Application of Expensive Multi-objective Optimization Algorithms on Chip Packaging Design

Yunzhong Xiao
SDIM, SUSTech
soler_xyz@berkeley.edu

Lingfeng Li
SDIM, SUSTech
11912405@mail.sustech.edu.cn

Jingyi Lai
SDIM, SUSTech
11912038@sustech.edu.cn

***Abstract*—We proposed a new method to solve the multi-objective problems in chip packaging design by applying machine learning algorithms. Seven algorithms were implemented and analysed to optimize warpage and max von Mises stress influenced by five parameters in our finite element model. We built a finite element model in Abaqus and construct its interfaces with python and matlab code. The black-box function is tested on classical multi-objective algorithms such as MOEA/D-EGO, NSGA2. In order to find a superior algorithm, we also implemented state of the art surrogate-assisted expensive multi-objective optimization algorithms including θ-DEA-DP, MOEA/D-ASS and qEHVI. An empirical study on the better performed algorithms is accomplished in our experiment. With further comparison and analysis, we demonstrate the superiority of our method of applying machine learning technique to tackle multi-objective optimization problems in chip package design industry.**

***Index Terms*—Surrogate-assisted evolutionary computation, expensive multi-objective optimization, machine learning, Bayesian optimization, chip packaging, finite element modeling.**

I. Introduction

In the current rush, semiconductor design and manufacturing is one of the most promising areas in the race to apply machine learning to huge markets and applications. With the increasing complexity and overall scale of chip design in the post-Moore's-law era, chip packaging has become an issue. Scale and complexity create more uncertainty, redesigns, missed schedules, and so on[1]. More design variables should be considered and multiple objectives would be optimized simultanously. In such situation, using machine learning to assist chip packaging design can help to reduce the waste of time and manpower. And it is true that applying ML to offload the work is becoming a trend in chip field[2].

A. Literature review

Actually, international scholars have researched various difficulties with regarded to chip packaging assisted by machine learning and made varying degrees of progress. The M.Yasunaga team improved signal integrity by using STL to compensate for distortion of the waveform[3]. H.Zaidan's team apply machine learning algorithm to optimize the structure such as hot slot[4]. H. Manimegalai's team used reinforcement learning algorithm as the layout algorithm, combined with support vector machine (SVM), to quickly and accurately find the layout method that meets the requirements of temperature distribution[5]. Li Yongsheng studied dynamic thermal management method of 3D integrated packaging based on fluid refrigeration technology and machine learning algorithm[6]. Ji Hang established DNN, CNN, SVR, KNN and linear regression prediction model to predict the far-field radiation of 0.2 GHz to 20 GHz output by input the selected seven key package structure parameters[7]. In Dai Weijing's study, an easy-to-use and effective optimization workflow based on Bayesian optimization and Gaussian process is implemented and further improved by applying simulated annealed algorithm[8].

Our exploration is targeted on warpage and stress. For now, a few researches have paid attention to the combination of machine learning and the above two concerns. In 2020, a study of this kind demonstrated a method that based on neural network to learn relationship between the different panel level package geometry sizes with warpage values and try to optimize and evaluate the warpage[9]. In the same year, another study developed a model involving SVM and Random Forest to solve the problem of stress in chip packaging[10]. However, these two problem are barely studied simultanously.

B. Limitations of current approaches

Nevertheless, the existing research still has some shortcomings. Thermal models for TSV metal cores are of lacking. So do thermal problems with electromagnetic pulse(EMP). Crosstalk shielding and common-mode noise shielding are not abundant enough and there is no effective design method. There are many testing steps and complex testing indexes, which need to consume a lot of time and cost[6]. In Dai's study, the decay factor for the annealing schedule to adjust has been chosen rather arbitrarily in current application, and not tested for other values. Thus, further insight about effects and guideline of changing this factor will be helpful to make the work flow more robust[8]. Nevertheless, the studies of stress inside the chip, detecting holes and other problems are also lacking. Most of the scholars only focused on one single objective, the research for multi-objective optimization(MOO) in chip packaging field is absolutely ponderable.

C. Related Expensive MOO Works

Many scholars focus on expensive MOO algorithms. The most famous and earliest algorithm is NSGA2[11] proposed

almost 20 years ago. Then Qingfu Zhang and Hui Li proposed MOEA/D in 2007[12], which is the original version of a lot of the algorithms that followed. To improve the performance, scholars implemented surrogate models into the evolution algorithms, like Parego[13] and MOEA/D-EGO[12].

On this basis, others try to improve the algorithms more by changing or proposing new surrogate models, acquisition functions, and other details. However, a multi-objective problem may not just have one Pareto front. In practical applications, some multi-objective problems have different Pareto sets with the same objective values and these problems are defined as multimodal multi-objective optimization problems[14]. Liang focuses on the distribution of decision space when many people just consider the objective space.

