Received 12 October 2022; revised 18 November 2022; accepted 19 November 2022. Date of publication 23 November 2022; date of current version 22 December 2022. The review of this article was arranged by Associate Editor James Morris.

Digital Object Identifier 10.1109/OJNANO.2022.3224061

Large-Scale Optimization of Decoupling Capacitors Using Adaptive Region Based Encoding Scheme in Particle Swarm Optimization

DINESH JUNJARIYA (Student Member, IEEE), AND JAI NARAYAN TRIPATHI (Senior Member, IEEE)

Department of Electrical Engineering, Indian Institute of Technology Jodhpur, Jodhpur 342037, India

CORRESPONDING AUTHOR: JAI NARAYAN TRIPATHI (email: jai@iitj.ac.in)

This work was supported by the Science and Engineering Research Board, Government of India, through SRG scheme under Grant SRG/2021/002239.

ABSTRACT Power delivery networks are responsible for supplying clean power to the integrated circuits. Power supply noise plays a critical role in determining the performance of high-speed very large scale integration circuits and systems. In order to maintain power integrity in high-speed systems, decoupling capacitors are used to maintain low impedance of the PDN to eventually minimize power supply noise. However, the discrete optimization problem of selecting decoupling capacitors becomes computationally challenging in the systems having stringent power integrity (PI) requirements. In this work, a novel approach using the Social-Learning Particle Swarm Optimization (SLPSO) technique along with Adaptive Region Search (ARS) is used to tackle the Large-Scale Optimization Problem (LSOP) of decoupling capacitor placement. Region Search (RS) is used to guide particles, followed by ARS to dynamical search for the local best positions and for particles to move faster across the search space while maintaining the diversity of the population. To demonstrate the proposed approach, three practical case studies are presented. The obtained results are compared with current state-of-the-art approaches. The proposed approach drastically reduces computation time and is consistent with better results than other approaches. This consistency of improvement in CPU time in the results of all the examples validates the proposed approach.

INDEX TERMS Decoupling capacitor placement, metahueristic optimization, particle swarm optimization, PDN, power integrity, social-learning particle swarm optimization.

I. INTRODUCTION

With the advancement in nanotechnology over the past few decades, it has become possible to fabricate electronic switching devices in nano-dimensions. In the realm of modern Very Large Scale Integration (VLSI) systems, it is desirable to have consumer products having multiple features as well as having very low cost. This has become possible by the nanometer-scale size of the transistors, which enable very high operating frequencies. Following Moore's law, so far the size of the transistors continues to get smaller. The influence of nanoscale dimensions, however, is considerably more noticeable than any other feature in the current state-of-the-art high-speed designs.

Signal Integrity (SI) and Power Integrity (PI) issues, in general, are becoming much more prevalent with the advancement of nanotechnology. In addition to the nano-dimensional switching devices, nanotechnology has also advanced passive interconnects in nano dimensions. However, primarily due to the bandwidth limitations of passive interconnects, SI/PI issues come into picture. Therefore, it is very much rational to relate SI/PI issues with nanotechnology. As due to the dimensions of transistors/switching devices being in nanoscale leads to a very narrow design margins; these nano devices based Integrated Circuits (ICs) have to deal with a number of signal integrity problems, including surface roughness, electromagnetic interference (EMI) effects, crosstalk, and reflection. Studying how interconnects affect system performance in terms of SI/PI is therefore very crucial [1], [2], [3], [4].

With the increasing operating frequencies of high-speed VLSI systems, demand for low-noise power supply to drive active circuits of ICs has become more important than before. Power Delivery Networks (PDNs) are responsible for supplying ideal DC-like (low noise) power to the load. PDNs consist of Voltage Regulator Modules (VRMs), on-chip interconnects, boards and package+die parasitics [5]. These components of a practical PDN contribute to a non-ideal impedance profile, which results in power supply ripples, also referred to as Power Supply Noise (PSN). The major contributor to PSN is Simultaneous Switching Noise (SSN) [6], [7]. The SSN originates from the switching activity of millions of transistors in the core circuitry. As the supply voltage decreases with the advancement of nanotechnology, PSN becomes more crucial to the performance of ICs at different frequency ranges, impacting the signal integrity of the high-speed systems significantly. As the technology nodes scale down, the supply voltage also lowers down while the clock frequency and density increases. Due to this increased complexity as well as the reduced noise budgets, the tolerance limits for power supply noise becomes much more relevant in the lower technology nodes, compared to the technology nodes having higher device dimensions. This work attempts to reduce power supply noise by optimization in in-package decoupling capacitors.

To maintain minimum power supply noise, PDN impedance should be controlled within the permissible limit. The worst-case peak current and noise provide a limit for the highest permissible impedance of the PDN. This highest permissible impedance of a PDN is also known as target impedance, Z_t, and is given as [8]:

$$Z_t = \frac{\Delta V_n}{I_{max_t}} \tag{1}$$

where I_{max_t} is the worst case transient current and ΔV_n is the maximum tolerable voltage noise present in the system. The probability of system functionality failure at a rated performance depends on the PDN ratio, which is defined as the ratio of PDN impedance (Z_{pdn}) to the target impedance Z_t. A value of less than one for this ratio indicates a low probability of PDN performance failure and vice versa. In practice, obtaining the low values of the PDN ratio and the PDN impedance is expensive, as it requires deploying more components and more layers in the package/board. Using decoupling capacitors (decaps) on the package or board to obtain a low PDN impedance is a simpler and more cost-effective solution. The decaps are selected based on the anti-resonance frequencies in the impedance profile of the PDN [8]. However, having multiple ports to place decaps and having multiple capacitors to choose from, increases the available decap-port combinations. Hence, the intuitive placement of decaps is not viable since placement and testing for each decap-port combination can be very tedious. The aim is to select a set of capacitors and corresponding ports on which they will be placed efficiently, meeting the system requirement. Computational Intelligence based methods have shown to be efficient and practical in tackling such large-scale optimization problem (LSOP) [9], [10].

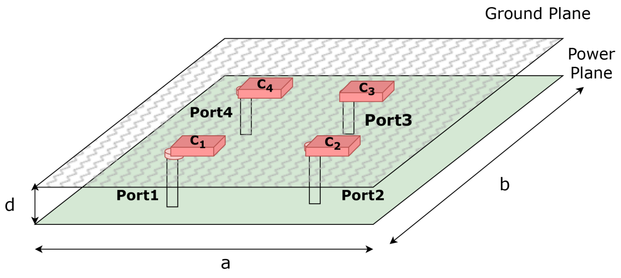

FIGURE 1. Decoupling Capacitor Placement on ports.

A proven time-effective optimization strategy for the decoupling capacitor optimization problem is to use metaheuristic algorithms [11]. Matrix-based metaheuristic optimization has recently been used to speed up the process even further [12]. There are many other metaheuristic algorithms for optimization, but Particle Swarm Optimization (PSO) has emerged as one of the best and quickest methods to solve such problems [13], [14]. However, the primary limitation of this technique is that PSO may take a very long time for large-scale optimization tasks to converge to a solution. Such large-scale optimization problems can be made time-efficient using adaptive region-encoded methods along with the PSO [15].

