Comparison of Swarm Optimization and Memetic Algorithm for Systolic Mapping of Texture Analysis

Bagavathi C and Dhivyadevi R
Department of ECE
Sri Krishna College of Engineering and Technology, India
bagavathic@skcet.ac.in, dhivyadevi.r@kpriet.ac.in

Siddharthraju K
Department of ECE
KPR Institute of Engineering and Technology, India
k.siddharthraju@kpriet.ac.in

Dinesh P
Managing Partner
SAI Incubation Centre, India
osmdinesh@gmail.com

Abstract - Systolic processors offer a hardware design which can accommodate more functions in a small footprint. Hardware utilization efficiency can be enhanced by appropriately designating the intended hardware with a task in space and time through parallel computing platforms. Regular algorithms known for their computational complexity can be mapped to systolic array by dependence graphs, which allot hardware to the design data. Manual mapping techniques tend to be tedious with more inaccuracy and calls for efficient mapping techniques, automated through algorithmic procedures. Texture Analysis marks the preliminary progression of image analysis and interpretation. Automotive systems, Robotics, Industrial processing and similar automated applications can be simplified through texture analysis. This work deals with employing evolutionary algorithms for mapping texture analysis onto systolic architecture. Memetic Algorithms (MA) and Particle Swarm Optimization (PSO) algorithms were comparatively studied and the efficiency of designing a parallel architecture through systolic array is analyzed through cost function and processing time.

Keywords - Systolic arrays, Parallel computing, Mapping Texture Analysis, Memetic algorithm, Particle swarm algorithm

1. Introduction
Current trends have an escalating upscale in development through advancements in design procedures and available resources. Recent improvements in designing and manufacturing demand low power, minimum hardware, high speed and good accuracy. Parallel designs that offer high throughput is an apt choice but results in a memory access conflict in a hierarchical database. By designing a proper memory fetch strategy, optimized designs of hardware and software can be derived and ported to an application-specific instruction-set (ASIP), dedicated Field Programmable Gate Arrays (FPGA) [1] or an Application Specific Integrated Circuit (ASIC) processor.

Hardware algorithms are hardwired techniques with a fixed set of processors and interconnections. Iterative procedure on hardware algorithms requires a huge area of design units and high computation time. Reusing the available hardware with proper scheduling methods to obtain a compact regular structure would be a better solution. Systolic arrays are processor units that can be designed and scheduled appropriately to compute a train of sequences through complete utilization of the available hardware. The architecture is primarily developed to increase hardware reliability, cost-efficiency and accuracy through parallelism and pipelining on standard image processing algorithms. Iterative algorithms run similar operations over various inputs and can be parallelized through systolic implementation. Texture analysis is an iterative image classification method known to improve the perception of the image through mean and variance analysis over a shifting window. Texture analysis is analyzed based on feature-based evaluation metrics such as contrast, coarseness, busyness and strength. Mapping a regular texture analysis algorithm to a systolic processor proceed by selecting the allotment of an atomic computation to a processor and managing the time corresponding to scheduling successive processes on the available hardware [2].

Emergent complexity is a behavior where larger entities arise through interactions among smaller entities and this phenomenon has been the basement for Evolution. Existing species evolve and develop necessary characteristics through an expensive process of increase in complexity. Evolutionary algorithms (EA) define the process of improvement of a group of individuals through variations applied to size of population and impact of reproduction mechanism. These variations had lead to a new class of meta-heuristic algorithms grouped under a domain called ‘Bio-inspired Mechanisms’ that work to solve a heuristic problem in a random manner. Applications of evolutionary methods are vast and one such use of EA is to find the Efficient mapping procedure for systolic arrays. From a huge set of available solutions
for mapping, the best solution is selected by ensuring
constant improvement in quality and quality of each
solution being analyzed through a cost function.
Evolutionary algorithms for processor mapping have
become popular through its robust nature of ensuring best
result in every run. This thesis concerns the application of
evolutionary algorithms such as genetic algorithms [3],
memetic algorithm [4] and the optimisation of particulate
swarms [5] in a systolic way.

The paper is organized as follows: Section 2 deals with
the elucidation of systolic architecture and mapping
procedure for systolic arrays, Section 3 enlists different
evolutionary algorithms, Section 4 with texture analysis,
an example to implement systolic arrays, Section 5
analyses the results of applying evolutionary algorithms
for mapping systolic architecture for texture analysis.

