Chapter 4
Machine Learning-Based Experimental Design in Materials Science

Thaer M. Dieb and Koji Tsuda

Abstract In materials design and discovery processes, optimal experimental design (OED) algorithms are getting more popular. OED is often modeled as an optimization of a black-box function. In this chapter, we introduce two machine learning-based approaches for OED: Bayesian optimization (BO) and Monte Carlo tree search (MCTS). BO is based on a relatively complex machine learning model and has been proven effective in a number of materials design problems. MCTS is a simpler and more efficient approach that showed significant success in the computer Go game. We discuss existing OED applications in materials science and discuss future directions.

Keywords Materials design · Optimal experiment design · Machine learning

4.1 Introduction

Materials design and discovery is a fundamental issue in materials science and engineering. The design of composite material structure, that achieves certain quality metrics, is often the problem of selecting the optimal solution from a search space [1, 2]. Traditionally, this process depends on personal experience and expensive trial-and-error experiments. To accelerate this process, several optimal experimental design (OED) algorithms have been proposed aiming to reduce the number of required experiments [3–8]. Figure 4.1 illustrates the materials design process by an optimal experimental design approach. Given a space of candidates S, OED aims to

T. M. Dieb · K. Tsuda (✉)
National Institute for Materials Science, Tsukuba, Japan
e-mail: tsuda@k.u-tokyo.ac.jp

T. M. Dieb
e-mail: MOUSTAFADIEB.Thaer@nims.go.jp

T. M. Dieb · K. Tsuda
Graduate School of Frontier Sciences, The University of Tokyo, Kashiwa, Japan

K. Tsuda
Center for Advanced Intelligence Project, RIKEN, Tokyo, Japan

I. Tanaka (ed.), *Nanoinformatics*, https://doi.org/10.1007/978-981-10-7617-6_4

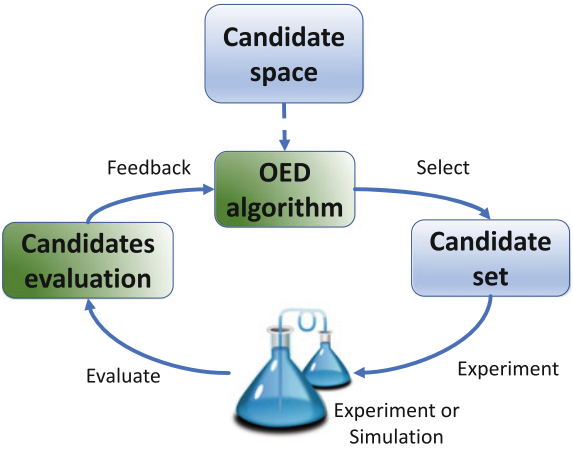

Fig. 4.1 Optimal experimental design (OED) algorithm process. For a predetermined number of iterations, OED algorithm selects a candidate set from the candidate space for experimentation. The experimental outcomes are then exploited for a better selection in the next iteration

find the best candidate that optimizes a black-box function $f(s)$, whose evaluation is possible only by an experiment. Starting from a random set of candidate solutions, an OED algorithm iteratively selects a set of candidate solutions for experiments. Experimental results are fed back to the OED algorithm to make further decisions. In many cases, experiments are replaced by simulators such as first-principle calculation.

In this chapter, we review the applications of two OED algorithms in the materials science domain. The first is Bayesian optimization (BO) [9], which has been proven effective in many materials design and discovery studies [1, 2, 6, 7, 10–13]. In BO methods, a machine learning model is employed to reconstruct the black-box function $f(s)$. In addition, the uncertainty of prediction is also taken into consideration in candidate selection. The second is Monte Carlo tree search (MCTS) that showed exceptional performance in computer Go [14]. MCTS explores a tree-shaped search space and is more efficient than BO in most cases. In a recent study [8], MCTS was applied to a Si-Ge alloy design problem and shown to be applicable to large-scale design problems.

