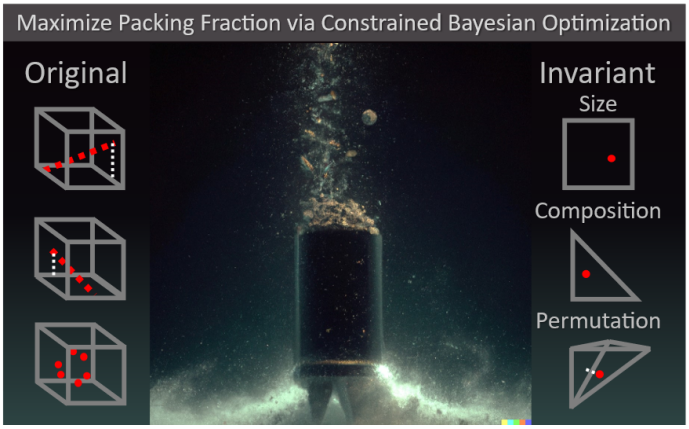
The most compact search space is not always the most efficient: A case study on maximizing solid rocket fuel packing fraction via constrained Bayesian optimization

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? This type of solution degeneracy often exists in physics-based simulations and wet-lab experiments, 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) can significantly affect Bayesian optimization search efficiency via the Ax platform. We use a physics-based particle packing simulation with seven to nine tunable parameters, depending on the search space compactness, that represent three truncated, discrete log-normal distributions of particle sizes. This physics-based simulation exhibits three qualitatively different degeneracy types: size-invariance, compositional-invariance, and permutation-invariance. We assess a total of eight search space types which range from none up to all three constraint types imposed simultaneously. We find that leaving the search space unconstrained leads to a large variance in the outcome and that on average, the most constrained search space is not always the most efficient. Likewise, the least constrained search space is not always the least efficient. While specific cases vary, in general, the removal of degeneracy tends to improve optimization efficiency. We recommend that optimization practitioners in the physical sciences carefully consider the impact of removing search space degeneracies on search efficiency before running expensive optimization campaigns.

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

1. Introduction

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-

*Corresponding author.
Email addresses: sterling.baird@utah.edu (Sterling G. Baird), sparks@eng.utah.edu (Taylor D. Sparks)

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), respectively. 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 exploitation (high-performance) and exploration (high-uncertainty) 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] among other applications such as 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].

A non-exhaustive list of popular global optimization schemes, in order of (typically) increasing efficiency, 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 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 less efficient but more computationally tractable genetic algorithms are used. Likewise, for its straightforward implementation and low computational requirements, pseudo-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.

Recently Liang et al. [27] benchmarked Bayesian optimization techniques for several materials science tasks. They raised awareness of the utility of anisotropic kernels over isotropic kernels. 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. [27], Hickman et al. [28] observed the effect of model choice and task on single- and multi-objective search efficiency, except with a constraint imposed. Hickman et al. [28] performed tests on analytical objective functions and emulators (i.e. models) trained on experimental data and demonstrate favorable performance of the `Gryffin` and `Dragonfly` optimization packages under constrained conditions.

It is well-known in the mathematical programming (optimization) community that problem formulations can introduce degeneracy or symmetry [29]. Moreover, alternate model formulations that break symmetry/degeneracy are essential for many integer programming (optimization) algorithms. 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. In a similar vein, we devote attention to avoiding degeneracy and symmetry in a materials-specific context.

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 three simultaneous constraints and instead seek to determine the effect of search space choices on efficiency. In this work, we pose the

[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).

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 [30].

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 [31], 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" [32], 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[2] BO to maximize packing fraction using low-fidelity (noisy) packing simulations which mimics common materials informatics datasets and tasks. We emphasize that the packing fractions reported in [31] 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 B 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.

2. Methods

We seek to maximize particle packing fraction, f_{volFrac}, (Section 2.1) subject to any combination (including none) of up to three invariance constraint types totaling eight 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:

$$\begin{aligned} \max_{f_{\text{volFrac}}} \quad & f_{\text{volFrac}}(\tilde{X}, S, P) \\ \text{s.t.} \quad & \text{size invariance constraint,} \\ & \text{composition invariance constraint,} \\ & \text{permutation invariance constraint(s)} \end{aligned}$$

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

where $\tilde{X}$ and S represent log-normal size distribution parameters and P represents the fractional prevalence of each of three particle types, totalling seven, eight, or nine degrees of freedom depending on the combination of constraints. Detailed information regarding the constraints is provided in Appendix A.

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 [31]. Theoretical details of the particle packing simulations are given in Davis and Carter [33] and Webb and Davis [34], for which a summary is provided in the second paragraph of the motivation section in Hall et al. [31]. A proprietary Windows executable 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 [31] in Appendix B.