The rest of the article is as follows. Section II presents the description of the system used and the problem statement. Section III discusses the conventional PSO and SLPSO algorithms. Section IV deals with a brief explanation of SLPSO-ARS for the problem of optimization of decoupling capacitors. Section V discusses the results obtained from the experiments. Section VI concludes this paper.

II. PROBLEM DESCRIPTION

This study uses a practical system used earlier in [12], [16]. It's a usual practice to select capacitors based on the anti-resonance points in the PDN impedance profile and then to place them as close to the core circuit as feasible [8]. This arrangement of capacitors help to achieve desired taret impedance of the PDN.

Decap placement in the PDNs becomes extremely challenging when several ports are available, as shown in Fig. 1 and a large number of capacitors are available to choose from. The key objective of this work is to minimize the number of decaps used while maintaining the impedance of the PDN lesser than the target impedance (Z_t). As shown in Fig. 1, the power plane and the ground plane are always in a pair. Ports are the terminals that are used to place decoupling capacitors. Fig. 1 shows an example of such a power plane pair having multiple ports: $Port1$, $Port2$, $Port3$ and $Port4$ available to place decaps marked as C_1, C_2, C_3 and C_4, respectively. Based on the system requirement more number of ports can be introduced.

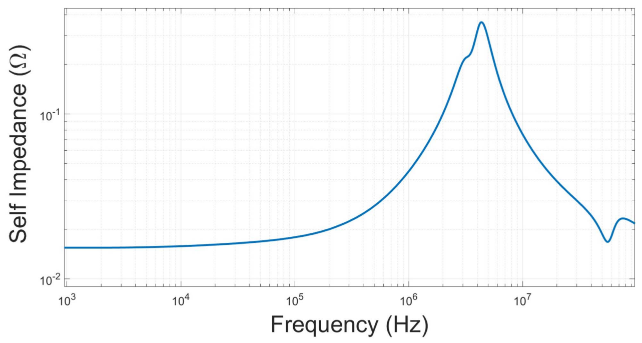

FIGURE 2. Self-impedance of PDN (without decoupling capacitors).

The self-impedance of the PDN under consideration, as measured at the IC pad, is shown in Fig. 2. It represents the total/cumulative impedance of all the components of the PDN: Voltage Regulator Module (VRM) and interconnects present on the package, chip and on the board. For the board and package models, a commercial 3D solver is used to extract s-parameters; and for the on-chip PDN, a Chip Power Model (CPM) [17] is used. The dimension of the Z_{pdn} considered for this study is $21\times21\times1391$, where number 21 refers to the number of ports, and the number 1391 represents the number of frequency points. Next, Z-parameters are derived from the S-parameters. This work focuses on in-package decap optimization since all the available ports are on the package.. As shown in Fig. 2, the maximum self-impedance value of this PDN when no decoupling capacitors are placed is 361.2 mΩ.

When decoupling capacitors are placed at the ports of the PDN, the updated cumulative self-impedance of the PDN, Z_{eq}, can be computed as [17] :

$$Z_{eq} = \left(Z_{pdn}^{-1} + Z_{decap}^{-1}\right)^{-1} \tag{2}$$

where Z_{pdn} represents the Z-parameter matrix of the PDN with dimension as $Z_{p\times p\times f}$. The Z-parameter matrix for the decoupling capacitors is represented by Z_{decap}, which is a diagonal matrix having the same dimension as Z_{pdn}. Z_{decap} has diagonal elements as impedance of the decoupling capacitors indexed at the port number they are placed on. The alternative formulation of (2) is:

$$Z_{eq} = \left(Y_{pdn} + Y_{decap}\right)^{-1} \tag{3}$$

The objective of this optimization problem is to reduce the maximum of self-impedance of the PDN in a given frequency range, below the target impedance (Z_t). This is a minimizing problem, hence the objective function for this study is the maximum value of the self impedance associated with the port, which is defined as :

$$Z_{obj} = \max(Z_{eq}(i, i, f)) \tag{4}$$

where f is some particular frequency and i is the port where self impedance is measured. Objective function defined in (4) is a function of two kinds of variables: the port numbers and the capacitors which are placed on the corresponding ports.

III. THEORY

A. PARTICLE SWARM OPTIMIZATION

This optimization algorithm was first introduced by Kenndy and Eberhart in 1995 [18]. Like many other optimization algorithms, it is based on swarm intelligence behaviour. However, unlike the popular metaheuristic algorithms, e.g. ant colony or stochastic diffusion search algorithm, genetic algorithms [19], [20] and virtual ant algorithm [21]; PSO does not need mutation operators, which makes it relatively simpler. At its core, it heavily relies on randomization and inter-particle communication. It becomes a lot simpler to implement PSO as, unlike the genetic algorithm, the parameters are not encoded/decoded into binary strings. Swarms are referred to as populations in PSO, and each member of the population is known as a particle. Particles are randomly placed in the defined search space, and the positioning of the particle outputs a solution to the target function which is to be optimised. Based on the solution obtained by the particle, it is moved across the search space with a velocity (V), which depends on its historical performance and the performance of other particles in the search space. The movement of a particle in the PSO algorithm is represented by its position (X) and velocity (V), which are given as :

$$V_{id}^{l+1} = V_{id}^{l} + r_1.c_1.\left(X_{id}^{p} - X_{id}^{l}\right) + r_2.c_2.\left(X_{id}^{g} - X_{id}^{l}\right) \tag{5}$$

$$X_{id}^{l+1} = X_{id}^{l} + V_{id}^{l+1} \tag{6}$$

Here, $d = 1, 2, \ldots, D$, where D represents the number of dimensions and l represents the current iteration value. Here, c_1 and c_2 are acceleration coefficients also known as cognitive component and social component, respectively; while r_1 and r_2 are randomly generated numbers in the range of [0,1]. The local best ($lbest$) is a particle's past position which gives the best fitness value of that particle so far and is represented by X_{id}^{p}; while X_{id}^{g} represents the current global best denoted by $gbest$ which is the position having the best fitness value among all the particles so far.