This chapter is organized into four sections. Section 4.2 discusses the Bayesian optimization method and its applications in materials design and discovery, while Sect. 4.3 is dedicated to Monte Carlo tree search. Section 4.4 concludes this chapter with a brief look at other available OED approaches.

4.2 Bayesian Optimization

In machine learning communities, Bayesian Optimization (BO), aka kriging, has become a very popular tool for optimization problems recently [15–17]. BO is a sequential design strategy to optimize an expensive black-box function $f(s)$. Derivatives of f are not required. The difference between Bayesian optimization and earlier

models that used regression [18] is that, BO methods not only consider the predicted merit of candidates, but also quantify uncertainty as the predictive variance. Based on this variance, BO can determine where to query $f(s)$ next to achieve maximum performance. In this section, we will briefly describe a basic BO method, then review several applications in the domain of materials design and discovery.

4.2.1 Method

Assume that each candidate is represented using a set of N descriptors. The candidate set is then described as a set of points $S = \{s_1, ..s_m\}$ in an N-dimensional space. We are looking for the best point $s_{opt} \in S$ that maximizes a target black-box function $f(s)$. It is very common, particularly in materials science and engineering domain, that the cost of querying $f(s)$ is very high. It is necessary to find the optimal solution s_{opt} with as few queries as possible.

Bayesian optimization methods maintain a probabilistic model of $f(s)$, most commonly Gaussian process (GP) [19] (Fig. 4.2). Initially, a number of candidates are randomly selected and $f(s)$ is obtained for each of them. GP is trained using these data and the user obtains a nonlinear regression function and its predictive variance. In BO, an aquisition function quantifies how promising a candidate is, and depends both on the regression function and predictive variance. There are three typical choices: maximum probability of improvement, maximum expected improvement, and Thompson sampling [9]. The aquisition function is applied to all remaining candidates and the one with the largest value is selected for next experimentation.

The importance of uncertainty evaluation was investigated by Balachandran et al. [2]. They aimed to find the optimal design of M_2AX family of compounds, where the interest is focused on elastic properties [bulk (B), shear (G), and Young's (E) modulus]. Balachandran et al. compared BO with the selection with predicted values of support vector machines and showed that using uncertainty lead to better performance.

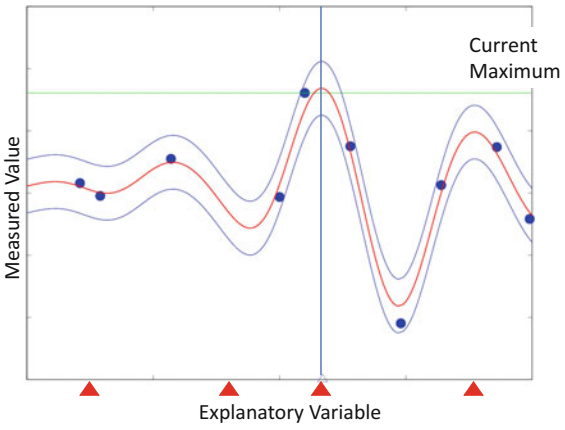

Fig. 4.2 Illustration of Bayesian optimization (BO). Gaussian process provides a regression function (red curve) and its variance (blue curves). Candidate points are shown as red triangles. The next candidate is selected based on an aquisition function

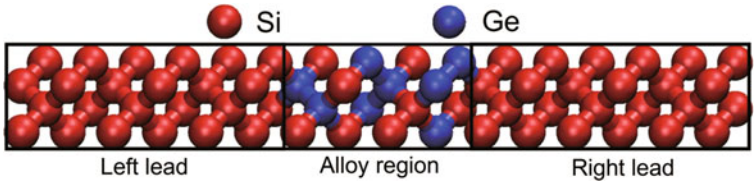

Fig. 4.3 Si-Ge interfacial structure between two Si leads. In this case, the interface region is made up of 16 atoms