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. [35] found that mapping symmetrically related sets of parameters to an irreducible representation (i.e. a fundamental zone in crystallographic terms[3]) exhibited distinct advantages related to accuracy and computational efficiency of distance calculations[4]. 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 simulation that exhibits size invariance (e.g. unitless simulations) [36, 37] (referred to as "size"), a set of parameters that is represented as a composition or formulation (i.e. $Al_2O_3 \equiv 0.4Al + 0.6O$ where $0.4 + 0.6 = 1.0$) [38–49] (referred to as "comp"), or a set of parameters that exhibits permutation invariance (e.g. $Al_2O_3 \equiv O_3Al_2$) [35, 39, 50] (referred to as "order"). A ubiquitous example exists in image processing, where machine learning algorithms often rely on data augmentation to account for rotational invariance [51], a type of invariance which is not addressed in this work. When no additional parameter constraints other than lower and upper limits are used, we refer to this as "Bounds-only".

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 [52] or t-distributed stochastic neighbor embeddings [53] in that only redundant information is lost[5] and parameters retain domain-specific, interpretable meaning.

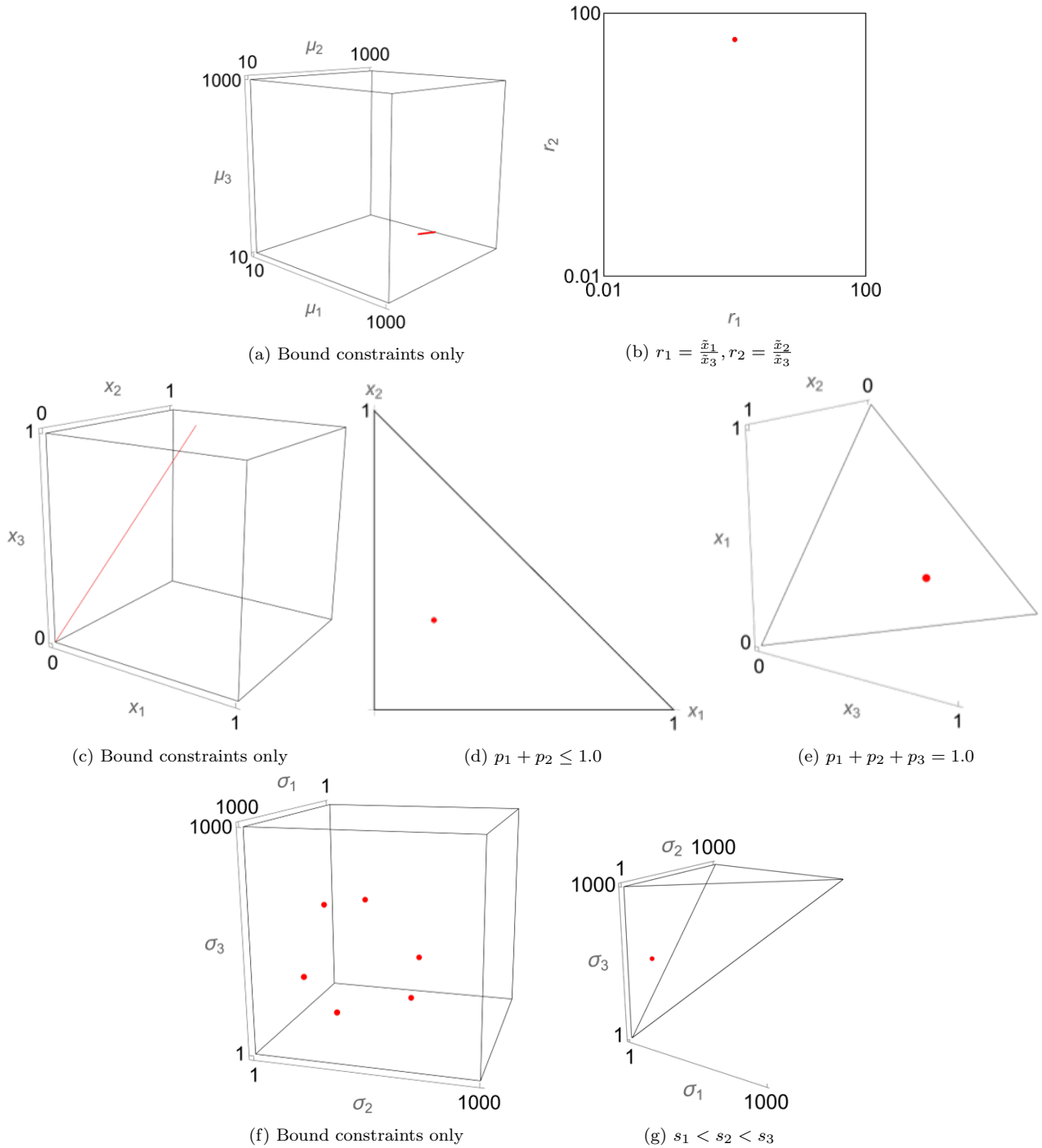
The Ax `SearchSpace` objects corresponding to each of the eight search spaces explored in this work are given in Section S8.

