MULTI-OBJECTIVE OPTIMIZATION USING EVOLUTIONARY ALGORITHMS FOR COMPUTATION INTENSIVE APPLICATIONS

A DISSERTATION

Submitted in partial fulfillment of the requirements for the award of the degree
of
MASTER OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING

By

DIVYA GOEL

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE -247 667 (INDIA)
JUNE, 2010

Candidate Declaration

I hereby declare that the work being presented in the dissertation report titled "**Multi-Objective Optimization using Evolutionary Algorithms for Computation Intensive Applications**" in partial fulfillment of the requirement for the award of the degree of Master of Technology in Computer Science and Engineering, submitted in the Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, is an authentic record of my own work carried out under the guidance of Dr Rajdeep Niyogi and Dr M.V.Kartikeyan in the Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee. I have not submitted the matter embodied in this dissertation report for the award of any other degree.

Dated: 3.6.10

Place: IIT Roorkee.

Divya Goel [illegible]

Certificate

This is to certify that above statements made by the candidate are correct to the best of our knowledge and belief.

Dated: 3.6.10

Place: IIT Roorkee.

[illegible]

Dr Rajdeep Niyogi
Assistant Professor,
Department of Electronics
and Computer Engineering.

[illegible]

Dr. M.V.Kartikeyan
Professor,
Department of Electronics
and Computer Engineering.

Acknowledgements

I am thankful to Indian Institute of Technology Roorkee for giving me this opportunity. It is my privilege to express my deep sense of gratitude and indebtedness to my guide **Dr. Rajdeep Niyogi** whose invaluable guidance, constant motivation, patience and enthusiasm were immensely helpful in carrying out this work. I am very grateful to my co-guide **Dr. M.V.Kartikeyan**, for his guidance and direction, immense encouragement and moral support to work in micro-/ millimeter wave domain. I am also very thankful to **Dr. Ankush Mittal** for encouraging me to work on this field, and for guiding me from time to time to proceed in the right direction. Special thanks to **Dr. S. N. Sinha,** Head of ECE Department, for facilitating the necessary requirements to carry out our work and for supporting us at critical juncture during the thesis submission.

I would also like to thank Dr. Narendra Chauhan for his constant support, without which I am sure my work might have hit a dead end, my colleague Ragini Jain for helping me a lot throughout my dissertation work and Mr. Ashwini Arya for his kind help in report writing.

I also acknowledge the computer center, Indian Institute of Technology Roorkee and NVIDIA Corporation, for the use of NVIDIA graphics card resource that has contributed to this research.

I take this opportunity to extend my sincere thanks to all my friends at IIT Roorkee for the constant support throughout my stay here. I am highly grateful to my parents and my brother for their love and blessings which always motivated me to work better. Finally I thank god for being kind to me and driving me through this journey.

[illegible]

(Divya Goel)

Abstract

Evolutionary Algorithm (EA) possesses several characteristics that are desirable to solve real world optimization problems up to a required level of satisfaction. Multiobjective Evolutionary Algorithms (MOEAs) are designed with regard to two common goals, fast and reliable convergence to the Pareto set and a good distribution of solutions along the front. In his work, evolutionary algorithms based approaches for multi-objective optimization have been studied. The particle swarm optimization has been studied in detail. The Particle Swarm Optimization (PSO) is a stochastic, population-based algorithm for search and optimization from a multidimensional space.

In this dissertation, multi-objective particle swarm optimization has been implemented for two problem domains. The first problem domain is: designing and optimizing the micro-/millimeter wave components, in this optimal design of two microstrip antenna, (proximity coupled dual-frequency and compact triple-band) and the optimized design of non-linear tapper has been presented. The second problem domain is: association rule mining, in this rules have been generated for two market basket type database (randomly generated and Mondrian foodmart dataset) using multi-objective particle swarm optimization. We have also parallelized the multi-objective particle swarm optimization on GPU for benchmark problem (DTLZ6) and real life problem (association rule mining) and the speed up in running time for both the problems have been presented.

The code for optimization is implemented on MATLAB 2006b, and we have used the EM simulator IE3D for antenna design. The experimental platform used for parallelization is based on Intel(R) Xeon(R) CPU E5420 @ 2.50 GHz, 2.49 GHz, 16.0 GB RAM, NVIDIA Quadro FX 3700, and Windows XP (x64), and we have used visual studio 2005 for the sequential and parallelization code.

List of Figure

List of Table

1.1 General Introduction

For the past few decades, engineering design and optimization [1] have been important areas of research. Its goal is not only to achieve a feasible design, but also to achieve the design objectives. In engineering design activities, the design objective could be to minimize the cost of production or to maximize the efficiency of production. When multiple objectives are present, the optimization problem is called a multi-objective optimization (MO) problem. An optimization algorithm is a procedure which is executed iteratively by comparing various solutions till the optimum or a satisfactory solution is found [1].

A significant portion of research and application in the field of optimization considers a single objective, although most real-world problems involve several objectives which conflicts with each other (such as simultaneously minimizing the cost of fabrication and maximizing product reliability). Since no one solution can be termed as an optimum solution to multiple conflicting objectives, the resulting multi-objective optimization problem (MOP) resorts to a number of trade-off optimal solutions [2].

Classical optimization methods can at best find one solution in one simulation run and are susceptible to the shape of the Pareto front. So, these methods are inconvenient to solve multi-objective optimization problems.

The evolutionary algorithm (EA) [3] stands for a class of stochastic optimization methods that simulate the process of natural evolution. EA overcomes the issues of classical mathematical programming techniques. They can simultaneously deal with a set of possible solutions which allows us to find several members of the Pareto optimal set in a single run of the algorithm and they are less susceptible to the shape or continuity of the Pareto front [4].

Multi-objective Evolutionary Algorithms (MOEA) have become increasingly popular in a wide variety of applications engineering, industrial, scientific [4, 5]. MOEAs are used in engineering applications like Aerodynamic Design Optimization, electromagnetic devices optimization, optimizing groundwater monitoring networks, city and regional planning; in scientific applications like medicines (for medical image processing, computer-aided diagnosis, treatment planning, and data mining etc), to identify interesting qualitative features in biological sequences; in industrial applications like Supply Chain Management, cellular manufacturing, design of fluid power systems etc.

Various MOEA's have been proposed during the last two decades. Genetic Algorithm is one of the most famous methods and used frequently in Evolution computation field. There also exist many bio-inspired heuristics for multi-objective optimization and different evolution-based multi-objective evolutionary algorithm. The most important among them are the particle swarm optimization (PSO) and differential evolution, whose use has become increasingly popular in multi-objective optimization.

In this work PSO technique is used for optimization. PSO [6] is one form of swarm intelligence inspired by social behavior of bird flocking or fish schooling. The PSO system consists of population (swarm) of potential solutions called particles. Each particle in the swarm adjusts its position in the search space based on the best position it has found so far as well as the position of the known best-fit particle of the entire swarm, and finally converges to the global best point of the whole search space.

In recent years, PSO has been used increasingly as an effective technique for solving complex and difficult optimization problems in practice. PSO has been successfully applied to problems such as function optimization, artificial neural network training, fuzzy system control, blind source separation, machine learning and so on.

Compared to genetic algorithm, PSO has the advantage of easy implementation and has few parameters to adjust, while maintaining strong abilities of convergence and global search.

In spite of those advantages, PSO still needs a long time to find solutions for large scale problems, as it requires a large number of fitness evaluations, which are usually done in a sequential way on CPU. A promising approach to overcome this limitation is to parallelize these algorithms [7].

In recent years, Graphics Processing Unit (GPU) which has traditionally been a graphics-centric workshop has shifted its attention to the non-graphics and general-purpose computing applications. Because of its parallel computing mechanism and fast float-point operation, GPU has shown great advantages in scientific computing fields.

In order to perform general-purpose computing on GPU more easily and conveniently, some platforms have been developed, such as Brook GPU (Stanford University), CUDA (Compute Unified Device Architecture, NVIDIA Corporation) [8]. These platforms have greatly simplified programming on GPU.

1.2 Motivation

Evolutionary algorithm offers various advantages for the design applications, which are as follows:

- The requirement/necessity of the exclusive domain specific knowledge is reduced.
- These methods are adaptive and scalable, so, they can be applied to the design applications of many engineering disciplines.
- They can handle many design constraints, variables and objectives, simultaneously.
- They can avoid the chance of getting the local optima.
- They can easily be interfaced with electromagnetic (EM) simulators, due to which the laborious task of optimizing design parameters can be converted to computer simulations.

Due to the above advantages and enormous interest for PSO in latest research, we have worked on multi-objective particle swarm optimization (MOPSO) in this dissertation work.

1.3 Problem Statement

The problem undertaken in this dissertation can be divided as:

- To study the evolutionary algorithms for multi-objective optimization.
- To use Particle Swarm Optimization (PSO) for optimization of micro-/millimeter wave components and data mining problem.

- To parallelize the optimized benchmark (DTLZ6) and data mining problems on CUDA for reducing the computation time.

The problems that we have taken for optimization are, proximity coupled dual-frequency microstrip antenna, compact triple-band microstrip antenna, nonlinear taper and association rule mining.

1.4 Organization of the Report

The organization of this dissertation report is as follows:

Chapter 2 gives the basics of evolutionary algorithm and its approaches for multi-objective optimization. Further this chapter discusses the particle swarm optimization and its techniques for multi-objective optimization. The algorithm for multi-objective particle swarm optimization, that we have used for our implementation in other chapters have also been presented in this chapter.

Chapter 3 discusses the multi-core architecture of CUDA programming environment, which we have been used in this dissertation work.

Chapter 4 starts with the brief introduction of association rule mining and then give multi-objective particle swarm optimization approach for rule mining.

Chapter 5 starts with the brief introduction of microstrip antenna and then gives the optimal design for three problems using multi-objective particle swarm optimization.

Chapter 6 describes the parallel implementation of multi-objective particle swarm optimization and compares the performance for benchmark problem (DTLZ6) and real life problem (association rule mining) on CPU and GPU.

Chapter 7 concludes the dissertation report and gives suggestion for future work.

Chapter 2 Evolutionary Algorithm Based Approaches for Solving Multi-Objective Problem (MOP)

2.1 Concept of Evolutionary Algorithm

An Evolutionary algorithm is characterized by three features:

1. A set of solution candidates.
2. A mating selection process is performed on this set.
3. Several solutions may be combined in terms of recombination to generate new solutions.

The solution candidates are called individuals and the set of solution candidates is called the population. Each individual represents the encoded form of possible solution, i.e., a decision vector, to the problem at hand.

The mating selection process usually consists of two stages: fitness assignment and sampling. In the first stage, the individuals in the current population are evaluated in the objective space and then assigned a scalar value, the fitness, reflecting their quality. Afterwards, the mating pool is created by random sampling from the population according to the fitness values.

Then, the variation operators are applied to the mating pool. With EAs, there are usually two of them, namely the recombination and the mutation operator. Finally, environmental or survivor selection determines which individuals of the population (we can use the latter set as the new population or can combine both sets and deterministically choose the best individuals for survival) and the modified mating pool form the new population [3]. The general scheme of Evolutionary Algorithm is shown as a flow chart in Figure 2.1.

Multi-objective Evolutionary algorithm (MOEA) can yield a whole set of potential solutions, which are all optimal in some sense. The two fundamental goals in MOEA design are - guiding the search towards the Pareto set and keeping a diverse set of non-dominated solutions.

The first goal is assigning scalar fitness values to the individuals in the presence of multiple optimization criteria. The second goal concerns selection in general as we want to avoid that the population contains mostly identical solutions (with respect to the objective space and the decision space). Finally, a third issue which addresses both of the above goals is elitism, i.e., the question of how to prevent non-dominated solutions from being lost.

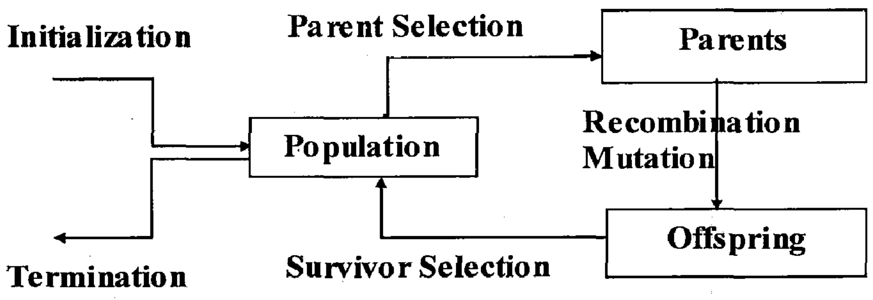


Figure 2.1: The general scheme of Evolutionary Algorithm as a flow chart [9]

2.2 Multi-Objective Optimization

Multi-objective optimization problem (MOP) is defined as follows: Minimizing/maximizing m objectives.

$$\text{Find: } \overline{x} = (x_1, x_2 \ldots x_n)$$
$$\text{Min/ Max: } \overline{y} = F(\overline{x}) = (f_1(\overline{x}), f_2(\overline{x}) \ldots fm(\overline{x}))$$
$$\text{Subject to: } g_j(\overline{x}) \leq 0; j = 0, 1, 2 \ldots k$$
$$h_l(\overline{x}) = 0; l = 0, 1, 2 \ldots e$$

Where n is the number of design variables, m the number of objective functions, k the number of inequality constraints, and e is the number of equality constraints. $\overline{x} = (x_1, x_2 \ldots x_n) \in X$ is an n-dimensional decision variable vector and X is the decision variable space. $\overline{y} = (y_1, y_2 \ldots y_n) \in Y$ is an m-dimensional objective vector and Y is the objective space.

A decision vector $\overline{u} \in X$ is said to strictly dominate another decision vector $\overline{v} \in X$ denoted by $\overline{u} \prec \overline{v}$, if and only if (iff) $\forall i \in \{1, \ldots m\}: f_i(\overline{u}) \leq f_i(\overline{v})$ and $\exists j \in \{1, \ldots m\}: f_i(\overline{u}) < f_i(\overline{v})$. A decision vector $\overline{x} \in X$ is said to be **Pareto optimal** with respect to X iff there is no other decision vector that dominates $\bar{x}$ *in X*. The set of all Pareto optimal solutions in the decision variable space is called **Pareto optimal set** and the corresponding set of objective vector is called **Pareto optimal front.** The example of Pareto front for bi-objective space is shown in Figure 2.2.

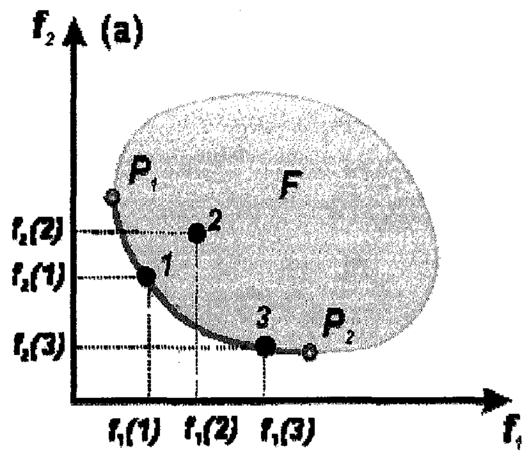


Figure 2.2: Example of a bi-objective space (f1, f2). We assume a minimization prob. The Pareto front is the boundary between the points P1 and P2 of the feasible set F.

