A Comparative Study on Optimization Algorithms in PowerSynth 2

Mehran Sanjabiasasi
EECS, University of Arkansas
Fayetteville, AR
mehrans@uark.edu

Imam Al Razi
Intel Corporation
Hillsboro, OR
imam.al.razi@intel.com

H. Alan Mantooth
EECS, University of Arkansas
Fayetteville, AR
mantooth@uark.edu

Yarui Peng
EECS, University of Arkansas
Fayetteville, AR
yrpeng@uark.edu

***Abstract*—Multi-Chip Power Modules (MCPM) are a critical component in power conversion applications. Power modules and their layout optimization have been considered a crucial step to achieving maximum performance. PowerSynth 2 (PS2) is an electronic design automation (EDA) tool for the generation and optimization of power module layouts. Currently, v2.0 uses NSGA-II and Randomization for layout optimization. However, existing NSGA-II implementation is not aware of the layout hierarchy, resulting in a less desirable solution space than Randomization. To address this limitation, this research presents a hierarchical optimization framework for the layout synthesis process in PowerSynth 2. Experimental results show that proposed hierarchical algorithms improve over the existing optimization algorithms. Moreover, MOPSO is faster than NSGA-II to converge to Pareto Front with similar solution space coverage.**

***Index Terms*—Electronic Design Automation (EDA), Layout Optimization, Hierarchical Optimization, PowerSynth**

I. Introduction

Nowadays, power converter design automation is attracting the attention of researchers. As power modules are the most critical components in power converters, an efficient and effective design methodology is a critical research problem [1]. There is some previous research on automatic power module design, mostly focusing on three parts: layout generation, modeling, and optimization. However, it is still an open question to improve computational runtime, modeling accuracy, design flexibility, and scalability.

A graph model is proposed in [2] to describe heterogeneous layouts with all interconnectivity and design constraints. They implement NSGA-II for tradeoffs in loop inductance and branch mismatch. A sequence methodology is proposed in [3], [4], to describe the relative position of switches and devices. For layout optimization, a 1D binary design string is considered with the Genetic Algorithm (GA). Another research [5] developed a Multi-Objective Electro-Thermal design framework for chip layout optimization of power modules by NSGA-II. In [6] the module's geometry and layout are optimized to reduce its maximum temperature and capacitive coupling to the baseplate. ANSYS and Multi-objective Genetic Algorithm are used for analysis and design. Researchers in [7] introduced a novel high-order finite element algorithm to optimize the thermal layout of 3D stacked multichip modules. For optimal thermal layout distribution, Particle Swarm Optimization (PSO) is used to solve the slow convergence and challenging global optimization problems encountered in traditional algorithms.

The current leading framework for MCPM layout optimization is PowerSynth [8]. In [9], the hierarchical corner stitching data structure with a constraint graph evaluation technique is used to optimize the MCPM layouts. The PowerSynth 2 [10] has demonstrated a complete and lab-validated design automation flow for high-density (2D/2.5D/3D) and heterogeneous designs. Currently, two optimization algorithms are considered in v2.0: NSGA-II and Randomization (RAND).

In this paper, our key contributions are: (1) A hierarchical optimization framework for power module layout optimization with improved NSGA-II implementation; (2) A new layout optimization algorithm based on Multi-Objective Particle Swarm Optimization (MOPSO); (3) A qualitative comparative study using various performance indicators on the latest optimization algorithms.

II. Existing Optimization Algorithms in PS2

The PowerSynth v2.0 uses NSGA-II [11] and the Randomization Algorithm [12] for layout optimization. For NSGA-II, the layout engine encodes the layout information into a design string. The optimization algorithm then performs crossover and mutation to create a new generation of offspring, which is evaluated in terms of fitness. The process continues to filter out better-fitted design strings, which are then decoded by the layout engine into final design layouts. For RAND, the layout engine varies the edge weights randomly while evaluating the hierarchical constraint graphs. This build-in algorithm tightly couples with the layout generation process and allows for all generated layouts to be evaluated in parallel.

A. Flat-Level Genetic Algorithm (Old NSGA-II)

NSGA-II is a popular elitist non-dominated sorting genetic algorithm used in multi-objective optimization. It starts with an initial population. These solutions are evaluated and ranked based on objective values. Then, binary tournament selection is applied to the initial population to select parents. Then crossover and mutation operators are implemented to generate

This material is based on work supported by The National Science Foundation under Grant No. EEC-1449548. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the National Science Foundation.

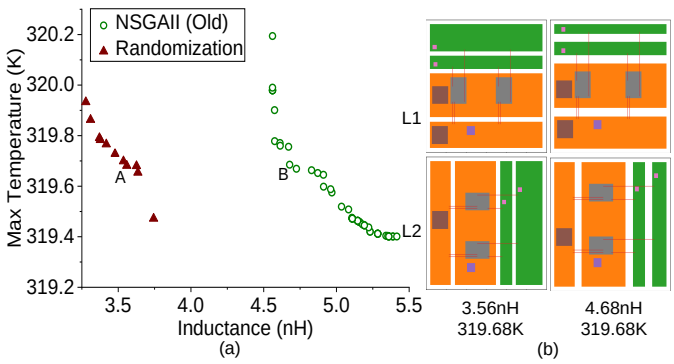

Fig. 1. (a) Parato Front comparison between existing RAND and NSGA-II on a 3D Design, with (b) two selected layouts.

offspring populations. Crossover for creating offspring swaps one part or more between two parents. On the other hand, mutation only changes one or some variables of each parent to create offspring. The probability for crossover is high, while low for mutation. After that, they are evaluated for fitness function values. The Non-dominated sorting is performed on a combination of both the offspring population and the current population in each generation to select the best non-dominated set for the next generation. Eventually, after sufficient generations of selection, the Pareto Front will be created.

