Surrogate-Assisted Evolutionary Multi-Objective Optimization for Hardware Design Space Exploration

Renzhi Chen
National Innovative Institute of Defense Technology
Beijing, 100080, China
chenrenzhi1989@gmail.com

Ke Li
Department of Computer Science
University of Exeter
Exeter, EX4 4QF, UK
k.li@exeter.ac.uk

Abstract

Hardware design space exploration (DSE) aims to find a suitable micro-architecture for the dedicated hardware accelerators. It is a computationally expensive black-box optimization problem with more than one conflicting performance indicator. Surrogate-assisted evolutionary algorithm is a promising framework for expensive multi-objective optimization problems given its surrogate modeling for handling expensive objective functions and population-based characteristics that search for a set of trade-off solutions simultaneously. However, most, if not all, existing studies mainly focus 'regular' Pareto-optimal fronts (PFs), whereas the PF is typically irregular in hardware DSE. In the meanwhile, the gradient information of the differentiable surrogate model(s) is beneficial to navigate a more effective exploration of the search space, but it is yet fully exploited. This paper proposes a surrogate-assisted evolutionary multi-objective optimization based on multiple-gradient descent (MGD) for hardware DSE. Empirical results on both synthetic problems with irregular PFs and real-world hardware DSE cases fully demonstrate the effectiveness and outstanding performance of our proposed algorithm.

1 Introduction

Dedicated hardware accelerator has become an emerging area in machine learning Lee and Verma [2013], Chen et al. [2016], Auten et al. [2020]. Hardware design space exploration (DSE) Srinivasan et al. [1998], Ipek et al. [2006], Nardi et al. [2019] involves finding a suitable micro-architecture for the dedicated hardware accelerator. This is in practice computationally expensive, e.g., the synthesis and implementation for a deep binary neural network design takes hours on the field programmable gate arrays (FPGAs) Zhou et al. [2017]. Furthermore, it is not uncommon that a hardware engineer needs to consider more than one performance indicator, while they are usually conflicting with each other, as known as multi-objective optimization problems (MOPs). For example, the CPU frequency and the energy consumption are two confrontational indicators in a CPU design Bai et al. [2021].

Due to the population-based characteristics, evolutionary algorithms (EAs) have been widely recognized as an effective approach for MO Deb [2001]. One of the major criticisms of EAs is its daunting amount of function evaluations (FEs) required to obtain a set of reasonable solutions. To mitigate this issue, data-driven evolutionary optimization, guided by surrogate models of computationally expensive objective functions, have become as a powerful approach for expensive optimization Jin et al. [2019]. However, most, if not all, existing studies are mainly designed and validated on synthetic benchmark test problems characterized as 'regular' triangle-like Pareto-optimal fronts (PFs). Unfortunately, this is unrealistic in the hardware DSE scenario as PFs are featured as incomplete, and/or badly-scaled, it is surprising that the research on handling MOPs with irregular PFs is lukewarm in the context of data-driven EMO. In addition, the evolutionary operators are directly derived from

2022 NeurIPS Workshop on Gaussian Processes, Spatiotemporal Modeling, and Decision-making Systems.

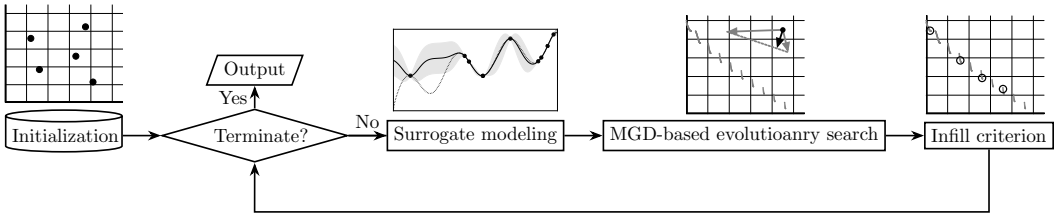

Figure 1: The flow chart of the proposed SAEMO/MGD.

the EA (e.g., crossover and mutation Deb and Agrawal [1995] and differential evolution Storn and Price [1997]). The gradient information of differentiable surrogate model(s) is beneficial to navigate a more effective exploration of the search space, but it is yet fully exploited.

Bearing these considerations in mind, this paper proposes a surrogate-assisted evolutionary multi-objective optimization based on multiple-gradient descent (MGD) Désidéri [2014] for multi-objective hardware DSE. Our basic idea is to leverage the latent information of surrogate models, gradient(s) in particular, to explore promising candidate solutions and recommend a batch of promising candidate solutions for parallel expensive function evaluations (FEs), each of which is a synthesis and implementation of a design point. Empirical results on both synthetic problems with irregular PFs and real-world hardware DSE cases fully demonstrate the effectiveness and outstanding performance of our proposed algorithm.

