Face Detection by Hybrid Genetic and Ant Colony Optimization Algorithm

S.Venkatesan M.E.
Ph.D., Research Scholar/CSE
Anna University of Technology, Coimbatore,
Coimbatore, India

Dr.S.Srinivasa Rao Madane Ph.D.,
Principal & Professor
Priyadarshini College of Engineering,
Vaniyambadi, India

ABSTRACT
Over the last Twenty years, several different techniques have been proposed for computer recognition of human faces. The localization of human faces in digital images is a fundamental step in the process of face recognition. In this paper, a Hybrid algorithm is proposed to detect faces using Ant Colony Optimization and Genetic programming algorithms. Evolutionary process of Ant Colony Optimization algorithm adapts genetic operations to enhance ant movement towards solution state. The algorithm converges to the optimal final solution, by accumulating the most effective sub-solutions.

Keywords
Feature extraction, Genetic Programming, ACOG Algorithm, Ant Colony Optimization .

1. INTRODUCTION
The ant colony optimization algorithm (ACO) is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs. This algorithm is a member of ant colony algorithms family, in swarm intelligence methods, and it constitutes some metaheuristic optimizations. To implement the algorithm it is necessary to get basic ideas of swarm intelligence.

1.1 Operation of a Face Detection System
Most detection systems carry out the task by extracting certain properties (e.g., local features or holistic intensity patterns) of a set of test images acquired at a fixed pose obviously converting into grayscale images (e.g., upright frontal pose) in an off-line setting. To reduce the effects of illumination change, these images are normalized and preprocessed to enhance[13] the images by histogram equalization [9,10] or standardization (i.e., zero mean unit variance) [11]to compensate for the lighting conditions and improve the contrast of the image.. Based on the extracted properties, these systems typically scan through the entire image at every possible location and scale in order to locate faces. The extracted properties can be either manually coded (with human knowledge) or learned from a set of data as adopted in the recent systems that have demonstrated impressive results [9, 10, 11]. In order to detect faces at different scale, the detection process is usually repeated to a pyramid of images whose resolution are reduced to 320 X 420 factor from the original image [9, 10]. Such procedures may be expedited when other visual cues can be accurately incorporated (e.g., color and motion) as pre-processing steps to reduce the search space. As faces are often detected across scale, the raw detected faces are usually further processed to combine overlapped results and remove false positives with heuristics (e.g., faces typically do not overlap in images) [10] or further processing (e.g., edge detection and intensity variance).

Numerous representations have been proposed for face detection, including pixel-based [9, 10], parts-based [6, 4, 7], local edge features [9], Haar wavelets [10] and Haar-like features [11, 10]. While earlier holistic representation schemes are able to detect faces [9, 10], the recent systems with Haar-like features [11, 12] have demonstrated impressive empirical results in detecting faces under occlusion. A large and representative training set of face images is essential for the success of learning-based face detectors. From the set of collected data, more positive examples can be synthetically generated by perturbing, mirroring, rotating and scaling the original face images [9, 10]. On the other hand, it is relatively easier to collect negative examples by randomly sampling images without face images [9,10].

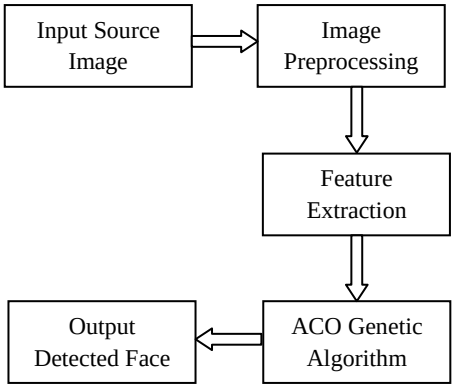

Fig 1. Block Diagram of Face Detection Using the Proposed Method

1.2 Image Enhancement
The face images may be of poor contrast because of the limitations of the lighting conditions. So histogram equalization is used to compensate for the lighting conditions and improve the contrast of the image. Let the of a digital face image consists of the color bins in the range [0,C<=1] , where ri is the i-th color bin, pi is the number of pixels in the image with that color bin and n is the total number of pixels in the image. For any r in the interval [0, 1], the cumulative sum of the bins provides with some scaling constant. Histogram equalization is performed by transforming the function s=T(r), which produces the mapping with the allowed range of pixel values, i.e., a level s for every pixel value r in the original image and 0 <= T (r) <= 1 for 0<= r <= 1.