B. Randomization (RAND)

RAND is a built-in algorithm based on the layout generation procedure. It can generate an arbitrary layout number of solutions by randomizing edge weights of constraint graphs. The detailed algorithm is described in [12]. Since the layout generation is a hierarchical process [9] and the RAND algorithm follows the hierarchy structure of the layout engine, compared with NSGA-II, which is unaware of the layout hierarchy, the RAND algorithm performs better in terms of solution distribution and spreading. However, the NSGA-II can reach optimized solutions faster since the layout generation is guided by fitness.

C. Performance Comparisons

Fig. 1 compares these two optimization algorithms for a sample case design. In this case, RAND obtains a better Pareto front solution set than NSGA-II with lower parasitic inductance. However, it is important to note that RAND has no guidance in the search process, which explains why the number of Pareto solutions is lower than that of NSGA-II. Unlike RAND, the current implementation NSGA-II is unaware of the layout hierarchy, which limits its ability to find a better solution than RAND. The entire layout is flattened into a single design string consisting of all variables. The layout organization information is lost during this process, resulting in some parts of the design space never being reached. To address this limitation, a hierarchical optimization framework is proposed in the following section.

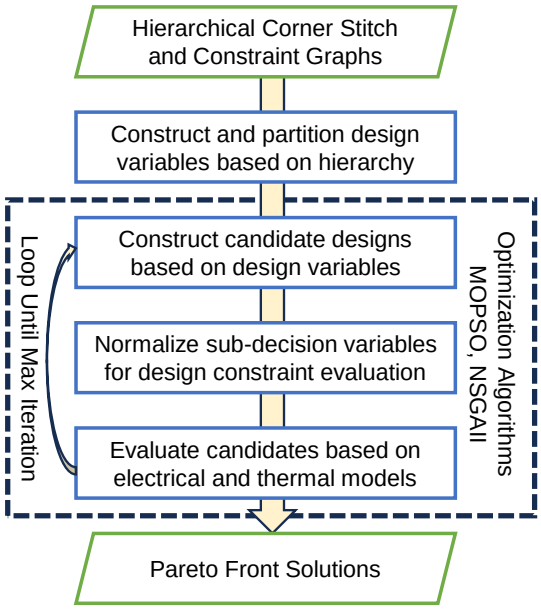

Fig. 2. Proposed Hierarchical Optimization Flow for PowerSynth 2.

III. Hierarchical Optimization Algorithms

A. Hierarchical Genetic Algorithm (New NSGA-II)

To improve old NSGA-II, the design string should contain not only the design variables in the parent node but also consider the child nodes. Thus, the design string follows the hierarchical structure of the layout engine and will consist of some sub-design strings. Then, based on the new structure of the design string, the initial population is generated and according to each sub-design string normalized. After that, each population is evaluated. In each generation, each offspring is normalized and evaluated until the Pareto Front is achieved.

B. Multi-Objective Particle Swarm Optimization (MOPSO)

The latest addition is one of the most promising stochastic search methodologies because of its easy implementation and high convergence speed. MOPSO starts with the swarm population, which is followed by evaluation and density assessment of candidate solutions. Then, the positions of particles are archived as non-dominated solutions in an external repository. In every iteration, the speed of each particle is calculated based on its local best-known position, and global-best positions found by the entire swarm. After that, the new position is updated based on its current position and velocity [13]. Finally, the process follows up with an evaluation of the new particle and an update of the repository content.

The detailed implementation is shown in Algorithm 1. We set these MOPSO parameters: The inertia weight a equals 0.9, and two random numbers b and c in the range of [0, 1], and Mutation probability $1/numberofDV$) is applied on 1/3 population. $C1$ and $C2$ are the random numbers in the range of [1, 2]. These variables depend on the design layout and are divided into two groups, horizontal and vertical, which are then computed independently. Based on the layout

Algorithm 1: MOPSO Workflow

```
Initialize External Repository (ER)
for each particle do
    Initialize the particle's position & velocity randomly
    Evaluate particle
    Update the best personal value and archive it in ER
while Maximum Iteration is not met do
    for each particle do
        Select a leader from the ER
        V = a * V + b * (Pbest - POP) + c * (BestER(h) - POP)
        POP = POP + V
        Perform Mutation with the Swarm
        Evaluate particle
        Update the best personal value and archive it in ER
    update leader in ER
Return ER as a Pareto Front Solutions
```

hierarchy structure, each node has a movable room between its maximum and minimum allowable position. This room is distributed by the RAND algorithm in the upper level, while some of these variables are also mapped into the lower level as needed. Therefore, the dependency of variables across different levels must be considered [10].