2 Proposed Method

This section briefly introduce the implementation of our proposed surrogate-assisted evolutionary multi-objective optimization based on multiple-gradient descent (dubbed SAEMO/MGD). As the flowchart shown in Figure 1, SAEMO/MGD starts with an initialization step based on an experimental design method such as Latin hypercube sampling Santner et al. [2018]. Note that these initial samples will be evaluated based on the computationally expensive objective functions.

- Surrogate modeling: This step builds a surrogate model by using the Gaussian process regression Rasmussen and Williams [2006] for each expensive objective function based on the samples evaluated on this expensive objective function so far. Here we use radial basis function (RBF) kernel because it is infinitely differentiable. Since there are ordinal variables in the multi-objective hardware DSE, this paper applies a simple rounding method to replace the continuous variable by its closest integer.
- MGD-based evolutionary search: This step aims to search for a set of promising candidate solutions $\mathcal{P} = \{\hat{\mathbf{x}}^i\}_{i=1}^{\tilde{N}}$, which are assumed to be an appropriate approximation to the PF, based on the surrogate model built in the surrogate modeling step. In a nutshell, the basic idea of MGD Désidéri [2014] is to iteratively update a solution so that all objective functions can thus be improved equally. More detailed discussion of this step can be found in Section 1 of the supplemental document[1].
- Infill criterion: This step aims to pick up $1 \leqslant \xi \ll \tilde{N}$ promising solutions from $\mathcal{P}$ for expensive FEs. These newly evaluated solutions are then used to update the training dataset for the next iteration. Since FEs in the hardware DSE can be carried out in parallel given the availability of platform, such batched recommendation provides an actionable way for parallelization. In this paper, we propose a simple infill criterion based on the individual Hypervolume Zitzler and Thiele [1999] contribution (IHV). Specifically, the IHV of each candidate solution $\mathbf{x} \in \mathcal{P}$ is calculated as:

$$\texttt{IHV}(\mathbf{x}) = \texttt{HV}(\mathcal{P}) - \texttt{HV}(\mathcal{P} \backslash \{\mathbf{x}\}), \tag{1}$$

  where $\texttt{HV}(\mathcal{P})$ evaluates the Hypervolume of $\mathcal{P}$. Then, the top ξ solutions in $\mathcal{P}$ with the largest IHV are picked up for the expensive FEs.

[1]The supplemental document is located in https://tinyurl.com/2p9kuezm.

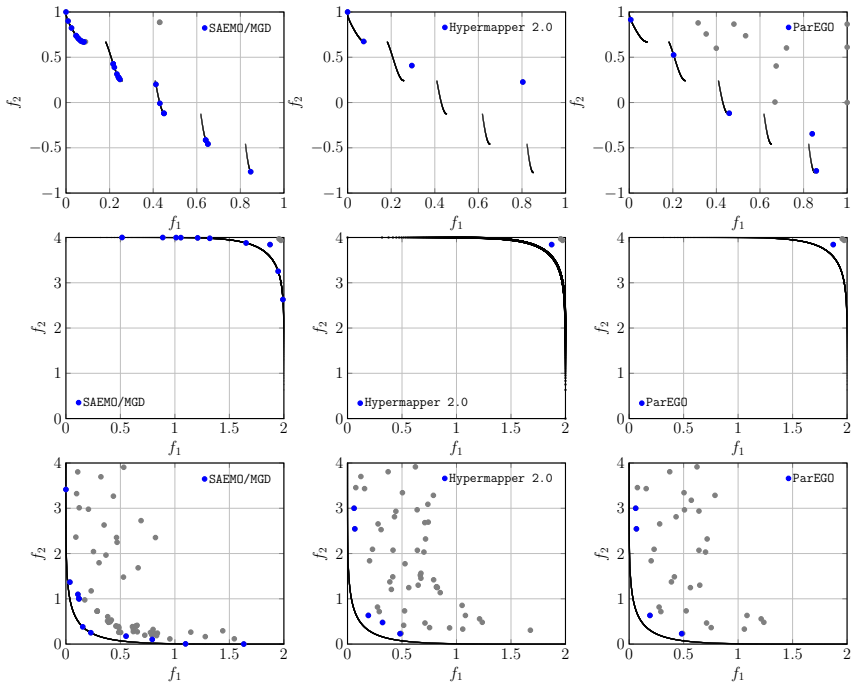

Figure 2: Solutions obtained by SAEMO/MGD, HyperMapper and ParEGO on ZDT3 (first row), WFG43 (second row) and WFG44 (third row). The PF is represented as black dots •. The non-dominated solutions are represented as blue cirles •, while those dominated ones are shown as gray circles •.

