SSRG International Journal of Electrical and Electronics Engineering
ISSN: 2348-8379/ https://doi.org/10.14445/23488379/IJEEE-V10I1P115

Volume 10 Issue 1, 160-167, January 2023

Original Article

Optimal FSM's State Encoding for Low power using Dynamic Boundary Difference Mutation Strategy in Evolutionary Programming

Deepti Raj[1], AB Kalpana[2], Manoj Kumar Singh[3]

[1] *Electronics & Telecommunication Engineering, Dayananda Sagar College of Engineering, Karnataka, India*
[2] *Electronics & Communication Engineering, Bangalore Institute of Technology, Karnataka, India*
[3] *Manuro Tech Research Private Limited, Karnataka, India*

[1] *Corresponding Author : deepthi-tce@dayanandasagar.edu*

Received: 06 December 2022 Revised: 12 January 2023 Accepted: 19 January 2023 Published: 29 January 2023

Abstract - *In this paper, a new computation intelligent approach based on evolutionary programming is applied to minimize the weighted hamming distance among states to reduce the switching frequency for low-power design in Finite State Machines. A mutation strategy is proposed to carry better exploration of solution space by deploying a dynamic differential approach between the current solution position and solution domain boundary limits. The proposed method is compared against the performances of the standard form of mutation strategies based on the self-adaptive version of Gaussian and Cauchy mutation and an advanced version of particle swarm optimization. The different combinations of Gaussian and Cauchy mutations are also examined. The performances of the proposed solution were superior and computationally efficient in comparision to all others. The robustness against variability is excellent over a large number of runs.*

Keywords - *Finite state machine, State encoding, Low power, Gaussian mutation, Cauchy mutation, Evolutionary programming.*

1. Introduction

One of the key issues with synthesizing sequential machines is state assignment in FSMs. We aim to reduce the average switching activity for an FSM in the state variables by limiting the amount of bit changes during state transitions. We have modified our approach to present a state encoding procedure that minimizes the Hamming distance between the states' codes with high transition probabilities using a probabilistic description of an FSM.

The currently employed technologies for designing sequential circuits often include a number of distinct steps, among which the encoding phase is an important one. This work applies a mutation strategy in evolutionary programming (EP), which provides dynamic change to boundary difference (DBDEP) to the solution with a generation basis. The evolutionary computation community follows a simple but powerful approach when there is a need to design an evolutionary algorithm: "larger change in the beginning to explore faster and smaller change as it moves towards convergence to avoid optima miss". The proposed solution has followed the same concept by providing the dynamic change in the difference between the current position and boundary limit to explore the surrounding. The proposed solution has shown faster and optimal convergence.

2. FSM Encoding for Low Power

The formal definition of an FSM can be given as a 5-tuple system as: $M = (S; I; O; T; a)$, where the parameters representing the finite input (I), output (O) space, finite state space(S), and transition relation ($T: I \times S \rightarrow O$ or $T: S \rightarrow O$) for Melay or Moore machine and 'a' a next state transfer function ($a: I \times S \rightarrow S$). There is an involvement of injective mapping ($f: S \rightarrow B^n$) in state assignment coding where 'n' is the length of the code and satisfies a relation ($n \geq [\log2(S)]$). B^n represents an 'n' dimensional Boolean space.

A graph $G(V, E)$ that has an edge ($e_{ij} \in E$) that reflects a transition from state Si toSj can be used to depict the State transition graph (STG), which carries vertex($Si \in V$). Assuming P_{si} represents the probability of the state Si and p_{ij} represents the conditional (state) transition probability from state Si to state Sj. Considering a STG as a Markov chain which is a representation of a finite state Markov process and has memory less characteristic, the probability of a state is defined through the limiting state probability theorem as the limiting value approached as it is run for an infinite amount of time. Following that, $P_{ij} = p_{ij}P_{si}$ is used to compute the total state transition probability (Pij) for a transition from state Si to state Sj. The amount of switching between two states is represented by the sum of all state transition probabilities between them. Wij = Pij + Pji, as a weight between the two states that are thus assigned to the single edge that connects them. A weighted graph corresponding to an STG is formed by replacing all the transitions between two states with a weighted edge. The weight on an edge indicator can define the state assignment of the connected states. By providing shorter distance codes to states, greater weight edges signify states with higher transition probabilities and reduce switching frequency. Therefore, having a Minimum Weighted Hamming Distance can be a cost-effective strategy for reducing power consumption (MWHD). Mathematically

$$\sum_{S_i S_j \epsilon S} W_{ij} H\left(S_i, S_j\right) \tag{1}$$

Where W_{ij} is the weight of the edge and $H\left(S_i, S_j\right)$ is the Hamming distance between the assigned state code for the states S_i and S_j.