2. Systolic Architecture
Systolic arrays are a group of processors that work on a
controlled flow of data through synchronized input and
output. Systolic arrays were first named by [6]. The
maiden research dealt with the implementation of
processors that are similar in structure and function. The
design units are regulated and controlled by a control
system in terms of computation and data storage. The
performance of a systolically designed system can be
measured by the throughput of a peripheral device which
feeds on the system. A highly concentrated architecture
demands high efficiency and low demand for supporting
organizations that supply data from local connections [7].
The data is circulated into the systolic array every
computation cycle and the module-based design of systolic
arrays ensures the data fetching, data processing and data
forwarding to the next stage of design. This process
improves the throughput and eliminates the need for large
number of processors [8].

Systolic arrays are known for its innate pipelined
structure and qualify as a good parallel computing system.
The array is synchronized with discrete time steps with
sufficient time separation between successive operations.
Unlike forced pipelining in a signal processing chain,
systolic arrays can be extended in multiple spatial
domains. The design units are used less at the start and end
of the feed sequence, they are fully operating on the
remaining data resulting in increased throughput.

Scalability, regular design and reconfigurability are
some features for which systolic arrays are famous for.
The systolic implementation of matrix multiplication [9]
uses \( N_1 \times N_2 \) processors for multiplying two matrices of
size \( N_1 \times N_3 \) and \( N_3 \times N_2 \). The same architecture is
available for multiplying matrices of lesser dimensions
accounting for its scalability. Regular architecture results
from its uniformity in the design of processing units.
Reconfigurability can be interpreted from its
implementation in FPGA providing hardware and software
flexibility.

Systolic arrays allegedly modularize the computations
that require long processing time and power hungry
hardware structures [10].

The basic structure of systolic architecture is
demonstrated in the Figure 1.

![Figure 1: General Architecture for Systolic Array](image)

3. Evolutionary algorithms
Evolutionary algorithms [11] belong to a class of non-
traditional techniques which mimic the biological behavior
of organisms to obtain the solution. Evolutionary
algorithms [12] form the basis of computational
intelligence which interprets the environmental variables
and performs sensibly on population-based
improvement methods. It duplicates the personality of
evolution in various species: group of ants, swarm of birds,
school of fishes, groups of frogs etc [13]. The nature of
self-improvement and discarding unnecessary talents has
been a manner of evolution starting from unicellular
organisms. As shown in Figure 2, EA methods try to
design a copycat mechanism of natural evolution [14].
An example to appreciate evolution is the disappearance of coccyx or tailbone from humans due to its least use for the species. Evolutionary algorithms are of four categories: Genetic algorithm (GA), Genetic Programming (GP), Evolutionary Programming (EP) and Evolutionary Strategy (ES) [15]. The scope of this work is limited to Genetic algorithm, Memetic Algorithm (MA) and Particle Swarm Optimization (PSO).

Evolutionary algorithms are best suited for systolic arrays because of their inherent nature of parallelism [16],[17]. Efficient turbulence handlings as the algorithm starts with random candidates of a solution, computational integrity by trying to achieve the required fitness [18], application of domain intelligence rather than a bird’s eye view of a problem are some of the benefits of EA.

Evolutionary algorithms have been employed in various applications such as computer aided-design of electronic circuits [19], face recognition [20], object tracking [21], medical applications [22], optimization problems [23], energy management [24], stochastic applications cite [25], scheduling problems [26][27], machine learning problems [28] and artificial intelligence.

Memetic Algorithm

Memetic algorithms (MA) [29] are group of stochastic search methods that employ evolution and local search strategy. MA was named by Moscato following the notion of Dawkin’s meme. MAs are best suited for a proximate solution of NP hard problems [30]. The outstanding feature of MA is the exploitation of entire knowledge of the problem justifying the nomination of MA to be named as Hybrid Evolutionary algorithms (HEA). Memetic techniques enhance the global search methods of EA by adopting local search methods in a restricted group of individuals. It follows a standard of evolving the individuals from one population to another by processing and enhancing the candidates. This process ensures the attainment of optimum faster through a synergy mechanism of two optimization methods. Any advantageous information available in local area can be used to guide the search to a better solution. Local search methods iteratively examine a set of points in a neighborhood of the current solution and replace the current solution with a better neighbor if one exists. Local search methods in hybridizing EAs are used to improve the information content of a population in one or more phases of implementation. MAs can implement local search to the descendants and skip the mutation process [31].