2.3 Classification of Multi-objective Evolutionary Algorithm

There are several possible ways to classify MOEAs. The following classification is based on the type of selection mechanism adopted as given by Carlos A. Coello in [5]:

1. Aggregating Functions
2. Population-based Approaches
3. Pareto-based Approaches

2.3.1 Aggregating Functions

The most straightforward approach to deal with multi-objective problems is to combine them into a single scalar value (e.g., adding them together). These techniques are known as "aggregating functions", because they combine (or "aggregate") all the objectives of the problem into a single one. Aggregating functions are underestimated by MOEA researchers mainly because of the limitation of linear aggregating functions (i.e., they cannot generate non-convex portions of the Pareto front regardless of the weight combination used).

2.3.2 Population based Approaches

In this approach, the population of an EA is used to diversify the search, but the concept of Pareto dominance is not directly incorporated into the selection process. The classical example of this sort of approach is the Vector Evaluated Genetic Algorithm (VEGA), proposed by Schaffer. VEGA basically consists of a simple genetic algorithm with a modified selection mechanism. At each generation, a number of sub-populations are generated by performing proportional selection according to each objective function in turn. Thus, for a problem with *k* objectives, *k* sub-populations of size ***M/k*** each are generated (assuming a total population size of *M*). These sub-populations are then shuffled together to obtain a new population of size *M*, on which the genetic algorithm applies the crossover and mutation operators.

VEGA has several problems, from which the most serious is that its selection scheme is opposed to the concept of Pareto dominance. For example, there is an individual that encodes a good compromise solution for all the objectives (i.e., a Pareto optimal solution), but it is not the best in any of them, it will be discarded. Schaffer suggested some heuristics to deal with this problem. One interesting aspect of VEGA is that despite its drawbacks it remains in current use by some researchers mainly because it is appropriate for problems in which we want the selection process to be biased and in which we have to deal with a large number of objectives.

2.3.3 Pareto based Approaches

In this approaches, we consider MOEAs that incorporate the concept of Pareto optimality in their selection mechanism. A wide variety of Pareto-based MOEAs have been proposed in the last few years. Some of the Pareto-based approaches of Genetic Algorithm (GA) as given by Carlos A. Coello in [5, 10] are as follows:

1. **Goldberg's Pareto Ranking:** Goldberg [11] suggested moving the population toward Pareto Front, by using a selection mechanism that favors solutions that are non-dominated with respect to the current population. The basic idea is to find the set of individuals in the population that Pareto non-dominated by the rest of the population. These individuals are then assigned the highest rank and eliminated from

further contention. Another set of Pareto non-dominated individuals are determined from the remaining population and are assigned the next highest rank. This process continues until the population is suitably ranked. He also suggested the use of some kind of niching technique to keep the GA from converging to a single point on the Pareto Front.

2. **Multi-Objective Genetic Algorithm (MOGA):** Fonseca and Fleming [12] proposed a ranking approach different from Goldberg's scheme. In this case, each individual in the population is ranked based on how many other points dominate them. All the non-dominated individuals in the population are assigned the same rank as 1 and obtain the same fitness, so that they all have the same probability of being selected. MOGA uses a niche-formation method in order to diversify the population.

3. **Non-dominated Sorting Genetic Algorithm (NSGA):** This method was proposed by Srinivas and Deb [13], and is based on several layers of classifications of the individuals as suggested by Goldberg [11]. Before selection is performed, the population is ranked on the basis of non-domination: all non-dominated individuals are classified into one category with a dummy fitness value, which is proportional to the population size, to provide an equal reproductive potential for these individuals.

   An offshoot of this approach, the NSGA-II [14], uses elitism and a crowded comparison operator that ranks the population based on both Pareto dominance and region density. This crowded comparison operator makes the NSGA-II considerably faster than its predecessor while producing very good results.

4. **Niched Pareto Genetic Algorithm (NPGA):** Horn and Nafpliotis [15] proposed a tournament selection based on Pareto dominance. Two members of the population are chosen at random and they are each compared to a subset of the population. If one is non-dominated and the other is not, then the non-dominated one is selected. If there is a tie (both are either dominated or non-dominated), then fitness sharing decides the tourney results.

The Genetic Algorithm based on different approaches has been discussed above. There also exist many bio-inspired heuristics for multi-objective optimization and different evolution-based multi-objective evolutionary algorithm. The most important among them are the particle swarm optimization (PSO) and differential evolution. However, other bio-inspired algorithms such as artificial immune systems and ant colony optimization have also been used to solve multi-objective optimization problems.

2.4 Particle Swarm Optimization (PSO)

Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Eberhart and Kennedy [6], inspired by social behavior of bird flocking or fish schooling. The high convergence speed and relative simplicity of PSO motivate researchers to solve multi-objective problems by using PSO.

The PSO system consists of a population (swarm) of potential solutions called particles, fly through the problem space by following the current optimum particles. Each particle maintains a memory which helps it in keeping the track of its coordinates in the problem space which are associated with the best solution (fitness) it has achieved so far.

Particle Swarm has two operators: Velocity update and Position update. During each generation, each particle is accelerated toward the particles previous best position and the global best position.

2.4.1 PSO: Algorithm

PSO is initialized with a group of random particles (solutions) and then searches for optima by updating generations. Each particle has a position $\overline{X} = (x_1, x_2 \ldots. x_D)$ and a velocity $\overline{V} = (v_1, v_2 \ldots. v_D)$ in a variable space. In generation $t+1$, the velocity and the position are updated as follows:

$$v_{id}^{t+1} = \omega v_{id}^{t} + c_1 r_1 (p_{id}^{t} - x_{id}^{t}) + c_2 r_2 (p_{gd}^{t} - x_{gd}^{t}) \quad \ldots\ldots (1)$$

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \quad \ldots\ldots (2)$$

where $i=\{1,\ldots..N\}$ and N is the population size, $d=\{1, \ldots..D\}$ and D is the dimension of search space, ω is the inertia weight, c_1 and c_2 are two positive constants, r_1 and r_2 are random values in the range [0, 1], p^t_{id} is the personal best of the particle i in generation t, p^t_{gd} is the global best of population in generation t.

The performance of each particle is evaluated according to a pre-defined fitness function, which is related to the problem concerned.

2.5 Multi-Objective Particle Swarm Optimization (MOPSO)

Particle Swarm optimization (PSO) [6] was originally designed for solving single objective optimization problems. The initial population of particles is initialized with random solutions. For every generation, each solution moves toward the global Pareto front by updating its velocity, the best solution a particle has achieved so far and follows the best solution achieved among the population of solutions.

2.5.1 Some Techniques

PSO-based multi-objective optimization has following technique to get local best particle or global particle as given in [16]:

1. **Aggregating method:** Parsopoules K E [17] first took PSO solving Multi-objective problem using fixed weight, adaptive weight and vector evaluation method. Weighted-vector method often can't get appropriate weights for special optimization problem. Vector evaluation method always couldn't provide satisfied solutions for Multi-objective Optimization (MO) problems.
2. **Pareto-based method:** Ray T and Liew K M [18] combined Pareto dominance with PSO to solve multi-objective problems. They got a set of elitist solutions by Pareto-dominance, from which a global optimal particle was chosen by roulette. The approach uses a nearest neighbor density estimator to promote diversity.
3. **Sigma method:** Mestaghim S [19] selected global best particle (or local best particle) according to the distance between current solutions and archive solutions. Sigma method began with the initialization of a number of solutions. If difference between fitness of the initial solutions was small, it would lead to selection press inadequate and PSO algorithm convergence slowly.
4. **Dynamic neighborhood method:** Hu X and Eberhart R [20] defined one of objective as optimization objective, other objective as fixed objective. Each

particle has different neighbors in each generation. This method is sensitive about the order of optimization objective and neighbor objective.

5. **Multi-population method:** Konstantinos E.Parsopoulos et al [21] divided population into many subpopulations, and each subpopulation executes PSO independently according to one of the objective function. Every subpopulation exchange information with each other to obtain Pareto-optimal solution. Great number of particles increased computational time.
6. **Li [22]** incorporated the main mechanisms of NSGA-II into PSO. In his algorithm, the population of the particles was combined with the personal best position, and the best was selected from this new population to compose the next population.
7. **C.R. Raquel, Prospero C. Naval, Jr. [23]** incorporated the concept of crowding distance into PSO and the distribution of non-dominated solutions was improved on the Pareto front.

In this work, the extended algorithm of the single-objective PSO to handle multi-objective optimization problems is implemented.

2.5.2 Algorithm

Step1. Initialize population size, the archive size and the maximum number of iterations according to the problem concerned. The positions are initialized randomly and velocities are initialized to zero.

Step2. The particles are evaluated for all the objectives of the problem. The non-dominated particles (i.e. Pareto optimal solutions) are stored in the archive. The initial value of the personal best (pbest) for each particle is equal to its initial position and the global best (gbest) is initialized to the best particle in the population.

Step3. Repeat for each iteration .

a) Randomly select global best from the archive.

b) Update the new velocity and position of the particles according to equations (1) & (2).

c) If particles position goes beyond the boundaries, then it is reintegrated by having the decision variable take the value of its corresponding lower or upper boundary and its velocity is multiplied by -1 so that it searches in the opposite direction.

d) Compute fitness values of the newly generated swarm.

e) Insert all new non-dominated solution in population into the archive if they are not dominated by any of the stored solutions. All dominated solutions in the archive by the new solution are removed from the archive.

f) Update the personal best solution of each particle in population. The personal best is chosen as the best solution among its new position and its personal best.

g) Increment iteration counter by one.

Step4. Until maximum number of iterations are reached.

In this algorithm the mechanism of crowding distance and mutation can be incorporated. The crowding distance computation can be used on global best selection and in the deletion method of an external archive of non-dominated solutions. The crowding distance mechanism together with a mutation operator maintains the diversity of non-dominated solutions in the external archive and results in better convergence towards the Pareto front [23].

The crowding distance value of a solution provides an estimate of the density of solutions surrounding that solution. Crowding distance is calculated by first sorting the set of solutions in ascending objective function values. The crowding distance value of a particular solution is the average distance of its two neighboring solutions.

Chapter 3 nVidia CUDA architecture

Multicore architecture is the solution for the ever growing demand of more and more computational power. As the trend is growing in the market more and more varieties of multicore chips are developed. In this work we have used NVidia CUDA architecture to attain performance improvement over sequential machines. Let us see this architecture in detail.

3.1 GPU

The GPU refers to the commodity off-the-shelf 3D Graphics Processing Units, which are specifically designed to be extremely fast at processing large graphics data sets for rendering tasks. General purpose computing on the GPU is an active area of research.

The programmable GPU has evolved into a highly parallel, multithreaded; many core processor with tremendous computational horsepower and very high memory bandwidth. Its capabilities have increased dramatically in the past few years and the current generation of GPUs has higher floating point performance than the most powerful (multicore) CPUs [8].

The reason behind the discrepancy in floating-point capability between the CPU and the GPU is that the GPU is specialized for compute-intensive, highly parallel computation and therefore designed such that more transistors are devoted to data processing rather than data caching and flow control, as schematically illustrated by Figure 3.1. More specifically, the GPU is well suited to address problems that can be expressed as data parallel computations [8].

The GPU contains hundreds of cores that work great for parallel implementation. The programming is done in SIMD style where same code is worked on different data locations. Until recently a graphics API was needed to code on GPUs which made coding for non graphics oriented calculations tough. Trying to work around this limitation nVidia released CUDA which allows GPUs to be programmed using a variation of C.

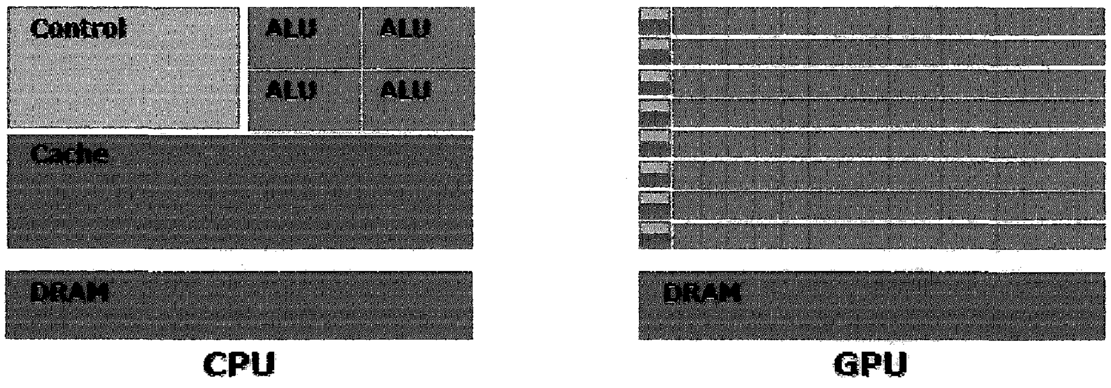


Figure 3.1: Transistor division in CPU and GPU [8]

3.2 CUDA

CUDA (or Compute Unified Device Architecture) is a parallel programming model and software environment developed by Nvidia [8]. It was designed as a middle-ware to allow application software that transparently scales its parallelism on GPU. The core concepts involved with CUDA are a hierarchy of thread groups, shared memories, and barrier synchronization. The thread hierarchy allows user to divide his task in a similar hierarchy, where coarse sub-problems can be solved independently and finer pieces that can be solved cooperatively in parallel using shared memory. CUDA achieves all this using a minimal extension to C thus maintaining a low learning curve for programmers already familiar with the standard programming language.

3.3 Programming Model

CUDA extends C by allowing the programmer to define C functions, called *kernels*, that, when called, are executed N times in parallel by N different *CUDA threads*, as opposed to only once like regular C functions.

A kernel is defined using the **__global__** declaration specifier and the number of CUDA threads for each call is specified using a new <<<...>>> syntax:

```
// Kernel definition
__global__ void vecAdd(float* A, float* B, float* C)
{
int i = threadIdx.x;
C[i] = A[i] + B[i];
}
```

```
int main()
{
// Kernel invocation
vecAdd<<<1, N>>>(A, B, C);
}
```

Each of the threads that execute a kernel is given a unique *thread ID* that is accessible within the kernel through the built-in **threadIdx** variable. In the above kernel definition, each of the threads that execute vecAdd() performs one pair-wise addition.

3.3.1 Thread Hierarchy

The batch of threads that executes the Kernel is organized as a grid of thread blocks, so that the total number of threads is equal to the number of threads per block times the number of blocks. These multiple blocks are organized into a one-dimensional or two-dimensional grid of thread blocks as illustrated by Figure 3.2.