This paper proposes a hierarchical optimization framework to solve this issue. This hierarchical optimization flow is summarized in Fig. 2, and its implementation is explained in Algorithm 2 and Algorithm 3. According to the Hierarchical Constraint Graphs, the Decision Variables are grouped into a list of design strings. An example horizontal design string is constructed as follows:

$$\begin{gathered}\{[H_1, H_2, H_3, \ldots, H_n][h_{11}, h_{12}, h_{13}, \ldots, h_{1m}] \\ [h_{21}, h_{22}, h_{23}, \ldots, h_{2m}] \ldots [h_{n1}, h_{n2}, h_{n3}, \ldots, h_{nm}]\} \\ \text{subject to} \sum_{n=1}^{N} H_n = Room \quad \text{and} \sum_{m=1}^{M \leq N} h_{nm} = H_n\end{gathered} \tag{1}$$

where, H_n and h_{nm} indicate the nth upper level and its m_{th} lower level decision variables, respectively. $Room$ is the available room. The vertical ones are processed similarly.

Algorithm 2 depicts a high-level workflow for constructing horizontal and vertical decision variables in a hierarchical structure. The process starts with bottom-up constraint propagation for each tree from leaf to root to reserve enough room for the child node within the parent node. Upon reaching the root node of each sub-tree, the number of flexible edges and the ID number of this node are determined. New design variables are concatenated to form the new design string. This procedure is repeated on each sub-tree from the root to the leaf, traversing all flexible edges while demining their IDs. Eventually, Algorithm 3 is performed based on the generated decision strings to search for better layouts.

Algorithm 2: Construction of Decision Variables (DV)

```
Inputs: HCG, VCG, Layout hierarchy, User constraints
Outputs: HDV, VDV
for each tree from leaf to root do
    Perform bottom-up constraint propagation
Evaluate the root node and determine the ID and numbers of flexible edge weights
Append them to the HDV list
for each sub-tree from root to leaf do
    Determine the ID and Numbers of flexible edge weights
    Append them to the HDV list
Perform Optimization Algorithms
```

Algorithm 3: Optimization Algorithms Workflow

```
Generate initial population for each sub-decision variables randomly
for each population do
    Normalized each sub-decision variables
    Evaluate and find the best solution
while Maximum Iteration is not met do
    Generate new solutions based on the existing ones
    for new solutions do
        Normalized each sub-decision variables
        Evaluate and find the best
    update the best solution
Find out the Pareto Front from the best solutions
```

In Algorithm 3, the optimization algorithms start with an initial population that is randomly generated for each sub-decision variable. The main loop of the algorithm constructs a new population by evolving the existing population. The next step normalizes each sub-decision variable for every population, which is evaluated in terms of inductance and maximum temperature. This process repeats until the maximum number of iterations is satisfied, and consequently, the Pareto Front solution set is generated.

To illustrate the proposed approach, a 3D configuration design has been considered, as shown in Fig. 3. The layout geometry script, consisting of two layers (L1 and L2), is shown in Fig. 3(a). As an example, we only consider L1. The illustration of the horizontal corner stitch (HCS) of L1 is shown in Fig. 3(b). Furthermore, The HCS for each group of L1 (T1, T2, and T3) is shown in Fig. 3(c). The bottom-up propagation process of the design constraints is shown in Fig. 4. The detailed algorithms employed for solving constraints and evaluating locations are based on previous work [9]. For ID1 in Fig. 4, there are eight flexible edges to be considered, namely [P1, P2, P3, S4, P5, P6, S5, P7], which should be appended to the decision variables list. For the T1 group (ID2), both S2 and E2 have been considered in ID1 and, thus, E1 and S1 should be appended to the decision variables list. Similarly, for the T2 group (ID3), S3 and E4

