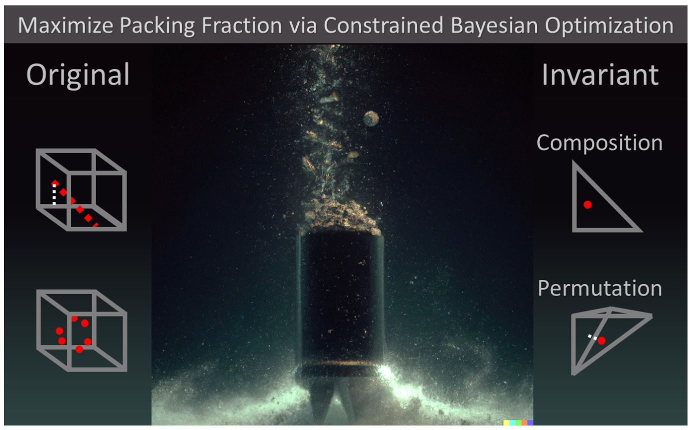
Compactness Matters: Improving Bayesian Optimization Efficiency of Materials Formulations through Invariant Search Spaces

Sterling G. Baird[a,*], Jason R. Hall[a,b], Taylor D. Sparks[a]

[a] *Department of Materials Science and Engineering, University of Utah, Salt Lake City, UT 84108, USA*
[b] *Northrop Grumman Innovation Systems, 9160 UT-83, Corinne, UT 84307*

Abstract

Would you rather search for a line inside a cube or a point inside a square? Physics-based simulations and wet-lab experiments often have symmetries (degeneracies) that allow reducing problem dimensionality or search space, but constraining these degeneracies is often unsupported or difficult to implement in many optimization packages, requiring additional time and expertise. So, are the possible improvements in efficiency worth the cost of implementation? We demonstrate that the compactness of a search space (to what extent and how degenerate solutions and non-solutions are removed) affects Bayesian optimization search efficiency. Here, we use the Adaptive Experimentation (Ax) Platform by Meta™ and a physics-based particle packing simulation with eight or nine tunable parameters, depending on the search space compactness. These parameters represent three truncated log-normal distributions of particle sizes which exhibit compositional-invariance and permutation-invariance characteristic of formulation problems (e.g., chemical formulas, composite materials, alloys). We assess a total of four search space types which range from none up to both constraint types imposed simultaneously. In general, the removal of degeneracy through problem reformulation (as seen by the optimizer's surrogate model) improves optimization efficiency. We recommend that optimization practitioners in the physical sciences carefully consider the trade-off between implementation cost and search efficiency before running expensive optimization campaigns.


Maximize Packing Fraction via Constrained Bayesian Optimization
Original
Invariant
Composition
Permutation

Keywords: constrained Bayesian optimization, constrained adaptive design, concurrency scheduler, Ax platform, particle packing fraction, machine learning invariance

1. Introduction

1.1. Background

Materials informatics tasks are characterized by small, sparse, noisy, multi-scale, heterogeneous, and high-dimensional datasets [1]. The search spaces associated with these tasks are often non-linearly correlated, discrete, and/or non-linearly constrained. Some representative examples are dopant concentration interactions, experimental instrument limitations, and adherence to chemical parsimony (i.e., the unlikelihood of finding materials with more than 5 or 6 elements present), respec-

*Corresponding author.

Email addresses: `sterling.baird@utah.edu` (Sterling G. Baird), `sparks@eng.utah.edu` (Taylor D. Sparks) tively. Due to small/expensive-to-sample datasets, Bayesian optimization (BO) is often chosen for materials discovery and process optimization problems [2–11] for its excellent search efficiency. BO is an adaptive design technique that involves leveraging prior belief about the solution to a problem and updating the belief in the context of new information. One of the greatest strengths of Bayesian models via e.g., Gaussian processes is the elegant trade-off between *exploiting* high-performance regions and *exploring* high-uncertainty regions through acquisition functions[1].

BO has been used to create and adaptively refine surrogate models for physics-based simulations whether acting directly as the surrogate model [2, 4, 5, 9, 12–20] or tuning hyperparameters of a surrogate model [3, 21]. Other examples include experimental discovery [7, 9, 11] and crystal structure prediction [22–25]. A review of Bayesian optimization applied to materials science in general is given in Kotthoff et al. [26] with a review of advanced Bayesian optimization methods applied to materials science given in Arróyave et al. [27].

In addition to BO, there are other global optimization algorithms, each with strengths and weaknesses. A non-exhaustive list of popular global optimization schemes, in order of typically increasing efficiency and computational complexity is given: manual tuning, grid search, random sampling, Sobol sampling, genetic algorithms, and BO. For inexpensive evaluations (hundreds of thousands of evaluations), random or Sobol sampling is typically preferred. For moderately expensive evaluations (tens of thousands of evaluations), genetic algorithms and scalable BO are typically preferred. Finally, for expensive-to-evaluate functions (hundreds to thousands of evaluations), BO is typically preferred. Exact BO scales poorly with dataset size, for which variational Gaussian Process models or scalable surrogate models may be used instead [28–31]. Likewise, for its straightforward implementation and low computational requirements, quasi-random sampling is preferred for large datasets. Grid-based searches in high dimensional spaces tend to be inefficient due to systematic sparse regions in the center of hyperboxes that make up the high-dimensional grid. Manual tuning by humans can often lead to local optimization and inefficient searches.

[1]"Acquisition functions are mathematical techniques that guide how the parameter space should be explored during Bayesian optimization. They use the predicted mean and predicted variance generated by the Gaussian process model" (`https://tune.tidymodels.org/articles/acquisition_functions.html`).

1.2. Related Work

Recently Liang et al. [32] benchmarked Bayesian optimization techniques for several materials science tasks. They raised awareness of the utility of anisotropic kernels over isotropic kernels[2]. They found that certain algorithms may perform well on certain tasks while performing poorly on others, highlighting the need for a careful task-based choice of models. In addition, they mentioned the computational advantages of random forest models relative to Gaussian processes despite being slightly less efficient overall.

Similar to Liang et al. [32], Hickman et al. [33] observed the effect of model choice and task on single- and multi-objective search efficiency, except with a constraint imposed. Hickman et al. [33] performed tests on analytical objective functions and emulators (i.e. models) trained on experimental data and demonstrated favorable performance of the `Gryffin` [34] and `Dragonfly` [30] optimization packages under constrained conditions.

It is well-known in the mathematical programming (optimization) community that problem formulations can introduce degeneracy or symmetry [35]. Moreover, alternate model formulations that break symmetry/degeneracy are essential for many integer programming (optimization) algorithms [36]. Much effort in the operations research community focuses on how to model problems to avoid degeneracy/symmetry and how these features can impact algorithm performance. Likewise, symmetries have also been addressed for traditional de-

[2]Incidentally, anisotropic kernels are used by default in this work and are a default of the Ax platform.

sign of experiments[3] [36] (e.g., using discrete search spaces with constraints imposed in advance). In a similar vein, we devote attention to avoiding degeneracy and symmetry in a materials-specific context.

1.3. Contributions

Here, we focus on a single adaptive design method and a single task (maximizing volume fraction of physics-based particle packing simulations) with up to two simultaneous constraints and instead seek to determine the effect of search space choices on efficiency. In this work, we pose the question:

> How does creating an irreducible representation for an adaptive design search space affect search efficiency for small search budgets and a high-variance particle packing objective function?

Solid rocket fuel propellants consist of several different types of particles (i.e. a formulation), where the size mean and standard deviation generally follow a log-normal distribution and are controlled by milling parameters and milling time, respectively. In particular, longer milling times tend towards lower standard deviations. High packing fractions are important when increased stability of solid rocket fuels is desired. Physics-based simulations are often used prior to experimental synthesis due to the energetic nature of the formulation constituents (in particular, ammonium perchlorate), made soberingly apparent in the PEPCON disaster in 1988, a chemical explosion that caused two fatalities, hundreds of injuries, and ∼$100 million worth of damage [38].

While necessary and useful, physics-based simulations are often expensive. In addition to increasing approximately with the square of the number of particles, computational runtime for a converged particle packing simulation can vary by orders of magnitude (e.g. 20 CPU min to 20+ CPU hours) as a function of frictional force computations which in turn depend on surface contact area. In general, an appropriate combination of small and large particles leads to additional surface contact area compared to homogeneous particle sizes and high packing fractions. Incidentally, the simulations which are most favorable in terms of high packing fractions are also the most expensive in terms of computational runtime. However, this is not mutually exclusive—computationally expensive simulations can also lead to undesired, low packing fractions. These points suggest the need for efficient optimization of the simulation search space.

In prior work [39], several iterations of adaptive design (also referred to as sequential learning) consisting of exploratory data analysis were followed by a classification-based approach. For the latter, rather than perform regression and take candidates with the best numerical predictions, solutions were classified based on their likelihood of being "extraordinary" [40], meaning falling in a top x% of all candidates in terms of performance. This resulted in 13 330 packing simulations and and a maximum packing fraction of 0.826. In this work, we instead focus on a small search budget with no pre-existing training data and carry out concurrency-limited[4] BO to maximize packing fraction using low-fidelity (noisy) packing simulations. These characteristics mimic common materials informatics datasets and tasks. We emphasize that the packing fractions reported in Hall et al. [39] should not be compared directly with the packing fractions reported in this work. This is due to significant differences in how the distributions were parameterized and translated into ParPack simulation input files. See Appendix C and Section S1 for discussion and other content related to the differences. Another significant difference arises from the use of far fewer number of particles in this work (25 000 instead of 1.5×10^6). See Section S3 for the convergence behavior of volume fraction vs. number of particles which shows an initial steep rise in the mean volume fraction. Additional information related to packing fractions as a function of the number of particles may be found in Vågberg et al. [41], Baranau and

[3]Cao et al. [37] addresses traditional design of experiments in a materials context.

[4]Concurrency refers to multiple processes occurring simultaneously without explicitly depending on each other.

Tallarek [42].