3 Empirical Study

In this section, we empirically investigate the performance of our proposed SAEMO/MGD compared against HyperMapper Nardi et al. [2019] and ParEGO Knowles [2006]. The prior one is a widely recognized DSE approach and the latter one is a popular surrogate-assisted evolutionary multi-objective optimization algorithm in the literature. Three synthetic benchmark test problems ZDT3 Zitzler et al. [2000], WFG43 and WFG44 Huband et al. [2006] with irregular PFs and several hardware DSE cases are used to constitute our benchmark suite. To have a quantitative evaluation of the performance of different algorithms, we use the widely used Hypervolume (HV) as the performance metric. To have a statistical interpretation of the significance of comparison results, we apply the widely used non-parametric Wilcoxon signed-rank test Wilcoxon [1945] at the 0.05 significance level to conduct a pairwise comparison. The experimental setup is delineated in Section 2 of the supplemental document.

3.1 Results on synthetic benchmark test problems

From the plots of solution distribution shown in Figure 2 and the comparison of HV values shown in Table 4 in the supplemental document, we can clearly see the superior performance of our proposed SAEMO/MGD over the other two peer algorithms. More specifically, as for ZDT3, only SAEMO/MGD can find non-dominated solutions on all five disconnected PF segments. As for WFG43 with a strong convex PF, the non-dominated solutions obtained by SAEMO/MGD are on the PF and they dominate those found by HyperMapper and ParEGO, which are crowded on the 'knee' region away from the PF. As for WFG44 with a strong concave PF, most non-dominated solutions obtained by SAEMO/MGD are on the PF whereas the other two peer algorithms are struggling to converge to the PF.

3.2 Results on hardware DSE cases

In this subsection, we consider optimizing the mirco-architecture of three typical dedicated hardware accelerators including the general matrix multiplication (GEMM), a common operator in linear algebra, the stochastic gradient descent (SGD), a gradient-based optimizer widely used for training

Table 1: The number of parameters and the size of the design space for the DSE problems

DSE Problem	Number of parameters	Size of the design space
GEMM	6	2.7×10^4
SGD	5	3.3×10^4
k-means	6	8.9×10^3

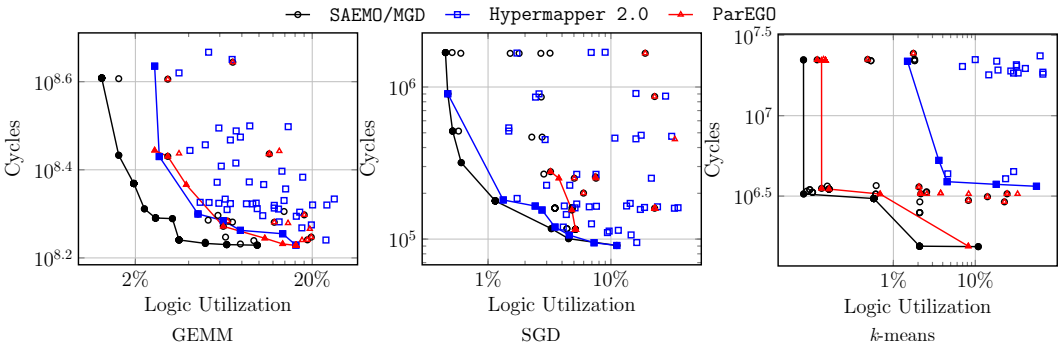

Figure 3: Solutions obtained by `SAEMO/MGD`, `HyperMapper` and `ParEGO` for GEMM (left), SGD, k-means. Both axis are in a log scale. The x-axis is compute logic, reported as a percentage of the total LUT capacity of the FPGA. The y-axis is the cycles taken to solve the problem. The non-dominated solutions are represented as solid circles •, squares ■ or triangles ▲, respectively; whereas those dominated ones are shown as the corresponding open shapes ○, □, △, respectively.

neural networks, and the k-means algorithm. Our experiments are developed upon Spatial in the `Xilinx VIRTEX VC707`. In particular, Spatial is a high-level synthesis (HLS) tool for the design of accelerators implemented on reconfigurable spatial architectures such as FPGAs Koeplinger et al. [2018]. It facilitates engineers to explore the design space including the hardware-specific abstractions for control, memory, and design tuning. In our experiments, we consider optimizing two confrontational objective functions including the logic utilization and the execution cycles. The number variables and the size of the design space are listed in Table 1 while more detailed information can be found in Section 2.2 of the supplementary document. Each algorithm is allocated with 10 initialized samples and the computational budget is limited to 40 function evaluations.

From the results shown in Figure 3 and Table 2, it is clear to see that our proposed `SAEMO/MGD` is the most competitive algorithm as the non-dominated solutions always dominate those obtained by `HyperMapper` and `ParEGO`. It is interesting to note that `HyperMapper` and `ParEGO` are comparable for the GEMM design while `HyperMapper` is better than `ParEGO` on the SGD design and the result is flipped on the k-means design.