4.2.2 COMBO: Bayesian Optimization Package

With the increasing popularity of applications of Bayesian optimization to materials design problems, there was a need to develop an efficient tool to support this process. We implemented an open source package for Bayesian optimization in python (COMBO: COMmon Bayesian Optimization library, https://github.com/tsudalab/combo) [11]. Thompson sampling, random feature maps and one-rank Cholesky update made it particularly suitable to handle large training datasets. It was shown that COMBO is more efficient than a GP implementation in scikit-learn (http://scikit-learn.org). To make it usable by non-experts, COMBO is parameter-free and can easily be used in various materials design problems. COMBO was first applied to optimize crystalline interface structures [10], where the aim is to find the best translation parameters with lowest grain boundary energy. It is reported that more than 50 times speedup was observed in comparison to random design.

4.2.3 Designing Phonon Transport Nanostructures

In a recent paper, Ju et al. [7] studied thermal conductivity in Si-Ge nanostructures. They applied COMBO to search for maximum and minimum interfacial thermal conductance (ITC) across all configurations of Silicon and Germanium (Fig. 4.3). Binary representation was used to describe the position of each atom in the structure: 1 and 0 represent the Ge and Si atom respectively. It is reported that the optimal solution was reached after exploring only 3.4% of the total number of candidates (12870).

4.3 Monte Carlo Tree Search

Large-scale problems are not rare cases in materials design and discovery. For example, finding the optimal configuration of two elements in a materials crystal structure with x sites involves exploring a search space with the size 2^x. When $x = 10$, the size

of the space is 1024. The space size increases exponentially with the number of sites x (for $x = 20$, the size becomes 1048576). Since BO applies an aquisition function to all candidates, the computational time becomes inhibitive for large x.

The significant success of Monte Carlo tree search (MCTS) [20] in computer Go game [14] inspired researchers to develop similar approaches in different research areas including other type of games [21–24]. MCTS is a guided-random best-first search method that models the search space as a gradually expanded tree. Additionally, MCTS does not involve costly matrix operation like GP, making it very scalable for large-scale search spaces. We recently applied MCTS to atom assignment problems in Fig. 4.3 and showed that MCTS is more efficient in BO in large-scale problems [8].

4.3.1 Method

Assume a material structure s with p positions. Each position has to be assigned by an atom from set A. We are looking for the best assignment of length p from the set of all possible assignments. The evaluation of a structure is given by a black-box function $f(s)$ corresponding to either an experiment or simulation.

MCTS uses a tree data structure to represent the search space (Fig. 4.4). A node at level n of the tree corresponds to the assignment of $a \in A$ into n-th position. The maximum depth of the tree is p. A solution is defined by a path from the root to a leaf node at level p. MCTS constructs only a top part of the search tree and it is expanded gradually to promising areas. At a node at depth $n < p$, only a part of the solution is obtained. To obtain a full solution, MCTS uses a technique called *rollout*, i.e., completing the solution by random assignment of atoms in the remaining positions. After a full solution is made, $f(s)$ is evaluated and recorded as the immediate merit of the node that the rollout started.

At the beginning, only the root node exists. The search continues until a prerequested number of iterations are finished. In each iteration, MCTS has four steps (Fig. 4.4): selection, expansion, simulation, and backpropagation. The pseudo-code of MCTS is shown as Algorithm 1. In the selection step, MCTS starts from the root and traverses down following the path of the most promising child. Children of the node are scored with different methods. The most common one is the Upper Confidence Bound (UCB) score [20],

$$ucb_i = \frac{z_i}{v_i} + C\sqrt{\frac{2\ln v_{parent}}{v_i}}, \tag{4.1}$$

where z_i is the accumulated merit of the node, i.e., the sum of immediate merits of the all downstream nodes, v_i is the visit count of the node, v_{parent} is the visit count of the parent node, and C is the constant to balance exploration and exploitation. In the expansion step, one or more child (depending on the implementation) are created