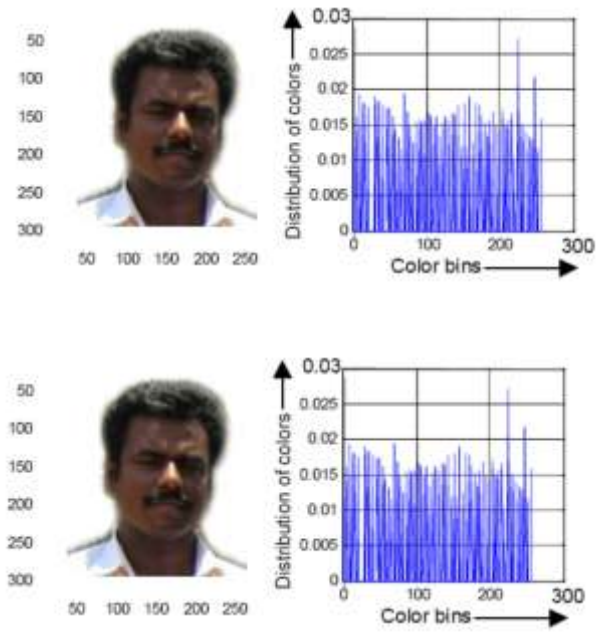

Fig. 2.Histogram Equalization of the Image

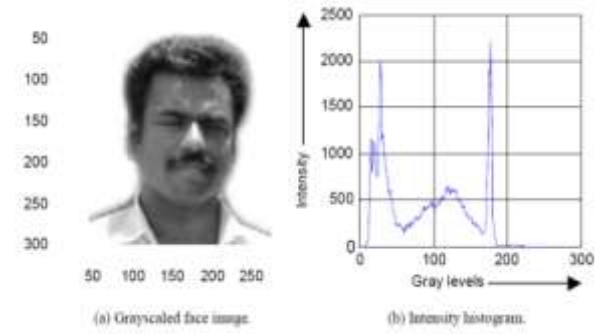

Fig 3.Intensity Histogram of Face

ACO belongs to the class of metaheuristics[3], which are approximate algorithms used to obtain good enough solutions to hard CO problems in a reasonable amount of computation time. Other examples of metaheuristics are tab search simulated annealing, and evolutionary computation. The inspiring source of ACO is the foraging behavior of real ants. When searching for food, ants initially explore the area surrounding their nest in a random manner. As soon as an ant finds a food source, it evaluates the quantity and the quality of the food and carries some of it back to the nest. During the return trip, the ant deposits a chemical pheromone trail on the ground. The quantity of pheromone deposited, which may depend on the quantity and quality of the food, will guide other ants to the food source. Indirect communication between the ants via pheromone trails enables them to find shortest paths between their nest and food sources. This characteristic of real ant colonies is exploited in artificial ant colonies in order to solve CO problems.

Real ant behavior is a great example of intelligent behavior. They are not only capable of finding out the minimum distance from their residing to any food source, but also of reacting in a proper manner to accommodate the environmental changes like addition of extra obstacle in path. For this they deposit a certain amount of pheromone in path while walking. After some time the shortest path becomes rich in pheromones, since ants going on that path will make more rounds in less time than other ants. Every time an ant comes out of its burrow, it chooses the path richest in pheromone. Thus after some time all the ants start walking on this path and other path get faded since pheromone decays with time.

