Solving Container Loading Problem Using Improved Genetic Algorithm

Pragya Gupta
Department Of Computer Science and Engineering
SSCET, Bhilai

Rajesh Tiwari
Associate Professor, Department Of Computer Science and Engineering
SSCET, Bhilai

Abstract

In this paper we present Elitism based Compact Genetic Algorithm to solve three dimensional bin packing or Container Loading Problem. The three dimensional bin packing problem is the problem of orthogonally packing of set of boxes into a minimum of three dimensional bin. This algorithm uses a probability vector to represent the bit probability of 0 and 1 and model the distribution of generation. Unlike previous works which concentrates on using either a heuristic rule or an optimization technique to find an optimal sequence of the packages which must be loaded into the containers, the proposed heuristic rule is used to partition the entire loading sequence into a number of shorter sequences. Each- partitioned sequence is then represented by a species member. The procedure involves the use of a heuristic rule and a genetic algorithm search. The heuristic rule used covers a scheme which involves a classification of the packages into three distinct groups: large-sized, medium-sized and small-sized package groups.

1. Introduction

The fundamental aim of bin packing is to pack a collection of objects into well-defined regions called bins, so that they do not overlap. In the real world, the critical issue is to make efficient use of time and space. In computational complexity theory, the bin packing problem is a combinatorial NP-hard problem. In it, objects of different volume must be packed into a finite number of capacity V in a way that minimizes the number of bins and do not overlap. The bin packing problem can also be seen as a special case of cutting stock problem. When the number of bin is restricted to 1and each item is characterized by both a volume and a value, the problem of maximizing the value of items that can fit in the bin is known as the knapsack problem.

In the manufacturing and distribution industries packing items into boxes or bins is an important material handling activity. According to the typology introduced by Wäscher etal, a large variety of different bin packing problems can be distinguished, depending on the size of items shape of items and the capacity of bins. There are many variations of this problem, such as 2D packing, 3D packing, linear packing, packing by weight, packing by cost, and so on. They have many applications, such as filling up containers, loading trucks with weight capacity, creating file backup in removable media and technology mapping in Field-programmable gate array semiconductor chip design. Over the last three decades, the bin-packing problem has been studied by researchers in various forms. Research in this area began with the classical one dimensional bin-packing problem, which served as a foundation for the analysis of approximation algorithms. It was one of the first combinatorial optimization problems for which performance guarantees were investigated. Since then, the problem has been broken down into several different versions based on various factors such as geometry of the objects, number of bins, nature of the problem and its constraints. All of these versions are very different from each other except for one common property - they all contain a capacity constraint. The bin or bins that need to be packed have a finite capacity that cannot be exceeded.

Two dimensional bin packing problem is concerned with packing different sized objects (most commonly rectangles) into fixed sized, two-dimensional bins, using as few of the bins as possible. Applications of this type of problem are often found in stock cutting examples, where quantities of material such as glass or metal, are produced in standard sized, rectangular sheets. Demands for pieces of the material are for rectangles of arbitrary sizes, no bigger than the sheet itself. The problem is to use the minimum number of standard sized sheets in accommodating a given list of required pieces. Three dimension bin

packing problem is an extension of two dimension bin packing problem. The three-dimensional bin packing problem (3BP) consists of determining the minimum number of three-dimensional rectangular containers (bins) into which a given set of n three dimensional rectangular items (boxes) can be orthogonally packed without overlapping.

Three dimensional packing problems have been studied in the literature by means of many different aspects of application and are differentiated in several ways. However, since these problems occur in different domains, few works have directly dealt with the container loading. The container loading problems have been classified as knapsack and bin-packing problems in the literature. In the knapsack version, the container space available is fixed and packing all the boxes may not be possible. The objective of the knapsack problem is generally to maximize the packed volume. However, bin-packing problem tries to minimize the required container costs. Many computer-assisted approaches have been developed to solve the bin packing problem. Regarding waste space minimization for one to three dimensional bin packing problems, exact solution methods based on branch and bound procedure have been applied to solve the problem. Besides the branch and bound method, heuristic approaches are also widely used for solving the bin packing problem. The heuristic methods are particularly useful for problems with a high complexity, for which deterministic methods like branch and bound approach are often unable to find the solution within a reasonable amount of time. Metaheuristics such as evolutionary algorithms simulated annealing and tabu search have also been widely applied to solve the combinatorial bin packing optimization problem. Among these methods, the evolutionary algorithm is usually hybridized with a heuristic placement approach for solving the bin packing problem. The sequence of the item to be packed is usually constructed as a sequence chromosome, which is decoded via a packing heuristic for generating a packing plan to be evaluated against the objective function to obtain the quality of the solution. This paper presents the use of a elitism based compact genetic algorithm for solving a three dimensional bin packing problem. Unlike previous works which concentrates on using either a heuristic rule or an optimization technique to find an optimal sequence of the packages which must be loaded into the containers, the proposed heuristic rule is used to partition the entire loading sequence into a number of shorter sequences. Each- partitioned sequence is then represented by a species member.