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

[3] 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)

[4] 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

[5] "Only redundant information is lost" assumes that the constraints imposed are consistent with the actual behavior of the objective function.



(a) Bound constraints only

(b) $r_1 = \frac{\tilde{x}_1}{\tilde{x}_3}, r_2 = \frac{\tilde{x}_2}{\tilde{x}_3}$

(c) Bound constraints only

(d) $p_1 + p_2 \leq 1.0$

(e) $p_1 + p_2 + p_3 = 1.0$

(f) Bound constraints only

(g) $s_1 < s_2 < s_3$

Figure 1: Simple visualization examples of how imposing various types of constraints affects the solution space and search dimensions. Irreducible (b,d,e,g) and reducible (a,c,f) search spaces for size (b,a), compositional (d,e,c), and permutation (g,f) invariance constraints are given. Solutions are given in red, and the search space bounds are given in black. In the top row, applying the size-invariance constraint to a line solution in a cube (b) reduces the search space to a point solution in a square (a). Note how the bounds have changed. In the middle row, applying a linear equality compositional constraint to a line solution in a cube (c) results in a point solution on a triangle embedded in a cube (e). 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 (c) results in a point solution in a triangle (d), albeit with some distortion introduced. In the bottom row, imposing two permutation-invariance constraints on a set of symmetric point solutions in a cube (f) reduces the search space to a smaller polyhedron and a single point solution (g).

Table 1: Summary of 9 non-reparameterized simulation parameters and their bounds. $\tilde{x}_i$, s_i, and p_i correspond to log-normal mean, log-normal standard deviation, and fractional prevalence (i.e. composition) for each of the three particle distributions.

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

erate search spaces are given in Figure 1.

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 [54].

In prior work [21], a high-dimensional scheme named sparse axis-aligned subspaces Bayesian optimization (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, Gaussian process expected improvement (GPEI) to limit the computational expense[6].

We refer to the GPEI implementation within the Ax platform as `GPEI`.[7] 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 assists 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. [55].

10 Sobol iterations precede 90 Bayesian optimization iterations. All Sobol iterations were required to be completed before moving on to the

[6] 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 $10 \times 90 \times 8 = 7200$ SAASBO iterations might require several months of usage on CHPC's Beehive (Windows) machine compared with a few weeks of usage using GPEI iterations.

[7] 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.

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 conditioning[9] or similar to generate a batch of candidates. 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 RayTune 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 multiple "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

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, for which the mean and standard deviation are calculated. We run repeats of the simulations using the best-predicted parameters because the calculated packing fraction is non-negligibly stochastic for the relatively low number of particles used. In other words, the objective function is noisy. 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. 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

[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. [55] for two types of conditioning: sequential greedy approaches and "fantasy" models. See also Wilson et al. [56].

[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.

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

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.

As an additional perspective, leave-one-out cross-validation (LOO-CV) is performed for each final optimization dataset and visualized via parity plots (Figure S25).

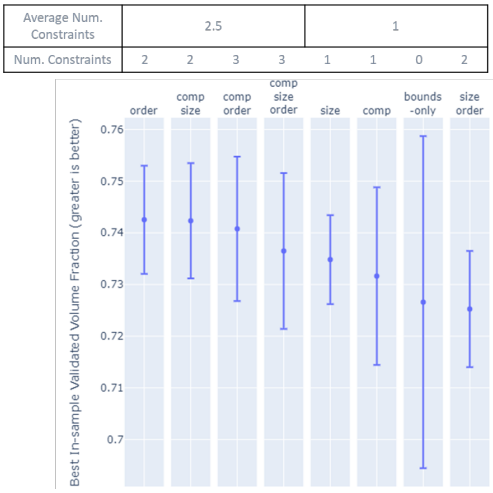
3. Results and Discussion

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

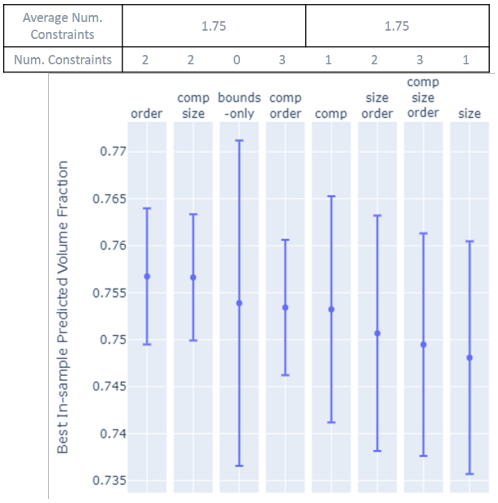
3.1. Effect of Search Space Irreducibility on Efficiency