The inertia weight (ω) is an essential parameter of the PSO algorithm, and its selection is relevant to the balance between local and global search capabilities, impacting the convergence performance of the algorithm. A recommended value of ω uses the fewest number of iterations to discover the best solution. The inertia weight value (ω) in an iteration can be given as :

$$\omega = \omega_{min} + (\omega_{max} - \omega_{min}).\left(\frac{l_{max} - l}{l_{max}}\right) \tag{7}$$

Here, ω_{max} and ω_{min} represent the starting and ending values of ω during iteration l, while l_{max} represents the maximum number of iterations. With increasing maximum number of iterations (l_{max}), inertia weight ω decreases. During the early operations, when l is small, the algorithm's searching ability is strong due to the large value of ω. As the number of iterations increases, the local search becomes more refined as the value of ω is smaller than that during the initial movement. Considering inertia weight while the particle across search space, the

Algorithm 1: Particle Swarm Algorithm for Decap Optimization.

```
Input: $Y_{pdn} = Y_{p\times p\times f}, Y_{m\times f}, Z_t, NP, N_{itr}, c_1, c_2, \omega_{min}$
 and $\omega_{max}$
Output: $Z_{obj}, C, P$
1:  while ($Z_{obj} > Z_j$) do
2:    $N_d = N_d + 1; l = 0;$
3:    Generate initial population $X^{l=0}$ and velocity vector $V^{l=0}$
4:    Update initial personal best population $X^p$
5:    Check initial global best (minimum) $Z_{obj}$
6:    while ($l > l_{max}$) do
7:      $l = l + 1;$
8:      for $i = 1, 2, \ldots, NP$ do
9:        for $d = 1, 2, \ldots, N_d$ do
10:         Update velocity ($V^l_{i,d}$) as per (5)
11:         Update position ($X^l_{i,d}$) as per (6)
12:         Check for boundary conditions
13:       end for
14:       Compute $Z^l_{obj,i}$ for new position $X^l_i$
15:       Update $lbest$ position $X^p_i$ accordingly
16:     end for
17:     Check current $gbest$ position $Z_{obj}$ and $X^g$ at $Z_{obj}$
18:   end while
19: end while
Output: $Z_{obj} = max(Z_{eq}(1,1))$, where
 $(Z_{eq})_f = \left(Y^{-1}_{pdn} + Y^{-1}_{decap}\right)^{-1}_f, \forall \in [0, f_{max}], X^g = [P\ C]$
```

velocity of a particle is given as :

$$V^{l+1}_{id} = \omega V^l_{id} + r_1 c_1 \left(X^p_{id} - X^l_{id}\right) + r_2 c_2 \left(X^g_{id} - X^l_{id}\right) \tag{8}$$

After initializing particles randomly in the search space and updating their positions with the help of their velocities, the objective function is evaluated for each particle. The *lbest* and *gbest* are updated based on the calculation of fitness values of the particles from the objective function as

$$X^p_i = X^l_i, \text{ if } Z^l_{obj,i} < Z^p_{obj,i} \tag{9}$$

$$X^g = X^l_i, \text{ if } Z^l_{obj,i} < Z^g_{obj} \tag{10}$$

Fig. 3 depicts the process of update of velocity and position of a particle in PSO. The presented optimization problem discussed in Section II is computationally expensive, as its objective function requires the inversion operation of a large-size matrix. However, this work aims to propose a method that minimizes the number of fitness evaluations needed without compromising the convergence of the population in global optimum. The flow of the PSO algorithm used for decap placement can be well understood by Algorithm-1, where C and P are optimum capacitors and their corresponding ports.

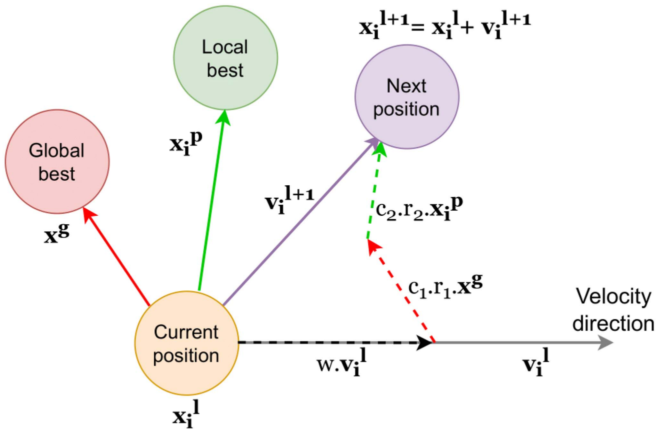

FIGURE 3. Position update of a particle in PSO.

B. SOCIAL LEARNING PARTICLE SWARM OPTIMIZATION

Large-scale optimization (LSOP) problems are the ones that contain large number of variables (dimensions), have a vast search space and may contain many local optima, making it challenging to solve them with typical evolutionary computation (EC) techniques. Among the several large-scale optimization algorithms, the Social Learning Particle Swarm Optimization (SLPSO) has been proven to provide consistence results [22]. SLPSO updates each particle (excluding the best particle) by learning from any particle within the current swarm that has a better fitness value than the particle itself, helping SLPSO to maintain the diversity of the swarm. However, large scale optimization problem (LSOP) demands that the algorithm retain diversity to look for the optimal global solution in a large search space and has a faster convergence rate.

The SLPSO has been proven to perform well with LSOPs [22]. Each particle in SLPSO, excluding the best one, learns from any particle that gives better fitness vale than itself. SLPSO lets particles learn from other particles rather than simply *gbest* and *pbest*, as in regular PSO. As a result, SLPSO can boost population diversity as particles are not too much influenced by *gbest*, allowing them to avoid prematurely approaching the local optima. At the start of each generation, particles in the current swarm are sorted based on their fitness ratings, from the worst to the best. The particles will then update each dimension of them by learning from only one particle (excluding the best particle) having better fitness value than it.

If multiple particles are better than a given particle, the current particle will choose one particle randomly to control its update. The particle may also consider position of other particles to guide its different dimensions. Thus, the population becomes more diverse since each particle may learn from several particles that give better solutions than itself. Particle i selects some particle k to learn from it in its d^{th} dimension as

$$v^{t+1}_{id} = r_1 v^t_{id} + r_2 \left(x^t_{kd} - x^t_{id}\right) + \epsilon r_3 . \left(\bar{x_d}^t - x^t_{id}\right) \tag{11}$$

$$x_{id}^{t+1} = \begin{cases} x_{id}^{t} + v_{id}^{t+1}, & \text{if rand}\,(0,1) < P_i, \\ \text{otherwise.} \end{cases} \tag{12}$$

In (12), r_1, r_2 and r_3 are three random numbers in [0,1], $\mathbf{x}$ and $\mathbf{v}$ are position and velocity vectors, respectively; while t represents the current generation; ϵ is known as social influence factor and $\bar{x}_d^t$ is the average value of position vector in d^{th} dimension. Here, k is distinctly selected for every dimension, however, k has to be a particle better than the current particle. P_i, also known as learning probability, controls whether or not the particle needs to be updated. It is related to the the rank of the particle. Each particle has a distinct learning probability, and better particles will have lower learning probabilities. Since particles are sorted from the worst to the best, P_i is calculated as :

$$P_i = \left(1 - \frac{i-1}{NP}\right)^{\mu \log\left(\left\lceil \frac{D}{M} \right\rceil\right)} \tag{13}$$

where, i is the i^{th} particle from the population, NP and D represent swarm size and the dimension of the problem respectively; μ and M are set to 0.5 and 100, respectively, (accordingly to [15], [22]). P_i signifies the probability of learning for a particle with different fitness values. One with a worse fitness value tends to have a larger learning probability (as particles are sorted from the worst to the best) and value of i will be lower for that particle, thus higher chances to get its position updated. These particle search for a better solution and vice-versa for a particle with a higher index or with a relatively good fitness value. P_i also depends on the dimension of the problem. When D is large, value of P_i will decrease to retain the population diversity in the search plane, avoiding early convergence to a local solution.