Threads within a block can cooperate among themselves by sharing data through some shared memory and synchronizing their execution to coordinate memory accesses. Such synchronization is possible by means of a programming primitive __syncthreads() as exposed by CUDA API. This serves as barrier synchronization. The number of threads per block is restricted by the limited memory resources of a processor core. On NVIDIA Tesla architecture, a thread block may contain up to 512 threads.

Each thread is identified by its thread ID. The index of a thread and its thread ID relate to each other in a straightforward way: For a one-dimensional block, they are the same; for a two-dimensional block of size *(Dx, Dy)*, the thread ID of a thread of index *(x, y)* is *(x + y Dx)*; for a three dimensional block of size *(Dx, Dy, Dz)*, the thread ID of a thread of index *(x, y, z)* is *(x + y Dx + z Dx Dy)*.

In addition to the variable threadIdx, CUDA also have a few other built-in variables namely gridDim, blockIdx and blockDim. The gridDim gives the dimension of grid, i.e the number of blocks within the grid. The blockIdx variable gives the index of the thread's parent block within the grid, and blockDim which gives the number of threads per block. The gridDim and blockDim are being supplied in the call to kernel as the first and second parameter respectively, to the <<<>>> syntax.

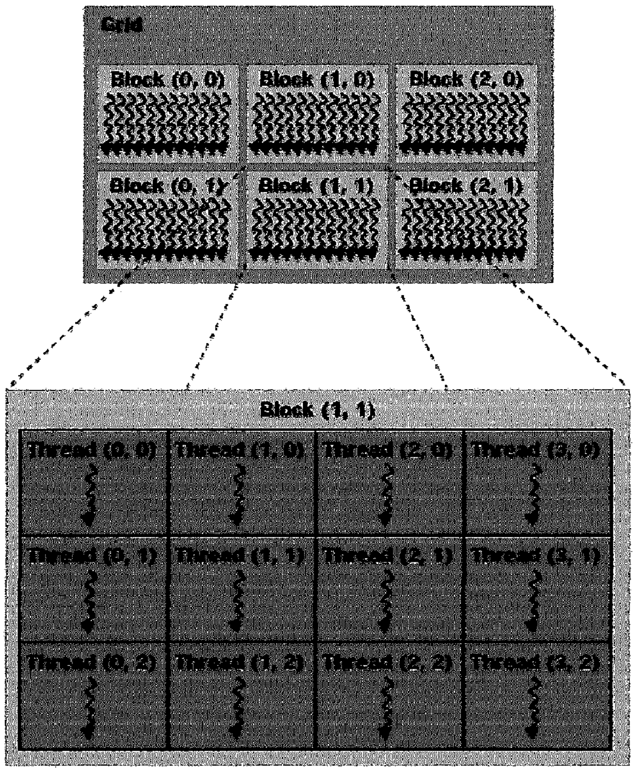
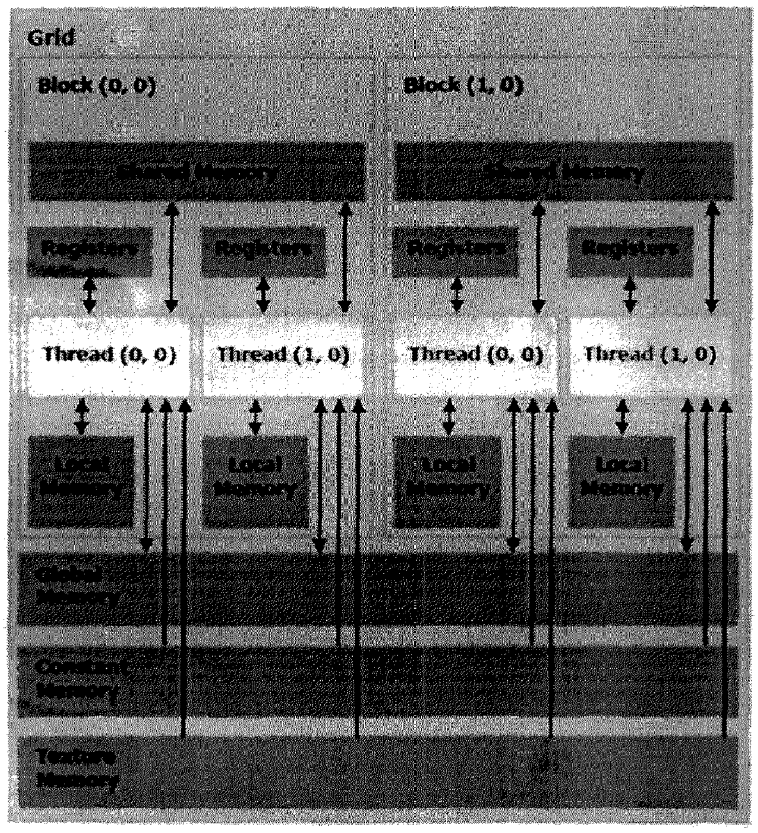


Figure 3.2: Thread Hierarchy in CUDA [8]

3.3.2 Memory Hierarchy

CUDA threads may access data from multiple memory spaces during their execution as illustrated by Figure 3.3. Each thread has a private local memory. Each thread block has a shared memory visible to all threads of the block and with the same lifetime as the block. Finally, all threads have access to the same global memory. There are also two additional read-only memory spaces accessible by all threads: the constant and texture memory spaces. The global, constant, and texture memory spaces are persistent across kernel launches by the same application. In this work, we have used local, shared and global memory for our implementation.

Memory management at runtime on the GPU RAM is done using CUDA API equivalents. The general procedure is to allocate memory on both host and device RAM, using cudaMalloc function call for the device memory. The data contents are copied from host memory to device memory using cudaMemcpy function. Writing data directly onto device memory from CPU code is not possible. The kernel calls are then made to do appropriate processing on the data. The processed data contents are copied back from the device to the host using cudaMemcpy function.



Fiugre.3.3: Memory Model of CUDA

3.4 GPU Architecture

The Tesla architecture is one of the architectures of Nvidia which support CUDA. It is built around a scalable array of multi-threaded Streaming Multiprocessors (SMs). When a CUDA program on the host CPU invokes a kernel grid, the blocks of the grid are enumerated and distributed to multiprocessors with available execution capacity. The threads of a thread block execute concurrently on one multiprocessor. As thread blocks terminate, new blocks are launched on the vacated multiprocessors.

A multiprocessor consists of eight Scalar Processor (SP) cores, two special function units for transcendentals, a multithreaded instruction unit, and on-chip shared memory. The multiprocessor creates, manages, and executes concurrent threads in hardware with zero scheduling overhead.

To manage hundreds of threads running several different programs, the multiprocessor employs a new architecture we call SIMT (single-instruction, multiple-thread). The multiprocessor maps each thread to one scalar processor core, and each scalar thread executes independently with its own instruction address and register state. The multiprocessor SIMT unit creates, manages, schedules, and executes threads in groups of 32 parallel threads called warps. Individual threads composing a SIMT warp start together at the same program address but are otherwise free to branch and execute independently.

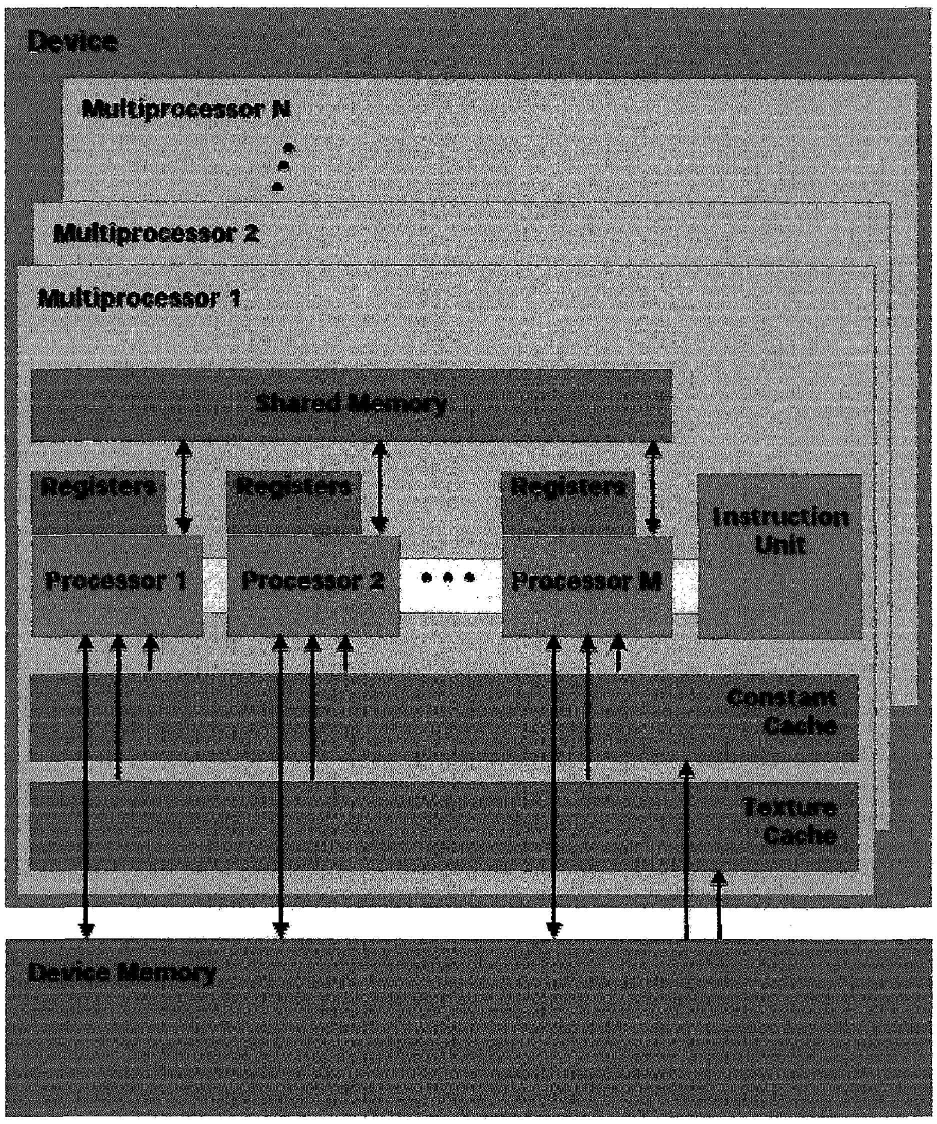


Figure 3.4: GPU Hardware model [8]

When a multiprocessor is given one or more thread blocks to execute, it splits them into warps that get scheduled by the SIMT unit. The way a block is split into warps is always the same; each warp contains threads of consecutive, increasing thread IDs with the first warp containing thread 0. Every instruction issue time, the SIMT unit selects a warp that is ready to execute and issues the next instruction to the active threads of the warp. A warp executes one common instruction at a time, so full efficiency is realized when all 32 threads of a warp agree on their execution path. If threads of a warp diverge via a data dependent conditional branch, the warp serially executes each branch path taken; disabling threads that are not on that path, and when all paths complete, the threads converge back to the same execution path. Branch divergence occurs only within a warp; different warps execute independently regardless of whether they are executing common or disjointed code paths. The GPU hardware model is shown in Figure 3.4.

Chapter 4 MOPSO for Association Rule Mining

4.1 Introduction

With the rapid growth in size and number of available databases, mining for knowledge, regularities or high-level information from data became essential to support decision-making and predict future behavior [24, 25]. Data mining techniques, used for achieving the above goals, can be classified into the following categories: classification, clustering, association rule mining, sequential pattern analysis, prediction, data visualization etc. [24, 25, 26].

Association rule mining is one of the important tasks of data mining intended towards decision support. It is the process of finding some relations among the attributes/attribute values of a huge database. These relationships can be represented as an IF–THEN statement. IF <some conditions are satisfied> THEN <predict some values of other attribute(s)>. The conditions associated in the IF part is termed as Antecedent, 'A' and those with the THEN part is called the Consequent, 'C'. So, symbolically we can represent this relation as $A \rightarrow C$ and each such relationship that holds between the attributes of records in a database fulfilling some criteria are termed as an association rule.

The association rule mining first discovers all the frequent patterns (set of items) and then constructs the rules from such patterns. Commonly used example is in market basket analysis, where an association rule $A \rightarrow C$ means if consumer buys the set of items A, then he/she probably also buys items C. These items are typically called as itemsets.

Different optimization methods for association rule mining have been proposed [27, 28]. The process is too resource-consuming, especially when there is not enough available physical memory for the whole database. A solution to encounter this problem is to use evolutionary algorithms, which reduce both cost and time of rule discovery.

Many existing algorithms visualize rule-mining as single objective problem, in which they try to measure the quality of generated rule by considering only one evaluation criterion, i.e., confidence factor or predictive accuracy. This criterion evaluates the rule depending on the number of occurrence of the rule in the entire database, and can have certain limitations.

1. The generated rule may have a large number of attributes involved in the rule thereby making it difficult to understand.
2. These algorithms may extract some rules from the data that can be easily predicted by the user.
3. These algorithms do not give any importance towards the rare events, i.e., interesting rules.

In this dissertation work, we have visualized association rule-mining as multi-objective problem and has taken four objectives to evaluate/visualize the rules as in [29], Support, Confidence, Comprehensibility and Interestingness.

4.2 Problem Statement

We have used multi-objective particle swarm optimization (MOPSO) algorithm, which is defined in chapter 2, for mining the rules of market basket dataset. Market basket dataset consist of data/ transactions generated as a result of customer purchases of items from a supermarket. Our objective is to minimize the time for obtaining the rules.

4.2 MOPSO Approach in Rule Mining

Evolutionary multiobjective (EMO) techniques in rule mining can be roughly categorized into two approaches [30]. In one approach, each rule is evaluated according to multiple rule evaluation criteria such as support and confidence. An EMO algorithm is used to search for Pareto-optimal rules. In the other approach, each rule set is evaluated according to multiple rule set evaluation criteria such as accuracy and complexity. An EMO algorithm is used to search for Pareto-optimal rule sets.

In this chapter, we have considered the first approach for rule mining. Each particle of the swarm represents the rule, $A \rightarrow C$ and each dimension, d of the particle represents each attribute/item of the dataset respectively [29, 31]. Firstly, all particle positions are initialized for D-dimension space with the random generation of values from 0 to 3, where each value in the particle/ rule, $A \rightarrow C$ is represented as:

The attribute is in the antecedent (A) part of the rule, if the value is 0
The attribute is in the consequent (C) part of the rule, if the value is 3
The attribute does not exists in the rule, if the value is 1or 2

With the above random generation of particle position values, we will obtain the values for the particles in D-dimension space. So, if the dataset has 6 attributes (A, B, C, D, E, F), then the particle sequence in 6-dimension space can be (0, 1, 3, 0, 2, 1), which would mean AD→C. For each rule/particle, the database is scanned to count the support (A), support(C), support (A∪C), and four objectives Support, Confidence, Comprehensibility and Interestingness are calculated as:

$$\text{Support} = (\text{support } (A \cup C)) / (n)$$

$$\text{Confidence} = (\text{support } (A \cup C)) / (\text{support } (A))$$

$$\text{Comprehensibility} = \log(1 + \text{no of attribute}(C)) / \log(1 + \text{total no of attribute})$$

$$\text{Interestingness} = (\text{Confidence}) * ((\text{support } (A \cup C) / (\text{support}(C))) * (1\text{-Support})$$

where *n* is number of records in the dataset, ***no of attribute (A)*** is the number of attributes in the particle which are Antecedent (A), ***total no of attribute*** is the number of attributes in the particle which are either Antecedent (A) or Consequent (C). The algorithm for support count in fitness function of MOPSO is shown below.