2. Methods

We seek to maximize particle packing fraction, f_{volFrac}, (Section 2.1) subject to any combination (including none) of up to two invariance constraint types totaling four search spaces. Each search space represents the same unique solutions but with varying levels of degeneracy/symmetry (Sections 2.1 and 2.2). The optimization problem is summarized as:

$$\begin{aligned} \max_{f_{\text{volFrac}}} \quad & f_{\text{volFrac}}(\tilde{X}, S, P) \\ & \tilde{x}_i \in \tilde{X}, s_i \in S, p_i \in P \\ \text{s.t.} \quad & p_1 + p_2 <= 1.0, p_3 = 1 - (p_1 + p_2) \\ & \text{(composition invariance constraint)}, \\ & s_1 < s_2 < s_3 \\ & \text{(permutation invariance constraint(s))}, \end{aligned} \tag{1}$$

where $\tilde{X}$ and S represent vectors of truncated log-normal size distribution parameters ($\tilde{x}_i$ and s_i) and P represents a vector of fractional prevalences of each of three particle types (p_i), totaling eight or nine degrees of freedom depending on the combination of constraints. We will refer to the compositional and permutation constraints as "comp" and "order", respectively. Log-normal distributions are computed as given in `scipy.stats.lognorm` with $\tilde{x} \equiv$ scale and $s \equiv$ shape (Appendix B). The truncated probability density function is given by:

$$\Pr(x,\tilde{x}_i, s_i) = \frac{e^{-\frac{\log^2\left(\frac{x}{\tilde{x}_i}\right)}{2s_i^2}}}{\sqrt{2\pi}xs_i} \tag{2}$$

where x, $\tilde{x}_i$, and s_i represent particle radius, log-normal median of the i-th particle type, and log-normal shape parameter of the i-th particle type, respectively.

Truncation is carried out by subjecting Eq. (2) to:

$$\frac{1}{\sqrt{m}}\tilde{x}_i \leq \text{x} \leq \sqrt{m}\tilde{x}_i \tag{3}$$

where x, $\tilde{x}_i$, and m represent particle radius, log-normal median of the i-th particle type, and max allowable ratio between any two particle sizes, respectively. In this work, m is fixed to the value of 16 to limit computational complexity.[5]

Converting to a discretized distribution is carried out by computing the percent point function (`scipy.stats.lognorm.ppf`) for particle radii at uniformly spaced quantiles within quantile bounds derived from Eq. (3) (see Section S1 for details). The final set of particle radii is then the sum of the three distributions weighted by the fractional prevalences:

$$\sum_{i=1}^{3} \text{Prd}\left(x_i, \tilde{x}_i, s_i, m\right) p_i \tag{4}$$

where $\text{Prd}(x_i, \tilde{x}_i, s_i, m)$ and p_i represent truncated discrete log-normal size distribution of the i-th particle type and fractional prevalence of the i-th particle type, respectively.

A summary of the 9 original simulation parameters and the bounds used in this work are given in Table 1. See Appendix B for additional details of the reparameterizations applied and constraints imposed in this work. A visual summary of these constraints and their corresponding degenerate search spaces are given in Figure 1.

We also describe our Bayesian optimization strategy in Section 2.3. Finally, we describe our validation setup involving running repeat simulations using the parameter combinations predicted as best for each search space in Section 2.4.

2.1. Particle Packing Simulations

The simulations involve dropping particles sampled from a predefined distribution of particle sizes inside of a cylinder at randomized locations [39]. Theoretical details of the particle packing simulations are given in Davis and Carter [43] and Webb and Davis [44], for which a summary is provided in the second paragraph of the motivation section in Hall et al. [39]. A proprietary Windows executable

[5] As discussed in Section 1.3, large ratios can lead to high packing fractions, yet cumbersome frictional force computations.

Table 1: Summary of 9 non-reparameterized simulation parameters and their bounds. $\tilde{x}_i$, s_i, and p_i correspond to log-normal median of particle radii, log-normal shape parameter, and fractional prevalence (i.e. composition) for each of the three particle distributions. For additional description, see Appendix B.

Name	Min	Max
$\tilde{x}_1$	1	5
$\tilde{x}_2$	1	5
$\tilde{x}_3$	1	5
s_1	0.1	1
s_2	0.1	1
s_3	0.1	1
p_1	0	1
p_2	0	1
p_3	0	1

for ParPack was used. While the executable is not made available, the functions and scripts provided at `https://github.com/sparks-baird/bayes-opt-particle-packing` can be adapted to other problems or used as a reference for custom implementations. Additionally, we describe qualitative differences between the representation of particle distributions in this work vs. prior work [39] in Appendix C.

2.2. Reducible and Irreducible Search Spaces

In this work, a reducible search space is a search space that exhibits identical solutions for different parameterizations that can be collapsed to a single solution and a single parameterization (i.e. an irreducible search space) through reparameterization or imposition of constraints. Baird et al. [45] found that mapping symmetrically related sets of parameters to an irreducible representation (i.e. a fundamental zone in crystallographic terms[6]) exhibited distinct advantages related to accuracy and computational efficiency of distance calculations.[7] Similar to the crystallographic representation, reducibility in this work focuses on leveraging domain knowledge about the relationship between input parameters of an otherwise "black-box" objective function to restrict the search space through reparameterization. Examples are a set of parameters that is represented as a composition or formulation (i.e. $Al_2O_3 \equiv 0.4Al + 0.6O$ where $0.4 + 0.6 = 1.0$) [46–57] (referred to as "comp") or a set of parameters that exhibits permutation invariance (e.g. $Al_2O_3 \equiv O_3Al_2$) [45, 47, 58] (referred to as "order"). When no additional parameter constraints other than lower and upper limits are used, we refer to this as "Bounds-only".

Other examples of formulation-type optimization problems that exhibit compositional and permutation invariances include:

- Hygiene products
- Dental composites
- Chemistry reactions
- Metal alloys
- Additive manufacturing resins
- Foods

Other types of constraints exist, such as rotational invariance for certain image processing tasks which have often been addressed via data augmentation [59]. Another example applicable to but not addressed in this study is a simulation that exhibits size invariance (e.g. unitless simulations) [60, 61]. This is not addressed due to the nonlinear transformation of the search space bounds that would be required to ensure the search space is not expanded or reduced relative to the other search spaces. The effect of distribution parameters on distribution shape in the context of size invariance is explored in Section S2.

[6]A fundamental zone in crystallography, which contains only one parameter combination out of a set of symmetrically related parameter combinations (e.g. crystal misorientation and/or grain boundary plane normal directions)

[7]The symmetry degeneracy is separate from the inclusion or exclusion of a degenerate dimension via rigid principal component analysis transformation which did not significantly impact model accuracy

We note that the reparameterizations and imposition of constraints in this work are separate from (usually) lossy dimensionality reduction techniques such as Uniform Manifold Approximation and Projection [62] or t-distributed stochastic neighbor embeddings [63] in that only redundant information is lost and that parameters retain domain-specific, interpretable meaning.

The `SearchSpace` objects used by our optimization package of choice, Meta's Adaptive Experimentation (Ax) Platform, corresponding to each of the four search spaces explored in this work are given in Section S8. More details about our usage of the Ax Platform are given in Section 2.3.

2.3. Adaptive Experimentation Platform and Ray-Tune

While many excellent packages for BO exist, we choose Meta's (formerly Facebook) Adaptive Experimentation (Ax) platform for "its relative ease-of-use, modularity, developer support, and model sophistication" [21] and refer to this as Ax. We refer readers interested in learning advanced optimization topics to the official Ax tutorials (`https://ax.dev/tutorials/`) and to a set of tutorials geared towards chemistry and materials science (`https://github.com/sparks-baird/self-driving-lab-demo/blob/main/notebooks/README.md`) via a minimal working example for autonomous scientific discovery [64].

In terms of optimization steps, 10 Sobol iterations precede 40 Bayesian optimization iterations. All Sobol iterations were required to be completed before moving on to the Bayesian optimization iterations. Alternatively, setting the number of Sobol iterations to the default of twice the number of parameters and/or reducing `min_observed_trials` (i.e. able to evaluate second step trials before completing first step) may have been appropriate choices which we do not expect to significantly impact the findings in this study.

In this work, we use a scheduler method (first in, first out) for the Bayesian optimization trials. A maximum of five workers were made available to the scheduler, and candidate generation had additional CPU RAM resources available.

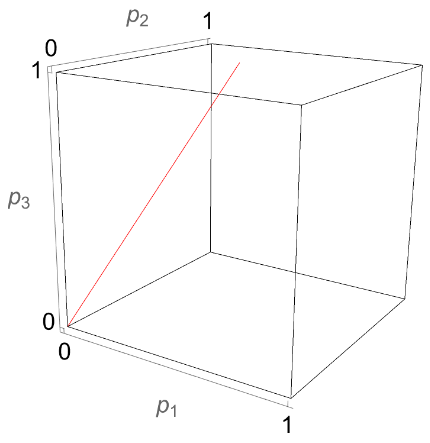
Because trial runtimes can vary between a few CPU minutes to over a CPU day as a function of the trial parameters, using a scheduler algorithm with multiple CPUs is likely more efficient in terms of clock time[8] than sequential optimization and batch optimization. Sequential optimization is a straightforward implementation where only one iteration runs at a time, and candidate generation for the next iteration does not occur until the results from the previous iteration are available. Batch optimization, by contrast allows for multiple trials to run in parallel and necessitates using batch conditioning[9]. Batch optimization is related to scheduler optimization in that multiple trials can run simultaneously, but is better suited for tasks where runtimes within a batch are approximately similar. This is because all trials within a batch have to complete before moving onto the next batch iteration which can result in poor utilization of the compute devices (e.g. CPU cores left in an idle state). A scheduler mitigates this issue by generating new candidates and assigning them to "workers" (i.e. CPU cores) as soon as one is available. During the generation step, all currently available data (including recently completed trials) is considered. The scheduler can be thought of as a manager that dynamically assigns tasks of varying difficulties to employees to maximize throughput.

More sophisticated scheduling algorithms also exist, for which we refer the reader to the Ray-Tune Trial Schedulers documentation. These types of scheduling algorithms can be applied to any combination of offline/online[10] and computational/experimental tasks, especially when there are multi-

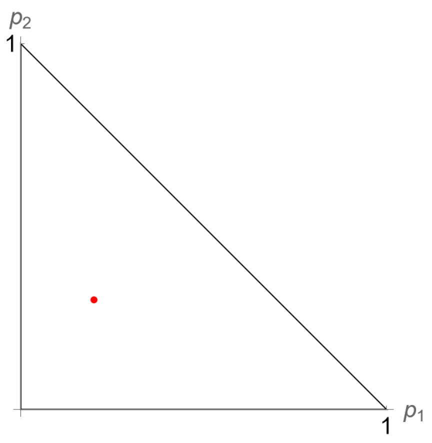
[8]Clock time is the real time between start and finish rather than the total CPU time used (possibly across multiple devices).