MAs are similar to GA which forms a group of elements (memes). The procedure involves selection of a group of memes (candidate solutions) and evolution of memes towards the optimal solution by crossover and mutation along with personal experience of the individual memes. The inheritance of favorable traits by descendants is generally encouraged in all evolutionary methods. Memetic algorithm was the first method to introduce the Baldwin’s learning of traits by a candidate of evolutionary process. The inheritance methods of MA can be designed in two ways:

- Lamarckian - Local stage result replaces the least fit individual in the population
- Baldwinian - original individuals of the population are retained when the results approach the outcome of local search

The common method of inheritance is Lamarckian method. The local improvement factor along with information variation when added to MA allows for faster convergence compared to GA. Genetic algorithms suffer from a longer processing time as there is a possibility of gene diversion when the genetic operators are acted upon the population and the search does not take any local improvements. The stopping condition for MA can be total number of iterations before reaching target, number of iterations for which the target value has been s Figure or satisfactory target value. The size of memes generally ranges from 40 to 1000. The number of iterations is usually more than 100 and better results have been observed for more number of iterations [32]. Memetic algorithms have been used in solving optimization problems [33], software testing [34], medical imaging [35], bioinformatics [36], object recognition [37],[38] and scheduling [39],[40].

Particle Swarm Optimization

As shown in Figure 3, Particle swarm optimization [41] is a trajectory evolving algorithm which imitates a flock of birds or bird-like objects (boids) [42] trying to reach the destination [43]. Binary Particle Swarm Optimization (BPSO) [44] is an improvement suggested for PSO with the particles assuming binary values for position. The position $X_p$ and velocity $V_p$ of each particle is given as shown in Equation (1) and Equation (2),

$$X_p = [x_{p1}, x_{p2}, x_{p3}, \ldots, x_{pn}]$$

$$V_p = [v_{p1}, v_{p2}, v_{p3}, \ldots, v_{pn}]$$
The position of a particle is randomly generated. Initial velocity range is $[0.0 \sim 1.0]$. Each particle is updated with its position.

The new position of a particle is updated based on Equation (3), Equation (4), Equation (5) and Equation (6)

\[
V_p(i+1) = v_{inertia} + v_{cognitive} + v_{global}
\]

Where
\[
v_{inertia} = w \times V_p(i)
\]
\[
v_{cognitive} = c_1 \times \text{rand} \times (p_{best} - x_p(i))
\]
\[
v_{global} = c_2 \times \text{rand} \times (g_{best} - x_p(i))
\]

The overall working of the array is based on the matrix (p) and the scheduling matrix (s). The matrix for assigning the processor assigns numerous data to available processors. The scheduling matrix determines how long a certain processor is to enter the next data. The vector (d) specifies the basic processor architecture and defines a collection of data to be interpreted by the same processor. These three vectors define the mapping of an algorithm onto hardware. For the mapping to be feasible, vectors chosen must satisfy space and timing constraints. Processor space vector and projection vector must be orthogonal ensuring that only one set of data at any point of time can be processed at a specific processor. Timing constraint dictates that no two sets of data should be mapped to the same processor at a specific instant of time. Hardware utilization efficiency (HUE) defines the amount of reusability in the defined hardware for a specific projection vector (d) and scheduling vector (s). For an efficient design, HUE should be close to 1.

The search of exact set of vectors obeying the feasibility constraints is a data-intensive process where each location in space and time has to be evaluated iteratively.
The quest for optimum matrix normally includes heuristic [59],[60] which is a general search approach that slopes the search path in the best way possible for immediate goals [61],[62]. Alternative methods such as evolutionary algorithms or more organized methods can be applied to reduce the computation time and resources.

**Image Texture Analysis**

The categorization of or segmentation of textural features in respect of the small component type, density and steadiness direction shall be known as Texture Analysis. In many machine learning and image processing algorithms, analysis of texture features is a mandatory process for applications such as texture classification, denoising, image reconstruction and feature extraction. All texture features based on first-order statistics of the gray value distributions are invariant to the orientation and scale of object. The estimate of the parameters for texture segmentation requires variance and the mean to be averaged over a local neighborhood for normalization. The mean is calculated over a window of $n \times n$, with 'n' being an odd number for convenient selection of pixels averaged over a local neighborhood for normalization.

$$T_v(x,y) = \frac{1}{n^2} \sum_{s=-\frac{n-1}{2}}^{\frac{n-1}{2}} \sum_{t=-\frac{n-1}{2}}^{\frac{n-1}{2}} [g(x+s, y+t) - m]^2$$

$$m = \frac{1}{n} \sum_{s=-\frac{n-1}{2}}^{\frac{n-1}{2}} \sum_{t=-\frac{n-1}{2}}^{\frac{n-1}{2}} g(x+s, y+t)$$

(8)

In a texture analysis problem, the gray scale intensity values $g(x,y)$ of a window are analyzed. To analyze the variations within a region around a point, texture value $T_v(x,y)$ of that point can be computed. The quality of texture analysis applied to an image can be quantified through parameters such as coarseness, contrast, busyness and texture strength [63],[64]. In Equation (8), 'm' denotes the function for mean of gray values of the image within the window of size N. The variables 's' and 't' in Equation (8) represents the extent of window function for calculating texture. The summation is completed when the computation advances in z-direction until it reaches the edge of the image. The result of texture analysis is an edge-detected image which emphasizes on the high intensity variations on the edge.