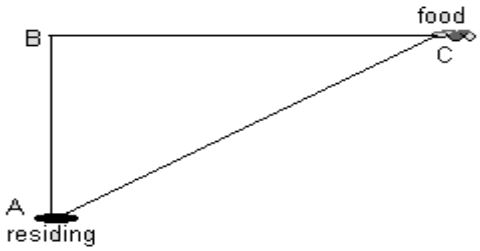

Fig 2. Scenario example

Ant Colony Metaheuristic in AI Robotics is based on this behavior. We take any one path and assume it to be optimum, and if we get any path having better optimization then we replace this path with that one. Finally we get an optimum approximation. central component of an ACO algorithm is a parametrized probabilistic model, which is called the pheromone model. The pheromone model consists of a vector of model parameters T called pheromone trail parameters. The pheromone trail parameters, which are usually associated to components of solutions, have values_*i*, called *pheromone values*. Figure 3 shows the chance of moving an ant to other of the surrounding cells.

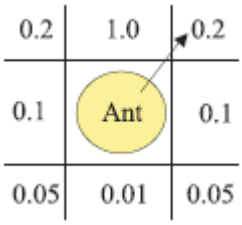

Fig.4. Chance for moving to adjacent cells

The pheromone model is used to probabilistically generate solutions to the problem under consideration by assembling them from a finite set of solution components. At runtime, ACO algorithms update the pheromone values using previously generated solutions. The update aims to concentrate the search in regions of the search space containing high quality solutions. In particular, the reinforcement of solution components depending on the solution quality is an important ingredient of ACO algorithms. It implicitly assumes that good solutions consist of good solution components. 4 To learn which components contribute to good solutions can help assembling them into better solutions. In general, the ACO approach attempts to solve an optimization problem by repeating the following two steps: candidate solutions are constructed using a pheromone model, that is, a parameterized probability distribution over the solution space. The candidate solutions are used to modify the pheromone

values in a way that is deemed to bias future sampling toward high quality solutions.

2. BACKGROUND AND RELATED WORK

Genetic Algorithms (GA) have been used to evolve computer programs for specific tasks, and to design other computational structures. The recent resurgence of interest in AP with GA has been spurred by the work on Genetic Programming (GP). GP paradigm provides a way to do program induction by searching the space of possible computer programs for an individual computer program that is highly fit in solving or approximately solving the problem at hand. The genetic programming paradigm permits the evolution of computer programs which can perform alternative computations conditioned on the outcome of intermediate calculations, which can perform computations on variables of many different types, which can perform iterations and recursions to achieve the desired result, which can define and subsequently use computed values and subprograms, and whose size, shape, and complexity is not specified in advance.

GP use relatively low-level primitives, which are defined separately rather than combined a priori into high-level primitives, since such mechanism generate hierarchical structures that would facilitate the creation of new high-level primitives from built-in low-level primitives. Unfortunately, since every real life problem are dynamic problem, thus their behaviors are much complex, GP suffers from serious weaknesses random systems. Chaos is important, in part, because it helps us to cope with unstable system by improving our ability to describe, to understand, perhaps even to forecast them. Ant Colony Optimization (ACO) is the result of research on computational intelligence approaches to combinatorial optimization originally conducted by Dr. Marco Dorigo[1][2], in collaboration with Alberto Colorni and Vittorio Maniezzo. The fundamental approach underlying ACO is an iterative process in which a population of simple agents repeatedly construct candidate solutions; this construction process is probabilistically guided by heuristic information on the given problem instance as well as by a shared memory containing experience gathered by the ants in previous iteration. ACO has been applied to a broad range of hard combinatorial problems. Problems are defined in terms of components and states, which are sequences of components. Ant Colony Optimization incrementally generates solutions paths in the space of such components, adding new components to a state. Memory is kept of all the observed transitions between pairs of solution components and a degree of desirability is associated to each transition depending on the quality of the solutions in which it occurred so far. While a new solution is generated, a component y is included in a state, with a probability that is proportional to the desirability of the transition between the last component included in the state, and y itself. The main idea is to use the self-organizing principles to coordinate populations of artificial agents that collaborate to solve computational problems. Self-organization is a set of dynamical mechanisms whereby structures appear at the global level of a system from interactions among its lower-level components. The rules specifying the interactions among the system's constituent units are executed on the basis of purely local information, without reference to the global pattern, which is an emergent property of the system rather than a property imposed upon the system by an external ordering influence. For example, the emerging structures in the case of foraging in ants include spatiotemporally organized networks of pheromone trails. The aim of this work is to enhance the ability of ACO by using GP technique.