We summarize validated, predicted, and model accuracy in Figure 2. For the validated results (Figure 2a), on average, the search spaces with the greatest number of constraints are not the most efficient; however, one is close within a small margin. Likewise, the search space with the least number of constraints is not always the least efficient, though the least constrained margin also falls within a small margin of being the least efficient. We note that when "order" and "size" appear together, there is only a single order constraint due to an incompatibility described in Appendix A.1. While results vary for specific combinations of search spaces, on average (and as expected), constraints tend to have a positive impact on optimization efficiency; the top-performing 50% of search spaces has on average 2.5 constraints per search space while the lower 50% has on average 1 constraint per search space.

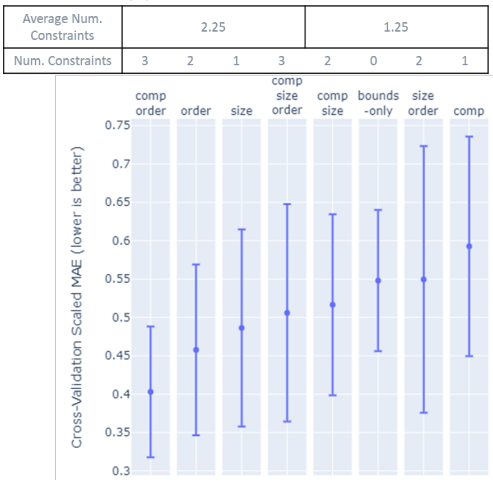
The validated (Figure 2a) and predicted (Figures 2b and 3) outcomes do not always align with each other. For example, while "order" and "comp/size" both rank highly in terms of predicted and validated outcomes, "comp/order" ranks highly in terms of validated outcomes but lower in terms of predicted outcomes. Likewise, "size/order" ranks



(a) Validation



(b) In-sample predictions



(c) Cross-validation $\frac{\text{MAE}_{\text{GPEI}}}{\text{MAE}_{\text{dummy}}}$

Figure 2: Packing fraction mean and standard deviation of (a) validated (using 50 repeat runs) and (b) best in-sample predicted results for each of the 8 search space types and 10 seeded optimization campaigns, sorted by decreasing mean packing fraction. (c) LOO-CV scaled mean absolute error (MAE), where MAE is normalized by MAE of a "dummy model" MAE (predictions are the mean of the observed values).

highly in terms of predicted outcomes but is the lowest in terms of mean validated performance. Indeed, we observe a different trend than that of Figure 2a, where the average number of constraints per search space are equal for the top-performing and lower 50% of search spaces (1.75 constraints per search space). 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.

The LOO-CV scaled MAE results (Figure 2c) follow a similar trend to validation of the best in-sample predictions (Figure 2a), where search spaces with more constraints tend to have better performance. 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 performance 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 [57–59]. 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" [60].

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 2b) and the LOO-CV scaled errors (Figure 2c), 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.

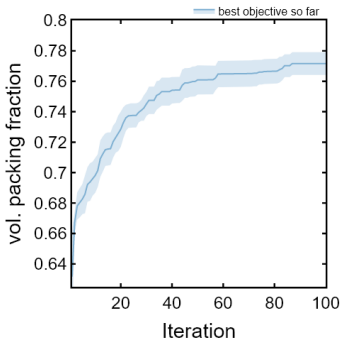
In general, the order (permutation) constraint is found more often in the top-performing 50% of search spaces than in the lower half. On the other hand, the scaling reparameterization tends to be found equally between the top-performing 50% of search spaces and the lower half[12]. This may indicate that when presented with a choice of removing symmetric images (permutation constraint) vs. collapsing a dimension via rescaling, removal of the former is preferred. We also think it is likely that use of data augmentation as described in Appendix A.3 will result in better performance than the order constraint; implementation difficulties and limitations of a data augmentation approach are discussed in Appendix A.3.