**Table 1: Parameters for Texture Analysis Evaluation**

| Parameter     | Description                          |
|---------------|--------------------------------------|
| Coarseness    | The basic pattern of a texture       |
| Contrast      | The capacity to clearly differentiate neighboring pixels |
| Busyness      | Rapid intensity changes from one pixel to another |
| Texture       | Easiness of dening a texture from its region |

Parameters such as contrast, coarseness, busyness and strength as mentioned in Table 1 are used to evaluate the resultant image after texture analysis. Texture Processing may also separate image data into information that is easier to view and used in different applications, including industrial automation, imagery retrieval, medical and remote sensing.

**Selection of Best Architecture**

The search technology is used to find the right matrices to reduce costs. The cost function involves the sum and number of cycles used to complete the image size algorithm. In order to emphasize efficient hardware usage, higher weightage is given to the number of processors [65]. When the feasibility constraints are being satisfied, the edge mapping of the algorithm can be derived.

Edge mapping connects the nodes of a software design in the dependence graph to the computational nodes at the physical design. If an edge 'e' exists in the space representation of a dependence graph, then an edge is introduced between the concerned processors with appropriate delay. Edge mapping concludes the systolic array implementation and this work ends with selection of number of processors using methods alternative to heuristics.

In optimization problems, the optimal solution tends to occur at the boundary points of the space limited by the constraints of the problem [66]. From the processor space vector and scheduling vector, the number of processing elements and the total time required for completion of the algorithm can be derived.

$$N_{SPE} = PE_{\text{max}} - PE_{\text{min}} + 1$$

(9)

The Equation (9) specifies the number of scheduled processing elements $N_{SPE}$ needed for effective hardware usage to complete the mission accounting. The maximum and minimal computing sections used to the algorithm are shown in the processor allocation matrix $p$ and the matrix $q$ belonging to a search space S is indicated by $PE_{\text{max}}$ and $PE_{\text{min}}$.

The processors are arranged in the quest (index) space with the Cartesian coordinates as shown in Equation (10).

$$PE_{\text{max}} = \max(p^q, q \in S)$$

$$PE_{\text{min}} = \min(p^q, q \in S)$$

(10)

The maximum and minimal computing sections used to the algorithm are shown in the processor allocation matrix $p$ and scheduling matrix $s$ also the allocation of processors in space as shown Equation (11).

$$N_{\text{cycle}} = \max(s^q(p-q)+1)$$

(11)

The cost function is chosen by giving higher weightage to number of processors to emphasize on hardware efficiency as shown Equation (12).

$$Cost_{\text{function}} = 0.75 \times N_{SPE} + 0.25 \times N_{\text{cycle}}$$

(12)

From the processor space vector and scheduling vector, the number of processing elements and the total time required for completion of the algorithm can be derived.

**5. Results and Discussion**

This work deals with evaluating texture analysis process through defined parameters and by designing the mapping vectors for systolic array through evolutionary mapping process.

Texture analysis is applied to a gray image obtained from a tricolor image. The parameter N implying window width is assigned a value 3, 5 and 7. The different sized images are taken to evaluate the variation of parameters.
based on number of pixels and window size of Texture Analysis.

Complexity of the algorithm increases with increase in number of pixels per window and image size. Texture analysis applied on some standard images as shown in Fig. 5. Since texture analysis is a sequential process, the computations are executed in order and consume design time. Table 2 lists the computational time of texture analysis for various images. When the image size increases, evaluation time of the texture is more. Processing time reduces with increase in window size for smaller images. When the images are large, for a window size of 5, the algorithm takes more time to complete.

Table 2: Evaluation Execution Time Vs Window Sizes (N) of Test Images

| N  | Image (a) | Image (b) | Image (c) | Image (d) | Image (e) | Image (f) |
|----|-----------|-----------|-----------|-----------|-----------|-----------|
| 3  | 1.14 s    | 1.02 s    | 1.19 s    | 1.25 s    | 2.25 s    | 6.73 s    |
| 5  | 1.06 s    | 1.18 s    | 1.57 s    | 1.78 s    | 2.67 s    | 9.69 s    |
| 7  | 1.03 s    | 1.01 s    | 1.49 s    | 1.8 s     | 3.17 s    | 13.1 s    |

It can be deduced that the images of higher pixel dimensions require more execution time for a specified window size and for small images the time reduces with increase in window size. The time increases with increase in window size for images of dimension more than 300 x 300 pixels.