Table 2: Comparison results of Hypervolume on the hardware DSE problems

DSE Problems	`SAEMO/MGD`	`Hypermapper`	`ParEGO`
GEMM	**0.979(2.3E-3)**	0.957(3.1E-3)†	0.959(2.5E-3)†
SGD	**0.970(1.3E-4)**	0.942(9.4E-5)†	0.715(7.6E-4)†
K-means	**0.997(5.7E-4)**	0.835(8.9E-5)†	0.986(4.8E-4)†

According to Wilcoxon rank sum test at the 0.05 significance level, † indicate that the corresponding algorithm is significantly worse than `SAEMO/MGD` .

4 Conclusion

This paper develops a surrogate-assisted evolutionary multi-objective optimization based on multiple-gradient descent to solve hardware DSE problems featured by the computationally expensive objective function evaluations and irregular PFs. Our proposed algorithm has shown competitive performance on both synthetic benchmark test problems with irregular PFs and three hardware DSE cases.

5 Acknowledgment

This work was supported by UKRI Future Leaders Fellowship (MR/S017062/1), NSFC (62076056), EPSRC (2404317), Royal Society (IES/R2/212077) and Amazon Research Award.

References

Adam Auten, Matthew Tomei, and Rakesh Kumar. Hardware acceleration of graph neural networks. In *DAC'20: Proc. of the 57th ACM/IEEE Design Automation Conference*, pages 1–6. IEEE, 2020.

Chen Bai, Qi Sun, Jianwang Zhai, Yuzhe Ma, Bei Yu, and Martin D. F. Wong. Boom-explorer: RISC-V BOOM microarchitecture design space exploration framework. In *ICCAD'21: Proc. of the IEEE/ACM International Conference On Computer Aided Design*, pages 1–9. IEEE, 2021.

Yunji Chen, Tianshi Chen, Zhiwei Xu, Ninghui Sun, and Olivier Temam. DianNao family: energy-efficient hardware accelerators for machine learning. *Commun. ACM*, 59(11):105–112, 2016.

Kalyanmoy Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

Kalyanmoy Deb and Ram Bhushan Agrawal. Simulated binary crossover for continuous search space. *Complex Systems*, 9, 1994.

Kalyanmoy Deb and Ram Bhushan Agrawal. Simulated binary crossover for continuous search space. *Complex Syst.*, 9(2), 1995.

Kalyanmoy Deb and Mayank Goyal. A combined genetic adaptive search (GeneAS) for engineering design. *Computer Science and Informatics*, 26:30–45, 1996.

Jean-Antoine Désidéri. Multiple-gradient descent algorithm for Pareto-front identification. In William Fitzgibbon, Yuri A. Kuznetsov, Pekka Neittaanmäki, and Olivier Pironneau, editors, *Modeling, Simulation and Optimization for Science and Technology*, volume 34 of *Computational Methods in Applied Sciences*, pages 41–58. Springer, 2014.

Simon Huband, Philip Hingston, Luigi Barone, and R. Lyndon While. A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Trans. Evol. Comput.*, 10(5):477–506, 2006.

Engin Ipek, Sally A. McKee, Rich Caruana, Bronis R. de Supinski, and Martin Schulz. Efficiently exploring architectural design spaces via predictive modeling. In *ASPLOS'06: Proc. of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 195–206. ACM, 2006.

Yaochu Jin, Handing Wang, Tinkle Chugh, Dan Guo, and Kaisa Miettinen. Data-driven evolutionary optimization: An overview and case studies. *IEEE Trans. Evol. Comput.*, 23(3):442–458, 2019.

Joshua D. Knowles. ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Trans. Evol. Comput.*, 10(1):50–66, 2006.

David Koeplinger, Matthew Feldman, Raghu Prabhakar, Yaqi Zhang, Stefan Hadjis, Ruben Fiszel, Tian Zhao, Luigi Nardi, Ardavan Pedram, Christos Kozyrakis, and Kunle Olukotun. Spatial: a language and compiler for application accelerators. In *PLDI'18: Proc. of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 296–311. ACM, 2018.

Kyong-Ho Lee and Naveen Verma. A low-power processor with configurable embedded machine-learning accelerators for high-order and adaptive analysis of medical-sensor signals. *IEEE J. Solid State Circuits*, 48(7):1625–1637, 2013.

Luigi Nardi, Artur L. F. Souza, David Koeplinger, and Kunle Olukotun. Hypermapper: a practical design space exploration framework. In *MASCOTS'19: Proc. of the 27th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, pages 425–426. IEEE Computer Society, 2019.