3. Proposed Solution: A Mutation Strategy Based on the Dynamic Positional Difference from the Boundary Limit to the Current Position of the Solution

Iterative processes of random variation and selection are used in the two-step, population-based process of evolution. This procedure can be carried out by creating probable solutions to a problem and using random numbers drawn from a predetermined distribution to come up with fresh solutions. A selection criterion is necessary to decide which solution should be kept and which should be abandoned. The process's validity depends on various variables and operators, including the population size, the type and amount of random variation, the number of "parent" solutions, and others. For example, when a predetermined maximum number of generations or a reasonable error tolerance has been met, the algorithm terminates. The procedure may be written as the difference equation given by Eq.2.

$$X\ (t\ +\ 1) = \mathcal{O}_s\left(\ \mathcal{O}_V(X\ [t])\right) \tag{2}$$

Where X[t] is the population at time **t** under a representation X, $\mathcal{O}_V$ is a random variation operator and $\mathcal{O}_s$ is the selection operator. A wide range of desirable representations, selection methods and variation operators are available. The validness of an evolutionary algorithm relies on the interplay between the operator's $\mathcal{O}_s$ and $\mathcal{O}_V$ as applied to a chosen representation X and initialization X [0]. Compared to genetic algorithms, which often operate on a separately programmed transform of the goal variables, the advantage of evolutionary programming and evolution methods is that these algorithms directly operate on the real values to be optimized.

The advanced standard form of EP contains the self-mutation strategy where the Gaussian distribution or Cauchy distribution is applied to provide the change. The involved strategy parameters were self-adaptive, and the solution population parameters also changed. The Cauchy distribution can provide a larger change in comparison to the Gaussian distribution. Such large changes by Cauchy distribution can be useful at the early stage of evolution, where there is more diversity, and it needs a high level of exploration. The one-dimensional Cauchy density function centered at the origin can be defined in Eq.3, and the corresponding distribution function can be defined in Eq.4.

$$f_t(x) = \frac{1}{\pi}\frac{t}{t^2+x^2} \tag{3}$$

$-\infty < x < \infty$ where $t > 0$ is a scalar parameter.

$$F_1(x) = \frac{1}{2} + \frac{1}{\pi}\left(arctan\left(\frac{x}{t}\right)\right) \tag{4}$$

Each solution was taken as a pair of real-valued vectors $(\bar{x}_i, \bar{\sigma}_i\ ), \forall\ i\ \epsilon\ \{1,2,3, \dots . N\}$ with their dimensions corresponding to the number of variables. The initial components of each $\bar{x}_i, , \forall\ i\ \epsilon\ \{1,2,3, \dots . N\}$ were selected in accordance with a uniform distribution ranging over a presumed solution space. The value of$\bar{\sigma}_i, \forall\ i\ \epsilon\ \{1,2,3, \dots . N\}$, the so-called strategy parameters were initialized with some value. Offspring were generated from each parent by Eq.5, while the parameters upgradation took place by Eq.6.

$$\bar{x}_i'(j) = \bar{x}_i(j) + \bar{\sigma}_i(j)R_j \tag{5}$$

$$\bar{\sigma}_i'(j)\bar{\sigma}_i(j). exp\left(\tau' N(0,1)\tau\ N_j(0,1)\right) \tag{6}$$
$$\forall\ j\ \epsilon\ \{1,2,3, \dots . r\}$$

Where $\bar{x}_i'(j), \bar{x}_i(j),\ \bar{\sigma}_i'(j) and\ \bar{\sigma}_i(j)$ denote the j[th] component of vectors $\bar{x}_i', \bar{x}_i,\ \bar{\sigma}_i' and\ \bar{\sigma}_i$ respectively. R_j is random variable(Gaussian or Cauchy). $N(0,1)$ denotes a standard Guassian random variable. $N_j(0,1)$ Indicates that the random variable is sampled for each new value of the counter j. The scaling factors$\tau\ and\ \tau'$ are robust exogenous parameters and depend upon the problem dimension inversely.

In the standard form of EP, either Gaussian mutation or the Cauchy mutation strategy is applied. Each distribution has its own advantage and limitation. The small change of Gaussian distribution may not be very suitable at the early stage, but at a later stage, it may be more useful, while the larger change of Cauchy distribution very useful for the early stage but may not be good for the later stage. Instead of depending upon the capabilities of the distribution function, the proposed solution has used either side of available solution space from the current solution position while maintaining the principle of larger change at the beginning while lowering the change level with time. In short, the available distance from the boundary limit to the current position is dynamically used as the changing size. The mathematical function of the proposed mutation strategy to produce the offspring is shown in Eq.9. For the kth dimension of parent' i', the kth dimension value for the corresponding offspring is given by Eq7.

$$z_i'(k)z_i(k)(-1)^r\left[\mathcal{OP}-\mathcal{OP}\times R^{\left(1-\frac{k}{T}\right)^a}\right] \quad (7)$$

$$\mathcal{OP}=\begin{cases} UBL-z_i \; if \;\; r=0 \\ z_i-LBL \;\; if \;\; r=1 \end{cases}$$

Where,
r: $a\ number\ randomly\ sampled\ from\ set\ of$ $\{0,1\}$,
R: $a\ random\ number\ generated\ through\ U$$[0,1]$
UBL & LBL are the upper boundary limit and lower boundary limit correspondingly k: current iteration number
T: maximum allowed number of iterations
a: scaling factor

As it clear from Eq.7 under the probabilistic environment, there is either side of boundary limit have been considered to provide the additive as well as deductive change. This will cause of exploration more broadly. Dynamic reduction in change observed with increasing value of iteration. The function block diagram of the proposed solution has shown in Fig.1.

