. Trelea, “The particle swarm optimization algorithm: convergence analysis and parameter selection,” *Information Processing Letters*, vol. 85, Issue 6, Mar. 2003.

[23] O. Wing, J. V. Behar, “Circuit design by minimization using Hessian matrix,” *IEEE Transactions on Circuits and Systems*, vol. 21, no. 5, Sep. 1974.

[24] A private conversation with Marcelo Marazzi in Mathworks.com Newsgroup.

[25] A powerpoint presentation from Systems Realization Laboratory, Georgia Institute of Technology. Available online: http:// www.srl.gatech.edu/education/ ME6103/NLP-Unconstrained-Multivariable.ppt

[26] MATLAB Optimization Toolbox 3.0.2 Documentation [Online]. Available: http://www.mathworks.com/products/optimization/index.html

[27] Optimization Toolbox: For Use with Matlab [online]. Available: http://www.math.ntnu.no/~hek/Optimering2005/OptimizationToolboxDocumentationVersion3.pdf

APPENDIX

HYBRID ALGORITHM

Global algorithms are known for their slower convergence to the true global optimum once the optimum region is found. For example, the DIRECT converges to the optimum region with few number of function evaluations but reaches the true global optimum taking a large number of additional function evaluations. For a costly objective function evaluation like the one used in this study, it means waste of computing power. This drawback of the global algorithms can be overcome by combining it with local gradient-based algorithms, which are known for their faster convergence. This hybrid approach improves the efficiency of the algorithm and also avoids the need to specify a good initial point for the derivative-based methods. A hybrid algorithm combining the global algorithm (DIRECT) and local algorithm (SQP) is developed in this section. SQP algorithm is available in *fmincom.m* function as part of the Matlab optimization toolbox.

Sequential Quadratic Programming (SQP) is a common gradient-based approach for solving non-linear problems. It is available as *fmincon.m* function in the Matlab Optimization Toolbox function library [26]. It starts from an initial point x_n and an initial approximation of the Hessian H of the objective function. There are three main stages in reaching an optimal solution:- updating the H matrix, form and solve a Quadratic Programming (QP) sub-problem to get a search direction, and line search to obtain a new approximation of the potential solution.

At each iteration, a positive definite H matrix is updated using quasi-Newton approximation approach. The function *fmincon.m* uses the Broyden, Fletcher, Goldfarb and Shanno (BFGS) method for the Hessian matrix approximation [27]. The Hessian update is done using the following equation:

$$H_{n+1} = H_n + \frac{q_n q_n^T}{q_n^T s_n} - \frac{H_n^T H_n}{s_n^T H_n s_n} \tag{13}$$

where $s_n = x_{n+1} - x_n$ and $q_n = \nabla f(x_{n+1})$. Starting from an initial approximation of the solution, a Quadratic Programming (QP) problem is solved at each iteration of the SQP method, yielding a direction in the search space. To this direction, a vector is obtained through line search, in order to produce a sufficient decrease of a merit function. This point is considered the new approximation of the solution.

The implementation of the hybrid algorithm is illustrated using the flowchart given in Figure A.1.