The two key operations in an intelligent search strategy are exploration and exploitation. Among them, exploration ability affects convergence speed and exploitation ability affects the diversity of the solution set. Xilu Wang proposed an adaptive Bayesian approach, whose main idea is to tune the hyperparameter in the acquisition function according to the search dynamics to determine which candidate solutions are to be evaluated using the expensive real objective functions. Prof. Zhenkun Wang integrated the ASS strategy, the CoMOGP model, and the ALCB into a novel algorithm called MOEA/D-ASS[15]. The ASS strategy helps to select subproblems preferably. The ALCB method modifies the γ in the original method, LCB, and balances the exploration and exploitation better. Yuan Yuan considered the dominance relationship and proposed θ dominance in 2016. Then he used deep FNN as the surrogate model and combined Pareto dominance and θ dominance to get a better algorithm called θ-DEA-DP[16]. He also used directional retrieval to keep the diversity. The deep FNN may cause the overfitting in the training process, so yuan only used 3 layers in his work. To solve MOO in high dimension, Jianqing Lin proposed a surrogate assisted evolutionary algorithm (SAEA) with an adaptive dropout mechanism in 2021[17], which took advantage of the statistical differences between different solution sets in the decision space to guide the selection of some crucial decision variables. Genghui Li proposed a three-level radial basis function (TLRBF)-assisted optimization algorithm for expensive optimization in 2021[18]. Although the algorithm focus on a single objective, the three search procedures at each iteration are still useful for the problem of expensive multi-objective optimization. Rojas surveyed the most relevant kriging-based infill algorithms in 2020[19]. These algorithms perform a sequential search of so-called infill points, used to update the kriging metamodel at each iteration, but none of them so far incorporates an effective way to deal with heterogeneous noise.

Some platforms integrated MOO algorithms and created a graphical interface to help other scholars develop their algorithms and do experiments easier. Y Tian developed Platemo with Matlab in 2017[20] and AutoOED in 2021[21]. Balandat improved Botorch in 2020[22] as well.

As for application, MOO algorithms started to be used in real-world problems for many years, not just about chip package. Zeng proposed a multi-objective optimization design method for power module packaging and solved it by using NSGA-II[23]. Prinz presents a novel trust-region method for the optimization of multiple expensive functions He applied this method to a bi-objective optimization problem in fluid mechanics, the optimal mixing of particles in a flow in a closed container[24]. Real-world problems may have unexpected questions, such as the ways of normalization, poor distribution, and strange Pareto front.

Existing methods have improved in many areas. To some extent, they have acceptable performance in practical multi-objective optimization problems. However, an algorithm that does well in one area cannot mean it does well in another. In practical engineering problems, how to weigh which algorithm is the best, and how to combine the advantages of existing methods to get better answers are things we still need to explore.

D. Contributions

In this paper, we aim to apply machine learning algorithms to implement the optimization of a multi-objectives optimization problem in chip packaging design. Based on Dai's study, we design a chip model with 2 objectives: maximum von Mises stress and the extent of warpage, and 5 design variables including coefficient of thermal expansion of Epoxy molding compound (EMC), the thickness of EMC, die, die-attach layer and substrate[8]. 7 algorithms are implemented and compared to find the best set of parameters.

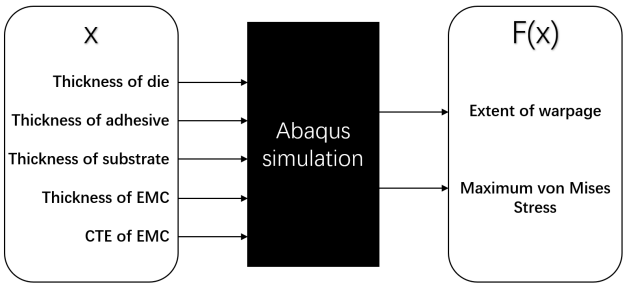


Fig. 1. Illustration of the blackbox function

II. Preliminaries and Background

This section presents the basic concepts that are related to our work.

A. Multi-Objective Optimization

Multi-objective optimization is to simultaneously optimize two or more objective functions without explicitly balancing them, in others words, there is no solution to optimize every objective function at the same time.

These objective functions are often contradictory – optimizing one of them "harms" the others – so it is impossible to find a solution that is optimal for all of them. Therefore, the solution of multi-objective optimization is a group of solutions,

in which each solution has its unique advantages over other solutions. This advantage can be measured by a number of criteria, one commonly used method being Pareto optimality.

As for expensive MOO, it costs more to do simulation so that it is supposed to get better result with less evaluations. We can define a MOO problem as:

$$\begin{aligned} \text{minimize} \quad & \mathrm{f}(\mathrm{x}) = (f_1(\mathrm{x}), \ldots, f_m(\mathrm{x}))^\top \\ \text{subject to} \quad & \mathrm{x} \in \Omega \subset \mathbb{R}^d \end{aligned}$$

B. Evolutionary Algorithms

Evolutionary algorithms are not a specific algorithm, but a "cluster of algorithms". The inspiration of evolutionary algorithms refers to the evolutionary operations of organisms in nature, which generally include basic operations such as gene coding, population initialization, crossover and mutation operators.

Evolutionary algorithm has natural advantages in solving multi-objective problems. Commonly used evolutionary multi-objective optimization algorithms include NSGA-II based on congestion distance measurement proposed by Deb[25] and MOEA/D algorithm based on decomposition thought proposed by Qingfu Zhang[12].

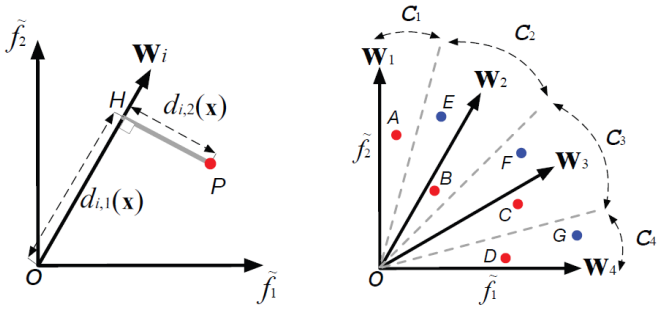
C. Dominance

There are two kinds of dominant relationships. The first one is classical Pareto Dominance while the second one is θ Dominance proposed by Yuan[16].