IV. SLPSO-ARS FOR DECAP OPTIMIZATION

Even as SLPSO is reported to deliver good results for LSOPs, it often results in a slow convergence. Since, to maintain diversity among particles, it uses different learning information to guide different particle positions and dimensions. Also, not all the particles are updated in each generation, which is a drawback. In order to improve the convergence speed, a local search strategy named Adaptive Region Search (ARS) is used, combined with a region encoding scheme (RES). This encoding scheme and local search strategy with SLPSO form SLPSO-ARS [15].

A. REGION ENCODED SCHEME (RES)

RES forms a radius r_i around a particle i which is no longer a single point $x_i = \{x_{i1}, x_{i2} \ldots x_{iD}\}$ but enclosed space with some radius r_i. Radius r_i has x_i as the center of the region. This allows the particle to give a region-based solution rather than a point-based solution. This region allows covering a wide range in the search space.

Point-based encoding scheme and region-based encoding scheme are compared in Fig. 4(a) and (b). In Fig. 4, solid dots represent particles $p_i = \{p_1, p_2, p_3, p_4, p_5\}$. As in Fig. 4(a), a particle holds one single position in a point-based encoding scheme; however, in region based encoding scheme, as in Fig. 4(b), each particle p_i has its own region defined by a radius r_i. Every particle in region search (RS) can search locally in its region within radius r_i and finds the best position for the particle in that region. This method often helps in improving the convergence faster as local search in large spaces is more efficient than an evolutionary operation search.

If conventional PSO is used, the particle x_i learns from other particles with better current position but are in proximity to a local optimum solution, drawing the particle i towards the near local optima. To avoid convergence to a local optimum, RES comes in handy to find global optimum and improves the convergence speed of the algorithm.

B. ADAPTIVE REGION SEARCH (ARS)

ARS is established on RES, improving the chances of the particle obtaining a better solution by region search. Region search allows the radius r_i of the region of the particle x_i to change and adjust adaptively.

In Fig. 4(c), the ARS strategy is shown in which five particles p_1, p_2, p_3, p_4 and p_5 are placed randomly and the optimization problem is to search for the top position of the landscape. The arrows represent the directions that the particles are about to follow. When comparing to Fig. 4(b), the particles p_2, p_3 and p_5 have found better position p_2^{new}, p_3^{new} and p_5^{new}, respectively, in the current generation. So, they move to the new positions and their radii r_i are increased as these new better positions indicate that the optimum position is near by and therefore increasing radius would let the particles move faster to the desired optimum position. Particles p_1 and p_4 cannot find better positions compared to the original ones which indicates that their current positions are in promising regions, and a large radius may make the particle overstep the optimal solution. ARS strategy therefore reduces the radius r_1 and r_4 to r_1^{new} and r_4^{new}, respectively.

Thereafter, RS is performed on the first P particles which give best fitness values. The dimension of a particle is perturbed inside the region with radius r_i centred at particle with a probability ρ, also known as perturbation probability. In simpler words, it is a search in proximity of particle x_i within radius r_i. Suppose, a better solution is found while perturbation of a particle, the current position of the particle is replaced by perturbed position and the region with radius r_i centres around the new position. This change of radius allows the population to move across the local optimum positions and thus increases the speed of convergence in LSOPs. The d^{th} dimension of the particle which is perturbed is generated as :

$$x_i' = \begin{cases} x_{i_d} + \mathcal{N}(0,1).r_i, & \text{if } d == n \text{ or rand}\,(0,1) < \rho \\ x_{i_d}, & \text{otherwise.} \end{cases} \tag{14}$$

Here x_i' and x_{i_d} are d^{th} dimension position of the perturbed particle and the original particle, respectively. The r_i is the radius of the region enclosed by the particle i, and n is a randomly generated integer uniformly distributed in [1, D],

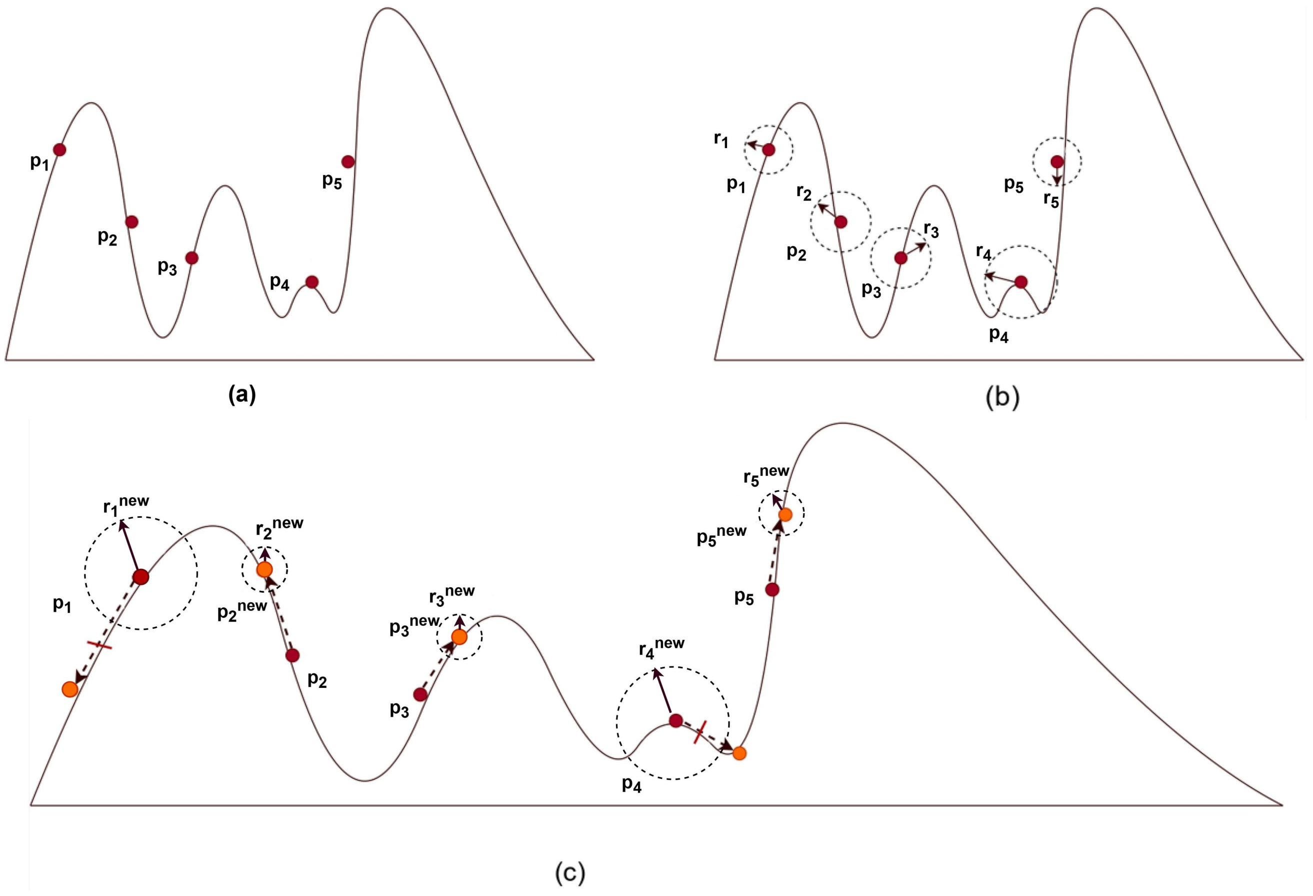

FIGURE 4. **(a) Point-based encoding scheme. (b) Region Encoding Scheme. (c) Adaptive Region Search strategy.**