[9]Because joint acquisition is not always tractable, conditioning is often used such that later suggestions are conditioned on the predicted outcomes of earlier suggestions in the batch. See Appendix F.2 of Balandat et al. [65] for two types of conditioning: sequential greedy approaches and "fantasy" models. See also Wilson et al. [66].

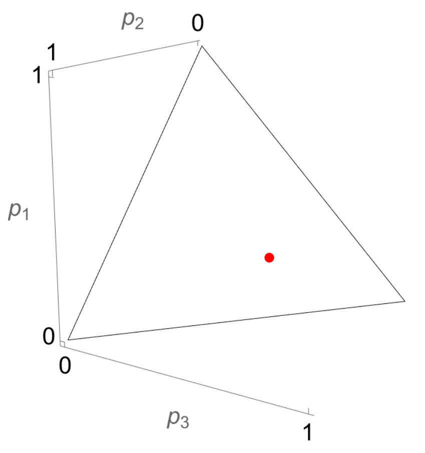
[10]Offline vs. online adaptive design can be thought of as whether or not a script needs to be restarted multiple times or is closed-loop where all iterations can be run to completion without exiting the script.



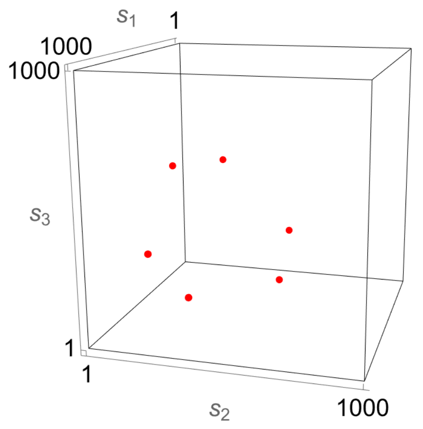
(a) Bound constraints only
(bounds-only)



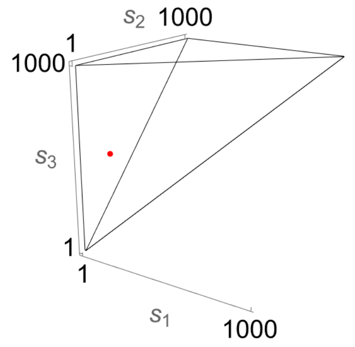
(b) $p_1 + p_2 \leq 1.0, p_3 = 1 - (p_1 + p_2)$
inequality composition (comp)



(c) $p_1 + p_2 + p_3 = 1.0$
equality composition (not implemented)



(d) Bound constraints only
(bounds-only)



(e) $s_1 < s_2 < s_3$
permutation (order)

Figure 1: Top row: composition degeneracy (Appendix B.1). Bottom row: permutation symmetry (Appendix B.2). Simple visualization examples of how imposing various types of constraints affects the solution space and search dimensions. Irreducible (b,c,e) and reducible (a,d) search spaces for compositional (b,c,a) and permutation (e,d) invariance constraints are given. Solutions are given in red, and the search space bounds are given in black. Applying a linear equality compositional constraint to a line solution in a cube (a) results in a point solution on a triangle embedded in a cube (c). This requires an additional rigid transformation to represent it in only two dimensions. Also in the middle row, reparameterizing the linear equality compositional constraint as a linear inequality constraint and imposing that on a line solution in a cube (a) results in a point solution in a triangle (b), albeit with some distortion introduced. Imposing two permutation-invariance constraints on a set of symmetric point solutions in a cube (d) reduces the search space to a smaller polyhedron and a single point solution (e). For additional description of parameters, see Table 1, Section 2, and Appendix B.

ple "workers" (e.g., CPUs, robots, experimentalists); however, the most straightforward and perhaps most ubiquitous application of scheduler algorithms is for online computational optimization tasks (e.g., simulations). Likewise, readers may be interested in the many state-of-the-art search algorithms supported via the RayTune interface.

2.4. Validation

Validation procedures carried out for the maximization of packing fraction via repeat evaluations of the objective function and a description of in-sample model predictions are given in Section 2.4.1. Cross-validation procedures used to assess model accuracy are described in Section 2.4.3.

2.4.1. Repeat observations

Each optimization campaign is repeated 10 times (each using a unique, fixed seed for the random number generator) with the fixed design budget and setup as described in Section 2.3. The best in-sample predictions[11] are validated by running the particle packing simulation for the best candidate 50 times.

We run repeats of the simulations using the best-predicted parameters because the the objective function is noisy for a low number of simulated particles. This validation of best in-sample predictions allows us to provide a fair comparison of the effect of each representation on search efficiency relative to each other.

2.4.2. In-sample predictions

Computing the in-sample predictions once all data has been collected for a given optimization campaign provides an alternative method to compare performance across different search spaces, especially when evaluating an objective function repeatedly is not feasible. To elaborate, a single observation is unlikely to be the same as the mean response if the same parameter set were tried multiple times. Rather than choose the sample with the best observed data, an alternative is to take a fitted model that is robust to noise and avoids overfitting, and of the samples that have been tested so far, choose the sample with the highest predicted (rather than observed) objective function value.

[11]In-sample predictions (meaning predictions for trials that completed) are used rather than the raw observed data due to noise in the latter.

This validation step is central to the findings of this study in determining the efficiency of search spaces. If the mean validated packing fraction for a given optimization campaign is higher than that of another optimization campaign, use of this validation approach bolsters our confidence that, on average, the former campaign is more efficient.

See Figure S7 for details related to the convergence behavior of the particle packing simulations as a function of particle size for a specific high-cost set of parameters.

2.4.3. Model Accuracy

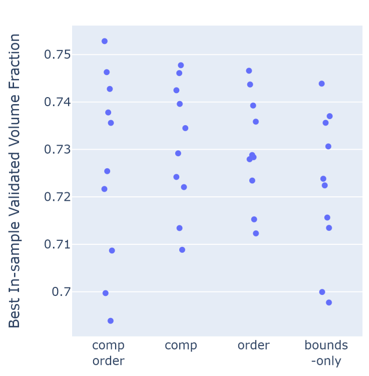
As an additional perspective, leave-one-out cross-validation (LOO-CV) [67], is performed for each final optimization dataset and visualized via parity plots (Figure S25). This is a technique where one datapoint is held out from a model for evaluating performance and the model is trained on the rest of the datapoints, and the process is repeated for every datapoint. For more information on LOO-CV, see `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.LeaveOneOut.html`. The average LOO-CV mean absolute error (MAE) is computed at the end of each optimization campaign and is the model accuracy referred to in Section 3.1.

3. Results and Discussion

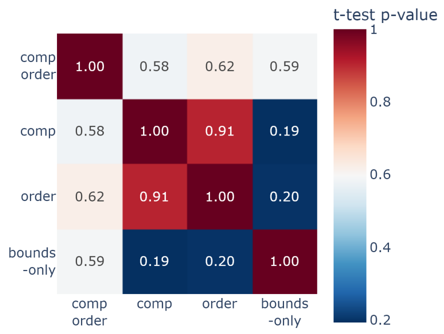
We present predicted and validated outcomes (Section 3.1) and interpretable model characteristics for the search spaces (Section 3.2).

3.1. Effect of Search Space Irreducibility on Efficiency

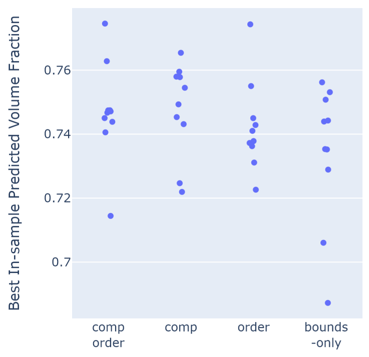
We summarize validated performance (Section 2.4.1), predicted performance (Section 2.4.2), and model accuracy (Section 2.4.3) in Figure 2. Figure 2a represents the situation where the best performing candidate (as determined by the surrogate model and restricted to the set of samples



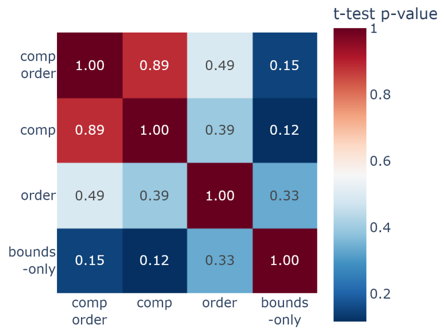
(a) Validation



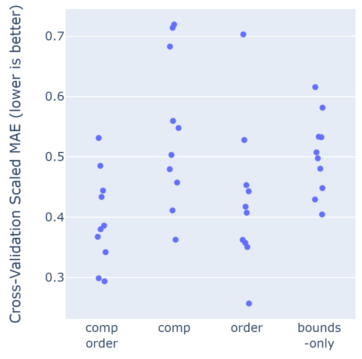
(b) Validation t-test



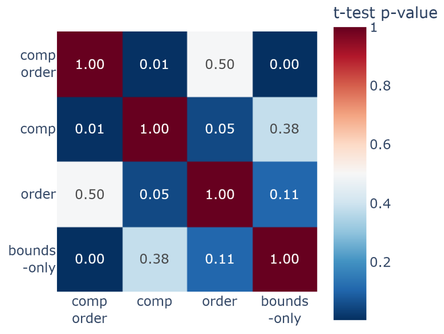
(c) In-sample predictions



(d) In-sample predictions t-test



(e) Cross-validation $\frac{\mathrm{MAE}_{\mathrm{GPEI}}}{\mathrm{MAE}_{\mathrm{dummy}}}$



(f) Cross-validation $\frac{\mathrm{MAE}_{\mathrm{GPEI}}}{\mathrm{MAE}_{\mathrm{dummy}}}$ t-test

Figure 2: Packing fraction box plots and raw data of (a) validated (using 50 repeat runs) and (c) best in-sample predicted results for each of the four search space types and 10 seeded optimization campaigns, sorted by decreasing mean packing fraction. (e) LOO-CV scaled MAE, where MAE is normalized by MAE of a "dummy model" MAE (predictions are the mean of the observed values). Pairwise t-test p-values for (b) validated, (d) best in-sample predicted results, and (f) LOO-CV scaled MAE. Lower p-values indicate a higher statistical significance that the means of the two distributions being compared are different from each other.