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

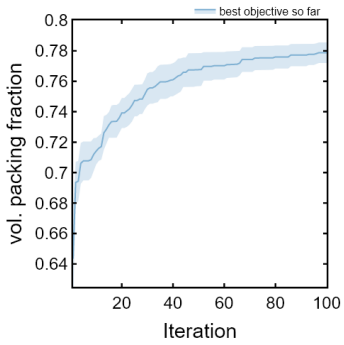
3.2. Interpretable Model Characteristics

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

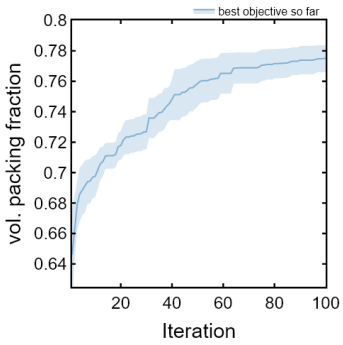
[12]The questionable performance of the scaling reparameterization may no longer be present when using advanced optimization algorithms that explicitly treat scaling along certain dimensions (i.e. non-stationary lengthscales). See for example `https://botorch.org/tutorials/bo_with_warped_gp` [61].



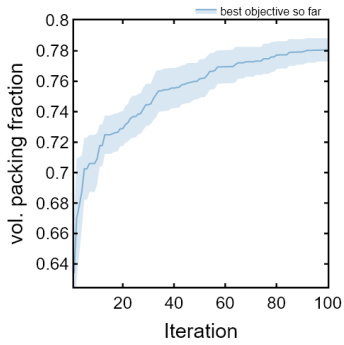
(a) Bound constraints only



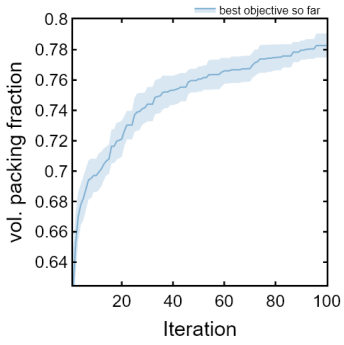
(b) Composition reparameterization



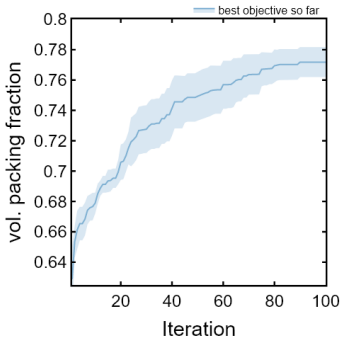
(c) Size reparameterization



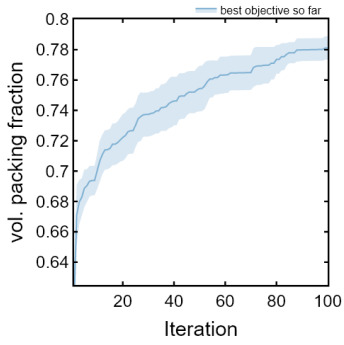
(d) Order constraint



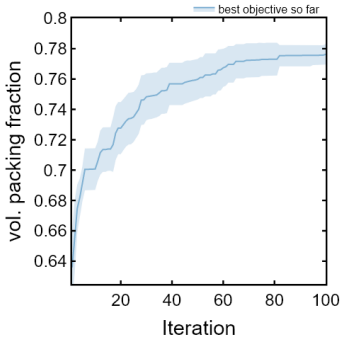
(e) Composition and size reparameterizations



(f) Composition reparameterization and order constraint



(g) Size reparameterization and order constraint



(h) Composition and size reparameterizations and order constraint

Figure 3: Best objective vs. iteration for eight search spaces using GPEI. By default, Ax uses the the best in-sample predictions rather than the noisy measured values unless a reasonable fit is not obtained.

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[13] across 10 seeded runs are given in Figure 4. Note that each search space is characterized by different sets of features, ranging from seven to 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.

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.

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 eight 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.

We notice that search spaces involving the size constraint tend to have greater local distortions to the predicted landscape relative search spaces without the size constraint. This may be due to the nonlinear transformation involved with implementing the size constraint. There are also several cases where a local distribution of datapoints exhibits a linear behavior. This could be due to a greater focus on exploitation rather than exploration as the optimization progresses.

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.

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

4. Cautions against generalization

This work reveals an example where the intuitive design spaces choices do not 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 work, we did not determine the root cause of this discrepancy in a satisfactory way; thus, we cannot generalize specific findings of this work to other cases. Even with a clear determination of the root cause, any finding should be corroborated by multiple objective functions.

In Section 5, we provide further topics of study that can help to elucidate the root causes of the phenomena described in this work.