Pareto Dominance is a kind of dominance relationship. Given 2 solutions A and B with 2 dimensions, if a1 is better than b1 when a2 is not worse than b2, we call A Pareto dominances B.

θ Dominance[16] is similar to Pareto Dominance to some degree. But it keeps diversity better. As the figure shown, the objective space is divided into N clusters by N+1 weight vectors. As for any solution in this space, we choose the nearest weight vector to compute the penalty boundary intersection (PBI) function which is defined as

$$\mathcal{F}_i(\mathrm{x}) = d_{i,1}(\mathrm{x}) + \theta d_{i,2}(\mathrm{x})$$



(a) Distances $d_{i,1}$ and $d_{i,2}$ (b) Objective space division

Fig. 2. θ Dominance

D. Surrogate Model

Surrogate model is a common optimization method used in expensive black-box problem. The calculation result of the surrogate model is quite close to the original model, but the calculation amount is smaller. The surrogate model is built by calculating the responses (outputs) of the original model at a carefully selected number of points (inputs). While evolutionary algorithms(EAs) such as NSGA-II are popular choices for solving MOO problems, they still suffer from high sample complexity, rendering them infeasible for optimizing expensive-to-evaluate black-box functions[26]. Surrogate model assisted algorithms such as Bayesian optimization, however, provides a much more sample-efficient alternative.

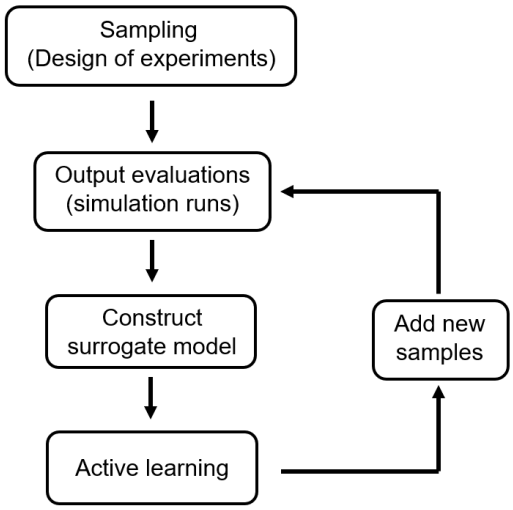


Fig. 3. Surrogate modeling workflow

E. Bayesian Optimization(BO)

Bayesian Optimization is a sample-efficient optimization method that utilizes a probabilistic surrogate model to make principles decisions to balance exploration and exploitation.[27] The typical surrogate is Gaussian Process(GP). Evaluating the surrogate is dramatically cheap and fast comparing to evaluate the true black-box function, therefore the numerical optimization can assist to find the maximizer of the acquisition function to evaluate next on the true black-box function.[28] BO sequentially selects new points to evaluate and updates the model to incorporate the new observations.

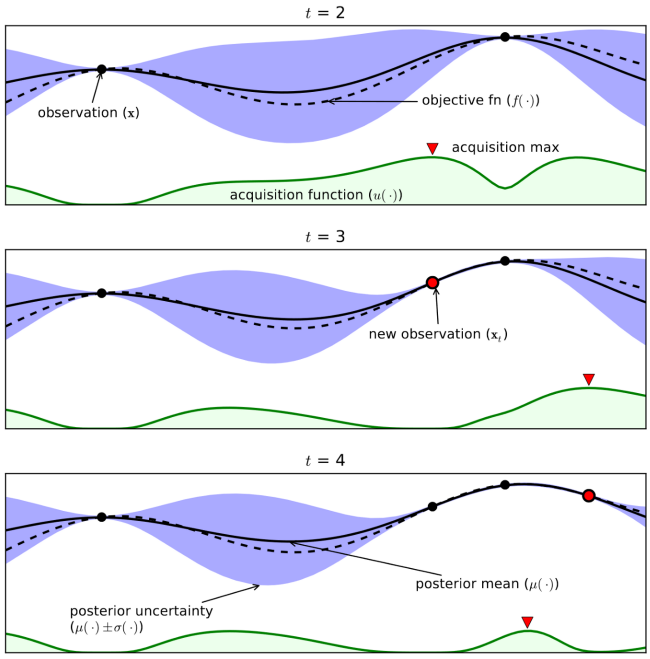


Fig. 4. Curve fitting gets improved while sample data increases

F. Acquisition function

The acquisition function is constructed according to a posterior probability distribution, and the next most "potential" evaluation point is selected by maximizing the acquisition function. At the same time, an effective collection function can ensure that the selected sequence of evaluation points minimizes the total loss.

G. Latin Hypercube Sampling

Latin hypercube sampling (LHS) is a statistical method for generating a pseudo-random sample of parameter values from a multidimensional distribution. When sampling a function of N variables, the range of each variable is divided into M equally probable intervals. M sample points are then placed to satisfy the Latin hypercube requirements, this forces the number of divisions, M, to be equal for each variable.