where D being the dimension of the problem (it checks that the perturbed particle x_i' is not same as the original particle x_i). $\mathcal{N}(0, 1)$ is a randomly generated Gaussian distributed value. Perturbation probability (ρ) determines if the position dimension of the particle is changed or not while region search. Since we only select a subset of the best P number of particles according to the fitness values, from the current swarm to do a region search, these particles may already have very valuable information in most of the dimension of the position of a particle, and a perturbation in all dimensions may be damaging. Hence, a relatively low value of ρ is advantageous to assure the best particle properties substantially, and most of the favourable dimensions will not be lost.

Suppose, the value is set too high for ρ in that case, the best particles will have more dimensions modified, which will cause the perturbed particles to perform poorly because of the more significant difference from the original best particles. As a result, the value of ρ in SLPSO-ARS is not very large. Fitness value of a perturbed particle x_i' decides the value of x_i. If the fitness value of x_i' is better than that of x_i, it is replaced with x_i'. This process is repeated for T perturbed particles in the search region with radius r_i.

After T times of region search, if no perturbed particle that is better than the original particle is discovered, the original particle might just have discovered an approximate global or local optimal solution, which can also be seen as an optimal solution in its defined region. In this case, the radius of the region enclosed by the particle will be reduced. If not, the radius of the region enclosed by the particle will be expanded to seek the optimal solution across a larger search region. The radius of the i^{th} particle is updated as :

$$r_i = \begin{cases} r_i.c, & \text{Among T particles, no better particle is found} \\ r_i/c, & \text{otherwise.} \end{cases} \tag{15}$$

Here c is the scaling parameter lying in range of (0,1). The radius of the region enclosed by the particle, r_i has a maximum value, denoted by r_{max}. Set $r_i = r_{max}$ if the value of r_i exceeds r_{max}. Over the course of evolution, the value of r_{max} gradually decreases. r_{max} can be determined as :

$$r_{max} = r_0.\frac{FEmax - FE + 1}{FEmax}, \tag{16}$$

here radius is initialized with radius r_0. $FEmax$ and FE represent maximum number of fitness evaluations and current number of fitness evaluation, respectively.

C. SLPSO-ARS

In order to enhance the effectiveness of SLPSO for fast convergence in LSOPs, SLPSO-ARS is proposed in [15]. It incorporates the novel local search technique known as ARS, with SLPSO which can speed up swarm convergence despite preserving population variety. The evolutionary process primarily uses SLPSO in SLPSO-ARS, but the key modification is adding the ARS strategy for guiding the movement of the particles. All of the particles in the current swarm are ranked from the worst to the best according to their fitness values at the end of each generation. After that, a region search is conducted using the top P particles. Given that, choosing only a relatively small portion of the population to perform RS, the algorithm maintains the property of the original population to a considerable extent to preserve the diversity since other particles are unchanged. The algorithm will also choose the top P particles to perform RS based on a rank, which has a better probability of looking for adjacent optima. As a result, the ARS method can help accelerating population convergence to a global optimum.

SLPSO-ARS only chooses a small number of particles with higher rank to carry out RS, which not only helps in saving FE but also has a greater likelihood of discovering the close-by optimum solutions. The region is initialized with radius r_0 as :

$$r_0 = \frac{ubound - lbound}{10}, \tag{17}$$

where $lbound$ and $ubound$ represent the lower and upper bounds of the search space, respectively.

D. PROPOSED APPROACH

As described and discussed above, SLPSO-ARS significantly reduces the computation time for LSOPs. Therefore, in this work it is proposed to solve the present optimization problem using SLPSO-ARS. The complete flow of SLPSO-ARS can be understood by Fig. 5.

Using the flow given in Fig. 5, the optimization problem is mapped to SLPSO-ARS. The steps used to solve the problem is summarised in Algorithm 2. Further, the pseudo-code in Algorithm-2 presents a step-by-step algorithmic structure of SLPSO-ARS. The algorithm takes the admittance matrix ($Y = Z^{-1}$) of PDN as input along with target impedance, the number of perturbed particles and other SLPSO parameters. The particles are distributed across the search space, which in this case are the sets of the capacitors and the ports on which they are place on.

SLPSO-ARS restricts the movement of these particles to search for local optimum in their respective regions which are adaptively expanded or reduced in accordance with the better fitness values of other particles and its own perturbed particle. A particles selected to perform perturbation is based on its current fitness value. This selective choice of particles to be perturbed in the population reduces the total number of fitness evaluations (FE) to be performed. A particle moves to a new position of its perturbed particle if the fitness value obtained is better than its own fitness value, which in present optimization problem is the maximum impedance of the PDN over entire frequency range of interest. Outputs of the proposed SLPSO-ARS algorithm are the capacitors required to maintain desired impedance along with their corresponding ports and the maximum impedance which can be achieved across the frequency range of interest.

Algorithm 2: SLPSO-ARS Algorithm for Decap Optimization.

```
Input: $Y_{pdn} = Y_{p\times p\times f}, Y_{m\times f}, Z_t, N, r_1, r_2$ and $r_3$
Output: $Z_{obj}, C, P$
1:  $N = 100; \varepsilon = 0.1; P = 5; T = 5; \rho = 0.01;$
    $c = 0.5;$
2:  Randomly generate $N$ particles, calculate the fitness
    value of each particle and set initial radius to $r_0$
3:  For SLPSO, sort all the particles from the worst to
    thalgorithme best fitness value;
4:  FEs = FEs + N;
5:  while $FEs \leq maxFEs$ do
6:    for $i = 1$ to $N - 1$ do
7:      if $rand(0, 1) \leq P_i$ then
8:        for $d = 1$ to $D$ do
9:          $k = randi\_int(i + 1, N)$;
10:         Update the particles using (11) and (12);
11:       end for
12:       Calculate the fitness value of particle i;
13:       FEs = FEs + 1;
14:     end if
15:   end for
16:   Sort all the particles according to the fitness
      values and select the top $P$
17:   particles;
18:   // ARS
19:   for $i = 0$ to $P$ do
20:     for $j = 1$ to $T$ do
21:       n = rand_int(1,D);
22:       for $d = 1$ to $D$ do
23:         Perturb the particle according to (14);
24:       end for
25:       if $f(x_i') < f(x_i)$ then
26:         $x_i = x_i'$;
27:       end if
28:       FEs = FEs + 1;
29:     end for
30:     Modify the region radius $r_i$ of the particle $i$
        according to (15).
31:   end for
32:   Sort all the particles based on the fitness values
      from the worst to best.
33: end while
Output: $Z_{obj} = max(Z_{eq}(1, 1)$, where
  $(Z_{eq})_f = (Y_{pdn}^{-1} + Y_{decap}^{-1})_f^{-1}, \forall \in [0, f_{max}], X^g = [P\ C]$
```