2. Related Work

This section gives an overview of some of the approaches and optimization techniques that have been used in the past for the two and three-dimensional bin-packing problem.

Due to the nature of the complexity of the problem, relatively less work has been done in analyzing bounds when compared to the actual development of heuristics and algorithms for the problems in two and three dimensions. Among the theoreticians in this field, Silvano Martello and Daniel Vigo have been prominent researchers in the areas of numerical simulation and combinatorial optimization. Their work focused on packing rectangular shapes into the least number of bins. They have presented a lower bound for two-dimensional bin-packing problems with rectangular shapes that may be rotated by 90 degrees. They have proved that the worst-case performance ratio of the sum of the area of the packed rectangles to the area of the container is 1/4. A branch and bound algorithm was used to test the effectiveness of the lower bound. Albano and Sapuppo resorted to heuristic search methods used in artificial intelligence to optimize the layout of irregular shaped two-dimensional patterns on large stock sheets. They developed a deterministic solution to the allocation problem using the A* heuristic search method. A simple set of rules was used to place the patterns on the sheet metal. After a pattern was placed, a profile that separated the available space from the occupied and wasted space was generated. This profile aided in the placement of the next pattern. For the n remaining patterns, k orientations were sampled on the current profile and of the n x k possibilities, a fixed number of successors were chosen based on the least amount of wasted space. Information of the chosen successors was maintained in the form of a directed graph where the edges represented the amount of wasted space and the nodes represented the patterns in particular orientations. Another heuristic search method was explored by Robert Mcgee for the online packing of three-dimensional irregular shaped objects into a cylindrical drum. The parts and container were modeled with the help of a voxel data structure. There was no attempt made to optimize the packing order of the objects. The objective of the heuristic was only to minimize the void space and trapped space that was created from each placement of an object in the container. Void space was defined as the space directly placement of an object in the container. Void space was defined as the space directly below the object just placed, that was not already occupied by the previously placed objects. The space between the object just placed and the walls of the container was considered as trapped space if this space contained too few continuous voxels. In order to place the objects into the container with minimal computation, a data structure called a chain code matrix was used to keep track of the surface voxels of the parts in the container. The heuristic used a

brute force method of checking for all possible placements of the object on the chain code matrix surface and with a one voxel translation resolution. For each position, an orientation resolution of 360/theta was used about all three axes of rotation. theta was a user-preset parameter. For each placement, a quick surface interference check was performed to check the feasibility of the placement. If a feasible placement was found, it was checked for stability. The void space and trapped space were then computed if the object was found to be stable. The best placement of all the feasible and stable placements was then chosen based on the least void space and trapped space that it created. Cagan et al. used simulated annealing to optimize a three-dimensional offline bin-packing problem for irregular shapes. The packing problem was formulated as a multi-objective optimization problem. Each item possessed an attractive force based on its distance to the centroid of the container. Penalty forces were given to volumes lying outside the container and to intersecting volumes. These individual forces were then summed up and weighted. The objective of the simulated annealing algorithm was to minimize the weighted sum. An octree data structure was used to model the items and a multi-resolution modeling technique was implemented to reduce the amount of time taken for interference checking. Ono and Watanabe used a genetic algorithm to optimize the usage of sheet metal when arbitrary two-dimensional patterns had to be cut out of it .Their approach used the ordering of the patterns to model the chromosomes. This set the search space to n! where n is the number of patterns. The fitness of the chromosome was evaluated based on a heuristic called the Layout Determining Algorithm (LDA) that was used to find the placement of the patterns on the sheet metal with no mutual overlap. The fitness was a measure of the sheet length used for a particular ordering or chromosome. The LDA moves the pattern along the sheet metal's height and width until it finds a position for which the pattern does not overlap the previously placed patterns.