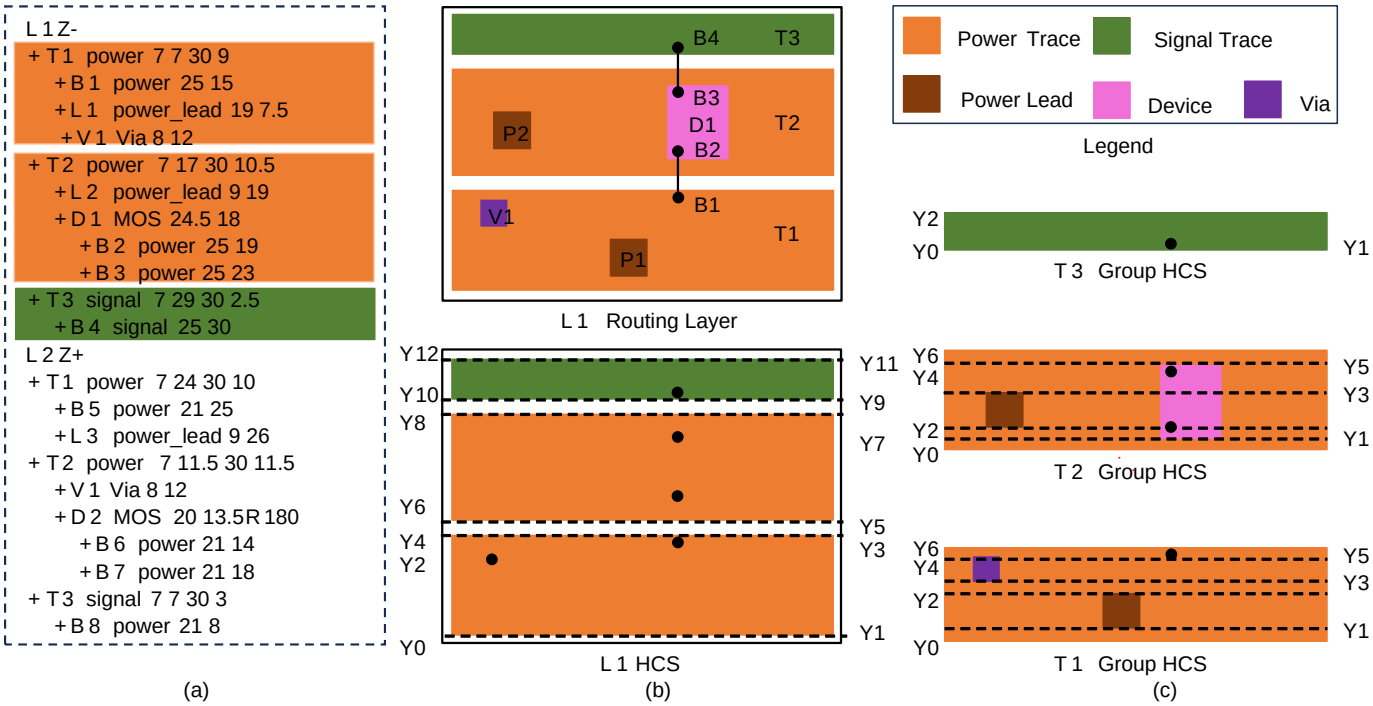

Fig. 3. 3D Layout Example: (a) Geometry script, (b) Horizontal corner Stitch (HCS) of L1, (c) HCS of each group

should be appended, while all edges for the T3 group (ID4) have been considered in ID1. Therefore, the Vertical Decision variables list is constructed as follows:
[[P1, P2, P3, S4, P5, P6, S5, P7], [E1, S1], [E4, S3]].
This same process is applied to Horizontal Decision variables. The first part of the proposed method has been done (Algorithm 1).

An example of the initial population for the vertical is represented as follows:
Pop1 = [[0.3543, 0.7758, 0.1531, 0.0759, 0.4987, 0.3244, 0.4044, 0.3440][0.4378, 0.8389][0.2598, 0.02393]].
Consequently, the normalized of each list for pop1 is:
Normalized Pop1 = [[0.12, 0.26, 0.05, 0.03, 0.17, 0.11, 0.14, 0.12][0.34, 0.66][0.92, 0.08]].
For the evaluation, the edge weights should be calculated. They are calculated by each normalized ID list multiplied by the available room. After performing Algorithm 2 the available room is calculated from the difference between the given floorplan size, and the minimum size. E.g., ID1 has a room of 10. Thus, the edge weights for ID1 are [**1.2**, 2.6, 0.5, 0.3, **1.7**, 1.1, 1.4, 1.2]. The available room for ID2 and ID3 are 1.2 and 1.7, respectively. After the random room distribution, the edge weights for ID2 are [0.408, 0.792], while those for ID3 are [1.564, 0.136].

IV. Performance Indicators

Several performance indicators are introduced to compare the efficiency of the multi-objective algorithms [14]. These indicators are categorized into four groups: convergence, cardinality, distribution, and spread [15]. In this paper, five well-established indicators are considered. These include Modified Generational Distance (GD^+), Modified Inverted Generational Distance (IGD^+), Epsilon (ϵ), Hypervolume (HV), and Error Ratio (ER). Table I indicates the comparison between these indicators. For HV, the greater value is considered better performance, while for other mentioned indicators, the lower the better.

TABLE I
The Comparison between Indicators

Indicators	GD^+	IGD^+	ϵ	HV	ER
Convergence	+	+	+		
Cardinality				+	+
Distribution		+		+	
Spread			+	+	
Preference	Lower	Lower	Lower	Higher	Lower

- **Modified Generational Distance (GD^+):** The indicator measures the average distance between each solution of a given Pareto Front and the closest solution on the reference Pareto Front.
- **Modified Inverted Generational Distance (IGD^+):** This indicator is the same as GD^+, but it measures the distance from the reference Pareto Front to a given Pareto Front.
- **Epsilon (ϵ):** The Epsilon indicator measures the maximum distance between the reference Pareto Front and a given Pareto Front.
- **Hypervolume (HV):** The HV is the most commonly used indicator. It indicates the size of the space covered by Pareto Front.