3. GENETIC PROGRAMMING

Some specific advantages of genetic programming are that no analytical knowledge is needed and still could get accurate results. GP approach does scale with the problem size. GP does impose restrictions on how the structure of solutions should be formulated. There are several variants of GP, some of them are: Linear Genetic Programming (LGP), Gene Expression Programming (GEP), Multi Expression Programming (MEP), Cartesian Genetic Programming (CGP), Traceless Genetic Programming (TGP) and Genetic Algorithm for Deriving Software (GADS). Cartesian Genetic Programming was originally developed by Miller and Thomson for the purpose of evolving digital circuits and represents a program as a directed graph. One of the benefits of this type of representation is the implicit re-use of nodes in the directed graph. Originally CGP used a program topology defined by a rectangular grid of nodes with a user defined number of rows and columns. In CGP, the genotype is a fixed-length representation and consists of a list of integers which encode the function and connections of each node in the directed graph. The genotype is then mapped to an indexed graph that can be executed as a program. In CGP there are very large numbers of genotypes that map to identical genotypes due to the presence of a large amount of redundancy. Firstly there is node redundancy that is caused by genes associated with nodes that are not part of the connected graph representing the program. Another form of redundancy in CGP, also present in all other forms of GP is, functional redundancy. Simon Harding and Ltd introduce computational development using a form of Cartesian Genetic Programming that includes self-modification operations.

The interesting characteristic of CGP is:
1. More powerful program encoding using graphs, than using conventional GP tree-like representations, the population of strings are of fixed length, whereas their corresponding graphs are of variable length depending on the number of genes in use.
2, Efficient evaluation derived from the intrinsic feature of sub graph-reuse exhibited by graphs.
3. Less complicated graph recombination via the crossover and mutation genetic operators.

4. FEATURE EXTRACTION

Roughly speaking, background, object, edges, pixel intensity and noises are what constitute an image. Face detection by genetic algorithm is in essence to classify the different contents into different classes. Where, features are primary elements, which must be representative and comprehensive. Feature extraction, which influences not only the representation of image information but also the accuracy and efficiency of the subsequent algorithm, is significant in face detection. By analyzing the image, a feature extraction method is proposed.

4.1 Proposed ACO Genetic Algorithm (ACOG)

A combinatorial optimization problem is a problem defined over a set C = c1....cn of basic components. A subset S of components

represents a solution of the problem; $F \subseteq 2^c$ is the subset of feasible solutions, thus a solution S is feasible if and only if $S \in F$. A cost function z is defined over the solution domain, $z : 2^C \dashrightarrow R$, the objective being to find a minimum cost feasible solution S*, i.e., to find S*: $S^* \in F$ and $z(S^*) \leq z(S)$, $\forall S \in F$. They move by applying a stochastic local decision policy based on two parameters, called trails and attractiveness.

By moving, each ant incrementally constructs a solution to the problem. The ACO [14] system contains two rules:

1. Local pheromone update rule, which applied whilst constructing solutions. 2. Global pheromone updating rule, which applied after all ants construct a solution. Furthermore, an ACO algorithm includes two more mechanisms: trail evaporation and, optionally, daemon actions. Trail evaporation decreases all trail values over time, in order to avoid unlimited accumulation of trails over some component. Daemon actions can be used to implement centralized actions which cannot be performed by single ants, such as the invocation of a local optimization procedure, or the update of global information to be used to decide whether to bias the search process from a non-local perspective. At each step, each ant computes a set of feasible expansions to its current state, and moves to one of these in probability. The probability distribution is specified as follows. For ant k, the probability of moving from state t to state n depends on the combination of two values: the attractiveness of the move, as computed by some heuristic indicating the priori desirability of that move; the trail level of the move, indicating how proficient it has been in the past to make that particular move: it represents therefore an a posteriori indication of the desirability of that move.