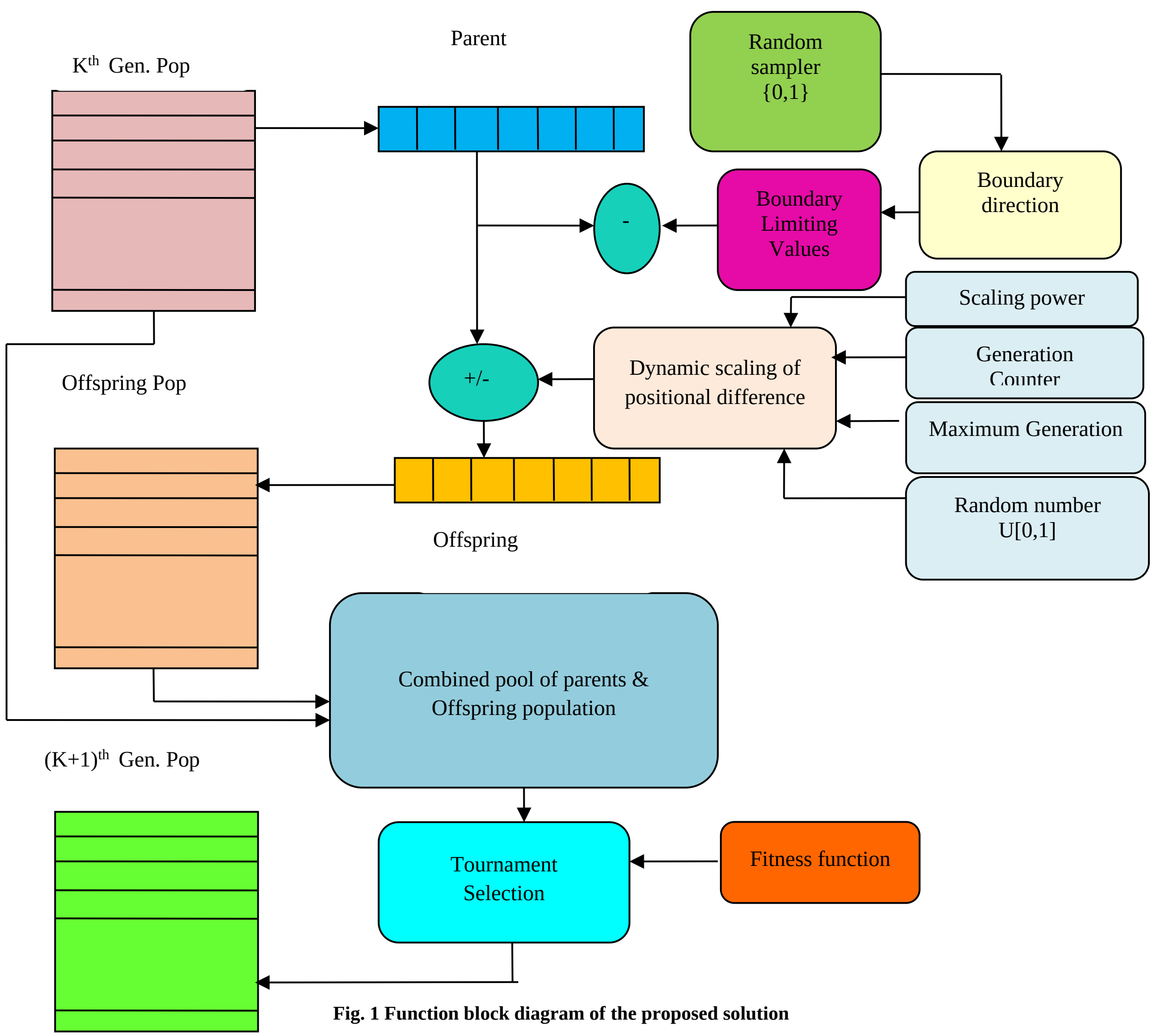
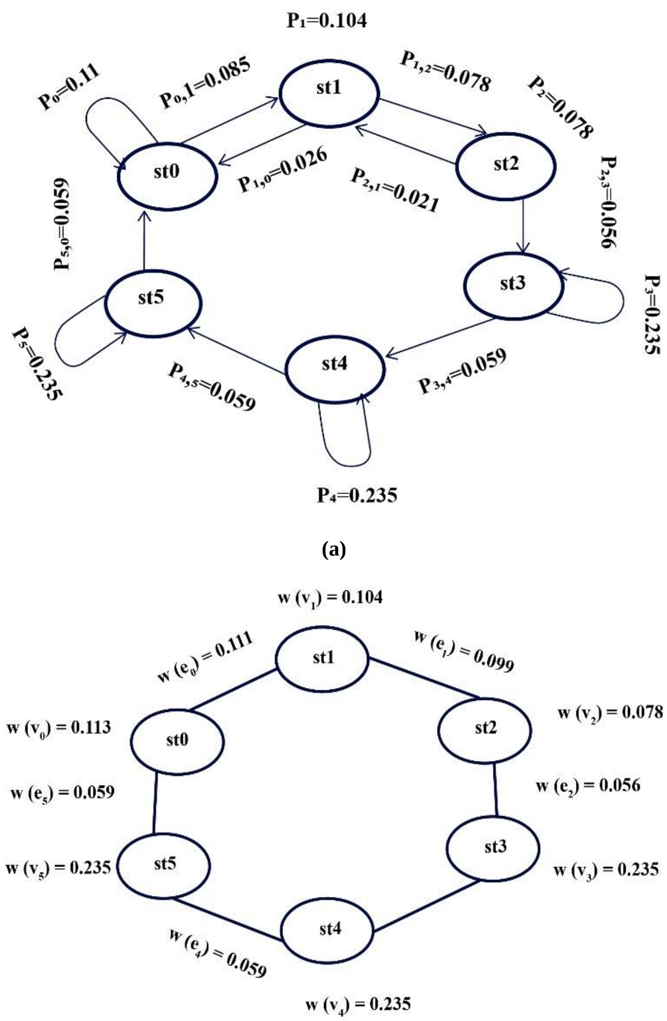