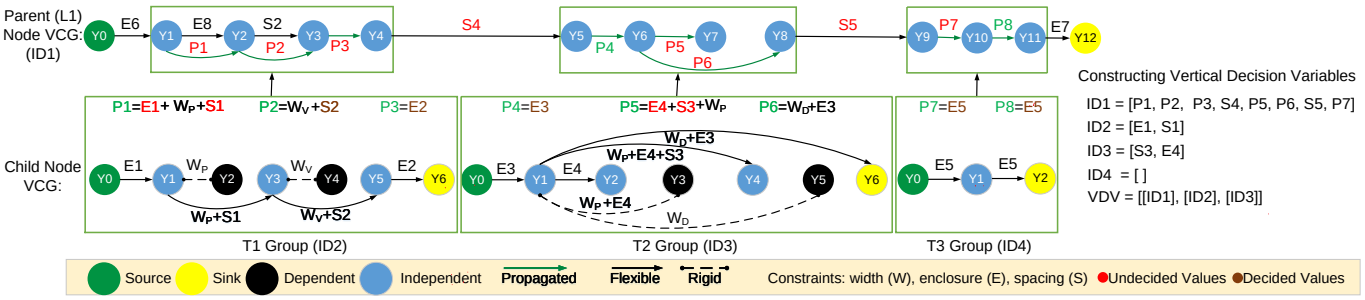

Fig. 4. Bottom-up constraint propagation illustration for HCS shown in Fig. 3

TABLE II
SUMMARY OF TEST CASE DESIGNS

Design	2D/3D	Packaging	Paral. Sw.	Cooling Side	Final Size (mm^2)
Case 1	2D SiC	Wire-bonded	2	Single	40 × 50
Case 2	2D SiC	Wire-bonded	2	Single	40 × 50
Case 3	3D SiC	Metallic post	3	Double	30 × 15
Case 4	3D SiC	Wire-bonded	2	Double	30 × 30

- **Error Ratio (ER):** The ER indicator considers the number of non-dominated solutions of the given Pareto Front, which belongs to the reference Pareto Front.

V. Design Cases and Experimental Results

A. Case Studies

To demonstrate their performance, the proposed methods are applied to several test cases. The results are compared with the current optimization algorithms. In addition, the proposed hierarchy-aware NSGA-II and RAND are compared with MOPSO. The optimization target is to minimize the power loop inductance and maximum temperature by varying the layout floorplan and component placement. It is worth mentioning that the reference Pareto Front is obtained by combining the results from all algorithms.

In this study, we consider 2D and 3D configuration case studies [16]. The summary of case studies is mentioned in Table II. The specifications are summarized in Table II. Case 1 and Case 2 are 2D Half-bridge SiC modules with two switches in parallel. On the other hand, Case 3 and Case 4 are 3D Half-bridge SiC modules that consist of four and two layers, respectively. In Case 3, metallic post-type vias are used for vertical connection, while in Case 4, bonding wires are used. For 2D designs, each algorithm generates 400 layouts. For 3D design, RAND generates 400 layouts, while others generate 200 each. For HV calculation, reference points are considered (18, 400), (26, 402), (2.7, 360), and (5.5, 321) for each case, respectively.

B. Analysis Results and Discussion

The analysis results for case studies are shown in Table III. The hierarchical method performs better than the current optimization structure in all indicators. For Case 1 and Case 2, in both the GD^+ and IGD^+ convergence indicators, MOPSO performs better compared to NSGA-II. MOPSO and NSGA-II obtain similar HV in both cases, while for Case 2, MOPSO obtains better Epsilon. In Case 3, MOPSO bests in all indicators, though HV is almost the same for both algorithms. In Case 4, MOPSO clearly outperforms NSGA-II in all indicators. Overall, with hierarchy considerations, MOPSO is better than NSGA-II, particularly in terms of GD^+ and IGD^+, proving that MOPSO is faster to converge to the Pareto solutions.

TABLE III
COMPARISON OF DIFFERENT ALGORITHMS

Case Studies	Indicators	Current Method		Proposed Method	
		RAND	NSGAII	MOPSO	NSGAII
Case 1	DG^+	1.172	2.149	**0.049**	0.360
	IGD^+	1.165	0.948	**0.042**	**0.056**
	ϵ	1.719	1.160	**0.118**	0.278
	HV	41.72	44.41	**59.37**	**57.38**
	ER	1.000	1.000	**0.407**	0.593
Case 2	DG^+	0.310	1.295	**0.044**	0.197
	IGD^+	0.977	2.007	**0.032**	0.173
	ϵ	3.731	5.214	**0.419**	**0.516**
	HV	42.71	25.83	**47.95**	**45.61**
	ER	0.977	1.000	**0.295**	0.727
Case 3	DG^+	0.129	0.104	**0.023**	0.049
	IGD^+	0.205	0.111	**0.032**	0.043
	ϵ	0.771	0.380	**0.156**	0.221
	HV	5.252	5.743	**6.267**	**6.206**
	ER	0.893	0.893	**0.571**	0.714
Case 4	DG^+	0.354	0.078	**0.000**	0.303
	IGD^+	0.598	0.080	**0.004**	0.284
	ϵ	1.163	0.137	**0.024**	0.626
	HV	3.411	6.161	**6.617**	4.886
	ER	0.607	0.893	**0.571**	1.000

Fig. 5 shows the comparison among different algorithms for Case 1. The proposed method performs better than the current method in almost all aspects. In addition, MOPSO obtained a better Pareto Front solution than NSGA-II. The values of HV and Epsilon over the number of layout generations are shown in Fig. 5(b), and Fig. 5(c), respectively. We find that MOSPO generates optimal results faster than NSGA-II for this case.