Algorithm for Support Counting in MOPSO

Input: An association rule (particle) A→C

Output: Support count of antecedent, consequent and the rule

```
1: for each transaction T in database
4:      if (antecedent occurs in T)
5:           increment antecedent support
6:      end if
7: end for
8: Use same algorithm to count the support of consequent (C) and the whole rule (A∪C)
```

Then the best particles (Pareto set) found based on the fitness function, which is the problem's objective function (i.e. based on above four objectives), are inserted in the archive. After that, the iterative process initiates. The particle's velocity and position, on the next iteration, are calculated by the equations (1) and (2), as mentioned in chapter 2, but with the slight modification due to boundary conditions (the dimension values of each particle should be within (0,3)). The modified form of equations (1) and (2) are shown below as equation (3) and (4) respectively.

$$v_{id}^{t+1} = (\omega v_{id}^{t} + c_1(p_{id}^{t} - x_{id}^{t}) + c_2(p_{gd}^{t} - x_{gd}^{t}))\%4 \quad \text{.........} (3)$$

$$x_{id}^{t+1} = ((x_{id}^{t} + v_{id}^{t+1}))\%4 \quad \text{.........} (4)$$

where $w=1$ and $c1 = c2 = 2$. As an example, let's consider the particle position as (0, 1, 3, 0, 2, 1), initial velocity (0,0,0,0,0,0), particle's best position (0, 1, 3, 0, 2, 1) and global best particle position as (0,0,1,3,2,1), then its modified velocity and positions in next iteration is:

$v = ((0, 0, 0, 0, 0, 0) + 2 \times ((0, 1, 3, 0, 2, 1) - (0, 1, 3, 0, 2, 1)) + 2 \times ((0, 0, 1, 3, 2, 1) - (0, 1, 3, 0, 2, 1)))\ \%\ 4$

$v = (0, 2, 0, 2, 0, 0)$

$x = ((0, 1, 3, 0, 2, 1) + (0, 2, 0, 2, 0, 0))\ \%\ 4$

$x = (0, 3, 3, 2, 2, 1)$

Then the fitness function of the new particle (0, 3, 3, 2, 2, 1), is calculated on the basis of four objectives and algorithm proceeds in the same manner as mentioned in chapter-2 till the stopping criteria is reached.

Total number of comparisons in this method is the number of transactions in dataset multiplied by the number of itemsets generated by MOPSO.

$$\text{Number of contrasts} = \text{transactions} \times \text{itemsets}$$

In the above method, to evaluate each association rule A→C, the database is repeatedly scanned to compare to the whole database with A, B, and A∪B itemsets. Number of comparisons for each itemset is equal to number of transactions. To reduce the number of comparisons and running time, we can apply clustering on database.

If an itemsets occurs in a transaction, then minimum size of the transaction is the size of itemset. In the other words, to count the support count of an itemset, it must be compared with transactions, having their size greater or equal to the itemset [32]. In this method, an extra phase is processed before rule generations as shown below.

Algorithm for Clustering database [32]

Input: Data table containing D attributes/ columns where each column contains true (item occurs in transaction) or false (items does not occur in transaction).

Output: D arrays/tables where each array/table represents a cluster (Cluster_Table(s) represents the cluster, whose itemset size is's', where $s = \{1, 2 \ldots D\}$).

```
1: for each transaction T in database
2:     s = number of items (true values) in T
3:     Add T to Cluster_Table(s)
4: end for
```

Second step is to change the support counting of the above method. The MOPSO scans the whole transactions; while the Cluster Based MOPSO (CB-MOPSO) prevents some unnecessary comparisons. The algorithm for support count in CB-MOPSO is shown below.

Algorithm for Support Counting in CB-MOPSO [32]

Input: An association rule $A \rightarrow C$

Output: Support count of antecedent, consequent and the rule

```
1: s = number of items in antecedent (A)
2: for k = s to Max_size_of_transactions
3:        for each transaction T in Cluster_Table(k)
4:                if (antecedent occurs in T)
5:                increment antecedent support
6:                end if
7:        end for
8: end for
9: Use same algorithm to count the support of consequent (B) and the whole rule (A∪B)
```

Suppose $f(i)$ as number of generated itemsets having their size (number of items) equal to i. $s(i)$ is the size of cluster(i) (i.e. the number of transaction in i^{th}cluster). Number of comparisons for each itemset is shown in Equation (2). So, the total number of comparisons is:

$$TotalContrasts = \sum_{i=1}^{D} f(i) \sum_{j=i}^{D} s(j)$$

OR

$$TotalContrasts = \sum_{i=1}^{D} s(i) \sum_{j=1}^{i} f(j)$$

where D represents no. of attributes/ items/ columns in the database.

4.3 Results

We have implemented and compared the results of MOPSO and CB-MOPSO for rule mining on two datasets:

- A randomly generated dataset
- Mondrian Foodmart dataset [33]

In randomly generated dataset, we have generated 40,000 records with 40 attributes. In Mondrian foodmart dataset [33] we have 5,546 data records with 70 attributes, we have taken only consumable items as attributes. We have taken swam size as 1000 and number of iterations as 100, number of rules generated in randomly generated and Mondrian foodmart dataset with each case MOPSO and CB-MOPSO, is shown in Table 4.1.

TABLE.4.1 NUMBER OF RULES GENERATED IN RANDOMLY GENERATED AND MONDRIAN FOODMART DATASET WITH MOPSO AND CB-MOPSO

Dataset	Algorithm	No. of records	No. of attributes	Swarm size	No. of iteration	No. of rules generated
Randomly generated	MOPSO	40,000	40	1000	100	119
	CB-MOSO					108
Mondrian foodmart	MOPSO	5,546	70			20
	CB-MOPSO					18

We can see from Table 4.1, that number of rules generated in each case MOPSO and CB-MOPSO are approximately same. The difference in number of rule generated is due to the random generation of particles.

We have compared the results of MOPSO and CB-MOPSO on the basis of speedup and comparisons decrease, as shown in Table 4.2 and Table 4.3. The speedup and comparisons decrease are defined as:

$$\text{Speedup} = \frac{(\text{avg. time(MOPSO)- avg. time(CB-MOPSO))}}{(\text{avg. time(MOPSO))}} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (5)$$

$$\text{Comparison decrease} = \frac{(\text{avg. totalcomp.(MOPSO)- totalcomp.(CB-MOPSO))}}{(\text{avg. totalcomparison(MOPSO))}} \quad \ldots\ldots (6)$$

Table 4.2 and Table 4.3 shows the results from running both methods in randomly generated dataset and Mondrian foodmart dataset. The speed up has varied from 9.07% for randomly generated dataset to 73.74% for Mondrian foodmart dataset, this shows that speed up is greatly affected by the distribution of clusters.

TABLE.4.2 COMPARISON OF CBMOPSO AND MOPSO PERFORMANCE ON RANDOMLY GENERATED DATASETS

Algorithm	Avg. itemset size	Avg. time	Avg. total comparison	Speed up	Comparisions decrease
MOPSO	20.19	161976.3	4,000,000,000	9.07	44.7
CB-MOPSO		147282.5	2,210,676,643		

TABLE.4.3 COMPARISON OF CBMOPSO AND MOPSO PERFORMANCE ON MONDRIAN FOODMART DATASETS

Algorithm	Avg. itemset size	Avg. time	Avg. total comparison	Speed up	Comparisions decrease
MOPSO	35.01	28225.44	554,600,000	73.74	98.19
CB-MOPSO		7411.219	10,009,708		

Chapter 5 MOPSO for designing micro-/millimeter wave components

The antenna designs are carried out using commercially available electromagnetic (EM) simulation tools. The latest EM simulator use conventional optimization methods for optimization of design parameters. The difficulties with these local search methods are that they require a proper initial guess; otherwise the chances of getting local optimum solutions are very high. Moreover, they can handle only a few numbers of design parameter and design constraint.

Evolutionary optimization methods such as genetic algorithms (GAs) and particle swarm optimization (PSO) have been successfully used to solve electromagnetic problems; as these methods can be easily interfaced with EM simulators [34]. (Due to which the laborious task of optimizing design parameters can be converted to computer simulations)

Particle swarm optimization (PSO) has been introduced into the EM community in [35, 36]. The enormous interest in applying PSO technique to antenna designs is evident due to the wide range of practical problems that can be solved by this novel, nature-inspired, evolutionary algorithm [37]. PSO technique is easy to implement and has a few parameters to adjust, while maintaining strong abilities of convergence and global search.

5.1 Microstrip Antenna

Applications that require low-profile, light weight, easily manufactured, inexpensive, conformable antennas often use some form of a microstrip radiator. The microstrip antenna (MSA) [38] is a resonant structure that consists of a dielectric substrate sandwiched between a metallic conducting patch and a ground plane. The MSA is commonly excited using a microstrip edge feed or a coaxial probe. The patch is generally made up of metal like copper, gold etc. which can take any shape. The canonical forms of the MSA are the rectangular and circular patch MSAs. The rectangular patch antenna in Figure 5.1 is fed using a microstrip edge feed and the circular patch antenna is fed using a coaxial probe.

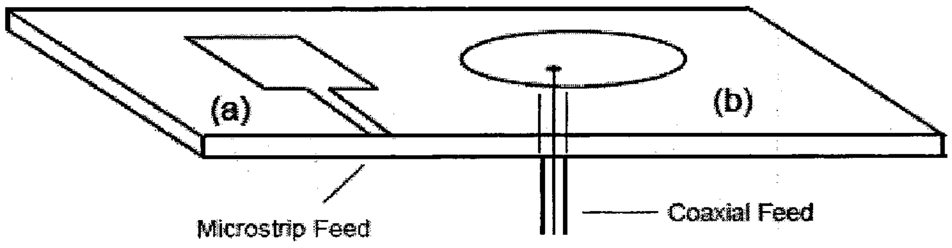


Figure.5.1 Rectangular patch antenna using a microstrip edge feed and the circular patch antenna using a coaxial probe feed.

For a good antenna performance, a thick dielectric substrate having a low dielectric constant is desirable since this provides better efficiency, larger bandwidth and better radiation. However, such a configuration leads to a larger antenna size. In order to design a compact microstrip patch antenna, higher dielectric constants must be used which are less efficient and result in narrower bandwidth. Hence a compromise must be reached between antenna dimension and antenna performance.

5.2 Experiments

In this chapter, we have presented the design and optimization of specific micro-/millimeter wave components using MOPSO. For this, we have taken three problems:

1. Design of Proximity Coupled Dual-Frequency Microstrip Antenna, to obtain the frequency response at GPS (1.575 GHz) and Bluetooth (2.4 GHz – 2.484 GHz) frequency bands of wireless communication with minimum return loss and maximum bandwidth [39].

2. Design of compact triple-band Microstrip Antenna, to obtain the frequency response at GSM (~900 MHz) and WLAN (~2.4 GHz and ~5.2 GHz). The design optimization has been carried out to obtain the minimum return loss at all three bands and maximize bandwidth for the first band.

3. The design and optimization of a nonlinear taper for a 200 KW, long-pulse strat-up gyrotron at 170 GHz has been chosen for this study. The operating mode is $TE_{24,8}$. The design optimization has been carried out to give the maximum

transmission in the operating mode with very less mode conversion. This gyrotron will serve as a start-up gyrotron for ITER or ITER-like machines.

In this work, MATLAB is used for implementation. In the design of antenna using PSO algorithm, the fitness function of PSO is interfaced with the EM simulator (IE3D). In this implementation, IE3D is invoked in the iterative loop of the optimization algorithm. The geometry is simulated for each particle using the IE3D in command (batch mode), in each iteration. After the simulation is over the resultant .sp file is interpreted and return loss, s_{11} parameter values for the desired frequencies, f_i are obtained from it and the bandwidth (BW) of the desired frequencies is calculated by reading the .sp file to get f_l and f_h.

$$BW(i) = f_h\ (f_h > f_i \text{ and } s_{11} = -10\ db) - f_l\ (f_l < f_i \text{ and } s_{11} = -10\ db)$$

By applying MOPSO for designing microstrip antenna, we can obtain the class of optimum deigns (Pareto front). And we can choose the proper antenna of our interest among this class. We can also use the resulted Pareto front to study the trade-off between different objectives considered in optimization.

The MOPSO algorithm defined in chapter 2 is used in this chapter for the design and optimization of problems. In equation (1), inertia weight is taken as 0.4 and c1=c2=1.

5.2.1 Optimal Design of Proximity Coupled Dual-Frequency Microstrip Antenna for Wireless Communications

I. Antenna Design

In proximity coupled (also known as electromagnetically coupled) microstrip antenna configuration, the radiating patch, fabricated on a dielectric substrate, is excited by a microstrip line on another substrate, as shown in Figure 5.2. The microstrip patch fabricated on dielectric substrate consists of two square sections of unequal patch dimensions, as shown in Fig 5.3 and this patch is proximity coupled at the common corner of the two patch sections by a microstrip line which is excited by a co-axial connector. The dimensions of the two sections of the patch were adjusted to obtain two operating frequencies, GPS (Global Positioning Satellite System) (1.575) and Bluetooth (2.4-2.484) frequency bands with sufficient bandwidths.

In this work, IE3D software is used for designing and simulating the microstrip antenna. IE3D simulating software developed by Zeeland Software Inc., USA) is a full wave electromagnetic simulation for the microwave and millimeter wave integrated circuits.