3. Container Loading Problem

The efficiently packing the goods into onto a distribution pallet or within a restricting box can be modeled as a pallet or container loading problem. Generally, once the products have been manufactured and packaged, the processes that follow would be shipment loading and transportation. During one shipment, a number of containers will be used to hold all sorts of packages, depending upon the shipment order. Usually, the packages themselves would be in different sizes and shapes. Hence, it is highly desirable to be able to load the containers with these packages in such a way that there are as little empty spaces as possible. Such problem is commonly referred to as a container loading problem. In brief, the assumptions regarding the container loading problem which will be explored in this paper can be described as follows.

1 The shape of all packages is rectangular and can be roughly divided into three groups where the packages that belong to the same group have similar sizes. These three package groups are small-sized, medium-sized and large sized package groups.

2 The shape of container is also rectangular where they can hold a number of packages provided that the maximum capacity limit is not exceeded.

3 The packing scheme considered can be described as being orthogonal and guillotineable. **A** packing arrangement is said to be orthogonal when the sides of a package are always parallel with the sides of other neighbouring packages or those of the container. On the other hand, a packing scheme is said to be guillotineable if and only if it can be created by recursively bipartitioning the container volume with straight guillotineable cuts. Each of these cuts separates a rectangular volume within the container volume into two likewise rectangular pieces.

4 It is possible to arrange the packages into a number of layers within each container where each layer of packages will be in a left-bottom justified format. In other words, it is assumed that there is a horizontal boundary hyper-plane between two consecutive layers of packages.

5 Within the same layer of packages stored inside the container, the packages will be arranged into a front-left justified format.

6 The loading process will start from the loading of the large-sized packages. After all large-sized packages have been loaded, the loading process will continue with the loading of medium-sized packages and small-sized packages, respectively. The loading process will continue until all packages are loaded into the containers.

With this form of the loading process, the complete loading sequence can be viewed as a series of three loading sequences being joined together. In addition, any attempts on reshuffling the orders of the packages from the same package group in each sequence can be viewed as a sequencing problem. This reduces the container loading problem into three sequence-based sub problems. With this treatment on the loading process, the problem considered has been formulated into a simplified problem which has a structure suitable for the optimization using the elitism based compact genetic algorithm. The background on the elitism based compact genetic algorithm and how it can be applied to the container loading problem will be given in the following section.

4. Genetic Algorithm

Genetic algorithm is an efficient searching tool that was invented by John Holland. Genetic Algorithm is started with a set of solutions (represented by chromosomes) called population. Genetic algorithm in general, it's used to find a maximum or minimum value of a given function using the concept of biological chromosomes and genes. In GA any of the solve methods is shown with a list of parameters called chromosomes. A collection of chromosomes called population. GA uses from the information obtained from previous generations for reaching to optimal answer. At each generation the chromosomes are studied and the fitness value is calculated. GA tries to reach to an near optimal answer helping of selection, crossover and mutation. For solving any problem by genetic algorithm, eight components must be defined:

Representation (definition of individual): Representation represents each chromosome in the real world. A chromosome is a set of parameters which define a proposed solution to the problem that the genetic algorithm is trying to solve.

Fitness function: These function shows the fitness of each chromosome. It is used to evaluate the chromosome and also controls the genetic operators.

Population: The role of the population is to hold possible solution.

Parent selection mechanism: The role of parent selection is to distinguish among individuals based on their quality, in particular, to allow the better individuals to become parents of the next generation.

Recombination operators: Recombination operator selects two chromosomes and then produces two new children from them.

Mutation operators: Mutation operator selects one chromosome and then produces one new child from it by a slight change over the parent.

Survivor selection mechanism: The role of survivor selection is to distinguish among individuals based on their quality.

Termination Condition: The condition of ending the running of genetic algorithm.