To demonstrate the efficiency of the proposed methodology seven distinct floorplan sizes are considered, ranging from 1225 mm to 2225 mm. For each floorplan size, 400 solutions are generated using RAND and MOPSO. The results are shown in Fig. 6, which indicates that MOPSO performs better

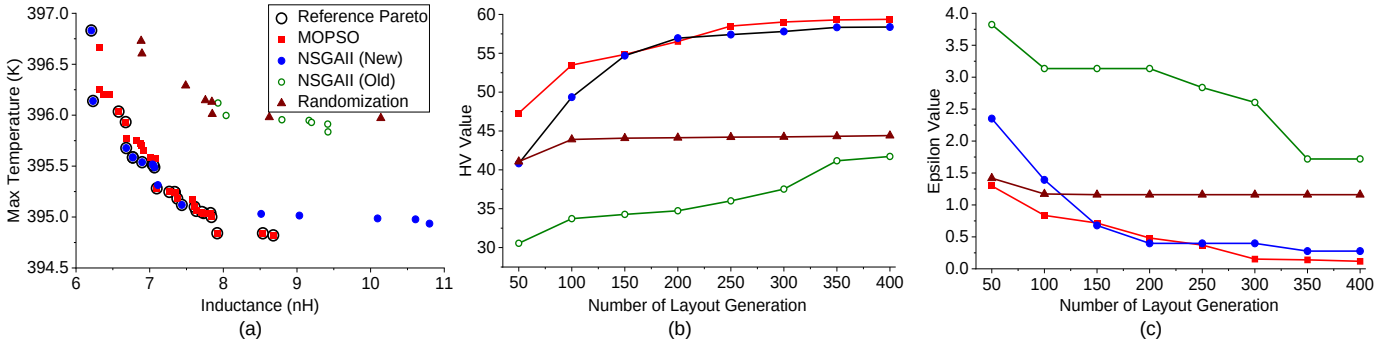

Fig. 5. Comparison of different algorithms in Case 1: (a) Pareto Front, (b) Hypervolume, (c) Epsilon

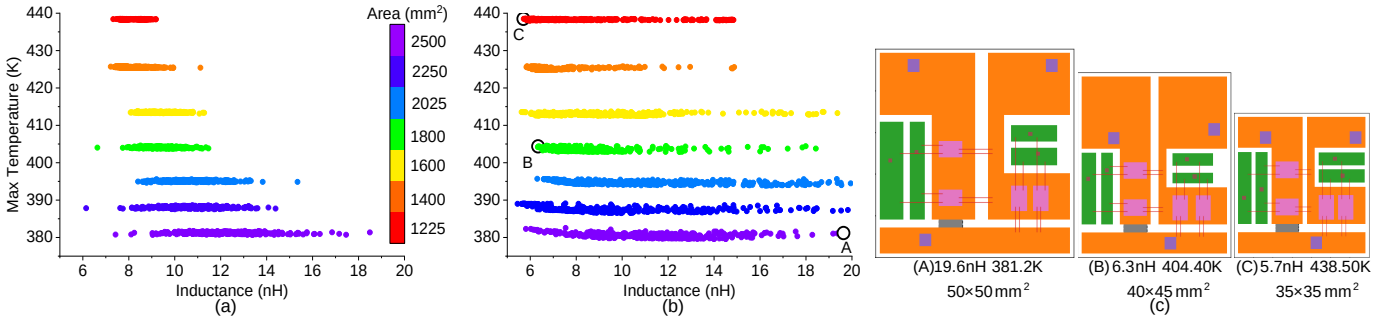

Fig. 6. Case 1 Generated solution space with seven different floorplan sizes: (a) RAND, (b) MOPSO (c) Layouts of the three selected solutions by MOPSO

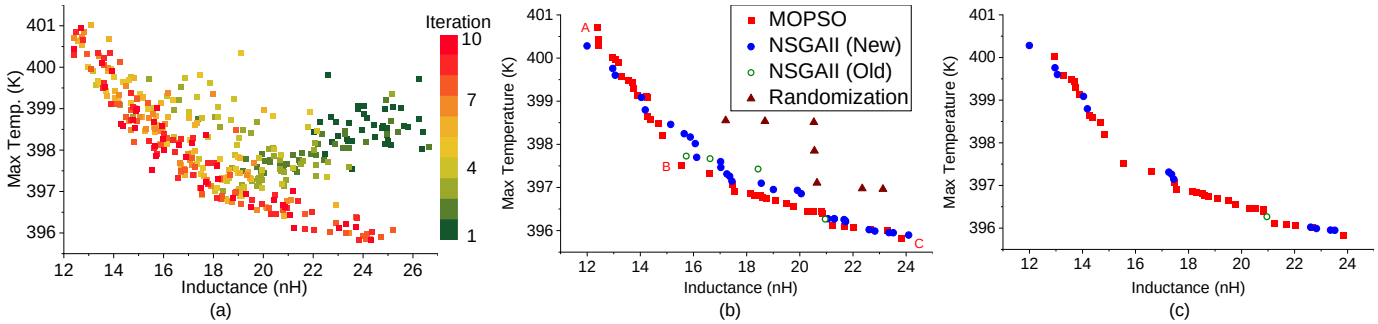

Fig. 7. Comparison of different algorithms in Case 2: (a) MOPSO Solution space over iteration, (b) Pareto Front (c) Global best solutions

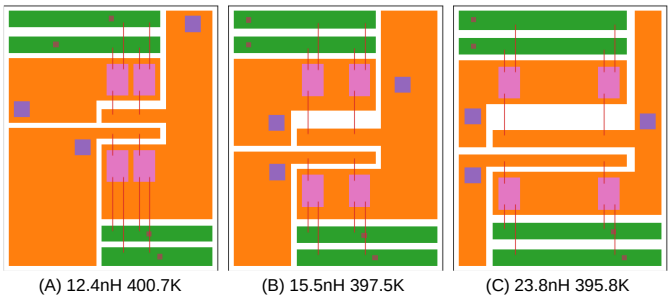

Fig. 8. Case 2: Three solutions generated by MOPSO. All layouts are 40×50 mm^2.

than RAND in all floorplan sizes. The spread and distribution of the solution space generated by MOPSO are wider than that of RAND. From the MOPSO Pareto Front, three solutions are selected and labeled in Fig. 6(b). The corresponding layouts of these solutions are shown in Fig. 6(c). Layout A has the highest inductance value but the lowest temperature as it has the largest floorplan size. Conversely, Layout C has worse thermal results but better electrical performance, which is reasonable as it has the smallest floorplan size. Layout B represents a balanced tradeoff between the two extreme choices.