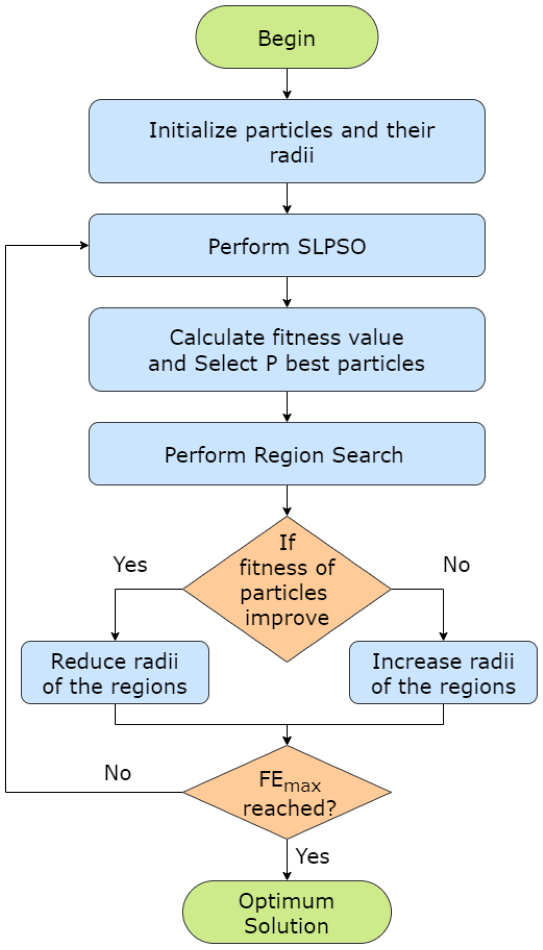
Begin
Initialize particles and their radii
Perform SLPSO
Calculate fitness value and Select P best particles
Perform Region Search
If fitness of particles improve
Yes
No
Reduce radii of the regions
Increase radii of the regions
No
FE$_{max}$ reached?
Yes
Optimum Solution

FIGURE 5. Flowchart for genral SLPSO-ARS.

V. RESULTS

As described in [8], [23], [24], S-parameters are found to be more accurate than RLC parameters for impedance modelling; hence this work uses S-parameter data from PDN and decoupling capacitors for formulating the optimization problem. Three case studies are performed with their respective datasets of decaps. All three case studies contain the same PDN data but have different dimensions of the decaps datasets. The dimesnsions of the datasets are as following:

- Case-study 1: 1000 × 1391
- Case-study 2: 2000 × 1391
- Case-study 3: 3348 × 1391

In these case studies, the datasets have 1000, 2000 and 3348 decaps to choose from for placement. The dimension of Z_{pdn} used for this study is $21\times21\times1391$, where the number of ports is represented by 21 and 1391 denotes the frequency data points. Port-1 is chosen to measure the self impedance Z_{11}, to keep the equivalent self-impedance minimum. As Port-1 is used to measure self-impedance, rest 20 ports are available for the placement of decoupling capacitors. To meet the system requirements, Z_t is kept as 60 mΩ.

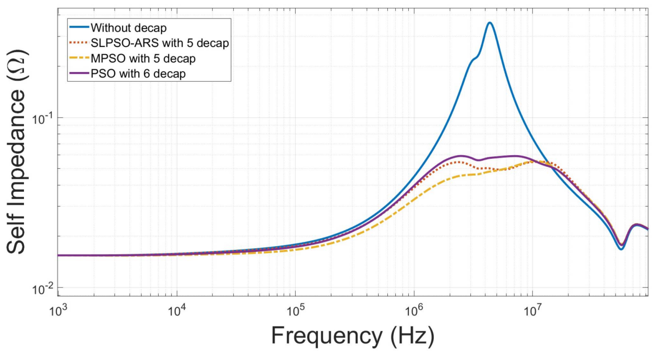

FIGURE 6. Optimal Impedance of PDN for Case-study 1 .

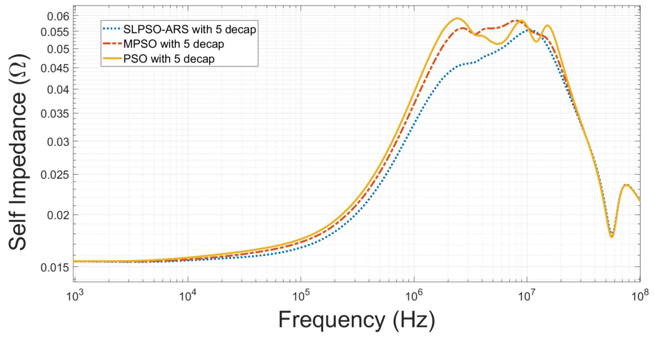

FIGURE 7. Optimal Impedance of PDN for Case-study 2.

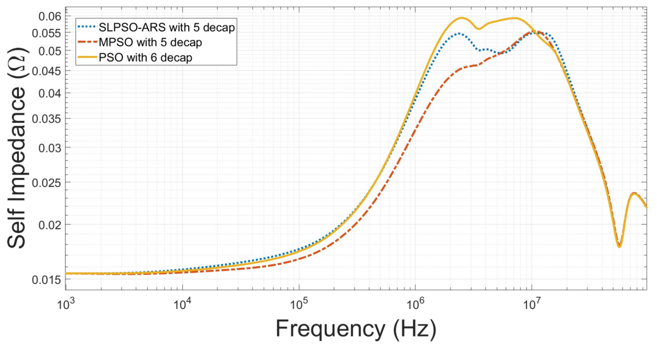

FIGURE 8. Optimal Impedance of PDN for Case-study 3.

The population size for MPSO and PSO is set to 50, while for algorithm using SLPSO-ARS it has been set to 100. Maximum fitness evaluation performed ($FEmax$) is set at 1000 for SLSPO-ARS. These studies are done using MATLAB R2019b and executed on a system with Intel Xeon 6 cores 3.6 GHz and 64 GB of RAM. The inertia weight parameter of PSO and MPSO: $w_{max} = 0.9$ and $w_{min} = 0.4$, while acceleration coefficient are set at $c_1 = 1.49$ and $c_2 = 1.48$. Results for 10 separate runs performed for all three case studies are shown in Table 1. Table 1 consists of the CPU time required for computation, T (in sec), and the decaps used for obtaining desired impedance i.e. N_d.