Table 2: Validation particle packing fraction mean and standard deviation (std) across 10 campaigns for each of the four search spaces.

type	mean	std
comp-order	0.726	0.02
comp	0.731	0.014
order	0.73	0.011
bounds-only	0.722	0.015

Table 3: Best in-sample predicted particle packing fraction mean and standard deviation (std) across 10 campaigns for each of the four search spaces.

type	mean	std
comp-order	0.747	0.015
comp	0.748	0.015
order	0.742	0.014
bounds-only	0.734	0.022

that were actually probed) is run again on the objective function using 50 repeats (see Section 2.4). Since each campaign was run 10 times, 10 "best" samples each had 50 validation repeats, resulting in 500 total evaluations of the objective function per search space. First the mean within each campaign is taken (i.e., mean of 50 repeats) and then box plots and raw data for the 10 means are reported in the figure. On the other hand, Figure 3 represents the raw observed data across each of the campaigns. There are 50 iterations in each campaign and 10 campaigns, so each validation of a search space includes 500 evaluations of the objective function. Likewise, Figure 2c represents the best in-sample predictions (i.e., the sample predicted to be best after all 50 iterations have completed) averaged across each of the 10 campaigns. The distinction between Figure 2c and Figure 3 is that Figure 2c shows *model predictions* at the *completion* of the optimization campaign whereas Figure 3 shows *raw data* as the optimization progresses. Finally, Figure 2e shows *scaled model error*.

In general, the constrained search spaces tend to have better performance than the bounds-only search space. For example, the "comp-order", "comp", and "order" validated (Table 2 and Figure 2a) and best in-sample results (Table 3 and Figure 2c) exhibited higher mean packing fractions than bounds-only. In the case of validation results for "comp" and "order", the difference in distribution mean relative to bounds-only had t-test p-values of 0.19 and 0.20 (Figure 2b). We interpret this to mean that the true means for "comp" and "order" are likely to be greater than the bounds-only case. However, the p-value for "comp-order" was much higher (0.59), indicating less certainty that the means of the two distributions are unequal. Likewise, "comp-order" and "order" exhibited lower model error than bounds-only, though "comp" typically had a larger model error. This indicates that the model has a more difficult time accurately predicting objective function values when permutation symmetries are left unreduced; however, it is interesting that the same behavior is not reflected in the optimization performance.

Table 4: LOO-CV MAE mean and standard deviation (std) across 10 campaigns for each of the four search spaces.

type	mean	std
comp-order	0.396	0.077
comp	0.544	0.126
order	0.428	0.121
bounds-only	0.503	0.066

The "comp-order", "comp", and "order" search spaces typically exhibited differences in the mean relative to bounds-only with a higher statistical significance for the best in-sample predictions (Figure 2d) compared with the validation results (Figure 2b). The statistical significance was also greater (lower p-value) for the model error (Figure 2f) relative to the validation results (Figure 2b).

While these results show that in general it is better to remove degeneracies, we think it is likely that an explicit treatment of the composition constraint as a linear equality constraint (rather than a linear inequality constraint) (Appendix B.1) and treatment of the permutation constraint through data augmentation or a permutation-invariant kernel (Appendix B.2) will reveal an even more pronounced superiority over the bounds-only case.

In situations where the validation is much more expensive or infeasible to perform, optimization practitioners must carefully consider whether they can "trust" the best in-sample predictions when the objective function is subject to large levels of noise. We think it is likely that in a reduced-noise environment, the predicted and validated outcomes will be better aligned.

Given the use of a low-budget, high-dimensional Bayesian optimization task, it may be reasonable to assume that the points sampled during each optimization campaign tend more towards exploration rather than exploitation. However, for datasets that are highly concentrated (e.g. due to local optimization or exploitation search behavior), the model accuracy may appear unrealistically high. In an extreme case, if the same parameterization is sampled at every iteration, the cross-validated accuracy will only be limited by the stochasticity of the objective function. Thus, use of LOO-CV scaled MAE is better suited for comparing results where exploration is present and the choice of hyperparameters does not cause significant clustering of sampled points. To circumvent this, an alternative cross-validation scheme may be used where domain-informed clusters or groups are used to define cross-validation splits [68–70]. The direct analogue to LOO-CV is leave-one-group-out cross-validation (also referred to as leave-one-cluster-out cross-validation), which is used to "[prevent] overly optimistic extrapolative predictive performance" [71].

While performing validation via repeated runs of the same parameters is the most robust of the three sets of summarized results (Figure 2) in terms of comparing different search spaces, it is also the most expensive because it requires additional objective function evaluations. When performing validation through repeated runs of best in-sample predictions for objective functions with large variance is not feasible during the optimization design phase, we encourage optimization practitioners to use multiple "notions-of-best" to assess superiority of one model configuration vs. another. While there are differences between the ranking trends of best in-sample predictions (Figure 2c) and the LOO-CV scaled errors (Figure 2e), the search space with only order constraints ("order") performs well in both of these cases. Interestingly, this search space is also the one that performs the best in the high-fidelity notion of best, the validated volume fractions.

As mentioned previously in this section, we think it is likely that use of data augmentation will result in better performance than simply removing mirror images; implementation difficulties and limitations of a data augmentation approach are discussed in Appendix B.2.

Individual optimization results for each of the seeded runs is given in Section S5.

3.2. *Interpretable Model Characteristics*

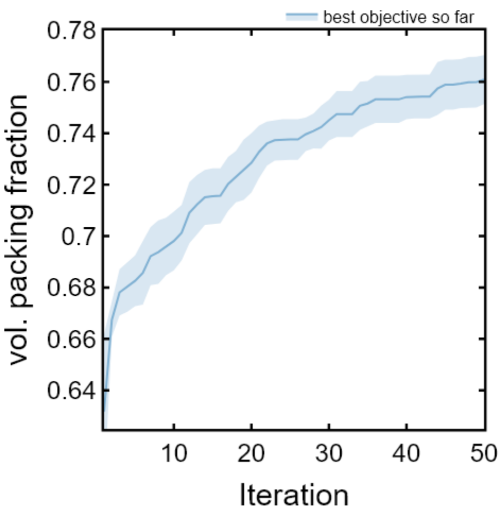
In this subsection, we probe some interpretable characteristics of the `GPEI` model through feature importances (Section 3.2.1) and 2D contours for two of the compositional variables (Section 3.2.2). Particle size distribution visualizations are given in Section S3.1) and LOO-CV results are provided in Section S6. Plots with additional information for particle size distribution visualizations, feature importances, and 2D contours are given in Sections S1, S4 and S7, respectively.

3.2.1. *Feature Importances*

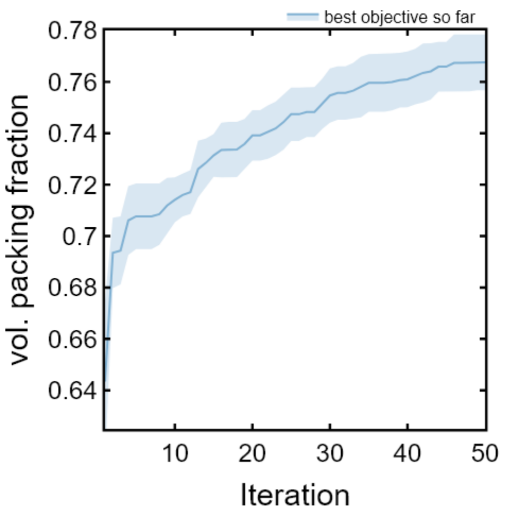
Average feature importances[12] across 10 seeded runs are given in Figure 4. Note that each search space is characterized by different sets of features with eight or nine total features.

One of the characteristics that stands out is the large standard deviations for many of the features. In other words, separate optimization runs did not necessarily assign the same features as being most important, and this behavior was observed across many search spaces. The large variances observed for many of the search spaces indicate that no one parameter is universally important for the simulations. This can be explained by the representation of parameters as fractional contributions of distributions. Since we know that the physics-based simulation takes an intermediate representation of the combined, discrete particle size distribution (Eq. (4)), and each parameter affects this final

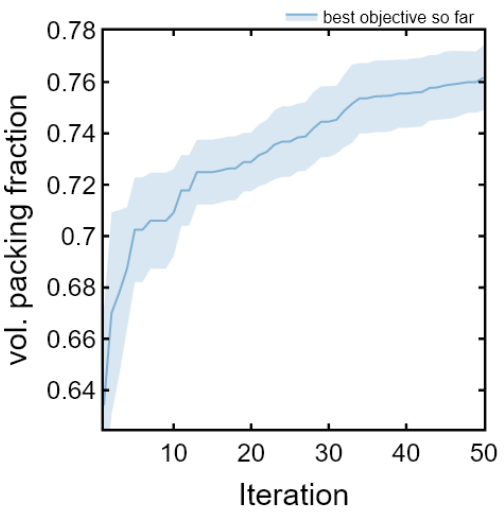
[12]Feature importances are based on based on the inverse lengthscales of an anisotropic Gaussian kernel.



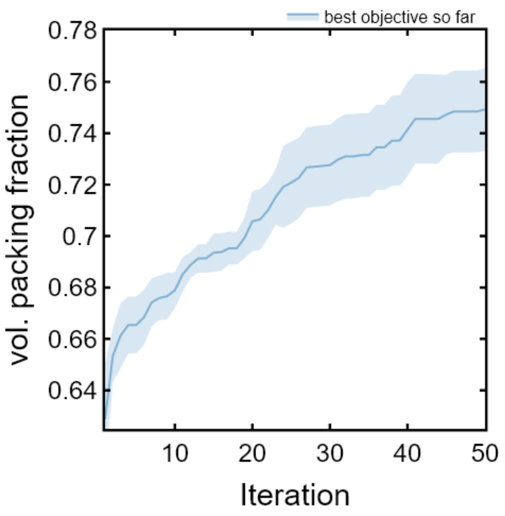
(a) Bounds-only



(b) Composition (comp)



(c) Permutation (order)



(d) Composition (comp) and permutation (order)

Figure 3: Average best observed objective vs. iteration for each of the four search spaces using Gaussian process expected improvement (GPEI). The individual observations are subject to noise. Standard deviations for each of the 10 campaigns are plotted as colored bands.

distribution, there are strong dependencies between parameters. Additionally, in the cases where the permutation constraint is not imposed, the model is just as likely to assign one set of distribution parameters as important compared with another.

We would have expected that for search spaces with the best predicted and validated outcomes, the feature importances would have tighter standard deviations than others; however, this does not appear to be the case. This may be reflective of the interactions of the problem formulations with the mechanics of the physics-based particle packing simulation, where the parameters are highly interdependent.

Individual feature importances based on the fitted, inverse lengthscales of the anisotropic Gaussian kernels for for seeds 10, 11, 12, 13, and 14 are given in Section S4.