Similarly, Fig. 7(b) compares different algorithms for Case 2 with three different optimized layouts shown in Fig. 8. From Fig. 7(b), the proposed method performs better than the current method. In addition, MOPSO obtains a comparable Pareto Front solution to NSGA-II. Fig. 7(a) shows the solution space of MOPSO over iterations. The solution space from iteration 1 (initial population) expands with more iterations, and newly created designs concentrate toward the Pareto Front.

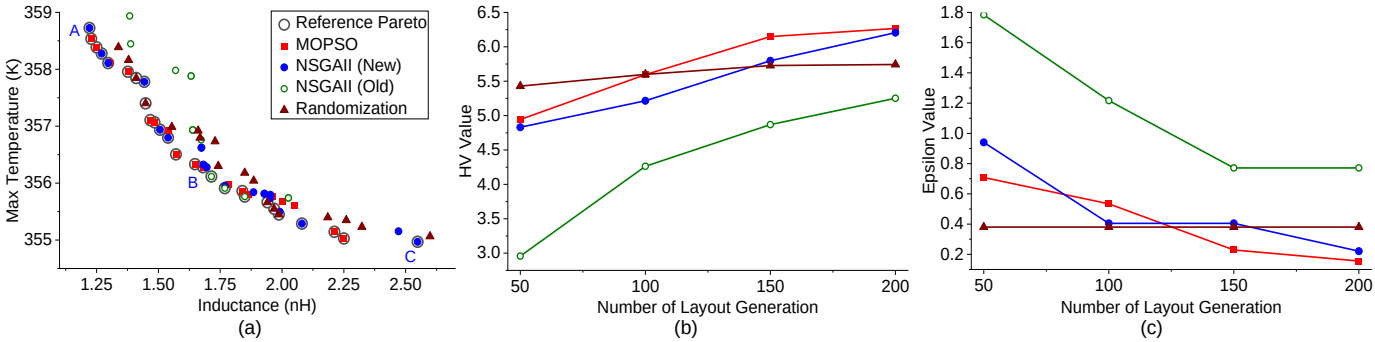

Fig. 9. Comparison of different algorithms in Case 3: (a) Pareto Front, (b) Hypervolume, (c) Epsilon

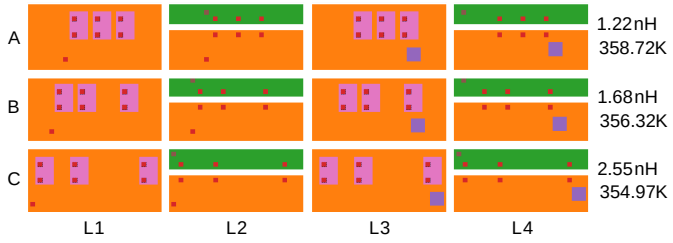

Fig. 10. Case 3: Three solutions generated by NSGAII. All layouts are 30×15 mm^2.

TABLE IV
RUNTIME COMPARISON(UNIT IN MIN).

Case study	RAND	NSGAII (Old)	NSGAII (New)	MOPSO
Case 1	15.5	16.0	16.0	16.0
Case 2	15.4	14.8	15.0	15.0
Case 3	20.0	100	103	103
Case 4	9.0	63.0	64.0	65.0

In addition, observing from the solution space movement over the number of iterations, the proposed algorithms demonstrate a higher convergence speed, as shown in Table III. In addition, Fig. 7(c) gathers the global best solutions obtained from all algorithms. Almost all Pareto Front solutions belong to our proposed method, MOPSO, with hierarchical design strings.

Fig. 9 displays a comparative analysis of different algorithms for Case 3, a 3D module with four layers. From Fig. 9(a), we can deduce that the proposed methodology outperforms the current method. Furthermore, MOPSO obtained a comparable Pareto Front solution to NSGA-II. Figs. 9(b) and 9(c) illustrate the trend of HV and Epsilon across the layout generations. From the NSGA-II solution space, three solutions are selected and labeled in Figure 9(b). The corresponding layouts of these solutions are shown in Fig. 10. Layout A has the lowest inductance value compared to the two other layouts with the highest temperature. On the other hand, Layout C has better thermal performance but worse electrical results. Among them, layout B is considered a balanced Electro-thermal solution.

C. Runtime Comparison

Runtime is measured on a server with dual Intel Xeon Silver 4210 processors. The runtime comparison result for all case studies is shown in Table IV. For the 2D case studies, each algorithm generates 400 layout generations. In Case 1, both NSGA-II and MOPSO algorithms take 16 minutes, while RAND takes 15.5 minutes. For the 3D cases, RAND generates 400 layouts, while both NSGA-II and MOPSO generate 200 each. In Case 3, RAND takes 20 minutes with the support of parallel computing, while the old NSGA-II takes 100 minutes. With the proposed methodology, the runtime for NSGA-II and MOPSO is almost the same, around 103 minutes. Similar results are found in Cases 2 and 4. Based on these, the runtime of the proposed algorithms is comparable to the old NSGA-II, while RAND takes the lead with 5-7 times faster runtime. This is because RAND creates all solutions first, and then evaluates all solutions independently with parallelization. Note that it is possible to accelerate NSGA-II and MOPSO by evaluating all designs in each iteration with parallel computing. This programming improvement will be implemented in a future version.