Fig. 1 Function block diagram of the proposed solution

4. Experimental Results

For the experimental purpose, the FSM benchmark problem 'bbtas' is considered. The state probabilities, total transition, and weighted graph for FSM of 'bbtas' is shown in Fig.2



(a)

(b)

Fig. 2 FSM "bbtas" (a) State probabilities and total transition and (b) weighted graph

In the experiment, five different forms of mutation strategies are applied, as shown in Fig.6. The self-adaptive form of Gaussian (GMEP) and Cauchy mutation(CMEP), as discussed in section 4, are applied to create the offspring as shown in Fig.3 (i) and (ii). Two combined variations are also considered to capture the benefits of both distributions.

In one case, the mean of offspring generated from the Gaussian mutation and Cauchy mutation are considered as final offspring (MGCMEP) as shown in Fig.3 (iii), while in other cases, among the two offspring generated by Gaussian and Cauchy mutation, the better offsprings are considered as final offspring (HGCMEP) as shown in Fig.3 (iv). The proposed form of mutation strategy based on dynamic boundary difference from the current position (DBDEP) is shown in Fig.3 (v). The search for a solution through the algorithm is done in the integer domain. The fitness value is obtained after transforming the integer value in a binary domain where the total weighted hamming distance is estimated. A total of 6 different states are available in the considered example; hence, 3-bit encoding of each state is needed (different possible states with 3 bits are 2^3=8). Hence upper and lower limits of the solution boundary were 0 and 7.

For the experimental purpose, the population size is considered 10 for all the algorithms, and the total allowed number of generations is 100. There are 100 independent trials given to capture the variability. For Gaussian and Cauchy mutation-based strategies, the spread parameter values are considered 0.01. For all algorithms, tournament selection is applied where the total opposition numbers were 4, which is 20% of the combined parent and offspring population. Larger opposition members can cause more favor to select the higher fitness solution. At the same time, very low values will have more chance for a low fitness solution to have the same score as high fitter solutions. If solution values cross the boundaries limitation, random values are selected within the boundaries range to replace the out-of-range values. The complete experiments have developed in the MATLAB environment

In DBDEP, the scaling factor 'a' contributes significantly to the convergence speed. Different values {2, 4, 5, 6, 8} were experimented with to estimate the optimal scaling factor value. For each value of 'a' from the set, the process was repeated by 20 independent trials to get variability. The mean final convergence generation number were taken correspondingly to each value of 'a' were {32, 43, 17, 29, 23}. It can be observed that there is a significant reduction in the required generation for the final convergence with the value of 'a' DBDEP value of 'a' was considered as 5.

Table 1. Statistical performance of total WHD over 100 trials

	NOS-SAPSO	GMEP	CMEP	MGCMEP	HGCMEP	DBDEP
Min	0.8860	0.8860	0.8860	0.8860	0.8860	**0.8860**
Max	1.1160	1.1160	0.8860	1.2020	1.1960	**0.8860**
Mean	0.9158	0.8952	0.8860	0.9497	0.9235	**0.8860**
Std.Dev	0.0776	0.0453	0.0000	0.1059	0.0867	**0.0000**