H. B. Nielsen, S. N. Lophaven, and J. Søndergaard. DACE — A MATLAB Kriging toolbox, 2002.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, 2006. ISBN 026218253X.

Thomas J. Santner, Brian J. Williams, and William I. Notz. *The Design and Analysis of Computer Experiments, Second Edition*. Springer-Verlag, 2018.

Vinoo Srinivasan, Shankar Radhakrishnan, and Ranga Vemuri. Hardware software partitioning with integrated hardware design space exploration. In *DATE'98: Proc. of the 1998 Design, Automation and Test in Europe*, pages 28–35. IEEE Computer Society, 1998.

Rainer Storn and Kenneth V. Price. Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.*, 11(4):341–359, 1997. doi: 10.1023/A:1008202821328.

Rui Wang, Robin C. Purshouse, and Peter J. Fleming. Preference-inspired co-evolutionary algorithms using weight vectors. *Eur. J. Oper. Res.*, 243(2):423–441, 2015.

Frank Wilcoxon. Individual comparisons by ranking methods. 1945.

Yuteng Zhou, Shrutika Redkar, and Xinming Huang. Deep learning binary neural network on an FPGA. In *MWSCAS'17: Proc. of the IEEE 60th International Midwest Symposium on Circuits and Systems*, pages 281–284. IEEE, 2017.

Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Trans. Evol. Comput.*, 3(4):257–271, 1999.

Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evol. Comput.*, 8(2):173–195, 2000.

A MGD-based Evolutionary Search

The working mechanism of this `MGD-based evolutionary search` step is given in Algorithm 1.

Algorithm 1: MGD-based Evolutionary Search

Input: Surrogate models for objective functions $\mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), \cdots, f_m(\mathbf{x}))^\top$
Output: Candidate solutions set $\mathcal{P}$

Initialize a candidate solution set $\mathcal{P} = \{\hat{\mathbf{x}}^i\}_{i=1}^{\tilde{N}}$ based on Latin hypercube sampling;
while *stopping criterion is not met* **do**
  Generate set of offspring solutions $\mathcal{Q}$ from $\mathcal{P}$ based on SBX crossover;
  for *each solution* $\hat{\mathbf{x}}^i \in \mathcal{Q}$ **do**
    Calculate the gradients $\nabla\overline{g}_j(\hat{\mathbf{x}}^i)$ of surrogate models at $\hat{\mathbf{x}}^i$, where $j \in \{1, \cdots, m\}$;
    Find a nonnegative unit vector$\mathbf{w}^* = (w_1^*, \cdots, w_m^*)^\top$ that satisfies:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \left\| \sum_{j=1}^{m} w_j \nabla\overline{g}_j(\hat{\mathbf{x}}^i) \right\|, \tag{2}$$

    where $\mathbf{w} = (w_1, \cdots, w_m)^\top$, $\sum_{i=1}^m w_i = 1$ and $w_i \geqslant 0$, $i \in \{1, \cdots, m\}$;
    Obtain a directional vector $\mathbf{u}^*$ as:

$$\mathbf{u}^* = \begin{cases} \underset{1 \leqslant j \leqslant m}{\operatorname{argmax}} \left\|\nabla\overline{g}_j(\hat{\mathbf{x}}^i)\right\|, & \text{if } \sum_{j=1}^m w_j^* \nabla\overline{g}_j(\hat{\mathbf{x}}^i) = 0 \\ \underset{1 \leqslant j \leqslant m}{\operatorname{argmin}} \left\|\nabla\overline{g}_j(\hat{\mathbf{x}}^i)\right\|, & \text{if } \exists i, j \in \{1, \cdots, m\}, \langle \nabla\overline{g}_j(\hat{\mathbf{x}}^i), \nabla\overline{g}_j(\hat{\mathbf{x}}^i) \rangle > \delta \\ \sum_{j=1}^m w_j^* \nabla\overline{g}_j(\hat{\mathbf{x}}^i), & \text{otherwise} \end{cases} \tag{3}$$

    where $\langle *, * \rangle$ is the acute angle between vectors, and $\delta = \min\left\{ \left\|\nabla\overline{g}_j(\hat{\mathbf{x}}^i)\right\| \right\}_{j=1}^m$;
    Exploit $\hat{\mathbf{x}}$ along the direction of MGD $\mathcal{Q} \leftarrow \mathcal{Q} \bigcup \{\hat{\mathbf{x}}^i + \eta \mathbf{u}^*\} \backslash \{\hat{\mathbf{x}}^i\}$
  Amend the updated solution to $\mathcal{P} \leftarrow \mathcal{Q}$;

As discussed in Désidéri [2014], the multiple-gradient descent (MGD) is a natural extension of the single-objective gradient to finding a PF. In a nutshell, its basic idea is to iteratively update a solution $\mathbf{x}$ along a 'specified' direction so that all objective functions can thus be improved.