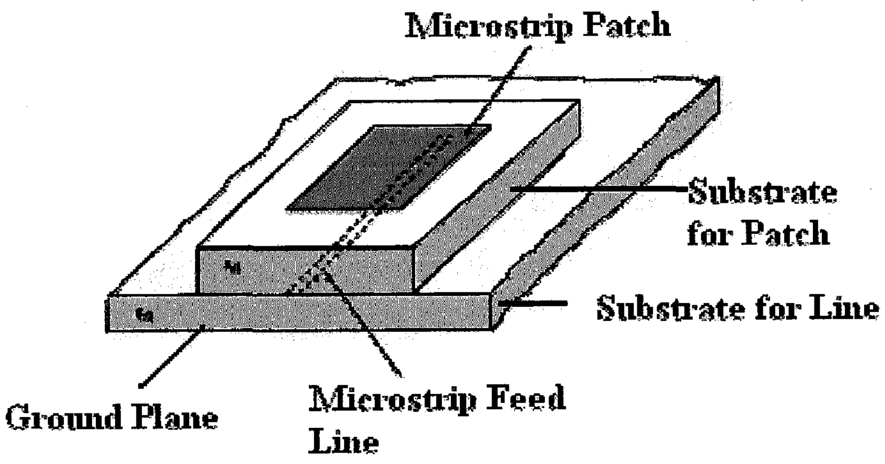


Figure.5.2: Proximity Coupled Microstrip Antenna [39]

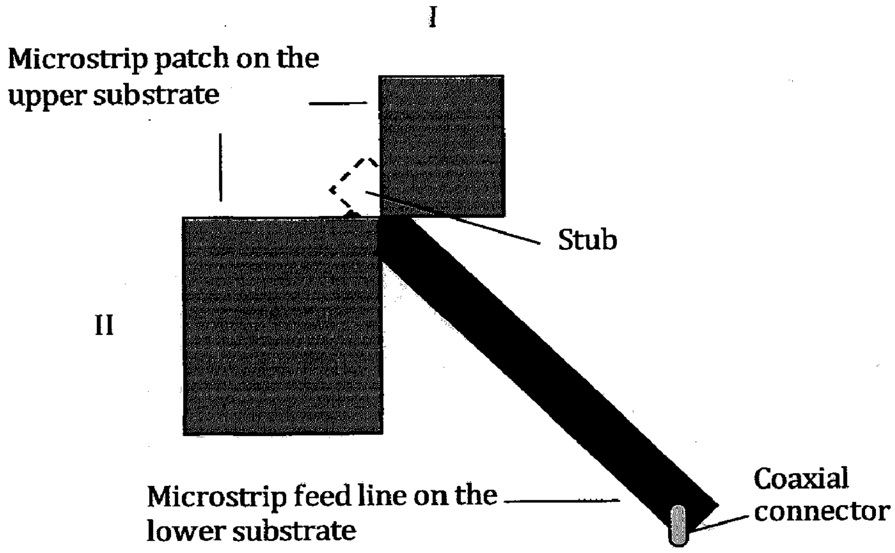

I
Microstrip patch on the upper substrate
Stub
II
Microstrip feed line on the lower substrate
Coaxial connector

Figure.5.3: Antenna Geometry of the Dual-Frequency Proximity Coupled Microstrip Antenna [39]

The design specifications for Dual-Frequency Proximity Coupled Microstrip Antenna are mentioned in Table 5.1.

TABLE 5.1 DESIGN SPECIFICATIONS FOR DUAL-FREQUENCY PROXIMITY COUPLED MICROSTRIP ANTENNA

Design Specifications	Value
Bandwidth (GPS)	20 MHz(1.575 GHz)
Bandwidth (Bluetooth)	84 MHz (2.4-2.484 GHz)

The optimization is aimed to achieve two objective functions: minimum return loss and maximum bandwidth over the operating band (GPS and Bluetooth).

The objective functions are as follows:

1) The corresponding fitness function, *func₁* for return loss used here is quite similar to [40], which is as follows:

$$func_1 = -\left(\sum_{i=1}^{2} w_i \times \frac{s_{11}(i)}{-30} + G_i \right)$$

$$G_i = \begin{cases} 1, \text{ if } S_{11}\,(f_i) \leq -10 \text{ dB}, \\ 0, \text{ if } S_{11}\,(f_i) > -10 \text{ dB}, \end{cases}$$

2) The corresponding fitness function, $func_2$ for bandwidth is as follows:

$$func_2 = \sum_{i=1}^{2} w_i \times \frac{1.0}{BW(i) + 1.0}$$

$i=1$and 2 for GPS (1.575GHz) and Bluetooth (2.4-2.484GHz) band respectively. w_i is the weighting value, which has been selected after a number of preliminary runs, given as: $w_1=0.4$ and $w_2=0.6$, s_{11} is the return loss and BW is the bandwidth for the frequency band.

II. Results

The simulation is run for 100 iterations with 25 particles. The range of values selected for the simulation is shown in the Table 5.2:

TABLE 5.2 RANGE OF DESIGN PARAMETERS FOR DUAL-FREQUENCY PROXIMITY COUPLED MICROSTRIP ANTENNA

Design Parameter	Range
Dimension of square patch (Section I)	[25.0, 32.0] mm
Dimension of square patch (Section II)	[40.0 46.0] mm
Stub length	[4.0 7.5] mm
Length of microstrip feed line	90.0 mm (constant)

The optimized results using PSO is shown here, by considering only one fitness function, *func*$_1$. The PSO algorithm has certainly created an optimal dual band antenna which has the following geometry parameters shown in Table 5.3:

TABLE 5.3 OPTIMIZED VALUE OF DESIGN PARAMETERS FOR DUAL-FREQUENCY PROXIMITY COUPLED MICROSTRIP ANTENNA USING PSO

Design Parameter	Optimized value
Dimension of square patch (Section I)	27.61 mm
Dimension of square patch (Section II)	43.46 mm
Stub length	4.059698 mm

The progress of the PSO routine as a function of the number of iterations is shown in Figure 5.4. The optimal design of proximity coupled dual-frequency microstrip antenna has reached after 84 iterations.

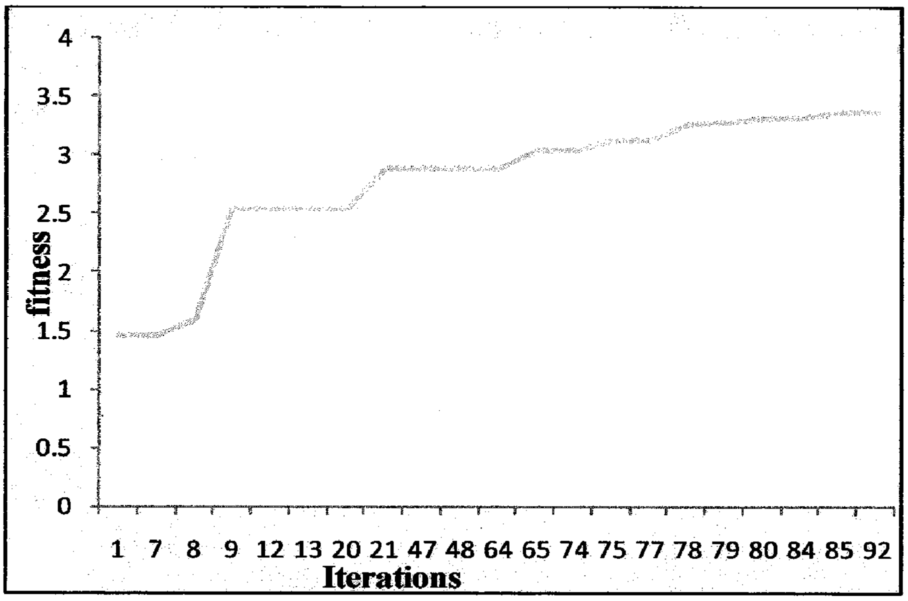


Figure 5.4: Fitness of the best-designed antenna during the progress of the PSO algorithm for the optimal design of Proximity Coupled Microstrip Antenna

Figure 5.5 shows the simulation frequency response of the return loss for the proposed antenna. The simulation result shows the two excited resonant modes at frequencies 1.572 and 2.421, which are the same as those we put in the fitness function of the PSO process.

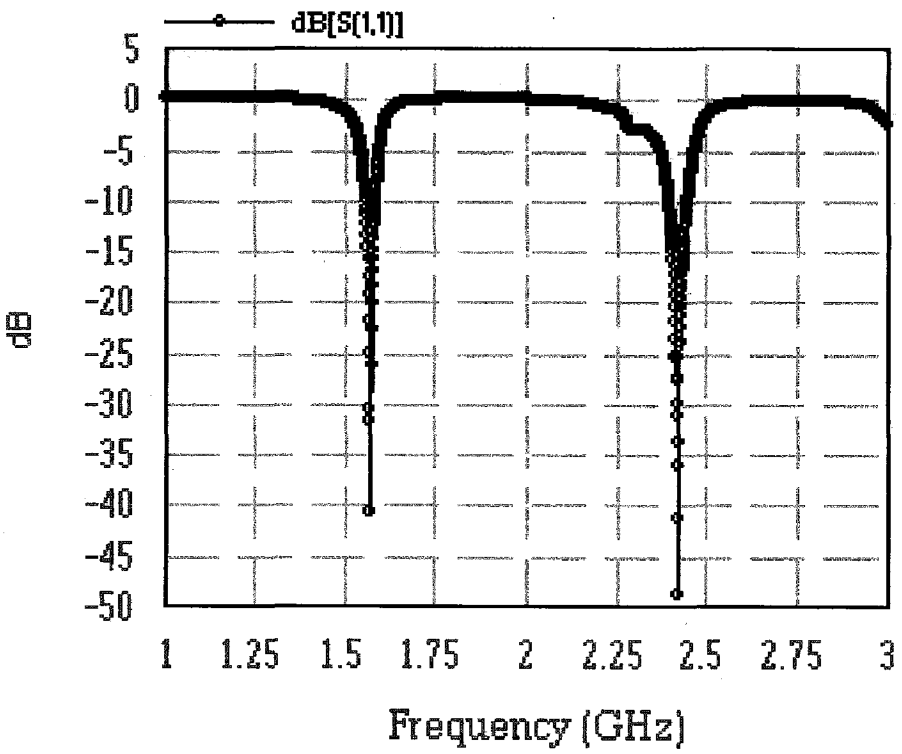


Figure 5.5: Return losses against frequency of Dual-Frequency Proximity Coupled Microstrip Antenna using PSO

The simulated result shows that, PSO has optimized the performance of given microstrip antenna, as we have obtained return loss, s_{11} as:

s_{11} = -48.99 db for Bluetooth

s_{11} = -40.83 db for GPS

The bandwidth obtained at two frequency bands through simulation result is:

BW = 25MHz (1.561-1.586) for GPS

BW = 50MHz (2.399-2.449) for Bluetooth

By applying PSO, we have obtained minimum return loss, but the obtained bandwidth is less than the required bandwidth, which can have adverse effect after the fabrication; i.e. it is possible that we may not obtain the resonance at the band which has less bandwidth.

So to overcome this problem, we have applied MOPSO for designing microstrip antenna for two objectives return loss and bandwidth.

In this work, we have shown one of the Pareto optimal solutions we have obtained using MOPSO-CD [23] in figure 5.6, for designing and optimizing Proximity Coupled Dual-Frequency microstrip antenna for two objectives, minimum return loss and maximum bandwidth, and the optimized value of design parameters for this Pareto optimal solution is shown in Table 5.4.

TABLE 5.4 OPTIMIZED VALUE OF DESIGN PARAMETERS FOR DUAL-FREQUENCY PROXIMITY COUPLED MICROSTRIP ANTENNA USING MOPSO

Design Parameter	Optimized value
Dimension of square patch (Section I)	27.5 mm
Dimension of square patch (Section II)	43.7 mm
Stub length	4.0 mm

The simulation result shows the two excited resonant modes at frequencies 1.574 and 2.428 with return loss, s_{11} as:

s_{11} = -34.68 db for Bluetooth
s_{11} = -28.19 db for GPS

The bandwidth obtained at these two frequency bands through MOPSO simulation result is:

GPS – 30MHz (1.561-1.589)
Bluetooth – 80MHz (2.39-2.47)

The bandwidth obtained for two bands (GPS and Bluetooth) through MOPSO is approximately equal to the required bandwidth, i.e we have obtained the required results using MOPSO. The Pareto front showing the trade-off between the two objectives, $func_1$

and *func*$_2$ for obtaining optimal design of dual-frequency proximity coupled microstrip antenna using MOPSO is shown in Figure 5.7

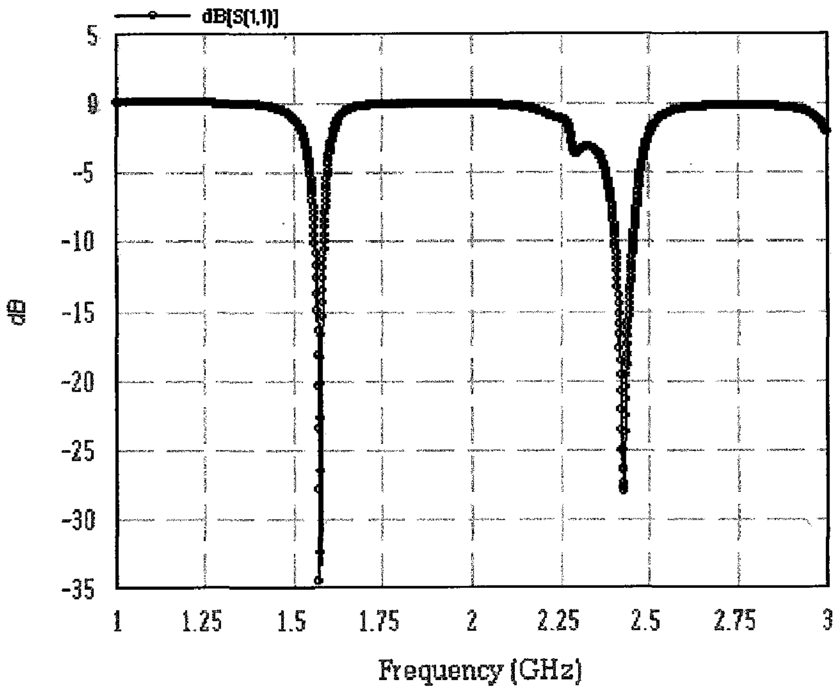


Figure 5.6: Return losses against frequency of Dual-Frequency Proximity Coupled Microstrip Antenna using MOPSO

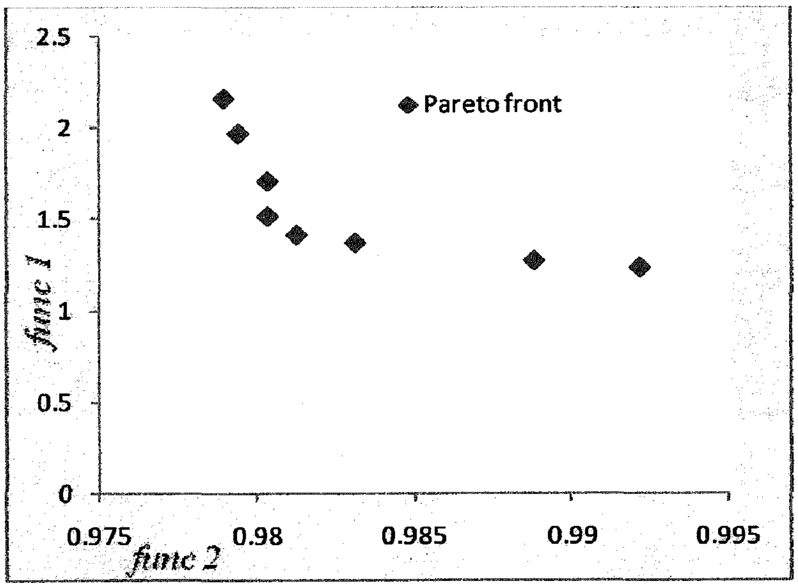


Figure.5.7: Pareto front of Dual-Frequency Proximity Coupled Microstrip Antenna using MOPSO for the two objective functions (func$_1$ and func$_2$)

5.2.2 Optimal Design of Compact Triple-Band Microstrip Antenna to obtain the frequency response at GSM and WLAN

I. Antenna Design

The antenna design of compact triple-band microstrip antenna was available in CST simulator. But for our work, the antenna design is required in IE3D simulator, as we have interfaced the MOPSO code with the IE3D simulator. So, we have designed the antenna geometry in IE3D simulator.