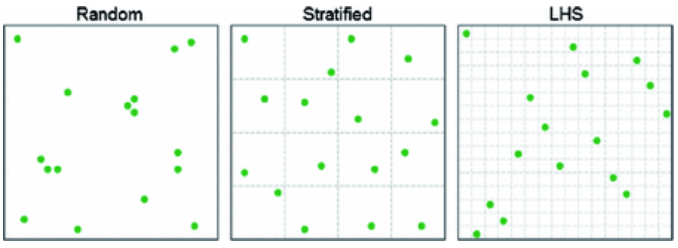


Fig. 5. Different Sampling Methods

H. Hypervolume

Given a reference point $r \in \mathbb{R}^M$, the hypervolume indicator (HV) of a finite approximate Pareto set $\mathcal{P}$ is the M-dimensional Lebesgue measure λ_M of the space dominated by $\mathcal{P}$ and bounded from below by

$$\boldsymbol{r} : \mathrm{HV}(\mathcal{P}, \boldsymbol{r}) = \lambda_M \left(\bigcup_{i=1}^{|\mathcal{P}|} [\boldsymbol{r}, \boldsymbol{y}_i] \right)$$

In which $[\boldsymbol{r}, \boldsymbol{y}_i]$ denotes the hyper-rectangle bounded by vertices $\boldsymbol{r}$ and $\boldsymbol{y}_i$.

I. Hypervolume Improvement

Given a Pareto set $\mathcal{P}$ and reference point $\boldsymbol{r}$, the hypervolume improvement (HVI) of a set of points $\mathcal{Y}$ is:

$$\mathrm{HVI}(\mathcal{Y}, \mathcal{P}, \boldsymbol{r}) = \mathrm{HV}(\mathcal{P} \cup \mathcal{Y}, \boldsymbol{r}) - \mathrm{HV}(\mathcal{P}, \boldsymbol{r})$$

J. Chip Warpage and von Mises Stress

Stress inside the chip and warpage are two common problems in chip packaging. Warpage is a big issue in packaging industry. Large warpage in chip package can cause serious consequence, such as difficulty with post encapsulation processes like reflow, which may stop the manufacturing process from continuing[29]. As for stress, because of the mismatch in the thermal expansion coefficient between the chip and materials used in the modules, large local deformation of the chip caused by thermal stress increases sharply. Such a localized high stress gives rise to fractures such as delamination or cracking of materials composing the modules[30].

K. Modeling

The tested packaging in Abaqus is based on Dai's model6. Package is simplified to neatly stacked cubic model.

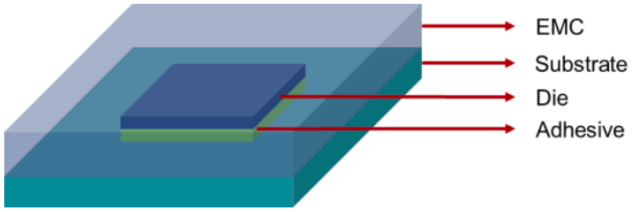


Fig. 6. Single-die package model,Source: Dai Weijing, 2020

The five design variables used in the range are shown in table I:

TABLE I
DESIGN SPACE

Parameters	min	max
Thickness of die (mm)	0.2	0.32
Thickness of adhesive (mm)	0.02	0.04
Thickness of substrate (mm)	0.2	0.3
Thickness of EMC (mm)	0.55	0.95
CTE of EMC (ppm/°C)	8	12

To make the output more recognized, we have the 4 parameters of thickness multiplied a coefficient of 1000. After simulation, we normalize the output. The model is able to print out two values of targets – the maximum von Mises stress and the displacement of warpage.

III. Algorithms Implemented

A. θ-DEA-DP

θ-DEA-DP is an expensive multi-Objective evolutionary optimization algorithm proposed by Yuan in 2020[16].

Such an algorithm uses deep feedforward neural networks (FNNs) as the surrogate model. Both Pareto dominance and dominance are used to select a representative solution for each weight vector. The two-stage Preselection strategy is something like an acquisition function. The first stage randomly takes a representative solution. If the solution is null, it will explore the -non-dominated solution in this cluster to enhance the diversity. If the solution is not null, it will divide every solution into four sets according to the dominance relationship between the selected solution and others predicted by neural networks. The second stage is to select the best one in the previous stage with the expected dominance number (EDN) which is defined as:

$$e\left(\mathbf{z}_i\right)=\sum_{j\neq i} I\left(\hat{t}\left(\mathbf{z}_i,\mathbf{z}_j\right)=1\right)\hat{p}\left(\mathbf{z}_i,\mathbf{z}_j\right)$$

EDN in terms of θ-dominance is similar. Finally, it adds two EDNs and chooses the biggest one to evaluate.