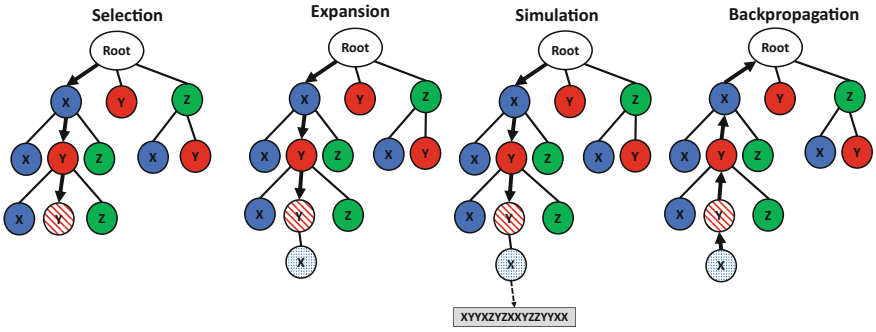

Fig. 4.4 Monte Carlo tree search (MCTS) for a three atom assignment problem. Atoms are to be assigned to a set of available positions. The search space is modeled as a decision tree where each node denotes a possible assignment. MCTS repeats four steps in each iteration: In the selection step, a promising leaf node is chosen by following the child with the best score. The expansion step adds a number of children nodes to the selected one. In simulation, a full solution is created by random rollout for each expanded node. The backpropagation step updates nodes' information along the path back to the root for a better selection in the next iteration

under the selected node. For each expanded child, a full solution is obtained through rollout, then evaluated using $f(s)$ and recorded in the simulation step. Finally, in the backpropagation step, the node information z_i, v_i is updated to be used for better selection in the next iteration.

4.3.2 MDTS: A Python Package for MCTS

We developed a python package of the MCTS algorithm that solves atom assignment problems [8]. The package named MDTS (Materials Design using Tree Search) is available at https://github.com/tsudalab/MDTS. MDTS is a parameter-free tool that automatically sets the only hyperparameter of MCTS algorithm (C) to obtain the best performance based on the target application. Following a similar idea to [25], MDTS controls C adaptively at each node as follows:

$$C = \frac{\sqrt{2}J}{4}(f_{max} - f_{min}), \tag{4.2}$$

where J is a meta-parameter initially set to one and increased whenever the algorithm encounters a so-called *dead-end* leaf to allow more exploration. f_{max} and f_{min} are the maximum and minimum immediate merits in downstream nodes.

To investigate the efficiency of MDTS, we compared the application of MDTS and an efficient Bayesian optimization package [11] to design optimal Silicon-Germanium (Si-Ge) alloy interfacial structures (Si:Ge = 1:1) in order to achieve both minimum and maximum thermal conductance [7]. The total computation time was

Start
make root node *root* ▷ Each node has 2 values, z: accumulated merit, v: visit count
solutions_set ← ∅
while within number of iterations **do**
$n \leftarrow$ SELECTION(*root*)
if n is not a maximum depth leaf **then**
children ← EXPANSION(n)
for all *child* ∈ *children* **do**
solution ← SIMULATION(*child*)
$e \leftarrow$ evaluate *solution* using experiment or computation
BACKPROPAGATION(*child*, e)
solutions_set ← [*solutions_set*, *solution*]
end for
end if
end while
return argmax(*solutions_set*)
Finish

function SELECTION(*node*)
if *node* has no children **then**
return *node*
else
bst_child ← $argmax(\frac{node.z}{node.v} + C\sqrt{\frac{2ln(parent.v)}{node.v}})$ ▷ *parent* is the parent of *node*
return SELECTION(*bst_child*)
end if
end function

function EXPANSION(*node*)
for all possible children **do**
make node *child*
add *child* to children of the *node*
end for
return all children of the *node*
end function

function SIMULATION(*node*)
structure ← the path from the root to *node*
if *node* is not a maximum depth leaf **then**
structure ← complete the solution randomly ▷ random rollout
end if
return *structure*
end function

function BACKPROPAGATION(*node*, e)
$node.z \leftarrow node.z + e$
$node.v \leftarrow node.v + 1$
if *parent* is not None **then** ▷ *parent* is the parent of *node*
return BACKPROPAGATION(*parent*, e)
end if
end function