The design specifications for Dual-Frequency Proximity Coupled Microstrip Antenna are mentioned in Table 5.5.

TABLE 5.5 DESIGN SPECIFICATIONS FOR COMPACT TRIPLE-BAND MICROSTRIP ANTENNA

Design Specifications	Value
Bandwidth (GSM)	70 MHz (890-960 MHz)
Bandwidth (WLAN system in 2.4 GHz band)	84 MHz (2.4-2.484 GHz)
Bandwidth (WLAN system in 5.2 GHz band)	200 MHz (5.150-5.350 GHz)

For obtaining triple band behaviour of the antenna U-slots have been used. As the frequencies that are required are very widely spaced from each other we require a dielectric substrate with low dielectric constant and higher thickness. Hence foam having dielectric constant of 1.18 and a thickness of 2.59 cm has been used as a dielectric material. We have used the shorting wall to reduce the size of patch.

Two U-slots make three current paths and hence the patch resonates at three frequencies but the shorting wall that is used here shorts the current in one path. Hence three U-slots have been used to obtain triple band behaviour. Coaxial cable is used as the feeding method. The antenna geometry is shown ifFigure 5.8.

The parameters used for optimization of the design are listed out in Table 5.6. We can see from above Table 5.6, that we have 4 constant and 11 variables. So, particles of swarm have 11-dimensional space.

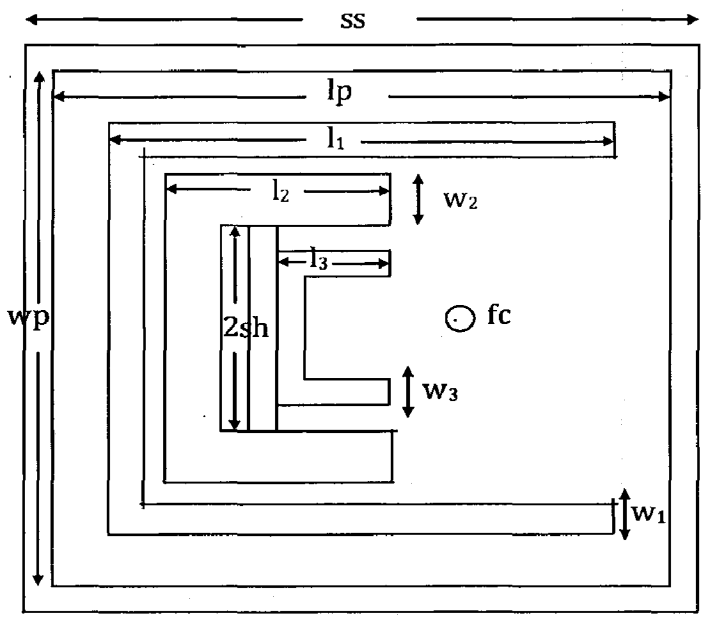


Figure.5.8: Structure of the Compact Triple-Band Microstrip Antenna

TABLE 5.6 DIFFERENT PARAMETERS OF THE LAYOUT OF COMPACT TRIPLE-BAND MICROSTRIP ANTENNA

Parameter Name	Description	Range of values
fc	Centre of feed point	[0.0 , 8.0]
ss	Side of substrate	23.0(constant)
lp	Length of patch	21.0(constant)
wp	Width of patch	21.0(constant)
sh	Length of shorting wall	10.0(constant)
shp	Position of shorting wall	[7.0, 10.0]
r1x	X coordinate of corner of outer slot	[15.0, 20.0]
r1y	Y coordinate of corner of outer slot	[16.0, 19.0]
r2	Lower corner of outer slot	[15.0, 17.0]
s1x	X coordinate of corner of middle slot	[0.0, 5.0]
s1y	Y coordinate of corner of middle slot	[11.0, 15.0]
s2	Lower corner of middle slot	[8.0, 13.0]
t1x	X coordinate of corner of inner slot	[0.0, 3.0]
t1y	Y coordinate of corner of inner slot	[6.0, 9.0]
t2	Lower corner of inner slot	[3.0, 6.0]

The optimization is aimed to achieve two objective functions: minimum return loss and maximum bandwidth over the operating bands (GSM and WLAN).

The objective functions are as follows:

1) The corresponding fitness function, f_1 for return loss used here is quite similar to [40], which is as follows:

$$func_1 = -\left(\sum_{i=1}^{3} w_i \frac{s_{11}(i)}{-30} + G_i \right)$$

$$G_i = \begin{cases} 1, \text{ if } S_{11}\ (f_i) \leq -10 \text{ dB}, \\ 0, \text{ if } S_{11}\ (f_i) > -10 \text{ dB}, \end{cases}$$

2) The corresponding fitness function, f_2 for bandwidth is as follows:

$$func_2 = \frac{1.0}{BW(i=1)+1.0}$$

i=1, 2 and 3 for (GSM; 890-960 MHz) and wireless local area network (WLAN) systems in the 2.4 GHz (2400-2484 MHz) and 5.2 GHz (5150-5350 MHz) bands respectively.

w_i is the weighting value, which has been selected after a number of preliminary runs, given as: w_1=0.4, w_2=0.3 and w_3=0.3, s_{11} is the return loss and BW is the bandwidth for the frequency band.

II. Results

The Figure 5.9 shows that result obtained for compact triple band using MOPSO. We can see that the results obtained are not very good. The reason for this is; we have designed the CST design in IE3D simulator, and IE3D simulator is unable to simulate the design of compact triple band properly. The IE3D simulator has split the geometry into number of small geometries, due to which the geometry has not simulated in the required manner.

The above optimization takes lot of time, as the geometry is very complex, as tens of minutes is consumed in the single IE3D simulator call. So, it is required to parallelize the optimization of antenna design.

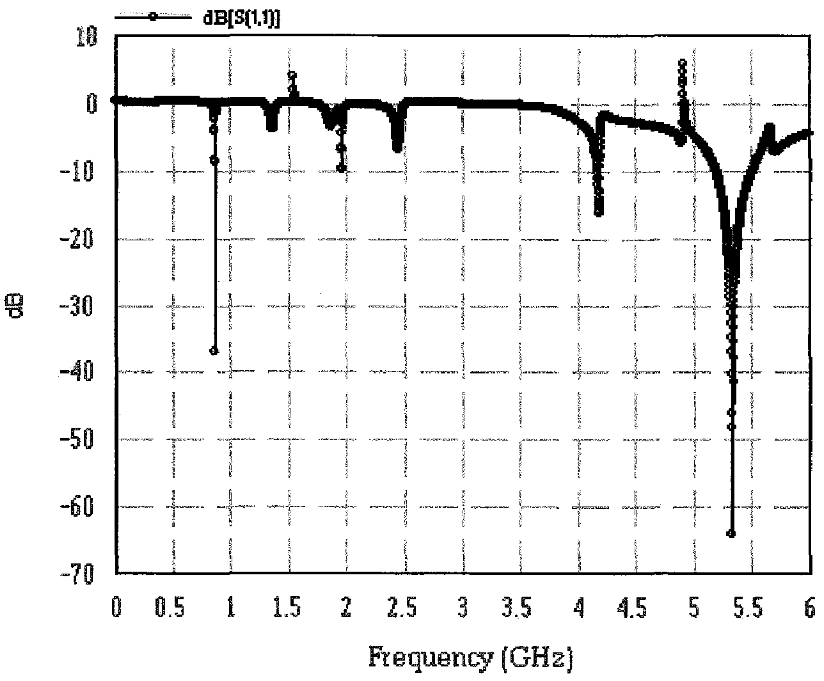


Figure 5.9: Return losses against frequency of Compact Triple-Band Microstrip Antenna using MOPSO

5.2.3 Design and optimization of a nonlinear taper for a 200 KW, long-pulse strat-up gyrotron at 170 GHz

Gyrotron output system consists of an output taper which connects the interaction region with the main waveguide system, a quasi-optical mode converter, and the RF window. The nonlinear taper should provide a perfect match between interaction region and the output waveguide with negligible mode conversion [41].

In our work, we have used a raised cosine taper profile as it yields less mode conversion than the other tapers. A schematic diagram of a raised-cosine taper considered in this work is shown in Figure 5.10.

The scattering matrix method is very fast and accurate for taper analysis. The analysis of taper was carried out using a dedicated scattering matrix code [42]. The tapered parts were divided like a flight of stairs as shown in Figure 5.10. The scattering coefficient of each step was calculated by using a dedicated scattering matrix code. The scattering matrix code is invoked iteratively from the optimization loop of PSO algorithm, for each particle.

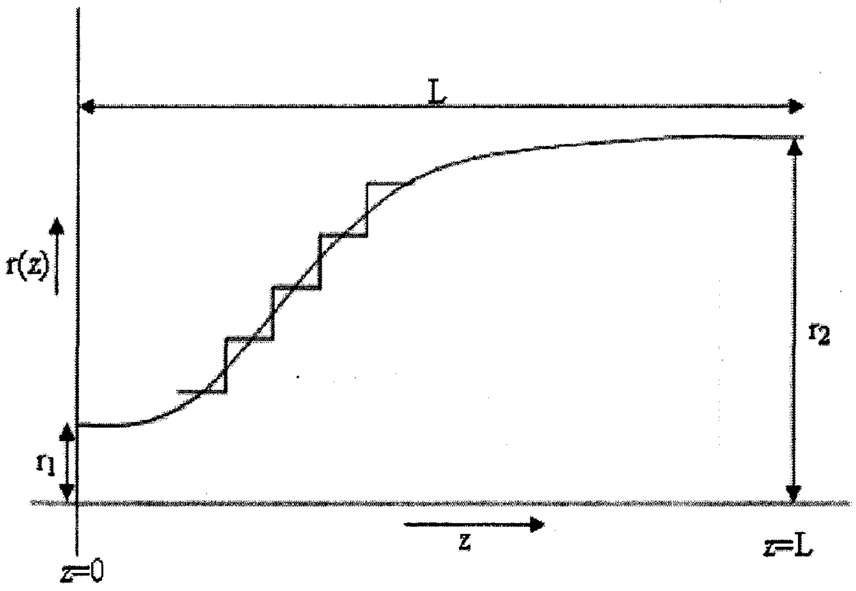


Figure 5.10: The raised-cosine taper profile

The design specifications of non-linear taper are mentioned in Table 5.7.

TABLE 5.7 DESIGN SPECIFICATIONS FOR NON-LINEAR TAPER

Design Specifications	Value
frequency	170 GHz
power	200 KW
operating mode	$TE_{24,8}$
taper profile	raised-cosine

The objective of the non-linear taper design for 170 GHz, 200 KW, CW gyrotrons is to obtain maximum transmission coefficient (s_{21}-parameter). The s_{21} is obtained for the $TE_{24,8}$ operating mode from the output.dat file generated by the the scattering matrix code.

The objective function is:

$$func_1 = -S_{21}(TE_{24,8})$$

I. Results

In this work, the design optimization of raised cosine taper for 170 GHz, 200 KW, CW gyrotrons has been carried out to give the maximum transmission in the operating mode ($TE_{24,8}$) with very less mode conversion, using PSO. The design parameters for non-linear tapper are radius of taper at input end (r_1), radius of taper at output end (r_2), length of taper (L), number of sections (N) and gamma (γ). The simulation is run for 100 iterations with 10 particles. The range of design parameters considered for the simulation are mentioned in Table 5.8.

TABLE 5.8 RANGE OF DESIGN PARAMETERS FOR RAISED COSINE TAPER FOR 170 GHZ, 200 KW, CW GYROTRONS

Design Parameter	Range
L	[100, 150] mm
r_1	16.27 mm (constant)
r_2	17.039 mm (constant)
N	[50, 500]
γ	[0.1, 1.0]

The optimized value of design parameters l, n and γ is shown in Table 5.9, which gives maximum the transmission coefficient (s_{21}) in the operating mode ($TE_{24,8}$), with very less mode conversion, using PSO.

TABLE 5.9 OPTIMIZED VALUE OF DESIGN PARAMETERS FOR RAISED COSINE TAPER FOR 170 GHZ, 200 KW, CW GYROTRONS

Design Parameter	Optimized value
L	100 mm
N	480
γ	0.559
s_{21} ($TE_{24,8}$)	99.0267

On varying the radius of taper at output end (r_2) with the other design parameters, we have obtained maximum transmission coefficient (s_{21}) in the operating mode ($TE_{24,8}$), with very less mode conversion, using PSO:

Range of r_2 = [16.5, 22.5] mm

The optimized value of design parameters l, n and γ is shown in Table 5.10.

TABLE 5.10 OPTIMIZED VALUE OF DESIGN PARAMETERS FOR RAISED COSINE TAPER FOR 170 GHZ, 200 KW, CW GYROTRONS WITH VARYING RADIUS OF TAPER AT OUTPUT END

Design Parameter	Optimized value
L	100 mm
N	479
γ	0.67
r_2	21.71
s_{21} ($TE_{24,8}$)	99.4385

We have also observed the effects of varying gamma parameter on the taper synthesis and the transmission coefficient. The effect of the gamma parameter is shown in Figure 5.11.