Algorithm 1 Framework of the Proposed θ-DEA-DP
Source: Yuan Yuan, 2021

$\{w_1, w_2, ..., w_N\} \leftarrow$ InitializeWeightVectors(m)
$\mathbb{P} \leftarrow$ LatinHypercube(n)
evals $\leftarrow 0$
$\mathbb{A} \leftarrow \emptyset$
for $x \in \mathbb{P}$ **do**
  Evaluate(x)
  evals $\leftarrow$ *evals* $+ 1$
  $\mathbb{A} \leftarrow \mathbb{A} \cup x$
end for
p-net $\leftarrow$ Initiate-Pareto-Net($\mathbb{A}$)
θ-net $\leftarrow$ Initiate-θ-Net($\mathbb{A}$)
$\{x_1^*, x_2^*, ..., x_N^*\} \leftarrow$ Get-θ-Reps($\mathbb{A}$)
$\{y_1^*, y_2^*, ..., y_N^*\} \leftarrow$ Get-Pareto-Reps($\mathbb{A}$)
$\mathbb{P} \leftarrow$ TruncatePopulation($\mathbb{P}$,N)
while *evals* $<$ MaxEval **do**
  $j \leftarrow$ ChooseTargetClusterIndex(N)
  $\mathbb{Q} \leftarrow$ GenerateOffsprings(P, N)
  $z^{\leftarrow}$ TwoStagePreSelection($\mathbb{Q}, x_j^*, y_j^*$,p-net,θ-net)
  Evaluate(z^*)
  evals $\leftarrow$ *evals* $+ 1$
  $\mathbb{A} \leftarrow \mathbb{A} \cup z^*$
  Update-Pareto-Net(p-net,$\mathbb{A}$)
  Update-θ-Net(θ-net, $\mathbb{A}$)
  Update-θ-Reps($\{x_1^*, x_2^*, ..., x_N^*\}, \mathbb{A}$)
  Update-Pareto-Reps($\{y_1^*, y_2^*, ..., y_N^*\}, \mathbb{A}$)
  $\mathbb{P} \leftarrow$ TruncatePopulation($\mathbb{P} \cup z^{,N}$)
end while

B. qEHVI

qEHVI is a one-step Bayes-optimal policy for maximizing the hypervolume dominated by the Pareto frontier[31]. It is based on expected hypervolume improvement (EHVI), which scales to highly parallel evaluations of noisy objectives. The approach is made feasible by a general-purpose, differentiable, cached box decomposition (CBD) implementation that speeds up critical computations needed to account for uncertainty introduced by noisy observations and generate new candidate points for highly parallel batch or asynchronous evaluation. The CBD-based approach solves the fundamental problem of scaling parallel EHVI-based methods to large batch sizes, reducing time and space complexity from exponential to polynomial.

C. qParEGO

qParEGO[32] is another batch variant of ParEGO that uses compositional Monte Carlo objectives and sequential greedy candidate selection. ParEGO algorithm aggregates multiple objective values of a solution into a single function value via a parameterized weight vector. Next, a solution is selected for evaluation by maximizing the expected improvement (EI) criterion with respect to the current aggregation function. By choosing a different weight vector in each iteration ParEGO is expected to implicitly maintain the diversity of evaluated solutions.

D. MOEA/D-EGO

MOEA/D-EGO[33] extends ParEGO to the batch setting using multiple random scalarizations while considering all aggregation functions rather than a single one in each iteration and maximizes their corresponding EI values simultaneously using MOEA/D-DE[34] in order to generate several points for function evaluation.

E. MOEA/D-ASS

Wang proposed a novel algorithm, MOEA/D-ASS, that integrated the ASS strategy, the CoMOGP model, and the ALCB in 2021[15]. The ASS strategy with the framework is shown in III-E.

1) CoMOGP: The CoMOGP model is a kind of improved GP model. With S subproblems, we can build a joint MOGP model instead of T GP models independently. In this case, we can get a better result if these subproblems are strongly similar. Nevertheless, the result will be worse if they have few things in common. The CoMOGP does well in such conditions. By transferring messages across subproblems and catching the individual feature of each subproblem, it can get better results no matter the similarity is strong or weak. The covariance function between two subproblems can be written like that:

$$k_{tt'}\left(\mathrm{x},\mathrm{x}'\right)=\begin{cases}\sum_{q=1}^{Q} a_{t,q}^2 k_q\left(\mathrm{x},\mathrm{x}'\right)+k_t\left(\mathrm{x},\mathrm{x}'\right), & t=t',\\ \sum_{q=1}^{Q} a_{t,q}a_{t',q}k_q\left(\mathrm{x},\mathrm{x}'\right), & t\neq t',\end{cases}$$

the prediction mean and variance at a new point x is:

$$\hat{\mathrm{g}}\left(\mathrm{x}^*\right)=K_{*n}K_y^{-1}\mathrm{y},$$
$$\Sigma_*\left(\mathrm{x}^*\right)=K_{**}-K_{*n}K_y^{-1}K_{n*},$$

Algorithm 2 MOEA/D-ASS
Source:Zhenkun Wang, 2021

Input: MOP; n_I : the initial population size; $FE_{\max}$: the maximal number of function evaluations; N : the subproblem size; T : the number of subproblems selected in each iteration, n_T : the size of the training data assigned to each subproblem.

Output: NP:the set of non-dominated solutions in EP;

1: **Step 1 Initialization:** Generate n_I initial solutions for function evaluation $EP = \left\{\left(\mathrm{x}^i, \mathrm{f}^i\right)\right\}_{i=1}^{n_I}$ and set $FE = n_I$; construct N subproblems $G = \{g_i\}_{i=1}^{N}$.
2: **while** $FE < FE_{max}$ **do**
3: **Step 2 Subproblem selection:** Use the ASS strategy to select a set of T subproblems G_S from G.
4: **Step 3 Training data allocation:** For each subproblem in G_S, assign n_T evaluated solutions as its training data.
5: **Step 4 Model training:** Train a CoMOGP model for the subproblems in G_S.
6: **Step 5 Model assisted optimization:** Optimize the subproblems in G_S by evaluating solutions via the trained CoMOGP model and an acquisition function, to obtain T candidate solutions $\left\{\mathrm{x}^i\right\}_{i=1}^{T}$.
7: **Step 6 Function evaluation:** Evaluate $\left\{\mathrm{x}^i\right\}_{i=1}^{T}$ with the original objective functions, and add them to EP.
8: **Step 7 Update:** Let $FE = FE + T$; update z^* and ST for each subproblem in G.
9: **end while**