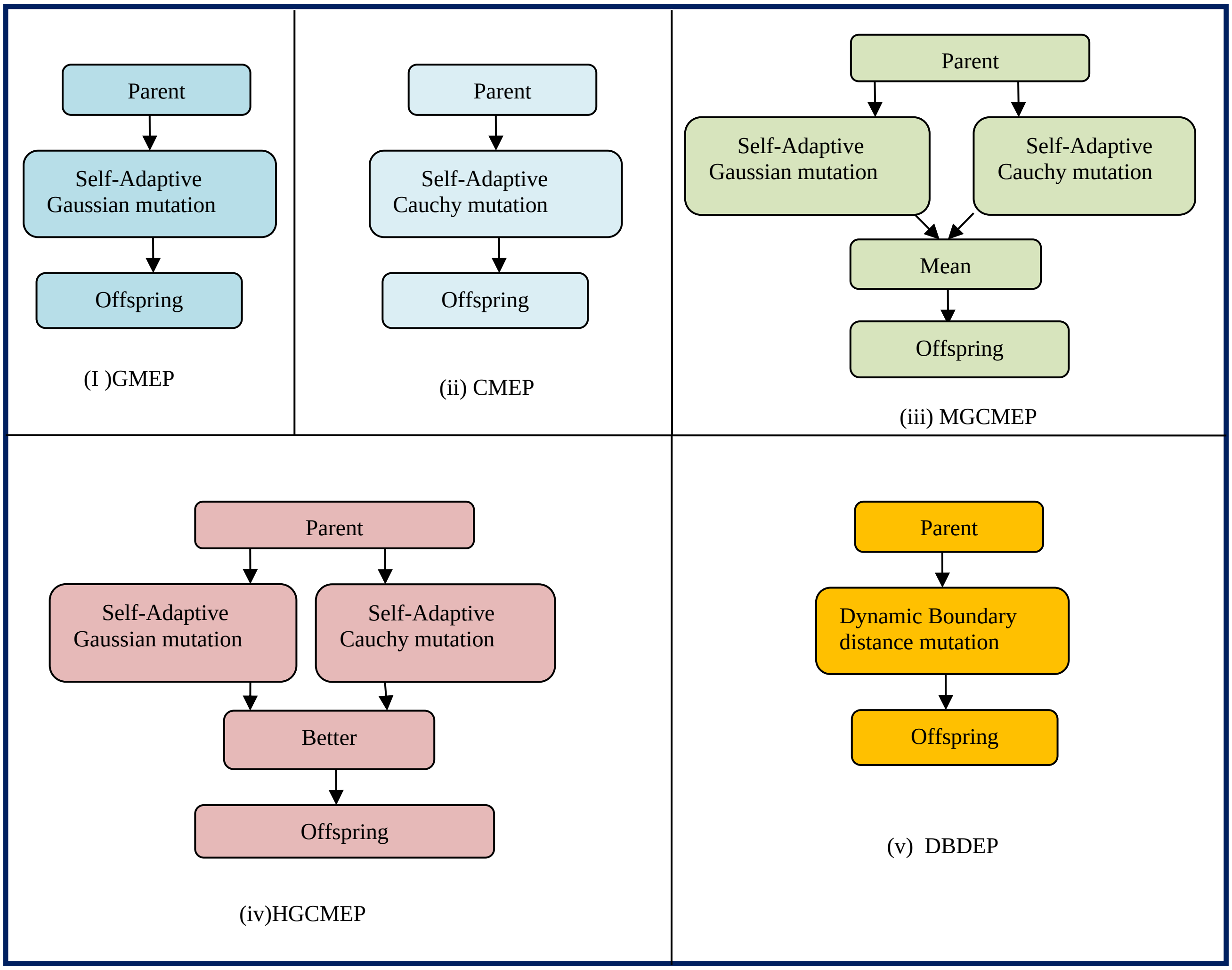


Fig. 3 Different mutation strategies in EP applied to generate offspring

Table 2. Optimal convergence success rate (%) over 100 trials

NOS-SAPSO	GMEP	CMEP	MGCMEP	HGCMEP	DBDEP
87	96	100	73	84	100

Table 3. Statistical performance of convergence generation over 100 trials

	NOS-SAPSO	GMEP	CMEP	MGCMEP	HGCMEP	DBDEP
Min	1.0	1.0	1.0	1.0	1.0	**1.0**
Max	47.0	68.0	67.0	100.0	100.0	**40.0**
Mean	8.07	17.24	13.06	29.37	32.28	**9.34**
Std.Dev	6.95	17.99	13.62	33.59	34.19	**8.14**

The obtained total weighted hamming distance over 100 trials is shown in Table 1, while the success rate in delivering the optimal value(0.8860) is presented in Table 2. It is observed from Table 1 that NOS-SAPSO has a better mean WHD value of 0.9158 and a success rate of convergence is 87% which is better against the combined mutation strategy performances, which are 0.9497 and 0.9235 with a success rate of 73% and 84% for MGCMEP and HGCMEP correspondingly. It is also observed that the performances of HGCMEP are better than HGCMEP. The performance of CMEP is better in comparision to GMEP, which is 0.8860 against 0.8952 with a success rate of 100% against 96%. The performance of DBDEP is excellent and has a WHD value of 0.8860, the global value with a success rate of 100%. The performances of CGEP and DBDEP are the same from the WHD point of view, but the difference is that where there is a lesser number of generations needed to obtain the global solution. The performance of NOS-SAPSO is also satisfactory from a convergence generation point of view. In fact, it has the best 8.07 number of generations for successful