The optimum value of the self impedance of the PDN obtained by three algorithms using the minimum number of decaps is shown in Figs. 6, 7, and 8. Fig. 6 shows impedance

TABLE 1. Simulation Results for 10 Independent Runs

Case-study 1						Case-study 2						Case-study 3					
PSO [11]		MPSO [12]		SLPSO-ARS		PSO [11]		MPSO [12]		SLPSO-ARS		PSO [11]		MPSO [12]		SLPSO-ARS	
Nd	T	Nd	T	Nd	T	Nd	T	Nd	T	Nd	T	Nd	T	Nd	T	Nd	T
8	903.69	6	646.4	6	279.38	6	679.61	7	702.4	6	268.279	7	771.43	5	547.63	6	297.71
7	740.43	7	722.35	5	233.43	8	879.22	7	727.41	7	341.9	8	870.43	6	676.77	5	206.94
6	687.65	6	638.81	6	260.25	6	693.18	5	568.89	6	276.03	7	739.43	6	521.29	6	261.02
6	648.99	6	672.88	5	249.76	7	793.64	5	580.77	6	280.96	6	725.53	7	734.31	6	283.16
5	564.25	7	794.14	7	352.07	5	605.62	6	618.15	7	337.27	8	908.68	8	782.61	5	237.35
6	636.43	5	561.63	6	270.18	7	863.34	8	832.32	7	338.02	7	747.06	6	724.95	6	260.2
8	843.69	5	621.53	6	277.71	7	877.96	7	752.48	5	201.57	6	711.82	6	560.37	7	312.19
6	654.12	6	667.65	7	329.24	6	796.44	6	623.45	7	330.8	7	748.45	5	512.86	7	320.47
7	772.231	7	759.74	5	237.52	6	668.04	6	674.78	6	289.85	7	766.86	6	630.2	6	259
8	851.197	5	604.34	7	332.55	7	831.16	6	663.68	5	221.94	9	1000.23	6	578.23	6	281.63

TABLE 2. Performance Summary of Algorithms

Criteria	Case-study 1			Case-study 2			Case-study 3		
	PSO [11]	MPSO [12]	SLPSO-ARS	PSO [11]	MPSO [12]	SLPSO-ARS	PSO [11]	MPSO [12]	SLPSO-ARS
N_{avg}	7	6	6	7	7	6	7	6	6
N_{min}	5	5	5	5	5	5	6	5	5
T	730.26	668.94	282.20	768.82	674.43	288.66	798.99	626.92	271.96
SLPSO-ARS gain in CPU time	**w.r.t. PSO**		**61.35%**	**w.r.t. PSO**		**62.45%**	**w.r.t. PSO**		**65.96%**
	w.r.t MPSO		**57.81%**	**w.r.t. MPSO**		**57.19%**	**w.r.t. MPSO**		**56.61%**

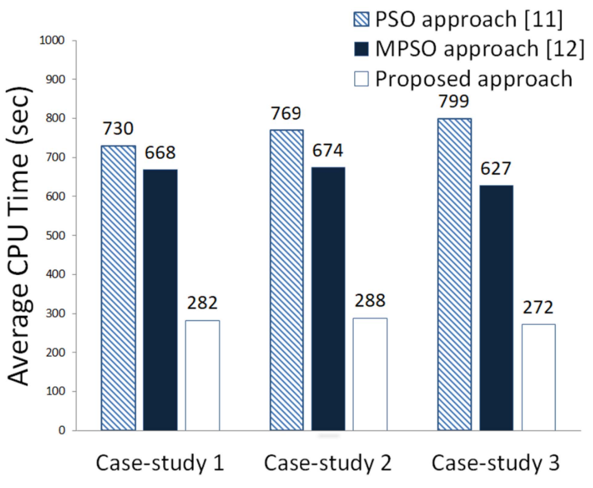

FIGURE 9. Average computational time comparison of PSO, MPSO and SLPSO-ARS for all case studies.

for Case-study 1 along with impedance of PDN without using any decap. The impedance profile obtained from all the case studies are similar in all approaches, however, the computation time differs. The proposed method of SLPSO-ARS has reduced computation time significantly, as shown in Fig. 9.

Performance comparison is presented in Table 2 for all the case studies. N_{avg} and N_{dmin} represent the average number of decaps required and the minimum number of decaps required in the runs in Table 1, where T is the average CPU run time. Gain in CPU run time of 61.35%, 62.45% and 65.96% are achieved when compared to PSO for Case-study 1, Case-study 2 and Case-study 3, respectively, and a CPU run time gain of 57.81%, 57.19% and 56.61% is observed when compared to MPSO for Case-study 1, Case-study 2 and Case-study 3, respectively, as shown in Table 2.

In the presented case studies, the average decoupling capacitors required to meet the required impedance are 6 and 7, as shown in Table 2. However, except in the PSO algorithm in Case-study 3, the least number of decaps required for system performance comes out to be five decaps.

In practice, more capacitors can be available for placement, and more capacitors may be required to achieve PDN impedance within the permissible range, which increases the dimensions in the problem. The gain in computation time for SLPSO-ARS will significantly improve as the problem increases its dimensionality.

VI. CONCLUSION

A method for effective selection and placement of decoupling capacitors in a power delivery network is discussed in this study. The proposed method uses Social Learning Particle Swarm Optimization (SLPSO) and Adaptive Region Search (ARS) to improve the computation time of LSOP of decoupling capacitors placement compared to the conventional metaheuristic algorithms. The SLPSO-ARS optimization strategy outperforms traditional metaheuristic algorithms in terms of computational time efficiency. This study considers a practical power distribution network, and the target impedance of the system is achieved using the least number of decoupling capacitors. Three case studies that compare the performance of the conventional and proposed methodologies are also provided.

REFERENCES

[1] T. Pathade, Y. Agrawal, R. Parekh, and M. G. Kumar, "Structure fortification of mixed CNT bundle interconnects for nano integrated circuits using constraint-based particle swarm optimization," *IEEE Trans. Nanotechnol.*, vol. 20, pp. 194–204, 2021.

[2] M. Sanaeepur, "Crosstalk delay and stability analysis of MLGNR interconnects on rough surface dielectrics," *IEEE Trans. Nanotechnol.*, vol. 18, pp. 1181–1187, 2019.

[3] M. Sahoo, P. Ghosal, and H. Rahaman, "Modeling and analysis of crosstalk induced effects in multiwalled carbon nanotube bundle interconnects: An ABCD parameter-based approach," *IEEE Trans. Nanotechnol.*, vol. 14, no. 2, pp. 259–274, Mar. 2015.

[4] M. R. Khezeli, M. H. Moaiyeri, and A. Jalali, “Analysis of crosstalk effects for multiwalled carbon nanotube bundle interconnects in ternary logic and comparison with Cu interconnects,” *IEEE Trans. Nanotechnol.*, vol. 16, no. 1, pp. 107–117, Jan. 2017.