For the chosen images in window sizes 3, 5 and 7, texture parameters such as coarseness, contrast, occupationality and texture intensity are evaluated. Coarseness of an image reduces with increase in number of pixels and depending on the intensity values, the dependence of coarseness on window size becomes unpredictable. Contrast increases with increase in both number of pixels and window size as shown in Figure 4.

Fig. 4: Comparison of Coarseness and Contrast for different window size for Texture Analysis

Busyness and Texture strength are dependent on the intensity values and does not follow a determinable pattern as shown in Figure 5. Depending on the quality of input images, the values vary randomly.

Systolic arrays can be implemented with texture analysis through efficient mapping techniques such as heuristics, memetic algorithm and particle swarm optimization. The mapping procedure is done through coding in MATLAB tool. Texture analysis is a four dimensional problem with two dimensions from the Cartesian coordinates of the image under consideration and other two dimensions from the window of evaluation. The end result of the product is a matrix indicating the processor and time details of the mapped function. The four dimensional texture analysis problem results in two 1 x 4 matrices representing processor matrices in x- and y-direction and one 1 x 4 matrix indicating the scheduling matrix. The scope of this work is limited in finding the algorithm that maps texture analysis problem to a minimum number of processors with time efficiency.

Table 3: Particle Swarm Optimization implementation of Texture Analysis

| Trial | Average Value | Processing Time | Cost function | N_PE | N_cyl |
|-------|---------------|-----------------|---------------|------|-------|
| 1     | 73.3          | 3.569           | 48.5          | 42   | 68    |
| 2     | 312.7         | 3.833           | 47.5          | 38   | 76    |
| 3     | 186.4         | 3.405           | 65.25         | 72   | 45    |
| 4     | 42.5          | 3.494           | 37            | 30   | 58    |
| 5     | 280           | 3.509           | 110           | 14   | 20    |

Original population, location and speed are altered and the solution increases with various iterations [67]. The findings show that the processing time is roughly constant, since there is no sharing of information between people, unlike MA. The optimal result from PSO ended with 30 processors and the cycle for evaluation is 58. The cost function resulted was 37. Memetic algorithm is implemented with the same set of parameters. A meme is a group of individuals and the number of individuals for MA is 40 with 4 individuals per meme as shown in Table 4.
In the computation process, local improvements are given higher weightage in moving the algorithm forward. In Memetic algorithm too, the number of iterations vary over a wide range. The program executed for 0.536 s before converging to an optimal result of 16 processors (4 x 4 cells) in 17 cycles which is the minimum number of processors of all the above mentioned techniques.

### Table 5: Comparison of heuristics, pso and ma

| Algorithm   | Evaluation time | \( N_{FE} \) | \( N_{syst} \) |
|-------------|----------------|---------------|---------------|
| Heuristics  | 1.07 s         | 9 × 9         | 20            |
| PSO         | 1.09 s         | 5 × 6         | 58            |
| MA          | 0.536          | 4 × 4         | 17            |

Systolic mapping technique can be chosen based on the requirement of hardware efficiency and latency. When particle swarm optimization is used, it maps the array with less delay and more processors but results in more number of cycles for the algorithm to conclude. As shown in Table 5, the above analysis cannot be concluded with few algorithms. The results indicate that the memetic algorithm qualifies as a better choice for systolic array implementation of texture analysis as shown in Table 3. As a future work, Algorithms such as Ant colony optimization, Shuffled frog Leap algorithm and Differential Evolution for systolic mapping can be evaluated.

### 6. Conclusion

Systolic arrays are a pristine solution of computationally intensive applications, like real time image processing and applications which require accuracy, such as medical image classification. Texture analysis has been sampled out from a huge set of image processing applications to demonstrate the computational efficiency offered by systolic arrays. Efficient mapping techniques are inevitable for a compact design of parallel arrays that reuse the available hardware to the maximum extent. Of the four discussed algorithms, memetic algorithm has been found to result in a compact design due to its Lamarckian inheritance property and local search methodology. As a future work, texture analysis results can be used to train a network of processors for image classification and decision making.

### References

[1]. P. H. Langade, and S.B. Patil, “A Survey on Systolic Array Multiplier and its Implementation on FPGA,” vol. 4, no. 5, pp. 1299-1302, 2015.

[2]. J.H. Weston, C.N. Zhang and Hua Li, “Some Space Considerations of VLSI Systolic Array Mappings,” IEEE, pp. 375-381, 2000.

[3]. L. D Whiteley, A. E. Howe, S. Rana, J.P. Watson, L. Barbulescu, “Comparing Heuristic Search Methods and Genetic Algorithms for Warehouse Scheduling,” In SMC’98 Conference Proceedings, IEEE International Conference on Systems, Man, and Cybernetics, vol. 3, pp. 2430 – 243, 1998.