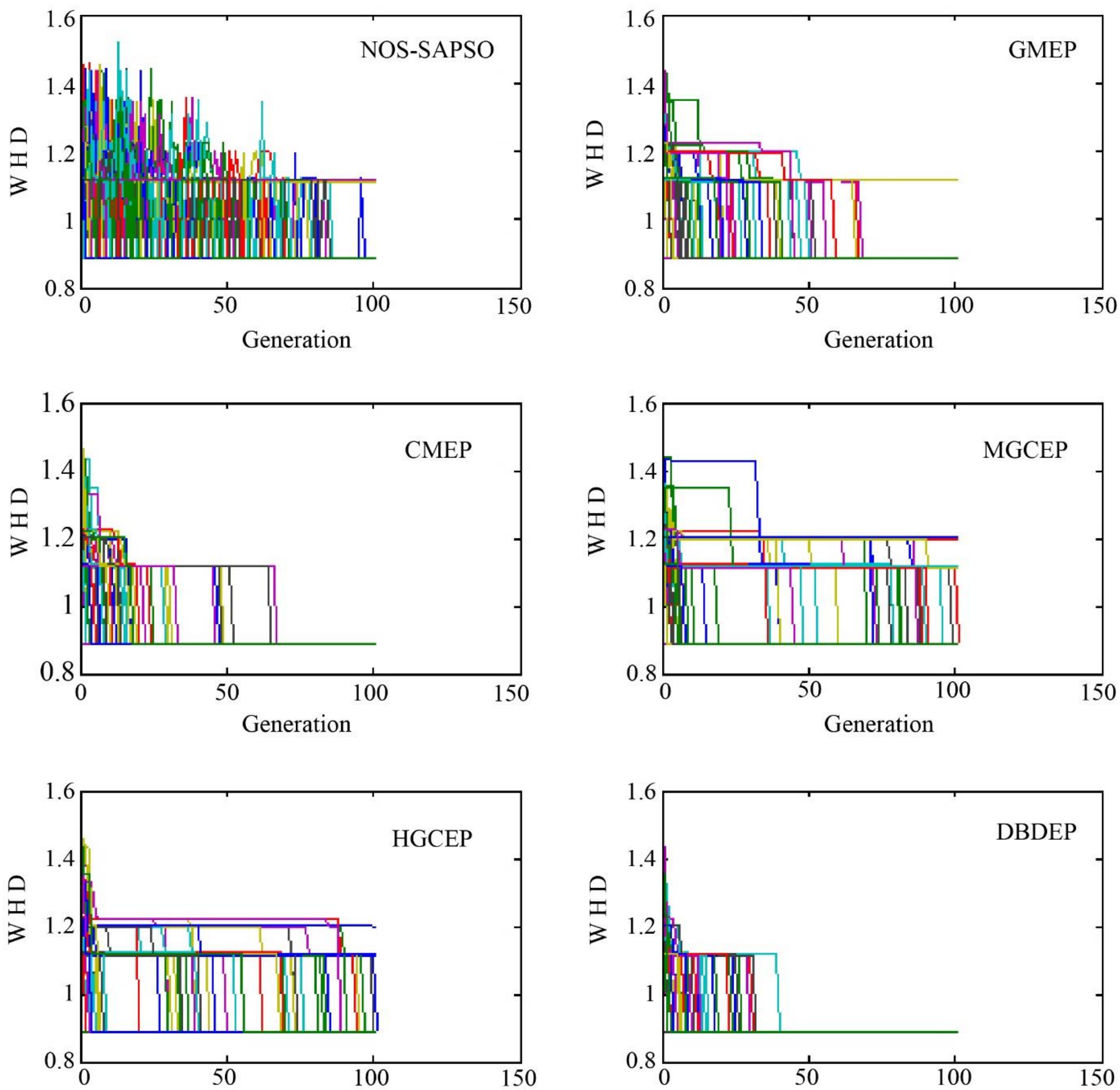


Fig. 4 Convergence path by algorithms over 100 independent trials

convergence but has a low success rate compared to CMEP and DBDEP. The convergence characteristics by different algorithms over 100 independent trials have shown in Fig.4. It can be observed that NOS-SAPSO has shown a more diverse nature in the convergence path while DBDEP has shown consistency over trials. The advantages of applying the heuristic approach to obtain the optimal encoding can be understood from Table 4, where rather than having a single solution, multiple solutions (74 different encoding schemes) having the same WHD have been explored by DBDEP under 100 trials. This diverse solution can help design the hardware to suit the surrounding environment or reduce cost.

5. Conclusion

Designing the low-power FSM by reducing the switching activities by assigning the optimal state encoding was the core theme of the work. The objective of achieving the optimal state encoding has been achieved with various different forms of EP and PSO. A new approach has been utilized the available space from the boundary limit in a scaled manner to search the nearby region. Larger change at the beginning and lowering the change with generation has helped the proposed algorithm DBDEP to converge faster and optimally. The proposed solution has shown a very high level of consistency and repeatability against the Gaussian and Cauchy mutation strategies and their combination.