The initial random solutions may be regarded as the random points in the highly dimensional search space or the fitness landscape of the GA. Since they are randomly generated, it is likely that these points or individuals would cover most of the search space leaving no major segment of the search space as that is regarded as the final solution of the algorithm. blank. The fitness of these individuals is the level in which they are located in the fitness landscape with the fitter individuals lying at the lower levels. The fitness value gives a decent idea regarding the locality of the area in which the individual is found. The different individuals may be regarded as different search agents, which make this problem as a multi-agent system for collaborative search for the global minima. Now the task is to move the agents or the individuals in such a manner that they reach the global minima. A fitter individual or an individual with lower objective value is likely to be found at a place near the global minima. This factor attracts the other individuals towards this individual. As a result the different individuals jump towards the other more fit individuals at the next generation. The weaker individuals may die in the process and stronger ones may produce more individuals in their vicinity. Also the various individuals move by some amount on their own in random directions to look for the possibility of global minima in their vicinity. Again the fitness landscape may be guessed based on the position and fitness values of the agents. Each of these denotes a new possibility of global minima. This process goes on and on. It is highly likely that while traveling the individuals meet the global minima. Towards the end, all individuals converge to some point.

Encoding Of Chromosome

Chromosome in some way stores solution which it represents. This is called representation [encoding] of the solution. There are number of way to represent solution in such way that it is suitable for genetic algorithm [binary, real number, vector of real number, permutations, and so on] and they are mostly depend on nature of problem.

Individual Representation

One of the fundamental design issues is the manner in which a solution is represented to solve the problem in a GA. This is also referred by the term of problem encoding. The basic solution of a problem in its native form is called as the phenotype representation. This may be specific to the problem and presented in a manner that the algorithm working over the problem understands. This needs to be converted in a manner that the GA can work over. This representation is known as the genotype representation of the individual. The most commonly used representations are in form of bit string, numeral vectors or a tree based representation.

Scaling

The individuals in a population pool of the GA have some chance of going to the next generation and surviving in the genetic competitions (Whitley 1989). This depends upon their fitness value, with the fitter individuals having better possibility of going to the next generation. The possibility of surviving and being selected for the next generation is termed as the expectation value of the individual. This value denotes the expectation that an individual may have for survival. A higher expectation value denotes a fitter individual. The task of assigning the expectation value to the various individuals in a population is termed as scaling. The scaling is an important operation in the

GA since a change in scaling methodology might result in the algorithm being more or less biased towards the fitter individuals. Based on these concepts, there are three commonly used scaling mechanisms. These are fitness based scaling or proportional scaling; rank based scaling and top scaling.

Selection

The expected values calculated using the scaling mechanism gives a fair idea of the possibility of the individual to go to the next generation (Goldberg and Deb1991; Chakraborthy 1996). The major task now is to actually select individuals based on the expected values. This is done by the operation of selection. Selection takes into account the expectation values and selects the required number of individuals that participate in the genetic process. The selection follows the Darwin's theory of survival of the fittest. Here only the fittest individuals survive and the others are eliminated from the population pool, or do not survive in the subsequent generations. We discussed the role of the GA's as a search problem in the fitness landscape. Now we know that the different individuals or search agents are located at different locations with different fitness values. Our assumption states the fitness landscape to be relatively simple with not many hills and valleys. In such a context we may assume the agents at locations with high fitness values to be located at un-interesting locations. There would be no use in continuing the search of these agents since they are likely quite away from the global minima. The other agents that have better fitness values are likely to be more close to the global minima. Hence it would be judicious to shift the attention from the agents at poor fitness values to the agents at high fitness values. This means that we kill the agents at the poor fitness value locations and in return produce them at the locations with high fitness value. This job is done by the selection operation.