[5] J. N. Tripathi, V. K. Sharma, and H. Shrimali, “A review on power supply induced jitter,” *IEEE Trans. Compon. Packag. Manuf. Technol.*, vol. 9, no. 3, pp. 511–524, Mar. 2019.

[6] H. H. Chen and D. D. Ling, “Power supply noise analysis methodology for deep-submicron VLSI chip design,” in *Proc. 34th Annu. Des. Automat. Conf.*, 1997, pp. 638–643.

[7] M. Saint-Laurent and M. Swaminathan, “Impact of power-supply noise on timing in high-frequency microprocessors,” *IEEE Trans. Adv. Packag.*, vol. 27, no. 1, pp. 135–144, Feb. 2004.

[8] J. N. Tripathi, J. Mukherjee, P. R. Apte, N. K. Chhabra, R. K. Nagpal, and R. Malik, “Selection and placement of decoupling capacitors in high speed systems,” *IEEE Electromagn. Compat. Mag.*, vol. 2, no. 4, pp. 72–78, Oct.–Dec. 2013.

[9] R. Kruse et al., *Computational Intelligence*. Berlin, Germany: Springer, 2011.

[10] M. Gendreau and J.-Y. Potvin, “Metaheuristics in combinatorial optimization,” *Ann. Oper. Res.*, vol. 140, no. 1, pp. 189–213, 2005.

[11] S. Hemaram and J. N. Tripathi, “Metaheuristic optimization of decoupling capacitors in a power delivery network,” in *Proc. IEEE Int. Joint EMC/SI/PI EMC Europe Symp.*, 2021, pp. 554–558.

[12] A. Jain, H. Vaghasiya, and J. N. Tripathi, “Efficient selection and placement of in-package decoupling capacitors using matrix-based evolutionary computation,” *IEEE Open J. Nanotechnol.*, vol. 2, pp. 191–200, 2021.

[13] S. Hemaram and J. N. Tripathi, “Optimal design of a decoupling network using variants of particle swarm optimization algorithm,” in *Proc. IEEE Int. Symp. Circuits Syst.*, 2021, pp. 1–5.

[14] P. Kadlec, M. Marek, M. Štumpf, and V. Šeděnka et al., “PCB decoupling optimization with variable number of capacitors,” *IEEE Trans. Electromagn. Compat.*, vol. 61, no. 6, pp. 1841–1848, Dec. 2019.

[15] J.-R. Jian, Z.-G. Chen, Z.-H. Zhan, and J. Zhang, “Region encoding helps evolutionary computation evolve faster: A new solution encoding scheme in particle swarm for large-scale optimization,” *IEEE Trans. Evol. Comput.*, vol. 25, no. 4, pp. 779–793, Aug. 2021.

[16] H. Vaghasiya, A. Jain, and J. N. Tripathi, “A machine learning based metaheuristic technique for decoupling capacitor optimization,” in *Proc. IEEE 26th Workshop Signal Power Integrity*, 2022, pp. 1–4.

[17] E. Kulali, E. Wasserman, and J. Zheng, “Chip power model - A new methodology for system power integrity analysis and design,” in *Proc. IEEE Elect. Perform. Electron. Packag.*, 2007, pp. 259–262.

[18] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Proc. Int. Conf. Neural Netw.*, 1995, vol. 4, pp. 1942–1948.

[19] D. E. Goldberg, *Genetic Algorithms*. Noida, UP, India: Pearson Educ. India, 2013.

[20] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA, USA: Addison-Wesley, 1989.

[21] X.-S. Yang et al., “Application of virtual ant algorithms in the optimization of CFRP shear strengthened precracked structures,” in *Proc. Int. Conf. Comput. Sci.*, 2006, pp. 834–837.

[22] R. Cheng and Y. Jin, “A social learning particle swarm optimization algorithm for scalable optimization,” *Inf. Sci.*, vol. 291, pp. 43–60, 2015.

[23] K.-B. Wu et al., “Optimization for the locations of decoupling capacitors in suppressing the ground bounce by genetic algorithm,” in *Proc. Photon. Electromagn. Res. Symp.*, 2005, pp. 411–415.

[24] S. Piersanti, F. de Paulis, C. Olivieri, and A. Orlandi, “Decoupling capacitors placement for a multichip PDN by a nature-inspired algorithm,” *IEEE Trans. Electromagn. Compat.*, vol. 60, no. 6, pp. 1678–1685, Dec. 2018.

DINESH JUNJARIYA (Student Member, IEEE) is currently working toward the Undergraduate degree in electrical engineering with the Indian Institute of Technology (IIT), Jodhpur, India. His research interests include VLSI, integrated circuits (IC’s), electromagnetic interference for high speed chip-package-systems, neuromorphic circuits, power, and signal integrity.

JAI NARAYAN TRIPATHI (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from the Indian Institute of Technology Bombay, Mumbai, India, in 2014. He is currently an Assistant Professor with the Indian Institute of Technology, Jodhpur, India, and an Adjunct Research Professor with Carleton University, Ottawa, ON, Canada. He was with STMicroelectronics, India, for almost 7 years, where he was involved with the design issues of various high-speed Systems-on-Chip (SoCs). In 2016 and 2017, he was a Visiting Scientist with the Politecnico di Torino, Turin, Italy, where he was also a Visiting Postdoctoral Fellow, in 2016. He has authored or coauthored more than 100 research papers in refereed journals, as book chapters and in the proceedings of top international conferences. His research interests include signal integrity, power integrity, electromagnetic interference/electromagnetic compatibility, metaheuristic optimization, and RF circuits. He was the recipient of the Young Investigator Training Program Research Award by Associazione Di Fondazioni E Di Casse Di Risparmio Spa (ACRI), Italy, in 2016 and 2017, consecutively, and the Grant in 2021 for a period of two years from Department of Science and Technology, Government of India, to work on a power integrity problem. His papers were also the recipient of the Best Paper awards in a couple of international conferences. He was an Invited Speaker in IEEE EDAPS 2015, Seoul, South Korea, where he was also a Session Co-Chair for the session High-Speed Channels and Interconnects. Dr. Tripathi was the TPC Member for more than 20 international conferences including the premier conferences such as IEEE EPEPS, IEEE VLSI Design, IEEE EDAPS and the TPC Co-Chair for IEEE EDAPS 2018. He was also a Reviewer for many international journals, such as IEEE TCAS-I: Regular Papers, IEEE TCAS-II: Express Briefs, IEEE TVLSI, IEEE TEMC, IEEE TPES, PIER, Microelectronics Journal. He has delivered invited talks with various universities and international forums. He is the Member of a Technical Committee TC-12 EDMS of IEEE EPS Societyand the Best Paper Award Committee of IEEE EDAPS 2020. He is an Associate Editor for IEEE TRANSACTIONS ON SIGNAL AND POWER INTEGRITY, IEEE TRANSACTIONS ON COMPONENTS, PACKAGING AND MANUFACTURING TECHNOLOGY and IEEE Electromagnetic Compatibility Magazine, and IEEE OPEN JOURNAL OF CIRCUITS AND SYSTEMS.