5. Future Work

We plan to address the following topics in future work:

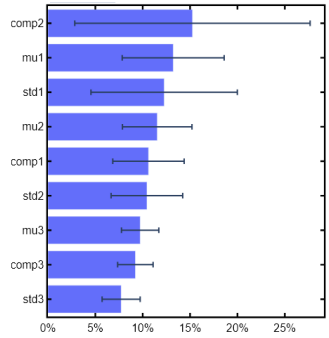
1. Are these results corroborated by alternative, open-source particle packing simulations? [62–64]

2. To what extent can multi-fidelity optimization reduce total search cost? [65–71]

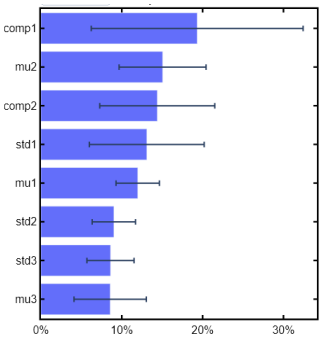
A number of questions may be interesting to explore in future work:

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

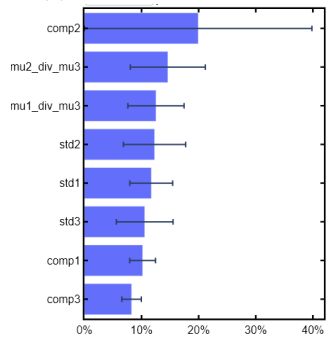
- In a reduced noise environment (i.e. better convergence through a larger number of dropped simulation particles), do the interpretable model characteristics follow more consistent trends across repeated campaigns?



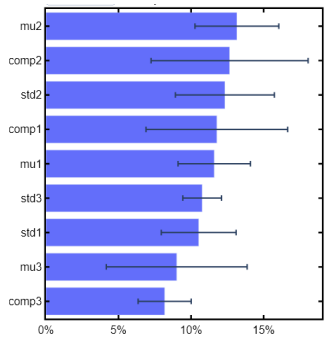
(a) Bound constraints only



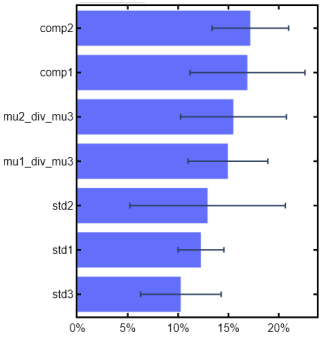
(b) Composition reparameterization



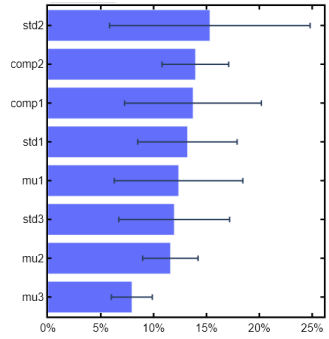
(c) Size reparameterization



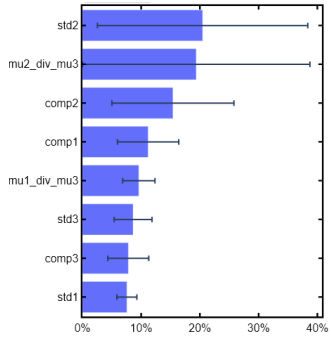
(d) Order constraint



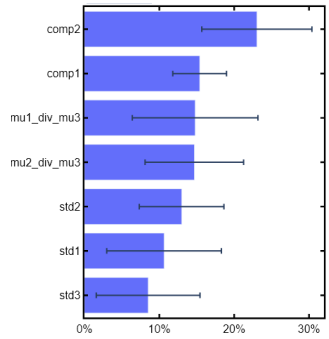
(e) Composition and size reparameterizations