Crossover

Crossover does the task of recombination of two individuals to generate individuals of the new generation. The new individual carries some of the characteristics of the first parent and the other characteristics from the other parent. The new individual generated in the process may be fitter than the parents or may be weaker. This depends upon the selection of the characteristics by the individual. The selection of good characteristics from both the parents that result in high fitness results in better individuals and vice versa. The fitness of the individual is computed by the overall performance as a combined representation of the entire individual chromosome. Hence nothing certain can be assured regarding the fitness of individual unless it is measured. Crossover results in constant exchange of characteristics in the individual. GA derives a lot of computational optimality as a result of this continuous exchange of characteristics in between the individuals. It may sometimes be viewed as the phenomenon that tries to recombine two individuals hoping that the resulting individual is better in terms of fitness than both parents. The final result of the GA is the fittest individual. It is hence necessary to generate fitter individuals than parents so that the fitness of the best individual of the population pool keeps getting optimized and can finally be returned by the system. In terms of the fitness landscape search, this operation has a major role to play in the convergence of the individuals towards some point. The various individuals lie at various locations in the fitness landscape. The crossover of two individuals results in the generation of an individual at a point in-between the parent's in the fitness landscape. Hence after a complete cycle of crossover operations, the resulting individuals are found somewhere in-between to where the parents were found. This contracts the entire search space of the GA. Again after the next iteration, the new individuals would be generated between the parents. This would further result in contraction of the search space. Hence at each iteration, we cut off a large part of the search space where the global minima cannot lie as per the present fitness values. This contraction of the search space continues and towards the end all individuals lie at almost the same place. Every time the individuals are likely to be attracted more towards the individuals with high fitness values. As the algorithm runs, new areas in the fitness landscape may be explored by the various individuals in their search and the system may escape from the local minima and get to the global minima.

Mutation

Mutation is responsible for the addition of new characteristics into the individual. Crossover alone largely does the continuous exchange of characteristics into the individuals in a population. However the optimality cannot be achieved unless the individuals have new characteristics added to them. This operation is performed by the mutation operator. In this operation we randomly change the characteristics of individuals by some amount governed by the mutation rate. This changes the individual and the new individual that emerges may be fitter or weaker than the parent individual. If the changes applied were good, the new individual is fitter and vice versa. The fitter individuals survive in the genetic process and the weaker ones are eliminated as per the Dwain's survival of the fittest theory. Hence if the added characteristics were good, the other individuals would copy them and these would get replicated in the next generations by the crossover operations. If the characteristics

were not good the individual might die in a few generations and the characteristics might not spread to as many individuals in the population.

These processes are combined to make the complete genetic algorithm as follows.

1. A random population of chromosomes (binary string of 0'1 & 1's) is generated.
2. A fitness value for each chromosome in the population is determined.
3. A biased (based on fitness) random selection of two parents is conducted.
4. The crossover operation is applied to the selected parents.
5. The mutation process is applied to the children.
6. The new population is scanned and used to update the "best" chromosome across the generations.

This process will stop after a fixed number of generations or when the best chromosome has a fitness which exceeds the approximation level.

5. Compact Genetic Algorithm

CGA use a probability vector to represent a bit probability of 0 or 1 and model the distribution of generation. All individuals are generated from the probability vector and each bit of probability vector is initialized to 0.5 when the algorithm started. In each step two individuals are generated from the probability vector. Certain regulations are used to compare the two individuals. The winner of the comparison is responsible for updating the probability vector according to its own gene value. The pseudo code of general compact genetic algorithm is as following.

x is size of population, y is length of solution

Step1. Initialize Probability vector
For i: =1 to y do p [i]:= 0.5;

Step2. Generate two solutions from probability vector
a:= **generate** p[i]; b:= **generate** p[i];

step3. Let them compete
W, L: = **compete** [a , b]

Step4. Update the probability vector
For i: = 1 to y do
If W [i] ≠ L[i] then
If W [i] == 1 then p[i]:=p[i] +1/x;
Else p[i]:= p[i] -1/x;

Step5. Check if the probability vector has converged.
Go to Step2, if it is not satisfied.

Step6. The probability vector represents the final solution.

6. Proposed Elitism Based Compact Genetic Algorithm

proposed elitism based compact genetic algorithm

The main difference between ECGA and traditional parallel genetic algorithms is that ECGA is an island-based GA that operates on probability vectors and exchanges probability vectors between neighbors instead of individual population. Moreover, it is suitable for parallel hardware implementation due to the EC-CGA employs two dimensional array structure and communication occurs only between neighbors. The concept of EC-CGA is to parallelize or divide a large problem into smaller tasks and to solve the task simultaneously using multiple genetic algorithms. EC-CGA operates on probability vectors

Elitism

Elitism provides a means for reducing genetic drift by ensuring that the best chromosome(s) is allowed to pass/copy their traits to the next generation. Through random sampling of the finite population Genetic drift is used to explain/measure stochastic changes in gene frequency. Some genes of chromosomes may turn out to be more important to the final solution than others When the chromosomes representing decision variables that have a reduced “salience” to the final solution do not experience sufficient selection pressure, genetic drift may be stalled. Therefore, it is important to maintain adequate selection pressure, as demanded by the application, in order to avoid this. In other words, the arrest of genetic drift reflects the failure to exert adequate selection pressure by increasing the tournament size or by some form of elitism. Since elitism can increase the selection pressure by preventing the loss of low “salience” genes of chromosomes due to deficient selection pressure, it improves the performance with regard to optimality and convergence of GAs in many cases. However, the degree of elitism should be adjusted properly and carefully because high selection pressure may lead to premature convergence.