5. ACOG ALGORITHM

An ACOG is differing from previous algorithm. It uses genetic programming to enhance performance. It consists of two main sections: initialization and a main loop, where Genetic Programming is used in the second sections. The main loop runs for a user defined number of iterations. These are described below:

1. Initialization:

Set initial parameters that are system: variable, states, function, input, output, input trajectory, output trajectory. Set initial pheromone trails value. Each ant is individually placed on initial state with empty memory.

2. While termination conditions not meet
   do
   Construct Ant Solution:

Each ant constructs a path by successively applying the transition function the probability of moving from state to state depend on as the attractiveness of the move, and the trail level of the move.

Apply Local Search

Best Tour check: If there is an improvement, update it.

Update Trails:

A. Evaporate a fixed proportion of the pheromone on each road.

B. For each ant perform the "ant-cycle" pheromone update.

Reinforce the best tour with a set number of "elitist ants" performing the "ant-cycle"

Table 1 Parameter settings [12]

Chromosome Length	32 bits
Population Size	150
Number of Generation	300
Cross over probability	0.7
Mutation Probability	0.01

Initial Population: Generate randomly a new population of chromosomes of size N: x1, x2,…..x_n.Assign the cross over probability P_C and the mutation Probability P_M.

Evaluate the Fitness function for each chromosome in the population.

Fitness Function: To determine where a selected region is a face or not a function need to assign a degree of fitness to each chromosome in every generation. The fitness of a chromosome is defined as the function of the difference between the intensity value of the input image and that of the template image measured for the expected location of the chromosome. That is for each chromosome n, fitness function is defined as [12]

$$f(n) = 1 - \frac{\sum_{(x,y) \in W} \left| f(x,y) - f_{n,t}(x,y) \right|}{B_{\max} \times xSize \times ySize}$$

where B_{max} is the maximum brightness of the image, *xSize* and *ySize* are the number of pixels in the horizontal and vertical directions of the image, *W* is the window, f *and* $f_{n,t}$ are the intensity values of the original image and the template image when it is justified for the *n*-th position of the chromosome, respectively.

Selection: Select a pair of chromosomes for mating use the roulette wheel selection procedure, where each chromosome is given a slice of a circular roulette wheel. The area of the slice within the wheel is equal to the chromosome fitness ration obviously the highly fit chromosomes occupy the largest areas, where the chromosomes with least fit have much smaller segments in the wheel. To select chromosome for mating a random number is generated in the interval [0.100], and the chromosome whose segment spans the random number is selected.

Cross over: Produce two offspring from two parent chromosomes. Cross over operator chooses a crossover point where two parent chromosomes break and then exchanges the chromosomes parts after that point. As a result two offspring are generated by combining the partial features of two chromosomes. If a pair of chromosomes does not takes place, and the offspring are created as exact copies of each point. This research employs single point cross over, two point cross over and uniform cross over operators. The crossover points are selected randomly within the chromosome for exchanging the contents.

Mutation: Apply the conventional mutation operation to the population with a mutation rate $P_{M.}$ For each chromosome generate a random value between [0,1].If the random value is less than P M choose a bit at a random location to flip its value from 0 to 1 or 1 to 0. The parameter setting approach is shown in Table 1.

By applying the above operation, based on pheromone trails. The operations are applied to individual(s) selected from the population with a probability based on fitness.
End While

The Performance of Genetic Process:

Genetic generation process involves probabilistic steps, because of these probabilistic steps, non convergence and premature convergence, i.e. convergence to a globally sub-optimal result, problems become inherent features of genetic generation process. To minimize the effect of these problems, multiple independent runs of a problem must be made. Best-of-run individual from a ll such multiple independent runs can then be designated as the result of the group of runs. If every run of GPG were successful in yielding a solution, the computational effort required to get the solution would depends primarily on four factors: population size, M, number of generation that are run, g, (g must be less than or equal to the maximum number of generation G) the amount of processing required for fitness measure over all fitness cases, and the amount of processing required for test phase e, we assume that the processing time to measure the fitness of an individual is its run time, P. If success occurs on the same generation of every run, then the computational effort E would be computed as follows:

$$E= (M \bullet g \bullet \beta \bullet e)$$

Since the value of e is too small with respect to other factors, we shall not consider it. However, in most cases, success occurs on different generations in different runs, then the computational effort E would be computed as follows:

$$E= (M\bullet g\ avr \bullet \beta)$$

Where gavr is the average number of executed generations Since GPG is a probabilistic algorithm, not all runs are successful at yielding a solution to the problem by generation G. Thus, the computational effort is computed in this way, first determining the number of independent runs R needed to yield a success with a certain probability. Second, multiply R by the amount of Processing required for each run, that is. The number of independent runs R required to satisfy the success predicate by generation i with a probability z which depends on both z and P (M, i), where z is the probability of satisfying the success predicate by generation i at least once in R runs defined by:

$$z = 1 - [1- P (M, i)] R$$

P (M,i) is the cumulative probability of success for all the generations between generation 0 and generation i. P (M, i) is computed after experimentally obtaining an estimate for the instantaneous probability Y (M, i) that a particular run with a population size M yields, for the first time, on a specified generation i, an individual is satisfying the success predicate for the problem]. This experimental measurement of Y (M, i) usually requires a substantial number of runs. The computational effort E, is the minimal value of the total number of individuals that must be processed to yield a solution for the problem with z probability (ex: z = 99%):

$$E= M \bullet (\bullet g + 1) \bullet \beta \bullet R$$

Where •g is the first generation at which minimum number of individual evaluation is produced, it is called best generation. •g value is incremented by one since generation •g must also run to reach the solution. From the above equation computational effort depends on the particular choices of values for M, G, P (M, i), and the effort required for fitness evaluation, hence, the value of E is not necessarily the minimum computational effort possible for the problem.

6. EXPERIMENTAL RESULTS

6.1 A Speed of the detector

The AdaBoost-based face detector demonstrated that faces can be fairly reliably detected in real-time (i.e., more than 15 frames per second on 320 by 240 images with desktop computers) under partial occlusion. While Haar wavelets were used for representing faces and pedestrians, they proposed the use of Haar-like features which can be computed efficiently with integral image. Despite the excellent run-time performance of boosted cascade classifier, the training time of such a system is rather lengthy. In addition, the classifier cascade is an example of degenerate decision tree with an unbalanced data set. Numerous algorithms have been proposed to address these issues and extended to detect faces in multiple views.

There are numerous metrics to gauge the performance of face detection systems, ranging from detection frame rate, false positive /negative rate, number of classifier, number of features, number of training images, training time, accuracy and memory requirements. In addition, the reported performance also depends on the definition of a "correct" detection result. Several post-processing algorithms have been proposed to better locate faces and extract facial features.

This method particularly well adapted to real time applications. In fact, the computed model needs few operations to be applied on an image. Previous works show that the elapsed time for face detection is about 50 seconds. In our approach we try to reduce this to a greater extent and also to increase the true positive rate than Haar wavelet based method adaboost training method.

6.2 Statistical Results

The proposed ACOG algorithm detects the set of test images of size 320 x 240 resolution in Intel Dual Core processor of 2.8 GHz PC using MATLAB 7.9 at an average speed of 5.2 secs. The proposed algorithm excels at 8 out of 10 test images better than the Adaboost, and aar Wavelet based Training Figure 4 shows the graphical comparison of detection algorithms plotting test images in x axis and run time in seconds in y axis.