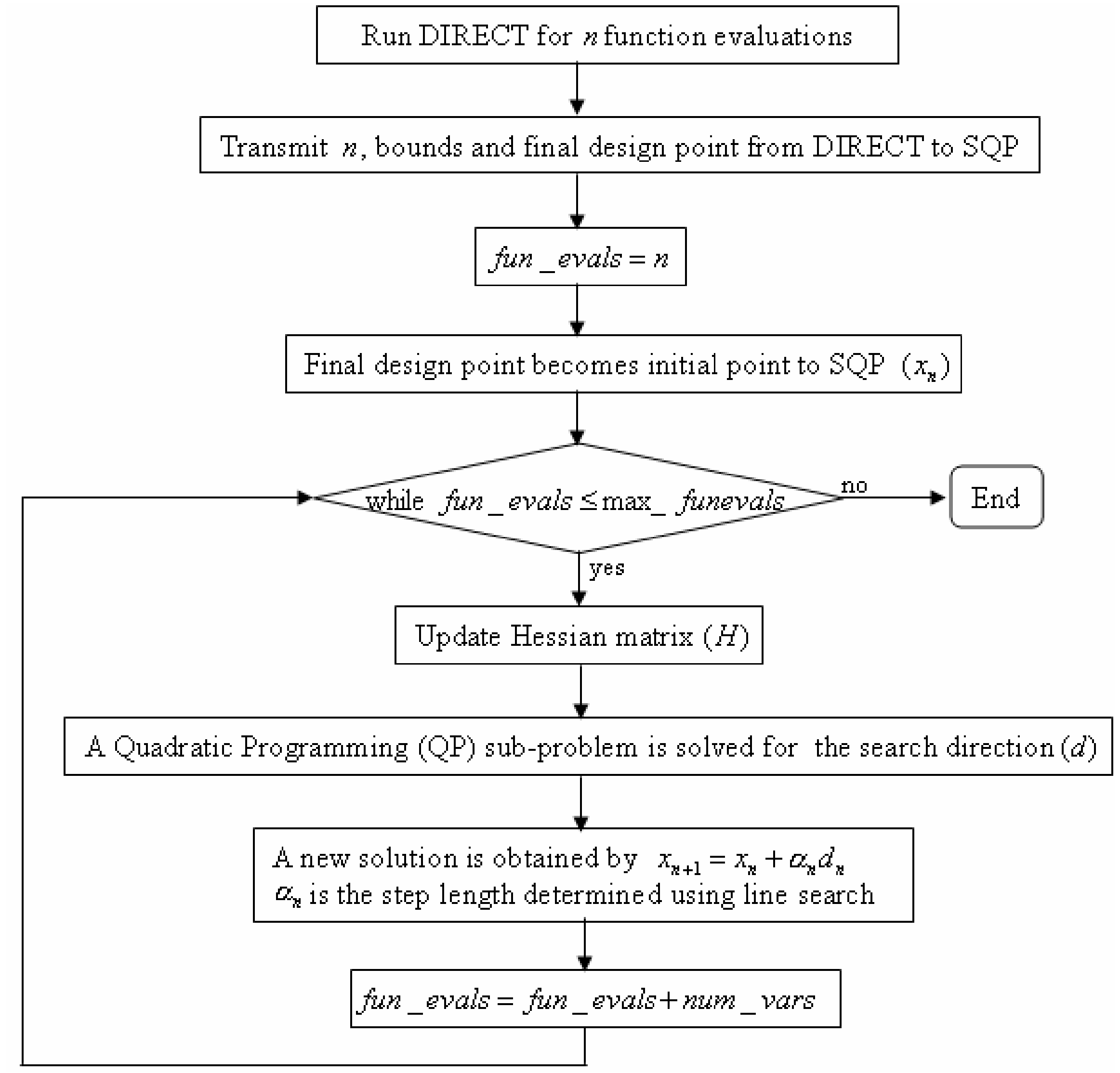

Run DIRECT for n function evaluations
Transmit n, bounds and final design point from DIRECT to SQP
$fun\_evals = n$
Final design point becomes initial point to SQP (x_n)
while $fun\_evals \le \max\_funevals$
no
End
yes
Update Hessian matrix (H)
A Quadratic Programming (QP) sub-problem is solved for the search direction (d)
A new solution is obtained by $x_{n+1} = x_n + \alpha_n d_n$
α_n is the step length determined using line search
$fun\_evals = fun\_evals + num\_vars$

Figure A.1 Flowchart showing the hybrid algorithm implementation

The Hybrid algorithm begins by running the global algorithm DIRECT for a specified number of iterations n. The final design point and the number of function evaluation n are transmitted to the SQP algorithm. The SQP algorithm starts at the final design point obtained from the DIRECT algorithm. The SQP algorithm will be iterated until the maximum number of function evaluations $\max\_funevals$ and returns the optimal solution.

To test its efficiency, the hybrid algorithm is applied to minimize two test functions – Rosenbrook's Banana function and 3-Hump Camel Back function followed by a comparison between the DIRECT, SQP and Hybrid algorithm. The efficiency is based on the least number of function evaluations each algorithm takes to reach the assumed objective function error value of 0.01.