2) ALCB: The traditional LCB acquisition function is defined as:

$$u(\mathrm{x}) = \hat{g}_i(\mathrm{x}) - \gamma\hat{\sigma}_i(\mathrm{x})$$

In MOEA/D-ASS, promoted acquisition function, ALCB, modify γ as following format to

$$\begin{cases} \gamma = \gamma_1 + \gamma_2 \\ \gamma_1 = 10\exp\left(-0.02|NP|^2\right) \\ \gamma_2 = \frac{0.1}{1+\exp\left(100\left(\hat{g}_i(\mathrm{x}) - g_i^*\right)\right)} \end{cases}$$

F. NSGA-II

NSGA-II is a fast non-dominated multi-objective optimization algorithm with elite reservation strategy. It is based on Pareto optimal solution.

The basic idea of NSGA-II is as follows: N initial populations are randomly generated. After non-dominated sorting, the first generation of offspring population is obtained by selection, crossover and mutation of genetic algorithm. Starting from the second generation, the crowding degree of each non-dominant layer is calculated by combining parent population and offspring population. Suitable individuals are selected according to non-dominant kinship and crowding degree to form a new parent population. A new offspring population is generated through the basic operation of genetic algorithm, and so on in a similar fashion, until the ending condition of the program is satisfied.

IV. Experiment and Analysis

A. Experiment Design

The chosen 7 algorithms are tested under the same conditions, which are:

1) The number of independent runs is 7.
2) The initial sampling method is Latin Hypercube Sampling while the number of sampling is 54.
3) Number of evaluations is strictly constrained to 120.
4) The objectives are normalized to 0 to 1 after the preliminary experiment, and the original ranges of warpage and von Mises stress are 0 to 0.025mm, 130 to 230N respectively.

All of the hyperparameters for each algorithm are set according to their recommended values.

B. Performance Indicator

There are so many performance indicators to judge a result good or not. We choose Hypervolume(HV) here since we don't know the Pareto Front before the experiment and the number of objectives is two so the computational cost of HV is acceptable. We are supposed to choose a reference point R, which is [1,1] according to the above normalization.

C. Experiment Results

The Pareto frontiers and the HV values of the 7 algorithms are shown in Figure.7. The following table II displays the comparison of the best, the median and the mean HV values of the algorithms.

TABLE II
Comparison between different algorithms

Algorithms	Best	Median	Mean
qEHVI	0.7602	0.7563	0.7521
qParEGO	0.7627	0.7532	0.7541
Random Scalarization	0.7583	0.7492	0.7499
MOEA/D-ASS	0.7577	0.7453	0.7428
θ-DEA-DP	0.7596	0.7361	0.7354
MOEA/D-EGO	0.7012	0.6890	0.6874
NSGA-II	0.6876	0.6555	0.6538

D. Observation and Analysis

Table II shows the best, median and mean value of Hypervolume in 7 algorithms, and the above pictures show the sampling points of each algorithm, the non-dominated points are colored orange. We can derive the following observations from our experiments:

1) Surrogate-assisted algorithms are dramatically outperformed non-surrogate-assisted algorithms. We tested NSGA-II and NSGA-III in our problem and abandoned NSGA-III since they have similar performance, both 15 percents worse than the top3 surrogate-assisted algorithms.