(f) Composition reparameterization and order constraint



(g) Size reparameterization and order constraint



(h) Composition and size reparameterizations and order constraint

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

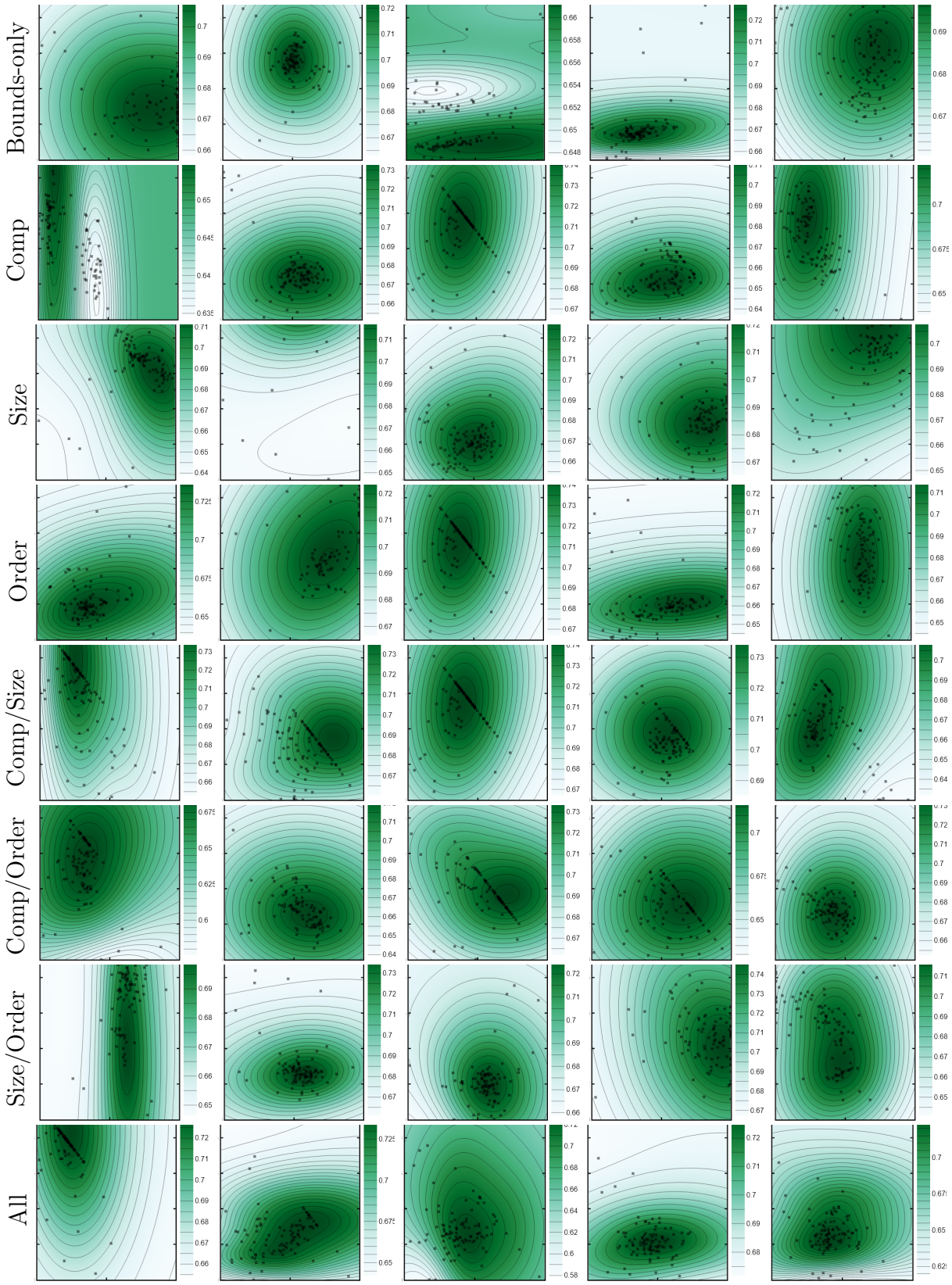


Figure 5: 2D contour plots of `comp2` (y-axis) vs. `comp1` (x-axis) for each of the eight search spaces (rows) and each of the five seeded optimization runs using GPEI.

- 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[14])
- 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 [72])?
- 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)?
- How does search space reducibility scale to multi-objective problems?
- Does replacing the Bayesian model with SAASBO perform better on average and/or change the ranking results?
- Will applying a (e.g. log) transform to the `scale` parameters when using the size reparameterization improve the optimization results?
- Could using a heteroskedastic noise assumption improve the performance of the size-reparameterized search space?
- Could regularization terms or optimization algorithms geared towards ill-posed inverse problems mitigate degeneracy issues?

6. Conclusion

We show that, on average, the most compact search space is not the most efficient for a physics-based simulation. Likewise, the least compact search space is not always the least efficient. However, the top 50% of search spaces had more constraints than the lower 50% of search spaces (10 vs. 4), indicating that on average, the presence of constraints tended to improve optimization efficiency. Best in-sample predictions and validated outcomes did not always align when compared across optimization campaigns, likely due to the high-noise objective function. Average GPEI feature importances were characterized by large standard deviations for all search spaces, making it difficult to interpret. We caution optimization practitioners to carefully assess the influence of linear and non-linear constraints or reparameterizations on their search spaces, especially when expensive physics-based simulations or wetlab experiments and noisy objectives are involved. Pairing efficient search spaces with state-of-the-art optimization algorithms has the potential to dramatically improve optimization success relative to more standard approaches.

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

Appendix A Reparameterizations and Constraints

A.1 Size Invariance

An example of a scaling or size invariant objective function is given in Eq. (1):

$$f_{\text{volFrac}}(\tilde{X}, \text{S}, \text{P}) = f_{\text{volFrac}}(\text{a}\tilde{X}, \text{S}, \text{P}), \text{a}>0 \quad (1)$$

where $\tilde{X}$, S, P, a, and $f_{\text{volFrac}}(\cdot)$ represent vector of log-normal medians (scale parameters), vector of log-normal shape parameters, vector of fractional prevalence (i.e. composition) for each particle, a positive, real-valued constant, and volume fraction function/simulation, respectively.