VI. CONCLUSIONS AND FUTURE WORK

In this work, PowerSynth 2 optimization algorithms have been updated from planar to hierarchical. In addition, a new MOPSO is proposed as a faster alternative to existing NSGA-II and RAND. For performance comparisons, five different indicators and runtime are considered to evaluate different aspects of the optimization algorithms. All algorithms are tested on four different design cases. The results verify that the hierarchical optimizations significantly improve result quality and solution space size. Moreover, MOPSO is comparable to NSGA-II in terms of distribution and spread but achieves a faster convergence speed. These proposed hierarchical algorithms will be open-sourced and released as PowerSynth v2.1 for public testing. Parallel computing will be enabled in a future version to accelerate the runtime of these algorithms further.

ACKNOWLEDGMENT

The authors would like to thank Dr. Zahra Saadatizadeh, David Agogo-Mawuli, and Dr. Tristan Evans for suggestions.

REFERENCES

[1] K. Hermanns, Y. Peng, and H. A. Mantooth, "The Increasing Role of Design Automation in Power Electronics: Gathering What Is Needed," *IEEE Power Electronics Magazine*, vol. 7, no. 1, pp. 46–50, Mar. 2020.

[2] Y. Zhou, Y. Jin, Y. Chen, H. Luo, W. Li, and X. He, "Graph-Model-Based Generative Layout Optimization for Heterogeneous SiC Multichip Power Modules With Reduced and Balanced Parasitic Inductance," *IEEE Transactions on Power Electronics*, vol. 37, no. 8, pp. 9298–9313, 2022.

[3] P. Ning, F. Wang, and K. D. Ngo, "Automatic Layout Design for Power Module," *IEEE Transactions on Power Electronics*, vol. 28, no. 1, pp. 481–487, 2013.

[4] P. Ning, X. Wen, Y. Mei, and T. Fan, "A Fast Universal Power Module Layout Method," in *2015 IEEE Energy Conversion Congress and Exposition (ECCE)*, 2015, pp. 4132–4137.

[5] S. Fukunaga and T. Funaki, "Chip Layout Optimization of SiC Power Modules Based on Multiobjective Electro-Thermal Design Strategy," *IEEJ Journal of Industry Applications*, vol. 11, no. 1, pp. 157–162, 2022.

[6] N. Rajagopal, E. Gurpinar, B. Ozpineci, and C. DiMarino, "Electro-Thermal Optimization of Common-Mode Screen for Organic Substrate-Based SiC Power Module," in *2022 IEEE Energy Conversion Congress and Exposition (ECCE)*, 2022, pp. 1–8.

[7] Y. Chen, D. Zhao, F. Liu, J. Gao, and H. Zhu, "Thermal Layout Optimization for 3D Stacked Multichip Modules," *Microelectronics Journal*, vol. 139, p. 105882, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0026269223001957

[8] T. Evans, Q. Le, S. Mukherjee, I. Al Razi, T. Vrotsos, Y. Peng, and H. A. Mantooth, "Powersynth: A Power Module Layout Generation Tool," *IEEE Transactions on Power Electronics*, vol. 34, no. 6, pp. 5063–5078, Jun. 2019, highlighted Paper.

[9] I. Al Razi, Q. Le, T. Evans, S. Mukherjee, H. A. Mantooth, and Y. Peng, "PowerSynth Design Automation Flow for Hierarchical and Heterogeneous 2.5D Multi-Chip Power Modules," *IEEE Transactions on Power Electronics*, vol. 36, no. 8, pp. 8919–8933, 2021.

[10] I. Al Razi, Q. Le, T. Evans, H. A. Mantooth, and Y. Peng, "PowerSynth 2: Physical Design Automation for High-Density 3D Multi-Chip Power Modules," *IEEE Transactions on Power Electronics*, vol. 38, no. 4, pp. 4698–4713, 2023.

[11] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A Fast and Eitist Multiobjective Genetic Algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[12] I. Al Razi, Q. Le, H. A. Mantooth, and Y. Peng, "Hierarchical Layout Synthesis and Optimization Framework for High-Density Power Module Design Automation," in *International Conference on Computer-Aided Design*, Nov. 2021, pp. 1–8.

[13] C. Coello, G. Pulido, and M. Lechuga, "Handling Multiple Objectives with Particle Swarm Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 256–279, 2004.

[14] C. Audet, J. Bigeon, D. Cartier, S. Le Digabel, and L. Salomon, "Performance Indicators in Multiobjective Optimization," *European Journal of Operational Research*, vol. 292, no. 2, pp. 397–422, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221720309620

[15] M. Li and X. Yao, "Quality Evaluation of Solution Sets in Multiobjective Optimisation: A Survey," *ACM Computing Surveys (CSUR)*, vol. 52, no. 2, pp. 1–38, 2019.

[16] "PowerSynth." [Online]. Available: https://github.com/e3da/