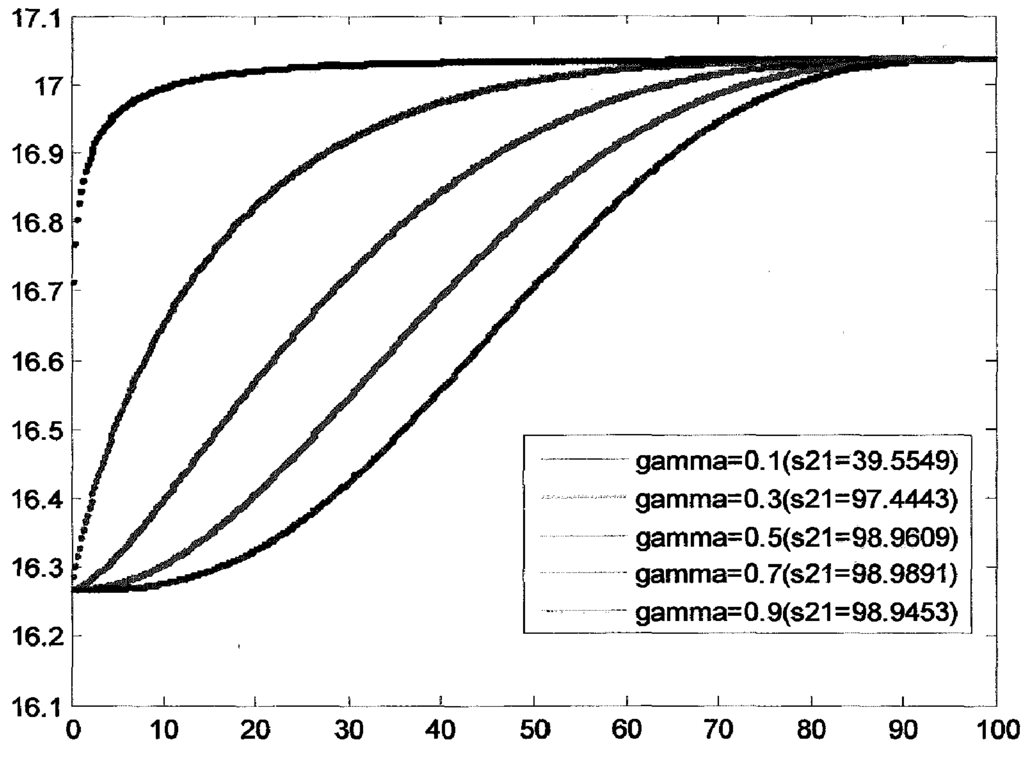


Figure 5.11: Contours of raised-cosine taper showing the effect of parameter gamma (γ) (*L=100, N=480, r2=17.039 mm*)

PSO has the advantage of easy implementation, while maintaining strong abilities of convergence and global search. In spite of those advantages, PSO still needs a long time to find solutions for large scale problems, such as problems with large dimensions and problems which need a large swarm population for searching in the solution space. The main reason for this is that the optimizing process of PSO requires a large number of fitness evaluations, which are usually done in a sequential way on CPU, so the computation task can be very heavy and thus running speed of PSO may be quite slow [43]. A promising approach to overcome this limitation is to parallelize these algorithms.

6.2 Implementation of MOPSO on GPU

The difference between a CPU function and a GPU kernel is that execution of the kernel should be parallelized. So we must design the parallelization methods for all the sub-processes of optimizing by MOPSO. In this section, we present a model for implementing parallel MOPSO (MOPSO-CD) on GPU.

The model is illustrated in Figure 6.1. In this model, the most computationally intensive part, fitness function evaluation, compute velocity are performed on GPU in parallel i.e. synchronously for each particle of the swarm. For this, firstly the data is transferred from CPU to GPU, then the threads are created in CUDA which is equal to the number of particle N, and then the computations within the iterative process are performed on GPU.

Sub-processes within the iteration are parallelized. For all the sub-processes, the iteration is only applied to dimension index $d = (1, 2..., D)$ while on CPU, it should also be applied to the particle index $j = (1, 2 ...N)$, where N is population size. The reason is that the arithmetical operation to all the N data in dimension d is done in parallel (synchronously) on GPU. In the same dimension d $(d= 1....D)$, the position, velocity and fitness of all particles are updated in parallel. The general parallelized algorithm for all the sub-processes is illustrated in Algorithm below. The sub-process, Update Archive cannot be parallelized, as each particle in the swarm population has to be checked sequentially for non-dominance criteria.

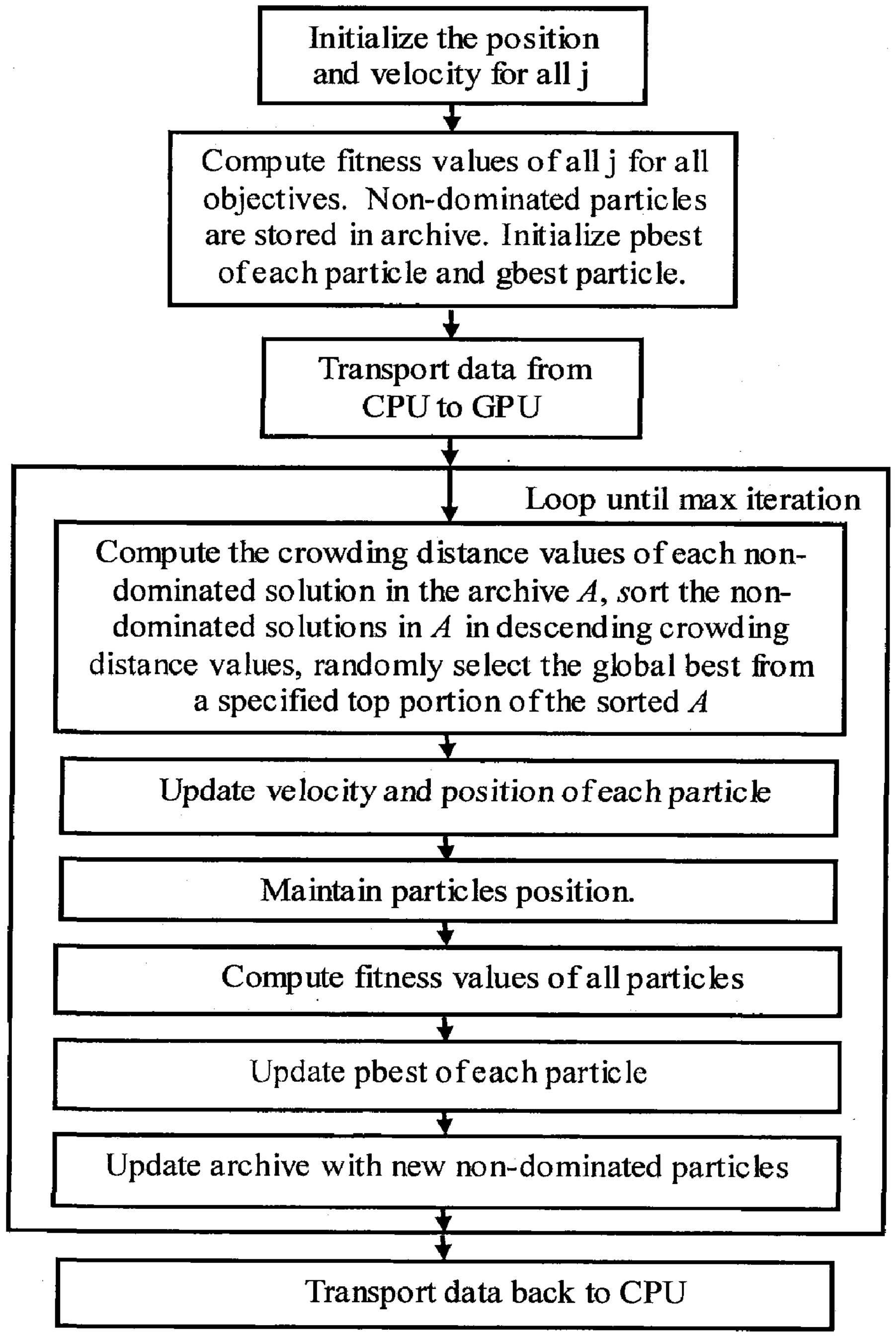

Initialize the position and velocity for all j
Compute fitness values of all j for all objectives. Non-dominated particles are stored in archive. Initialize pbest of each particle and gbest particle.
Transport data from CPU to GPU
Loop until max iteration
Compute the crowding distance values of each non-dominated solution in the archive A, sort the non-dominated solutions in A in descending crowding distance values, randomly select the global best from a specified top portion of the sorted A
Update velocity and position of each particle
Maintain particles position.
Compute fitness values of all particles
Update pbest of each particle
Update archive with new non-dominated particles
Transport data back to CPU

Figure.6.1: Parallel Implementation of MOPSO on CUDA

The mechanism of crowding distance (for replacing solutions in the archive with new solutions, when archive is full.) and mutation can be added to the above algorithm for maintaining the diversity of non-dominated solutions and to better convergence towards the Pareto front.

General Algorithm for all parallelized sub-process on CUDA

```
Step1. Initialize - set the 'block size' and 'grid size', with the number of threads equaling
to the number of particles N.

Step2. For each dimension d do
    ➢ Map all threads to the N position values one-to- one
    ➢ Load N data from global to shared memory
// Do operations to thread j (j = 1....N), synchronously
    ➢ Apply sub-process operation to all N data in parallel (for fitness function, fitness
      value corresponding to each objective is calculated)
    ➢ Store the results
End for
```

A. Data Organization

CUDA offers global memory to share data among different kernels. The global memory only allows the allocation of one dimensional array, so only one-dimensional array are used here for storing data, including the position, velocity and fitness values of all the particles.

Let us assume that the problem has *D* variables/dimension and *F* objectives, and the swarm population is *N*. So, the array of position and velocity is represented with the length of *D*N*, and fitness with the length of *F*N*. An array *X* of length *D*N* which is used to store the position value of the particles, can be seen in the Figure 6.2.

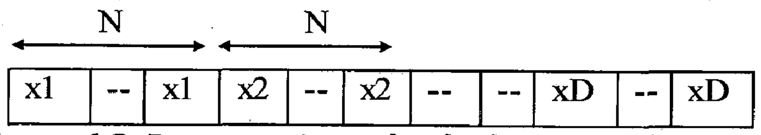


Figure.6.2: Representing individuals on global memory

The same variables from all individuals are grouped and form a tile of N values in the global memory [44]. On the other hand, the efficiency of accessing the same variables of all individuals in parallel will be reduced, if an individual is mapped to D consecutive locations, because the simultaneous memory accesses cannot be coalesced and multiple memory transactions are required.

But these one dimensional arrays should be logically seen as a two-dimensional array *Y*. An element with the index of *(i, j)* in *Y* corresponds to the element in *X* with the index *(i*N+ j)* where *i*-th is the dimension of the *j*-th particle in the swarm. So, the array of position and velocity is represented with the length of *D*N*, and fitness with the length of *F*N*.

B. Random Number Generation

During the process of optimization, MOPSO needs lots of random numbers for velocity updating. As random numbers generation on GPU is very tricky, so we have rather generated random numbers on CPU and transport them to GPU. However the data transportation between GPU and CPU is quite time consuming, so we do not want to transfer them to GPU, during each iteration of MOPSO.

To resolve this problem we have generated *M* (*M* >>*D*N*) random numbers on CPU before running MOPSO. Then they are transported to GPU once and stored in an array *R* on the global memory. When the velocity updating is carried through, we just pass two random integer numbers *P1,P2* from CPU to GPU, then *2*D*N* numbers can be drawn from array *R* starting at *R(P1)* and *R(P2),* respectively.

6.3 Experiments

The experimental platform used is based on Intel(R) Xeon(R) CPU E5420 @ 2.50 GHz, 2.49 GHz, 16.0 GB RAM, NVIDIA Quadro FX 3700, and Windows XP (x64).

In this chapter, we have compared the CPU and GPU implementation results of following problems:

1. MOPSO on benchmark function DTLZ6 [23], which is the minimization problem with 3 objectives and 22 variables.
2. MOPSO for rule mining, the problem has 4 objectives for maximization.

We have called the MOPSO run on CPU and GPU as CPU-MOPSO and GPU-MOPSO respectively. The Speedup is defined as the times that GPU-MOPSO runs faster than CPU-MOPSO.

$$\gamma = \frac{T_{CPU-MOPSO}}{T_{GPU-MOPSO}} \quad \text{.......} (7)$$

where γ is speedup, $T_{GPU\text{-}MOPSO}$ and $T_{CPU\text{-}MOPSO}$ is the time taken by GPU-MOPSO and CPU-MOPSO respectively.

6.3.1 Results of MOPSO on benchmark function DTLZ6

To optimize the function DTLZ6 we have taken the number of iterations as 2000 and the size of population, defined as N is varied from 100 to 2000 in this run. The Result of DTLZ6 on CPU-MOPSO and GPU-MOPSO is shown in Table 6.1.

The running time and speedup versus population size is shown in Figure 6.3 and Figure 6.4 respectively. We can see from the Figure 6.3 that the running time of GPU-MOPSO and CPU-MOPSO is proportional to the swarm population, namely the time increase linearly with swarm population, keeping the other parameters constant. And the figure 6.4 shows that the speed up increases with swarm population

TABLE.6.1: RESULT OF DTLZ6 ON CPU-MOPSO AND GPU-MOPSO

N	$T_{GPU\text{-}MOPSO}$	$T_{CPU\text{-}MOPSO}$	γ
100	3.7820	4.5310	1.1980
500	8.7650	22.6250	2.5812
1000	16.0310	48.7970	3.04391
1500	21.0160	75.6559	3.5999

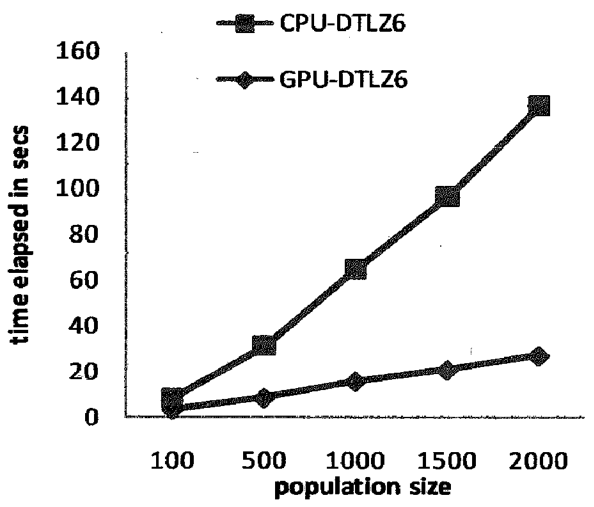


Figure.6.3: Running Time and Swarm Population for benchmark problem (DTLZ6) using MOPSO on CPU and GPU

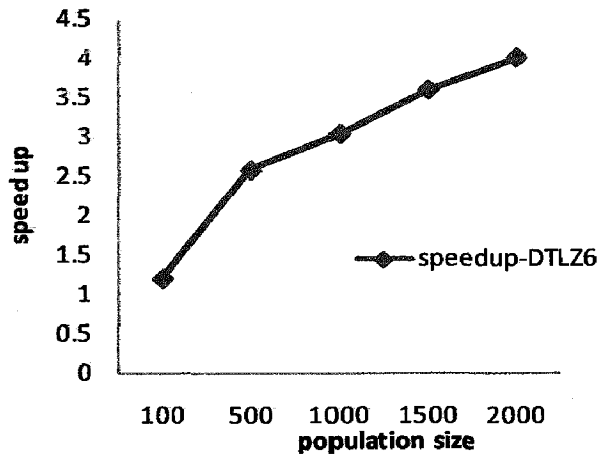


Figure.6.4: Speedup and Swarm Population for benchmark problem (DTLZ6) using MOPSO

6.3.2 Results of MOPSO for rule mining

In this we have generated the rules for mondrian foodmart dataset. The description of mondrian foodmart dataset is provide in chapter 5, it has 70 attributes and 5576 number of records. We have taken the number of iterations as 100 and the size of population, defined as *N* is varied from 500 to 2000 in this run. The result of Mondrian foodmart dataset on CPU-MOPSO and GPU-MOPSO is shown in Table 6.2.