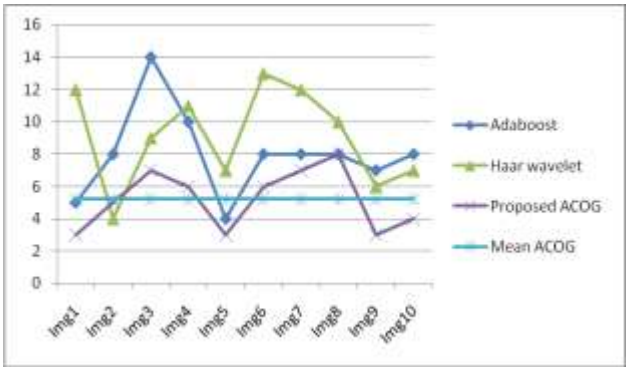

Fig. 5 Test Images vs Run Time

7. CONCLUSION

The effectiveness of the Face detection algorithm has been tested both in simple and complex background for different types of face and non face images of size 320X420 resolution. This

algorithm is capable of detecting the faces in the images with different backgrounds. A rotated human face can also be detected even if the face is under shadow, wearing glasses, or under bad lighting conditions. The performance of the proposed algorithm is up to the expectation and results are experimented with several test images and found to detect faces statistically best than the existing face detecting algorithms.

8. REFERENCES

[1] M. Dorigo,V. Maniezzo, A. Colorni, Ant system: optimization by a colony of cooperating agents, IEEE Trans. Systems, Man, Cybernet.-Part B 26 (1) 29–41.

[2] M. Dorigo, T. Stützle, Ant Colony Optimization, MIT Press, Cambridge, MA, 2004.

[3] C. Gagné,W.L. Price, M. Gravel, Comparing an ACO algorithm with other heuristics for the single machine scheduling problem with sequence-dependent setup times,J. Oper. Res. Soc. 53 (2002) 895–906.

[4] L.M. Gambardella, M. Dorigo, Ant colony system hybridized with a new local search for the sequential ordering problem, INFORMS J. Comput. 12 (3) (2000) 237–255.

[5] F. Glover, Tabu search—Part I, ORSAJ. Compute. 1 (3) (1989) 190–206. [31] F. Glover, Tabu search—Part II, ORSAJ. Comput. 2 (1) (1990) 4–32.

[6] F. Glover, G. Kochenberger (Eds.), Handbook of Metaheuristics, Kluwer Academic Publishers, Norwell, MA, 2002.

[7] M. Guntsch, M. Middendorf, Pheromone modification strategies for ant algorithms applied to dynamic TSP, in: E.J.W. Boers, J. Gottlieb, P.L. Lanzi, R.E. Smith, S. Cagnoni, E. Hart, G.R. Raidl, H. Tijink (Eds.), Applications of Evolutionary Computing: Proc. EvoWorkshops 2001, Lecture Notes in Computer Science, Vol. 2037, Springer, Berlin, Germany, , pp. 213–222.

[8] W.J. Gutjahr, A generalized convergence result for the graph-based ant system metaheuristic, Tech. Report 99-09, Department of Statistics and Decision Support Systems, University of Vienna, Austria.

[9] Sung, K.K., Poggio, T.: Example-based learning for view-based human face detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(1) (1998) 39–51.

[10] Rowley, H., Baluja, S., Kanade, T.: Neural network-based face detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(1) (1998) 23–38.

[11] Viola, P., Jones, M.: Robust real-time face detection. International Journal of Computer Vision 57(2) (2004) 137–154.

[12] Md.Golam Moazzam and Md. Al-Amin Bhuiyan:A Novel Approach for Human Face Detection Using Genetic Algorithm

[13] S.Venkatesan and M.Karnan:Advvnced Classification using Genetic Algorithm and Image Segmnetation For Improved Face Detection.,computer research and Developemnt 2010 second InternationalConference(ICCRD)7-10 May2010 on Page364-368

[14] S.Venkatesan and M.Karnan: Edge and Characteristics Subset Selection in images using ACO ,Computer research and Developemnt 2010 Second International Conference (ICCRD)7-10 May 2010 on Page 369-372

9. AUTHORS

Dr.S.Srinivasa Rao Madane Principal and Professor Department of Computer Science & Engineering in Priyadarshini Engineering Vaniyambadi Tamilnadu India.His Area of Interest includes Neural Networks, Image processing, Analog and Digital communication

S.Venkatesan Pursuing Ph.D., in Department of Computer Science and Engineering in Anna University of Technology Coimbatore, Tamilnadu India. His area of interest includes Image Processing, Soft Computing, Pattern Recognition, and Optimization Techniques.