The step 2-3 of Compact genetic algorithm should be replaced as follows in Elitism based compact genetic algorithm.

Parameters. E_{chrom} : elite chromosome, N_{chrom} : new chromossome

Step 2. Generate one chromosome from the probability vector
If the first generation then
E_{chrom} := generate(p);
N_{chrom} := generate(p);

Step 3. Let them compete and let the winner inherit persistently
Winner, loser := compete (. E_{chrom} , N_{chrom});
E_{chrom} := winner;

7. Conclusion

In this paper, elitism based compact genetic algorithm has been proposed for solving a 3D container loading problem. The procedure involves the use of a heuristic rule and elitism based genetic

algorithm search. Packages are classified into three groups: large-sized, medium-sized and small-sized package groups. The introduction of this heuristic rule results in the partitioning of the entire loading sequence into three sub sequences where each sub-sequence consists of the loading orders of the packages from the same group. With the use of this sequence partitioning scheme, a search for the best combination of three sub-sequences is then performed by utilizing a elitism based compact genetic algorithm. The search population in this case is composed of three sub-populations where an individual in each subpopulation represents a loading sequence of the packages from the same group.

Acknowledgment

I would like to thank my project guide for guidance and thank H.O.D sir for using Computer lab and thank Director sir for using resources and thank colleague and friends for supporting and thank the anonymous referees for their helpful comments and suggestion that have improved the quality of this manuscript.

REFERENCES

[1] E. Falkenauer. A hybrid grouping genetic algorithm for bin packing. *Journal of Heuristics*, 2(1):5–30, 1996

[2] E. Coffman Jr, M. Garey, and D. Johnson. Approximation algorithms for bin packing: a survey. In D. S. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*, chapter 2, pages 46–93. PWS Publishing Company, Boston, MA, 1996.

[3] A. Lodi, S. Martello, and M. Monaci, "Two-dimensional packing problems: A survey," European Journal of Operational , vol.141, pp.241-252, 2002

[4] D. Pisinger, and M. Sigurd. "the two-dimensional bin packing problem with variable bin sizes and costs". Discrete Optimization ,vol. 2, pp.154-167, 2005.

[5] H. Iima and T. Yakawa. A new design of genetic algorithm for bin packing. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, volume Vol.2, pages 1044 – 1049, 2003.

[6] G. Harik, F. G. Lobo, and D. E. Goldberg, "The compact genetic algorithm," *IEEE Trans. Evol. Comput.*, vol. 3, pp. 287–297, Nov. 1999.

[7] K. Sastry and D. E. Goldberg, "On extended compact genetic algorithm," in *Proc. Late Breaking Papers in Genetic and Evolutionary Computation Conf. (GECCO)*. San Francisco, CA, 2000, pp. 352–359.

[8] I. Correia, L. Gouveia, and F. Saldanha-da-Gama. "Solving the variable size bin packing problem with scretized formulations". Computer and Operations Research , vol. 35, pp. 2103-2113,2008.

[9] G. O. Frank,N. Ntene, and V. V. Jan H, "New and improved level heuristics for the rectangular strip packing and variable-sized bin packing problems". European Journal of Operational Research, vol. 203,pp.306-315, 2010.

Pragya Gupta received the B.E degree in computer science & engineering from the Institute of Technology, Guru Ghasidas Vishwavidyalaya, Bilaspur, India**,** in 2008. And Pursuing M.E from the Shri Shankaracharya Group of Institutions, Bhilai, India.

Rajesh Tiwari has done his M.E. (CTA) from Shri Shankaracharya College of Engg. and Tech.,Bhilai(C.G),India.. Presently he is working as an Associate. Professor in Dept. of CSE in Shri Shankaracharya College of Engg. and Tech.,Bhilai(C.G),India..