Reference

[1] A.E.A. Almaini et al., “State Assignment of Finite State Machines Using a Genetic Algorithm,” *IEE Proceedings - Computers and Digital Techniques*, vol. 142, no. 4, pp. 279-286, 1995. *Crossref*, http://doi.org/10.1049/ip-cdt:19951885

[2] Valentin Hristov, and Gergana Kalpachka, “Use of Web Based Calculator of Genetic Algorithms for Optimal Coding the States of Finite State Machines,” *2014 10th France-Japan/ 8th Europe-Asia Congress on Mecatronics*, Tokyo, Japan, pp. 274-278, 2014. *Crossref*, http://doi.org/10.1109/MECATRONICS.2014.7018571

[3] Rizki Mardian, and Kosuke Sekiyama, “In-vitro DNA-based Finite State Machine,” *2015 International Symposium on Micro-Nano Mechatronics and Human Science, IEEE*, pp. 1-6, 2015. *Crossref*, http://doi.org/10.1109/MHS.2015.7438306

[4] P. Rahul Reddy, "Analysis of Low-Power and Area-Efficient Shift Registers using Pulsed Latch," *SSRG International Journal of Electronics and Communication Engineering*, vol. 3, no. 1, pp. 1-4, 2016. *Crossref*, https://doi.org/10.14445/23488549/IJECE-V3I1P102

[5] G. Hasteer, and P. Banerjee, “A Parallel Algorithm for State Assignment of Finite State Machines,” *IEEE Transactions on Computers*, vol. 47, no. 2, pp. 242-246, 1998. *Crossref*, https://doi.org/10.1109/12.663772

[6] E. Vidal et al., “Probabilistic Finite-State Machines - Part I,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 7, pp. 1013-1025, 2005. *Crossref*, https://doi.org/10.1109/TPAMI.2005.147

[7] Namrata Joshi, and Ravi Kumar Jangir, "Optimized Design of Low Power and High Speed Ring Counter," *SSRG International Journal of Electronics and Communication Engineering*, vol. 6, no. 2, pp. 1-4, 2019. *Crossref*, https://doi.org/10.14445/23488549/IJECE-V6I2P101

[8] Y. Shi, and R. Eberhart, "A Modified Particle Swarm Optimizer," *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 69–73, 1998. *Crossref*, https://doi.org/10.1109/ICEC.1998.699146

[9] J. Kennedy, "The Particle Swarm: Social Adaptation of Knowledge," *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 303–308, 1997. *Crossref*, https://doi.org/10.1109/ICEC.1997.592326

[10] Qunfeng Liu, "Order-2 Stability Analysis of Particle Swarm Optimization," *Evolutionary Computation*, vol. 23, no. 2, pp. 187–216, 2015. *Crossref*, https://doi.org/10.1162/EVCO_a_00129

[11] Christopher W. Cleghorn, and Andries P. Engelbrecht, "Particle Swarm Stability: A Theoretical Extension using the Non-Stagnate Distribution Assumption," *Swarm Intelligence*, vol. 12, no. 1, pp. 1-22, 2018. *Crossref*, https://doi.org/10.1007/s11721-017-0141-x

[12] Anjali Raman et al., "Design of a Low Power 16- Bit CSLA Using Binary to Excess-1 Converter," *SSRG International Journal of VLSI & Signal Processing*, vol. 7, no. 2, pp. 11-13, 2020. *Crossref*, https://doi.org/10.14445/23942584/IJVSP-V7I2P103

[13] Pietro Ducange, Francesco Marcelloni, and Michela Antonelli, “A Novel Approach Based on Finite-State Machines with Fuzzy Transitions for Nonintrusive Home Appliance Monitoring,” *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, 2014. *Crossref*, https://doi.org/10.1109/TII.2014.2304781

[14] Hana Kubátová, “Finite State Machine Implementation in FPGAs,” *Design of Embedded Control Systems, Springer*, Boston, MA, pp. 175-184, 2005. *Crossref*, https://doi.org/10.1007/0-387-28327-7_15

[15] V. V. Solov’ev, “Minimization of Mealy Finite-State Machines by using the Values of the Output Variables for State Assignment,” *Journal of Computer and Systems Sciences International*, vol. 56, no. 1, pp. 96-104, 2017. *Crossref*, https://doi.org/10.1134/S1064230717010129

[16] Shaik Hameeda Noor et al., "Design of Low Power Encoder Using Switch Logic," *SSRG International Journal of Electronics and Communication Engineering*, vol. 5, no. 10, pp. 6-8, 2018. *Crossref*, https://doi.org/10.14445/23488549/IJECE-V5I10P102

[17] Nadia Nedjah, and Luiza de Macedo Mourelle, “Synchronous Finite State Machines Design with Quantum-Inspired Evolutionary Computation,” *Hardware for Soft Computing and Soft Computing for Hardware, Studies in Computational Intelligence*, Springer, Cham, vol. 529, pp. 119-154, 2014. *Crossref*, https://doi.org/10.1007/978-3-319-03110-1_9

[18] Jayesh Diwan, and Nagendra P. Gajjar, "A Novel NCL Threshold Gate Implementation for Low Power Asynchronous Designs using FinFETs," *International Journal of Engineering Trends and Technology*, vol. 70, no. 5, pp. 299-305, 2022. *Crossref*, https://doi.org/10.14445/22315381/IJETT-V70I5P232

[19] G. Sutter et al., “Low-Power FSMs in FPGA: Encoding Alternatives,” *Integrated Circuit Design Power and Timing Modeling, Optimization and Simulation PATMOS*, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, vol. 2451, pp. 363–370, 2002. *Crossref*, https://doi.org/10.1007/3-540-45716-X_36