Algorithm 1: Monte Carlo tree search

divided into design time and simulation time. The former is the time needed by the OED algorithm to select the next candidates, and the later is the time needed to query the target function $f(s)$, i.e., time to compute the thermal conductance for the candidate solution in this particular application. When the number of positions is smaller than 24, Bayesian optimization showed better efficiency due to its sophisticated machine learning algorithm. However, for larger problems, the design time of BO gets prohibitively long and MDTS was better in finding the best solution quickly.

4.3.3 Discussion

Use of the rollout is the basis of MCTS. It enables systematic space exploration without needing to generate the whole search space. In MDTS, the rollout is random, but it can possibly be improved using machine learning. For example, Yee et al. proposed a new MCTS algorithm with machine learning in continuous action spaces [26], where the UCB score is modified using kernel regression. It should be possible to apply this approach to materials science as well.

It is important to consider the balance between design time and simulation time. MCTS methods are most useful when the simulation time is short. The long design time of a more inefficient machine learning-based approach can appear less problematic when the simulation time is longer [8].

4.4 Concluding Remarks

Optimal experimental design (OED) methods are gaining more importance recently in the field of materials science and engineering due to popular need to reduce the cost of materials design and discovery. In this chapter, we presented two OED methods and their applications in materials design. Bayesian optimization (BO) is a well-established method with several successful applications; however, it struggles with large-scale problems. A new approach using Monte Carlo tree search (MCTS) has emerged with competitive search efficiency and superior scalability. In the future, a hybrid approach combining machine learning and MCTS may achieve even better design efficiency.

Other available OED methods include evolutionary algorithms such as genetic algorithms [27, 28]. Such methods are scalable, but they have many parameters to tune (such as crossover and mutation rates). With limited data available a priori, as in most cases in materials design and discovery, tuning parameters may be difficult. Other sequential learning (SL) methodologies have been proposed. For example Ling et al. have implemented a new OED approach based on random forests with uncertainty estimates [29]. The proposed framework is scalable to high-dimensional parameter spaces. Wang et al. proposed a nested-batch-mode sequential learning method that suggests experiments in batches [30]. In order to increase the efficiency of BO, some

researchers proposed a new surrogate model which combines independent Gaussian Processes with a linear model that encodes a tree-based dependency structure, which can transfer information between overlapping decision sequences [31]. In their approach, Jenatton et al. designed a specialized a two-step acquisition function that explores the search space more effectively.

Acknowledgements This work was supported by a Grant-in-Aid for Scientific Research on Innovative Areas 'Nano Informatics' (Grant No. 25106005) from the Japan Society for the Promotion of Science (JSPS).