[4]. Poonam Garg, “A comparison between Memetic algorithm and Genetic Algorithm for the cryptanalysis of simplified Data Encryption Standard Algorithm,” International Journal of Network Security and Its Applications, vol. 1, no. 1, pp. 34 – 42, 2009.

[5]. E. Garcia-Gonzalo, J. L. Fernandez-Martinez, “A Brief Historical Review of Particle Swarm Optimization (PSO),” Journal of Bioinformatics and Intelligent Control, Vol. 1, pp.3-16, 2012.

[6]. H. T. Kung, and C. E. Leiserson, “Systolic Arrays (for VLSI),” Sparse Matrix Symposium, pp. 256 -- 282, 1978.

[7]. Ivan E. Sutherland, and Carver A. Mead, “Microelectronics and Computer Science,” Scientific American, vol. 237, pp. 210-228, 1977.

[8]. J. Fortes, K. Fu and B. Wah, “Systematic approaches to the design of algorithmically specified systolic arrays,” ICASSP 85, IEEE Int. Conf. Acoust. Speech, Signal Processing, vol. 10, pp. 300-303, 1985.

[9]. S. Subathradevi and C. Venilla, “Systolic array multiplier for augmenting data center networks communication link,” Cluster Computing pp. 1-11, 2018.

[10]. R.P. Brent, H.T. Kung, and F.T. Luk, “Some linear-time Algorithms for Systolic Arrays,” Cornell University, 2010.

[11]. Emad Elbeltagi, Tarek Hegazy, Donald Grierson, “Comparison among five evolutionary-based optimization algorithms,” Advanced Engineering Informatics, vol. 19, pp. 43-53, 2005.

[12]. W. E. Hart, “A theoretical comparison of evolutionary algorithms and simulated annealing,” Sandia National Labs., Albuquerque, NM (United States), 1995.

[13]. Srikanta Patnaik, Xin-He Yang, Kazumi Nakamatsu, “Nature-Inspired Computing and Optimization: Theory and Applications,” Springer, 2017.

[14]. P. Wang, Q. Lu Z. Fan, “Evolutionary design optimization of MEMS: a review of its history and state-of-the-art,” Cluster Computing, pp. 1-7, 2018.

[15]. R. Kruse, C. Borgelt, C. Braun, S. Mostaghim and M. Steinbrecher, “Computational intelligence: A methodological introduction,” Springer, 2016.

[16]. Darrell Whitley, “An Overview of Evolutionary algorithms: practical issues and pitfalls,” Information and Software Technology, vol. 43, pp. 817-831, 2001.

[17]. E. Alba, and M.Tomassini, “Parallelism and evolutionary algorithms,” IEEE Transactions on Evolutionary Computation, vol. 6, no. 5, pp. 443 – 462, 2002.

[18]. Bagavathi C and Saraniya O, “Evolutionary Mapping Techniques for Systolic Computing System,” Deep Learning and Parallel Computing Environment for Bioengineering Systems, Academic Press, ISBN 9780128167182. https://doi.org/10.1016/B978-0-12-816718-2-00020-8, pp 207-223, 2019.

[19]. R. S. Zebulun, M. A. Pacheco, and M. M. B. Vellasco, “Evolutionary electronics: automatic design of electronic circuits and systems by genetic algorithms,” CRC press, 2018.

[20]. Kamal Yurtkan and Hasan Demirel, “Entropy-based feature selection for improved 3D facial expression recognition,” Signal, Image and Video Processing, vol. 8, no. 2, pp. 267-277, 2014.

[21]. Mukesh A. Zaveri, S.N. Merchant, and U.B. Desai, “Evolutionary algorithm for data association and IMM-based target tracking in IR image sequences,” Signal, Image and Video Processing, vol. 7, pp. 27 – 43, 2013.

[22]. B. Nakisa, M. N. Rastgoo, D. Tjondronegoro and V. Chandran, “Evolutionary computation algorithms for feature selection of EEG-based emotion recognition using mobile sensors,” Expert Systems with Applications, vol. 93, pp. 143-155,2018.

[23]. R. Steinbach, “Successful application of evolutionary algorithms in engineering design,” Journal of Bionic Engineering, vol. 7, no. 4, pp. 199-211, 2010.

[24]. H. Karami, M.Ehteram, S. F.Mousavi, S. Farzin, O.Kisi and A.El-Shafie, “Optimization of energy management and conversion in the water systems based on evolutionary algorithms,” Neural Computing and Applications, pp. 1-14, 2018.
Advances in Computing, Communication, Automation and Biomedical Technology

[25] T. Back, D.B. Fogel, Z. Michalewicz, “Handbook of Evolutionary Computation,” Oxford Univ. Press, 1997.