TABLE.6.2: RESULT OF MONDRIAN FOODMART DATASET ON CPU-MOPSO AND GPU-MOPSO

N	$T_{CPU-MOPSO}$	$T_{GPU-MOPSO}$	γ
100	6.941	3.203	2.167
500	14.818	6.359	2.3302
1000	30.657	11.344	2.7024
1500	46.698	15.516	3.01

The running time and speedup versus population size is shown in Figure 6.5 and Figure 6.6 respectively. We can see from the Figure 6.5 that the running time of GPU-MOPSO and CPU-MOPSO is proportional to the swarm population, namely the time increase linearly with swarm population, keeping the other parameters constant. And the figure 6.6 shows that the obtained speed up is almost constant with swarm population.

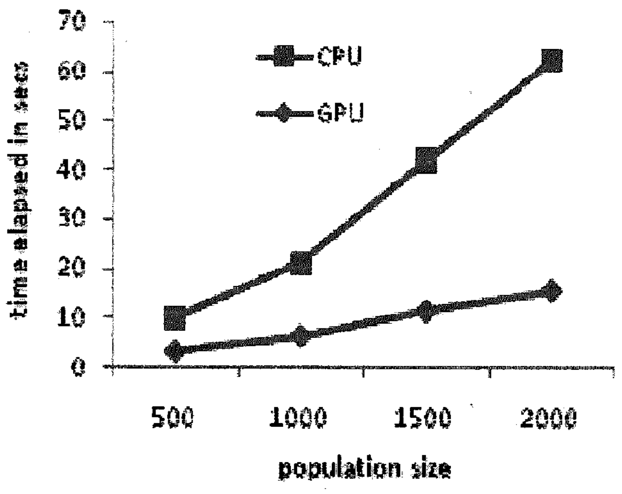


Figure.6.5: Running Time and Swarm Population for association rule mining using MOPSO on CPU and GPU

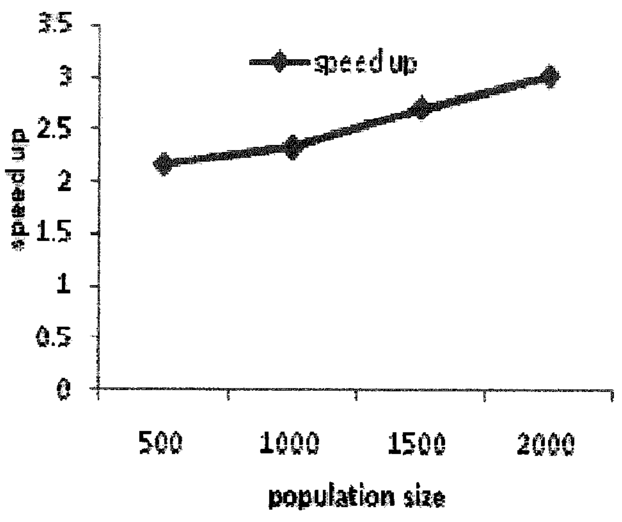


Figure.6.6: Speedup and Swarm Population for association rule mining using MOPSO

Chapter 7 Conclusion and Future Work

7.1 Conclusion

In this dissertation, we have applied multi-objective particle swarm optimization (MOPSO) algorithm in two real-life domains; data mining (for association rule mining) and micro-/millimeter wave (for optimized design of microstrip antenna).

We have used MOPSO for rule generation in market basket dataset. In this we have also added an extra step before rule generation step, for clustering the dataset according to itemset size and has called it cluster based – MOPSO (CB-MOPSO). In our proposed work, we have shown that CB-MOPSO gives more optimized results compared to MOPSO. As the number of rules generated in CB-MOPSO is same as in MOPSO, but in much less time.

We have used particle swarm optimization algorithm (PSO) for the design and optimization of micro-/millimeter wave components; we have optimized the design of two microstrip antenna (Proximity Coupled Dual-Frequency for wireless communication and compact triple band microstrip antenna for GSM and WLAN) and nonlinear tapper for a 1.0-1.3 MW, long-pulse strat-up gyrotron at 127.5 GHz. We have optimized the performance of microstrip antenna and nonlinear tapper by choosing the most appropriate configuration parameters. We have also shown that, by applying particle swarm optimization in microstrip antenna design to obtain the minimum return loss, we have comprised the bandwidth. So, to take into account both bandwidth and return loss (and some other objectives), we should use MOPSO for designing and optimizing the microstrip antenna.

The implementation of MOPSO (for benchmark (DTLZ6) and rule mining problem) on GPU based on the software platform of CUDA from NVIDIA Corporation is also presented in this work. GPU-MOPSO has the following features: the running time of GPU-MOPSO is greatly shortened over CPU-MOPSO, the running time and swarm population size take a linear relationship, swarm population can be very large, and the larger the population is, the faster GPU-MOPSO runs than CPU-MOPSO.

7.2 Future Work

In our proposed work, the geometry of compact triple band microstrip antenna is very complex. And it took number of days to simulate the microstrip antenna using EM simulator (IE3D. The optimization algorithm has to trigger the solver (IE3D) thousands of times during its run. The single call of IE3D can take few seconds to half an hour to simulate the geometry of microstrip antenna, depending on the complexity of geometry.

The future scopes to the present work are stated as below:

- Implementation of MOPSO for design optimization of microstrip antenna on multi-core architecture (Due to the limited memory of NVIDIA graphics card, antenna design may not be carried out on CUDA environment. Antenna design requires huge file transfer from CPU to GPU.)
- MOPSO can be applied for the design problems of other engineering disciplines.
- Different Evolutionary algorithms (bacterial foraging optimization, ant colony optimization, differential evolution) can be applied on presented problems. It would be useful to compare the results with MOPSO.
- More computation intensive applications can be parallelized using CUDA environment.

References

[1]. K. Deb, *Optimization for engineering design: algorithms and examples*, N.Delhi : PHI, 2000.

[2]. K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, John Wiley & Sons, 2001.

[3]. Eckart Zitzler, Marco Laumanns, and Stefan Bleuler. "A Tutorial on Evolutionary Multiobjective Optimization". In Xavier Gandibleux et al., editor, *Metaheuristics for Multiobjective Optimisation*, pages 3–37, Berlin, 2004. Springer. Lecture Notes in Economics and Mathematical Systems Vol. 535.

[4]. Carlos A. Coello Coello, "Twenty Years of Evolutionary Multi-Objective Optimization": A Historical View of the Field, *IEEE Computational Intelligence Magazine*, 2006.

[5]. Carlos A. Coello Coello and Gary B. Lamont, editors. "Applications of Multi-objective Evolutionary Algorithms. World Scientific, Singapore", 2004. ISBN 981-256-106-4.

[6]. J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: *Proceedings of IEEE International Conference on Neural Networks*, vol. 4, 1995, 1942–1948.

[7]. You Zhou and Ying Tan. "GPU-based Parallel Particle Swarm Optimization. Evolutionary Computation, 2009. CEC'09". IEEE Congress on 18-21 May 2009 Page(s):1493 – 1500.

[8]. NVIDIA Corporation: NVIDIA CUDA compute unified device architecture programming guide. NVIDIA Corporation, Jan 2007 (CUDA programming Guide 2.0.

[9]. Eiben, A. E.; Smith, J. E. Introduction to Evolutionary Computing. Springer, Cap. 2, 2003, 15-35. ISBN: 3-540-40184-9. Available at: <http://www.cs.vu.nl/%7Egusz/ecbook/Eiben-Smith-Intro2EC-Ch2.pdf>. Accessed on: 03 Apr 2005.

[10]. Carlos A. Coello Coello. "A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques". *Knowledge and Information Systems. An International Journal*, 1(3):269–308, August 1999.

[11] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.

[12] Carlos M. Fonseca and Peter J. Fleming. "Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization". In Stephanie Forrest,

editor, *Proceedingsof the Fifth International Conference on Genetic Algorithms*, pages 416–423, San Mateo, California, 1993. University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers.

[13] N. Srinivas and Kalyanmoy Deb. "Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms". *Evolutionary Computation*, 2(3):221–248, Fall 1994.

[14] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-11. IEEE Transactions on Evolutionary Computation, 6(2):182-197, April 2002.

[15] Jeffrey Horn, Nicholas Nafpliotis, and David E. Goldberg. "A Niched Pareto Genetic Algorithm for Multiobjective Optimization". In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, volume 1, pages 82–87, Piscataway, New Jersey, June 1994. IEEE Service Center.

[16]. YI Hong-Xia, XIAO Liu, LIU Pu-Kun, "Intelligent Algorithms for solving multiobjective optimization problems", WiCOM'08 4th international conference on 12-14 Oct, 2008, pages 1-5.

[17] Parsopoules K E, Vrahatis M N,"Particle Swarm Optimization Method in Multiobjetive Problems [A],"Proceedings ACM Symposium on Applied Computing[C]. 2OO2. 603-607.

[18]. T. Ray, K.M. Liew, A swarm metaphor for multiobjective design optimization, Engineering Optimization 34 (2) (2002) 141–153.

[19]. Mostaghim S, Teich J, "Strategies for Finding Local Guides in Multi-objective Particle Swarm Optimization(MOPSO)[A], "Proceedings of the IEEE Swarm Inteligence Symposium [C].2003.26-33.

[20]. Hu X, Eberhart R,"Multiobjective Optimization Using Dynamic Neighborhod Particle Swarm Optimization [A]," Proceedings of the IEEE Congress on Evolutionary Computation[C]. 2002.

[21]. Konstantinos E. Parsopoulos, Dimitris K. Tasoulis, and Michael N.Vrahatis. Multiobjective optimization using parallel vector evaluated particle swarm optimization. In Proceedings of the IASTED International Conference on Artificial Intelligence and Applications (AIA 2004), volume 2, pages 823–828, Innsbruck, Austria, February 2004. ACTA Press.

[22]. X. Li, "A nondominated sorting particle swarm optimizer for multiobjective optimization", Proceedings of the Conference on Genetic and Evolutionary Computation, vol. 2723, Springer, Berlin, Germany, 2003, pp. 37–48.

[23]. C.R. Raquel, Prospero C. Naval, Jr., "An effective use of crowding distance in multi- objective particle swarm optimization", in: Proceedings of the Conference on Genetic and Evolutionary Computation, ACM Press, New York,NY, USA, 2005, pp. 257–264.

[24] M.J. Berry, G. Linoff, Data Mining Techniques for Marketing, Sales and Customer Support, John Wiley and Sons, New York, 1997.

[25] M.S. Chen, J. Han, P.S. Yu, Data mining an overview from a database perspective, IEEE Transactions on Knowledge and Data Engineering 6 (1996) 866–883.

[26]. K.J. Cios, W. Pedrycz, R.W. Swiniarski, "Data Miming Methods for Knowledge Discovery", Kluwer Academic Publishers, Boston, MA (2000)

[27] R. Agrawal, T. Imieliński,A. Swami, "Mining association rules between sets of items in large databases", Proceedings of the 1993 ACM SIGMOD international conference on Management of data, p.207-216, May 25-28, Washington, D.C., United States (1993)

[28] R. Agrawal , R. Srikant, "Fast Algorithms for Mining Association Rules in Large Databases ", Proceedings of the 20th International Conference on Very Large Data Bases, p.487-499, September 12-15 (1994)

[29]. B. Dehuri, A. Ghosh, "Muti Objective Association Rule Mining Using Genetic Algorithm", Information Sciences, Volume 163, pp: 123–133 (2004)

[30] Ishibuchi, H., Kuwajima, I., Nojima, Y.: Multiobjective Association Rule Mining. In: Proceedings of the PPSN Workshop on Multiobjective Problem Solving from Nature (2006)

[31]. Augusto de Almeida Prado G. Tor'acio, "Multiobjective Particle Swarm Optimization in Classification-Rule Learning", C.A. Coello Coello et al. (Eds.): Swarm Intel. for Multi-objective Prob., SCI 242, pp. 37–64, Springer-Verlag Berlin Heidelberg 2009

[32]. Ali Hadian, Mahdi Nasiri, Behrouz Minaei-Bidgoli, " Clustering Based Multi-Objective Rule Mining using Genetic Algorithm", *International Journal of Digital Content Technology and its Applications* Volume 4, Number 1, February 2010

[33] http://sourceforge.net/projects/mondrian

[34]. Chauhan, N.C., "Soft Computing Techniques for Design Applications In Microwave Domain," Ph.D Thesis, Indian Institute of Technology-Roorkee, India, 2009.

[35]. J. Robinson and Y. Rahmat-Samii, "Particle swarm optimization in electromagnetics," *IEEE Transactions on Antennas and Propagation*, vol. 52, no. 2, pp. 397–407, 2004.

[36]. Y. Rahmat-Samii, D. Gies, and J. Robinson, "Particle swarm optimization (PSO): a novel paradigm for antenna designs," *The Radio Science Bulletin*, vol. 305, pp. 14–22, 2003.

[37]. Nanbo Jin and Yahya Rahmat-Samii, "Particle Swarm Optimization for Antenna Designs in Engineering Electromagnetic", *Journal of Artificial Evolution and Applications*, Volume 2008, Article ID 728929, 10 pages.

[38]. Carver, K. R., Mink, J. W. "Microstrip Antenna Technology". IEEE transactions on Antennas and Propagation, v. AP-29, Jan. 1981, pp. 2-24.

[39]. Jibendu Sekhar Roy and Milind Thomas, "Investigations on A New Proximity Coupled Dual-Frequency Microstrip Antenna for Wireless Communication", Mikrotalasna revija, 2007, vol. 13, br. 1, str. 12-15.

[40] W.C. Liu, "Optimal Design of Dual band CPW-fed G-shaped monopole antenna for WLAN application," *Progress In Electromagnetic Research*, PIER 74, 2007, 21-38.

[41] M.V. Kartikeyan, E. Borie, and M. Thumm, Gyrotrons - High Power Microwave and Millimeter Wave Technology. Springer-Verlag, Berlin-Heidelberg, Germany, (2004).

[42] D. Wagner, M. Thumm, G. Gantenbein, W. Kasperek, and T. Idehara, "Analysis of a complete gyrotrons oscillator using the scattering matrix description". *Int. J. Infrared and Millimeter Waves*, vol. 19, no. 2, (1998).

[43]. You Zhou and Ying Tan. "GPU-based Parallel Particle Swarm Optimization. Evolutionary Computation,2009. CEC'09". IEEE Congress on 18-21 May 2009 Page(s):1493 – 1500.

[44]. Man Leung Wong. "Parallel multi-objective evolutionary algorithms on graphics processing units". In GECCO '09: *Proceedings of the 11th annual conference companion on Genetic and evolutionary computation conference*, pages 2515–2 522, New York, NY, USA, 2009. ACM.