Figure 4 gives an illustrative example for each of the three conditions given in equation (3) when $m = 2$. More specifically, when the gradients of two objective functions are in opposite directions

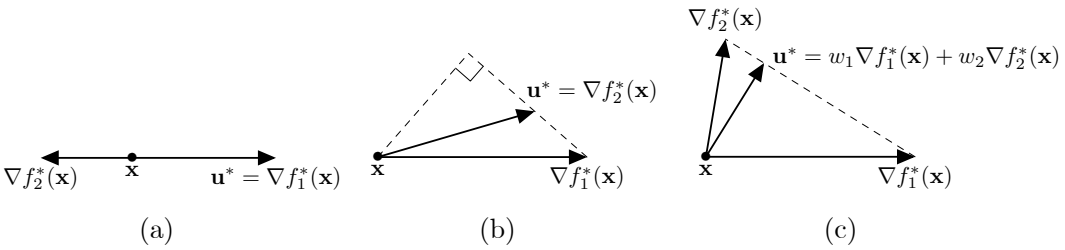

Figure 4: Illustrative examples of the calculation of $\mathbf{u}^*$ in equation (3).

as shown in Figure 4(a), $\mathbf{u}^*$ is chosen as the one with a larger Euclidean norm. If the gradients are too close to each other as shown in Figure 4(b), $\mathbf{u}^*$ is chosen as the one with a smaller Euclidean norm. Different from the linear weighted aggregation, the MGD works for non-convex PF. Therefore, we can expect a descent diversity in case the initial population is well distributed as the illustrative example shown in Figure 5. To be more specific, Figure 5(a) and Figure 5(b) shows the scenario in which solutions have not converged to PF. In this case, the direction of $\mathbf{u}^*$ points toward PF with all objective functions improving equally. If the solutions are well distributed in decision space like Figure 5(a), the solutions will maintain the diversity when converging to the PF like Figure 5(b). Figure 5(c) shows the scenario in which solutions have already converged to PF. In this case, the direction of $\mathbf{u}^*$ points alongside the PS.

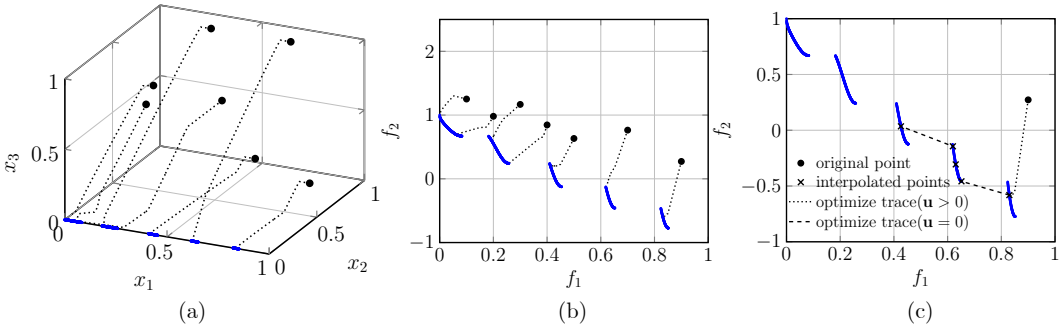

Figure 5: Illustrative examples of working mechanism of the `MGD-based evolutionary search` step.

B Experimental Setup

This section introduces our experimental setup including the definitions of the synthetic benchmark test problems along the real-world hardware DSE problems, the peer algorithms along with their parameter settings, and the performance metric and the statistical test.

B.1 Benchmark Test Problems

This paper considers ZDT3 Zitzler et al. [2000], DTLZ43 and WFG44 Wang et al. [2015] to constitute our synthetic benchmark test problems. Note that all these problems are with irregular PF. Specifically, the PF of ZDT3 consists of five disconnected segments, the PF of WFG43 is strong concave while that of WFG44 is strong convex.

For the ZDT3 problem, it is mathematically defined as:

$$\begin{aligned} \text{minimize} \quad & f_1(\mathbf{x}) = x_1 \\ \text{minimize} \quad & f_2(\mathbf{x}) = g(\mathbf{x}) \left(1 - \sqrt{x_1/g(\mathbf{x})} - x_1/g(\mathbf{x}) \sin(10\pi x_1)\right) \end{aligned}, \tag{4}$$