References

1. A. Seko, A. Togo, H. Hayashi, K. Tsuda, L. Chaput, I. Tanaka, Phys. Rev. Lett. **115**, 205901 (2015)
2. P.V. Balachandran, D. Xue, J. Theiler, J. Hogden, T. Lookman, Sci. Rep. **6**, 19660 (2016)
3. D. Reker, S.G. Drug, Discov. Today **20**, 458 (2015)
4. A.R. Oganov, C.W. Glass, J. Chem. Phys. **124**, 244704 (2006)
5. M. Ahmadi, M. Vogt, P. Iyer, J. Bajorath, H. Frhlich, J. Chem. Inf. Model. **53**, 553 (2013)
6. A. Seko, T. Maekawa, K. Tsuda, T. I. Phys. Rev. B **89**, 054303 (2014)
7. S. Ju, T. Shiga, L. Feng, Z. Hou, K. Tsuda, J. Shiomi, Phys. Rev. X **7**, 021024 (2017)
8. T.M. Dieb, S. Ju, K. Yoshizoe, Z. Hou, J. Shiomi, K. Tsuda, Sci. Tech. Adv. Mater. **18**, 498 (2017)
9. J. Snoek, H. Larochelle, R. Adams, *Advances in Neural Information Processing Systems*, pp. 2951–2959, 2012
10. S. Kiyohara, H. Oda, K. Tsuda, T. Mizoguchi, Jpn. J. Appl. Phys. **55**, 045502 (2016)
11. T. Ueno, T. Rhone, Z. Hou, T. Mizoguchi, K. Tsuda, Mater. Discov. **4**, 18 (2016)
12. R. Aggarwal, M.J. Demkowicz, Y.M. Marzouk, Modelling Simul. Mater. Sci. Eng. **23**, 015009 (2015)
13. T. Lookman, F. Alexander, K. Rajan (eds.), *Information Science for Materials Discovery and Design, Springer Series in Materials Science*, vol. 225 (Springer International Publishing, Switzerland, 2016)
14. D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, E.A. Lanctot, M. Nature **529**, 484 (2016)
15. D.R. Jones, M. Schonlau, W.J. Welch, J. Glob. Optim. **13**, 455 (1998)
16. S. Streltsov, P. Vakili, J. Glob. Optim. **14**, 283 (1999)
17. M.J. Sasena, Flexibility and efficiency enhancement for constrained global design optimization with kriging approximations. Ph.D. thesis, University of Michigan, 2002
18. D. Coulinga, R. Bernotb, K.M. Dochertyb, J.K. Dixona, E.J. Maginn, Green Chem. **8**, 82 (2006)
19. C.E. Rasmussen, C.K.I. Williams (eds.), *Gaussian Processes for Machine Learning* (MIT Press, 2006)
20. C. Browne, E. Powley, D. Whitehouse, S. Lucas, P. Cowling, P. Rohlfshagen et al., IEEE Trans. Comput. Intell. AI Games **4**(1), 1 (2012)
21. B. Arneson, R.B. Hayward, P. Henderson, I.E.E.E. Trans, Comput. Intell. AI Games **2**, 251 (2010)
22. J. Mehat, T. Cazenave, I.E.E.E. Trans, Comput. Intell. AI Games **2**, 271 (2010)
23. A. Rimmel, F. Teytaud, T. Cazenave, Appl. Evol. Comput. 501–510 (2011)
24. C. Mansley, A. Weinstein, M.L. Littman, Int. Conf. Automat. Plan. Sched 335–338 (2011)
25. L. Kocsis, C. Szepesvári, *Machine Learning: ECML 2006* (Springer, Berlin, 2006), pp. 282–293

26. T. Yee, V. Lisy, M. Bowling, in *International Joint Conference on Artificial Intelligence*, pp. 690–696, 2016
27. Patra, T.K., Meenakshisundaram, V., Hung, J., Simmons, D, Comb, A.C.S. Sci. *19*(2), 96 (2017). https://doi.org/10.1021/acscombsci.6b00136
28. W. Paszkowicz, K.D. Harris, R.L. Johnston, Comput. Mater. Sci. **45**(1), ix (2009). https://doi.org/10.1016/j.commatsci.2008.07.008
29. J. Ling, M. Hutchinson, E. Antono, S. Paradiso, B. Meredig, Integr. Mater. Manuf. Innov. (2017). https://doi.org/10.1007/s40192-017-0098-z
30. Y. Wang, K.G. Reyes, K.A. Brown, C.A. Mirkin, W.B. Powell, SIAM J. Sci. Comput. **37**, B361 (2015)
31. R. Jenatton, C. Archambeau, J. Gonzalez, M. Seeger, in *International Conference on Machine Learning*, pp. 1655–1664, 2017