3.2.2. 2D Contours through Parameter Space

2D contour plots of `comp2` (y-axis) vs. `comp1` (x-axis) for each of the four search spaces (rows) and each of the first five (out of a total of 10) seeded optimization runs are given in Figure 5. Only the first five are shown for brevity. The "comp-order" search space exhibits the greatest homogeneity between the five repeat campaigns compared with the other three search spaces ("comp", "order", and bounds-only), perhaps owing to the fact that it is the most compact representation of the four search spaces.

2D contours through parameter space with additional features such as estimated standard deviation error for seeds 10, 11, 12, 13, and 14 are given in Section S7.

4. Future Work

While in general, more compact design spaces tend to be more efficient, this work reveals an example where the intuitive design spaces choices do not always correspond to the most efficient design spaces. We surmise that this may also be the case in similar high-dimensional tasks involving complex physics-based simulations subject to large amounts of noise. In this section, we provide further topics of study that can help to elucidate the root causes of the phenomena described in this work.

We plan to address the following topics in future work:

1. Are these results corroborated by alternative, open-source particle packing simulations? [72–75]

2. To what extent can multi-fidelity optimization reduce total search cost? [27, 30, 75–80]

In addition to the two above, a number of questions may be interesting to explore in future work:

- What is the effect of irreducibility for high-dimensional optimization (i.e. 20+ parameters)? [6]

- In a reduced noise environment (i.e. better convergence through a larger number of dropped simulation particles), do the interpretable model characteristics follow more consistent trends across repeated campaigns?

- Do the results generalize to optimization of model accuracy (i.e. without regard to high-performance)? (e.g. via Negative Integrated Posterior Variance acquisition function[13])

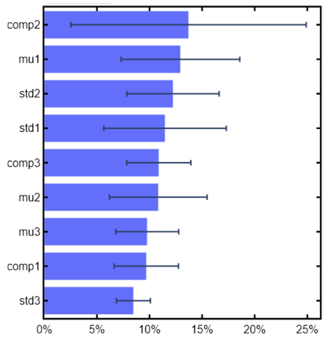
- Do the results generalize to wetlab experiments (as opposed to physics-based simulation)?

- How do these findings compare to other optimization algorithms (e.g. random search, genetic algorithms, random forest based BO [81])?

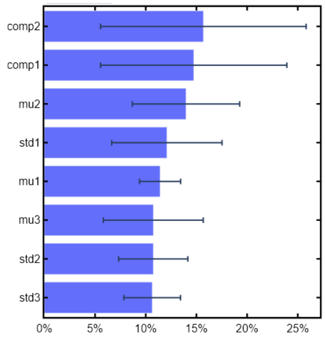
- Do larger datasets follow the same trend?

- Is there a significant difference in search efficiency when using a predefined list of candidates (i.e. supply candidates sampled from an irreducible search space)?

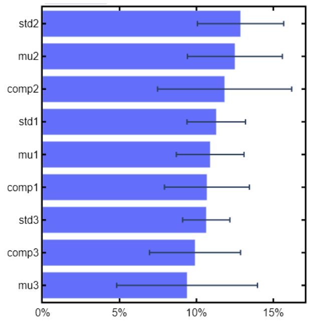
[13]For Negative Integrated Posterior Variance acquisition function usage in `Ax`, see `https://github.com/facebook/Ax/issues/930`



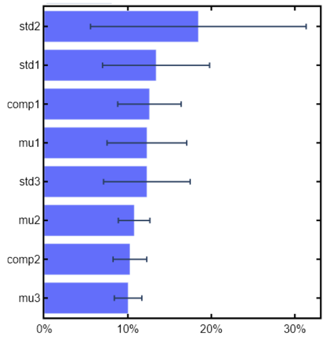
(a) Bounds-only



(b) Composition (comp)



(c) Permutation (order)



(d) Composition (comp) and permutation (order)

Figure 4: Average feature importances for the four search spaces across 10 seeded optimization runs using GPEI with standard deviations as error bars.

- How does search space reducibility scale to multi-objective problems?
- Does replacing the Bayesian model with sparse axis-aligned subspaces Bayesian optimization (SAASBO) perform better on average and/or change the ranking results?
- Could using a heteroskedastic noise assumption [82] improve the performance of the size-reparameterized search space?[14]
- Could regularization terms or optimization algorithms geared towards ill-posed inverse problems mitigate degeneracy issues?
- Do results hold for objective functions (other than packing fraction) that use the same underlying physical system? (e.g., average coordination numbers, standard deviation of Voronoi volumes around particles, asymptotic α-relaxation times [83])

5. Conclusion

The removal of degeneracies for a physics-based particle packing simulation tended to improve performance as assessed by three different judging criteria: best in-sample predictions, repeated evaluations of the best suggested parameter set, and cross-validated model accuracy. Simultaneous use of multiple judging criteria may be more robust when making decisions related to problem representation for optimization tasks. The methods and analysis applied in this work are applicable to mate-

[14]The objective function may be subject to heteroskedastic noise. In the limit of infinite particles, it seems likely that the variance of repeat sampling from the objective function for a fixed set of parameters will tend towards zero. Treatment of heteroskedastic noise is a promising avenue for future work, since heteroskedastic noise may be (at least one) reason for the surprising behavior for a few of the search space combinations. One way this can be implemented is by specifying the standard deviation (SEM in Ax terms) of repeat observations directly or using a model that infers heteroskedastic noise (see e.g., `https://github.com/facebook/Ax/issues/244` and `https://github.com/Ryan-Rhys/Heteroscedastic-BO`)

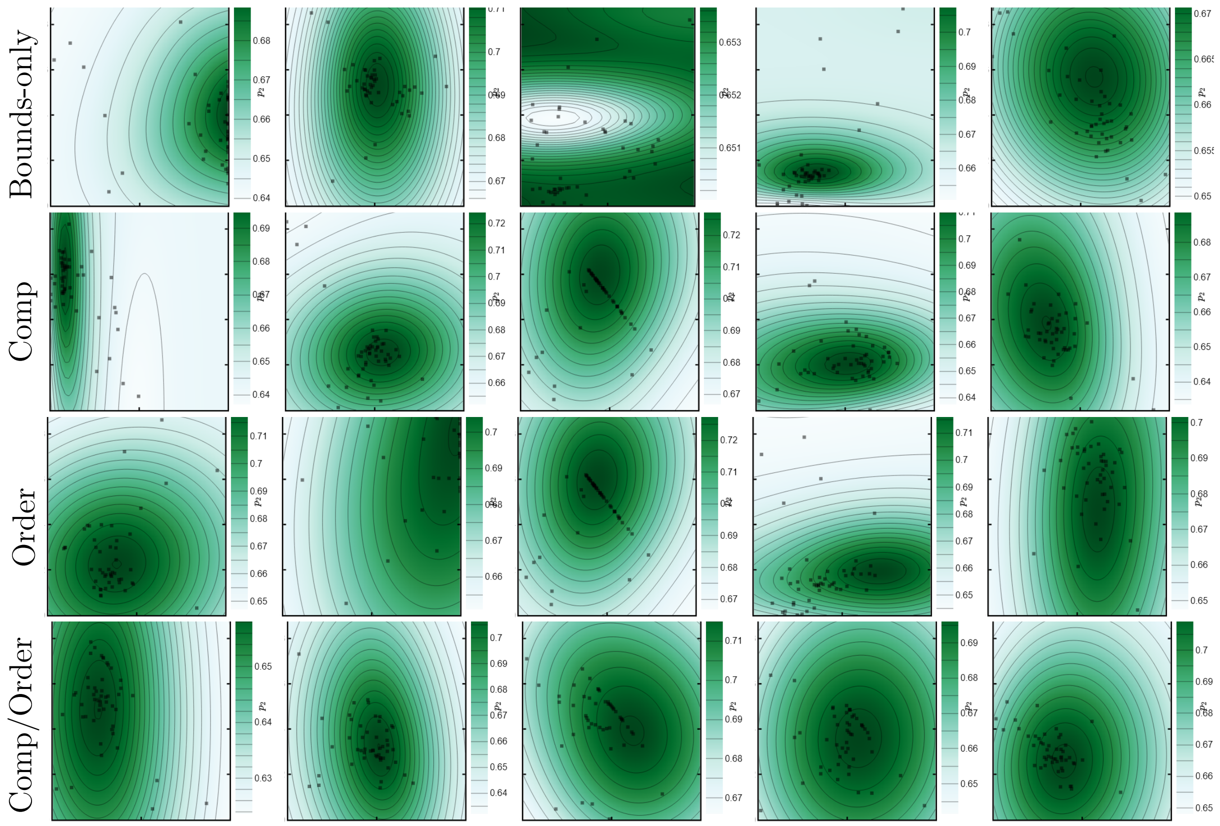


Figure 5: 2D contour plots of `comp2` (y-axis) vs. `comp1` (x-axis) for each of the four search spaces (rows) and the first five seeded optimization runs using GPEI.

rials science formulation problems such as composite materials, alloys, and composition-based materials discovery. We caution optimization practitioners in the physical sciences to carefully assess the influence of linear and non-linear constraints or reparameterizations on their search spaces, especially when expensive physics-based simulations or wet-lab experiments and noisy objectives are involved. Pairing efficient search spaces with state-of-the-art optimization algorithms can improve optimization success relative to more standard approaches.

Appendix A Bayesian Optimization Methods

In prior work [21], a high-dimensional scheme named SAASBO was used to optimize 23 hyperparameters within a design budget of 100 iterations and demonstrated superior performance over a more traditional (default) Bayesian optimization approach. Here, we use the default BO model, GPEI to limit the computational expense.[15]

We refer to the GPEI implementation within the Ax platform as `GPEI`.[16] As the name suggests, GPEI uses a Gaussian process surrogate model in conjunction with the expected improvement acquisition function. Gaussian process regression is a non-parametric Bayesian regression method that can be thought of as fitting an infinite-dimensional multivariate normal distribution to observed data. The expected improvement acquisition function consists in selecting the next point(s) to evaluate in a way that manages the trade-off between exploitation of candidates with high predicted performance and exploration of regions with high uncertainty. In the `GPEI` model, a Matern 5/2 kernel is used by default which allows for somewhat less smooth behavior than the radial basis function. Parameters are mapped to a range between 0 and 1 and the objective values are standardized (subtract mean, divide by standard deviation) per default transformation behavior within Ax, and parameter constraints are imposed as hard constraints. With BoTorch as the backend for Ax, Ax leverages auto-differentiation to perform gradient-enhanced optimization of acquisition functions. Finally, maximum a-posteriori estimation is used on the marginal log likelihood during acquisition function optimization. For additional details, please see Balandat et al. [65].