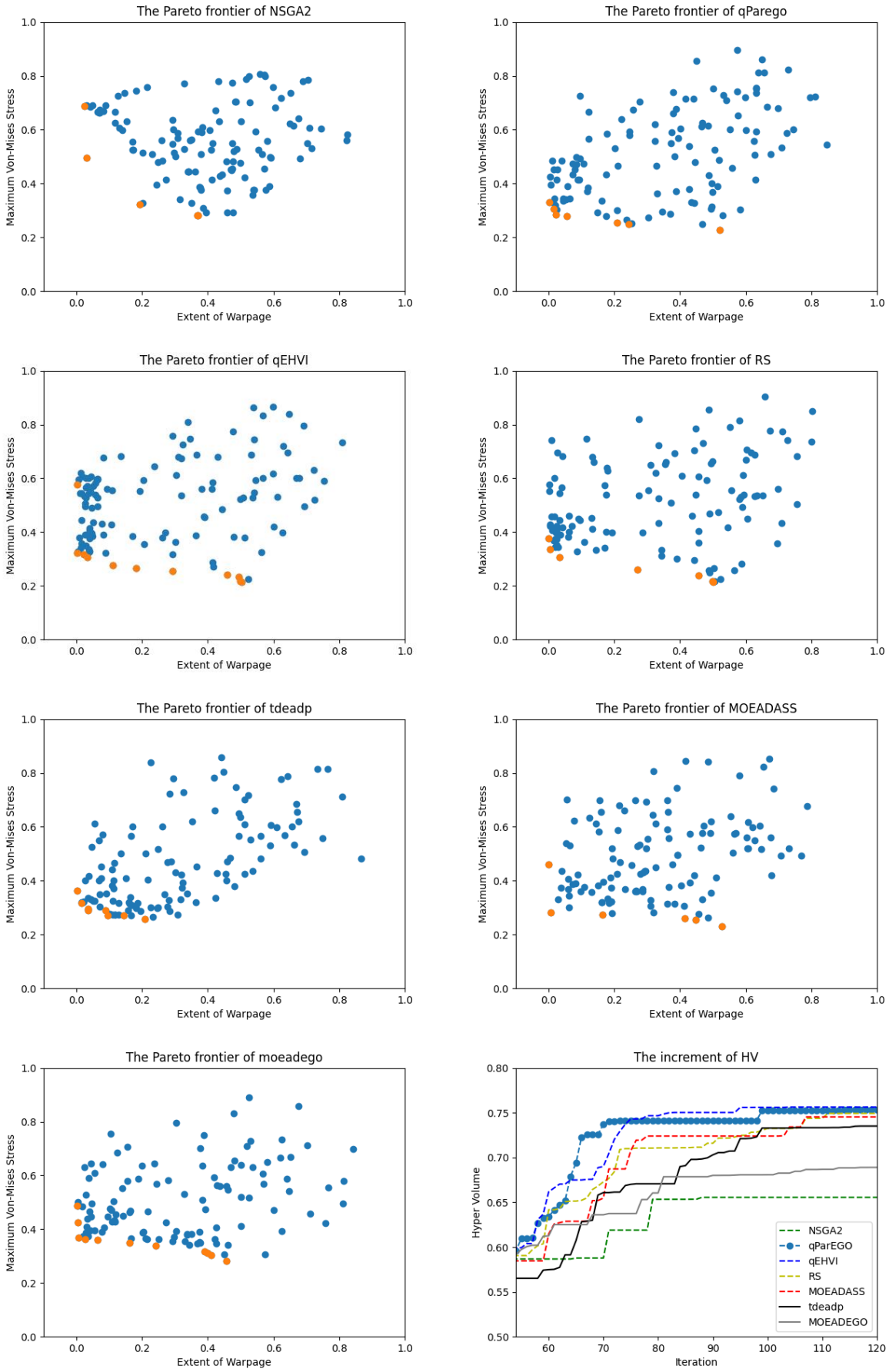


Fig. 7. PF of algorithms implemented and the HV values.

2) Bayes-optimal policy with Gaussian process surrogate has a great performance comparing to neural network surrogate. The top3 algorithms are qEHVI, qParEGO and RS, they are both Bayesian-surrogate-based. θ-DEA-DP algorithm we implemented utilized Feedforward Neural Network (FNN) and the performance is not good.
3) θ-DEA-DP has better distribution in the solution space according to the sampling map, we can tell its superiority in balance the solution distribution compare with other algorithms such as qParEGO and qEHVI.
4) qParEGO has great robustness, its performance is unexpectedly good and outperformed many state of arts algorithms.
5) qEHVI is the best algorithm among all of the seven algorithms we implemented, inferring from its Pareto Frontier, its distribution quality can still be improved.
6) Less offspring generated in each iterations in EGO usually leads to better performance, we tested the performance of NSGA-II in different number of offspring and we found that generating 1 offspring each time is 5 percents better than generating 10 offspring.

V. Conclusion and Future Work

A. Conclusion

In this paper, we discussed a new method to optimize the extend of warpage and maximum von Mises stress in chip packaging design industry. Finite element model was built in Abaqus software for simulation analysis, and the interfaces between Abaqus and Pycharm or MatLab are built to conduct the real-time estimation for sampling points. A wide range of algorithms are implemented in python or matlab, including qEHVI, qParEGO, θ-DEA-DP, MOEA/D-EGO, MOEA/D-ASS, random scalarization and NSGA-II. We tested those algorithms in our black box function and conducted a comparison. By presenting the Hypervolume results and Pareto frontier, we demonstrated the superiority of qEHVI in our problem and discussed the advantage of other algorithms in solution space distribution.

B. Future work

Due to time limitation, we have not conducted a perfect experiment, here are some future work we would like take a deep-dive in:

1) Further study in θ-DEA-DP, since we simply used the FNN surrogate to predict the dominance relationship of sampling points. Based on our research, Gaussian models performed better than FNNs, we might improved the performance significantly by change the surrogate model to Gaussian models. Nevertheless, we can also use other networks like Resnet to see the difference in θ-DEA-DP.
2) We will do more experiments to find the reason why qParEGO works well. If reasonable, we can implement our algorithm based on it.
3) There are many another aspects that we can implement, such as acquisition function, sub-problem selection and so on.

Acknowledgment

The authors acknowledge Prof. Zhenkun Wang for his untiring inculcation during the 2021 Fall semester.

References

[1] A. S. Mutschler, "Applying machine learning to chips," https://semiengineering.com/applying-machine-learning-to-chips/.

[2] B. Khailany, H. Ren, S. Dai, S. Godil, B. Keller, R. Kirby, A. Klinefelter, R. Venkatesan, Y. Zhang, B. C. Catanzaro, and W. J. Dally, "Accelerating chip design with machine learning," *IEEE Micro*, vol. 40, pp. 23–32, 2020.

[3] M. Yasunaga, I. Yoshihara, and J. H. Kim, "The design of segmental-transmission-line for high-speed digital signals using genetic algorithms," in *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*, 2003.

[4] H. Zaidan, R. Ahmad, and N. M. Ghazali, "Optimization of multilayer micro channels heat sinks cooling system using genetic algorithm," in *International Conference on Smart Sensors Application*, 2015.

[5] R. Manimegalai, E. S. Soumya, V. Muralidharan, B. Ravindran, and D. Bhatia, "Placement and routing for 3d-fpgas using reinforcement learning and support vector machines," in *VLSI Design, 2005. 18th International Conference on*, 2005.