[20] Dr.R.Carlo Novara, and Mohamed Djemai, "Design of a CMOS Multiplexer with Ultra Low Power using Current Mode Logic Technology," *SSRG International Journal of VLSI & Signal Processing*, vol. 3, no. 2, pp. 8-12, 2016. *Crossref*, https://doi.org/10.14445/23942584/IJVSP-V3I2P102

[21] Nitish Das, and P. Aruna Priya, “FPGA Implementation of Reconfigurable Finite State Machine with Input Multiplexing Architecture Using Hungarian Method,” *International Journal of Reconfigurable Computing*, vol. 2018, 2018. *Crossref*, https://doi.org/10.1155/2018/6831901

[22] Chu Donatus Iweh et al., "The Optimization of Hybrid Renewables for Rural Electrification: Techniques and the Design Problem," *International Journal of Engineering Trends and Technology*, vol. 70, no. 9, pp. 222-239, 2022. *Crossref*, https://doi.org/10.14445/22315381/IJETT-V70I9P223

[23] Abhishek Nag, Subhajit Das, and Sambhu Nath Pradhan, "Low-Power FSM Synthesis Based on Automated Power and Clock Gating Technique," *Journal of Circuits, Systems and Computers*, vol. 28, no. 05, p. 1920003, 2019. *Crossref*, https://doi.org/10.1142/S0218126619200032

[24] Ira Parashar et al., "Design and Simulation of 4*1 Mux Based on Low Power Design Techniques," *SSRG International Journal of VLSI & Signal Processing*, vol. 2, no. 1, pp. 17-21, 2015. *Crossref*, https://doi.org/10.14445/23942584/IJVSP-V2I1P105

[25] Sezer Gören, and F. Joel Ferguson, "On State Reduction of Incompletely Specified Finite State Machines," *C Computers & Electrical Engineering*, vol. 33, pp. 58-69, 2007. *Crossref*, https://doi.org/10.1016/j.compeleceng.2006.06.001

[26] Deepti Raj, A. B. Kalpana, and Manoj Kumar Singh, "State Encoding for Low Power in FSM Using Non-Oscillating Self-Adaptive Particle Swarm Optimization (NOS-SAPSO)," *Journal of Information and Optimization Sciences*, vol. 41, no. 2, pp. 509-519, 2020. *Crossref*, https://doi.org/10.1080/02522667.2020.1724615

[27] Niharika Agrawal, and Mamatha Gowda, "Power Flow Enhancement by TCSC using Two Different Types of Pulse Generators," *International Journal of Recent Engineering Science*, vol. 9, no. 1, pp. 31-38, 2022. *Crossref*, https://doi.org/10.14445/23497157/IJRES-V9I1P105

[28] J. Kennedy, and R. Eberhart, "Particle Swarm Optimization," *Proceedings of IEEE International Conference on Neural Networks IV*, pp. 1942–1948, 1995. *Crossref*, https://doi.org/10.1109/ICNN.1995.488968

[29] Simon M. Lucas, and T. Jeff Reynolds, "Learning Finite-State Transducers: Evolution Versus Heuristic State Merging," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 3, pp. 308-325, 2007. *Crossref*, https://doi.org/10.1109/TEVC.2006.880329

[30] Jae-hyeon So et al., "Optimal Design of Shared Transformer/Reactor for Improving Power Density," *International Journal of Engineering Trends and Technology*, vol. 69, no. 11, pp. 211-215, 2021. *Crossref*, https://doi.org/10.14445/22315381/IJETT-V69I11P227

[31] Robert Czerwinski, and Dariusz Kania, "State Assignment and Logic Optimization for Finite State Machines," *9th IFAC Workshop on Programmable Devices and Embedded Systems*, vol. 42, no. 1, pp. 39-44, 2009. https://doi.org/10.3182/20090210-3-CZ-4002.00011

[32] Liling Dong, Ning Wu, and Xiaoqiang Zhang, "Low Power State Machine Design for AES Encryption Coprocessor," *Proceedings of the International Multi Conference of Engineers and Computer Scientists*, vol. 2, 2015.

[33] Daniil Chivilikhin, and Vladimir Ulyantsev, "Learning Finite-State Machines with Classical and Mutation-Based Ant Colony Optimization: Experimental Evaluation," *BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence*, pp. 528-533, 2013. *Crossref*, https://doi.org/10.1109/BRICS-CCI-CBIC.2013.93

[34] M. Ostapchuk, and V. V. Solov'ev, "Minimizing the Finite-State Machines by Using the Values of Input Variables for Coding the Internal States," *Journal of Computer and Systems Sciences International*, vol. 57, no. 5, pp. 738-749, 2018. *Crossref*, https://doi.org/10.1134/S1064230718050106

[35] A. S. Klimowicz, and V. V. Solov'ev, "Structural Models of Finite-State Machines for their Implementation on Programmable Logic Devices and Systems on Chip," *Journal of Computer and Systems Sciences International*, vol. 54, no. 2, pp. 230-242, 2015. *Crossref*, https://doi.org/10.1134/S1064230715010074