In this work, we have chosen to represent the parameters as continuous variables rather than evaluating the acquisition function on a predefined list of candidates. Some optimization packages preferentially treat search spaces as fully discrete whereas others preferentially treat search spaces as continuous. The Ax Platform is designed and configured in a way that favors optimization of continuous spaces. To optimize over a predefined list of candidates in the Ax Platform requires digging into lower level components in order to evaluate the acquisition function directly (see e.g., https://github.com/facebook/Ax/issues/771). In addition to requiring more user expertise, evaluating quasi-random predefined candidates is typically less efficient than relying on internal maximization of the acquisition function. In cases where the problem is high-dimensional, the number of predefined points required to effectively sample the acquisition function may be prohibitive compared to the continuous case. This is not to suggest that evaluating acquisition functions over predefined points is undesirable, but rather that there are non-negligible implementation and computational costs associated with it. In the context of this work, this discussion applies to invariances that do not involve a change of variables (i.e., a linear equality constraint. See discussion in Appendix B.1).

[15]SAASBO is computationally expensive especially for larger design budgets. Normally, this would be fine for expensive simulations and experiments as well as comprehensive benchmarking (this work); however, we are limited to using a Windows executable to run the simulations with no set timeline for a corresponding Linux executable. Running parallelized repeat campaigns using University of Utah's Center for High-Performance Computing (CHPC) resources is straightforward using Linux software, but we are subject to additional constraints (i.e. fewer resources) when using Windows. Thus, a full campaign running $10 \times 90 \times 4 = 7200$ SAASBO iterations might require several months of usage on CHPC's Beehive (Windows) machine compared with a few weeks of usage using GPEI iterations.

[16]Generally, when referring to theory, we refer to the full name or abbreviation, and when referring to the model as implemented within Ax we use code formatting.

Appendix B Reparameterizations and Constraints

B.1 Compositional Constraint

The linear equality compositional constraint is given in Eq. (5):

$$\sum_{i=1}^{n} p_i{=}1 \tag{5}$$

where p_i and n represent fractional prevalence of the i-th particle type and number of particles, respectively.

However, linear equality constraints are not directly supported by most optimization packages[17] due to the difficulty of sampling from slices in higher-dimensional spaces (e.g. a triangle embedded in three dimensions, where a triangle naturally has zero volume). A straightforward solution is to reparameterize a linear equality constraint, albeit with some distortion of the original search space, as a linear inequality constraint [77] as Eq. (6):

$$\sum_{i=1}^{n-1} p_i{\le}1 \tag{6}$$

where p_i and n represent fractional prevalence of the i-th particle type and number of particles, respectively.

This is subject to the additional constraint that Eq. (7):

$$p_n{=}1\text{-}\sum_{i=1}^{n-1} p_i \tag{7}$$

where p_n, p_i, and n represent fractional prevalence of the n-th particle type, fractional prevalence of the i-th particle type, and number of particles, respectively.

A user needs to decide if applying the linear equality constraint through e.g., a predefined search space, is worth the implementation and computational cost given their optimization package of choice (whether this is Ax, another package, or a custom implementation). Other solutions such as Lagrange multipliers may also be used to implement linear equality constrained Bayesian optimization [84].

[17]Supporting linear equality constraints is on Meta's Adaptive Experimentation Platform wishlist. Linear equality constraints are, however, supported on the Adaptive Experimentation dependency, BoTorch, via proper sampling from a d-simplex. See https://github.com/facebook/Ax/issues/903 for additional context.

B.2 Permutation Invariance

An example of permutation invariance is given in Eq. (8):

$$\begin{pmatrix} f_{\text{volFrac}}\left[\tilde{x}_1,\tilde{x}_2,\tilde{x}_3,s_1,s_2,s_3,p_1,p_2,p_3\right]= \\ f_{\text{volFrac}}\left[\tilde{x}_2,\tilde{x}_1,\tilde{x}_3,s_2,s_1,s_3,p_2,p_1,p_3\right] \end{pmatrix} \tag{8}$$

where $\tilde{x}$, s, p, and $f_{\text{volFrac}}[\cdot]$ represent log-normal median, log-normal shape parameter, fractional prevalence, and volume fraction function/simulation, respectively.

One option to address the degeneracy here is to impose an order constraint Eq. (9):

$$s_i \le s_{i+1}\ \forall\ i \in \{i{,}n\text{-}1\} \tag{9}$$

where s_i and n represent log-normal shape parameter of the i-th particle type and number of particles, respectively.

An alternative, though not particularly amenable to BO (at least when data scaling is an issue) and in general intractable when the number of permutations is large, is to perform data augmentation in the original search space by including the repeat permutation data at "no additional cost". Additionally, we did not choose to perform data augmentation due to the difficulty of simultaneously implementing both reparameterizations/constraints within an `AxSearch` first in, first out scheduler framework.

The practical implementation of data augmentation for this work's optimization task is not straightforward and is of limited use in terms of combinatorial explosion when many variables are involved as well as data scaling limitations. This results in a high-cost scenario for possibly murky results and applicability to a more limited range of tasks (i.e. small data, few variables in the permutation constraint). Thus, we choose to focus on order constraints in this work. However, the effect of using data augmentation vs. order constraints on search space efficiency in combination with other constraints may be an interesting topic for future study.

Appendix C Differences Between This Work and Prior Work (Particle Size Distributions)

We note that the datasets in Hall et al. [39] used a positive linear relationship between mass fraction and particle size, which is opposite to what is described in Fig. 2 of Hall et al. [39] due to an error in how the distributions were processed in internal scripts. We formalize the representation of particle size distributions in this work as truncated, log-normal distributions. We emphasize that there is little to no correspondence between the parameters reported in this work and that of Hall et al. [39] due to the differences in distribution sampling. For a comparison of Hall et al. [39] vs. this work, see Figure S1 and Figure S2, respectively. To avoid any ambiguity, we define our parameters as $\tilde{x}$ (`scale` in) and s (`shape`) as used within `scipy.stats.lognormal` via e.g.:

```
lognorm.pdf(x, shape, scale=scale)
```

Infinite random sampling from the log-normal distribution as defined by $\tilde{x}$ results in an empirical distribution whose median is equal to $\tilde{x}$.

We provide representative examples of the truncated distributions sampled in this work in Figure S2, as well as log-normal distributions derived from grid-sampled parameter combinations Figures S3 and S4 based on the search bounds in Table 1 to give a better sense of distributions sampled in this work. Additionally, we demonstrate the convergence behavior of volume fraction as a function of number of particles per simulation in Section S3. There is a moderate run-to-run variation for simulations using this distribution and 25 000 particles Figure S7.

Glossary

BO Bayesian optimization 2, 3, 6, 13, 16, 17

GPEI Gaussian process expected improvement 12, 14–16

LOO-CV leave-one-out cross-validation 8–11

MAE mean absolute error 8–11

SAASBO sparse axis-aligned subspaces Bayesian optimization 14, 16

Conflicts of Interest

There are no conflicts of interest to declare.

Acknowledgement

We thank Max Balandat, Lena Kashtelyan, David Eriksson, and Bernard Beckermann for discussions related to use of the Ax platform and implementation of constraints. We thank Victoria K. Baird for discussions related to solid rocket fuels. We thank Ramsey Issa for checking the manuscript. Plots were produced via Plotly [85] and Ax's plotting wrapper functions. Several tables were formatted via an online formatter [86] and the auto-paper methodology [87] was used. Figures 1 and S3–S6 were produced via Mathematica. This work was supported by the National Science Foundation under Grant No. DMR-1651668.

Authors' Contributions

Taylor D. Sparks: Supervision, Project administration, Funding acquisition, Resources, Writing - Review & Editing. **Sterling G. Baird**: Supervision, Project administration, Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing - Original Draft, Writing - Review & Editing, Visualization. **Jason R. Hall**: Project administration, Software, Validation, Investigation, Writing - Original Draft, Writing - Review & Editing

Availability of Data and Materials

There is no external data associated with this study.

The processed data required to reproduce these findings is available to download from https://github.com/sparks-baird/bayes-opt-particle-packing as v1.0.0.

The code required to reproduce these findings is hosted at https://github.com/sparks-baird/bayes-opt-particle-packing as v1.0.0.

For questions, concerns, or discussion, please open an issue using the GitHub issue tracker at https://github.com/sparks-baird/bayes-opt-particle-packing/issues.

Financial Support and Sponsorship

S.G.B. and T.D.S. acknowledge financial support by the National Science Foundation under Grant No. DMR-1651668. J.R.H. acknowledges tuition support from Northrop Grumman Innovation Systems.

Conflicts of Interest

There are no conflicts of interest to declare.

References

[1] B. Meredig, Five High-Impact Research Areas in Machine Learning for Materials Science, Chemistry of Materials 31 (2019) 9579–9581. doi:10.1021/acs.chemmater.9b04078. arXiv:1704.06439.

[2] R. Dong, Y. Dan, X. Li, J. Hu, Inverse Design of Composite Metal Oxide Optical Materials based on Deep Transfer Learning, Computational Materials Science 188 (2021) 110166. doi:10.1016/j.commatsci.2020.110166. arXiv:2008.10618.

[3] R. Espinosa, H. Ponce, J. Ortiz-Medina, A 3D orthogonal vision-based band-gap prediction using deep learning: A proof of concept, Computational Materials Science 202 (2022) 110967. doi:10.1016/j.commatsci.2021.110967.

[4] S. Ju, T. Shiga, L. Feng, Z. Hou, K. Tsuda, J. Shiomi, Designing Nanostructures for Phonon Transport via Bayesian Optimization, Phys. Rev. X 7 (2017) 021024. doi:10.1103/PhysRevX.7.021024.

[5] M. Karasuyama, H. Kasugai, T. Tamura, K. Shitara, Computational design of stable and highly ion-conductive materials using multi-objective bayesian optimization: Case studies on diffusion of oxygen and lithium, Computational Materials Science 184 (2020) 109927. doi:10.1016/j.commatsci.2020.109927.

[6] A. Palizhati, S. B. Torrisi, M. Aykol, S. K. Suram, J. S. Hummelshøj, J. H. Montoya, Agents for sequential learning using multiple-fidelity data, Sci Rep 12 (2022) 4694. doi:10.1038/s41598-022-08413-8.

[7] A. Sakurai, K. Yada, T. Simomura, S. Ju, M. Kashiwagi, H. Okada, T. Nagao, K. Tsuda, J. Shiomi, Ultranarrow-Band Wavelength-Selective Thermal Emission with Aperiodic Multilayered Metamaterials Designed by Bayesian Optimization, ACS Cent. Sci. 5 (2019) 319–326. doi:10.1021/acscentsci.8b00802.