Reparameterizations for the log-normal mean are given in Eq. (2):

$$r_{\tilde{x},i} = \frac{\tilde{x}_i}{\tilde{x}_n} \; \forall \; \text{i} \in \{1,\text{n-1}\} \quad (2)$$

where $\tilde{x}_i$ and n represent log-normal median of the i-th particle (scale parameter) and number of particles, respectively.

In this work, $\tilde{x}_3$ is fixed to the value of `3.0`

(halfway between lower and upper bounds). Log-normal standard deviations are used as-is.

When size invariance and the order constraint are applied simultaneously, only the first two standard deviations are included in the order constraint.

A.2 Compositional Constraint

The linear equality compositional constraint is given in Eq. (3):

$$\sum_{i=1}^{n} x_i = 1 \tag{3}$$

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

However, linear equality constraints are not directly supported by most optimization packages[15] 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 [67] as Eq. (4):

$$\sum_{i=1}^{n-1} p_i \leq 1 \tag{4}$$

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

This is subject to the additional constraint that Eq. (5):

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

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

[15] 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.

A.3 Permutation Invariance

An example of permutation invariance is given in Eq. (6):

$$\begin{pmatrix} f_{\text{volFrac}}\left[\tilde{x}_1, \tilde{x}_2, \tilde{x}_3, s_1, s_2, s_3, p_1, p_2, p_3\right] = \\ f_{\text{volFrac}}\left[\tilde{x}_2, \tilde{x}_1, \tilde{x}_3, s_2, s_1, s_3, p_2, p_1, p_3\right] \end{pmatrix} \tag{6}$$

where $\tilde{x}$, s, p, and $f_{\text{volFrac}}[\cdot]$ represent log-normal median (scale parameter), 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. (7):

$$s_i \leq s_{i+1} \;\forall\; \text{i} \in \{\text{i,n-1}\} \tag{7}$$

where s_i and n represent log-normal shape parameter of the i-th particle 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 all three reparameterizations/-constraints within an `AxSearch` first in, first out scheduler framework. While possible using much lower level and custom implementations, there is an additional concern related to the simultaneous implementation of a size reparameterization, data augmentation, and bound constraints; in order for the bound constraints on the reparameterized mean and standard deviation parameters to encompass all possible values, the bounds must be extended to include extreme ratios for each reparameterized value (e.g. $[\frac{1.0}{5.0}, \frac{5.0}{1.0}]$ rather than $[\frac{1.0}{1.0}, \frac{5.0}{1.0}]$ in the case where $\tilde{x}_3 = 1.0$). Thus, it becomes difficult to deconvolve the direct effect of the data augmentation with the indirect effect on the size reparameterization bounds.

By contrast, when using an order constraint together with the size reparameterization, bound inflation can be avoided by applying the order constraint to only the first two standard deviations

rather than all three. As mentioned, 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 B Differences Between This Work and Prior Work (Particle Size Distributions)

We note that the datasets in Hall et al. [31] 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. [31] 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. [31] due to the differences in distribution sampling. For a comparison of Hall et al. [31] vs. this work, see Figure S1 and Figure S2, respectively. To avoid any ambiguity, we define our parameters as `scale` and `shape` (or `s`) 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 the `scale` parameter results in an empirical distribution whose median is equal to `scale`.

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, 14, 15

GPEI Gaussian process expected improvement 6, 10, 12–14

LOO-CV leave-one-out cross-validation 8, 9

MAE mean absolute error 8, 9

SAASBO sparse axis-aligned subspaces Bayesian optimization 6, 14

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 [73] and Ax's plotting wrapper functions. Several tables were formatted via an online formatter [74] and the auto-paper methodology [75] 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 experimental 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 net-

work 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] 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.

[28] 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.

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

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

[47] 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.

[48] 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.

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

[50] 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.

[51] 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.

[52] L. McInnes, J. Healy, J. Melville, UMAP: Uniform Manifold Approximation and Projection

for Dimension Reduction, arXiv:1802.03426 [cs, stat] (2020). arXiv:1802.03426.

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

[54] 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.

[55] 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.

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

[57] 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.

[58] 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.

[59] 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.

[60] 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.

[61] J. Snoek, K. Swersky, R. Zemel, R. Adams, Input Warping for Bayesian Optimization of Non-Stationary Functions, in: Proceedings of the 31st International Conference on Machine Learning, PMLR, 2014, pp. 1674–1682.

[62] 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.

[63] 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.

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

[65] 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.

[66] 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.

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

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

[69] 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.

[70] 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.

[71] 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.

[72] 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.

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

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

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