[26] C. J. Tan, S.C. Neoh, C.P. Lim, S. Hanown, W.P. Wong, C.K. Loo, and S. Nahavandi, “Application of an evolutionary algorithm-based ensemble model to job-shop scheduling,” Journal of Intelligent Manufacturing, vol. 30, no. 2, pp. 879–890, 2019.

[27] L. Lin, and M. Gen, “Hybrid evolutionary optimisation with learning for production scheduling: state-of-the-art survey on algorithms and applications,” International Journal of Production Research, vol. 56, no. (1-2), pp. 193-223, 2018.

[28] J.H. Holland, “Adaptation in Natural and Artificial Systems,” MI Ann Arbor: Univ. Michigan Press, 1975.

[29] Pablo Moscato, and Carlos Cotta, “A modern introduction to memetic algorithms,” In Handbook of metaheuristics, Springer, Boston, pp. 141-183, 2010.

[30] P. Moscato, and C. Cotta, “A gentle introduction to memetic algorithms,” In Handbook of metaheuristics, Springer, Boston, MA, pp. 105 -144, 2003.

[31] W. E. Hart, N. Krasnogor and J. E. Smith, (Eds.), “Recent advances in memetic algorithms,” Springer Science and Business Media, vol. 166, 2004.

[32] L. Paquete, and T. Stützle, “Stochastic Local Search Algorithms for Multiobjective Combinatorial Optimization,” A Handbook of Approximation Algorithms and Metaheuristics: Methodologies and Traditional Applications, vol 1, 2018.

[33] P. Moscato, and C.Cotta, “An Accelerated Introduction to Memetic Algorithms,” In Handbook of Metaheuristics, Springer, Cham, pp. 275-309, 2019

[34] G. Fraser, A. Arcuri, and P. McMinn, “A memetic algorithm for whole test suite generation,” Journal of Systems and Software, vol. 103, pp. 311-327, 2015.

[35] Farah, T.Y.Q. Nguyen, A. Groh, D. Guenot, P. Jeannesson, and C. Gobinet, “Development of a memetic clustering algorithm for optimal spectral histology: application to FTIR images of normal human colon,” Analyst, vol. 141, no. 11, pp. 3296-3304, 2016.

[36] D. A. Pelta, and N. Krasnogor, “Multimeme algorithms using fuzzy logic-based memes for protein structure prediction,” In Recent advances in memetic algorithms. Springer, Berlin, Heidelberg, pp. 49-64, 2005.

[37] M. Ghosh, S. Malakar, S. Bhownik, R. Sarkar, and M. Nasipuri, “Feature Selection for Handwritten Word Recognition Using Memetic Algorithm,” In Advances in Intelligent Computing. Springer, Singapore, pp. 103-124, 2019.

[38] H. Samma, C. P. Lim, J. M. Saleh, and S. A. Suandi, “A memetic-based fuzzy support vector machine model and its application to license plate recognition,” Memetic computing, vol. 8, no. 3, pp. 235-251, 2016.

[39] Osaba, E., Yang, X.S., Diaz, F., Onieva, E., Masegosa, A. D., and Perallos, A.: A discrete firefly algorithm to solve a rich vehicle routing problem modelling a newspaper distribution system with recycling policy,” Soft Computing, vol. 21, no. 18, pp. 5295-5308, 2017.

[40] M. Bereta, “Baldwin effect and Lamarckian evolution in a memetic algorithm for Euclidean Steiner tree problem,” Memetic Computing, pp. 1-18, 2018.

[41] Q. Wu, H. Liu, X. Yan “Multi-label classiﬁcation algorithm research based on swarm intelligence,” Cluster Computing, vol. 19, no. 4, pp. 2075-2085, 2016.

[42] Zhuhua Cui, and Zhonghui Shi, “Boid particle swarm optimisation,” International Journal of Innovative Computing and Applications, vol. 2, no. 2, pp. 77-85, 2009.

[43] Ganesh Kumar, Salman Mohagheghi, Jean-Carlos Hernandez Yamille delValle, “Particle Swarm Optimization: Basic Concepts Variants and Applications in Power Systems,” IEEE, pp. 171-195, 2008.

[44] Malik, Arif Jamal, and Farrukh Aslam Khan, “A hybrid technique using binary particle swarm optimization and decision tree pruning for network flow debugging,” Cluster Computing, vol. 21, no. 1, pp. 667-680, 2018.

[45] V. Priya and K. Unamaheswari, “Enhanced continuous and discrete multi objective particle swarm optimization for text summarization,” Cluster Computing, pp. 1-12, 2018.