[8] B. J. Shields, J. Stevens, J. Li, M. Parasram, F. Damani, J. I. M. Alvarado, J. M. Janey, R. P. Adams, A. G. Doyle, Bayesian reaction optimization as a tool for chemical synthesis, Nature 590 (2021) 89–96. doi:10.1038/s41586-021-03213-y.

[9] A. Talapatra, S. Boluki, T. Duong, X. Qian, E. Dougherty, R. Arróyave, Autonomous efficient experiment design for materials discovery with Bayesian model averaging, Physical Review Materials 2 (2018) 113803. doi:10.1103/PhysRevMaterials.2.113803. arXiv:1803.05460.

[10] Y. K. Wakabayashi, T. Otsuka, Y. Krockenberger, H. Sawada, Y. Taniyasu, H. Yamamoto, Bayesian optimization with experi-

mental failure for high-throughput materials growth, 2022. arXiv:2204.05452.

[11] Y. K. Wakabayashi, T. Otsuka, Y. Krockenberger, H. Sawada, Y. Taniyasu, H. Yamamoto, Machine-learning-assisted thin-film growth: Bayesian optimization in molecular beam epitaxy of SrRuO3 thin films, APL Materials 7 (2019) 101114. doi:10.1063/1.5123019. arXiv:1908.00739.

[12] G. Agarwal, H. A. Doan, L. A. Robertson, L. Zhang, R. S. Assary, Discovery of Energy Storage Molecular Materials Using Quantum Chemistry-Guided Multiobjective Bayesian Optimization, Chem. Mater. 33 (2021) 8133–8144. doi:10.1021/acs.chemmater.1c02040.

[13] H. C. Herbol, W. Hu, P. Frazier, P. Clancy, M. Poloczek, Efficient search of compositional space for hybrid organic–inorganic perovskites via Bayesian optimization, npj Comput Mater 4 (2018) 1–7. doi:10.1038/s41524-018-0106-7.

[14] R. Jalem, K. Kanamori, I. Takeuchi, M. Nakayama, H. Yamasaki, T. Saito, Bayesian-Driven First-Principles Calculations for Accelerating Exploration of Fast Ion Conductors for Rechargeable Battery Application, Sci Rep 8 (2018) 5845. doi:10.1038/s41598-018-23852-y.

[15] J. Järvi, P. Rinke, M. Todorović, Detecting stable adsorbates of (1S)-camphor on Cu(111) with Bayesian optimization, Beilstein J. Nanotechnol. 11 (2020) 1577–1589. doi:10.3762/bjnano.11.140.

[16] J. K. Pedersen, C. M. Clausen, O. A. Krysiak, B. Xiao, T. A. A. Batchelor, T. Löffler, V. A. Mints, L. Banko, M. Arenz, A. Savan, W. Schuhmann, A. Ludwig, J. Rossmeisl, Bayesian Optimization of High-Entropy Alloy Compositions for Electrocatalytic Oxygen Reduction**, Angewandte Chemie 133 (2021) 24346–24354. doi:10.1002/ange.202108116.

[17] W. Ye, X. Lei, M. Aykol, J. H. Montoya, Novel inorganic crystal structures predicted using autonomous simulation agents, Sci Data 9 (2022) 302. doi:10.1038/s41597-022-01438-8.

[18] M. Yu, S. Yang, C. Wu, N. Marom, Machine learning the Hubbard U parameter in DFT+U using Bayesian optimization, npj Comput Mater 6 (2020) 1–6. doi:10.1038/s41524-020-00446-9.

[19] Y. Zhang, D. W. Apley, W. Chen, Bayesian Optimization for Materials Design with Mixed Quantitative and Qualitative Variables, Sci Rep 10 (2020) 4924. doi:10.1038/s41598-020-60652-9.

[20] Y. Zuo, M. Qin, C. Chen, W. Ye, X. Li, J. Luo, S. P. Ong, Accelerating Materials Discovery with Bayesian Optimization and Graph Deep Learning, 2021. doi:10.48550/arXiv.2104.10242. arXiv:2104.10242.

[21] S. G. Baird, M. Liu, T. D. Sparks, High-dimensional Bayesian Optimization of Hyperparameters for an Attention-based Network to Predict Materials Property: A Case Study on CrabNet using Ax and SAASBO, arXiv:2203.12597 [cond-mat] (2022). arXiv:2203.12597.

[22] G. Cheng, X.-G. Gong, W.-J. Yin, Crystal structure prediction by combining graph network and optimization algorithm, Nat Commun 13 (2022) 1492. doi:10.1038/s41467-022-29241-4.

[23] T. Yamashita, S. Kanehira, N. Sato, H. Kino, K. Terayama, H. Sawahata, T. Sato, F. Utsuno, K. Tsuda, T. Miyake, T. Oguchi, CrySPY: A crystal structure prediction tool accelerated by machine learning, Science and Technology of Advanced Materials: Methods 1 (2021) 87–97. doi:10.1080/27660400.2021.1943171.

[24] T. Yamashita, N. Sato, H. Kino, T. Miyake, K. Tsuda, T. Oguchi, Crystal structure prediction accelerated by Bayesian optimization,

Phys. Rev. Materials 2 (2018) 013803. doi:10.1103/PhysRevMaterials.2.013803.

[25] T. Yamashita, H. Kino, K. Tsuda, T. Miyake, T. Oguchi, Hybrid algorithm of Bayesian optimization and evolutionary algorithm in crystal structure prediction, Science and Technology of Advanced Materials: Methods 2 (2022) 67–74. doi:10.1080/27660400.2022.2055987.

[26] L. Kotthoff, H. Wahab, P. Johnson, Bayesian Optimization in Materials Science: A Survey, 2021. doi:10.48550/arXiv.2108.00002. arXiv:2108.00002.

[27] R. Arróyave, D. Khatamsaz, B. Vela, R. Couperthwaite, A. Molkeri, P. Singh, D. D. Johnson, X. Qian, A. Srivastava, D. Allaire, A perspective on Bayesian methods applied to materials discovery and design, MRS Communications (2022). doi:10.1557/s43579-022-00288-0.

[28] D. E. Graff, E. I. Shakhnovich, C. W. Coley, Accelerating high-throughput virtual screening through molecular pool-based active learning, Chemical Science 12 (2021) 7866–7881. doi:10.1039/D0SC06805E.

[29] D. E. Graff, M. Aldeghi, J. A. Morrone, K. E. Jordan, E. O. Pyzer-Knapp, C. W. Coley, Self-focusing virtual screening with active design space pruning, 2022. doi:10.48550/arXiv.2205.01753. arXiv:2205.01753.

[30] K. Kandasamy, K. R. Vysyaraju, W. Neiswanger, B. Paria, C. R. Collins, J. Schneider, B. Poczos, E. P. Xing, Tuning Hyperparameters without Grad Students: Scalable and Robust Bayesian Optimisation with Dragonfly, arXiv:1903.06694 [cs, stat] (2020). arXiv:1903.06694.

[31] M. Scutari, C. Vitolo, A. Tucker, Learning Bayesian networks from big data with greedy search: Computational complexity and efficient implementation, Stat Comput 29 (2019) 1095–1108. doi:10.1007/s11222-019-09857-1.

[32] Q. Liang, A. E. Gongora, Z. Ren, A. Tiihonen, Z. Liu, S. Sun, J. R. Deneault, D. Bash, F. Mekki-Berrada, S. A. Khan, K. Hippalgaonkar, B. Maruyama, K. A. Brown, J. Fisher III, T. Buonassisi, Benchmarking the performance of Bayesian optimization across multiple experimental materials science domains, npj Comput Mater 7 (2021) 188. doi:10.1038/s41524-021-00656-9.

[33] R. J. Hickman, M. Aldeghi, F. Häse, A. Aspuru-Guzik, Bayesian optimization with known experimental and design constraints for chemistry applications, arXiv:2203.17241 [cond-mat] (2022). arXiv:2203.17241.

[34] F. Häse, M. Aldeghi, R. J. Hickman, L. M. Roch, A. Aspuru-Guzik, Gryffin: An algorithm for Bayesian optimization of categorical variables informed by expert knowledge, Applied Physics Reviews 8 (2021) 031406. doi:10.1063/5.0048164.

[35] L. Biegler, 4. Concepts of constrained optimization, in: Nonlinear Programming, 2010, pp. 63–90. doi:10.1137/1.9780898719383.ch4.

[36] V. Soloviov, On the Use of Symmetry in Optimal Design of Experiments, in: N. Balakrishnan, V. B. Melas, S. Ermakov (Eds.), Advances in Stochastic Simulation Methods, Statistics for Industry and Technology, Birkhäuser, Boston, MA, 2000, pp. 197–203. doi:10.1007/978-1-4612-1318-5_13.

[37] B. Cao, L. A. Adutwum, A. O. Oliynyk, E. J. Luber, B. C. Olsen, A. Mar, J. M. Buriak, How To Optimize Materials and Devices via Design of Experiments and Machine Learning: Demonstration Using Organic Photovoltaics, ACS Nano 12 (2018) 7434–7444. doi:10.1021/acsnano.8b04726.

[38] J. Reed, Analysis of the Accidental Explosion at Pepcon, Henderson, Nevada, May 4, 1988, Technical Report SAND-88-2902, 6610302, 1988. doi:10.2172/6610302.

[39] J. R. Hall, S. K. Kauwe, T. D. Sparks, Sequential Machine Learning Applications of Particle Packing with Large Size Variations, Integr Mater Manuf Innov 10 (2021) 559–567. doi:10.1007/s40192-021-00230-7.

[40] S. K. Kauwe, J. Graser, R. Murdock, T. D. Sparks, Can machine learning find extraordinary materials?, Computational Materials Science 174 (2020). doi:10.1016/j.commatsci.2019.109498.

[41] D. Vågberg, D. Valdez-Balderas, M. A. Moore, P. Olsson, S. Teitel, Finite-size scaling at the jamming transition: Corrections to scaling and the correlation-length critical exponent, Phys. Rev. E 83 (2011) 030303. doi:10.1103/PhysRevE.83.030303.

[42] V. Baranau, U. Tallarek, Beyond Salsburg–Wood: Glass equation of state for polydisperse hard spheres, AIP Advances 11 (2021) 035311. doi:10.1063/5.0036411.

[43] I. L. Davis, R. G. Carter, Random particle packing by reduced dimension algorithms, Journal of Applied Physics 67 (1990) 1022–1029. doi:10.1063/1.345785.