where

$$g(\mathbf{x}) = 1 + \frac{9}{n-1} \sum_{i=2}^{n} x_i, \tag{5}$$

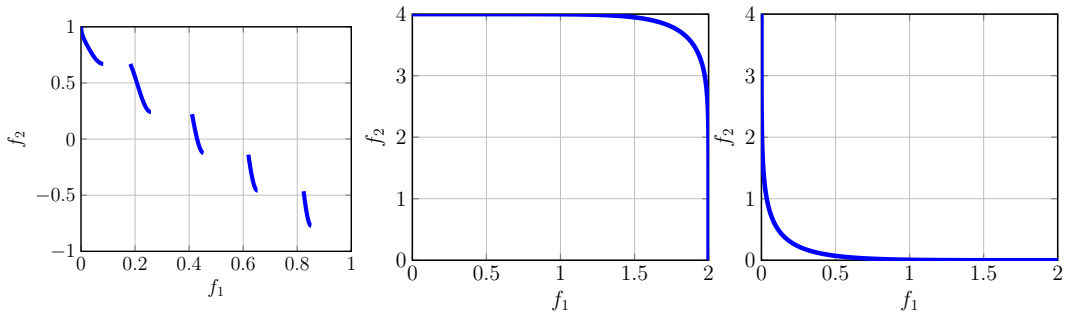

Figure 6: The PF for ZDT3 (left, disconnected), WFG43 (center, strong concave) and WFG44 (right, strong convex).

where n is the number of variables.

For the WFG43 problem, it is mathematically defined as:

$$\begin{array}{lll} \text{minimize} & f_1(\mathbf{x}) & = \Big(\sin(t_1(\mathbf{x})\pi/2)\sin\big(\sum_{i=2}^{n} t_i(\mathbf{x})\pi/2\big)\Big)^{0.25} \\ \text{minimize} & f_2(\mathbf{x}) & = \Big(1-\sin\big(t_1(\mathbf{x})\pi/2\big)\Big)^{0.25} \end{array}, \tag{6}$$

where

$$\mathbf{t}(\mathbf{x}) = \frac{1+\cos\big((4A+2)\pi(0.5-\frac{|\mathbf{x}-C|}{2(\lfloor C-\mathbf{x}\rfloor+C)})\big)+4B(\frac{|\mathbf{x}-C|}{2(\lfloor C-\mathbf{x}\rfloor+C)})^2}{B+2}, \tag{7}$$

where $A=30, B=10, C=0.35$, and

$$\begin{array}{l} z_1 = x_1/2 \\ z_n = x_n/(2n) \\ z_i = \big(x_i/(2i)-0.5\big)*z_n+0.5 \end{array}, \tag{8}$$

where $i \in \{2,\cdots,n-1\}$.

For the WFG44 problem, it is mathematically defined as:

$$\begin{array}{lll} \text{minimize} & f_1(\mathbf{x}) & = \Big(1-\cos(t_1(\mathbf{x})\pi/2)\sin\big(\sum_{i=2}^{n} t_i(\mathbf{x})\pi/2\big)\Big)^{4} \\ \text{minimize} & f_2(\mathbf{x}) & = \Big(\sin\big(t_1(\mathbf{x})\pi/2\big)\Big)^{4} \end{array}, \tag{9}$$

where

$$\mathbf{t}(\mathbf{z}) = \frac{1+\cos\big((4A+2)\pi(0.5-\frac{|\mathbf{z}-C|}{2(\lfloor C-\mathbf{z}\rfloor+C)})\big)+4B(\frac{|\mathbf{z}-C|}{2(\lfloor C-\mathbf{z}\rfloor+C)})^2}{B+2} \tag{10}$$

where $A=30, B=10, C=0.35$, and

$$\begin{array}{l} z_1 = x_1/2 \\ z_n = x_n/(2n) \\ z_i = \big(x_i/(2i)-0.5\big)*z_n+0.5 \end{array}, \tag{11}$$

where $i \in \{2,\cdots,n-1\}$.

The Pareto fronts of ZDT3, WFG43 and WFG44 are given in Figure 6.

B.2 Hardware DSE Problems

In our experiments, we also consider three hardware DSE problems.

- General matrix multiply (GEMM): The hardware accelerated GEMM uses the blocking technique. We consider a $M \times K$ matrix A multiply with a $K \times N$ matrix B. We will split A into $tile_m \times tile_k$ and B into $tile_k$ and $tile_n$ as shown in Figure 7. As a result,

the multiplying process is folded into three levels of loops. The first three variables are $tile_m$, $tile_k$ and $tile_n$. The other three p_1, p_2 and p_3 indicate whether these loops should be pipelining. Table 3 shows the detail of the variables. To validate the hardware design, we randomly generate 512×512 matrix multiply with 512×512 matrix and test it on the hardware.

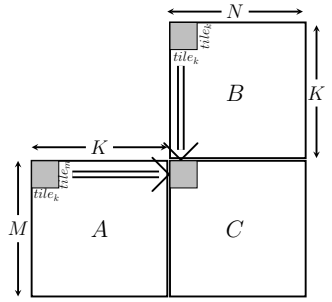

Figure 7: The blocking tecnique in GEMM.