[46] S. Das, A. Abraham, and A. Konar, “Particle swarm optimization and differential evolution algorithms: technical analysis, applications and hybridization perspectives,” In Advances of computational intelligence in industrial systems, Springer, Berlin, Heidelberg, pp. 1-38, 2008.

[47] B. Akay, “A study on particle swarm optimization and artificial bee colony algorithms for multilevel thresholding,” Applied Soft Computing, vol. 13, no. 6, pp. 3066 – 3091, 2013.

[48] R. Ugolotti, Y. S. Nashed, P. Mesejo, Ievkovi, L. Mussi, and S. Cagnoni, “Particle swarm optimization and differential evolution for model-based object detection,” Applied Soft Computing, vol. 13, no. 6, pp.3092-3105, 2013.

[49] A. Haldorai and A. Ramu, “Security and channel noise management in cognitive radio networks,” Computers & Electrical Engineering, vol. 87, p. 106784, Oct. 2020. doi:10.1016/j.compeleceng.2020.106784

[50] A. Haldorai and A. Ramu, “Canonical Correlation Analysis Based Hyper Basis Feedforward Neural Network Classification for Urban Sustainability,” Neural Processing Letters, Aug. 2020. doi:10.1007/s11063-020-10327-3

[51] S. R. Mishra, T. K. Mishra, A. Sarkar, and G. Sanyal, “PSO based combined kernel learning framework for recognition of first-person activity in a video,” Evolutionary Intelligence, pp. 1–7, 2018.

[52] S. Mirjalili, “Evolutionary Deep Neural Networks,” In Evolutionary Algorithms and Neural Networks. Springer, Cham, pp. 141-156, 2019.

[53] MP Wachowia, R Smolikova, Y Zheng, MJ Zurada, and AS Elmaghraby, “An Approach to Multimodal Biomedical Image Registration Utilizing Particle Swarm Optimization,” IEEE Transactions on Evolutionary Computation, vol. 8, no. 3, pp. 289-301, 2004.

[54] L Messerschmidt and AP Engelbrecht, “Learning to Play Games Using a PSO-Based Competitive Learning Approach,” IEEE Transactions on Evolutionary Computation, vol. 8, no. 3, 280-288, 2004.

[55] MA Abido, “Optimal Design of Power System Stabilizers Using Particle Swarm Optimization,” IEEE Transactions on Energy Conversion, vol. 17, no. 3, pp. 406 – 413, 2002.

[56] S Paterlini and T Krink, “Differential Evolution and Particle Swarm Optimization in Partitional Clustering,” Computational Statistics and Data Analysis, vol. 50, pp. 1220-1247, 2006.

[57] B Bala Tripura Sundari, “Dependence Vectors and Fast Search of Systolic Mapping for Computationally Intensive Image Processing Algorithm,” In Proceedings of the International Multiconference of Engineers and Computer Scientists, vol. 1, March (2011)

[58] Bagavathi C and Saraniya O, “A Novel Systolic Architecture Design for Image Processing,” International Journal of Science and Innovative Engineering & Technology, vol. 2, May 2018.

[59] L D Whitley, A.E. Howe, S. Rana, JP. Watson, L. Barubescu, “Comparing Heuristic Search Methods and Genetic Algorithms for Warehouse Scheduling,” In IEEE International Conference on Conference: Systems, Man, and Cybernetics, 1998.

[60] Nam Ling and M A Bayoumi, “Systematic Algorithm Mapping for Multidimensional Systolic Arrays,” Journal of Parallel and Distributed Computing, Academic Press, vol. 7, no. 2, pp. 368-382, 1989.

[61] C.M. Fiduccia, and R.M. Mattheyes, “A Linear-Time Heuristic for Improving Network Partitions,” 19th Design Automation Conference, Las Vegas, NV, USA, pp. 175-181, 1982.

[62] Judea Pearl, “Heuristics: Intelligent search strategies for computer problem solving,” The Addison-Wesley Series in Artificial Intelligence, Reading, Mass, 1985.

[63] Tuceryan, Mihran, and Anil K. Jain, “Texture analysis,” In The  Addison-Wesley Series in Artificial Intelligence, Chapter 4, 2007.

[64] Amadasun Moses, Robert King, “Textural Features Corresponding to textural properties,” IEEE Transactions on systems, man and cybernetics, vol. 19, no. 5, pp. 1264-1274, 1989.

[65] Keshab K Parhi, “VLSI digital signal processing systems: design and implementation,” John Wiley and Sons, 2007.

[66] Mr.S.Mithunraj, Dr.S.Sophia, “Smart Infotainment System For Vehicular Network”, in International Journal of Engineering & Technology, Vol. 7 No.3, 2018.

[67] Taha, “Operations research: An Introduction,” Eighth Edition, Pearson Education, 2008.