The equations for the Banana and the Camel function are given in equation (14) and (15) respectively. Their corresponding plots are shown in Figure A.2.

$$f(x,y)=100(y-x)^2+(1-x)^2 \tag{14}$$

$$f(x,y)=2x^2-1.05x^4+\tfrac{1}{6}x^6-xy+y^2 \tag{15}$$

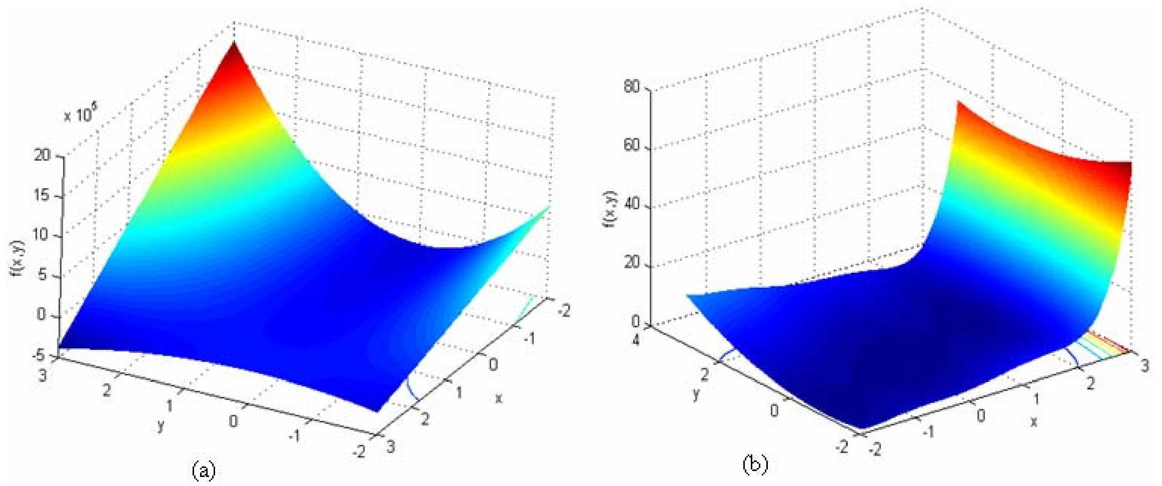


Figure A.2 Two test functions: (a) Rosenbrook's banana (b) 3-hump Camel back

The global minimum for the Banana function occurs at (1, 1) with a function value of 0. The global minimum for the Camel function occurs at (0, 0) with a function value of 0. The number of function evaluations the DIRECT algorithm is allowed to run (n) is assumed to be 10. Table A.1 and Table A.2 summarize the results obtained for the Banana and Camel function respectively.

Table A.1 Hybrid algorithm on Rosenbrook's banana function

Algorithm	Bounds [x; y]	Initial point	Function Evaluations	Final design point	Objective function value
DIRECT	[0 3; 0 3]	Not required	175	(1.03 1.05)	9e-3
SQP	[0 3; 0 3]	[2 2]	56	(0.996 0.992)	1.11e-4
Hybrid	[0 3; 0 3]	Not required	10+23=33	(0.992 0.981)	1.12e-3

Table A.2 Hybrid algorithm on Camel function

Algorithm	Bounds [x; y]	Initial point	Function Evaluations	Final design point	Objective function value
DIRECT	[-2 3; -2 3]	Not required	13	(-.055 -.055)	6.163-3
SQP	[-2 3; -2 3]	[2 2]	28	(-.000071 -.00019)	3.43e-8
Hybrid	[-2 3; -2 3]	Not required	10+8=18	(-.036 -.053)	3.03e-3

In the case of Banana function, the hybrid algorithm performed extremely well taking only 33 function evaluations to reach an objective value less than 0.01. In the case of the Camel function, the hybrid algorithm did not perform well compared to the DIRECT algorithm but it was better compared to SQP. This can be understood by the fact that DIRECT performed extremely well taking only 13 function evaluations to satisfy the stopping criterion.