[44] M. Webb, I. L. Davis, Random particle packing with large particle size variations using reduced-dimension algorithms, Powder Technology 167 (2006) 10–19. doi:10.1016/j.powtec.2006.06.003.

[45] S. G. Baird, E. R. Homer, D. T. Fullwood, O. K. Johnson, Five degree-of-freedom property interpolation of arbitrary grain boundaries via Voronoi fundamental zone framework, Computational Materials Science 200 (2021) 110756. doi:10.1016/j.commatsci.2021.110756.

[46] E. Sevgen, E. Kim, B. Folie, V. Rivera, J. Koeller, E. Rosenthal, A. Jacobs, J. Ling, Toward Predictive Chemical Deformulation Enabled by Deep Generative Neural Networks, Ind. Eng. Chem. Res. 60 (2021) 14176–14184. doi:10.1021/acs.iecr.1c00634.

[47] A. Y.-T. Wang, S. K. Kauwe, R. J. Murdock, D. Sparks, Compositionally-Restricted Attention-Based Network for Materials Property Predictions, npj Computational Materials (2021) 33. doi:10.1038/s41524-021-00545-1.

[48] C. Chen, S. P. Ong, AtomSets as a hierarchical transfer learning framework for small and large materials datasets, npj Comput Mater 7 (2021) 173. doi:10.1038/s41524-021-00639-w.

[49] A. Dunn, Q. Wang, A. Ganose, D. Dopp, A. Jain, Benchmarking materials property prediction methods: The Matbench test set and Automatminer reference algorithm, npj Comput Mater 6 (2020) 138. doi:10.1038/s41524-020-00406-3.

[50] A. R. Falkowski, S. K. Kauwe, T. D. Sparks, Optimizing Fractional Compositions to Achieve Extraordinary Properties, Integr Mater Manuf Innov 10 (2021) 689–695. doi:10.1007/s40192-021-00242-3.

[51] R. E. A. Goodall, A. A. Lee, Predicting materials properties without crystal structure: Deep representation learning from stoichiometry, Nat Commun 11 (2020) 6280. doi:10.1038/s41467-020-19964-7.

[52] V. Gupta, K. Choudhary, F. Tavazza, C. Campbell, W.-k. Liao, A. Choudhary, A. Agrawal, Cross-property deep transfer learning framework for enhanced predictive analytics on small materials data, Nat Commun 12 (2021) 6595. doi:10.1038/s41467-021-26921-5.

[53] D. Jha, L. Ward, A. Paul, W.-k. Liao, A. Choudhary, C. Wolverton, A. Agrawal, ElemNet: Deep Learning the Chemistry of Materials From Only Elemental Composition, Sci Rep 8 (2018) 17593. doi:10.1038/s41598-018-35934-y.

[54] D. Jha, K. Choudhary, F. Tavazza, W.-k. Liao, A. Choudhary, C. Campbell, A. Agrawal,

Enhancing materials property prediction by leveraging computational and experimental data using deep transfer learning, Nat Commun 10 (2019) 5316. doi:10.1038/s41467-019-13297-w.

[55] B. Meredig, A. Agrawal, S. Kirklin, J. E. Saal, J. W. Doak, A. Thompson, K. Zhang, A. Choudhary, C. Wolverton, Combinatorial screening for new materials in unconstrained composition space with machine learning, Phys. Rev. B 89 (2014) 094104. doi:10.1103/PhysRevB.89.094104.

[56] A. Vasylenko, D. Antypov, V. Gusev, M. Gaultois, M. Dyer, M. Rosseinsky, Element Selection for Functional Materials Discovery by Integrated Machine Learning of Atomic Contributions to Properties, Preprint, In Review, 2022. doi:10.21203/rs.3.rs-1334648/v1.

[57] L. Ward, A general-purpose machine learning framework for predicting, npj Computational Materials (2016) 7.

[58] S. G. Baird, K. M. Jablonka, M. D. Alverson, H. M. Sayeed, M. F. Khan, C. Seegmiller, B. Smit, T. D. Sparks, Xtal2png: A Python package for representing crystal structure as PNG files, JOSS 7 (2022) 4528. doi:10.21105/joss.04528.

[59] A. Géron, Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, " O'Reilly Media, Inc.", 2019.

[60] K. J. DeMille, A. D. Spear, Convolutional neural networks for expediting the determination of minimum volume requirements for studies of microstructurally small cracks, Part I: Model implementation and predictions, Computational Materials Science 207 (2022) 111290. doi:10.1016/j.commatsci.2022.111290.

[61] L. Onsager, Crystal Statistics. I. A Two-Dimensional Model with an Order-Disorder Transition, Phys. Rev. 65 (1944) 117–149. doi:10.1103/PhysRev.65.117.

[62] L. McInnes, J. Healy, J. Melville, UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction, arXiv:1802.03426 [cs, stat] (2020). arXiv:1802.03426.

[63] L. Van der Maaten, G. Hinton, Visualizing data using t-SNE., Journal of machine learning research 9 (2008).

[64] S. G. Baird, T. D. Sparks, What is a Minimal Working Example for a Materials Acceleration Platform?, SSRN Journal (2022). doi:10.2139/ssrn.4164234.

[65] M. Balandat, B. Karrer, D. R. Jiang, S. Daulton, B. Letham, A. G. Wilson, E. Bakshy, BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization, arXiv:1910.06403 [cs, math, stat] (2020). arXiv:1910.06403.

[66] J. T. Wilson, F. Hutter, M. P. Deisenroth, Maximizing acquisition functions for Bayesian optimization, arXiv:1805.10196 [cs, stat] (2018). arXiv:1805.10196.

[67] A. Vehtari, A. Gelman, J. Gabry, Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC, Stat Comput 27 (2017) 1413–1432. doi:10.1007/s11222-016-9696-4.

[68] S. K. Kauwe, J. Graser, A. Vazquez, T. D. Sparks, Machine Learning Prediction of Heat Capacity for Solid Inorganics, Integrating Materials and Manufacturing Innovation 7 (2018) 43–51. doi:10.1007/s40192-018-0108-9.

[69] B. Meredig, E. Antono, C. Church, M. Hutchinson, J. Ling, S. Paradiso, B. Blaiszik, I. Foster, B. Gibbons, J. Hattrick-Simpers, A. Mehta, L. Ward, Can machine learning identify the next high-temperature superconductor? Examining extrapolation performance for materials discovery, Mol. Syst. Des. Eng. 3 (2018) 819–825. doi:10.1039/C8ME00012C.

[70] F. Ren, L. Ward, T. Williams, K. J. Laws, C. Wolverton, J. Hattrick-Simpers,

A. Mehta, Accelerated discovery of metallic glasses through iteration of machine learning and high-throughput experiments, Science Advances 4 (2018) eaaq1566. doi:10.1126/sciadv.aaq1566.

[71] S. G. Baird, M. Liu, H. M. Sayeed, T. D. Sparks, Data-driven materials discovery and synthesis using machine learning methods, in: Reference Module in Chemistry, Molecular Sciences and Chemical Engineering, Elsevier, 2022. doi:10.1016/B978-0-12-823144-9.00079-0.

[72] V. Baranau, U. Tallarek, Another resolution of the configurational entropy paradox as applied to hard spheres, J. Chem. Phys. 147 (2017) 224503. doi:10.1063/1.4999483.

[73] V. Baranau, U. Tallarek, Beyond Salsburg–Wood: Glass equation of state for polydisperse hard spheres, AIP Advances 11 (2021) 035311. doi:10.1063/5.0036411.

[74] VasiliBaranov, VasiliBaranov/packing-generation: PackingGeneration 1.0.1.28, Zenodo, 2017. doi:10.5281/zenodo.580324.

[75] S. G. Baird, T. D. Sparks, Materials Science Optimization Benchmark Dataset for Multi-fidelity Hard-sphere Packing Simulations, 2023. doi:10.26434/chemrxiv-2023-fjjk7.

[76] J. M. Beaubien, The use of simulation for training teamwork skills in health care: How low can you go?, Quality and Safety in Health Care 13 (2004) i51–i56. doi:10.1136/qshc.2004.009845.

[77] K. T. Butler, F. Oviedo, P. Canepa, Machine Learning in Materials Science, American Chemical Society, Washington, DC, USA, 2022. doi:10.1021/acsinfocus.7e5033.

[78] C. Fare, P. Fenner, E. O. Pyzer-Knapp, A Principled Method for the Creation of Synthetic Multi-fidelity Data Sets, 2022. arXiv:2208.05667.

[79] S. Gong, S. Wang, T. Xie, W. H. Chae, R. Liu, Y. Shao-Horn, J. C. Grossman, Calibrating DFT Formation Enthalpy Calculations by Multifidelity Machine Learning, JACS Au 2 (2022) 1964–1977. doi:10.1021/jacsau.2c00235.

[80] D. Khatamsaz, B. Vela, P. Singh, D. D. Johnson, D. Allaire, R. Arróyave, Multi-objective materials bayesian optimization with active learning of design constraints: Design of ductile refractory multi-principal-element alloys, Acta Materialia 236 (2022) 118133. doi:10.1016/j.actamat.2022.118133.

[81] K. Hanaoka, Comparison of Conceptually Different Multi-Objective Bayesian Optimization Methods for Material Design Problems, Materials Today Communications (2022) 103440. doi:10.1016/j.mtcomm.2022.103440.

[82] R.-R. Griffiths, A. A. Aldrick, M. Garcia-Ortegon, V. Lalchand, A. A. Lee, Achieving robustness to aleatoric uncertainty with heteroscedastic Bayesian optimisation, Mach. Learn.: Sci. Technol. 3 (2021) 015004. doi:10.1088/2632-2153/ac298c.

[83] R. Higler, R. A. M. Frijns, J. Sprakel, Diffusion Decoupling in Binary Colloidal Systems Observed with Contrast Variation Multispeckle Diffusing Wave Spectroscopy, Langmuir 35 (2019) 5793–5801. doi:10.1021/acs.langmuir.8b03745.

[84] V. Picheny, R. B. Gramacy, S. Wild, S. Le Digabel, Bayesian optimization under mixed constraints with a slack-variable augmented Lagrangian, in: Advances in Neural Information Processing Systems, volume 29, Curran Associates, Inc., 2016.

[85] P. T. Inc., Collaborative data science, https://plot.ly, 2015.

[86] Create LaTeX tables online – TablesGenerator.com, https://www.tablesgenerator.com/, 2021.

[87] S. Baird, Auto-paper, https://github.com/sparks-baird/auto-paper, 2021.