[6] L. Yongsheng, "Research and optimization design of electrothermal characteristics of 3d integrated package," Ph.D. dissertation, Zhejiang university,.

[7] J. hang, "Research on high speed packaging integrated electromagnetic radiation suppression based on machine learning," Ph.D. dissertation, Zhejiang university,.

[8] W. Dai, K. Xue, J. Wu, and Z. Liu, "Effective parametric optimization for packaging design using bayesian optimization," in *2020 21st International Conference on Electronic Packaging Technology (ICEPT)*, 2020, pp. 1–6.

[9] Y. L. Tzeng, N. Kao, E. Chen, J. Y. Lai, Y. P. Wang, and C. Hsiao, "Warpage and stress characteristic analyses on package-on-package (pop) structure," in *2007 9th Electronics Packaging Technology Conference*, 2007, pp. 482–487.

[10] L. Jiahe, "Machine learning aided design optimization for micro-chip reliability improvement," in *2020 3rd World Conference on Mechanical Engineering and Intelligent Manufacturing (WCMEIM)*, 2020, pp. 131–135.

[11] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[12] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans-*

actions on evolutionary computation, vol. 11, no. 6, pp. 712–731, 2007.

[13] J. Knowles, "Parego: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 1, pp. 50–66, 2006.

[14] J. J. Liang, C. Yue, and B.-Y. Qu, "Multimodal multi-objective optimization: A preliminary study," in *2016 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2016, pp. 2454–2461.

[15] Z. Wang, Q. Zhang, Y.-S. Ong, S. Yao, H. Liu, and J. Luo, "Choose appropriate subproblems for collaborative modeling in expensive multiobjective optimization," *IEEE Transactions on Cybernetics*, 2021.

[16] Y. Yuan and W. Banzhaf, "Expensive multi-objective evolutionary optimization assisted by dominance prediction," *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2021.

[17] J. Lin, C. He, and R. Cheng, "Adaptive dropout for high-dimensional expensive multiobjective optimization," *Complex & Intelligent Systems*, pp. 1–15, 2021.

[18] G. Li, Q. Zhang, Q. Lin, and W. Gao, "A three-level radial basis function method for expensive optimization," *IEEE Transactions on Cybernetics*, 2021.

[19] S. Rojas-Gonzalez and I. Van Nieuwenhuyse, "A survey on kriging-based infill algorithms for multiobjective simulation optimization," *Computers & Operations Research*, vol. 116, p. 104869, 2020.

[20] Y. Tian, R. Cheng, X. Zhang, and Y. Jin, "Platemo: A matlab platform for evolutionary multi-objective optimization [educational forum]," *IEEE Computational Intelligence Magazine*, vol. 12, no. 4, pp. 73–87, 2017.

[21] Y. Tian, M. K. Luković, T. Erps, M. Foshey, and W. Matusik, "Autooed: Automated optimal experiment design platform," *arXiv preprint arXiv:2104.05959*, 2021.

[22] M. Balandat, B. Karrer, D. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy, "Botorch: A framework for efficient monte-carlo bayesian optimization," *Advances in neural information processing systems*, vol. 33, 2020.

[23] Z. Zheng, L. Xiaoling, L. Chaobiao, and R. Li, "Multi-objective optimization design of electric-thermal-mechanical power module package," *Proceedings of the CSEE*, vol. 17, 2019.

[24] S. Prinz, J. Thomann, G. Eichfelder, T. Boeck, and J. Schumacher, "Expensive multi-objective optimization of electromagnetic mixing in a liquid metal," *Optimization and Engineering*, vol. 22, no. 2, pp. 1065–1089, 2021.

[25] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[26] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary computation*, vol. 8, no. 2, pp. 173–195, 2000.

[27] E. Brochu, V. M. Cora, and N. De Freitas, "A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," *arXiv preprint arXiv:1012.2599*, 2010.

[28] S. Suzuki, S. Takeno, T. Tamura, K. Shitara, and M. Karasuyama, "Multi-objective bayesian optimization using pareto-frontier entropy," in *International Conference on Machine Learning*. PMLR, 2020, pp. 9279–9288.

[29] Y.-S. Lee, P.-Y. Lin, K.-T. Wu, H.-H. Lee, and S.-J. Hwang, "Effects of substrate structure on the warpage of flip chip ic packages," in *2018 13th International Microsystems, Packaging, Assembly and Circuits Technology Conference (IMPACT)*, 2018, pp. 66–70.

[30] H. Miura, N. Ueta, and Y. Sato, "Distribution of local thermal residual stress in thin chips stacked by flip chip structures," in *2006 International Microsystems, Package, Assembly Conference Taiwan*, 2006, pp. 1–4.

[31] S. Daulton, M. Balandat, and E. Bakshy, "Parallel bayesian optimization of multiple noisy objectives with expected hypervolume improvement," *arXiv preprint arXiv:2105.08195*, 2021.

[32] Daulton, Samuel and Balandat, Maximilian and Bakshy, Eytan, "Differentiable expected hypervolume improvement for parallel multi-objective bayesian optimization," *arXiv preprint arXiv:2006.05078*, 2020.

[33] Q. Zhang, W. Liu, E. Tsang, and B. Virginas, "Expensive multiobjective optimization by moea/d with gaussian process model," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 3, pp. 456–474, 2009.

[34] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii," *IEEE transactions on evolutionary computation*, vol. 13, no. 2, pp. 284–302, 2008.