Table 3: The parameters and their range for the GEMM case

variables	range
$tile_m$	8,12, $\cdots$, 60, 64
$tile_n$	8,12, $\cdots$, 60, 64
$tile_k$	8,12, $\cdots$, 60, 64
p_1	true,false
p_2	true,false
p_3	true,false

- Stochastic gradient descent (SGD): The hardware accelerated SGD is based on the parallelized SGD **?**. We consider optimizing a D dimensional vector $\mathbf{x}$ with gradients. As shown in Figure 8, the first variable is $tileSize$, which indicates the size of the small piece of $\mathbf{x}$ updated in one execution cycle. Another two variables ip and op indicate the inner loop size and the outer loop size. The inner loop size ip is the number of tiles executed in paralle. It is worth noting that load and store are not included in the inner loop. op is the number of the inner loops, as well as load and store units, executed in parallel. The algorithm includes two level of loops. The last two variables p_1, p_2 indicate whether these loops should be pipelining. Table 4 shows the detail of the variables. To validate the hardware design, we randomly generate 512-bit vectors to be optimized and test them on the hardware.
- k-means: The hardware accelerated k-means has six variables. As the k-means involves penalty of data, one of the main challenges for this hardware is load and store the data. The first variable BN is the number of points to load/store together. The second variable ip is the number of points calculated in parallel (inner loop). The third one op is the size of outer loops, which include inner loop and load and store unit. The algorithm includes three level of loops. The last three variables p_1, p_2 and p_3 indicate whether these loops should be pipelining.

Due to the resource constraint on dedicated hardware like FPGAs, GEMM, SGD and k-Means usually cannot be done in one operation. We need to split them into multiple operations (a.k.a. execution cycles). In the meanwhile, a LUT, which stands for a lookup table used to determine the output for a given input. The hardware design uses LUTs to implement the functionality. Higher LUTs utilization means higher power consumption and larger chip size. As a result, we try to achieve a trade-off between the execution cycles and the LUTs utilization.

B.3 Peer Algorithms and Parameter Settings

To validate the competitiveness of our proposed algorithm, we compare its performance with `HyperMapper` Nardi et al. [2019] and `ParEGO` Knowles [2006]. The prior one is a widely recognized DSE approach and the latter one is a popular surrogate-assisted evolutionary multi-objective optimization algorithm in the literature. We do not intend to delineate their working mechanisms

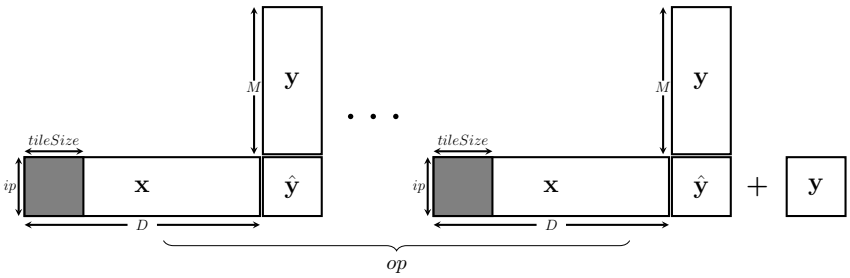

Figure 8: The parallelized SGD.

Table 4: The parameters and their range for the SGD case

variables	range
$tileSize$	4,8, $\cdots$, 124, 128
ip	1,2, $\cdots$, 15, 16
op	1,2, $\cdots$, 15, 16
p_1	True or False
p_2	True or False

Table 5: The parameters and their range for the K-Means case

variables	range
BN	96,192, $\cdots$, 960
ip	8,16, $\cdots$, 120, 128
op	2,4,8, $\cdots$, 64, 128
p_1	true,false
p_2	true,false
p_3	true,false

here while interested readers are referred to their original papers for details. The parameter settings are listed as follows.

- Number of function evaluations (FEs): The initial sample size is set to $11 \times n - 1$ for all algorithms and the maximum number of FEs is capped as 250 where n is the number of variables.
- Reproduction operators: The parameters associated with the simulated binary crossover Deb and Agrawal [1994] and polynomial mutation Deb and Goyal [1996] are set as $p_c = 1.0$, $\eta_c = 20$, $p_m = 1/n$, $\eta_m = 20$.
- Kriging models: As for the algorithms that use Kriging for surrogate modeling, the corresponding hyperparameters of the MATLAB Toolbox DACE Nielsen et al. [2002] is set to be within the range $[10^{-5}, 10^5]$.
- Batch size ξ: It is set as $\xi = 5$ for our proposed algorithms.
- Number of repeated runs: Each algorithm is independently run on each test problem for 31 times with different random seeds.